1 Introduction

Innovations in mobile communication, e-commerce, entertainment, and medicine all have roots in advances in semiconductor processing, which continually reduce the minimum feature size of transistors and wires, the basic building blocks of chips. This continual reduction supports ever-increasing numbers of transistors on a single chip, enabling them to perform increasingly sophisticated tasks. In addition, smaller transistors and wires have less resistance and capacitance, enabling both the higher performance and lower power that the integrated circuit market continually demands.

These manufacturing advances, however, also change the design challenges faced by circuit designers and the computer-aided-design (CAD) tools that support their design efforts. Beginning in the 1980s, wire resistance became an important factor to consider in performance and is now also important in analyzing voltage drops in power grids and long wires. Starting in the 1990s, higher mutual capacitance exacerbated the impact of cross-talk, which is now addressed by a new range of timing and noise analysis tools. And today, as the transistor's feature size approaches fundamental atomic limits, transistors act less like ideal switches and wires act less like ideal electrical connections. In addition, the increased variations both within a single chip and between chips can be substantial, making precise estimates of their timing and power characteristics virtually impossible [1]. Consequently, modern CAD tools must conservatively account for these new non-ideal transistor characteristics as well as their variability.

As part of this ever-changing technological backdrop, the relative merits of different circuit design styles change. The predominant circuit design style is synchronous design with complementary metal-oxide–semiconductor (CMOS) static logic gates and a global clock to regulate state changes. However, as process variability increases and the challenges of routing a global clock across a large chip become increasingly problematic, radically different design styles such as asynchronous design have become an increasingly interesting alternative. In its most general form, asynchronous design removes the global clock in favor of distributed local handshaking to control data transfer and changes of state. While academic research in this area can be traced back to the 1950s [2], it has taken until the late 1990s and 2000s for this technology to mature. Several start-up companies have begun to commercialize asynchronous design as a competitive advantage for

a wide variety of applications [45][46][47][48]. However, the mass application of asynchronous design has been an elusive goal for academic researchers and, while recent advances are promising, only time will tell whether this technology will take a larger foothold in the very-large-scale integration (VLSI) world.

There are many different types of integrated circuits (ICs) and the design style choice for a particular application depends on the relative performance, power, volume, and other market demands of the device. For example, traditionally, low-volume specialized products with only moderate power and performance requirements can use field programmable gate arrays (FPGAs), which provide reduced time to market and low design risk, primarily because of their reprogrammable nature. Higher-volume products with more aggressive power and performance requirements often require application specific integrated circuits (ASICs), which come at the cost of the increased design and verification effort associated with the finalizing of the manufacturing process.

Products that require significant programmability may also contain some type of microprocessor to enable software support. Products which require significant storage will contain large banks of on-chip memories. Both micro-processors and memory blocks are available on modern FPGAs and can be integrated into an ASIC in the form of intellectual property cores. In addition, dedicated chips for memory are critical in complex system design and can store billions of bits of data either in volatile or non-volatile forms.

Chips with high volumes, such as microprocessors, memory chips, and FPGAs, may be able to support *full-custom* techniques with advanced circuit styles, such as asynchronous design. In fact, asynchronous techniques have been used in memory for years and a recent start-up is the commercializing of high-speed FPGAs, which has been enabled by high-speed asynchronous circuits [42][43][48].

Most ASICs, however, rely on *semi-custom* techniques in which more constrained design styles are used. The relative simplicity of the constrained design style enables the development of CAD tools that automate large portions of the design process, significantly reducing design time. For asynchronous design to be adopted for ASICs, existing CAD tool suites must be enhanced with scripts and new tools to support asynchronous circuits. This chapter provides an overview of the general issues that guide this design choice. In doing so, it identifies the potential advantages of asynchronous design and the remaining challenges for its widespread adoption.

1.1 Synchronous design basics

Synchronous design has been the dominant methodology since the 1960s. In synchronous design, the system consists of sub-systems controlled by one or more clocks that synchronize tasks and the communication between blocks. In traditional semi-custom synchronous ASIC flows, combinational logic is placed in between banks of flip-flops (FFs) that store the data, as shown in Figure 1.1.

