CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Innovations in medicine, entertainment, e-commerce and mobile communication all generate roots in advances in semiconductor processing, which frequently decrease the minimized feature size of wires and transistors, which are all the basic building blocks of chips. This repeated reduction supports ever-rising amount of transistors on a single chip, allowing them to carry out increasingly sophisticated tasks. In addition, smaller wires and transistors have less capacitance and resistance, enabling both the higher performance and lower power which is today’s thirst of the market.

The predominant circuit design style is a synchronous design with complementary metal–oxide–semiconductor (CMOS) static logic gates and a global clock to regulate static changes. However, as process unpredictability rises and the challenges of routing a global clock across a large chip turn into increasingly problematic, fundamentally different design methodologies such as asynchronous design have become an increasingly interesting alternative. The global clock will be removed in asynchronous design by the favour of distributed local handshaking to manage changes of state and data transfer. Though, academic research in this area can be traced back to the 1950s, it has taken the late 1990s and 2000s for this methodology to develop. Several start-up companies have begun to commercialize asynchronous design as a
competitive advantage for a wide range of applications. However, the mass application of asynchronous design has been an elusive goal for academic researchers and recent advances are promising. Different types of integrated circuits (ICs) are there and the design style choice for a particular application depends on the relative performance, volume, power and other market needs 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. Moreover 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 (IP) 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. 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.

Designing of digital computers, control systems, data communications and electronic digital hardware based applications uses digital circuits. An electronic calculator with the input device keyboard and the output device numerical display is a digital system, similar to a digital computer. The function keys plus and minus are used to perform instructions in the calculator. The numeric keys enter the data. Results are displayed in numeric form. Some calculators with printing capabilities and programmable facilities resemble a digital computer.

Logic circuits may be combinational or sequential in digital systems. Outputs of a combinational circuit are determined directly from the present combination of inputs without regard to previous inputs. An information-processing operation fully specified logically is performed by combinational circuit using a set of Boolean functions. Memory elements (binary cells) are employed in sequential circuits in addition to logic gates. Their outputs are a function of the inputs and the state of the memory elements. The state of these memory elements is in turn a function of previous inputs. As a consequence, not only on present inputs, but also past inputs varies the outputs of a sequential circuit and the time sequence of inputs and internal states specifies the circuit behavior. Input variables, logic gates and output variables are present in a combinational circuit. The logic gates generate signals to the outputs by accepting signals from the inputs. This process provides required output data by transforming binary information from the given input data. Obviously, binary signals are used to represent both input and output data, i.e., they exist in two possible values, logic-1 and logic-0. Figure 1.1 shows block diagram of a combinational circuit. The variables
from an external source are \( n \) input binary and the variables to an external destination are \( m \) output binary.

In many applications, storage registers located either in the vicinity of the combinational circuit or in a remote external device are source and/or destination. By definition, the behavior of the combinational circuit is not influenced by an external register because, if it does, the total system becomes a sequential circuit.

![Figure 1.1 Block diagram of combinational circuits](image)

1.2 DESIGN PROCEDURE OF LOGIC CIRCUITS

Starting with the verbal outline of the problem and ending in a logic circuit diagram or a set of Boolean functions results in design of combinational circuits from which the logic diagram can be easily obtained. The steps to be followed are:

1. State the problem.
2. Determine the number of available input variables and required output variables.
3. Assign letter symbols for input and output variables.
4. Derive the truth table defining the required relationships between inputs and outputs.
5. Obtain the simplified Boolean function for each output.
6. Draw the logic diagram.

Input columns and output columns are present in truth table of a combinational circuit. For n input variables, the 1's and 0's in the input columns are obtained from the $2^n$ binary combinations. Examination of the stated problem determines the binary values for the output. For every valid input combination, an output can be equal to either 0 or 1. However, some input combinations may not occur as indicated by the specifications. Such combinations are don't-care conditions. The exact definition of the combinational circuit is specified by output functions in the truth table. It is important that the truth table should interpret verbal specifications correctly. Sometimes intuition and experience must be used by designer to arrive at the correct interpretation. There should be complete and exact word specifications. An incomplete combinational circuit will be obtained due to wrong interpretation in truth table.

1.3 SYNCHRONOUS SEQUENTIAL LOGIC

1.3.1 Design Procedure (Sequential Circuits)

The process of starting with a set of specifications and culminating in a logic diagram or a list of Boolean functions is procedure for design of a clocked sequential circuit. A combinational circuit is fully specified by a truth table, whereas a synchronous sequential circuit is made up of flip-flops and combinational gates. Choosing the flip-flops and then finding a combinational gate structure are involved in the design of the circuit, which results in a circuit that fulfills the stated specifications.

