Asynchronous Circuits: An Increasingly Practical Design Solution

Peter A. Beerel
Fulcrum Microsystems
Calabasas Hills, CA 91301
and
Electrical Engineering, Systems Division
University of Southern California
Los Angeles, CA 90089
pabeerel@usc.edu

Abstract

While ultra-deep-submicron design presents increasingly difficult challenges for standard synchronous design practices, recent research in asynchronous design techniques is making asynchronous circuits an increasingly practical alternative. These challenges include the increasing pressure for low-power, the growing challenge of predicting increasing impact of wire load and delay, and the performance penalty associated with supporting communication between different clock domains. This paper reviews the different solutions to these problems that the spectrum of existing asynchronous design techniques support. It focuses on techniques for fine-grain two-dimensional pipelining that yield ultra-high-speed at nominal power supplies and very low-energy at reduced power supplies.

1. Introduction

The performance and power gap between semi-custom standard-cell-based and full-custom synchronous design styles is typically more than 2X and seems to be growing larger [1]. Standard-cell-based synchronous methodologies restrict the form of clocking, registers, and combinational logic to conform to the abilities of mature CAD tools. Moreover, the challenging task of estimating and accounting for relatively increasing wire delays in deep sub-micron technology is limiting performance and causing the so-called *timing closure* problem. In full-custom methodologies, this problem is addressed through carefully designed macro cells that have superior wiring and density than possible using semi-custom techniques. In particular, these macro cells may take advantage of advanced forms of clocking, registers, and combinational logic, including various forms of dynamic logic, thereby facilitating higher performance and lower power at the cost of 2-3X increase in design time [1].

This basic tradeoff between design time and quality of design also exists in asynchronous design with some significant differences [3]. Asynchronous circuits are typically decomposed into blocks that synchronize and communicate data with some form of request/acknowledge handshaking. Asynchronous methodologies differ in the size of these blocks, ranging from blocks similar in size to synchronous semi-custom pipeline stages (20 fanout-of-four, FO4, deep) to fine-grain pipelining where each block implements the logic associated with something as small as a 1-bit addition (2 FO4 deep) along with the necessary handshaking to asynchronously receive inputs and send outputs. The basic tradeoff within each of these asynchronous techniques is the tolerance to timing variations. The most robust form of asynchronous circuit design makes very little assumptions about the delay of gates and wires, i.e., they can have unbounded delay. The most aggressive form of circuit design uses timing assumptions both within and across blocks that must be verified pre and post layout.

This paper analyzes these tradeoffs in more detail, focusing on recent design styles of asynchronous two-dimensional fine-grain pipelining. We assert that achieving high-performance in robust asynchronous circuits provides throughputs comparable to full-custom design with design-time that can be comparable to semi-custom efforts. When combined with the natural integration of fast and slow components, the optimal clock gating that arises from handshaking-based communication, and voltage scaling, the circuits can also yield lower energy consumption for a given performance.

2. Synchronous design styles: semi-custom versus full-custom

Semi-custom standard-cell-based design methodologies offer good performance with typically 12-month design times [1]. They are supported by a large array of mature

Proceedings of the International Symposium on Quality Electronic Design (ISQED'02)
0-7695-1561-4/02 $17.00 © 2002 IEEE

CAD tools that range from simulation, synthesis, verification, and test. A large library of standard-cell components that have been carefully designed, verified, and characterized supports the synthesis task. This library is generally limited to static CMOS gates because they are robust to different environmental loads and have high noise margins, thus requiring little block-level analog verification. Standard-cell designs also use standard clocking and limited gated-clocking strategies to facilitate automation and reduce design times, reducing the options for saving power. In addition, standard flip-flops are often used to simplify timing analysis despite the incurrence of significant data to clock output overheads.

Moreover, the time-to-market advantage of standard-cell based designs is being attacked by the increasingly difficult task of estimating wire delay. In deep-submicron design wire-delays are relatively increasing and the traditional separation of front and backend design tasks is breaking down because logic and gate design choices do not account for the amount of wire delay actually incurred. This timing-closure problem has forced numerous shipment schedules to slip. EDA vendors have now developed a new suite of emerging CAD tools that address aspects of physical design much earlier in the design process [26][27].