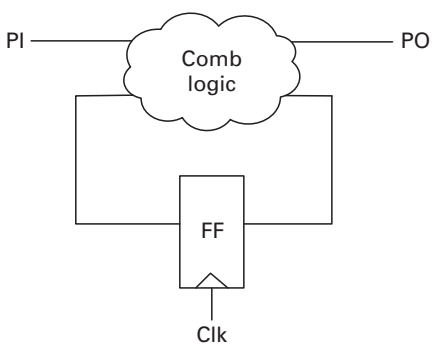

Figure 1.1. Traditional synchronous ASIC, showing the combinational logic and flip-flop. The primary inputs and outputs are denoted PI and PO.

The combinational block must complete its operation within one clock cycle under all possible input combinations, from all reachable states, in the worst-case operating environment. In this way, the clock ensures synchronization among combinational blocks by guaranteeing that the output of every combinational block is valid and ready to be stored before the next clock period begins. In fact, the data at the inputs of the FFs may exhibit glitches or hazards, as long as they are guaranteed to settle before the sampling clock edge arrives. In order to guarantee that the data is stable when sampled, the clock period should account for the worst-case delay including clock skew and all process variations.

Typical semi-custom design flows use a fixed set of library cells that have been carefully designed, verified, and characterized to support synthesis, placement, routing, and post-layout verification tasks. This library is generally limited to static CMOS gates, which, compared with more advanced dynamic logic families, have higher noise margins and thus require far less analog verification. The cells have accurate table-based characterization to support *static* timing and noise analysis rather than more computationally intensive analog simulation. In particular, timing constraints are reduced to the setup and hold times on FFs, which static timing analysis tools can verify reliably with minimal user input. This constrained methodology has facilitated the development of mature suites of CAD tools that yield relatively short, 12-month, design times.

Full-custom flows use more labor-intensive advanced design styles and less automated tools to obtain higher performance and lower power. In particular, full-custom ICs have clock frequencies that are three to eight times higher than those for semi-custom flows, owing, to their advanced architectures, micro-architectures, circuit styles, and manufacturing processes [4]. The differences include: the use of advanced pipelining, clock-tree design, registers, dynamic logic, and time borrowing; more careful logic design, cell design, wire sizing, floor-planning, placement, and management of wires; the better management of process variation; and, finally, accessibility to faster manufacturing processes.

In particular, full-custom designs can use a variety of forms of dynamic logic which offer significant improvements in performance and power consumption [1] [3][6]–[8]. For example, the application of dynamic logic in the IBM 1.0 GHz design yields 50% to 100% higher performance compared with its static logic equivalent [8]. In addition, advanced FFs and latches can reduce the overhead associated with clock skew and latch delays [7]. Moreover, latch delays can even be removed by means of multiple overlapping clocks and dynamic logic, in a widely used technique recently named skew-tolerant domino logic [6].

However, dynamic logic has lower noise margins than its static counterpart. Dynamic logic is also more difficult to characterize using table-based methods, and static timing analysis tools tend to have less accurate results for such circuits [3]. Consequently, more careful analog-level noise and timing verification is required. Managing this verification, along with the other more manual aspects of full-custom design, typically results in a significant increase in design time.

1.2 Challenges in synchronous design

Synchronous design has been the predominant design methodology largely because of the simplicity and efficiency provided by the global clock. The registers decompose the design into acyclic islands of combinational logic which facilitate efficient design, synthesis, and analysis algorithms. However, the global nature of the clock also leads to increasing design and automation challenges. In particular, the time-to-market advantage of standard-cell-based ASIC designs is being subverted by the increasingly difficult design challenges posed by modern semiconductor processes [4]. These challenges affect both high-performance and low-power ASICs, as described below.

1.2.1 Computer-aided design for high-performance

In earlier submicron designs, architecture, logic, and technology-mapping design could proceed before and somewhat independently from the placement and routing of the cells, power grid, and clocks because wire delays were negligible compared with gate delays. In deep-submicron design, however, interconnect has not scaled to the same degree as gates, and cross-talk between wires leads to substantial changes in wire delay. Consequently, wire delay increasingly accounts for a larger fraction of the critical path. In particular, the delays of long-range wires may account for up to 70% of the critical path of some high-performance designs [36]. This change in relative importance has caused the traditional separation of logic synthesis and physical design tasks to break down, because synthesis cannot now properly account for the actual wire delays and other geometrical effects. Since the late 1990s, this timing-closure problem and disconnect between synthesis and physical design has forced numerous shipment schedules to slip.

