An A-FPGA Architecture for Relative Timing Based Asynchronous Designs

Jotham Vaddaboina Manoranjan Kenneth S. Stevens
Department of Electrical and Computer Engineering
University of Utah

***Abstract*—This paper presents an asynchronous FPGA architecture that is capable of implementing relative timing based asynchronous designs. The architecture uses the Xilinx series-7 architecture as a starting point and proposes modifications that would make it asynchronous design capable while keeping it fully functional for synchronous designs. Even though the architecture requires additional components, it is observed when implemented on the 64-nm node, the area of the slice was increases marginally by 7%. The architecture leaves configurable routing structures untouched and does not compromise on performance of the synchronous architecture.**

I. Introduction

FPGAs play a predominant role in digital design world. As asynchronous design emerges into application in the ASIC world, there has been a growing interest in high performance asynchronous FPGAs (A-FPGA). The first of these FPGAs was proposed in 1994 [1]. Various other FPGAs were also presented over the years, such as the ones in [2], [3].

A significant use for an asynchronous FPGA will be to prototype and model asynchronous ASICs before manufacture. Increased manufacturing cost on sub-micron process nodes had made this critical. Clocking in the latest process nodes consumes a significant amount of power [4]. Also, accounting for skew in clock distribution has become expensive in term of design effort. A possible solution to this problem is the use of local clocking mechanisms rather than a global clock. In addition to this, creating smaller modules within designs with independent clocking can reduce the power associated with clock distribution. Asynchronous circuits inherently achieve both these improvements. Numerous studies have shown the power and performance benefits that asynchronous circuits can provide on ASICs [5], [6]. Specifically, relative timing based asynchronous designs have shown 2.4×, 2.4× and 3.2× benefits in terms of energy, area and performance respectively, when compared to synchronous versions of the designs [7]. The relative timing methodology uses explicitly defined timing constraints to guarantee the conformance of a circuit to its specification [8]. However, industrial acceptance of the technology is contingent on trust. This trust in function and performance can be bolstered with effective prototyping techniques.

Since most commercial FPGAs are built to target synchronous design, there are unique challenges associated with implementing asynchronous designs on these chips. Probably the most critical issue is to do with hazards that may occur in asynchronous implementations on synchronous FPGAs [9], [10].

Asynchronous FPGA architectures traditionally have been built for highly pipelined and high throughput applications. The ease of adding a pipeline stage and connecting modules with minimal design time overhead is an advantage of asynchronous design. In certain applications, such as cryptography, this can prove very valuable.

In this paper we propose an FPGA architecture that is capable of implementing relative timing based asynchronous designs. However from the perspective of the architecture becoming commercialized, we acknowledge the difficulty in getting consumers to buy into purely asynchronous FPGAs. Hence, this body of work has been guided by the intention of using a standard synchronous FPGA architecture and modifying it just enough to create a fully capable A-FPGA. The proposed architecture can be used either independently or in a combination of synchronous and asynchronous designs.

II. Background

Achronix Semiconductor Corp. was founded with the aim of commercializing a unique asynchronous FPGA architecture [11]. The motivation for the architecture was to develop a high throughput FPGA. This was achieved by building an architecture that was inherently composed of pipeline stages. The architecture uses novel configuration logic implementations to achieve this fine grained pipelining. The proposed benefits are: better pipelining, reduction in power consumption, resistance to process variation and ease of transferability of designs between traditional FPGAs and the A-FPGA.

The architecture utilized dual rail data encoding and a four phased protocol for handshaking. Since each bit in dual rail data encoding requires two signals for transmission, 80-90% of the area was covered by configurable routing resources and furthermore, 80-90% of the power was also consumed by the routing resources. A revision to the architecture aimed at removing some of the design overheads associated with the initial architecture was proposed [12]. The first major change was to replace the four phase protocol with a two phase protocol and the second was to use voltage scaling on certain signals. However, even though the revision does provide a better power metric, this comes at the cost of a 10% increase in area.