Full-custom design houses, however, have found that these challenges can be overcome with more manual design effort that incur longer design cycles of an average of 36 months. In addition, the use of advanced dynamic logic styles has been an area of growing interest in full-custom designs [10][22][23]. Domino logic is estimated to be 30% faster than static logic because of the improving logical effort derived by the removal of PMOS logic [24]. Traditional domino logic however still suffers from overhead associated with clock skew and latch delays. More advanced flip-flops and latches have been developed that somewhat improve the clock skew overhead and reduce the latch delays [25]. At the extreme, the latches and thus their overhead can be entirely removed using multiple overlapping clocks in a widely used technique, recently named skew-tolerant domino logic [23].

The basic cost of this higher performance is the reduced noise margin and the increased need for extensive analog verification, both pre and post layout. In particular, dynamic logic suffers from charge-sharing problems that require more manual and extensive analog simulation. In addition, many advanced forms of dynamic logic, have very aggressive timing assumptions that must be accounted for during the entire design process and verified both pre and post layout. These timing assumptions are particularly difficult to verify because they involve two-sided non-local constraints [10][36]. For example, in self-resetting domino logic, the precharge signal is a pulse that has a two-sided timing constraint. The pulse must be sufficiently long to allow the circuit sufficient time to precharge. The pulse, however, cannot be too long, however, because new data may arrive at the inputs of the dynamic gate while the precharge signal is still asserted, causing significant short-circuit current.

3. Asynchronous channel-based design

Many asynchronous designs are composed of blocks synchronizing and communicating data using handshaking via *channels*. These channels are a bundle of wires upon which a protocol controls the communication of data, also called a "token". Numerous forms of channels have been developed that trade off robustness to timing variations and power/performance. Bundled-data channels communicate with a single request-line bundled with a unidirectional data bus coupled with an acknowledgement wire, as illustrated in Figure 1a. These channels are area and power efficient but incur a timing assumption every place they are used. Alternatively, data can be sent with 1-of-N channels that use N data wires to send $\log_2 N$ bits of data, as illustrated in Figure 1b. The most well known form of this channel is dual-rail that uses two data wires per bit of data. These facilitate delay-insensitive communication between blocks, reducing the amount of timing verification required. Note that 1-of-4 channels provide 2-bits of data by changing only 1 wire, yielding lower power than conventional dual rail channels [8]. In addition, single-track 1-of-N channels are also possible in which the sender drives the wire in one direction whereas the receiver acknowledges the wire by resetting it, as illustrated in Figure1c. The handshaking protocols across these channels are either 2-phase, 4-phase, or a mixture.

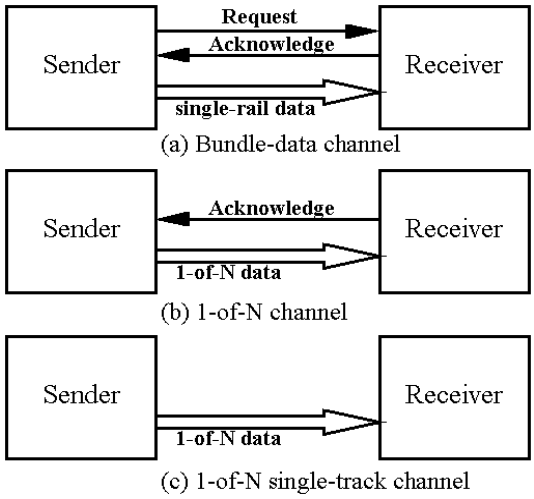

Figure 1. Three types of asynchronous channels.

An asynchronous block is a combination of datapath and control circuitry that may have multiple input channels to receive tokens and multiple output channels to send tokens. The operation of the block is to read a subset of input tokens, process the data, and send output tokens. In all

cases, the handshaking protocol guarantees that input tokens are processed only when valid data exists and output tokens are sent only when the output channels have reset. In linear pipelines, such as simple FIFOs, each pipeline stage, or block, has a single input and output channel. In real systems, however, non-linear pipelines are common, involving blocks with multiple input and output channels. In many cases, tokens on input channels are conditionally read based on the value of another input channel and output channels are conditionally written to based on the results of the computation. For example, a 2-way split block has two input channels, one for data and one for control, and two output channels. This block waits for tokens on both the (binary) control and data channels and the output channels to reset. Then, based on the binary value of the control token, it sends the data token to the corresponding output channel and then repeats [12].

