# A Design Tool for Modeling Asynchronous Dynamic Logic

Frank Sill
Frank Grassert
Claas Cornelius
Dirk Timmermann
Institute of Applied Microelectronics & Computer Engineering
University of Rostock, Germany

*Abstract*

For high performance designs, dynamic logic techniques have to be considered due to the promising high reachable frequencies. Such a technique is the True Single Phase Clock (TSPC) logic that allows designing circuits with standard cells and high speed potential. However, the disadvantages are a difficult clock tree design and high power consumption. Asynchronous logic has the potential to solve these problems. The used technique in this work, Asynchronous Chain-TSPC logic, assembles small asynchronous chains of dynamic logic gates into one period of the global clock. The results are shorter latency for calculations, power reduction due

to reduced overall input load and due to no need for latches as well as a simpler clock distribution network with increased clock skew tolerance and reduced clock load.

Current high level synthesis tools do not support automated synthesis and verification of asynchronous dynamic logic. Thus, this contribution presents a complete design flow for Asynchronous Chain-TSPC logic. We use the toolset DYNAMIC, which realizes a transformation of a combinational circuit into a pipelined structure and the tool AC-DYNAMIC which implements the conversion of a pipelined structure into an asynchronously clocked structure. Furthermore, AC-DYNAMIC is capable of verifying the timing behavior and undertakes optimizations. The design flow is exemplarily applied for a 32-bit-single-error-correcting circuit.

## Introduction

The exponential increase in performance and functionality has been the key to success for the microelectronics industry. However, in the past years power dissipation has reached alarming levels and has become the limiting factor for future performance enhancements. Gordon Moore, originator of the well-known Moore's law, pointed out this correlation at the worldwide largest conference on research and applications in microelectronics, the International Solid State Circuits Conference (ISSCC). He demanded extensive research efforts for reducing power dissipation in order to maintain the exponential performance increase of electronic systems in the forthcoming decade as well (Moore, 2003).

Asynchronous techniques are a promising, but also challenging, approach to cut down on power consumption. An example for such a technique is the asynchronous circuit style AC-TSPC (Asynchronous Chain True Single Phase Clock) which was developed at the University of Rostock (Grassert, 2005). AC-TSPC allows a reduction in energy dissipation of high performance processors to approximately one third. Generally, asynchronous approaches do not rely on a global clock signal to synchronize the logic elements which implement the system's functionality. Instead, AC-TSPC's logic elements begin to evaluate after receiving a start signal from its predecessors. Accordingly, every logic element creates a start signal after finishing its computation.

This type of signaling is most problematic for common circuit simulators which are specifically developed to handle synchronous designs (Sill, 2002). Thus, it is not possible with these simulators to verify the timing behavior and the evaluation results of asynchronous circuits. So, the absence of such automated design tools and design flows is one of the reasons why asynchronous concepts did not become widely accepted in spite of their advantages.

In this work the simulation and design tool AC-DYNAMIC is presented which tackles exactly this problem. With the help of complex algorithms for analysis, this tool models the timing behavior of each gate and examines the

evaluation of all appearing signal combinations. Thereby, for each signal a time frame can be defined in which the signal changes its state. This approach allows the additional consideration of process based variations of signal propagation delay which will have a tremendous impact in future technologies (Srivastava, 2005). Based on the simulation results the designer can finally determine the exact behavior of the AC-TSPC design and he can verify the signaling between all deployed gates. Furthermore, the designer receives recommendations to optimize the design in terms of performance.

The modeling is not limited to be used for AC-TSPC only so that the implemented algorithms can be used for other asynchronous design techniques as well which highly facilitates the development of these techniques because the verification is one of the main problems in chip design (Weste, 2005). Furthermore, AC-DYNAMIC is integrated in the standard design flow and, thus, does not require additional adjusted or modified design steps.

The next section describes the fundamentals of AC-TSPC's self-timed structure and the important completion detection with its problems in larger designs. The functionality of the developed simulation and design tool, which includes high level timing calculation based on the timing values of basic gates, is discussed in the following section before the results from transistor level simulations are presented and final conclusions of this work are drawn.