Another approach is to create an architecture that is capable of implementing multiple styles of asynchronous logic [13].

The design goes on to show the implementation of a delay insensitive adder based on a dual rail protocol, and also a quasi delay insensitive adder on bundled data. Both use a four-phase communication protocol. The architecture uses a Programmable Delay Element (PDE). The PDE allows the architecture to add and manipulate delays in the system easily. The architecture has primarily looked to resolve the tight connection between most asynchronous FPGA architectures and the design entry methodology. Even though the architecture is unique and capable of implementing many asynchronous styles, it is quite possible that in attempting to bridge this gap, the designers have compensated on over-all performance due to the addition of more features. Furthermore, the work does not elaborate on implementation based power/performance numbers.

A unique approach is an architecture that aims to conserve as much of the structure of conventional FPGAs as possible [14]. It seeks to keep the CLB and cluster logic, and replace or redesign the control logic in the interconnects. To avoid hazards on the communication signals, a delay insensitive model is used for the interconnect. Delay insensitive dual rail data encoding methods are used. The clock-based control system is replaced by inter-block delay insensitive signals. The architecture in the authors' opinion had the right idea to maintain conventional CLB structure to enable usage of EDA tools. However, the use of dual rail protocol again brings to the fore area and power issues associated with having multiple wires representing the same bit. Furthermore, conventional EDA tools can only be used for logic clustering into CLBs and logic elements. Once this is done, the architecture would require other tools to describe the asynchronous logic and implement it.

Globally Asynchronous locally synchronous Programmable Logic Array architecture (GAPLA) provides us with an evaluation of using asynchronous technology for routing in an FPGA [15], [16]. The architecture is based on synchronous logic blocks that are embedded in asynchronous islands. The synchronous block are contained within an asynchronous wrapper that has a local clock generator. The routing resources between these islands are asynchronous and use the bundled data protocol with 2-phase handshaking. GAPLA works under timing assumptions for its bundled data protocol.

There have been various proposed asynchronous FPGA architectures that target specific applications [17]–[20]. In these examples the application is security and cryptography. These FPGAs look to meet the need of high speed cryptography co-processors than can be reconfigured to accommodate the constantly evolving cryptography standards. These contributions primarily help ascertain the application based benefits of an asynchronous-capable FPGA.

The rest of the paper is organized as follows, section III states the motivation behind the design of the A-FPGA. Section IV briefly introduces relative timing as our design methodology. Section V describes the new architecture, and Section VI gives the implementation based results of the architecture.

III. Motivation for the Proposed Architecture

The work done as part of this paper is aimed at designing a commercially viable A-FPGA. The following summarizes the motivation for the proposed architecture:

- *Merging Synchronous and Asynchronous FPGAs*: A key component of the approach in this paper is building an FPGA architecture that is capable of implementing both synchronous and asynchronous designs. The intention is to use a common synchronous FPGA architecture and alter it just enough to allow for efficient implementation of asynchronous designs. The architectural modifications are done in a manner that does not compromise on the performance of synchronous implementations on the FPGA.
- *Bundled Data*: Bundled data single rail encoding is used in the architecture, as it uses a data path that is very similar to traditional synchronous architectures. As previously noted, delay insensitive dual rail data protocols results in a reported a 80-90% on die area dedicated to routing resources [12]. Using a dual rail protocol will require intricate architectural changes in the routing resources, whereas bundled data protocol is more suited for implementation on existing synchronous routing architectures.
- *Protocol*: There may be benefits in using two phase protocol over four phase protocols [12]. However, any general purpose A-FPGA should be capable of implementing both.
- *EDA Tools*: The capability to use existing EDA tools would be a tremendous boost to the commercial feasibility of the architecture. Asynchronous designs require communication protocols to be specified. In ASICs and in FPGAs, timing assumptions need to be made based on the handshaking protocols. This is achieved in one approach by specifying the circuit behavior as a petri-net [14]. This is probably the hardest part of building an EDA tool that is capable for supporting asynchronous circuits. The current set of academic and commercial tools look to achieve this in different ways.
  The Balsa tool achieves this through syntax-directed compilation of communicating handshaking components [21]. It uses high-level descriptions in the Balsa language. The Handshake Solutions' TiDE tool flow uses the Haste language and also other high level specifications such as MATLAB-Simulink to do this [22], [23].
  Another way to achieve the desired functionality is to derive Relative Timing (RT) constraints. Relative timing (RT) provides a methodology to model and verify circuits and protocols through timing assumptions [8].