The combinational circuits have been used in digital circuits, where the outputs at any instant of time depend upon the inputs present at that time.
Although combinational circuits are used in every digital system, most systems encountered in practice also include memory elements, which require that the system be described in terms of sequential logic.

Figure 1.2 shows the example of the block diagram of a sequential circuit. It consists of a combinational circuit to which memory elements are connected to form a feedback path. The devices capable of storing binary information within them are memory elements. The state of the sequential circuit is defined by binary information stored in the memory elements at any given time. The binary information from external inputs is received by sequential circuit. The binary value at the output terminals is determined by inputs together with the present state of the memory elements. The condition for changing the state in the memory elements is also determined by them. The sequential circuit’s external output is not only a function of external inputs, but also of the present state of the memory elements and is demonstrated in block diagram. The next state of the memory elements is also a function of external inputs and the present state. Thus time sequence of inputs, outputs and internal states specifies a sequential circuit. The sequential circuits are of two main types. Depending on the timing of their signals, they are classified into synchronous and asynchronous circuits. The knowledge of signals at discrete instants of time defines synchronous sequential circuit behavior. The order in which its input signals change defines the behavior of an asynchronous sequential circuit at any instant of time. Time-delay devices are memory elements commonly used in asynchronous sequential circuits. The finite time taken by the signal to propagate through the device determines the memory capability of a time-delay device.
In practice, required delay is produced by the internal propagation delay of logic gates which makes the physical time-delay units unnecessary. In gate-type asynchronous systems, logic gates constitute propagation delays of required the memory elements shown in Figure 1.2. Thus, a combinational circuit with feedback is regarded as an asynchronous sequential circuit. An asynchronous sequential circuit may become unstable because of the feedback among logic gates. Many difficulties may be imposed on the designer due to instability problem. The signals that affect the memory elements only at discrete instants of time must be employed in a synchronous sequential circuit. This goal can be achieved by using pulses of limited duration throughout the system, where one pulse amplitude represent logic 1 and pulse amplitude (or the absence of a pulse) represents logic 0. When two pulses arrive from independent sources to the input the difficulties with a system of pulse are unpredictable delays, separation of pulses and unreliable operation. Fixed amplitudes such as voltage levels for the binary signals are used in practical synchronous sequential logic systems. Master-clock generator is a timing device which generates a periodic train of clock pulses to achieve synchronization. The clock pulses are distributed throughout the system in such a way that with the arrival of the synchronization pulse only the memory element is affected.
Generally, the signals specifying required change in memory elements and clock pulses are applied together into AND gates. Only at instants the signal coincides with the arrival of clock pulses AND-gate outputs are produced. Clocked sequential circuits are synchronous sequential circuits that use clock pulses in the inputs of memory elements. The most frequently type encountered are Clocked sequential circuits. Instability problems are not manifested in them and their timing is easily divided into independent discrete steps, each of which is considered separately. The clocked type sequential circuits are discussed in this chapter. Flip-flops are the memory elements used in clocked sequential circuits. One bit information is stored in these circuits.

1.4 CHALLENGES IN ASYNCHRONOUS MODE OPERATIONS

The time sequence of inputs, outputs, and internal states specifies a sequential circuit. The change of internal state occurs in response to the synchronized clock pulses in synchronous sequential circuits. Clock pulses are not used in asynchronous sequential circuits. When there is a change in the input variables change of internal state occurs. Clocked flip-flops are the memory elements in synchronous sequential circuits. Un-clocked flip-flops or time-delay elements are the memory elements in asynchronous sequential circuits. The finite time taken for the signal to propagate through digital gates is responsible for memory capability of a time-delay device. A combinational circuit with feedback is similar to an asynchronous sequential circuit. The design of asynchronous sequential circuits is simple than that of synchronous circuit because it has timing problems involved in the feedback path. These timing problems are eliminated by triggering all flip-flops with the pulse edge in a properly designed synchronous system. During the short time of the pulse transition, the change from one state to another state occurs. The state of the
system is allowed to change immediately after the input changes as the asynchronous circuit does not use a clock. Even though a feedback path exists, care must be taken to ensure that each new state keeps the circuit in a stable condition.

