

Enhanced GALS Techniques for Datapath Applications

Eckhard Grass¹, Frank Winkler², Miloš Krstić¹, Alexandra Julius², Christian Stahl²,
and Maxim Piz¹

¹*IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany
{grass,krstic,piz}@ihp-microelectronics.com*

²*Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany
{fwinkler,julius,stahl}@informatik.hu-berlin.de*

Abstract. Based on a previously reported request driven technique for Globally-Asynchronous Locally-Synchronous (GALS) circuits this paper presents two significant enhancements. Firstly, the previously required local ring oscillators are avoided. Instead, an external clock with arbitrary phase for each GALS block is used. Details of the required wrapper circuitry, the proposed design flow and performance are provided. Secondly, to reduce supply noise, a novel approach applying individual clock jitter for GALS blocks is proposed. A simulation using the jitter technique shows that for a typical GALS system, the power spectrum of the supply current can be reduced by about 15 dB.

1 Introduction

New communication systems are being defined, developed and standardized at a rate that was not conceivable in the past. The rapidly changing markets put ever increasing pressure on the system designers. Reducing the time-to-market to its absolute limit is in many cases a necessity for successful commercial application.

This leads to an increasing demand for Intellectual Property (IP) blocks in the area of signal processing and communications. The use of verified and tested IP blocks can significantly reduce the design time of a system. However, achieving timing closure for a complex system is a nontrivial task. This problem is further amplified when several clock domains and clock gating are used.

The main aim of this paper is to alleviate the problem of system integration for complex designs in the area of communications and signal processing. Another issue which becomes more and more important for designers of complex systems is the reduction of Electromagnetic Interference (EMI). In particular for mixed-signal designs, the noise which is introduced into the system via power supply lines, substrate as well as by wire capacitance and inductance, can severely degrade the analog performance.

A possible solution to the problems above is the deployment of Globally-Asynchronous Locally-Synchronous (GALS) circuits. GALS can contribute to simplify and speed-up the system integration and reduce EMI as well as power dissipation. On the basis of GALS, rapid development of complex systems using existing IP blocks is facilitated [1, 2, 3].

In the past we have introduced a request driven GALS technique which is well suited for datapath applications [4, 5]. It benefits from low latency, and low power dissipation. This is achieved since no redundant clock pulses are supplied to the locally synchronous module (LSM). Each GALS block is driven with the request signal of its predecessor. For emptying internal pipeline stages each GALS block is fitted with a local ring oscillator. This system was successfully verified and tested in a GALS implementation of an IEEE 802.11a compliant baseband processor [5].

In this paper we propose two enhancements of this GALS technique. Firstly, the deployment of local ring oscillators turned out to be awkward and time consuming, both during design and test, since each ring oscillator needs individual tuning. Furthermore, the area and power penalty of the ring oscillators is deteriorating system performance. Therefore, the new versions of wrappers presented here will use an *external clock source* rather than ring oscillators.

Secondly, we propose the introduction of *jitter* for de-synchronizing the operation of different GALS blocks. This will reduce EMI and hence facilitate the integration of complex mixed-signal chips. Such introduction of blockwise jitter cannot easily be achieved with synchronous circuits.

The paper is organized as follows: In Section 2, after a brief review of existing solutions, the new generalized GALS wrapper is introduced. Section 3 describes possible solutions for locally introducing clock jitter to GALS blocks. In Section 4, we give some information on the design flow and verification. In Section 5, simulation results are reported and a comparison with previous implementations is given. Finally, conclusions are drawn in Section 6.

2 Concepts for Wrapper Implementation

2.1 Internally Clocked Wrapper

Our previous GALS designs were based on asynchronous wrappers, which had an integrated clock generator [5]. Those clock generators were implemented as ring oscillators. The advantage of this approach is that each GALS block can run on its own speed and the Electromagnetic Radiation (EMR) frequency spectrum of the complete chip is spread out due to different oscillator frequencies and phases. However, it is very tedious to implement the ring oscillators and to get their timing right. Furthermore, they may require re-tuning during operation. From a design automation point of view, it would be beneficial to use external oscillators instead. This would also reduce the power dissipation and silicon area of the wrapper.