IV. Relative Timing

Considering the various architectural guidelines discussed in the previous section, the RT based design methodology was chosen as the asynchronous circuit design technique that the proposed A-FPGA architecture would support.

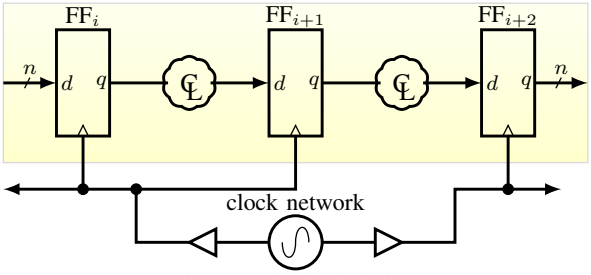

Fig. 1: Clocked design

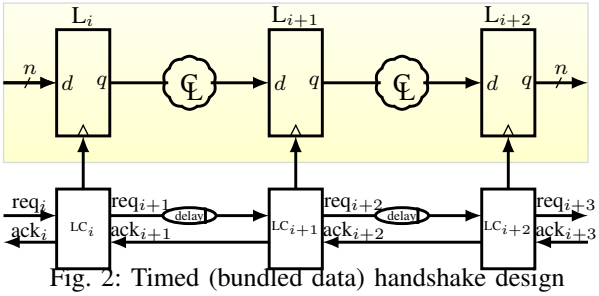

Fig. 2: Timed (bundled data) handshake design

RT methodology uses path based timing constraints to specify an order to events that may occur in an asynchronous circuit. When this order is enforced glitches in the circuit become unreachable. This guarantees correct behavior of the circuit. An instance of an RT constraint is shown below.

$$pod \;\mapsto\; poc_0 + m \;\prec poc_1; \tag{1}$$

The above RT constraint specifies the order of two events point-of-convergence poc_0 and poc_1 relative to an initial event point-of-divergent event pod. After the occurrence of the pod, the constraint causes poc_0 to always precede poc_1. On a circuit level this is realized by ensuring that the maximum delay between pod and poc_0 is less than the minimum delay between pod and poc_1. A margin is added for signal separation requirements and robustness.

RT constraints are used in conjunction with bundled data systems. Bundled data based asynchronous systems are partitioned into a control path and a data path. The data path is similar to that of a synchronous system, and consists of registers and combinational logic. In the absence of a global synchronizing clock, the role of synchronizing operation between different modules in the circuit is carried out by the control path. The control logic maintains and enforces the timing and functional relationship between various pipeline stages. This is done through handshaking and local clocking.

Fig. 1 shows an abstraction of a synchronous design with a global clock. The frequency and data path delay of the first pipeline stage is constrained by the following equation.

$$FF_i/clk{\uparrow}_j \;\mapsto\; FF_{i+1}/d + m \;\prec\; FF_{i+1}/clk{\uparrow}_{j+1} \tag{2}$$

Following a clock edge at a flip flop, the above constraint sequences arrival of new data at the flip flop corresponding to the next pipeline stage before the next respective clock edge.