## Self-timed structures

This section introduces the basic principles of self-timed structures. Therefore, differential dynamic logic is briefly explained that allows to easily detect completion of evaluation. Additionally, self-timed schemes in principle and the implemented asynchronous technique AC-TSPC are presented.

### *Dynamic logic and the differential usage*

The functionality of dynamic logic is depicted in figure 1 and compared to static CMOS (SCMOS), the most widespread circuit technique. The logic function of a dynamic gate is implemented solely with a network of either N-MOS or P-MOS transistors while the logic function in SCMOS is implemented with both N-MOS and P-MOS transistors in a complementary setup. Consequently, dynamic logic is faster, because of the smaller number

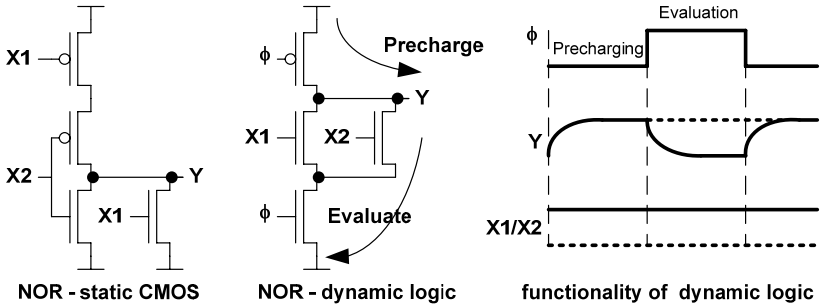


Fig. 1: NOR gate implemented in static CMOS and dynamic logic as presented together with the signaling behavior of dynamic logic

of transistors that contribute to the capacitive input load. In the given example of figure 1 only two transistors are needed compared to four in SCMOS. Furthermore, N-MOS transistors have a smaller input capacitance than P-MOS transistors when sized with equal driving strength.

Dynamic logic works in two phases, which are dictated by a clock signal Φ. The following explanation is valid for an N-logic block but can similarly be understood for P-logic as well. During precharge phase (clock low), the P-MOS transistor connects the output node Y of the dynamic gate to $V_{DD,}$ thus, charging the output capacitance at node Y to high. In the evaluation phase (clock high) the clocked N-MOS transistor is turned on while the P-MOS transistor is turned off. Depending on the input values of the logic tree (X1 and X2 in figure 1) the output node is possibly discharged to ground. Therefore, it is the evaluation phase that realizes the underlying logic function. A P-logic block works in a complementary manner with the clock signal being high during precharge and vice versa.

TSPC logic (see figure 2) uses alternating dynamic N-logic and P-logic blocks combined with N- or P-latches, respectively. Thereby, N- and P-blocks evaluate and precharge both in one clock cycle. The speed of the system is determined by the block with slowest action, i.e. the evaluation of a logic-block together with its latch (P or N) or the precharge of internal nodes (to high or low; $V_{DD}$ or ground, respectively).

Another representative of dynamic circuit techniques is the DOMINO logic that uses N-logic blocks only with subsequent static inverting logic, e.g. an inverter. This is required to allow cascading several gates sequentially which will be explained in the following. Consider the case that two simple dynamic gates (as shown in figure 1) are cascaded and use the same clock signal. The outputs are charged high during precharge so that

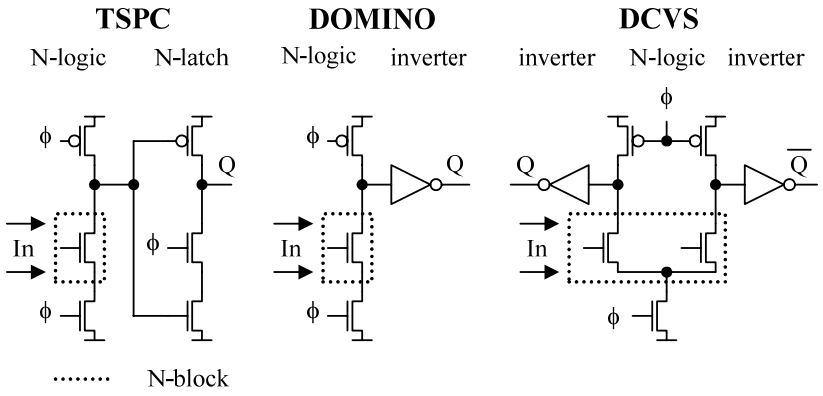


Fig. 2: Examples of three common dynamic circuit techniques: TSPC, DOMINO and dynamic DCVS

the second gate will temporarily start to discharge its output at the beginning of the evaluation phase. This results in signal degradation and eventually in malfunction. By using a static inverter, as shown in figure 2, each output of such a stage goes low during precharge. Therefore, subsequent logic trees can not connect to ground.