1.4.1 Asynchronous Sequential Logic

A variety of applications employs asynchronous sequential circuits. When speed of operation is important, especially in those cases where the digital system must respond quickly without having to wait for a clock pulse the asynchronous sequential circuits are used. In order to use in small independent systems that require only a few components these are more economical and it may not be practical to go to the expense of providing a circuit for generating clock pulses. The applications in which the input signals to the system may change at any time, independently of an internal clock uses asynchronous circuits. Asynchronous circuits can be done with the two units having communication between them and each unit must have its own independent clock. A mixed system where some part of the synchronous system has the characteristics of an asynchronous circuit is often produced by digital designers. The proper operation of the digital system can be verified by knowledge of asynchronous sequential logic behavior.

Asynchronous sequential circuits must be allowed to attain a stable state before the input is changed to a new value to ensure proper operation. It is impossible to have two or more input variables change at exactly the same instant of time without an uncertainty because of delays in the wires and the gate circuits. Hence changes in two or more variables are simultaneously prohibited. This restriction is that at any time only one input variable can change and the time between two input changes must be longer than the time
the circuit takes to reach a stable state. This type of operation is defined as fundamental mode. When the circuit is in a stable condition, the fundamental-mode operation assumes that the input signals change one at a time and only.

1.4.2 Asynchronous Circuit Principles

The Figure 1.3 shows a synchronous design style in which synchronous circuits are composed of combinational function blocks and registers. The global clock which triggers at the same time the memorization of the complete state of the circuit controls the circuit activity. The combinational circuits start the computation of the next state to be sampled at the next clock edge, when a new state is sampled and placed in the registers. All functional blocks correctly complete their operations when the clock signal is fixed and their data outputs are stable and ready to be sampled. The whole circuit is applied with a global timing assumption which provides the longest combinational path (critical path) must not exceed the clock period. Replacing the clock signal with some form of handshaking between adjacent registers results in synchronization in asynchronous circuits.

![Figure 1.3 Basic Structure of a Synchronous Circuit.](image)

The presence of data at their inputs and outputs controls computation of communicating stages present in an asynchronous circuit. Their data-flow model resembles their behaviour. One asynchronous stage
describes that the input port present in it receives data, the functional block computes it, and output port sends the result through a register. A communication protocol implemented within the control part of the stage controls the data communication over its ports rather than an external signal, like a clock. These kind of protocols are handshake protocols which require a bi-directional exchange of information between senders and receivers.

In an asynchronous circuit, the communication protocol is the basis for sequencing the rules. The computation in an asynchronous stage starts if and only if all the data required for the computation are available. The asynchronous stage releases the input ports only when the result can be stored in the register. Only when the output port is available outputs are given i.e. released by the next stage connected to it, at the end of the previous communication. In order to get rid of the clock, the implementation of the communication protocol must be done in each stage and helps to control the sequencing locally. Bit-level functions, word-level arithmetic functions or even complex algorithms are the different functions that can be implemented in each asynchronous stage. The following three main characteristics are always present in it and they are,

1. A unified communication protocol is important for all their channels.

2. The input data set at the highest speed is computed and result is placed at the output as soon as possible (minimum forward latency), while the synchronous circuits forward latency may not vary with the data computed. The clock period is constraint the forward latency.
3. A maximum throughput which corresponds to the maximum frequency at which the module can process the incoming data is also present in them (maximum cycle time).

As the asynchronous stage has to release the communication channels before accepting the next data set, this characteristic is not the inverse of the forward latency. In order to compute the throughput, have to be considered the forward and reverse phases in asynchronous circuits (Notice that is not a trivial relation because this depends on the employed handshaking protocol).

1.5 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.

Figure 1.4 Traditional synchronous ASIC
Figure 1.4 Showing the traditional synchronous ASIC, it contains the combinational logic and flip flop. The primary inputs and outputs are denoted PI and PO advantage of standard-cell-based ASIC designs is being subverted by the increasingly difficult design challenges posed by modern semiconductor processes. These challenges affect both high-performance and low-power ASICs, as described below. Synchronous logic also has two main disadvantages, as follows:

- The clock signal must be distributed to every flip-flop in the circuit. As the clock is usually a high-frequency signal, this distribution consumes a relatively large amount of power and dissipates much heat. Even the flip-flops that are doing nothing consume a small amount of power, thereby generating loss of heat energy in the chip.

- The maximum possible clock rate is determined by the slowest logic path in the circuit, otherwise known as the critical path. This means that every logical calculation, from the simplest to the most complex, must complete in one clock cycle. One way around this limitation is to split complex operations into several simple operations, a technique known as 'pipelining'. This technique is prominent within microprocessor design and helps to improve the performance of modern processors.

