Are Asynchronous ideas useful in FPGAs?

Peter Y. K. Cheung

Department of EEE, Imperial College London
London SW7 2BT
p.cheung@imperial.ac.uk

Abstract. Professor Yakovlev has contributed to the field of asynchronous designs and methodologies for over three decades. I have worked in the area of reconfigurable circuits and systems for almost as long. On this happy occasion celebrating his 60th birthday, it is fitting to make an attempt to examine possible synergy between the two fields and reflect upon where we might find a common intersection if one exists.

Keywords: asynchronous circuits, reconfigurable, FPGAs

1 Introduction

I have been looking forward to celebrate Professor Alex Yakovlev's academic achievements with him at his Festschrift for the past three years, ever since my own Festschrift held in 2013 [1]. At that time, he contributed a thoughtful article and spoke eloquently at the workshop, and I have been asking him when his own Festschrift will take place ever since.

Although I have known Alex and admired his work for over two decades, particularly in his contributions towards the modelling of asynchronous systems using Petri nets [2] [3]], he and I have not worked together on a project until very recently. I have dabbled in asynchronous circuits in the past, but my own area of research has, for the past two decades, been in the field of reconfigurable systems, particularly those using FPGAs. Therefore it is obvious for me to contribute on this happy occasion an article that focuses on both asynchronous ideas and FPGAs.

I have chosen to write this article completely from scratch. I want to take this opportunity to pontificate on the subject of asynchrony in the context of configurable and reconfigurable digital circuits. The nice thing about writing for a Festschrift is that I can be less rigorous and scientific, more reflective and opinionated, without the need to support my hypotheses and thoughts with objective and experimental data. This is like a columnist writing an article for the Financial Times, such as my favourite economist Mr John Kay, whose many insightful essays have taught me much.

2 My journey into Asynchronous Designs

Before I attempt to relate asynchronous circuits to FPGAs, it is worth putting my thoughts in context. My own experience in asynchronous circuits is limited. I was first introduced to the subject when Ivan Sutherland spent a 15-months sabbatical leave in our Research Group at Imperial College in the mid-80s. At that time, he was working on the design of an asynchronous multiplier [4] which also led to him developing the method of logical effort [5]. I was a young lecturer at the time trying to establish myself in the competitive world of academia. Through Ivan and his research assistant Ian Jones, I learned the basic principles of asynchronous circuits, such as asynchronous FIFO, the C-element, dual-rail signaling etc.. Subsequently, I recruited three research students to work with me on the subject and published a few insignificant papers [6] [7] [8]. We spent many hours debating among ourselves the pros and cons of asynchronous circuits, and through our discussions and arguments, we came to the conclusion that we wanted to work in the area of composability of asynchronous systems.

A few years later, my friend and colleague Wayne Luk from Oxford joined Imperial College. Mainly because of this, I moved my research to the area of reconfigurable systems.

3 Asynchronous Reconfigurable Logic

Early attempts to put asynchronous circuits in conventional synchronous FPGAs failed miserably. FPGA fabrics are fundamentally register-rich in architecture and early flip-flops in FPGAs were particularly prone to metastability problems. Asynchronous components such as the C-element could be implemented using fine-grain architectures on those early generations of FPGA fabrics such as the XC6200 [9] and Actel's ACT-1 devices [10]. Unfortunately implementing reliable asynchronous systems of reasonable size were very challenging. This prompted researchers to design asynchronous FPGAs with special circuit elements dedicated to the asynchronous design style.

There have been various attempts in designing asynchronous FPGA fabrics. One of the earliest efforts was by Scott Hauck and Carl Ebeling [11]. Montage was the first FPGA that included dedicated support for both asynchronous and synchronous circuits. It even came with mapping software to support designers. It was shortly followed by the work of Payne [12], Traver [13], Teifel [14] and Martin [15]. As someone who has a passing interest in asynchronous circuits and techniques, and made his research career in FPGA, I have frequently asked the question: why asynchronous FPGAs never took traction either in industry or in academia? The following is a brief exposition of my personal views in the matter.

