The Thesis Entitled

Asynchronous Design Methodology for an Efficient Implementation of Low Power ALU

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering

by

G. Sundar

Under the guidance of

Dr. Chitta Ranjan Mandal

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
India
July 2006

Certificate

This is to certify that the dissertation entitled **Asynchronous Design Methodology for an Efficient Implementation of Low Power ALU,** submitted by G. Sundar, a student in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India, for the award of the degree of Master of Science, is a record of an original work carried out by him under my supervision and guidance. This dissertation fulfills all requirements as per the regulations of this Institute and in my opinion has reached the standard needed for submission. Neither this dissertation nor any part of it has been submitted for any degree or any academic award elsewhere.

Dr. Chitta Ranjan Mandal
Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
West Bengal, India 721302

Abstract

The power consumption becomes an important issue in circuit design technologies. The power dissipation in high-performance CMOS VLSI circuits like microprocessors is becoming an increasing problem. One reason for the high power dissipation is the almost universal design approach synchronous circuits, which imposes global synchrony across a chip. This is achieved by applying a common clock to all the functional units on a chip and has the undesirable side effect of causing those units to dissipate power whether or not they are doing useful work. The main objective of designing the asynchronous circuits will be there is no master clock, the reduction in silicon by following domino logic with dual-rail logic and thus ensures the power consumption in designing the circuits.

We present a design technique for implementing asynchronous ALUs with CMOS domino logic and delay insensitive dual rail four-phase logic. It ensures economy in silicon area and potentially for low power consumption. The design has been described and implemented to achieve high performance in comparison with the synchronous and available asynchronous designs. This implementation justifies the claimed performance through the SPICE simulation results.

Keywords: Integrated Circuits, Design Styles, Domino Logic, Delay insensitive, 4-phase dual-rail logic, Arithmetic and Logic Structures.

Acknowledgement

This thesis would not have been a reality if my advisor Dr. C. R. Mandal hadn't given the best of moral support to me from the day I started my work here. His constant support coupled with his well-timed advices has helped me at every step of my work. I express my deep gratitude and indebtedness to him for giving me this golden opportunity.

I am particularly thankful to Prof. A. K. Majumdar, Prof. B. Majumdar, Prof. S. Ghosh, , Prof. I. Sengupta, Prof. B. D. Liu and Prof D. Sarkar for their constant encouragement and support. I would like to express my devout thanks to Dr. D. Samanta, who inspite of his busy schedule selflessly helped me throughout my project work.

I am also deeply indebted to Dr. Gunasekaran, the Registrar of the Institute and all my tamil friends for their constant help and encouragement throughout this period. I sincerely thank Dr. D. K. Nanda for his valuable support. Special thanks to Monalisaji and her family for the unstinted support by making me mentally tough enough to face and overcome difficult situations which came by.

I am grateful to Jayanto da, Anupam da, Prasenjit da and all the computer and Informatics Centre staff members for their help at times of need.

I am thankful to Ramesh, and my beloved sister Vijayakumari for their continuous inspiration and support throughout this period.

Last but not the least I express my gratitude to my mother and father for their sacrifice and patience in spite of many lost hours in their covetable association with me.

G. Sundar.

Dedicated to
My beloved Parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The power consumption becomes an important issue in design technologies. The power dissipation in high-performance CMOS VLSI circuits like microprocessors is becoming an increasing problem. Even when battery power portability is not an issue the 20 or 30 Watt consumption of the latest high-end processors makes it difficult to keep the silicon at an acceptable operating temperature. At lower performance levels the designers of battery powered systems must make difficult trade-offs between the processing demands of, for example, hand-writing recognition software and the minimum acceptable battery life of their products. The process advances which have caused CMOS to progress from a low power technology to a high power technology show no signs of abating and, if new approaches are not developed, state-of-the-art performance in twelve years time will only be delivered at the cost of power dissipations one or two orders of magnitude higher than today's. While there are developments that alleviate this problem, such as the trend towards 3 volt (and later 2 volt) operation, these do not go far enough to remove the possibility that power dissipation might limit the performance that a chip can deliver.

One reason for the high power dissipation is the almost universal design approach, which imposes global synchrony across a chip. This is achieved by applying a common clock to all the functional units on a chip and has the undesirable side effect of causing those units to dissipate power whether or not they are doing useful work. There is no

doubt that the synchronous approach to logic design has been very effective over the last two decades, and has enabled great advances to be made in the productivity of designers and their design tools and in the performance of machines. However there is now a resurgence opinion suggesting that it may be time to re-assess the merits of other design approaches. Higher speeds and larger chips are making the abstraction of global synchrony increasingly hard to sustain even on a single chip, with the power dissipation making this approach much less attractive. While it is possible to address power issue by gating clocks to individual units, this makes the clock skew problem much worse and is therefore needs for better solution.

The power consumption can be reduced by decreasing the supply voltage, load capacitance and frequency [1]. Several attempts are made to address this problem to reduce the switching activity of logic in redundant cycles [3] [21]. Xi et. a1 provides an optimization algorithm for buffer and device sizing under process variations [2]. Although it minimizes the skew, this methodology is limited when operating frequency is very high. On the other hand, Globally Asynchronous and Locally Synchronous (GALS) technique [10] [22] aims to eliminate the global clock, by partitioning the system into several synchronous blocks and communicating asynchronously among blocks. However, the global signaling protocol increases the total area-power penalty and affects performance of the system.