2.2 New Externally Clocked Wrapper

Externally clocked GALS wrappers deliver the same advantages as internally clocked wrappers since it is not necessary to deliver the global clock to all GALS blocks at the same clock phase. Hence, there is neither a global clock tree nor tight skew control needed. The request driven operation is not affected, whereas the operation for emptying pipeline stages in the LSM (time-out mode) is driven by an external clock rather than a ring oscillator. During time-out mode, neighbouring GALS blocks will operate in a *mesochronous* way, i.e. at the same frequency but at different phases. The principal architecture for the new externally clocked GALS wrapper is shown in Fig. 1. Instead of the ring oscillator, an arbiter circuit is needed.

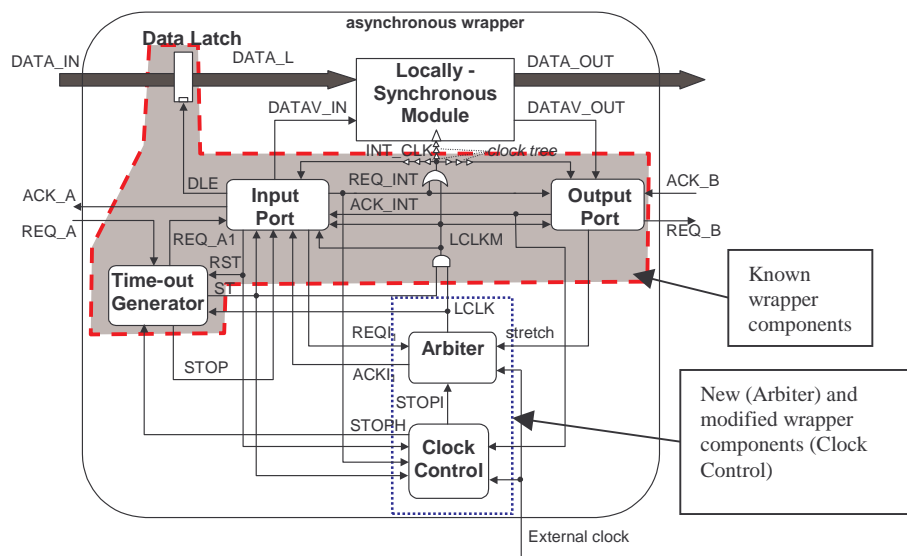


Fig. 1. Proposed new GALS wrapper with external clock source

In the following, we briefly discuss the issues related to the mesochronous clocking scheme.

Clock synchronisation: Unlike for a ring oscillator, the external clock cannot be stopped. The provision of the clock to the locally synchronous block has to be gated with an appropriate signal shown as *STOPI* in Fig. 1.

Since the external clock is not synchronized with the gating signal *STOPI*, their logic combination, potentially can result in a glitch. This glitch in turn could lead to a malfunction of the synchronous block. Hence, signal *STOPI* must change its value only during the phase when the clock signal is low. For this purpose some arbitration of the clock signal with *STOPI* is required. This is done in the new arbiter block shown in Fig. 2. The operation is explained in the following sections.

Stretching the Clock Signal: Another issue related to controlling the external clock, arises at the output of a LSM. While the GALS wrapper is outputting data vectors, halting the clock signal at level zero is required. This is needed since a downstream

GALS block may still be busy processing previous data and cannot acknowledge the current token.

Therefore, we have to arbitrate between incoming request signals and the external clock signal. If a request arrives and the clock signal line is high, no acknowledge signal will be granted. Alternatively, if request is high, the clock signal will be stretched until the request line is released.

In general, to avoid metastability during data transfer, the wrapper cannot receive any new data while the stretch signal is active. Compared to the wrapper based on a ring oscillator, we face new problems with the external clock arbitration. Halting and releasing the clock signal has to be realised in such a way that no hazards can occur.

Operation of New Wrapper with Arbiter: In order to prevent metastability, the arbiter is responsible for stretching the clock signal while there is handshake activity at the interface of the wrapper. This means, the clock signal is kept low when data is received at the input of the wrapper, during time out phase and when data is transferred at the output of the wrapper.

In the process of halting and releasing the clock, no hazards must be generated. The clock signal should only be stopped during its logic low phase. MUTEX elements M1 and M2 in Fig. 2 guarantee that the clock can only be stopped when $external_clk=0$ is detected by the arbiter. The third MUTEX M3 guarantees that the clock can only be released when $external_clk=0$. The asymmetric C-elements C1, C2 and C3 perform the glitch-free logic combination between various internal signals of the arbiter. The operation is as follows:

For example in Fig. 2, when *Stretch* goes high, signal *ste* will go to low. The condition for this state change is that $external_clock=0$.