4 The potential and reality of asynchronous in reconfigurable

On paper asynchronous circuits have many attractive features [16]. The idea of doing away with rigid timing imposed by synchronous circuit's clock signal can be appealing - even liberating! However, are these potential advantages relevant and applicable to FPGAs?

4.1 Clock skews

Potential - Eliminating the need to distribute clock signals throughout a chip also removes the problem of clock skews. Clock networks in modern FPGAs are complex and occupy significant amount of silicon area. Unlike ASIC designs, FPGA devices need to accommodate arbitrary application circuits not known to the FPGA architect at the time the clock networks are designed. The loading on nodes in a clock tree are also not known before hand. This exacerbates the clock skews problem even further.

Reality - Indeed handling clock skews on large FPGAs is a problem. It may be possible to measure the skews and mitigate their effects at run-time as suggested in [17]. The effectiveness of such an approach is unproven. However, having clocks in FPGAs is so fundamental to the entire FPGA design flow that industry simply chooses to solve the clock skews problem, no matter how challenging. Until recently, the approach taken has been to carefully design fixed, highly buffered, clock tree networks and to develop accurate timing models of the reconfigurable fabric including the programmable routing resources. The timing model is tightly coupling to place-and-route algorithms in the CAD tools.

Recently, Altera has taken a much more radical approach to this problem [18] in their latest Stratix-10 family of FPGAs. Instead of using fixed clock networks, they use a highly configurable, routable approach to customize the clock network based on the user's design. This approach potentially reduces the impact of clock skews and makes the technique scalable for future larger FPGAs. Xilinx has taken a different approach in their Ultrascale architecture [19]. Distribution of clocked is based on clock regions which, unlike Stratix-10, are not customizable. However, the clock grid comprises of routing tracks and distribution tracks, which provide optimal routing of the clock signals from the centre of the region (called clock root).

Both Altera and Xilinx have successfully tackled the clock skews problem in different ways. In both cases, clock skews will not be a show stopper as we move into future generations of larger, faster, and denser FPGAs.

4.2 Timing closure

Potential - Any designer who has grappled with fitting a large design onto an FPGA will tell you that timing closure is often a big headache. FPGA designs relies heavily on the flexible, programmable interconnect resources. The unpredictability in the delays of interconnects often presents many challenges. Even a minor change in a previously working design could require many hours of work in order to achieve timing closure again. The self-timed nature of asynchronous FPGAs could do away with this problem.

Reality - As size and density of FPGA devices keep increasing, achieving timing closure is expected to be increasingly difficult. However, FPGA manufacturers have made great strides in easing this through two advances. Xilinx has pushed stacked silicon interconnect technology that promises to deliver much higher capacity, higher data bandwidth and power efficiency [20]. Altera recently announced their highly pipelined Stratix-10 architecture where they include programmable pipeline registers within the routing fabric. Coupled with their CAD tools that are designed to exploit this architectural innovation, they mitigate the timing closure problem by simplifying circuit retiming with improved performance [21].

4.3 Reduced power

Potential - Having to distribute a clock signal everywhere, even when to those registers where the input signals remain unchanged consumes unnecessary power. Even though clock gating helps in reducing unnecessary clock propagation to inactive regions, the idea that data drives circuit activities can be very attractive. FPGAs are already much larger and consume more power than their ASIC equivalent. Asynchronous circuit could regain some of the power efficiency loss.

Reality - Although asynchronous circuits have the potential of consuming less power than their synchronous counterpart, FPGAs have never really been the technology of choice where power is the primary concern. One would not find many FPGAs used in battery operated mobile devices. Therefore reduced power consumption has never been a significant driver in pushing the asynchronous agenda in FPGAs. Furthermore, the advances in process technology and the continual reduction in power supply voltages have alleviated the concern of overheating in large FPGAs. For example, using TSMC 28nm low power (HPL) process, Xilinx has introduced three separate families of devices to meet the market's requirement: the Artix-7 devices for low power and low cost, the Kintex-7 devices for a balance in price and performance, and the Virtex-7 devices for high performance and high capacity [22]. In the meantime, Altera introducing Aria-10 and Stratix-10 families of FPGAs, both claiming significant improvements in power consumption for a given workload when compared with previous generations of FPGAs [23].