Fig. 2 shows an asynchronous bundled data circuit structure analogous to the synchronous structure shown in Fig. 1. The global clock network is replaced by individual controllers for each pipeline stage that carry out a handshaking protocol between them. The following equation defines an RT constraint for the circuit.

$$req_i{\uparrow} \;\mapsto\; L_{i+1}/d + m \;\prec\; L_{i+1}/clk{\uparrow}. \tag{3}$$

Each $req_i{\uparrow}$ handshake on LC_i indicates new data presented to pin d of L_i. The delays in the circuit are sized as per the above RT constraint. Hence, after $req_i{\uparrow}$, the maximum delay to L_{i+1}/d must be smaller than the minimum delay to $L_{i+1}/clk{\uparrow}$. This would ensure that valid data is present when it is latched.

The RT based design methodology also extends to the implementation of glitchless circuits. The control logic in Fig. 2 comprises burst-mode controllers that carry out handshaking between them. Burst-mode controllers are Mealy type finite state machines. The controllers use a "request-acknowledge" signal communication between them to carry out handshaking. This handshaking can be based on signal levels or transitions. A glitch on these handshaking signals can cause a miscommunication between controllers and result in the circuit settling in an unwanted state [24]. This can be fatal to circuit operation. These glitches and hazards are primarily caused due to unwanted signal transitions within the controller caused by unexpected internal or input signal events.

FPGAs built for synchronous designs often times make asynchronous designs prone to hazards. This is attributed to the inability to exercise a high level of control over routing delays and mapping processes [9], [10].

The Automatic Relative Timing Identifier based on Signal Traces (ARTIST) tool allows the generation of relative timing constraints on signal paths from a formal verification engine [25]. The tool creates RT constraints for a circuit that orders signal transitions making hazards unreachable. The relative timing methodology has been successfully applied to create functionally correct bundled data based asynchronous systems [26], and maps well to FPGA designs.

V. Proposed Architecture

In this section we propose certain changes to the Xilinx slice architecture that enable the implemention of RT based asynchronous designs on an FPGA. Fig. 3 and 4 show an abstraction of the Xilinx 7-Series FPGA architecture [27]. Each configuration logic block has two slices that are connected by a routing matrix. Each slice is comprised of four 6-input LUTs that can also be programmed as two 5-input LUTs. Fig. 3 shows the building blocks of a slice. The LUTs have various other components such as carry-chain logic and DFFs associated with it. This building block is used to construct a slice as shown in Fig. 4. For the purposes of this paper the architecture has been simplified, and the Xilinx 7-Series SLICEL was used to guide the design.

Fig. 3 and 4 also show the proposed changes to the traditional synchronous architecture. These changes are marked by

