FIFO Based Routing Scheme for Clock-less System

Dr. R.K.Sharma mail2drrks@gmail.com
ECE Dept NIT Kurukshetra
Kurukshetra, India

Dr. A.K.Gupta
ECE Dept.NIT Kurukshetra
Kurukshetra, India

Mansi Jhamb mansi.jhamb@gmail.com
USIT, Guru Gobind Singh Indraprastha University,Delhi, India

Vinod Kumar Khera vinodkhera@gmail.com
Guru Tegh Bahadur Institute of Technology, Guru Gobind Singh Indraprastha University, Delhi, India

Abstract

As a result of the increasing limitations and growing complexity of semi-custom synchronous design, asynchronous circuits are gaining interest. Asynchronous Systems when combined with the local synchronous logic have provoked renewed interest over recent years, as they have the potential to combine the benefits of asynchronous and synchronous design paradigms, in this paper a new technique using FIFO in order to overcome the limitation on timing imposed by slow routing is proposed. FIFOs are often used to safely pass data from one clock domain to another asynchronous clock domain.

Keywords: Clock-less, Asynchronous, FIFO, Gray Pointers, Clock-Distribution

1. INTRODUCTION

Increasing number of gates and clock speed is the trend of semiconductor industry these days. Synchronous designs run by a single clock imply difficulties with clock distribution and excessive power consumptions. The field of synchronous- to-asynchronous circuit conversion (a subarea of the asynchronous design style) is in the same situation as the whole asynchronous design style. There are number of advantages of using asynchronous logic:

- Absence of clock skews
- Average-case performance
- The event-driven nature of asynchronous design leads to circuits with low standby power,
- A power advantage of some asynchronous design techniques is the application of level-sensitive latches.
- Reduced electromagnetic interference

Numerous approaches for automated conversion of synchronous-to asynchronous circuits have been proposed in recent years. Each of the existing approaches has its specific advantages and drawbacks, but none of them can really generate asynchronous circuits with all the so often claimed advantages, totally beating the clocked circuits. In this paper we propose adding asynchronous routing to a multiple clock domain FPGAs using Gray code pointers that are synchronized into a different clock domain before testing for "FIFO full" or "FIFO empty" conditions.

2. LITERATURE SURVEY

Most systems (embedded / FPGAs) are designed with one or more global clocks. FPGAs with multiple clock domains must provide some mechanism for synchronizing data passing between them, which will increase latency and be prone to metastability. One solution is to pipeline the routing, as used in [1]. A potential problem of this is that the number of clock cycles allocated to the routing must be determined at the routing stage and may impact upon any cycle allocation assumed.

2.1 Challenges With Asynchronous Design

In an asynchronous discipline, however, there is no global clock. Instead, the system may respond to input transitions at any time. As a result, any undesired glitch may cause the system to malfunction. Because of this sensitivity to glitches, asynchronous designs often suffer from a number of problems.

Correctness**:** Many existing asynchronous design methods do not guarantee hazard-free implementations.

Flexibility: Many design methods impose harsh restrictions on the range of behaviors that can be handled, to ensure correct operation. Typically, designs are limited to single- input change only: once an input changes, no new input change can occur until the system is stable. This restriction aids in the design of correct circuits, since techniques to eliminate hazards for single-input changes are better-known and simpler than those used for more general multiple-input changes [24]. However, the resulting circuits are of limited use.

Compatibility**:** Many asynchronous methods are incompatible with existing inter-faces, such as synchronous interfaces. Instead, they may require the use of particular protocols, such as four-phase handshaking only (discussed below). This constraint limits the practicality of asynchronous designs for existing interfaces.

Performance: Finally, in practice, many asynchronous designs have poor performance. Hazards are often eliminated by slowing down circuits by adding delays. This strategy guarantees correct operation, but abandons the potential performance benefits of asynchronous design. In the past, such difficulties have made asynchronous circuits largely unusable in practical system design However, there has been substantial progress in overcoming these obstacles in recent years.