4.4 Improved performance

Potential - It is well known that asynchronous circuits allow *average-case* instead of *worst-case* performance. Synchronous FPGAs, for example, would have to wait for the ripple-carry chain to reach the most significant bit in an add operation before the sum is available. Asynchronous FPGAs with its inherent completion detection and signalling would allow on average faster operations.

Reality - The arguments for improved performance using asynchronous techniques have never been proven in industrial designs. Neither Sun Microsystem's asynchronous Sparc processor nor AMULET, the asynchronous ARM processor, demonstrated performance gain over their synchronous counterpart implemented using the technology. In any case, having better *average-case* performance gain is no advantage in many real-time applications where a guaranteed completion time is more important than having faster operation some of the times. Deterministic behaviour is often more desirable than indeterministic, data-dependent delay or latency. In any case, in the world of FPGAs, advancement in process technology, new transistor devices (e.g. FinFET transistors) and new FPGA architectures have continued to push the boundary of FPGA performances. In addition, the use of 2.5D and 3D integration in FPGAs significantly reduces interconnect delays, which are particularly important for FPGA designs where interconnect delays are often the main limitation to system level performance.

4.5 Composability and reusability

Potential - Delay insensitivity in asynchronous circuits allows the assembly and integration of modules forming a complete digital system with ease. This promotes the divide-and-conquer approach to design and allows previously designed and verified modules to be reused with minimal effort, very much like the way software library functions can be used in building a large programme. Therefore asynchronous techniques permit FPGA designers easier routes to perform reconfiguration of virtual hardware blocks even during runtime.

Reality - There have been much effort devoted to the idea of run-time reconfiguration (RTR) in FPGAs so that modules could be swapped in and out of the FPGA fabric, very much like the way we swap blocks of memory in a virtual memory system. The asynchronous paradigm where each module is self-contained with its own timing appears to be a perfect match to this particular vision of FPGAs. Unfortunately RTR has not really been widely used in industry beyond design upgrade or system retargeting. There are many reasons for this including the overhead in reconfiguration if it is done frequently, the difficult in verification of such a system (similar to the reason why self-modifying code is not encouraged in software), and the lack of good CAD tools to handle such a RTR system.

However, with ever increasing size and density in FPGAs, partial reconfiguration is now becoming not just a reality but a necessity. Putting a working design on an FPGA device with 2 million logic cells requires a design flow that allows composability and resuability. The FPGA industry is currently solving this problem, not with the asynchronous paradigm, but with high-level synthesis methodology.

There is now a widespread adoption of OpenCL as the language of choice for FPGA designs, particularly after the integration of powerful embedded CPU into the FPGA fabrics as found in Xilinx's and Altera's latest family of devices. This approach is totally opposite of that used in the asynchronous paradigm. In asynchronous, a successful design requires high degree of skills and understanding of hardware design down to the minute details. The approach taken by the FPGA industry is to push the hardware design skills required to a minimum and makes the FPGA fabric invisible. I personally will not bet against the scenario that very soon one could put designs on FPGAs without touching any low level descriptions such as Verilog or VHDL.

The introduction by Altera of highly pipelined flexible registers in its routing resources helps this process even further [21]. Now an OpenCL compiler can deal with the hardware compilation task without worrying about timing. The flexible routing registers can then be allocated subsequently to optimise the performance and the latency of a design. In my view, this is one of the most important advancements in FPGA architectures that enables effective high-level synthesis in recent years.

4.6 Tolerance to process and environmental variations

Potential - As technology scaling continues to advance and supply voltage continues to drop, uncertainties in delays in FPGA circuits due to variations in process, temperature, voltage and noise are beginning to post a challenge. Asynchronous circuits adapts to delay variation automatically. This is particularly significant if and when future electronic devices degrade faster with age. Older asynchronous FPGA chips will continue to function correct with reduced speed, while the synchronous counterpart could fail completely.

Reality - Mitigating the effect of variability with reconfigurable hardware has been the focus of my research in recent years. Alex contributed an article at my Festschrift on this very same topic [24]. FPGAs are inherently adaptable. It is therefore reasonable to ask the question how could one exploit the flexibility offered by FPGAs to mitigate the ever increasing variability factors found in modern integrated circuits?