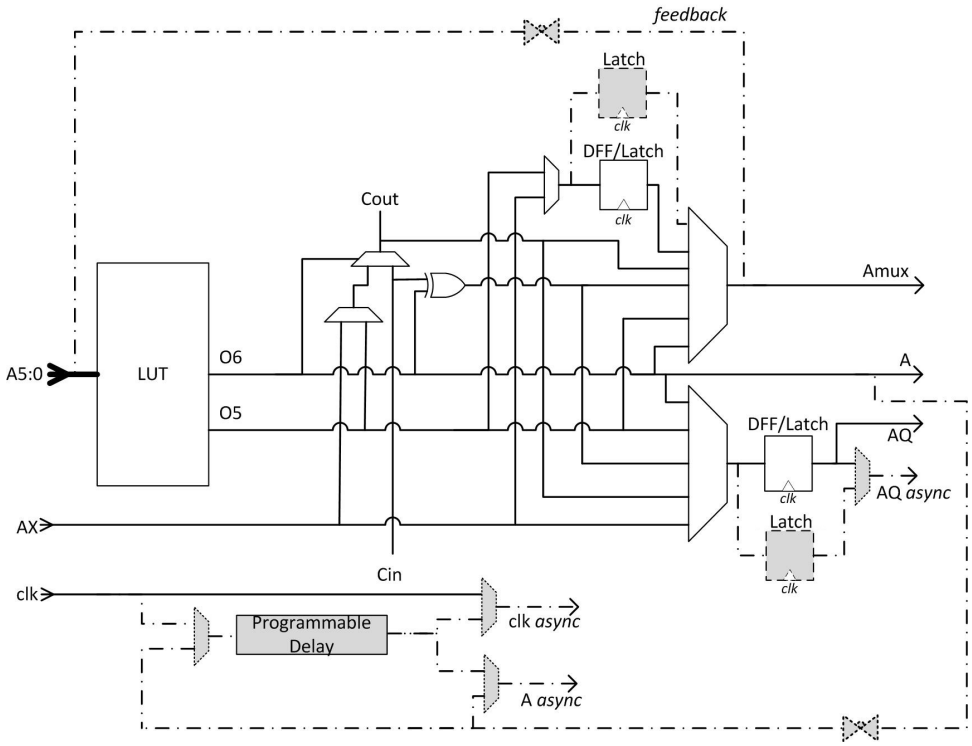

Fig. 3: Structure showing LUT and Logic Associated with Each LUT in a Slice. Additional logic to make is an A-FPGA has been shown with shaded elements and dashed-wires

shaded logic elements and dashed wires. It is important to note that it is possible to implement relative timed asynchronous designs on current FPGAS. The architectural changes being proposed here are aimed at creating a more favorable design fabric for RT asynchronous designs. The following changes are proposed:

Programmable Delay Structure:

It is easy to see how having a programmable delay element (PDE) can tremendously simplify implementing RT based asynchronous circuits on FPGAs. Since RT defines ordering of events in a circuit, the easiest way to achieve this would be to have control on the delays of various signals, particularly the clock and handshake control logic.

RT based asynchronous designs use a local clocking mechanism as shown in Fig. 2. Since controllers drive the local-clock for each pipeline stage, clock distribution with minimum skew is a challenge. There are various ways to do this on current FPGAs. The Xilinx 7-series devices have 32 global clock lines. The devices are also partitioned into clocking regions. Each clocking region can support up to 12 global clocks. Clocking regions also support regional clocks. Up to four unique regional clocks can be supported in each of the clock regions. Detailed information on the clocking structure on Xilinx series-7 devices can be found in [28]. These clock regions can be used to distribute the local clocks to various latches in the design, however, there is a limit on the number of clocks that can be distributed using this method in each region and also in the chip.

There are designs that could potentially require a much larger number of controllers [7]. Without, having to make changes to the routing structure on the chip, it would be possible to distribute clocks with minimal skew, using existing routing resources, with the addition of a PDE to the slices. Using the advance placement constraints it is possible to place all latches associated with a controller in close proximity to the controller on the chip, reducing the possible delays. Once the delays have been minimized the programmable delay element can be used to equate all the delays from the source clock pin to the destination latches. Delay can be added to the faster routing lines to match the delay of the slowest line. This final *clock to latch* delay can now be used to adhere to the RT constraints as discussed in section IV.

Also, as discussed, RT constraints help in building glitchless control burst-mode controllers. This may require adding delays in a combinational logic structure. Burst mode controllers do not use any latches or flip flops, but instead use local combinational logic feedback paths to implement state holding logic. Hence, it would be possible to commandeer the PDE to add the required additional delay to certain signals in the burst mode controllers to implement required RT signal ordering.