Another way to allow cascading dynamic gates is to use gates with different clocks so that a gate accepts the output signal of a preceding gate only if both evaluation phases overlap. In doing so, the evaluation phase of the first gate has to hold until the internal node of the second gate is fully settled. Otherwise, information is lost. Though, the shifted precharge of the first gate does not affect the settled outputs of the next one because of solely high to low transitions at the inputs.

The given examples allow deriving another circuit technique with differential outputs. The structure can be understood as two DOMINO gates with complementary logic trees and a shared clocked N-MOS transistor. An example of such an approach, the Differential Cascode Voltage Switch (DCVS) logic, is given in figure 2. The mode of operation is similar to DOMINO but the differential logic structure always evaluates two complementary output values. This allows using the differential outputs at the end of evaluation as completion detection. It should be mentioned that the same functionality can be realized if two independent domino stages are used instead.

## *Basic Self-timed Scheme*

Dynamic logic with a complementary structure and differential outputs can be arranged in an asynchronous way. There are two main approaches to set up a self-timed scheme for dynamic logic: gate outputs control the clocking of the previous or the clocking of the following gate (Krambeck, 1982). The first structure is advantageous and enables a simple implementation with minimum evaluation time. If the evaluated outputs of a gate have settled, a completion signal is generated which sets the previous gate back into the precharge phase because the inputs have successfully been processed – start precharge.

Similarly, completely precharged outputs set the previous gate into the evaluation phase (i. e. the own outputs have been precharged – start next evaluation at the predecessor; new inputs can be processed). Because the evaluation starts only with valid inputs, every gate is waiting for valid input signals during the evaluation phase. Therefore, evaluation time is only the sum of gate evaluation times with no extra delays. The computations start with the first gate and propagate the chain without being additionally delayed. Figure 3 shows such a self-timed structure with dynamic dual-rail gates. Here, a NOR gate is applied to generate the completion signal that can directly be used as the clock signal for the previous gate when a dual rail structure with DOMINO (Harris, 1997), (Yee, 2000) or DCVSL (Heller, 1984) is used.

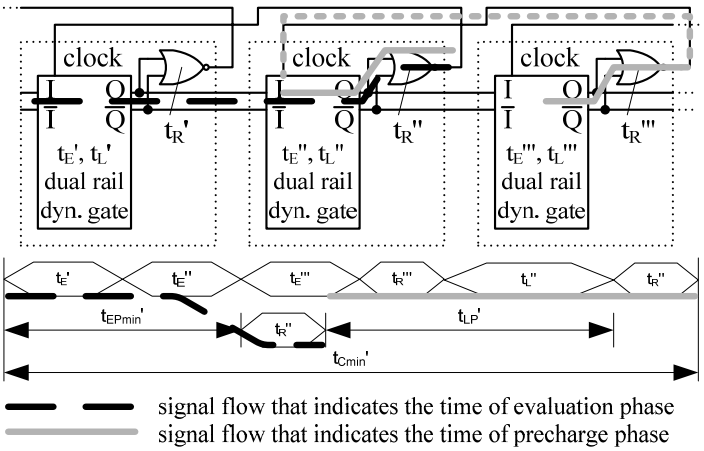


Fig. 3: Dual-rail self-timed structure and timing chart of the precharge and evaluation phase of the first logic level

## *AC-TSPC used in this work*

The particular characteristic of Asynchronous Chain True Single Phase Clock (AC-TSPC) logic is that asynchronous chains with self-timed structures are aligned by a global clock signal (see figure 4, left hand side). To integrate such chains of logic in a synchronous design, a single phase global clock with the same duration of high and low phases clocks the last gate of each chain (Grassert, 2002). Starting from there, all previous gates are connected via the self-timed scheme.