- Source-synchronous interfaces remove the time-of-flight limit on interconnection between ICs and require no controlled clock skew. Another advantage of source-synchronous interfaces is dramatically increased I/O frequencies. With increased bandwidth per I/O driver, the number of pins per
interface better matches the capabilities of available IC-packaging technologies. I/O-driver frequencies can reach five to 10 times the core-logic frequency.

- However, source-synchronous interfaces create new design-analysis challenges. Interface latency is not necessarily predictable; if your design requires predictable latency, overall interface latency increases. Increases in I/O speeds require more robust IC-package electrical performance. Because the I/O frequency can be much higher than that of the core logic, I/O-interface-logic complexity must grow to handle the frequency multiplication. Data bit-to-bit timing skews and "eye patterns" define overall link-operation frequencies, whereas you may have previously ignored these effects.

1.6 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 shown in the Figure 1.5. One common way of asynchronous design basics 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 from the blocks B, D, C, A and typically point-to-point. 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 the 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 create general statements in connection with 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.

![Asynchronous blocks communicating using channels](image)

**Figure 1.5 Asynchronous blocks communicating using channels**

Asynchronous design is more generally the level of effort put into the comparable synchronous design. Better technology migration potential - Integrated circuits will often be implemented in several different technologies during their lifetime. Early systems may be implemented with gate arrays, while later production runs may migrate to semi-custom or custom ICs. Greater performance for synchronous systems can often only be achieved by migrating all system components to a new technology, since again the overall system performance is based on the longest path. In many asynchronous systems, migration of only the more critical system components can improve
system performance on an average, since performance is dependent on only the current active path. Also, since many asynchronous systems sense computation completion, components with different delays may often be substituted into a system without altering other elements or structures.

1.6.1 Automatic Adaptation to Physical Properties

The delay through a circuit can change with variations in fabrication, temperature, and power-supply voltage. Synchronous circuits must assume that the worst possible combination of factors is present and clock the system accordingly. Many asynchronous circuits sense computation completion, and will run as quickly as the current physical properties allow. Robust mutual exclusion and external input handling - Elements that guarantee correct mutual exclusion of independent signals and synchronization of external signals to a clock are subject to metastability.

A metastable state is an unstable equilibrium state, such as a pair of cross-coupled CMOS inverters at 2.5V, which a system can remain in for an unbounded amount of time. Synchronous circuits require all elements to exhibit bounded response time. Thus, there is some chance that mutual exclusion circuits will fail in a synchronous system. Most asynchronous systems can wait an arbitrarily long time for such an element to complete, allowing robust mutual exclusion. Also, since there is no clock with which signals must be synchronized, asynchronous circuits more gracefully accommodate inputs from the outside world, which are by nature asynchronous. With all of the potential advantages of asynchronous circuits, one might wonder why synchronous systems predominate.
The reason is that asynchronous circuits have several problems as well. Primarily, asynchronous circuits are more difficult to design in an ad hoc fashion than synchronous circuits. In a synchronous system, a designer can simply define the combinational logic necessary to compute the given functions, and surround it with latches. By setting the clock rate to a long enough period, all worries about hazards (undesired signal transitions) and the dynamic state of the circuit are removed. In contrast, designers of asynchronous systems must pay a great deal of attention to the dynamic state of the circuit. Hazards must also be removed from the circuit, or not introduced in the first place, to avoid incorrect results. The ordering of operations, which was fixed by the placement of latches in a synchronous system, must be carefully ensured by the asynchronous control logic.

For complex systems, these issues become too difficult to handle by hand. Unfortunately, asynchronous circuits in general cannot leverage off, existing CAD tools and implementation alternatives for synchronous systems. For example, some asynchronous methodologies allow only algebraic manipulations (associative, commutative, and DeMorgan’s Law) for logic decomposition and many do not even allow these. Placement, routing, partitioning, logic synthesis, and most other CAD tools either need modifications for asynchronous circuits are not applicable at all. Finally, most of the advantages of asynchronous circuits are towards higher performance, it isn't clear that asynchronous circuits are actually faster in practice. Asynchronous circuits generally require extra time due to their signalling policies, thus increasing average-case delay. Whether this cost is greater or less than the benefits listed previously is unclear, and more research in this area is necessary.
1.6.2 High Performance

Performance can be measured in terms of system latency or throughput or a combination of both. 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.

1.6.3 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 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.

1.6.4 Clock Speed

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. 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. Domino logic is limited to full custom design flows because of its reduced noise margin and its timing assumptions are not currently supported by semi-custom design tools. Some asynchronous design flows embed domino logic within a pipeline template. 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. 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.