Fig. 3 shows the added PDE. In this case each LUT has a PDE associated with it. However, this may be an overkill. The need to add delay to a logic line may be rare and sporadic in terms of resource utilization. Hence adding a single PDE per slice would suffice. Transmission gates are used to reduce the load on the *A* output with the PDE is not being used, to

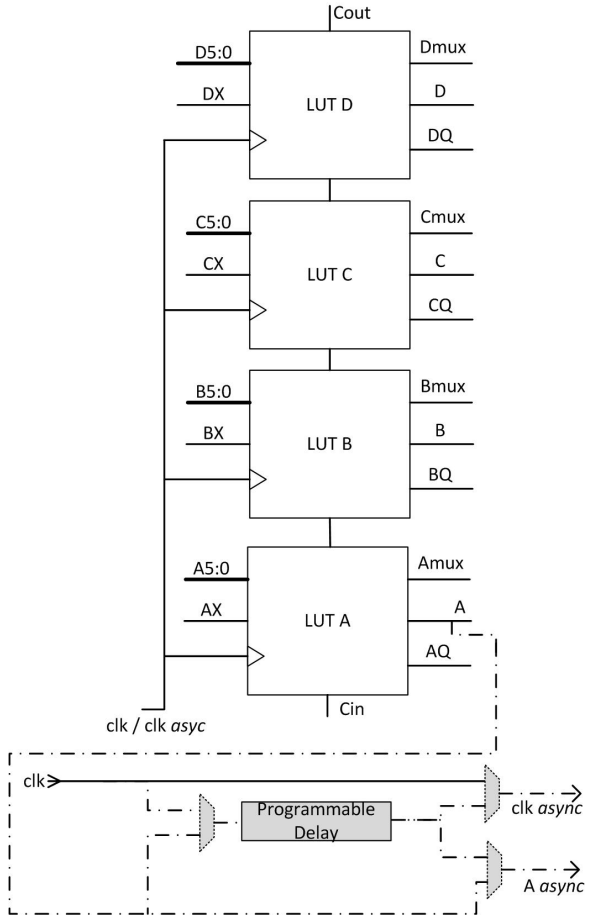

Fig. 4: Slice Structure. Additional logic to make an A-FPGA has been shown with shaded elements and dashed-wires

conserve power, and also keep the performance metrics of the output close to those of similar outputs without PDEs.

Simplified Feedback Routing:

Most burst-mode controllers use feedback signals for state holding. These feedback signals are used to stabilize the circuit in the right state. An input change before the feedback is allowed to stabilize the circuit can often lead to hazards as discussed in section IV. It is common to have RT constraints that require feedback paths to be fast. This allows the circuit to stabilize in its current state before a new input event.

Current FPGAs route feedbacks through the routing matrix. This incurs high delays. Faster feedback paths may allow significant boost in controller cycle times. An internal slice feedback structure is shown in Fig. 3. A transmission gate is used to turn the signal on and off. The signal needs to be physically tied to only one of the *A5:0* inputs. When the feedback route in invoked, the associated input must be left unconnected by the routing matrix. This will avoid two drivers for the same signal. Using the transmission gate again reduces the load on the *Amux* output when the feedback route is not in use. Also as with the PDE, it may be an over kill to have this feedback structure associated with each LUT. It may only be used for the implementation of the burst mode controllers on the FPGA fabric. Having one or two of the LUTs per slice associated with this feedback may very well suffice. The controllers are also usually a very small portion of the total design. Hence it may be a viable alternative to have only some slices with the feed-back structure and have these slices spread across the FPGA.

Latches:

Bundled data asynchronous designs use level triggered latches rather than edge triggered flip flop as registers and synchronizing elements. On ASICs, using latches instead of D-Flip Flops provides a significant power, area and performance benefit. The current Xilinx architectures provide us with sequential flip flops that can also be used as a latch. However, there is very little power or performance benefit in using these latches instead of the flip flops in current FPGA designs.

In terms of simply prototyping asynchronous ASICs on an A-FPGA, it could be possible to use the flip flops as latches as we are primarily concerned with functionality rather than performance. However, it is also the intention of the authors to present the proposed FPGA as a viable alternative to purely synchronous FPGAs. There are a wide variety of designs that could tremendously benefit from an asynchronous based implementation on an A-FPGA. Hence, the addition of latches to the architecture can help in achieving better power and performance number compared to synchronous implementations on the FPGA.