Asynchronous paradigm is one possibility but this does not really need the configurability offered by FPGAs. The approach I have taken so far has been to perform online measurements on FPGAs devices with embedded instrumentations. One can then exploit such information to adapt the electronic systems in order to deal with variations by exploiting the configurability inherent in FPGAs. For example, we have recently successful demonstrated efficient and effective circuits to measure delays and timing slacks in arbitrary circuit paths [25] [26] and power consumption in modules within FPGAs [27]. These techniques are particularly suitable for FPGA based designs for the following reasons. Not only are FPGA reconfigurable, they also come with fixed amount of hardware resources on a chip. Provided that a design does not fill the entire chip, there are generally spare hardware resources that one could deploy for such embedded instrumentations. Putting the unused resources to provide better adaptation not only improves performance, reduces power consumption, it also improves the reliability of the system. The challenge is in how to add such instruments without impacting on the already complicated design flow. For this, we can learn from the experience in the adoption of boundary scans in test. Boundary scan register insertion in ASIC is now an automatic and, at least to the designer, a more or less invisible process. It is my view that one day on-chip embedded instrumentations will be the same.

5 Concluding Remarks

When I first set out to write this article, I was intending to focus on the synergy between asynchronous and reconfigurable. As I was completing the article, I realised that my conclusion is rather negative towards asynchronous. This is both a surprise and a disappointment. I have always found the asynchronous paradigm rather beautiful. I also admire those who have made this their lifelong work. However, no matter how much I want to make asynchronous ideas relevant and useful to FPGAs, everywhere I turned, I was confronted with strong counter arguments. The reality of industrial practices also suggests that I am probably right.

Notwithstanding, I strongly believe that there is one aspect where the asynchronous paradigm is useful to FPGAs and this is in the integration of modules via the globally asynchronous, locally synchronous (GALS) methodology. If I look back in the history of electronics and the one asynchronous idea that we still use everyday is the serial communication using universal asynchronous receiver/transmitter (UART). Indeed communicates between multiple FPGAs are already relying on the very fast serial transceivers found on all modern FPGAs.

I have attending a few conferences on asynchronous in the past. Just like the FPGA community, the asynchronous community has spent many years searching for the "killer application" for their technology. May be the answer is that there is *no killer application*, but asynchronous is still a very useful glue. The paradigm, particularly with Alex's original field of research into modelling using Petri net, can act as the catalyst to many innovations in electronics.

In conclusion, while I still believe that asynchronous is useful in the field of reconfigurable, we are still searching for this intersection between the two fields. May be the recent effort of companies such as Achronix will show us the way. I am still waiting.

6 Acknowledgements

I want to thank Professor Alex Yakovlev for his friendship and encouragements that he generously offered to me over many years. I have enjoyed our many open and honest discussions, both technical and non-technical. I look forward to working with you for many years to come. May be we can even start something completely new in the future.

I also want to acknowledge the generous support of EPSRC in the following grants: PRiME: Power-efficient, Reliable, Many-core Embedded Systems (EP/K034448) and Variation-Adaptive Design in FPGAs (EP/H013784).