Asynchronous design approaches are therefore attracting renewed interest. New approaches to asynchronous designs are overcoming some of the difficulties, which had previously been impediments to cost-effective designs. Asynchronous designs tend naturally (in

CMOS) to use power only when doing useful work and interfacing disciplines produce more modular designs with an inherent potential for component reuse[5].

In asynchronous systems, idle parts of the chip consume negligible power [9] [12]. This feature is particularly valuable for battery – powered equipment, but it can also cut the cost of larger systems by reducing the need for cooling fans and air-conditioning to prevent them from overheating. The amount of power saved depends on the machine's pattern of activity. Systems with parts that act only occasionally benefit more than systems that act continuously.

Most computers have components, such as the floating-point arithmetic unit, that often remain idle for long periods. Furthermore, asynchronous systems produce less radio interferences than synchronous machines do [39]. Because a clocked system uses a fixed rhythm, it broadcasts a strong radio signal at its operating frequency and at the harmonics of that frequency. Such signals can interface with cellular phones, televisions and aircraft navigation systems that operate at the same frequencies. Asynchronous systems lack a fixed rhythm, so they spread their radiated energy broadly across the radio spectrum, emitting less at any one frequency. Yet another benefit of asynchronous design is that it can be used to build bridges between clocked computers running at different speeds. Many computing clusters, for instance, link fast PCs with slower machines [33]. These clusters can tackle complex problems by dividing the computational tasks among the PCs. Finally, although asynchronous design can be challenging, it can also be wonderfully flexible. Because the circuits of an asynchronous system need not share a common rhythm, designers

have more freedom in choosing the system's parts and determining how they interact. Moreover, replacing any part with a faster version will improve the speed of the entire system. In contrast, increasing the speed of a clocked system usually requires upgrading every part. Several researchers propose asynchronous approaches to cope with performance and timing issue. Tang et. al. design a 16-bit asynchronous ALU with asynchronous pipeline architecture [4]. In this approach, simple handshake cells embedded in pipeline stages make the ALU run fast. However, large power has consumed by this design while waiting for the incoming data. In contrast, by using Galois Field arithmetic logic and reduced switching activity in the latches, work proposed in [5] achieved low power in an asynchronous ALU design. Since the improper rotate-wire concept of data buses, the time required for each multiplication operation becomes larger and it results in degradation of the performance.

In reducing the time dependency of an asynchronous design of Quantum dot Cellular Automata (QCA), Hemani et al. [6] used GALS with delay-insensitive data encoding scheme. Here each gate has locally synchronized by corresponding clocking zone(s). Appropriate data forwarding and synchronization guaranteed at the gate-level that reduces the number of clocking zones and increases the circuit speed. Nevertheless, the overall timing of the circuit depends upon the layout. An asynchronous bundled-data pipeline for the matrix–vector multiplication core of discrete cosine transforms achieved 30% higher average throughput [16] based on a bit-partitioned carry-save multiplier. Due to its bulky control overhead, the controller drops its speed gain. These methods intend to improve the power consumption/performance of IC's using asynchronous methodology.

The basic block structure of the asynchronous circuit is given in Fig.1.1. The data is processed to next step through register on demand of request signal. The next request signal issued only on acknowledge of the previous process completion. This way clock is eliminated by the acknowledge signal for each process of data.

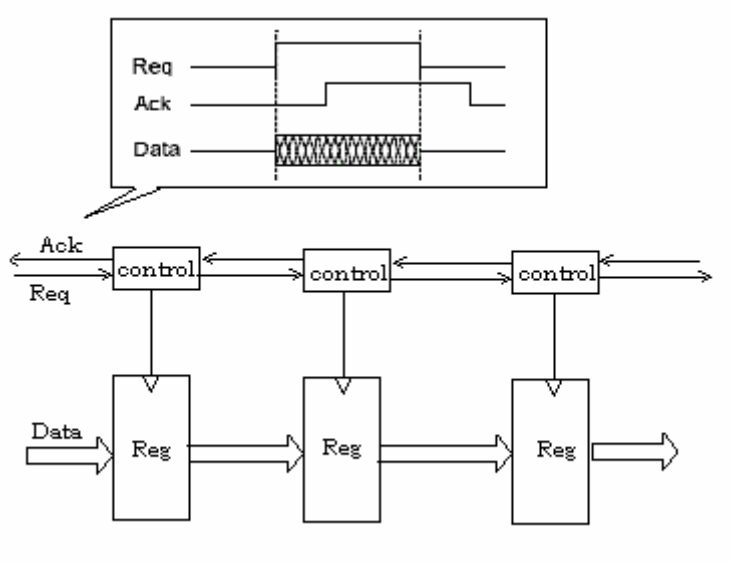


Fig.1.1 Basic block diagram of an Asynchronous Circuit.