VI. Results

The synchronous and the asynchronous FPGA architectures were implemented on the 64nm node. When architecture shown in Fig. 3 was implemented and compared to the synchronous architecture, it showed a 30% increase in area. This seemingly large area penalty is because of the primitive programmable delay structure that was used in the implementation. The area penalty can be significantly reduced to under 15% by using custom programmable delay techniques [29]. However, the total slice area, as shown in Fig. 4 had only a 7% increase in area. A significant portion in any FPGA are the routing resources that consume over 50% of the chip area. That being considered, the area increase to make a synchronous FPGA capable of implementing RT based asynchronous circuits falls below 4%. The implementation included only the logic and register structure of the slice. The configuration bits, usually implemented as SRAM cells, were not included; rather the configuration bits were treated as inputs to the design.

There was an insignificant change in the slice performance of the A-FPGA being used synchronously when compared to the performance of the synchronous architecture. This can be attributed to the fact that while being used in the synchronous fashion most additional A-FPGA components can be safely ignored.

VII. Conclusion

The paper presents an A-FPGA architecture that uses a traditional synchronous FPGA as a starting point and propose alterations to the slice architecture. The new A-FPGA

architecture can implement RT based asynchronous circuits. Changes are made only to the logic resources, leaving the routing resources untouched. The proposed changes include a programmable delay element for clock and logic signals, dedicated and faster feedback routing, and inclusion of latches as an alternative to the flip flop.

The new FPGA architecture can be used for synchronous or asynchronous designs, or for designs that have both components. The architectural changes have little or no impact on the synchronous performance of the FPGA. Only a small area penalty is incurred in the logic resources. It is speculated that the architecture will allow the transfer of benefits that asynchronous designs have provided on ASICs to the FPGAs fabric. However this remains to be verified with subsequent research.

VIII. Acknowledgments

This material is based upon the work supported by the National Science Foundation under Grant Number 0810408.

References

[1] S. Hauck, S. Burns, G. Borriello, and C. Ebeling, " An FPGA for Implementing Asynchronous Circuits," *IEEE DESIGN AND TEST OF COMPUTERS*, vol. 11, no. 3, pp. 60–69, 1994.

[2] C. Wong, A. Martin, and P. Thomas, "An Architecture for Asynchronous FPGAs," in *Field-Programmable Technology (FPT), 2003. Proceedings. 2003 IEEE International Conference on*, Dec 2003, pp. 170–177.

[3] A. Rettberg and B. Kleinjohann, "A Fast Asynchronous Re-configurable Architecture for Multimedia Applications," in *Integrated Circuits and Systems Design, 2001, 14th Symposium on.*, 2001, pp. 150–155.

[4] A. P. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*, 1st ed. Wiley-IEEE Press, 2000.

[5] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, no. 10, pp. 1904–1921, 2006.

[6] S. Rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev, "RAPPID: An Asynchronous Instruction Length Decoder," in *Advanced Research in Asynchronous Circuits and Systems, 1999. Proceedings., Fifth International Symposium on*, 1999, pp. 60–70.

[7] W. Lee, V. S. Vij, A. R. Thatcher, and K. S. Stevens, "Design of Low Energy, High Performance Synchronous and Asynchronous 64-point FFT," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, 2013, pp. 242–247.

[8] K. Stevens, R. Ginosar, and S. Rotem, "Relative timing [asynchronous design]," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, no. 1, pp. 129–140, 2003.

[9] K. Maheswaran and J. Lipsher, "A Cell Set for Self-Timed Design Using Xilinx XC4000 Series FPGAs," Tech. Rep., 1994.

[10] D. Oliveira, S. Sato, O. Saotome, and R. de Carvalho, "Hazard-Free Implementation of the Extended Burst-Mode Asynchronous Controllers in Look-Up Table based FPGA," in *Programmable Logic, 2008 4th Southern Conference on*, 2008, pp. 143–148.