2.2 Testing and Debugging

Testing for synchronous ASICs is made very efficient by the use of specialized scannable flip-flops, advanced tools for automated test-pattern generation, and IEEE test circuit standards such as the Joint Test Action Group (JTAG). The basic challenge associated with many asynchronous design styles is the presence of loops in the circuit that are not cut by specific latches or flip-flops. Many of these loops must be cut with additional circuitry in order to achieve sufficient observability and controllability in the circuit for test purposes and, perhaps more importantly, for automated test-pattern- generation techniques to be applicable. In addition, in comparison with synchronous design, asynchronous design can be difficult to debug. In synchronous design, when the circuit fails one can lower the clock frequency and investigate the failure. In asynchronous designs, however, there is no clock and the circuit operates at the maximum possible speed. The lack of this natural control can make debugging more challenging. Further with the use of multi-clock domain the testing of asynchronous logic offers lot of difficulties and challenges. Following section discusses few of the asynchronous architectures.

2.3 DLAP – Doubly-Latched Asynchronous Pipeline

A Doubly-Latched Asynchronous Pipeline approach (DLAP) has been first proposed by R. Kol and R. Ginosar [1] ,[2], where they claim that no single and complete methodology and a tool set have been demonstrated yet as for the design of large scale asynchronous systems. A design approach was proposed which avoids any explicit dependence on the clock . The circuit is synthesized by a commercial synchronous tool into a synchronous structure but, subsequently, it is converted into an asynchronous one. Kol and Ginosar replace each register by a pair of latches and the corresponding asynchronous controller, according to the DLAP design. The controllers are interconnected by

request and acknowledge handshake signals. Matched-delay lines are inserted on the request signal lines. While the conversion replaces the clocked registers by other types of latches, it retains (with possible minimal changes) the combinational logic that was automatically synthesized. In [1], the authors have developed algorithms for converting synchronous circuits into asynchronous ones, thus, exploiting some advantages of asynchronous circuits while retaining investments in synchronous designs and tools. They considered synchronous logic specified in VHDL, which is subsequently synthesized into netlists according to the common architecture of "register-and-cloud" pipelines [3], where "clouds" of combinational logic are separated by clocked registers. The authors also try to identify the best target asynchronous pipeline – a Doubly-Latched Asynchronous Pipeline (DLAP), which operates similarly to synchronous pipelines, and it is most suitable for synchronous-to-asynchronous conversion and, in certain important cases, it outperforms previous synchronous design. Doubly-Latched Asynchronous Pipeline is shown in Figure 1. It is designed for a single rail, 4-phase communication protocol between the stages.

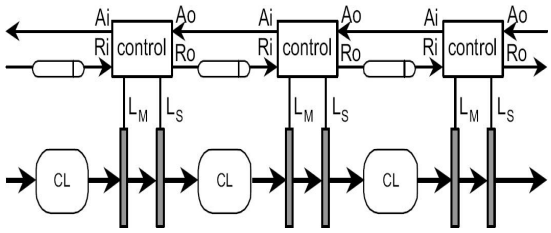

FIGURE 1: A Doubly-Latched Asynchronous Pipeline (DLAP).

By decoupling the pipeline stages, the authors of DLAP achieved the same operation as a synchronous master-slave pipeline does. If the delays of such a pipeline are balanced, DLAP operates with almost the same speed as a synchronous pipeline. Since all pipeline stages finish their computation at the same time, they can all latch the values concurrently into the master part of the registers (rewriting the bubbles), while the slave latches keep stored the values of the previous computation cycle. In the next computation cycle, all values stored in the master latches are simultaneously transferred to the slave latches. DLAP, like other asynchronous pipelines, takes advantage of variable delays. DLAP is truly decoupled – thanks to double latching, where a stage that has completed early can start processing the next data even if the following stage is still occupied. However the drawback of DLAP is area overhead of various DLAP converted circuits with the resulting area growth

2.4 De-Synchronization

De-synchronization approach to synchronous-to-asynchronous conversion proposed by J. Cortadella et al. in 2003, and has been addressed in various papers [5], [6], [7], [8], [9], [10]. Although the fundamental idea behind desynchronization is the same as DLAP [8] [1] and [11].
The de-synchronization involves three steps [8]:

1. Conversion of the flip-flop-based synchronous circuit into a latch-based Figure 2. D flip-flops are conceptually composed of master-slave latches.

