On-Chip Communication Design: Roadblocks and Avenues [*]

[Invited Talk - Extended Abstract]

Luca P. Carloni
EECS Dept., UC Berkeley, Berkeley, CA 94720
http://www-cad.eecs.berkeley.edu/HomePages/lcarloni

Alberto L. Sangiovanni-Vincentelli
EECS Dept., UC Berkeley, Berkeley, CA 94720
http://www-cad.eecs.berkeley.edu/HomePages/alberto

ABSTRACT

The semiconductor industry is experiencing a paradigm shift from "computation-bound design" to "communication-bound design": the number of transistors that can be reached in a clock cycle, and not those that can be integrated on a chip, will drive the design process. Interconnect latency will have a major impact on the design of on-chip communication architectures, which increasingly rely on wire pipelining to go beyond the capabilities of traditional wire buffering. The insertion of stateful repeaters *on long wires, instead of simply* stateless repeaters, *carries major consequences for the synchronous design methodology. This is the foundation of the design flows for the majority of commercial chips today, but, if left unchanged, will lead to an exacerbation of the timing closure problem for tomorrows design flows. New methodologies that regard the chip as a distributed system are necessary. Latency-insensitive design is a step in this direction.*

Categories and Subject Descriptors

B.7.1 [**Hardware**]: Integrated Circuits—*Types and Design Styles*; B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*.

General Terms

Design

Keywords

On-Chip Networks, Latency-Insensitive Design, GALS.

1. COMMUNICATION-BOUND DESIGN

On-chip communication threatens to become a roadblock for the continuing technology progress that has been at the basis of the success of the semiconductor industry. Back in 1997, D. Matzke [13] published a study containing a gloomy forecast on Moore's Law hampered by on-chip interconnect latency (predicted to be measured soon in tens of clock cycles). This forecast has been echoed in many subsequent works studying the challenges posed by deep sub-micron design (DSM). Despite the increase in number of metal layers and in aspect ratio and the introduction of copper metallization and low-κ dielectric insulators, the resistance-capacitance (RC) delay of an average metal line with constant length is getting worse with each process generation [2, 10]. The combined increase of RC delays, chip operating frequency, die size, and average interconnect length, makes interconnect delay become the largest fraction of the clock cycle time. In particular, the relative scaling properties of wires and gates are at the center of this phenomenon. It is true that those wires that scale in length together with gate lengths present approximately a constant resistance and a falling capacitance. However, a future system-on-chip (SOC) will present many *global* wires that cannot scale in length because they need to span across multiple on-chip modules to connect distant gates [11]. In fact, the intrinsic interconnect delay of a $1mm$ length wire for a $35nm$ technology is expected to be $250ps$, i.e. two orders of magnitude larger than the corresponding minimum-geometry transistor delay [10, 14]. By studying the effects of technology scaling on traditional microprocessor architecture, Agarwal *et al.* predict that for a $35nm$ technology, the latency to transmit a signal across the chip in a top-level metal wire will vary between 12 and 32 clock cycles, depending on the frequency of the system clock. Maybe even more importantly, they confirm the expectation that the fraction of the total chip area that can be reached in a single clock cycle will be quite small, likely not higher than 2% [1]. In fact, while the number of gates reachable in a cycle will not change significantly and the on-chip bandwidth will continue to grow, the percentage of the die reachable within one cycle is constantly decreasing. Hence, a major design paradigm shift must occur because designs will be bound by the amount of state and logic that can be reached within the required number of clock cycles (communication bound) and not anymore by the number of transistors that can be integrated on a single die (computation bound) [11].

[*]This research was supported in part by the NSF under the project ITR (CCR-0225610), by the SRC, and by the GSRC.

2. THE IMPACT OF WIRE LATENCY

While it has been demonstrated that interconnect impose primary limits on latency, energy dissipation, signal integrity, and design productivity for gigascale integration (GSI) [14], we focus here on the disruptive role played by interconnect latency with respect to traditional design practices [6]. The majority of today's commercial chips are realized using design flows that are based on synchronous design methodologies and are centered around the separation of two main stages: logic synthesis and physical design. In synchronous design, the chip operates under the control of a common clock signal, the system clock. This assumption, which strongly simplifies the system design and verification, translates into a main requirement: the delay of each combinational path (i.e., a signal path leaving a latch and traversing only combinational logic and wires before ending with another latch) must be smaller than the system clock period. Hence, the slowest combinational path (critical path) dictates the maximum operating frequency for the overall chip. But, since it is often the case that the operating frequency is given *a priori* as a design specification, each combinational path that presents a delay longer than the desired clock period in the final layout must be handled as a design exception that needs to be fixed. To fix these exceptions, designers use various techniques from wire buffering and transistor resizing to rerouting wires, replacing or resynthesizing modules, and even redesigning entire portions of the system. These techniques often force designers to many costly iterations between the various stages of the design flow before converging to a correct layout (*timing closure problem*). Among them, wire buffering is perhaps the most efficient but it carries precise limitations, because there is an optimal number of buffers that can be inserted on a wire of a given length to achieve a minimum transmission delay. If the required clock period is still smaller, it is necessary to go beyond wire buffering and break the long wire into smaller segments by inserting memory elements like latches or flip-flops (*wire pipelining*). This operation trades-off fixing a wire exception with increasing its latency by one or more clock cycles and allows to drive long wires with the same clock signal used to control short wires and logic gates. An alternative to wire pipelining is to drive long wires with slower clocks, thus effectively implementing the chip as a *multi-clock* system, where many isochronic regions exchange data at a speed slower than the one at which they operate. Finally, it is possible to avoid using clock mechanisms to drive long wires altogether and adopt instead *asynchronous* communication protocols to build globally-asynchronous locally-synchronous architectures (GALS). The common point among all these options is the fact that the chip must be regarded as a *distributed system* and this invalidates the main assumption of the synchronous design methodology. While this important consequence is well-understood for the case of multi-clock and GALS implementations, recent works on combining wire buffering and wire pipelining [8, 12] seem to neglect the key difference between inserting combinational buffers versus inserting latches or flip-flops. In fact, while the former are *stateless repeaters*, the latter are *stateful repeaters*, i.e. they are sequential elements that must be initialized and whose insertion modifies the latency (expressed in number of clock cycles) between the chip modules lying at the two extremes of the pipelined wire. In general, to initialize these elements and to interface them to the surrounding control logic, which was derived assumed a different communication latency, a certain amount of redesign must be performed with negative consequences on design productivity.