If the runtime of the chain is nearly half the clock cycle time, the evaluation will be delayed just before the last gate. However, this does not corrupt the functionality because the resulting structure of AC-TSPC still behaves like separated pipeline stages which are controlled by the global clock signal. The gates within the asynchronous chains (i. e. the pipeline stages) are classified in vertical so called chain stages where the outputs of the gates must connect only to gates in the following stage. As a gate can be connected to more than one gate with its outputs, this gate can be a part of different chains. To cope with this, the naming convention of virtual chains is established (see figure 4, right hand side). Such virtual chains represent all possible combinations of a gate within the different existing chains and are required in the simulator and design tool for the determination of timing behavior of the circuit.

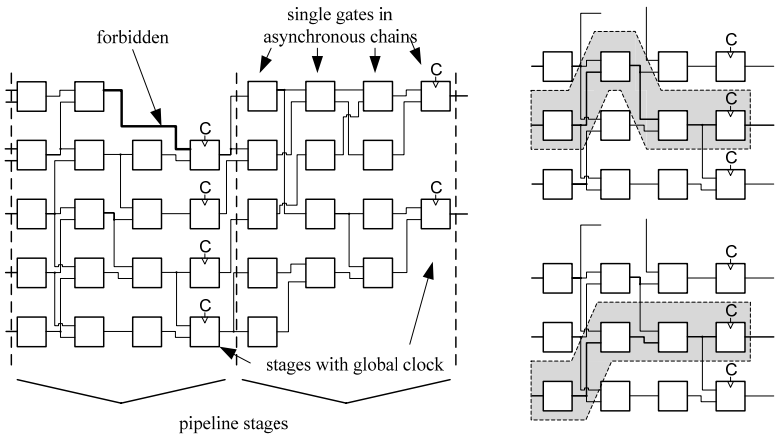


Fig. 4: Pipelined structure of AC-TSPC with globally clocked stages;
Virtual chains representing different signal paths

## Developed Design Flow

This section introduces a synthesis strategy for implementing complex designs using the self-timed logic structures which were developed and presented in "Dynamic Single Phase Logic with Self-timed Stages for Power Reduction in Pipeline Circuit Designs" (Grassert, 2001). To date, the ability to apply AC-TSPC structures to large and complex designs has been restricted by the missing automated timing analysis. The developed tool AC-DYNAMIC does not just allow such simulations but also supports further necessary steps in an automated design flow. We present a method to combine a standard Synopsys design flow with a SCMOS library and the developed library for the self-timed structures. Thereto, three additional steps after the standard synthesis are required before the actual simulation can take place. Five steps are required in total whereas the first one only needs to be performed once for all subsequent designs:

- Compiling of a new Synopsys synthesis library
- Synthesis with Synopsys design compiler
- Micro Pipeline Reorganization (MPR) done with DYNAMIC
- Setup of chain structure with AC-DYNAMIC
- Optimization performed with AC-DYNAMIC

This resulting design flow is depicted in figure 5 and the given steps are introduced in a more detailed manner in the following.

### *Development of a design library*

The development of a design library is a prerequisite before the actual design flow can be used. All necessary gates and logic functions have to be included in a new Synopsys (Synopsys, 2006) synthesis library. *HSPICE* was used for the gate level simulations before the library compiler was applied to create the library containing the results in terms of timing values, structure and logical function. Additionally, we collected the values for the minimum and maximum time of evaluation, precharge and generation of the self-timed signals which are to be used in AC-DYNAMIC for the timing analysis. The library itself has to be composed such that for every static gate in the standard library, a functionally equivalent dynamic dual-rail gate exists. Furthermore, the dynamic dual-rail gates with completion detection were designed.