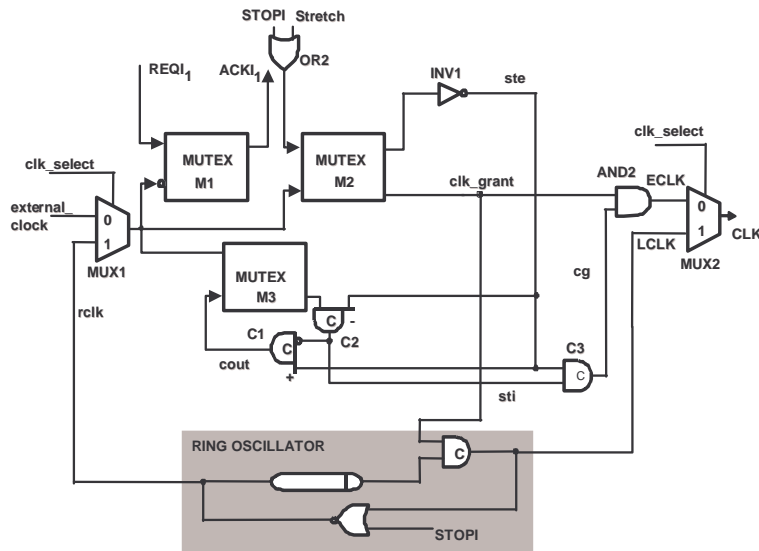


Fig. 2. Arbiter block for clock synchronisation

After that, the following scenario happens: $C2^- \rightarrow sti^- \rightarrow cg^-$ and hence, the clock $ECLK$ is disabled via AND2. When $stretch$ goes low than $ste^+ \rightarrow C1^+ \rightarrow C2^+$ (when $external_clock$ is low) $\rightarrow sti^+ \rightarrow cg^+$ and clock $ECLK$ is enabled again. In this scheme a possible race can occur between signals cg and clk_grant during arbitration. To avoid a hazard on $ECLK$, it is required that the path $external_clock \rightarrow M2 \rightarrow clk_grant$ is faster than path $stretch \rightarrow M2 \rightarrow ste \rightarrow sti \rightarrow cg$. However, this one-sided timing constraint will normally be fulfilled anyway or can easily be achieved.

2.3 Dual-Mode Wrapper with External and Internal Clock Source

A special version of the wrapper which can either be driven by a local ring oscillator or an external clock is termed ‘dual-mode wrapper’. Here, the advantages of the external clock source and the internal clock source are combined by fitting the ring-oscillator into the arbiter unit. Two multiplexers, MUX1 and MUX2, shown in Fig. 2, allow the selection of the appropriate clock source. In our implementation, the clock source can only be changed by a clk_select signal when the circuit is in idle state i.e. when all internal switching activity has ceased. The dual clock system allows during normal operation, depending on the environment, to use either the external or the internal clock source. This way, the circuit can be optimised in terms of data throughput, latency, power dissipation and EMI for different applications. If, for instance, reduction of EMI is of utmost priority, the internal oscillators can be used and the external clock can be completely switched off. If, on the other hand, reduction of power dissipation is of highest importance, the ring oscillators can be switched off and the external clock can be used. It can also support testing of GALS systems. Furthermore, this dual clock wrapper can be used for fault tolerant systems. If, for instance, an external unit detects that a quartz oscillator is broken, the local ring oscillators can be activated to keep the system function intact. The cost of the dual-mode wrapper is slightly higher power dissipation and circuit area.

3. Introduction of Clock Jitter in a GALS System

GALS systems using an external clock source do offer a lot of advantages. They are easy to design, consume low power and have small silicon area. However, compared to a system with local ring oscillators they partly re-introduce one disadvantage: EMI is increased since large parts of the circuit work exactly at the same frequency but with different phase. With local ring oscillators this is not the case. In some synchronous designs global *clock jitter* is used to reduce Electromagnetic Radiation (EMR) [6], [7]. However, this global clock jitter does not reduce peak power. In GALS systems we can introduce a blockwise clock jitter since the wrappers will take care of the data transfer between blocks. This blockwise approach will have the advantage of reducing both, the spectral energy of the clock frequency as well as peak power. In this case, we will combine clock jittering and clock phasing in one approach. Two techniques are conceivable:

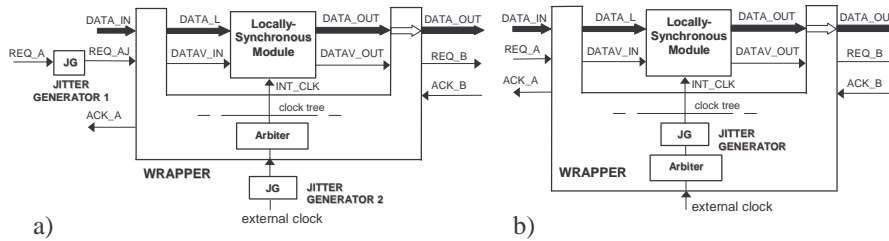


Fig. 3. Possible positions of jitter generator

a) It is possible to put one jitter generator (JG) in the signal path of the incoming REQ line of each block. Additionally, for operation in time-out mode, another JG should be inserted into the signal path of the external clock signal (see Fig. 3a).

b) It is possible to place one jitter generator (JG) in front of the clock input of the LSM (Fig. 3b). Both versions were tested and show similar results. However, the latter version requires less silicon area and has lower power dissipation.

Jitter Generator: In our GALS system, a jitter generator creates a random signal delay of a clock or request input signal. It consists of a True Random Number Generator (TRNG) or a Pseudo Noise Generator (PNG) with a sufficiently long period, and a programmable Delay Element (DE), shown in Fig. 4. The traditional jitter generation method is the Phase Modulation (PM). A voltage-controlled delay line with buffers was presented in [7]. For digital programmable delay elements, inverter chains are used. A multiplexer randomly selects a delay line output. Each signal event starts the next Pseudo Noise (PN) generation. For high clock frequencies simple Linear Feedback Shift Registers (LFSR) are preferred. In a given technology, they work up to nearly the maximum flip-flop toggle frequency. The effect of clock jitter was described in [6]. Using clock phase modulation, the spectral energy peaks were significantly reduced (up to 16 db). A higher modulation index gives lower spectral peaks, but reduces the performance slightly. Best results are achieved with random-noise phase-modulation. A high cycle-to cycle clock jitter can cause setup time violations in synchronous systems. Therefore, often triangular waveforms are used instead of PN sequences. They have lower clock-to-clock jitter, but less spectral improvements. Since the asynchronous wrappers do guarantee correct communication between GALS blocks, such systems can operate for a wide range of timing and jitter parameters.

Jitter simulation results: In order to estimate the effect of the jitter, we have created a MATLAB model of the supply current of a typical GALS system and compared it with the equivalent synchronous system. For the synchronous circuit, we assumed a triangular shape of the current waveform within one clock cycle with a 5 ns rise time and 10 ns fall time.

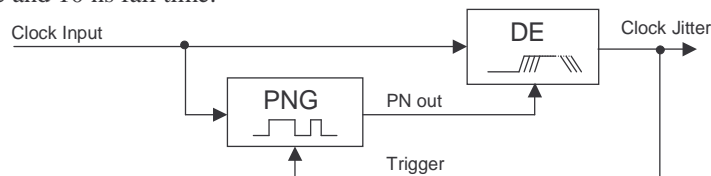


Fig. 4. Principal structure of a jitter generator

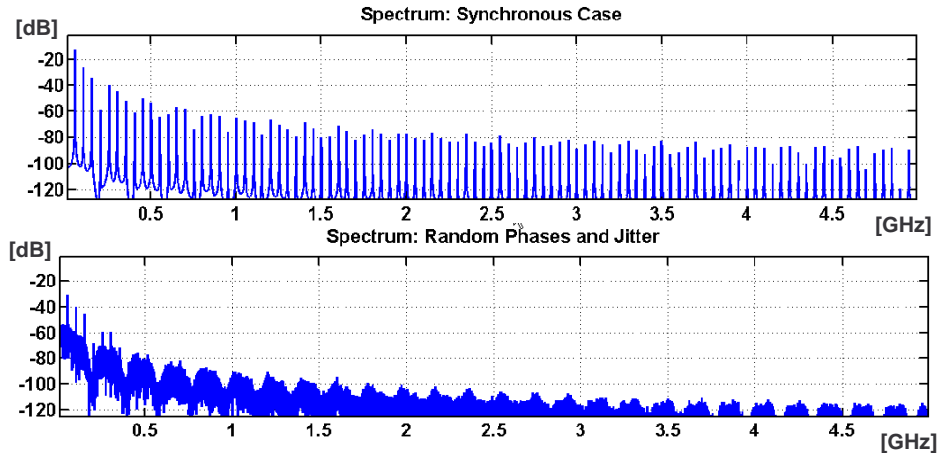


Fig. 5. Power spectrum of supply current for synchronous circuit (top) and its equivalent GALS implementation (bottom)