As mentioned earlier, the size and implementation design style of asynchronous blocks varies greatly among different design styles and this section reviews several different varieties.

3.1 Globally asynchronous locally synchronous (GALS) design

In globally asynchronous, locally synchronous the blocks are as large as synchronous design techniques can efficiently handle. A standardized channel interfaces allows synchronous blocks to communicate despite vast differences in internal frequency and design styles. This is particularly important because the Semiconductor Industry Association estimates that long-range wires may have a delay that is equivalent to between 5 and 20 clock cycles [1]. Either communication will need to operate at reduced frequencies or techniques to pipeline the communication must be adopted [29]. Because the latency of the communication will be particularly difficult to estimate before layout, latency-insensitive design techniques have gained significant interest [28]. Because the blocks are latency-insensitive, it is very easy to add additional storage buffers late in the design cycle to pipeline long-range channels, thereby increasing communication frequency with little to no latency overhead. A perceived advantage of GALS is that asynchronous circuits are limited to the periphery of blocks where careful design libraries and interface design techniques can be employed. One problem with these interface techniques, however, is that they require synchronizers which themselves incur a significant latency penalty.

3.2 Micropipelines

In micropipelines, first introduced in [21], the size of the communicating block is typically similar to that of a standard synchronous pipeline stage (~20 FO4 latency). The block consists of control and datapath logic. The datapath logic is typically designed using standard-cell static CMOS logic, enabling the full use of mature synchronous synthesis techniques. The control logic implements the request/acknowledge control circuitry that facilitates communication across the bundled-data channels between blocks. The tools Balsa and Tangram automatically generate the control logic using syntax translation from a high-level language [7][20]. The control logic can also be synthesized using Petri-net or FSM based specifications [9][17][18] or derived from recently developed bundled-data control templates [32]. In all cases, the control logic requires the use of a delay line that matches the longest delay in the datapath to guarantee that the control circuit generates requests only after data is valid.

Because the pipeline stages, or blocks, of a micropipeline are similar in size to that of synchronous pipelines, their worst-case performance is comparable to that of synchronous blocks. In fact, one of the big challenges in micropipeline design is to ensure that the matched delay line and control circuitry is designed carefully not to incur large control overhead. Careful timing verification is required to ensure that the delay line is long enough to provide sufficient timing margin. Asynchronous architectures, however, have the advantage that only blocks that are sent tokens are active and only those blocks actively processing tokens dictate the throughput of the system. This enables designers to obtain high average-case performance by focusing their efforts only on the most commonly activated blocks. In addition, advanced delay lines using speculative completion sensing can be used to provide some data-dependent delay, leading to improved average-case performance [11].

Micropipeline-based asynchronous designs have several other advantages. Compared to dual-rail approaches, they offer low area because they use single-rail datapath and channels. They offer low power because they remove the power-hungry clock distribution network and instead consist of blocks that consume power only when actively processing tokens. That is, micropipelines facilitate perfect gated-clocking at a relatively fine granularity [2][6]. Compared to synchronous designs, they also offer lower electromagnetic interference because the current used is more evenly spread over time [3]. This can be very important when considering mixing analog and digital circuits on a single substrate. Lastly, pipelining long-range communication channels late in the design cycle is relatively easy and does not incur significant latency overheads [8], mitigating the timing closure problem.

3.3 Fine-grain pipelining

A number of design styles targets higher performance by using much smaller communicating blocks. Perhaps the most aggressive design style is that proposed by Sun researchers called GasP [30]. GasP is a very aggressive fine-grain pipeline template style that uses a combination of single-track channels for communicating control and single-rail channels for communicating data. The datapath blocks are decomposed into fine-grain pipeline stages, or blocks, such that each block has a latency of 2-4 FO4 delays. The scheme supports multi-gigahertz rates but has complex two-sided timing assumptions in both the control and datapath that must be accounted during all stages of the design.