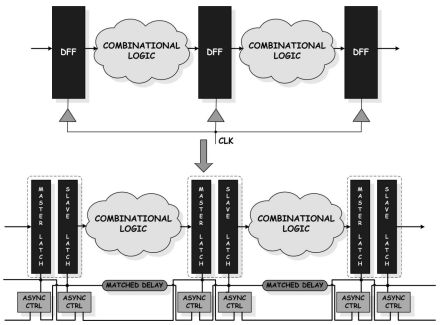

FIGURE 2 : Synchronous circuit and its de-synchronized equivalent

2. Generation of matched-delays for the combinational logic (denoted by rounded rectangles). Each matched-delay must be greater than or equal to the delay of the critical path of the corresponding combinational block. Each matched-delay serves as a completion detector for the corresponding combinational block.
3. Implementation of the local controllers.

The main problem is with the area overhead of around 28%, which is quite unprofitable [14],[15].

2.5 Theseus Logic's Approach

Theseus Logic, a company which developed a design flow for conversion of synchronous designs, specified in VHDL [22], into NCL-based asynchronous ones. The design flow and off- the-shelf simulation and synthesis components are depicted in Figure 3. The flow executes in two synthesis steps, as described in [22]. Theseus Logic's approach is different from the two previous approaches (DLAP and de-synchronization), since it starts at the register-transfer level, whereas they both start from the synthesized netlist. This requires the designer to have a proper RTL description of the design without any behavioural constructs. To synthesize and simulate an NCL circuit at the RTL using commercial tools, the tools must handle the NULL value and hysteresis behavior of threshold gates.

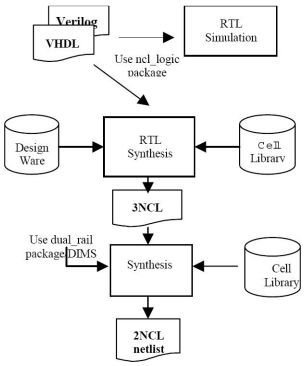

FIGURE 3 : RTL design flow for NCL

2.6. Phased Logic

Phased logic [25], [26] proposed by D. H. Linder and J. C. Harden is a delay-insensitive design methodology that seeks to restore the separation between logical and physical design by eliminating the need to distribute low skew clock signals and carefully balance propagation delays. However, unlike previous methodologies that avoid clocks, phased logic supports the cyclic, deterministic behavior of the synchronous design paradigm. However, the phased logic's approach to synchronous-to asynchronous conversion is not as straight forward and easy to understand as for example DLAP or de-synchronization. Papers [25], [26] describe phased logic using dual-rail approach named Level-Encoded two-phase Dual-Rail (LEDR) scheme but the authors mention that other dual-rail approaches are also possible to use (e.g. four-phase encoding, transition signaling, etc.).Figure 4 illustrates the key concept of LEDR encoding. The two sub signals are given subscripts v for "value" and t for "timing". A feature of the LEDR encoding that makes interfacing to phased logic easy is that the v sub signal always carries the logical value of the LEDR signal. When the value does not change but a timing transition is needed, the code word representing the current value in the opposite phase is transmitted. This creates a transition on the t sub signal.

