Overview of Asynchronous Logic

For the last three decades the focus of digital design has been primarily on synchronous, clocked architectures. However, as clock rates have significantly increased while feature size has decreased, clock skew has become a major problem. High performance chips must dedicate increasingly larger portions of their area for clock drivers to achieve acceptable skew, causing these chips to dissipate increasingly higher power, especially at the clock edge, when switching is most prevalent. As these trends continue, the clock is becoming more and more difficult to manage, while clocked circuits' inherent power inefficiencies are emerging as the dominant factor hindering increased performance. These issues have caused renewed interest in asynchronous digital design. Asynchronous, clockless circuits require less power, generate less noise, and produce less electro-magnetic interference (EMI), compared to their synchronous counterparts, without degrading performance. Furthermore, ***delay-insensitive (DI)*** asynchronous paradigms have a number of additional advantages, especially when designing complex circuits, like Systems-on-a-Chip (SoCs), including substantially reduced crosstalk between analog and digital circuits, ease of integrating multi-rate circuits, and facilitation of component reuse. Asynchronous circuits can even utilize a synchronous wrapper, such that the end user does not know that the internal circuitry is actually asynchronous in nature. Currently, companies such as ARM, Phillips, Intel, and others are incorporating asynchronous logic into some of their products using their own proprietary tools.

As demand increases for designs with higher performance, greater complexity, and decreased feature size, asynchronous paradigms will become more prevalent in the multi-billion dollar semiconductor industry, as predicted by the International Technology Roadmap for Semiconductors (ITRS), which envisions a likely shift from synchronous to asynchronous design styles in order to increase circuit robustness, decrease power, and alleviate many clock-related issues. Furthermore, ITRS states that asynchronous circuits will account for 19% of chip area within the next 5 years, and 30% of chip area within the next 10 years.

Asynchronous circuits can be grouped into two main categories: *bounded-delay* and *delay-insensitive* models. Bounded-delay models, such as *micropipelines* [1], assume that delays in both gates and wires are bounded. Delays are added based on worse-case scenarios to avoid hazard conditions. This leads to extensive timing analysis of worse-case behavior to ensure correct circuit operation. On the other hand, delay-insensitive circuits assume delays in both logic elements and interconnects to be unbounded, although they assume that wire forks within basic components, such as a full adder, are isochronic, meaning that the wire delays within a component are much less than the logic element delays within the component, which is a valid assumption even in future nanometer technologies. Wires connecting components do not have to adhere to the isochronic fork assumption. This implies the ability to operate in the presence of indefinite arrival times for the reception of inputs. Completion detection of the output signals allows for handshaking to control input wavefronts. Delay-insensitive design styles therefore require very little, if any, timing analysis to ensure correct operation (i.e., they are correct by construction), and also yield average-case performance rather than the worse-case performance of bounded-delay and traditional synchronous paradigms.

Most delay-insensitive methods combine *C-elements* with Boolean gates for circuit construction. A C-element behaves as follows: when all inputs assume the same value then the output assumes this value, otherwise the output does not change. Seitz's [2], DIMS [3], Anantharaman's [4], Singh's [5], and David's [6] methods are examples of DI paradigms that only use C-elements to achieve delay-insensitivity. On the other hand, both Phased Logic [7] and NULL Convention Logic (NCL) [8] target a library of multiple gates with *hysteresis* state-holding functionality. Phased Logic converts a traditional synchronous gate-level circuit into a delay-insensitive circuit by replacing each conventional synchronous gate with its corresponding Phased Logic gate, and then augmenting the new network with additional signals. NCL circuits are realized using 27 fundamental gates implementing the set of all functions of four or fewer variables, each with *hysteresis* state-holding functionality.

Seitz's method, Anantharaman's approach, and DIMS require the generation of all minterms to implement a function, where a minterm is defined as the logical AND, or product, containing all input signals in either complemented or non-complemented form. While Singh's and David's methods do not require full minterm generation, they rely solely on C-elements for speed-independence. NCL also does not require full minterm generation and furthermore includes 27 fundamental state-holding gates for circuit design, rather than only C-elements, thus yielding a greater potential for optimization than other delay-insensitive paradigms. Phased Logic also does not require full minterm generation and does not rely solely on C-elements for speed-independence; however, Phased Logic circuitry is derived directly from its equivalent synchronous design, not created independently, thus it does not have the same potential for optimization as does NCL. Furthermore, the Phased Logic paradigm has been developed mainly for easing the timing constraints of synchronous designs, not for obtaining speed and power benefits, whereas these are main concerns of other asynchronous paradigms.