1.6.5 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 be constantly providing a powerful clock that reaches all parts of the circuit. The removal of the global clock in favour of power-efficient control circuits can sometimes lead to significant power savings. Moreover, asynchronous architectures can reduce power consumption by minimizing data movement to need specific points. 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 and pagers).
The alternative in synchronous design would be a standby-power-inefficient circuit that continually poll 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. 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.

1.6.6 Modularity and ease of design

Another advantage of asynchronous design is the modularity that comes from the send and receives channel-based discipline. Potential advantages of asynchronous design 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. 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 in creating the backbone of networks on chips.

1.6.7 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 neighbouring 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 asynchronous micro-processor, which displayed a lower overall emission level and much less severe harmonic peaks than similar clocked circuits.

1.7 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 be done with the challenges they present in testing and debugging and the general lack of CAD tools that support asynchronous design.

1.7.1 Asynchronous Design Flows

There are many different possible design flows for asynchronous circuits. Three fundamental designs are described here to demonstrate the diversity of flows. 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 neighbours via channels. Each leaf cell is typically implemented with a small set of transistors, between 10 and 100. Early attempts to automate this process are encouraging, but today’s 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 and digital-signal processing chips. It has been commercialized by Fulcrum Microsystems and has led to remarkable results.

Introduction to circuit designs whose performance exceeds the resources available through semi-custom standard-cell design flows. 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 circuits. 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 and subsequent results are encouraging. 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. 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, an 80C51 micro-controller, and an Advanced RISC Machines (ARM) micro-processor that compare quite favourably 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.

1.7.2 Potential Advantages of Asynchronous Design

Asynchronous circuits have demonstrated potential benefits in many aspects of system design. 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.

1.7.3 Channel-Based Asynchronous Design

Asynchronous channels are often composed of a hierarchical network of blocks, which contain ports interconnected via asynchronous channels. These channels are simply a bundle of wires and a protocol for synchronizing computation and communicating data between blocks. The smallest block that communicates with its neighbours using asynchronous channels is called a leaf cell. Larger blocks that communicate via channels may be called modules. Numerous forms of channels have been developed that trade robustness to timing variations with improved power and performance.

1.7.4 Bundled-Data Channels

Figure 1.6 shows the element of bundled-data channels. Bundled-data channels consist of a single request line bundled with a unidirectional single-rail data bus that is coupled with an acknowledgement wire. In the typical bundled-data push channel, illustrated, the sender initiates the communication and tells the receiver when new valid data is available. Bundled-data channels can be implemented as shown in Figure 1.7 with two-phase handshaking, in which there is no distinction in meaning between the risings and falling transitions of the request and acknowledge handshaking wires. More specifically, both the rising and falling transitions of the request wire indicate the validity of new data at the output of the sender, and both the rising and falling transitions of the acknowledge signal indicate that the data has been consumed by the receiver, as illustrated. After the consumption of
the data, the sender is free to change the data. Consequently, it is the voltage transitions of these wires that carry meaning rather than their absolute voltage levels. Thus, this type of two phase protocol is often called transition signalling.

![Diagram of Bundled Data Channel]

**Figure 1.6  Elements of bundled data channel**

In contrast, in four-phase bundled-data protocols, only the request line going high signifies the validity of data and a single transition of the acknowledge signal identifies that the data has been consumed by the receiver and can be safely altered by the sender. In the narrow protocol, also known as the early protocol as shown in Figure 1.8(a), the rising transition of the

![Diagram of Transition Signal in Bundled Data Push Channel]

**Figure 1.7 Transition signal in a bundled data push channel**
acknowledge signal signifies that its data has been consumed and the single-rail data can be changed. After the acknowledge signal has risen the sender resets the request line and the receiver then resets the acknowledgement line, bringing both wires back to the initial state in preparation for the next token transfer. By contrast, in the late protocol shown in Figure 1.8(b), the data is valid from when the request falls to when the acknowledge signal falls, and the first two phases can be considered as a warm-up in preparation for the latter two active transitions. In the broad protocol as shown in Figure 1.8(c), the signals follow the same four transitions but the falling transition of the acknowledge signal signifies that the data can be reset. As the name implies, the time for which the data must be stable is much larger in the broad protocol than in the narrow protocol. It is to be noticed that in all these protocols the only purpose of two of the four phases of the communication is to reset the handshaking wires in preparation of the next data transfer. Bundled-data channels are area efficient because the request and acknowledge line overhead is distributed over the width of the entire data bus which encodes data with one wire per bit, as in a typical synchronous circuit. They are also power efficient because the activity on the data bus may be low and the power consumption of the handshaking wires is relatively small if the data path is wide.