Finally, it must be noticed that techniques combining global placement and retiming may help avoiding the need of wire pipelining as long as the original design specification contains a sufficient number of latches that can be distributed along the interconnect paths [9]. However, retiming carries as an intrinsic limitation the fact that the number of latches on any feedback loop in the design must remain constant.

3. LATENCY-INSENSITIVE DESIGN

Latency-insensitive design [3, 5] is a methodology that guarantees the robustness of the system's functionality and performance with respect to arbitrary latency variations. The foundation of latency-insensitive design is the theory of latency-insensitive protocols [4]. A latency-insensitive protocol controls communication among components of a patient system, a synchronous system whose functionality depends only on the order of each signals events and not on their exact timing. Designers can model the chip as a synchronous system whose modules communicate by exchanging signals on a set of point-to-point channels. The protocol guarantees that the system, if composed of functionally correct modules, behaves correctly, independent from the channel delays. Consequently, it is possible to automatically synthesize a hardware implementation of the system such that its functional behavior is robust with respect to large variations in communication latency. In practice, the channel implementation does not necessarily have a point-to-point structure. Still, the methodology orthogonalizes computation and communication because it separates the module design from the choice of the communication architecture, while enabling both *ad-hoc* wire pipelining and the automatic synthesis of each component interface logic. This separation is useful in several ways:

- it simplifies the system specification thanks to the synchronous hypothesis (designers can assume that intermodule communication takes one virtual clock cycle);
- it facilitates the fixing of design exceptions due to long global wires, because any number of special stateful repeaters (*relay stations*) can be inserted on them;
- it permits the exploration of tradeoffs in deriving the communication architecture up to the design process' late stages, because the protocol guarantees that the interface logic can absorb arbitrary latency variations.

While originally conceived to ease wire pipelining in single-clock chips, latency-insensitive design can be used also as a formal framework to develop tools for the automatic deploying of synchronous designs on distributed architectures, like GALS [7].

Considering the reports on the increasing usage of wire pipelining in high-performance microprocessor design [15] and the fact that high-end microprocessor designers have traditionally anticipated the challenges that ASIC designers are going to face in working with the next process generation, we think that latency-insensitive design represents a promising avenue for on-chip communication design with DSM technologies.

4. REFERENCES

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate versus ipc: The end of the road for conventional microarchitectures. In *27th Annual Intl. Symposium on Computer Architecture*, pages 248–259, June 2000.

[2] M. Bohr. Interconnect Scaling - The Real Limiter to High Performance ULSI. *IEEE International Electron Devices Meeting*, pages 241–244, Dec. 1995.

[3] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli. A methodology for "correct-by-construction" latency insensitive design. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 309–315. IEEE, Nov. 1999.

[4] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli. Latency insensitive protocols. In N. Halbwachs and D. Peled, editors, *Proc. of the 11th Intl. Conf. on Computer-Aided Verification*, volume 1633, pages 123–133, Trento, Italy, July 1999. Springer Verlag.

[5] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli. Theory of latency-insensitive design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1059–1076, Sept. 2001.

[6] L. P. Carloni and A. L. Sangiovanni-Vincentelli. Coping with latency in SOC design. *IEEE Micro*, 22(5):24–35, Sep-Oct 2002.

[7] L. P. Carloni and A. L. Sangiovanni-Vincentelli. A formal modeling framework for deploying synchronous designs on distributed architectures. In *FMGALS 2003: Formal Methods for Globally Asynchronous Locally Asynchronous Architecture*, Sept. 2003.

[8] P. Cocchini. Concurrent flip-flop and repeater insertion for high-performance integrated circuits. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 268–273, 2002.

[9] J. Cong and X. Yuan. Multilevel global placement with retiming. In *Proc. of the Intl. Symposium on Physical Design*, pages 208–213, June 2003.

[10] J. A. Davis, R. Venkatesan, A. Kaloyeros, M. Beylansky, S. Souri, K. Banerjee, K. Saraswat, A. Rahman, R. Reif, and J. Meindl. Interconnect Limits on Gigascale Integration (GSI) in the 21st Century. *Proc. of the IEEE*, 89(3):305–324, Mar. 2001.

[11] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. *Proc. of the IEEE*, 89(4):490–504, Apr. 2001.

[12] R. Lu, Z. G, C. Koh, and J. Chao. Flip-Flop and Repeater Insertion for Early Interconnect Planning. In *Proc. European Design and Test Conf.*, Mar. 2002.

[13] D. Matzke. Will Physical Scalability Sabotage Performance Gains? *IEEE Computer*, 8(9):37–39, Sept. 1997.

[14] J. D. Meindl. Interconnect opportunites for gigascale integration. *IEEE Micro*, 2003.

[15] E. Sprangle and D. Carmean. Increasing Processor Performance by Implementing Deeper Pipelines. In *29th Annual Intl. Symposium on Computer Architecture*, pages 25–36. IEEE, May 2002.