As a consequence, tighter integration of these CAD tools has been developed and a wide range of post-placement optimization, including gate resizing, buffer insertion, and wire width optimization, have become an essential part of CAD tool suites. In addition, sophisticated cross-talk analysis techniques that account for the switching window of various wires have been developed to determine the impact in delay associated with increasingly large cross-coupling capacitances. Understanding the impact of cross-talk during post-placement optimization is computationally challenging owing to the inherent feedback between accurate cross-talk analysis and post-placement optimizations. For example, switching windows can shift dramatically owing to buffer insertion, resulting in significant changes in the delay of neighboring wires. This computational challenge forces CAD tools to rely on approximations that yield non-optimal designs.

Despite the continued advances of modern CAD tools, the overall impact of these challenges has been an increasingly large performance gap between full-custom integrated circuits (ICs) and semi-custom ASICs. Moreover, it is predicted that high-performance semi-custom designs will remain three times slower than their full-custom counterparts despite all the potential advances in CAD tools [4]. In particular, conventional wisdom suggests that semi-custom CAD tools may never support dynamic logic because of the added complexity of noise and timing constraints and the lack of dynamic cell libraries.

1.2.2 Computer-aided design for low-power devices

As manufacturing feature sizes have decreased, transistors have become increasingly leaky and power budgets have become increasingly difficult to meet. This has motivated a range of improvements to the low-power ASIC flow. In particular, low-leakage power-efficient cell design, multiple supply voltages on a single die, gated power supplies, and more advanced clock-gating techniques are continually being developed and incorporated in ASIC flows. In addition, a few full-custom techniques for low power have also been explored and are just emerging in ASIC tools; these techniques include low-voltage swings for long-range wires. However, these low-swing circuits have reduced noise margins, which necessitates careful wire planning, shielding, and some analog verification. Moreover, it is well known that low-power latches can be used instead of flip-flops to reduce clock-tree capacitance.

1.3 Asynchronous design basics

As a result of the increasing limitations and growing complexity of semi-custom synchronous design, asynchronous circuits are gaining in interest. In the absence of a global clock that controls register and state updating, asynchronous designs rely on handshaking to transfer data between functional blocks. One common way

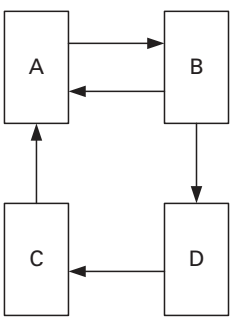

Figure 1.2. Asynchronous blocks communicating using channels.

to design asynchronous circuits is to organize the data transfer in *channels* that group a bundle of data wires using handshaking signals. The channels are unidirectional and typically point-to-point; they are represented as arrows in Figure 1.2. Notice that the bi-directional communication of data between blocks A and B requires two channels in opposite directions.

Many different design styles have been proposed for asynchronous circuits. They differ in the method in which data is encoded in channels, the handshaking protocol, and the number and type of timing assumptions required for the designs to work properly. These different design tradeoffs make it difficult to make general statements regarding the relative merits of the various benefits typically associated with asynchronous design. In particular, some design styles yield low power but are best suited for low-performance applications. Others yield high performance at the expense of more power and additional timing assumptions. All, however, provide a rigorous framework for exploring alternatives to synchronous design.

1.4 Asynchronous design flows

There are many different possible design flows for asynchronous circuits. Here we describe three, to demonstrate the diversity of flows being explored.

The first of these design flows is called *refinement* and involves the decomposition of asynchronous blocks into a hierarchical network of *leaf cells*, where a leaf cell is the smallest block that communicates with its neighbors via channels. Each leaf cell is typically implemented with a small set of transistors, typically between 10 and 100. Early attempts to automate this process are encouraging, but as of today industry relies on significant manual effort and a large re-usable library of macro cells (functional blocks, registers, and crossbars) and leaf cells. This technique was pioneered by Alain Martin at Caltech and has been applied to several asynchronous microprocessors [9] and digital-signal processing chips [25]. It has been commercialized by Fulcrum Microsystems and has led

to circuit designs whose performance exceeds that available through semi-custom standard-cell design flows (see e.g. [9]).