1.1 Motivation of the work

Nowadays it is essential that each application instrument of the electronic industry should be compact, consume less power and less delay performance. Researches are continuing to reduce the above three parameters especially in battery operated computing devices. The sources of energy consumption on a CMOS chip can be classified as static and dynamic power dissipation [18] [19]. The dominant component of energy consumption in CMOS is dynamic power

consumption caused by the actual effort of the circuit to switch. A first order approximation of the dynamic power consumption of CMOS circuitry [17] is given by the formula:

$$P = C * V^2 * f$$

where P is the power, C is the effective switch capacitance, V is the supply voltage, and f is the frequency of operation. The power dissipation arises from the charging and discharging of the circuit node capacitances found on the output of every logic gate. Every low-to-high logic transition in a digital circuit incurs a change of voltage, drawing energy from the power supply.

A designer at the technological and architectural level can try to minimize the variables in these equations to minimize the overall energy consumption. However, power minimization is often a complex process of trade-offs between speed, area, and power consumption [23]. This difficulty can be overcome by choosing the asynchronous design methodology where the dynamic power is almost avoided since there is no global clock. Also the transistors will be used at the time of demand only, which is unlikely with synchronous designs.

Most digital circuits designed and fabricated today are " synchronous". In essence, they are based on two fundamental assumptions that greatly simplify their design: (1) all signals are binary, and (2) all components share a common and discrete notion of time, as defined by a clock signal distributed throughout the circuit.

Asynchronous circuits are fundamentally different; they also assume binary signals but there is no common discrete time. Instead the circuits use handshaking between their components in order to perform

the necessary synchronization, communication, and sequencing of operations. Expressed in ‘ synchronous terms’ this results in a behaviour that similar to systematic fine-grain clock gating and local clocks that are not in phase and whose period is determined by actual circuit delays-registers are only clocked where and when needed [4].

The difference gives asynchronous circuits inherent properties that can be exploited to advantage in the areas listed and motivated below. Further its having the following advantages.

1. Lower power consumption, due to fine-grain clock gating and zero standby power consumption [11]
2. High operating speed, since operating speed is determined by actual local latencies rather than global worst-case latency. [14] [39]
3. Less emission of electro-magnetic noise, the local clocks tend to tick at random points in time. [39] [42]
4. Robustness towards variations in supply voltage, temperature, and fabrication process parameters, since timing based on matched delay and it can be insensitive to circuit and wire delays[40].
5. Better composability and modularity, because of the simple handshake interfaces and the local timing [36].
6. No clock distribution and clock skew problems, since there is no global signal that needs to be distributed with minimal phase skew across the circuit[37].

Fig 1.2(a) shows synchronous circuit. For simplicity the figure shows a pipeline, but intended to represent any synchronous circuit. When designing ASICs using hardware description languages and synthesis tools, designers focus mostly on tha data processing and assume the global clock. For example, a designer would express the fact that data clocked into register R3 is a function CL3 of the data clocked into R2 at the previous clock as the following assignment of variable: R3 : = CL3(R2). Figure 1.2(a) represents this high-level view with a universal clock.

When it comes to physical design, reality is different. Today's ASICs use a structure of clock buffers resulting in a large number of(possibly gated) clock signals as shown in Fig. 1.2(b). It is well known that it takes CAD tools and engineering effort to design the clock gating circuitry and to minimize and control the skew between the many different clock signals. In synchronous designs the problem of guaranteeing the two-sided timing constraints-the set up to hold time window around the clock edge-in a world that is dominated by wire delays is not an easy task. The buffer-insertion-and-resynthesis process that is used in current commercial CAD tools may not converge and, even if it does, it relies on delay models that are often questionable accuracy.