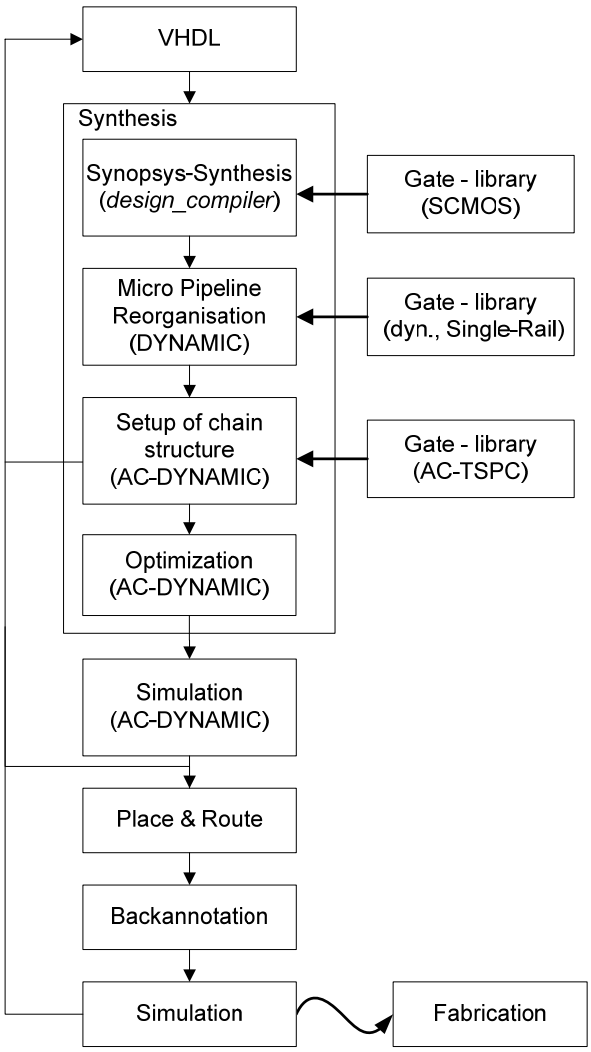


Fig. 5: Adapted standard design flow for the generation of integrated circuits with AC-TSPC logic

Automatic synthesis of dynamic pipeline stages requires a couple of additional steps. In the first place, a basic VHDL description is developed that includes the functionality in an abstract way. Then, the synthesis of the abstract design description is carried out with the design compiler and the developed library of static gates. At this point, a netlist with combinational logic only exists, i. e. without any clocking signals (see figure 6, left hand side).

This netlist has to be processed with the Micro Pipeline Reorganization (MPR) tool DYNAMIC (Wassatsch, 2002). Because every dynamic gate includes a register function, the tool can handle each logic gate as a pipeline stage. It replaces the combinational gates with equivalent single-rail dynamic gates. Then it includes simple registers in single wires to ensure correct timing behavior, because each signal has to be registered in every clock cycle. The required approach for the conversion of static gates into a pipelined structure is illustrated by a simple example in figure 6 (right hand side). The tool DYNAMIC finally generates a netlist for the fully pipelined logic. The condition for the use of the pipeline tool is the existence of a combinational netlist without any recursive connections. This means that no combinational feedback may exist, because the tool needs a well defined starting point and ending point for each signal path (refers to the virtual chains defined earlier). Where a combinational loop can be split into a forward part and a feedback part, the forward part can also be processed with the DYNAMIC tool.

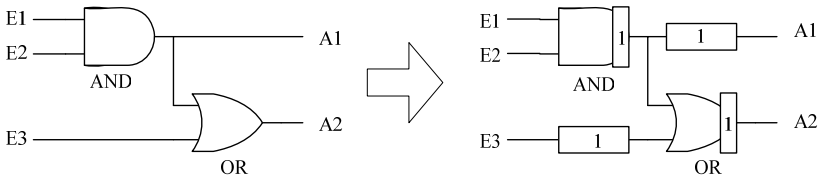


Fig. 6: Simple example for the conversion of static gates into a fully pipelined structure with equal path length

## *Setup of asynchronous chains*

The tool AC-DYNAMIC reads the netlist of the pipelined design and creates the AC-TSPC structure (see figure 4). Foremost, the dynamic single-rail gates are displaced by functional equivalent dynamic dual-rail gates with completion detection.