At the other extreme, the most robust form of asynchronous circuit design supports delay-insensitive communication between blocks meaning that communication will operate correctly independent of wire delay along the channel, reducing the dependence on timing analysis of long-range communication. Within a block, the design can be quasi-delay-insensitive (QDI) [15], meaning the circuit will operate correctly independent of gate delays but that some internal wire forks can be *isochronic*, meaning that the difference in arrival times of data at the different ends of the fork must be negligible. The communication between blocks is encoded using 1-of-N rail encoding. This facilitates the use of robust completion detection (CD) units, instead of matched delay lines, to synchronize communication of data across channels [14]. For example, to determine when a 1-of-2 dual-rail channel has valid data the data rails are fed into a simple OR gate. When applied to wide datapaths, CD units naturally yield blocks that have data-dependent delays. Unfortunately, at the same time incur large control overhead that can reduce even average performance [16].

Recently, Caltech researchers have proposed a solution to this problem that is referred to in this paper as two-dimensional fine-grain pipelining [5][12][13]. The basic idea is that the blocks have both small latency (~2 FO4 delays) and operate only a small number of bits (e.g., 4 bits). Completion operation of wide datapaths is simply pipelined across several communicating blocks and performed in parallel with subsequent datapath operations. In this way, the QDI assumption within a block is easier to meet and the completion detection overhead is relatively small. This means that high-performance can be achieved without sacrificing robustness to timing.

At USC, we noticed that by introducing timing assumptions to 2-D pipelines, even higher throughputs could be achieved. In fact, there is a tradeoff between the robustness to delay variations and performance. We recently have developed several different families of 2-D fine-grain pipeline templates that explore this tradeoff. In [33], we extend the use of single-track handshaking proposed in GasP to 1-of-N signaling of both data and control of fine-grain 2-D pipeline blocks. We describe single-track full-buffer (STFB) templates that define how arbitrary linear and non-linear pipelines can be generated with very high performance. In conjunction with researchers from Columbia University, we also have generalized high-speed linear pipeline structures [31] to non-linear pipelines with more tolerance to timing variations than our single-track designs [34]. Lastly, we developed new very robust pipeline templates that, by slightly relaxing the delay-insensitive channel restriction, achieves significant improvement in throughput and reduction in area over their Caltech QDI counterparts [35]. HSPICE simulations in 0.25u TSMC process with a 2.5V power supply at 25^{o}C for simple linear pipelines are given in Table 1 and illustrate the tradeoff between speed and robustness to timing.

Table 1: Performance vs. Robustness to Timing

2-D Template Style	Timing assumptions	Throughput
PCHB [12]	DI/QDI	772 MHz
RSPCHB [35]	QDI	920 MHz
LP2/2+ [34]	Moderate	1.0 MHz
STFB [33]	Aggressive	1.6 GHz

Compared to typical synchronous semi-custom flows these circuits yield performance that is 2-5X times higher, and are comparable to the most aggressive full-custom design styles. The key points to stress is that this design styles removes the need for complex clock distribution and in some cases greatly simplifies the timing closure problem. In particular, only those blocks deemed critical can be designed aggressively and this decision can be largely independent of the design style for less critical blocks. This is in sharp contrast to synchronous design where slow paths, even if rarely activated, need to be sped-up to meet timing.

An important additional advantage of these forms of 2-D template is their low latency. Unlike typical synchronous pipeline stages, they have no distinct register that stores pipeline data and thus do not suffer from latch propagation delay or setup and hold time overheads. Rather, the data is stored at the output of the dynamic logic using a weak staticizer that only negligibly impacts delay. When designed properly, the latency through an asynchronous system approximates the optimal latency through dynamic logic, similar to the most aggressive forms of skew-tolerant domino logic [23]. The key to this advantage is the use of dynamic logic and thus, as with many full-custom synchronous design techniques, a common need is CAD tools to analyze and solve charge sharing.