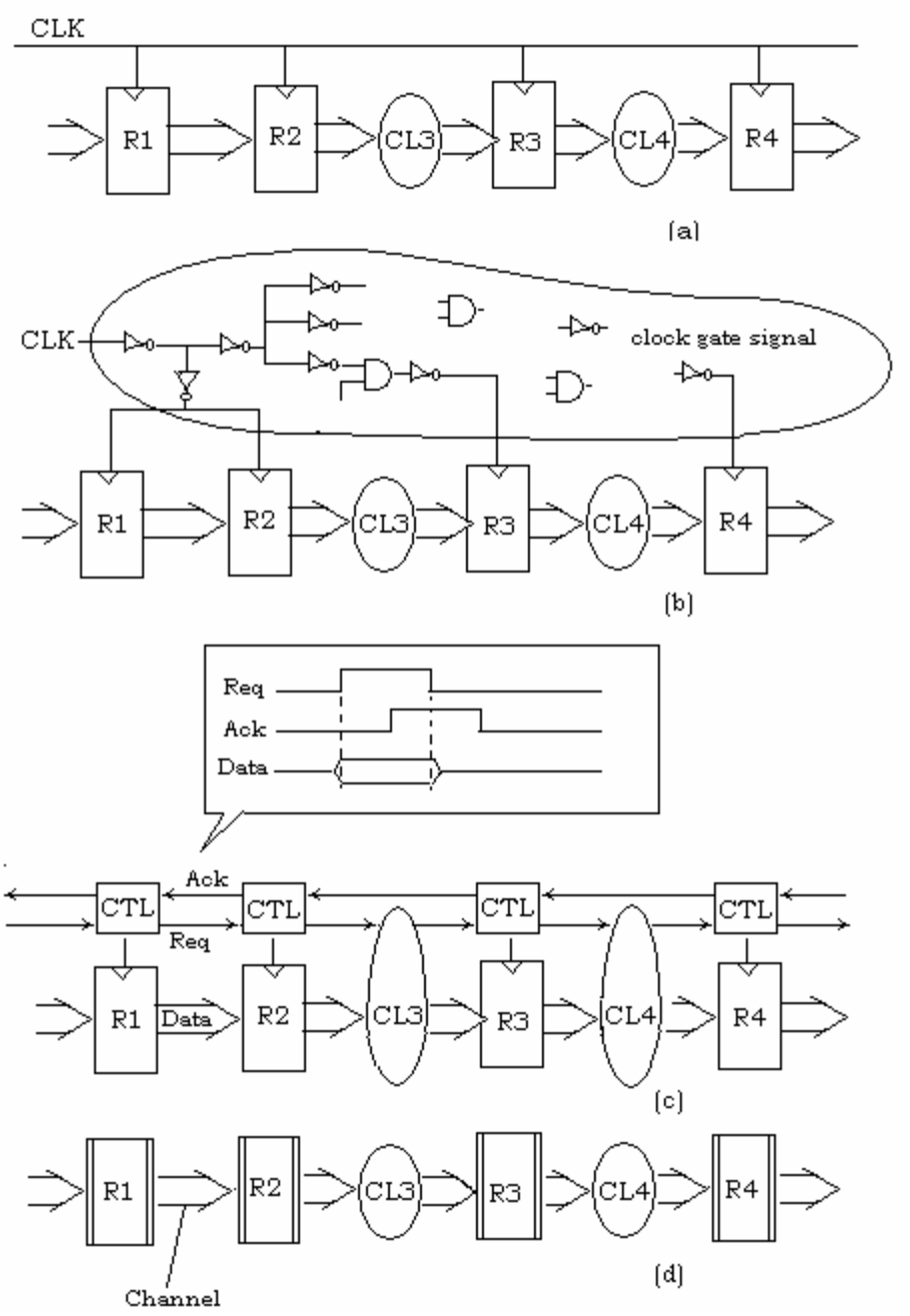


Fig. 1.2 (a) A synchronous circuit, (b) a synchronous circuit with clock drivers and clock gating, (c) an equivalent asynchronous circuit, and (d) an abstract data-flow view of the asynchronous circuit.

Asynchronous design provides an alternative solution to this limitation. In an asynchronous circuit the clock signal is replaced by some form of handshaking between neighbouring registers [4]; for

example the simple request-acknowledge based handshake protocol shown in Fig. 1.2(c). In the second chapter we look an alternative handshake protocols and data encoding, but before departing into those implementation details it is useful to take a more abstract view as illustrated in figure 1.2(d).

- The data and handshake signals connecting one register to the next in Fig. 1.2(c) as a " handshake channel" or "link,"
- The data stored in the registers as tokens tagged with data values (that may be changed along the way as tokens flow through combinational circuits), and
- The combinational circuits as being transparent to the handshaking between registers; a combinatorial circuit simply absorbs a token on each of its input links, performs its computation, and then emits a token on each of its output links.

Viewed this way, an asynchronous circuit is simply a static data-flow structure. Intuitively, correct operation requires that data token flowing in the circuit do not disappear, that one token does not overtake another, and that new tokens do not appear out of nowhere. A simple rule that can ensure this is the following [36]:

A register may input and store a new data token from its predecessor if its successor has input and stored the data token that the register was previously holding. [The states of the predecessor and successor registers are signaled by the incoming request and acknowledge signals respectively.]

Following this rule data is copied from one register to the next along the path through the circuit. In this process subsequent registers will often be holding copies of the same data value but the old duplicate data values will later be overwritten by new data values in a carefully ordered manner, and a handshake cycle on a link will always enclose the transfer of exactly one data-token. Understanding this "token flow game" crucial to the design of efficient circuits, and we will address these issues, extending the token-flow view to cover structures other than pipelines.

An important thing is that the "handshake-channel and data-token view" represents a very useful abstraction that is equivalent to the register transfer level (RTL) used in the design of synchronous circuits [47]. This data-flow abstraction, as we call it, separates the structure and function of the circuit from the implementation details of its components.

Another important consideration is that it is the handshaking between the registers that controls the flow of tokens, whereas the combinatorial circuit blocks must be fully transparent to this handshaking. Ensuring this transparency is not always trivial; it takes more than a traditional combinational circuit, so we will use the term "function block' to denote a combinational circuit whose input and output ports are handshake-channels or links. The synchronous circuit shown in Fig. 1.2(b) is "controlled" by clock pulses that are in phase with a periodic clock signal, whereas the asynchronous circuit in Fig. 1.2(c) is controlled by locally derived clock pulses are generated where and when needed. This tends to randomize the clock pulses over time, and is likely to result in less electromagnetic emission and a

smoother supply current without the large di/dt spikes that characterize a synchronous circuit.

we consider an approach of asynchronous ALU design to reduce the transistors count, power consumption and delay. We propose the application of delay insensitive dual rail logic and bundled data bounded delay model in our design. Since the proposed approach will have the all advantages of asynchronous circuit earlier discussed will be very useful in power consumption and occupying less space with average delay in battery operated devices.