References

[1] Luk, W. and Constantinides, G.A., Transforming Reconfigurable Systems: A Festschrift Celebrating the 60th Birthday of Professor Peter Cheung. Imperial College Press, 2015
[2] J. Cortadella and M. Kishinevsky and L. Lavagno and A. Yakovlev, Synthesizing Petri nets from state-based models. Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on, pp.164-171, 1995
[3] J. Cortadella and M. Kishinevsky and L. Lavagno and A. Yakovlev, Deriving Petri nets from finite transition systems. IEEE Transactions on Computers, 47 (8), pp. 859-882, 1998
[4] Sutherland, I. E., Micropipelines. Commun. ACM, 32 (8), pp. 720–738, 1989
[5] Sutherland, Ivan E. and Sproull, Robert F., Logical Effort: Designing for Speed on the Back of an Envelope. Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI, pp. 1-16, 1991
[6] Molina, P. A., Cheung, P. Y. K. Quai delay-insensitive bus for fully asynchronous systems. IEEE International Symposium in Circuits and Systems, Vol. 4, pp. 189-192, 1996
[7] Bormann, D. S., Cheung, P. Y. K. Asynchronous wrapper for heterogeneous systems. International Conference on Computer Design, ICCD'97 pp. 307-314, 1997
[8] Royal, A., Cheung, P. Y. K. Globally Asynchronous Locally Synchronous FPGA Architectures International Workshop on Field Programmable Logic and Applications, LNCS 2778, pp. 355-364, 2003
[9] Brebner, Gordon, A virtual hardware operating system for the Xilinx XC6200. Proceedings of the 6th International Workshop on Field-Programmable Logic and Applications, pp. 327–336, FPL'96, 1996
[10] Brunvand, Erik A Cell Set for Self-Timed Design using Actel FPGAs Report UUCS-91-013, Department of Computer Science, University of Utah August 10, 1991
[11] Hauck, S., Burns, S., Borriello, G., Ebeling, C. An FPGA for Implementing Asynchronous Circuits. IEEE Design & Test of Computers, 11 (3), pp. 60-69, 1994
[12] Payne, R. E. Self-Timed FPGA Systems. 5th International Workshop on Field Programmable Logic and Applications, pp. 21-35, 1995
[13] Traver, C., Reese, R. B., Thornton, M. A. Cell Designs for Self-Timed FPGAs. 14th Annual IEEE International ASIC/SOC Conference pp. 175-179, 2001
[14] Teifel, J., Manohar, R. Highly Pipelined Asynchronous FPGAs Proceedings of the 2004 ACM International Symposium on FPGA pp. 133- 142, 2004.

[15] Wong, C. G., Martin, A. J., Thomas, P. An Architecture for Asynchronous FPGAs. IEEE International Conference on Field-Programmable Technology pp. 170-177, 2003

[16] Hauck, S. Asynchronous Design Methodology: An Overview Proceedings of the IEEE, 83 (1), pp. 69 - 93, January 1995

[17] Sedcole, P., Wong, J. S., Cheung, P. Y. K. Characterisation of FPGA Clock Variability IEEE Symposium on VLSI, pp. 322 - 328, 2008

[18] Ebeling, C., How, D., Lewis, D., Schmit, H. Stratix 10 High Performance Routable Clock Networks Proceedings of the 2016 ACM International Symposium on FPGA, pp. 64-73, 2016

[19] Xilinx Inc. Ultrascale Architecture clocking Resources - User Guide. *www.xilinx.com/support/documentation/user_guides/ug572-ultrascale-clocking.pdf*

[20] Saban, S. Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency. White Paper WP380, Xilinx Inc., Dec 11, 2012

[21] Lewis, D., Chiu, G., Chromczak, J., Galloway, D., Gamsa, B. The Stratix 10 Highly Pipelined FPGA Architecture Proceedings of the 2016 ACM International Symposium on FPGA, pp. 159-168, 2016

[22] Mohson, E. Reducing System Power and Cost with Artix-7 FPGAs. White Paper WP423, Xilinx Inc., Dec 17, 2014

[23] Lim, S. Expect a Breakthrough Advantage in Next-Generation FPGAs. White Paper WP-01199, Altera Corp., June 2015

[24] Yakovlev, A. Enabling Survival Instincts in Electronic Systems: An Energy Perspective. In Transforming Reconfigurable Systems: A Festschrift Celebrating the 60th Birthday of Professor Peter Cheung Imperial College Press, 2015

[25] Wong, J. S. J., Cheung, P. Y. K. Timing Measurement Platform for Arbitrary Black-Box Circuits Based on Transition Probability. IEEE Transactions on VLSI Systems, 21 (12), pp. 2307-2320, 2013

[26] Levine, J. M., Stott, E., Cheung, P. Y. K. Dynamic voltage and frequency scaling with online slack measurement. Proceedings of the 2014 ACM International Symposium on FPGA pp. 65-74, 2014

[27] Hung, E., Davis, J. J., Levine, J. M., Stott, E. A., Cheung, P. Y. K., Constantinides, G. A. KAPow: A System Identification Approach to Online Per-module Power Estimation in FPGA Designs. Proceedings of the 24th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2016