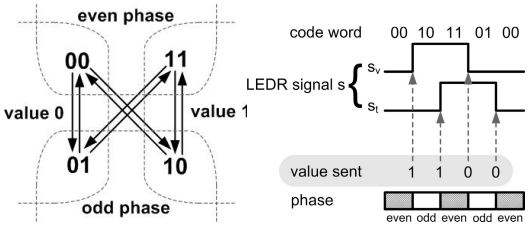

FIGURE 4: LEDR code word relationships (left) and waveforms (right)

3. ASYNCHRONOUS SYSTEMS

The term "Asynchronous" refers to circuits designed without clocks, also known as Self-Timed circuits, where the clock is replaced by handshaking signals. In a synchronous system, all blocks are assumed to have finished computation when a clock edge arrives. The blocks in Self-timed systems independently indicate completion by sending out a request and only proceed when that request has been acknowledged. Blocks only operate as needed, there is little redundant processing. Also, a self-timed system is very composable as blocks can be individually optimized and timing of one block does not affect another. We wish to exploit this feature for our FPGA architecture. Asynchronous designs require some circuit components which are rarely used for synchronous design. Ebergen [34] showed many of the circuit elements required to build delay insensitive circuits. Asynchronous FPGAs are not widely used. They are fraught with problems with hazards, critical races and metastability. Asynchronous circuits are hard to design and tools have only recently begun to reach maturity. Asynchronous buses are difficult to construct [35].It is also found that the additional completion detection circuitry required takes considerable area and power and slows the circuit down. Hence use of a Globally Asynchronous Locally Synchronous FPGA as a compromise between synchronous and asynchronous styles. We propose adding asynchronous routing to a synchronous FPGA using a new scheme.

4. PROPOSED SCHEME

Attempting to synchronize multiple changing signals from one clock domain into a new clock domain and insuring that all changing signals are synchronized to the same clock cycle in the new clock domain has been shown to be problematic [36]. FIFOs are used in designs to safely pass multi-bit data words from one clock domain to another. Data words are placed into a FIFO buffer memory array by control signals in one clock domain, and the data words are removed from another port of the same FIFO buffer memory array by control signals from a second clock domain. Conceptually, the task of designing a FIFO with these assumptions seems to be easy. The difficulty associated with doing FIFO design is related to generating the FIFO pointers and finding a reliable way to determine full and empty status on the FIFO. Early work on CLOCKLESS systems ([37] and [38]) introduced clock stretching or pausing. When data enters a synchronous system from an asynchronous environment, registers at the input are prone to metastability. To avoid this, the arrival of data is indicated by an asynchronous handshaking protocol. When data arrives, the locally generated clock is paused: in practice the rising edge of the clock is delayed. Once data has safely arrived, the clock can be released so data is latched with zero probability of metastability on the datapath. [14] used ME elements to arbitrate between the clock and incoming requests, which helped to eliminate metastability. Asynchronous wrappers, [39]were introduced ,standard components which can be placed around synchronous modules to provide the handshake signals and make them CLOCKLESS modules. The local clock generator in the proposed system is constructed from and inverter and a delay line, similar to an inverter ring oscillator as shown in figure 5. The problem with using inverters alone as a delay line is that it is difficult to accurately tune the clock period as process variations and temperature affect the delay. Hence accurate delay lines have been developed which are capable of maintaining a stable clock frequency. To make the cross domain communication, FIFO element is added to the ring as shown in figure 5. This arbitrates between the rising edge of the clock and an incoming request. Hence the clock is prevented from rising as the input registers are being enabled by the request and metastability is prevented. For each bundle of data a port controller, request and FIFO element is required. Only when all of the FIFO elements have been locked out by the clock is the rising

clock edge permitted to occur. This further avoids the problem of clock buffering as well which was there with the use of clock pausing technique using ME.

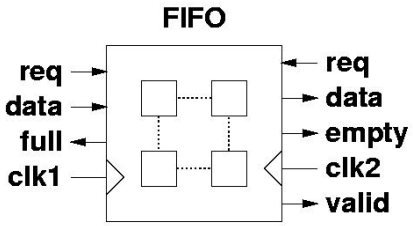

FIGURE 5: The basic architecture.