1.2 Brief Overview of our Work

In the present work, we consider an approach of asynchronous ALU design to reduce the transistors count, power consumption and delay. We propose the application of delay insensitive dual rail logic and bundled data bounded delay model in our design. To achieve higher performance and lower power operation, we design the circuits with CMOS domino logic. The use of domino logic reduces the transistor count, parasitic capacitances and ensures glitch-free circuit. The Muller C-element and Four-Phase Dual Rail Protocol are used for the completion detection.

This design uses conventional CMOS domino logic since its implementation supports the glitch free circuit and the capacitance of its output node is separating by interval and load capacitance [2]. Also it ensures the lower power consumption by reducing the parasitic capacitances and transistor count. Asynchronous circuits are fundamentally different from the synchronous counterpart and use

handshaking among components to perform the necessary synchronization, communication and sequencing of operations. The handshaking implementation may follow any one of these protocols, 4-phase bundled data, 2 –phase bundled data or 4-phase dual rail. In all protocols, Muller pipeline is used. The 4-phase dual rail has designed to combine encoding of data and request. We apply it in our circuit, because 4- phase dual rail protocol provides reliable synchronization, lower power consumption with simple and faster signal transition than 2-phase model [9].

The asynchronous ALU is implemented with 4-phase dual rail protocol and CMOS domino logic for single bit operation. The 32 bit ALU can be extended on concatenation of the same circuit with 32 blocks. The completion detection circuit detects each operation completion and sends back the necessary acknowledge signal. The power is consumed only at the time operation only, which is unique advantage of our asynchronous design than synchronous using master clock [43].

1.3 Organization of the Thesis

The design architecture and results are presented in the dissertation in the following five chapters.

Chapter 1: Motivation and a brief overview of our work is presented.

Chapter 2: Introduction to asynchronous design methodologies, designing the low power circuits and C-Muller pipelines.

Chapter 3: We consider our work to design an efficient asynchronous VLSI architecture design of low power ALU with four-phase dual rail protocol.

Chapter 4: In this chapter, simulation results through SPICE design tool are reported and compared with other published works.

Chapter 5: Summary of the work and future scope to extend our work is given.

Chapter 2

CMOS logics and Asynchronous Design Methodologies

In this chapter, we are discussing about the fundamentals of CMOS logic design styles namely static and dynamic CMOS design followed by the Asynchronous design methodologies available in detail. Our proposed Asynchronous ALU circuit is following the dynamic domino CMOS logic and 4-phase dual-rail protocol among the different logic styles available. Also in the final part of this section, the base of any asynchronous circuit C-Muller element and its pipeline, the 4-phase dual rail pipeline follows that are discussed.

2.1 Static CMOS

Although static CMOS logic is widely used for its high noise margins and relative ease of design, it is limited at running extremely high clock speeds. For applications requiring the fasted circuit speeds possible, dynamic CMOS logic has numerous advantages over static CMOS including not only higher speeds but also significantly reduced surface area. The advantages do not come without a cost however. Due to the nature of dynamic CMOS logic, undesired effects can occur within the circuit unless extra effort is put into the engineering design. Understanding the basic principles of Dynamic CMOS logic begins with first an understanding of the basic properties of MOSFET devices as well as the characteristics of static and pseudo-NMOS logic [23].

Due to the internal structure of MOSFET devices, an effective capacitance can be associated across all possible terminal

combinations of the gate, drain, source, and body. When charge is applied to these capacitances, the corresponding terminal voltage rises, and when the charge is removed, the terminal voltage decays just as if the terminal were modeled as a capacitor. Modeling the MOSFET terminals as capacitors is useful to explain the voltages and currents associated with the MOSFETs in a complex circuit. MOSFETs are characterized by the three modes of operation: Cutoff, Linear, and Saturated [27].

However, since current flows through the device for both the linear and saturated modes, it is useful to consider the MOSFET as ON in this conducting state or OFF when no current flows. For an NMOS, or n-channeled MOSFET, the device is only ON when the gate to source voltage, V_{SG}, is greater than the device threshold voltage, V_T.

For a PMOS, or p-channeled MOSFET, the device is only ON when the source to gate voltage, V_{SG}, is greater than the negative device threshold voltage, $-V_T$. For the purposes of this analysis, the input to the gates of the MOSFETs will either be high or low, V_{DD} or GND, respectively. Therefore, if the PMOS source is connected to V_{DD}, the PMOS will only be ON if the gate voltage is low. Likewise, if the NMOS source is connected to GND, the NMOS will only be ON when the gate voltage is high. Observing how the NMOS and PMOS work in conjunction to form the CMOS inverter circuit, Fig. 2.1, is a useful example to understand how these devices might be used in more complex circuitry.

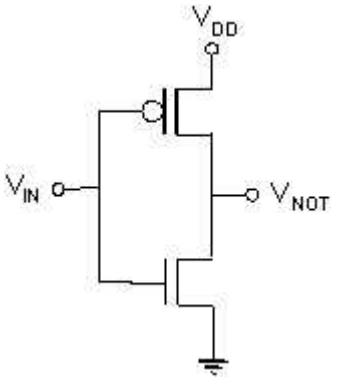


Fig.2.1 CMOS inverter circuit

When the input is low, the PMOS turns ON and the NMOS turns OFF. The output is simultaneously cut off from GND and charged high due to the 'pull-up' path to V_{DD} through the PMOS. Conversely, when the input is high, the PMOS turns OFF and the NMOS turns ON resulting in a 'pull-down' path to GND while the connection to V_{DD} is cutoff. When utilized in this fashion, the NMOS device is considered a 'pull-down' device, and the PMOS is considered a 'pull-up' device [23].