In the next step all input gates of the circuit are connected to the global clock signal before the last gates of all chains are connected with the global clock signal as well. Afterwards, the local completion detection signals are connected to the control inputs within the asynchronous chains. Due to irregular and complex interconnections of gates, the generation of self-timed signals used as clock signals for previous gates is not trivial. To ensure the correct order of events in the self-timed scheme, a clock signal must arrive in a defined period of time. For interconnected gates, the time periods of all participating gates must be respected. There are two main cases that have to be distinguished.

Firstly, a single completion detection signal is used as clock signal for two or more previous gates (see figure 7, left hand side). This restricts the timing behavior of the gates in the chain stage before the gate which generate the completion detection signal. The timing behavior of all previous gates, which are connected to the self-timed signal, must nearly be the same. Secondly, there are two or more gates which are connected jointly to the output signal of a single gate (see figure 7, right hand side). In this case, there is more than one completion signal that can be connected to the clock input. These signals must mostly be combined with additional logic, e.g. an OR gate. This logic has to ensure that the gate which generates the output signal does not change to precharge phase until all following gates

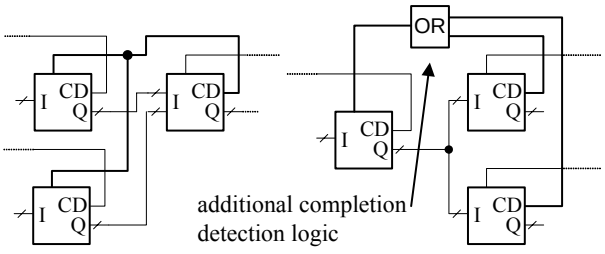


Fig. 7: Problems of completion detection due to interconnections between consecutive gates. Each block represents a dynamic dual rail gate with completion detection as shown in figure 3.

have completed their evaluation and have also generated their completion signals. The consideration of these two cases is also included in the developed tool and appropriate actions are taken, e. g. additional logic is inserted.

### *Optimization of AC-TSPC*

The optimization goals are low latency, high maximum frequency and little area. For this purpose, the tool AC-DYNAMIC can vary four parameters. These are the length of the chains, the completion detection logic and the length of the low and high phase of the clock signal. The tool calculates reference values for every configuration which is tested. At first, the maximum clock frequency for every possible chain length and for a symmetric or asymmetric clock signal is evaluated. The timing values of all gates and the given design constraints are used for this step. In the next step, the logic for completion detection is optimized. Finally, the tool searches for chains which consist of buffers only. These chains can be replaced by a single TSPC buffer gate, because the outputs of this gate have the same behavior as the outputs of a buffered chain. Thus power and area can be preserved.

### *Simulation*

After completing the optimization in terms of chain length, clock frequency and completion detection logic, the accurate timing behavior of the circuit is simulated by the tool AC-DYNAMIC. Because the exact evaluation and precharge timing values of a single gate depend on input vectors, temperature and process parameters, a minimum and a maximum evaluation or precharge time can be specified for each gate. The tool determines for each gate in a chain these minimum and maximum timing values and verifies that the self-timed signals do not violate these limits. To simulate the functional behavior, we use *HSPICE*.

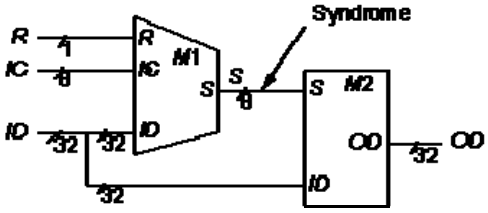


Fig. 8: Test design from the ISCAS-85 benchmark suite (C499/C1355: 32-Bit Single-Error-Correcting Circuit)

## Example

As an example we use the c1355 design from the ISCAS-85 benchmark suite which is a 32-bit single-error-correcting circuit (Hansen, 1999). The original c1355 circuit implemented in SCMOS has 41 inputs, 32 outputs, and 546 gates. The 41 inputs are combined to form an 8-bit internal bus S, which is then combined with 32 primary inputs to form the 32 primary outputs (see figure 8). The results in table 1 show that the static implementation has lower area (15% of the AC-TSPC implementation). But the maximum clock frequency of the AC-TSPC version of the circuit is almost three times higher than the one of the SCMOS version. It must be considered, that the values for the static version are measured without flip-flops. To prove the correct functionality of the simulator and to verify the achieved timing values, transistor level simulations were performed that validated the achieved results.