The peak supply current was assumed to be $I_{DD(\text{peak})} = 1$ A and the clock period was assumed to be 20 ns (50 MHz). For the GALS circuit, we assumed that the synchronous circuit is split into 10 blocks. The ‘size’ of those blocks was chosen to be such that their respective peak supply current is distributed as follows: Two blocks of 200 mA each, four blocks of 100 mA each and four blocks of 50 mA each. Rise time and fall time were not altered. Each of the 10 current waveforms was given an equally distributed random phase in the range of ± 10 ns. Afterwards, for each of the waveforms an equally distributed random jitter in the range of ± 1 ns was introduced. The ten waveforms, each representing the supply current of a GALS block, were added to compute the total supply current. Subsequently, the power spectrum of the total supply current was calculated using a Fourier Transform. The results of this Fourier Transform given in Fig. 5 show, that for a wide range of frequencies, the spectral components in the GALS system with jitter are reduced by about 15 dB when compared to the synchronous circuit. Furthermore, in the time domain, the supply current peaks are reduced to about 30% of the synchronous system.

4 Design Flow and Verification

The design flow used is a combination of the standard synchronous design flow in conjunction with the application of some asynchronous tools. For the synthesis of asynchronous controllers, we have used the 3D tool [8] for translation of burst mode specifications into Boolean equations. Asynchronous wrappers are modelled in structural VHDL code that can be mapped into gates, using synchronous synthesis tools. Using specific attributes and directives, the hazard free behaviour can be preserved.

Finally, for the layout, standard CAD tools were used. After wrapper synthesis, a formal verification was performed using the model checker LoLA.

Wrapper verification using LoLA: In order to validate the correctness of our proposed concept, and to investigate possible hazards in the system, we decided to perform a formal analysis of the asynchronous wrapper. Our modelling approach presented in [10] is a combination of event-based and level-based modelling (see [9]). For each gate type of the given wrapper, a Petri net pattern is built which describes edges and levels of all signals in the pattern. This way, a pattern preserves all information needed to detect hazards. The wrapper is a connected composition of several instances of the corresponding patterns. The question whether a hazard can occur in a gate is reduced to a model checking problem: the reachability of a particular marking in the Petri net.

To verify the wrapper's model we employed LoLA [11], an explicit model checker that features powerful state space reduction techniques. We also used the known reduction technique of *abstraction* to alleviate the problem of state space explosion (see [10]). As a result we detected a number of potential hazards, signal races, e.g. if in the arbiter *cout* is low and a falling edge at *lclk* occurs then there is a race between *clk_grant-* and *cg+* (see Fig. 2), and a deadlock which is caused by a nondeterministic choice in the input AFSM. In most cases the calculation whether a specific hazard marking is reachable takes LoLA less than one second.

Designing and analyzing the wrapper was an iterative procedure. For example, the arbiter was designed and afterwards analyzed. With the help of our analysis procedure we detected a hazard in the respective subcircuit which would strongly influence the clock signal. Due to this result, the arbiter was re-designed and the wrapper analyzed once again. Our hierarchical, pattern-based modelling approach allowed an easy and fast update of the Petri net model after design modifications.

5 Simulation Results and Comparison of Different Wrappers

The main parameters we have analyzed are silicon area, data rate, latency and power consumption. A comparison of those parameters is shown in Table 1.

Three different wrappers, one with the ring oscillator, the second using an external clock signal, and the dual mode version, were synthesized for our 0.25 μm CMOS technology. For all three circuits we simulated the data dependent power consumption of the wrappers with realistic gate-level timing and a switching activity based on 50 Msp/s data rate. The tool PrimePower was used for this purpose.

The largest silicon area is required for the circuit combining internal and external clock generation since it requires both the ring oscillator and additional logic to switch modes. The external clock generation requires by far the smallest silicon area since no ring oscillator is required. The throughput of the GALS system after synthesis was determined by simulation.