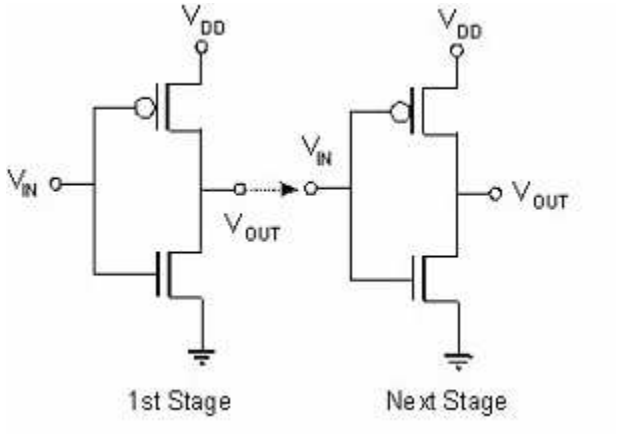


Fig.2.2 CMOS inverter Circuits in stages

By connecting the output of this circuit to the input of similar logic, the voltage and current characteristics can be determined by considering the capacitive effects associated with the input of this second stage. As shown in Fig. 2.2, a low input to the CMOS inverter charges the input of the second stage high due to the current from V_{DD} flowing out of the first-stage PMOS. A high input to the inverter removes any charge at the input of the second stage through the NMOS of the first stage.

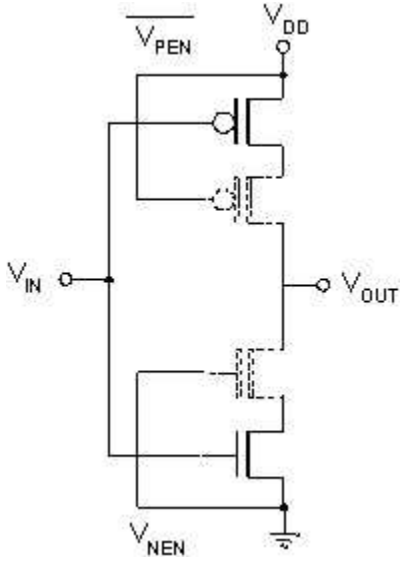


Fig.2.3 CMOS tristate inverter

The capacitive effects of the MOSFET terminals can also be used to store charge across the terminals temporarily. Consider the tristate inverter in Fig. 2.3. When V_{PEN} and V_{NEN} are low (note V_{PEN} is inverted), Vout is disconnected from both V_{DD} and GND leaving Vout floating. In this 'Z-state,' Vout must retain its previous voltage level. Ideally, any charge associated with the Z-state would remain across the terminals of the MOSFET indefinitely; however, due to parasitic

charge leakage, an originally high voltage in the output Z-state will decay to zero with time. If the system were run at speeds higher than the time needed for the leakage current to cause a logic error, the characteristics of the output Z-state can be utilized to vastly increase circuit speeds. This is the essence of dynamic CMOS.

Although there are many positive reasons for using static CMOS logic, there are also numerous drawbacks. Static devices inherently have more components and clocked transistors than dynamic devices. A full latch for example in the traditional static configuration may require 66 transistors [21]. A dynamic configuration performing the same function may require only 36 transistors [21]. The number of transistors used to construct a flipflop is also significantly reduced by using dynamic logic as opposed to fully static logic. Reducing the total number of transistors not only allows the overall device to be significantly smaller, but also reduces the power requirements of the system [20].

Most of the disadvantages of using static CMOS, however, are associated with the use of PMOS. Caused in part because hole mobilities are significantly slower than electron mobilities, PMOS devices must be much larger than NMOS devices for the two to have the same ability to transport a fixed amount of charge during a fixed time interval. The larger surface area needed to form a PMOS device than an NMOS device is not only a detriment to the overall chip size, but also increases the capacitance associated to the PMOS device. The larger capacitance and slower carrier mobilities associated with PMOS cause a greater time delay for the PMOS to charge up the capacitor associated with the next logic stage. This increased time delay

becomes a bottleneck when trying to design faster circuits. In standard CMOS logic, one PMOS device will always compliment an NMOS device. Altering this logic so that fewer PMOS devices are needed will vastly improve circuit performance.

One method to decrease the number of PMOS devices in the circuit is to use what is called pseudo-NMOS logic. Instead of using one PMOS for every NMOS device, pseudo-NMOS logic utilizes only one PMOS device as a load to all other NMOS logic as shown in Fig.2.4.