[11] J. Teifel and R. Manohar, "Highly Pipelined Asynchronous FPGAs," in *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, ser. FPGA '04, 2004, pp. 133–142.

[12] C. LaFrieda, B. Hill, and R. Manohar, "An Asynchronous FPGA with Two-Phase Enable-Scaled Routing." in *ASYNC*. IEEE Computer Society, 2010, pp. 141–150.

[13] N. Huot, H. Dubreuil, L. Fesquet, and M. Renaudin, "FPGA Architecture for Multi-Style Asynchronous Logic." in *DATE*. IEEE Computer Society, 2005, pp. 32–33.

[14] D. Shang, F. Xia, and A. Yakovlev, "Asynchronous FPGA Architecture with Distributed Control," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, May 2010, pp. 1436–1439.

[15] X. Jia and R. Vemuri, "The GAPLA: A Globally Asynchronous Locally Synchronous FPGA Architecture," in *Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on*, April 2005, pp. 291–292.

[16] ——, "A Novel Asynchronous FPGA Architecture Design and its Performance Evaluation," in *Field Programmable Logic and Applications, 2005. International Conference on*, Aug 2005, pp. 287–292.

[17] K. Sun, X. Pan, J. Wang, and J. Wang, "Design of A Novel Asynchronous Reconfigurable Architecture for Cryptographic Applications," in *Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on*, vol. 2, June 2006, pp. 751–757.

[18] K. Sun, L. Ping, J. Wang, Z. Liu, and X. Pan, "Design of a Reconfigurable Cryptographic Engine," in *Advances in Computer Systems Architecture*, ser. Lecture Notes in Computer Science, C. Jesshope and C. Egan, Eds. Springer Berlin Heidelberg, 2006, vol. 4186, pp. 452–458.

[19] T. Beyrouthy, A. Razafindraibe, L. Fesquet, M. Renaudin, S. Chaudhuri, S. Guilley, J.-L. Danger, and P. Hoogvorst, "A Novel Asynchronous e-FPGA Architecture for Security Applications," in *Field-Programmable Technology, 2007. ICFPT 2007. International Conference on*, Dec 2007, pp. 369–372.

[20] S. Chaudhuri, S. Guilley, P. Hoogvorst, J.-L. Danger, T. Beyrouthy, A. Razafindraibe, L. Fesquet, and M. Renaudin, "A Secure Asynchronous FPGA Architecture, Experimental Results and Some Debug Feedback," *CoRR*, vol. abs/1103.1360, 2011.

[21] A. T. P. Group, *The Balsa Asynchronous Synthesis System*, 2009. [Online]. Available: http://apt.cs.manchester.ac.uk/projects/tools/balsa/

[22] H. Solutions, *TiDE Manual*, 2007.

[23] A. Peeters and M. de Wit, "Haste Manual." Hanshake Solutions, 2007. [Online]. Available: www.handshakesolutions.com

[24] C. Myers, *Asynchronous Circuit Design*, 2001.

[25] Y. Xu and K. Stevens, "Automatic synthesis of computation interference constraints for relative timing verification," in *Computer Design, 2009. ICCD 2009. IEEE International Conference on*, 2009, pp. 16–22.

[26] K. Stevens, Y. Xu, and V. Vij, "Characterization of Asynchronous Templates for Integration into Clocked CAD Flows," in *Asynchronous Circuits and Systems, 2009. ASYNC '09. 15th IEEE Symposium on*, May 2009, pp. 151–161.

[27] X. Inc, *7 Series Configuration Logic Block: User Guide*, 2013.

[28] I. "Xilinx, *7 Series FPGAs CLocking Resources: User Guide*, 2014.

[29] S. Kobenge and H. Yang, "A Power Efficient Digitally Programmable Delay Element for Low Power VLSI Applications," in *Quality Electronic Design, 2009. ASQED 2009. 1st Asia Symposium on*, July 2009, pp. 83–87.