One potential cost of 2-D pipelining is increased area: each block has some form of completion detection unit and control circuitry to support handshaking with neighboring

block. Moreover, communication between two blocks uses dual-rail or 1-of-N signaling requiring significant more wiring bandwidth. However, there is no global clock to route and all storage elements are implemented through area-efficient staticized dynamic nodes. One recent aggressive asynchronous design achieved a factor of 3X performance improvement at the cost of less than a 20% increase in area [4].

Given the rate of technology scaling, the cost in area may not be as critical as their relative power. The increased control circuitry also can lead to higher energy consumption; however, two design features mitigate this point. First, in addition to removing the power-hungry global clock, 2-D fine-grain pipelining facilitates the ability to turn off specific bit-slices rather than complete functional units. One recent aggressive asynchronous design achieved 40% lower power than its synchronous counterpart, at the same voltage [4]. In addition, the high-throughput can yield low energy by reducing the power supply. In fact, given that energy is a function of the voltage squared, high-speed asynchronous designs may yield very low energy consumption for a given performance.

4. Conclusions

The introduction of 2-D pipelining in asynchronous circuits has demonstrated that high-performance can be achieved with design techniques that are very robust to timing, greatly simplifying timing analysis. Moreover, rather than having to ensure all paths through all blocks meet timing as in synchronous design, only those blocks deemed critical need be optimized, greatly simplifying the timing closure problem. For those blocks for which higher performance is needed, we reviewed a range of more aggressive pipeline templates that facilitate higher performance at the cost of increased design time to validate moderate to aggressive timing assumptions. Moreover, we argued that in many applications this high performance may be converted to low energy consumption for a given performance by reducing the power supply.

5. References

[1] International Technology Roadmap for Semiconductors: 1999 edition. http://public.itrs.net/files/1999_SIA_Roadmap/Home.htm.

[2] J. Kessels and P. Marston. Designing asynchronous standby circuits for a low-power pager. In *Proceedings of the IEEE*, 87(2): 257–267, Feb. 1999.

[3] C. H van Berkel, M. B. Josephs, S. M. Nowick. Scanning the technology: applications of asynchronous circuits. *Proceedings of the IEEE*, vol. 87:2, pp. 223-233, Feb. 1999.

[4] K. Stevens, S. Rotem, R. Ginosar, P. A. Beerel, C. J. Myers, K. Yun, R. Kol, C. Dike, and M. Roncken, An asynchronous instruction length decoder. In *IEEE Journal of Solid State Circuits*, 36(2): 217-228, Feb. 2001.

[5] A. J. Martin, A. Lines, R. Manohar, M. Nystroem, P. Penzes, R. Southworth, and U. Cummings. The design of an asynchronous MIPS R3000 microprocessor. In *Advanced Research in VLSI*, pp. 164–181, Sept. 1997.

[6] S.B. Furber, D.A. Edwards, and J.D. Garside. AMULET3: a 100 MIPS asynchronous embedded processor. In *Proc. International Conf. Computer Design (ICCD)*, Sept. 2000.

[7] A. Bardsley, D. A. Edwards. The Balsa Asynchronous Circuit Synthesis System. *FDL 2000*, Sept. 2000.

[8] W.J. Bainbridge, S. Furber Delay-insensitive system-on-chip interconnect using 1-of-4 data encoding. In Proc. ASYNC 2001, pp. 118—126, Mar. 2001.

[9] Chris J. Myers, Asynchronous Circuit Design, John Wiley and Sons, July 2001.

[10] W. Belluomini, C. J. Myers, H. Peter Hofstee. Verification of delayed-reset domino circuits using ATACS, In *Proc. of ASYNC*, pp 3 –12, 1998

[11] S.M. Nowick, K.Y. Yun, and P.A. Beerel. Speculative completion for the design of high-performance asynchronous dynamic adders. In *Proc. ASYNC*, pp. 210–223, Apr. 1997.

[12] Andrew M. Lines. Pipelined asynchronous circuits. Master's thesis, California Institute of Technology, 1996.

[13] U. Cummings, A. Lines, and A. Martin. An Asynchronous Pipeline Lattice Filter, In *Proc. of ASYNC*, pp 126-133, Nov. 1994.