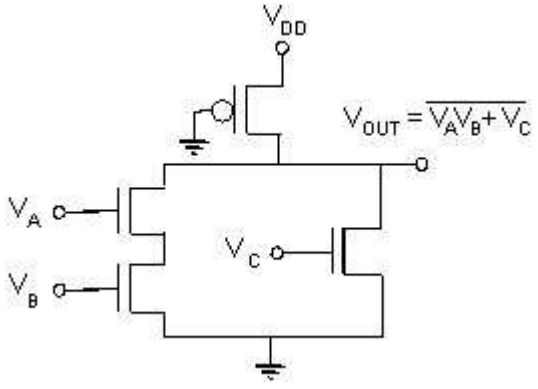


Fig. 2.4 pseudo- NMOS logic

Since the voltage at the gate of the PMOS is always GND, the PMOS device is always ON. The output then of the pseudo-NMOS circuit is selectively discharged to GND through the NMOS logic block. Since the NMOS devices in the ON state forms a pull-down path to GND and the PMOS device is always ON, there will be times during circuit operation where a path is formed from VDD to GND. The pseudo-NMOS logic must be ratio sensitive so as to minimize the loss in power dissipation. In other words, the PMOS must be ‘weak’ or small so as to have less capacitance associated with the device. In this

configuration, the charge will be pulled up much more slowly by the PMOS than it can be discharged through the NMOS devices. In this way, a pull-down path to ground through the NMOS logic block should easily pull down the output. When no pull-down path to ground exists via the NMOS logic, the output is then pulled high through the PMOS load. Although pseudo-NMOS logic can be utilized to reduce the number of PMOS components in the system, not only does the static power dissipation serve as a detriment, but the speed of the circuit is limited by the time necessary for the weak PMOS to charge up the output node [27].

2.2 Dynamic CMOS

An alternative logic that reduces the number of PMOS devices while also solving most of the problems associated with pseudo-NMOS logic is dynamic CMOS. The basic structure of dynamic CMOS logic is shown in Fig. 2.5.

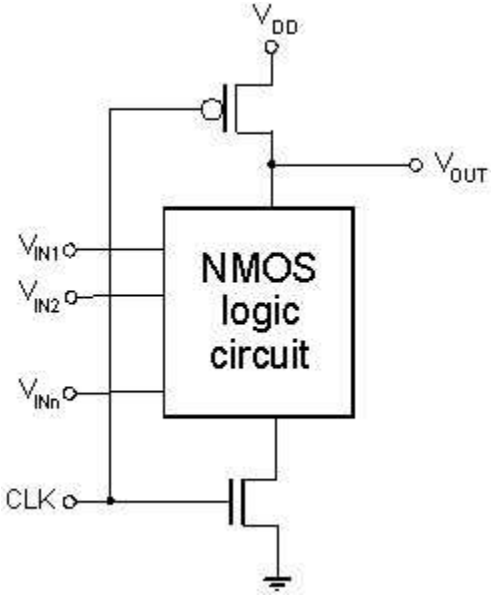


Fig. 2.5 Basic Structure of dynamic CMOS logic.

When the clock is low, the NMOS device is cut off while the PMOS is turned ON. This has the effect of disconnecting the output node from ground while simultaneously connecting the node to V_{DD}. Since the input to the next stage is charged up through the PMOS transistor when the clock is low, this phase of the clock is known as the 'precharge' phase. When the clock is high however, the PMOS is cutoff and the bottom NMOS is turned ON, thereby disconnecting the output node from V_{DD} and providing a possible pull-down path to ground through the bottom NMOS transistor. This part of the clock cycle is known as the 'evaluation' phase, and so the bottom NMOS is called the 'evaluation NMOS.' When the clock is in the evaluation phase, the output node will either be maintained at its previous logic level or discharged to GND. In other words, the output node may be selectively discharged through the NMOS logic structure depending upon whether or not a path to GND is formed due to inputs of the NMOS logic block [23]. If a path to ground is not formed during the evaluation phase, the output node will maintain its previous voltage level since no path exists from the output to V_{DD} or GND for the charge to flow away.

As an example, the Pseudo-NMOS circuit shown in Fig. 2.4 can be made into a dynamic logic structure by adding an evaluation NMOS and connecting it to a clock with the PMOS as shown in Fig. 2.6.

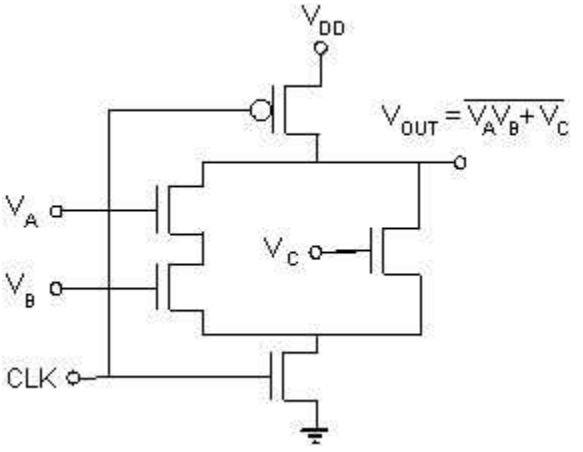


Fig. 2.6 example for dynamic CMOS logic

During the precharge phase, the output is pulled high through the PMOS in the ON state. When the clock goes high in the evaluation stage, the output will be data-dependent. If the input signals A AND B are high OR if C is high, a path to ground through the evaluation NMOS will be formed and the output node will be pulled low. If these conditions are not met, then the output will remain high. Regardless of the resultant logic level of the output node at the end of the evaluation phase, the output node will be pulled high again when the clock goes low for the next precharge phase.