Figure 1.8 (a) Early, Four phase bundled-data push protocol
A disadvantage of bundled-data channels, however, is that they involve a timing assumption, i.e. that the data at the receiver is valid upon the associated transition of the request line. Moreover, unlike in a synchronous circuit there is no associated clock that can be slowed down to ensure that this timing assumption is satisfied. Consequently, significant margins are typically necessary to ensure proper operation. These margins are on the forward latency path of the circuit, which is often critical to system performance.

One-of-N channel

A 1-of-N channel uses N data wires to send $\log_2 N$ bits of data, as illustrated in Figure 1.9 and is typically designed with four phases.
Figure 1.9 1-of-N channel

Figure 1.10(a) 1-of-2(dual-rail) channel

Figure 1.10(b) 1-of-4 channel
After a single data wire has risen, the acknowledge wire also rises, indicating that the data has been consumed by the receiver and that the sender can safely reset the risen data wire. Once the sender resets the data signal, the receiver can reset the acknowledge signal, completing the four phases. This protocol facilitates delay-insensitive communication between blocks in that the data and its validity are encoded on the same N wires rather on separate request and data wires. Consequently, no timing assumption is needed. This increases robustness to variations, reducing the amount of timing verification required. The most well-known form of this channel is dual-rail, also known as 1-of-2 as shown in Figure 1.10 (a) and uses two data wires or rails per bit of data. One wire is referred to as the data_1 or true wire and the other as the data_0 or false wire.

Handshaking is initiated when a data wire rises; this represents both the value of the data wire and the validity of the data. The receiver can detect the validity of the input data simply by OR-ing the data_0 and data_1 wires. In many templates the data triggers a domino dual-rail logic data path concurrently. Other forms of 1-of-N data path are also possible. A 1-of-4 channel as shown in Figure 1.10 (b), system communicates two bits of data by changing only one data wire, yielding a lower power consumption than dual-rail channels yet with no additional data wires per bit (i.e. two data wires are still used per bit communicated).

Higher radix channels are also possible and have additional power advantages but require more wires per data bit. For example, a 1-of-channel system requires $\frac{8}{3}$ data wires per bit rather than the two data wires per bit required by dual-rail and 1-of-4 channels. The most trivial form is a 1-of-1 channel, which has the same form as a bundled-data channel with no data and is used to synchronize operations between sender and receiver. It is also possible to implement 1-of-N channels with the acknowledge signal inverted,
in which case it is often referred to as an enable because a high value indicates that the channel is ready to accept a new token. In addition, as in bundled-data transition signalling, it is also possible to use transition signalling, in which each transition on the 1-of-N data wires represents a new token. However, creating senders and receivers that react to both phases of the data and control wires often leads to less efficient control circuits, owing to the substantially lower mobility in P-transistors compared with that in N-transistors. The 1-of-N+1 channel has an additional request signal between sender and receiver to enable higher performance.

1.7.5 Clocking Versus Handshaking

A synchronous circuit is shown in Figure 1.11(a). A pipeline is shown in the Figure 1.11(a) for simplicity, but it is intended to represent any synchronous circuit. Designers focus mostly on the data processing while designing ASICs using hardware description languages and synthesis tools and assume the existence of a global clock. The reality is different in case of physical design. Figure 1.11(b) shows a large number of (possibly gated) clock signals produced by structure of clock buffers used in today’s ASICs. Designing of the clock gating circuitry and controlling the skew between the many different clock signals is well known to take an engineering effort. The setup to hold time window around the clock edge in a world that is dominated by wire delays is not an easy task when considering the two-sided timing constraints. Current commercial CAD tools which uses the buffer-insertion and re-synthesis process is used and may not converge, even if it does, it relies on delay models that are often of questionable accuracy.
Figure 1.11 (a) A synchronous Circuit

Figure 1.11 (b) A Synchronous circuit with clock driver and clock gating

Figure 1.11 (c) an equivalent asynchronous circuit.
An alternative method is represented by asynchronous design. Handshaking between neighbouring registers is replaced by some form of clock signal in an asynchronous circuit the clock signal. For example Figure 1.11 (c) shows the simple request-acknowledge based handshake protocol. In the Figure 1.11 (c) as a “handshake channel” or “link,” the registers which stores the data as tokens tagged with data values (that may be changed along the way as tokens flow through combinational circuits) and the transparent nature of the combinational circuits and the handshaking between registers. A token on each of its input links is being absorbed by a combinatorial circuit and computation is performed and token on each of its output links is emitted. An asynchronous circuit is a simple static data-flow structure and the data tokens flowing in the circuit should not disappear for correct operation and one token should not overtake another and the new tokens should not appear anywhere. This is ensured by the following simple rule:

- When the successor has input and stored the data token that the register was previously holding, register will be the input and store a new data token from its predecessor [The states of the predecessor and successor registers are signalled by the incoming request and acknowledge signals respectively.]
- Following this rule, along the path through the circuit, data is copied from one register to the next. Copies of the same data value will be held in subsequent registers in this process but the old duplicate data values will later be overwritten by new data values in a carefully ordered manner and the transfer of
exactly one data token will always be enclosed by a handshake cycle.