The second design flow involves re-using synchronous synthesis tools and translating a synchronous gate-level netlist (which conveys connectivity information) into an equivalent asynchronous netlist. This involves removing the global clock and replacing it with local handshaking circuitry. This flow has the benefits of reducing the barrier of adoption for asynchronous design and employing the power of synchronous synthesis tools. This approach was initially proposed by a start-up company called Theseus Logic [28] and subsequent results are encouraging [26]–[32]. Nevertheless, more work is necessary for it to produce circuits that compete with manual decomposition.

The third design flow is based on syntax-directed translation from a high-level language that includes handshaking primitives such as *sends* and *receives* [15]–[17]. Syntax-directed translation makes the high-level estimation of power and performance characteristics computationally efficient and enables designers to control the resulting circuits more directly by altering the high-level language specification. The results of the translation are typically far from optimal, and forms of peep-hole optimization at gate level are needed to improve efficiency. This technique was pioneered by several researchers at Phillips Research and is now being commercialized by Handshake Solutions. It has produced very-low-power designs from a digital compact cassette (DCC) error detector [15], an 80C51 micro-controller [14], and an Advanced RISC Machines (ARM) micro-processor [44] that compare quite favorably with their synchronous counterparts.

There are also significant differences in back-end flows for these design flows. Some rely on using synchronous standard-cell libraries while others rely on the advantage of using non-standard gates such as C-elements and domino logic. In both cases, however, commercial place and route flows are being explored to automate the physical design.

In addition, commercial static timing and power analysis tools are used to verify timing assumptions pre- and post-layout, and synchronous test tools are being adopted for both automated test generation and test coverage analysis. In all these cases, the presence of combinational cycles in asynchronous circuits is a distinguishing feature that often stretches the capabilities of these tools and requires novel approaches.

1.5 Potential advantages of asynchronous design

Asynchronous circuits have demonstrated potential benefits in many aspects of system design (e.g. [9][18]–[23]). Their advantages include improvements in high-performance, low-power, ease of use and reduced electromagnetic interference (EMI) but these are not universally applicable. Some advantages may be application specific and dependent on the particular asynchronous circuit design style. Others depend on whether the comparison is made with semi-custom or full-custom

synchronous design or, more generally, the level of effort put into the comparable synchronous design.

1.5.1 High performance

Performance can be measured in terms of system latency or throughput or a combination of the two. The potential for high performance in asynchronous design stems from a number of factors. Several of these are inherent in any design lacking a global clock and are independent of the asynchronous design style. They include the following:

- *Absence of clock skew* Clock skew is defined as the arrival time difference of the clock signal to different parts of the circuit. In traditional standard-cell design, the clock period may need to be increased to ensure correct operation in the presence of clock skew, yielding slower circuits. In recent design flows, however, a portion of this skew is regarded as *useful skew* and is accounted for during logic synthesis, thus mitigating the impact on the feasible clock frequency. Moreover, in full-custom design, more sophisticated clock-tree analysis and design reduce this effect further. Nevertheless, as process variations increase, the clock-skew impact on clock frequencies is likely to grow.
- *Average-case performance* Synchronous circuit designers have to consider the worst-case scenario when setting the clock speed to ensure that all the data has stabilized before being sampled. However, many asynchronous designs, including those that use static logic, can have average-case delay due to a data-dependent data flow and/or functional units that exhibit data-dependent delay [37]. In both cases, the average-case delay may be less than the synchronous worst-case delay.

Other performance advantages are specific to different sub-classes of asynchronous designs and include the following:

- *Application of domino logic* As mentioned earlier, domino logic is often used in high-performance full-custom synchronous designs because its logical effort is lower than that required for static logic [6]. Domino logic is limited to full-custom design flows because of its reduced noise margin and because its timing assumptions are not currently supported by semi-custom design tools. Some asynchronous design flows embed domino logic within a *pipeline template* [22] [25]. Instead of a clock that controls the precharge and evaluate transistors, distinct asynchronous control signals are used. Recent research advances are aimed at determining whether these templates can be designed within a standard-cell flow for asynchronous design [33]–[36]. Compared with current ASIC synchronous flows, they have the potential performance advantages of dynamic logic as well as removal of the latency overhead and setup margins associated with explicit flip-flops and latches.
- *Automatic adaptation to physical properties* The delay on a path may change owing to variations in the fabrication process, temperature, or power supply

voltage. Synchronous system designers must consider the worst case and set the clock period accordingly. Many asynchronous circuits can adapt robustly to changing conditions, yielding improved performance [22][24]. This is far more difficult to achieve in a synchronous design, as the variations can be local whereas the impact on the clock is far more global.