Self-timed circuits can also be designed at the transistor level as demonstrated by Martin [9]. However, automation of this method would be vastly different than that of the standard synchronous approach, since it optimizes designs at the transistor level instead of targeting a predefined set of gates, as do the previously mentioned methods. Overall, NULL Convention Logic offers the best opportunity for integrating asynchronous digital design into the predominantly synchronous semiconductor design industry for the following reasons:

1) The framework for NCL systems consists of DI combinational logic sandwiched between DI registers, which is very similar to synchronous systems, such that the automated design of NCL circuits can follow the same fundamental steps as synchronous circuit design automation. This will enable the developed DI design flow to be more easily incorporated into the chip design industry, since the tools and design process will already be familiar to designers, such that the learning curve is relatively flat.
2) NCL systems are delay-insensitive, making the design process much easier to automate than other non-DI asynchronous paradigms, since minimal delay analysis is necessary to ensure correct circuit operation.
3) NCL systems have power, noise, and EMI advantages compared to synchronous circuits, performance and design reuse advantages compared to synchronous and non-DI asynchronous paradigms, area and performance advantages compared to other DI paradigms, and have a number of advantages for designing complex systems, like SoCs, including substantially reduced crosstalk between analog and digital circuits, ease of integrating multi-rate circuits, and facilitation of component reuse and technology migration.

As the trend towards higher clock frequency and smaller feature size continues, power consumption, noise, and EMI of synchronous designs increase significantly. With the absence of a clock, DI systems aim to reduce power consumption, noise, and EMI. DI circuits designed using CMOS exhibit an inherent idle behavior since they only switch when useful work is being performed, unlike clocked Boolean circuits that switch every clock pulse, unless specifically disabled through specialized circuitry, which itself requires additional area and power. DI circuits adhere to monotonic transitions between DATA and NULL, so there is no glitching, unlike clocked Boolean circuits that produce substantial glitch power. DI systems better distribute switching over time and area, reducing the occurrence of hot spots, peak power demand, and system noise, unlike clocked Boolean circuits where much of the circuitry switches simultaneously at the clock edge. Furthermore, DI systems are very tolerant of power supply variations, allowing cheaper power supplies to be used and voltage to be dramatically reduced to meet desired performance while decreasing power consumption. Therefore, a very fast DI circuit can be run at a lower voltage to reduce power consumption when high performance is not required. Other DI advantages include tolerance of vast temperature differences, making these circuits well suited for operation in harsh environments, like outer space, and easing the difficulty of integrating designs with non-harmonically related clock frequencies. Their main disadvantage is increased area, which is approximately 1.5 – 2 times as much as an equivalent synchronous design when using static CMOS gates, but less for semi-static CMOS gates. However, for large designs, such as SoCs, the processor core(s) normally require(s) less than ½ of the chip's total area, while the rest of the chip area consists of flash, cache, RAM, peripherals, etc., which are the same in both DI and synchronous implementations. Therefore, the increased area for the DI implementation of the processor core(s) is less significant, especially considering the increased robustness and numerous other advantages.

References

[1] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, Vol. 32/6, pp. 720-738, 1989.

[2] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, pp. 218-262, 1980.

[3] J. Sparso, J. Staunstrup, M. Dantzer-Sorensen, "Design of Delay Insensitive Circuits using Multi-Ring Structures," *Proceedings of the European Design Automation Conference*, pp. 15-20, 1992.

[4] T. S. Anantharaman, "A Delay Insensitive Regular Expression Recognizer," *IEEE VLSI Technical Bulletin*, Sept. 1986.

[5] N. P. Singh, "A Design Methodology for Self-Timed Systems," *Master's Thesis*, MIT/LCS/TR-258, Laboratory for Computer Science, MIT, 1981.

[6] I. David, R. Ginosar, and M. Yoeli, "An Efficient Implementation of Boolean Functions as Self-Timed Circuits," *IEEE Transactions on Computers*, Vol. 41/1, pp. 2-10, 1992.

[7] D. H. Linder and J. H. Harden, "Phased Logic: Supporting the Synchronous Design Paradigm with Delay-Insensitive Circuitry," *IEEE Transactions on Computers*, Vol. 45/9, pp. 1031-1044, 1996.

[8] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, pp. 261-273, 1996.

[9] A. J. Martin, "Compiling Communicating Processes into Delay-Insensitive VLSI Circuits," *Distributed Computing*, Vol. 1/4, pp. 226-234, 1986.