Tab. 1: Comparison of results for different implementations of the ISCAS-85 c1355 benchmark design

| | SCMOS | AC-TSPC |
|---|---|---|
| Gates | 546 | 2098 - (AC-TSPC)<br>184 - (logic for completion detection) |
| Max. Clock frequency | 193 MHz | 528 MHz |
| Latency | 5.15 ns | 7.57 ns |

## Conclusions and Recommendations

This contribution presents a solution for a fully supported design flow of AC-TSPC logic. This logic style combines latch-free evaluation with fast dynamic logic to reach maximum throughput in high performance applications. The presented tool generates netlists consisting of dynamic gates, calculates the exact timing behavior using a new developed strategy and compares these values with the functional limits given by the self-timed scheme. Therefore, this tool overcomes the lack of a missing synthesis flow and exact timing analysis. The results of the extensive work on timing calculation and on implementation are the very first expressive statements of the applicability of AC-TSPC. The simulation results of the tool were validated with transistor level simulations.

**Acknowledgment**

The authors acknowledge the SNUG technical committee member Kurt Baty for his assistance and helpful suggestions. Parts of this work were supported by the German Research Foundation (DFG) under grant number GRK-466 and the VIVA project.

## References

Grassert, F. and Timmermann, D., “Dynamic Single Phase Logic with Self-timed Stages for Power Reduction in Pipeline Circuit Designs”, IEEE International Symposium on Circuits and Systems (ISCAS), May 2001.

Grassert, F. and Timmermann, D., “Asynchronous Chain True Single Phase Clock Logik (AC-TSPC)”, 3. Schwerpunktkolloquium des DFG Schwerpunktprogramms Grundlagen und Verfahren verlustarmer Informationsverarbeitung (VIVA), ISBN: 3-00-008995-0, p.136 - 141, Chemnitz, March 2002.

Grassert, F., Verlustleistungsoptimierte Schaltungstechniken fuer hoechste Geschwindigkeiten, phd. thesis, Rostock, Germany, 2005.

Hansen, Yalcin, M. H., and Hayes, J. P., "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," IEEE Design and Test, vol. 16, no. 3, pp. 72-80, July-Sept. 1999.

Harris, D. and Horowitz, M. A., “Skew-Tolerant Domino Circuits”, IEEE Journal of Solid-State Circuits, Vol. 32, No. 11, Nov. 1997.

Heller, L. G., Griffin, W. R., Davis, J. W., and Thoma, N. G., “Cascode Voltage Switch Logic: A Differential CMOS Logic Family”, Proceedings of International Solid-State Circuits Conference, IEEE, 1984, pp. 16-17.

Krambeck, R. H., Lee, C. M., and Law, H.-F. S., “High-Speed Compact Circuits with CMOS”, IEEE Journal of Solid-State Circuits, IEEE, Vol. SC-17, No. 3, Jun. 1982.

Moore, G., “No Exponential Is Forever: But ’Forever’ Can Be Delayed!”, Invited Speech, ISSCC, San Francisco, 2003.

Sill, F., “Methoden zur Verlustleistungsabschaetzung“, study, Rostock, Germany, 2002.

Srivastava, A., Sylvester, D., and Blaauw, D.: “Statistical Analysis and Optimization for VLSI: Timing and Power“, 1. Edition, Springer, 2005.

Synopsys, Inc., 700 East Middlefield Road, CA 94043-4022 United States of America. Synopsys Synthesis and Simulation Tools, 2006 edition.

Wassatsch, A. Integration dynamischer Schaltungstechnik in einen Standard-CMOS-Design-Flow mit Anwendung in der digitalen Signalverarbeitung, phd. thesis, University of Rostock, Germany, 2002.

Weste, N. H. E., and Harris, D. “CMOS VLSI Design: A Circuits and Systems Perspective“, 3. Edition, Addison-Wesley, 2005.

Yee, G. and Sechen, C., “Clock-Delayed Domino for Dynamic Circuit Design”, IEEE Transactions on VLSI Systems, Vol. 8, No. 4 , Aug. 2000.