- *At-speed testing* Asynchronous design styles that embed data and validity into the same wires provide a potential additional performance advantage. In principle, it is possible to add logic to verify whether the asynchronous data arrives before a fixed clock, both during performance testing and/or on-line. This enables mixed asynchronous–synchronous designs which avoid the large margins associated with synchronous application-specific ICs, for which performance testing would not be affordable.

1.5.2 Low power

The constant activity of a global clock causes synchronous systems to consume power even though some parts of the circuit may not be processing any data. Even though clock gating can avoid the sending of the clock signal to the un-active blocks, the clock driver still has to constantly provide a powerful clock that reaches all parts of the circuit. The removal of the global clock in favor of power-efficient control circuits can sometimes lead to significant power savings. Moreover, asynchronous architectures can reduce power consumption by minimizing data movement to only where and when it is needed.

The event-driven nature of asynchronous design leads to circuits with low standby power, which provides a significant advantage in mobile applications that must react to external stimuli (e.g. smart cards [9] and pagers [12]). The alternative in synchronous design would be a standby-power-inefficient circuit that continually polls external signals.

A third power advantage of some asynchronous design techniques is the application of level-sensitive latches. Latches have less input capacitance and consume less switching power than comparable flip-flops, and their use can lead to substantial savings in power [13].

However, these power advantages are not universal in asynchronous design styles. In particular, some asynchronous circuits designed for high performance have more average transitions per data bit than comparable synchronous designs, owing to the dual-rail or other multi-rail data encoding and/or completion-detection logic. While the performance advantage associated with some techniques may be turned into lower power through voltage scaling, the overall power saving is not as clear and is likely to be application dependent.

1.5.3 Modularity and ease of design

Another advantage of asynchronous design is the modularity that comes from the send and receive channel-based discipline. Blocks that communicate using the

same handshaking discipline can very easily be connected, offering a plug-and-play approach to design. Moreover, the handshaking discipline offers an immediate notion of flow control within the design. If some stage is not ready to receive new tokens then the sender will block until the receiver is ready.

In general, well-designed asynchronous circuits are one form of latency-insensitive design, and they make changing the level of pipelining late in the design cycle substantially less disruptive. This is particularly advantageous in enabling long wires to be pipelined late in the design cycle, as we will explore in Chapter 4.

More generally, asynchronous circuits provide an effective component in designs that are globally asynchronous and locally synchronous. They can offer the high-throughput low-latency power-efficient interconnect technology that is essential to creating the backbone of networks on chips. This has been the focus of Silistix, an asynchronous start-up company out of the University of Manchester [45]. We will explore this aspect of asynchronous design in Chapter 14.

1.5.4 Reduced electromagnetic interference

In a synchronous design, all activity is locked into a very precise frequency. The result is that nearly all the energy is concentrated in very narrow spectral bands around the clock frequency and its harmonics. Therefore, there is substantial electromagnetic noise at these frequencies, which can adversely affect neighboring analog circuits. Activity in an asynchronous circuit is uncorrelated, resulting in a more distributed noise spectrum and lower peak noise. A good example of this is the Amulet 2e asynchronous micro-processor, which displayed a lower overall emission level and much less severe harmonic peaks than similar clocked circuits [20].

1.6 Challenges in asynchronous design

Despite these advantages, which have been evident for some time, asynchronous circuits are only now gaining acceptance in industry. Two main reasons have to do with the challenges they present in testing and debugging and the general lack of CAD tools that support asynchronous design.

1.6.1 Testing and debugging

Testing for synchronous ASICs is made very efficient by the use of specialized scannable flip-flops, advanced tools for automated test-pattern generation, and IEEE test circuit standards such as the Joint Test Action Group (JTAG). The basic challenge associated with many asynchronous design styles is the presence of loops in the circuit that are not cut by specific latches or flip-flops. Many of these loops must be cut with additional circuitry in order to achieve sufficient observability and controllability in the circuit for test purposes and, perhaps more