Table 1. Performance comparison of different GALS wrappers

Component Parametar	Int. wrapper (Ring oszi.)	Ext. wrapper (External clk)	Int/Ext wrapper (Dual-mode)
Total silicon area (μm^2):	79929	29879	84682
Number of gates:	1332	498	1411
Max. throughput - request mode (Msps)	119	133.9	148 / 148
Max. throughput - local mode (Msps)	86.9	79,4	96.2 / 82.6
Latency (ps)	7590	5150	6320 / 6290
Power (mW)	1.12	1.01	1.4 / 1.26

Results of this evaluation are given in Table 1. In our technology, the wrappers are operational up to about 150 Msps in request-driven mode. However, this number is due to some manual gate resizing, done after initial gate mapping. In local clock generation mode, the maximal throughput is in the range of 85 to 100 Msps. With external clocking, we achieve slightly lower speed of about 80 to 85 Msps. We have also generated latency figures for the different wrapper structures. The latency is defined as the time that data needs to pass from the last register stage of one GALS block to the first register stage of the subsequent GALS block in the data path. As we can see from Table 1, this time is in the range of 5.1 to 7.6 ns, whereby the best result is achieved with the externally clocked wrapper.

The power consumption figures presented in Table 1 are extracted from the simulation of different wrapper configurations using a realistic scenario of receiving, processing and transferring one data burst with 50 Msps datarate. In general, one wrapper consumes around 1 to 1.5 mW. This means, the wrapper does not cause any significant power overhead in our GALS system.

6 Conclusions

This paper proposes two significant enhancements to a previously reported request-driven technique for Globally-Asynchronous Locally Synchronous (GALS) circuits. To empty internal pipeline stages within locally synchronous modules our previously reported designs have used a ring oscillator for each GALS block. This technique causes much effort to design the oscillators and tune them to the correct frequency. Therefore, in this work, we have deployed an external clock instead. This clock can have an arbitrary phase for each GALS block. The attendant modifications to the GALS wrapper lead to reduced silicon area and lower power consumption when compared to the version with ring oscillators.

Secondly, to reduce power supply noise, electromagnetic radiation (EMR) and substrate noise, we propose the application of clock jitter to each individual GALS block. This reduces the spectral energy of the supply current variation as well as the peak supply current of a circuit. The blockwise application of clock jitter cannot easily be

achieved for synchronous systems since the data transfer between blocks would be problematic. To our knowledge, this is the first paper where a blockwise application of clock jitter in conjunction with GALS systems is proposed. Using a MATLAB simulation model we have shown that for a typical GALS circuit the spectral power of the supply current can be reduced by about 15 dB. In the time domain, supply current variations are reduced to about 30 % when compared to an equivalent synchronous circuit.

Further work will focus on implementation, fabrication and test of a real chip using the proposed techniques.

References

1. J. Muttersbach, *Globally-Asynchronous Locally-Synchronous Architectures for VLSI Systems*, Doctor of Technical Sciences Dissertation, ETH Zurich, Switzerland, 2001.
2. Robert Mullins, George Taylor, Peter Robinson, Simon Moore, „*Point to Point GALS Interconnect*“, IEEE Proc. ASYNC'2002, pp. 69-75, Manchester (UK), April 2002.
3. D. S. Bormann, P. Y. K. Cheoung, *Asynchronous Wrapper for Heterogeneous Systems*, In Proc. International Conf. Computer Design (ICCD), October 1997.
4. M. Krstić, E. Grass, *New GALS Technique for Datapath Architectures*, In Proc. International Workshop PATMOS, pp. 161-170, September 2003.
5. M. Krstić, E. Grass, C. Stahl, *Request-driven GALS Technique for Wireless Communication System*, Proceedings 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'2005), New York City, pp. 76-85, March, 2005.
6. Mustafa Badaroglu, Piet Wambacq, Geert Van der Plas, Stéphane Donnay, Georges Gielen, and Hugo De Man. *Digital Ground Bounce Reduction by Phase Modulation of the Clock*. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04) 2004.
7. Tian Xia Peilin Song, Keith A.Jenkins Jien-Chung Lo. *Delay Chain Based Programmable Jitter Generator*. Proceedings of the Ninth IEEE European Test Symposium (ETS'04) 2004.
8. Kenneth Yun, David Dill: *Automatic synthesis of extended burst-mode circuits: Part I and II*. IEEE Transactions on Computer-Aided Design, 18(2), pp. 101-132, Feb. 1999.
9. A. Yakovlev and A. M. Koelmans. *Petri Nets and Digital Hardware Design. LNCS: Lectures on Petri Nets II: Applications*, 1492, 1998.
10. Ch. Stahl, W. Reising, M. Krstić. *Hazard Detection in a GALS Wrapper: a Case Study*. Accepted for ACSD 2005.
11. K. Schmidt. *Lola – a low level analyser*. In Nielsen, M. and Simpson, D., editors, International Conference on Application and Theory of Petri Nets, LNCS 1825, page 465 ff. Springer-Verlag, 2000.