5. PROPOSED SYSTEM ARCHITECTURE

We propose converting a conventional, synchronous FPGA into a CLOCKLESS system. To do this we partition the FPGA into smaller blocks of FPGA cells. This may be done using either partial flow by XILINX, there are many methods exist in past. Within one of these blocks, the local connections are synchronous to a local clock for that block and hence the block resembles the original FPGA. However, longer communication channels between blocks become asynchronous.

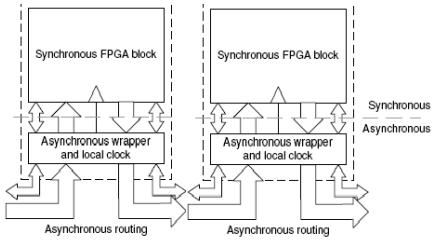

FIGURE 6: Proposed architecture

Figure 6 shows the proposed architecture in place around a block of FPGA cells. Note in particular the dividing line between the synchronous and asynchronous domains. All of the FPGA cells are in an isolated block above the line in the synchronous domain. Internally, the FPGA block could resemble any synchronous FPGA as it is hidden from the rest of the system. Below the line there is an asynchronous wrapper. Outside the asynchronous wrapper blocks are connected together using asynchronous routing. All of these blocks are explained in detail in the following section.

5.1 Asynchronous Wrapper

The asynchronous wrapper shown in figure 5 is formed of 2 components:

- Local clock generator
- Port controllers (FIFO).

The interface between the synchronous and asynchronous domains is facilitated by making the synchronous signals differential, so an event is created whenever a signal changes. Port controllers have been designed under the assumption that the synchronous block produces these differential signals and so they are a requirement of the circuit mapped to the FPGA block. Note that we require a separate clock tree for each locally synchronous block. Clearly using the global trees featured in current FPGAs would be wasteful; hence it is preferable to use a dedicated FIFO. Further, to prevent the size and delay of the FIFO from becoming too large, a limit of the size of the FPGA blocks within each wrapper is imposed.

5.2 Asynchronous Routing: the Routing Scheme

When data arrives at a register, that register must be disabled so it will not go metastable if the rising edge of the clock arrives at the same instant. Once the register has the control of the FIFO element ahead of the clock, the register can be enabled and the FIFO released. Transfer of data is facilitated by inserting FIFO into the routing. Unidirectional wires have been used exclusively

to make point-to-point connections rather than using buses. The configuration in the routing is greatly simplified and in particular if FIFO stages are used, they need only to operate in a single direction. The overall routing scheme is shown in figure 7. Instead, all long connections are made through a series of block-to-block connections. Each wire entering the FPGA block can either be routed to an input port or bypass the block completely. A connection between any two modules must pass through at least one FIFO, which helps reduce the time each module remains paused and eliminate deadlock

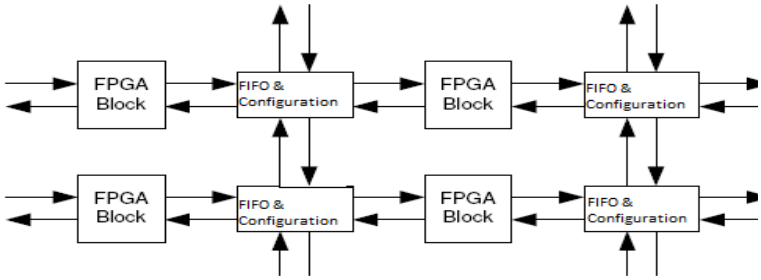

FIGURE 7: Routing Scheme

6. CONCLUSION

An extension to existing FPGA architecture has been presented in this work. However, the solution is not without its drawbacks. To address some of these problems, some alternative architecture may be required.