- Design of efficient circuits is completely based on “token flow game” and will address these issues later and the token-flow view is extended to cover structures other than pipelines.

- Finally, some more down-to-earth engineering comments may also be relevant.

The clock pulses that are in phase with a periodic clock signal controls the synchronous circuit in Figure 1.11(b), whereas Figure 1.11(c) showing the asynchronous circuit is controlled by locally derived clock pulses that can occur at any time; In order to check that clock pulses are generated where and when needed, local handshaking is ensured. As a result, the clock pulses are randomised over time and likely results in less electromagnetic emission and a smoother supply current without the large spikes and hence synchronous circuit is characterized.

1.7.6 Four Phase Handshake versus Single-Track Handshake

The validity and the neutrality of both inputs and outputs should be detected for pipeline stage in a four phase handshake protocol. When the pipeline is waiting for inputs and outputs to be reset during the second half of the four-phase protocol, no actual logic is being computed but it still consumes roughly half of the cycle time. Furthermore, the power consumed during their validity detection rivals that consumed in detection of inputs and outputs neutrality. The four phase handshake protocol is clearly not an ideal choice for energy efficiency due to the above characteristics. By practically eliminating the neutrality phase the single-track handshake [van Berkel and
Bink 1999] protocol tries to overcome weakness of four phase protocol. An overview of a single-track handshake protocol is shown in Figure 1.12. By sending the data token, the sender process initiates the communication. The receiver computes the logic using the data. Instead of sending an acknowledge signal back to the sender process, the receiver process resets the input tokens itself by pulling the data wires low through NMOS transistors once the data is no longer needed. This circuit shows only one discharge transistor, and there are as many NMOS discharge transistors as there are data wires. The sender detects the token consumption and gets ready to send the next token, as the data wires pulled low. Hence, eliminating the transitions associated with second part of the four phase protocol.

![Figure 1.12 Single-track handshake protocol.](image)

Single-track handshake templates have only limited work. Most of the prior work has focused to reduce the cycle time of asynchronous pipelines to less than 10 transitions and not on how to use these extra transitions to improve logic density and energy efficiency by using single-track handshake protocol.

Based on single-track handshake Protocol, (Ferretti et al, 2002) provide a family of asynchronous pipeline templates. Each of their pipeline templates contains only a small amount of logic similar to high throughput QDI circuits. Furthermore, some very tight timing margins that may require
significant post-layout analog verification are used by the 6-transition cycle
time pipelines. The control path of GasP (Sutherland and Fairbanks 2001)
bundled-data pipelines uses single-track circuits. However, the single-track
handshake protocol is not used in actual data path of the pipeline. For the
proposed pipeline templates, employs the single-track handshake protocol.
Increasing the logic density and energy efficiency of each pipeline stage is
focussed by the designers and not on reducing cycle time.

1.7.7 Globally Asynchronous Locally Synchronous (GALS) Design

Globally asynchronous locally synchronous (GALS) is a Model of
Computation (MoC) that emerged in the 1980s. It is based on the synchronous
MoC and on the asynchronous MoC. It allows to relax the synchrony
assumption to model and design computer systems consisting of several so-
called synchronous islands (the program of each such island obeying the
synchronous MoC) interacting with each other with asynchronous
communication, e.g., FIFOs (First In First Out).

A GALS circuit consists of a set of locally synchronous modules
communicating with each other via asynchronous wrappers. Advantages
include lower power consumption and electromagnetic interference (EMI).
GALS is sometimes used in system-on-a-chip (SoC) design.

The GALS MoC is a compromise between a completely
synchronous system (a single clock domain, perhaps with clock gating on
some registers) and a completely asynchronous circuit (every register can be
considered its own independent clock domain). Each synchronous subsystem
("clock domain") can run on its own independent clock frequency. In
synchronous systems, the clock signal is used for a variety of purposes. The clock signal is global in nature. During the clock edge, the flip-flops are updated and the new state ripples through the circuit to compute the next state. This provides a variety of structured design methods. The structured design of asynchronous circuits requires a timing discipline to replace the global clock. Simple request and acknowledge signalling can be used for this purpose. The subsystem on the transmitting side plays the active role and initiates the transition, whereas the subsystem on the receiving side waits and acknowledges. This is called handshaking.