[14] Alain J. Martin. Synthesis of asynchronous VLSI circuits. In J. Straunstrup, editor, *Formal Methods for VLSI Design*, chapter 6, pp. 237–283, 1990.

[15] Alain J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In William J. Dally, editor, *Advanced Research in VLSI*, pages 263–278. MIT Press, 1990.

[16] T. Nanya, A. Takamura, M. Kuwako, M. Imai, T. Fujii, M. Ozawa, I. Fukasaku, Y. Ueno, F. Okamoto, H. Fujimoto, O. Fujita, M. Yamashina, and M. Fukuma. TITAC-2: A 32-bit scalable-delay-insensitive microprocessor. In *Symposium Record of HOT Chips IX*, pages 19–32, Aug. 1997.

[17] M. Theobald and S.M. Nowick. Fast heuristic and exact algorithms for two-level hazard-free logic minimization, *IEEE Transactions on CAD*, vol. 17:11, pp. 1130-1147, Nov. 1998.

[18] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Trans. on Information and Systems*, Vol. E80-D, No. 3, pp. 315-325, Mar. 1997.

[19] M.A. Peña, J. Cortadella, A. Kondratyev and E. Pastor, Formal verification of safety properties in timed circuits, In *Proc. of ASYNC*, pp. 2-11, Apr. 2000.

[20] J. Kessels, A. Peeters The Tangram framework: asynchronous circuits for low power. *Proceedings of ASP-DAC*, pp. 255–260, 2001.

[21] I. E. Sutherland, Micropipelines. *Communications of the ACM*, vol. 32, no. 6, pp. 720-738, June 1989.

[22] S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immediato, and K. Jenkins. Asynchronous interlocked pipelined CMOS circuits operating at 3.3-4.5 GHz. In *IEEE ISSCC Digest of Technical Papers*, pp. 292–293, 2001.

[23] D. Harris and M. Horowitz. Skew-tolerant domino circuits. *IEEE Journal of Solid-State Circuits*, pp. 1702-1710, Nov. 1997.

[24] I. Sutherland, B. Sproull, D. Harris, "*Logical Effort: Designing Fast CMOS Circuits*", Morgan Kaufmann Publishers, San Francisco, CA: 1999.

[25] V. Stojanovic and V. G. Oklobdzija. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. *IEEE Journal of Solid-State Circuits*, 34(4):536--548, April 1999**.**

[26] http://www.synopsys.com/products/unified_synthesis/unified_synthesis_ds.pdf.

[27] http://www.cadence.com/products/pks.html.

[28] L.P. Carloni, K.L. McMillan and A.L. Sangiovanni-Vincentelli. Theory of latency-insensitive design. *IEEE Transactions on CAD*. Vol. 20, No. 9, Sept. 2001.

[29] T. Chelcea and S.M. Nowick. Robust Interfaces for Mixed-Timing Systems with Application to Latency-Insensitive Protocols**,** *IEEE/ACM Design Automation Conference (DAC)*, June 2001.

[30] I.E. Sutherland, and S. Fairbanks. GasP: a minimal FIFO control. In *Proc. of ASYNC*, pp. 46–53, March 2001.

[31] M. Singh and S.M. Nowick. High-throughput asynchronous pipelines for fine grain dynamic datapaths. In *Proc. of ASYNC*, pp. 198–209. 2000.

[32] S. Tugsinavisut and P.A. Beerel. Control circuit templates for asynchronous bundled-data pipelines. To appear in *DATE'2002.*

[33] M. Ferretti and P.A. Beerel. Single-track asynchronous pipeline templates using 1-of-N encoding. To appear in *DATE'2002*.

[34] R.O. Ozdag, M. Singh, S. Nowick, and P. A. Beerel. High-speed non-linear asynchronous pipelines. To appear in *DATE'2002*.

[35] R.O. Ozdag and P.A. Beerel. High-speed QDI asynchronous pipelines. To appear in *ASYNC'2002*.

[36] H. Kim, K. Stevens, P.A. Beerel. Relative Timing Based Verification of Timed Circuits and Systems. To appear in *ASYNC'2002*.