7. REFERENCES

[1]. Kol, R., Ginosar, R.: A Doubly-Latched Asynchronous Pipeline. In: Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD 1997), Austin, TX, USA, 12-15 October 1997, pp. 706–711 (1997)

[2]. Kol, R., Ginosar, R., Samuel, G.: Statechart Methodology for the Design, Validation, and Synthesis of Large Scale Asynchronous Systems. In: Proceedings of the 2nd International Symposium on Advanced Research in Asynchronous Circuits and Systems 1996 (ASYNC 1996), Fukushima, Japan, 18-21 March 1996, pp. 164–174 (1996)

[3]. Perry, D.: VHDL, 2nd edn. McGraw-Hill, New York (1994)

[4.] Branover, A., Kol, R., Ginosar, R.: Asynchronous Design By Conversion: Converting Synchronous Circuits into Asynchronous Ones. In: Proceedings of De sign, Automation and Test in Europe Conference and Exhibition 2004 (DATE 2004), 16-20 February 2004, vol. 2, pp. 870–875 (2004)

[5]. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: A concurrent model for de-synchronization. In: Proceedings of International Workshop on Logic Synthesis, Laguna Beach, CA, pp. 294–301 (2003)

[6]. Cortadella, J., Kondratyev, A., Lavagno, L., Lwin, K., Sotiriou, C.: From synchronous to asynchronous: An automatic approach. In: Proceedings of DATE, Paris, France, vol. 2, pp. 1368–1369 (2004)

[7]. Blunno, I., Cortadella, J., Kondratyev, A., Lavagno, L., Lwin, K., Sotiriou, C.: Handshake protocols for de-synchronization. In: Proceedings of International Symposium on Advanced Research Asynchronous Circuits Systems, Crete, Greece, pp. 149–158 (2004)

[8]. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: Desynchronization: Synthesis of Asynchronous Circuits from Synchronous Specifications. IEEE Transactions on CAD of Integrated Circuits and Systems 25(10), 1904–1921 (2006)

[9]. Andrikos, N., Lavagno, L., Pandini, D., Sotiriou, C.P.: A Fully- Automated Desynchronization Flow for Synchronous Circuits. In: Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC) 2007, San Diego, CA, USA, 4-8 June 2007, pp. 982–985 (2007)

[10]. Necchi, L., Lavagno, L., Pandini, D., Vanzago, L.: An ultralow energy asynchronous processor for Wireless Sensor Networks. In: Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC 2006), p. 78 (March 2006)

[11]. Varshavsky, V., Marakhovsky, V., Chu, T.A.: Logical timing (global synchronization of asynchronous arrays). In: Proceedings of the 1st International Symposium on Parallel Algorithm/Architecture Synthesis, Aizu- Wakamatsu, Japan, pp. 130– 138 (March 1995)

[12]. Benveniste, A., Caillaud, B., Guernic, P.L.: From synchrony to asynchrony. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 162–177. Springer, Heidelberg (1999)

[13]. Benveniste, A., Carloni, L., Caspi, P., Sangiovanni-Vincentelli, A.: Heterogeneousreactive systems modeling and correct-by-construction deployment. In : Alur, R., Lee, I. (eds.) EMSOFT 2003. LNCS, vol. 2855, pp. 35–50. Springer, Heidelberg(2003)

[14]. ˇSimlaˇst´ık, M., et al.: Clockless Implementation of LEON2 for Low- Power Applications. In: Proceedings of the 10th IEEE Workshop DDECS 2007, Krak´ow, Poland, April 11-13 (2007)

[15]. ˇSimlaˇst´ık, M., et al.: De-synchronized LEON2 Integer Unit. In: Proceedings of the6th Electronic Circuits and SystemsConference,Bratislava,Slovakia,September 6-7 (2007)

[16]. Elastix Corp., http://www.elastix-corp.com

[17]. Nanochronous Logic, http://www. nanochronous.com

[18]. Smirnov, A., Taubin, A.: Weaver Asynchronous (Self-Timed) Micropipeline Synthesis Flow,

[19]. Sutherland, I.: Micropipelines. Communications of the ACM, 720–738 (June 1989)