1.7.8 Advantages of Asynchronous circuit design

GALS design techniques employ the finer points of synchronous and asynchronous design methods to eliminate problems arising due to clock distribution, power dissipation and large area overhead. With the recent rise in the demand for System-on-a-Chip (SoC) designs, global clock distribution and power dissipation due to clock distribution are inevitable. In order to reduce/eliminate the effects of the global clock in synchronous designs and large area overhead in asynchronous designs, an alternative approach would be to utilize GALS design techniques. Not only do GALS designs eliminate the issue of using a global clock, they also have smaller area overhead when compared to purely asynchronous designs.

Among the various GALS design approaches proposed till date, this thesis focuses on the working and implementation of Asynchronous Wrapper designs proposed by (Muttersbach et al 2000). This thesis specifically addresses different approaches to incorporate the wrappers in VLSI circuits, rather than discussing the efficiency and viability of GALS
design techniques over purely synchronous or asynchronous approaches. It has been proven by researchers that GALS design approaches bring down power consumption due to the elimination of the global clock by small amounts, but there is also a drop in performance. Since the goal of this thesis is to introduce the reader to GALS design techniques and not prove their efficiency, it is out of the scope of this thesis to validate the results shown in.

In this aim to introduce the reader to GALS design techniques, first it provide a comparison of synchronous and asynchronous design approaches, and then discuss the need for GALS design approaches. It will address issues affecting GALS such as metastability, latency, flow control and local clock alteration.

An in-depth discussion and analysis of the wrapper design approach proposed is provided based on the state transition graphs (STGs) that characterize the port-controller AFSMs. Various data transfer channel configurations that incorporate the wrapper port-controllers are designed and realized through VHDL codes, with their functioning verified through simulation results. Design examples showing the working of asynchronous wrappers to achieve point-to-point, synchronous-synchronous and synchronous-asynchronous data communication are provided. Finally, a design example to achieve multi-point data communication is realized. This example incorporates a previously proposed idea. A modification to this idea by designing an arbiter that arbitrates between two separate requests coming into a multi-input port. Through the above design examples, the functionality and working of GALS asynchronous wrappers are verified and recommendations for modifications are made to achieve flexible multi-point data communication.
1.8 PIPELINING

Pipeline stages are introduced to increase the throughput of a design. When two input samples are processed at any given time, it is known as two stage pipeline. In order to process one input sample, half the number of the effective control steps is sufficient. This can be viewed through a different angle for a purpose. The number of control steps and can be improved by increasing the control steps for pipelining. Power management creates the stack needed and schedules the control signals by addition of new control steps.

1.9 QUASI-DELAY-INSENSITIVE CIRCUITS

Communication between different parallel processes is carried out in QDI circuit templates using 1-of-N encoded channels. A total of N wires are used to encode data with only one wire asserted at a time in a 1-of-N channel. 1-of-2 (dual-rail) or 1-of-4 encodings are mostly used in high throughput QDI circuits. Figure 1.13 shows a 1-of-4 encoded channel communication where validity is signified by setting one of the four data rails and neutrality is indicated by resetting of all four data rails. In most high speed QDI circuits a four phase handshake process is commonly used which is used to initiates the communication of sender process by sending data over the rails i.e. by asserting one of the data rails. The presence of data is detected by receiver process and sends an acknowledgement, once it no longer needs the data. At this point, all data rails are reset by the sender process. The neutrality of input tokens is detected by receiver process. Once it is ready to receive a new data token the acknowledge signal is de-asserted and the cycle repeats.
1.9.1 Pre-Charge enable Half-Buffer

A slightly modified version of pre-charge half buffer (PCHB) template is the pre-charge enable half-buffer (PCeHB) template and it is a workhorse for most high throughput QDI circuits. It is small as well as fast with a cycle time of 18 transitions. The logic function being computed is implemented by a pull-down stack transistor in a RSPCHB pipeline. Separate logic gates checks the input and output validity and neutrality. The overhead of explicit registers is removed by combining actual computation with data latching. In the case of explicit checking, due to the extra validity and neutrality detection logic gates there is associated high handshake overhead. Newly added logic stages and completion detection logic gates due to the extra transitions limit the energy efficiency gain. It is necessary to add extra logic stages to each pipeline stage. At alternative handshake protocols as well as some timing assumptions in QDI circuits to improve the energy efficiency of high throughput asynchronous pipelines.