[20]. Fant, K.M., Brandt, S.A.: Null Convention Logic, Theseus Research, Inc. (2002)

[21]. Sobelman, G.E., Fant, K.M.: CMOS Circuit Design of Threshold Gates with Hysteresis.In: Proceedings of the IEEE International Symposium on Circuits and Systems(ISCAS 1998), Monterey, CA, USA, 31 May-3 June 1998, vol. 2, pp. 61–64 (1998)

[22]. Ligthart, M., Fant, K., Smith, R., Taubin, A., Kondratyev, A.: Asynchronous designusing commercial hdl synthesis tools. In: Proceedings of International Symposiumon Advanced Research Asynchronous Circuits Systems, Eilat, Israel, pp. 114–125 (April 2000)

[23]. Sparsø, J., Furber, S.: Principles of Asynchronous Circuit Design: A Systems Perspective.Kluwer Academic Publishers, Dordrecht (2001)

[24]. ITRS: International Technology Roadmap for Semiconductors (1999), http://www.itrs.net/1999 SIA Roadmap/Home.htm

[25]. Linder, D.: Phased Logic: A Design Methodology for Delay-Insensitive, SynchronousCircuitry. PhD thesis, Mississippi State University (1994)

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

[27]. Dean, M., Williams, T., Dill, D.: Efficient Self-Timing, with Level- Encoded 2-PhaseDual-Rail (LEDR). In: Proceedings of the 1991

[28]. McAuley, A.: Four State Asynchronous Architectures. IEEE Tra. Computers41(2), 129–142 (1992)

[29]. Reese, R., Thornton, M., Traver, C., Hemmendinger, D.: Early Evaluation for PerformanceEnhancement in Phased Logic. IEEE Transactions on Computer AidedDe sign of Integrated Circuits and Systems 24(4), 532–550 (2005)

[30]. Thornton, M., Fazel, K., Reese, R., Traver, C.: Genereralized Early Evaluation inSelf-Timed Circuits. In: Proceedings of DATE 2002, Paris, France, pp. 255–259,March 4-8 (2002)

[31]. Reese, R., Thornton, M., Traver, C.: A Coarse-grained Phased Logic CPU. In:Proceedings of the 9th International Symposium on Advanced Research in AsynchronousCircuits and Systems (ASYNC 2003), Vancouver, BC, Canada, pp. 2–13 (May 2003)

[32]. Reese, R., Thornton, M., Traver, C.: A Fine-grained Phased Logic CPU. In: IEEEComputer Society's Annual Symposium on VLSI (ISVLSI 2003), Tampa, Florida,pp. 70–79 (February 2003)

[33]. Taubin, A., Cortadella, J., Lavagno, L., Kondratyev, A., Peeters, A.: De sign automation of real-life asynchronous devices and systems. Foundations and Trends R_in Electronic Design Automation 2(1), 1–133 (2007.

[34]Ebergen, C.: A Formal Approach to Designing Delay-Insensitive Circuits. Dis-tributed Computing 5 (1988) 107.

[35]. Molina, P.: The Design of a Delay-Insensitive Bus Architecture using Ha ndshake Circuits. PhD thesis, Imperial College (1997)

[36] Clifford E. Cummings, "Synthesis and Scripting Technique s for De signing Multi-Asynchronous Clock Designs," SNUG 2001 (Synopsys Users Group Conference, San Jose, CA, 2001) User Papers, March 2001, Section MC1, 3rd paper. [37] . Chapiro, D.M.: Globally Asynchronous Locally Synchronous Systems. PhD thesis, Stanford University (1984)

[38]. P·echou·cek, M.: Anomalous response times of input sychronisers. IEEE Transac-tions on Computers C-25 (1976) 133

[39] Bormann, D.S., Cheung, P.Y.K.: Asynchronous wrapper for heterogeneous systems. In: Proceedings of the International Conference on Computer Desi gn (ICCD). (1997) 307{314.