There are many advantages of using dynamic CMOS logic over static CMOS logic or Pseudo NMOS logic. The elimination of the complimentary PMOS transistors significantly reduces the transistor count needed to implement the various logic functions not only because the number of transistors is nearly half, but because the physical size of the PMOS transistors tend to be much larger than the size of an NMOS transistor. The switching speeds are also increased

using the dynamic logic configuration since the speed bottleneck caused by the lengthier time the PMOS requires to pull-up the output node is eliminated. Since this node is already precharged high through the PMOS during the precharge phase, the output node needs only to be selectively discharged during the evaluation phase. Discharging the output node through the NMOS devices is significantly faster than the time needed to charge up the output node through the PMOS device [22].

Although increased speed over static or Pseudo NMOS logic is a significant achievement in the dynamic logic, there are several potential problems with the implementation of this design that need to be considered. Since the basic dynamic CMOS logic configuration causes the output node to be disconnected from V_{DD} during the evaluation phase, even if the output is also disconnected from GND, the charge of the output node will begin to diminish due to the non-ideal effects of the system. Parasitic capacitances, for example, may leak the charge away from the output node and eventually cause a logic error [8]. Since there is, however, a finite time needed for the charge to erroneously escape, the use of faster the clock speeds will eliminate this kind of error. This implies however, that there is a minimum clock speed at which dynamic CMOS logic structures may be operated. It also eliminates the possibility to idle the basic dynamic CMOS logic circuit.

These drawbacks however, are not without a solution. In many cases, the specifications of the system do not require the circuit to ever idle or run at relatively slow clock speeds. In these cases, the fastest clock speed is desired, making the minimum clock speed of the dynamic

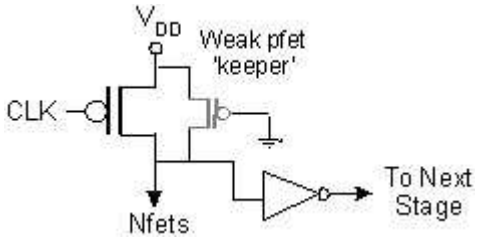


Fig. 2.7 Dynamic CMOS logic structure for minimum clock speed

logic configuration a non-issue [13]. In other cases, some of the static benefits can be introduced to the dynamic logic configuration with the addition of a ‘weak’-PMOS device added between the output node and V_{DD} as shown in Fig. 2.7. If the gate is connected to GND, this PMOS device will always be turned ON. Then, even in the evaluation phase, the output node will be connected in some capacity to V_{DD}. This PMOS, the ‘keeper,’ has the effect of maintaining the output node charge even at slower clock speeds. The keeper transistor is designed to be weak enough so that a path to GND through the NMOS logic block during the evaluation phase will significantly overpower the effects of the keeper PMOS and easily pull the output node to GND. Although this configuration has advantages, it does introduce another PMOS device into each stage and also causes excess power dissipation due to possibility of the connection from V_{DD} to GND through the NMOS devices and the PMOS keeper. When such a circumstance occurs, the NMOS and PMOS must ‘fight’ each other to pull-up or pull-down the output through VDD or GND respectively, and power is lost. For high-performance circuits, an alternative is clearly needed .

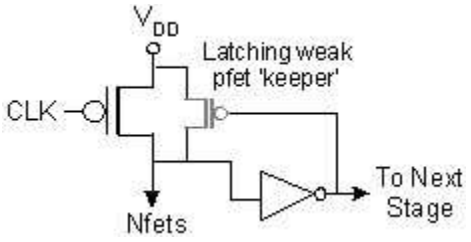


Fig. 2.8 Latching of weak PMOS in dynamic logic

The use of a keeper PMOS in dynamic logic could be further improved by connecting the gate of the keeper not to GND, but to the output node of the inverter stage as shown in Fig. 2.8. The keeper would now function as a latch cutting off whenever the output of the inverter is high. In this way, power dissipation is significantly reduced whenever a pull-down path to GND has been formed in the NMOS logic block since this would make the input to the inverter low and thus the output of the inverter high. When the output of the inverter is low however, as would be the case if no pull-down path to ground was formed in the NMOS logic block, the keeper PMOS would turn on and maintain the output high charge on the precharge node even at reduced clock speeds or an idle [17].

Other characteristics of dynamic CMOS logic that must be taken into consideration when designing dynamic logic are the problems that can occur when cascading the dynamic logic blocks [25]. Due to the finite pull down time of the NMOS logic block, during the very first portion of the evaluation phase, the output will always register an output high state for at least a brief moment in time before the output charge can be removed via the pull-down path to GND. This is considered a ‘racing’ problem since the logic is evaluated correctly only when the time to pull down the output node is

faster than the time needed for the briefly high output caused by the precharge phase to propagate as an erroneous logic signal to the next stage. Since the output node of one dynamic CMOS logic block is connected to an input of the next dynamic CMOS stage, an output high state however brief could complete a pull-down path to GND in the following stage and erroneously cause a discharge in the output of this next stage. Since the charge on the output node cannot be recovered until the next precharge phase, the logic error would remain and propagate through the system. Dynamic CMOS logic blocks should therefore not be directly cascaded. Note that care must also be taken to insure that the input logic signals to the NMOS logic block are correct and stable for the complete duration of the evaluation stage or a similar logic error could occur [17].

2.2.1 Domino Logic

The errors occurring due to cascaded dynamic logic blocks can be overcome by adding an inverter stage between the output of one stage and the input of another as in Fig. 2.9.

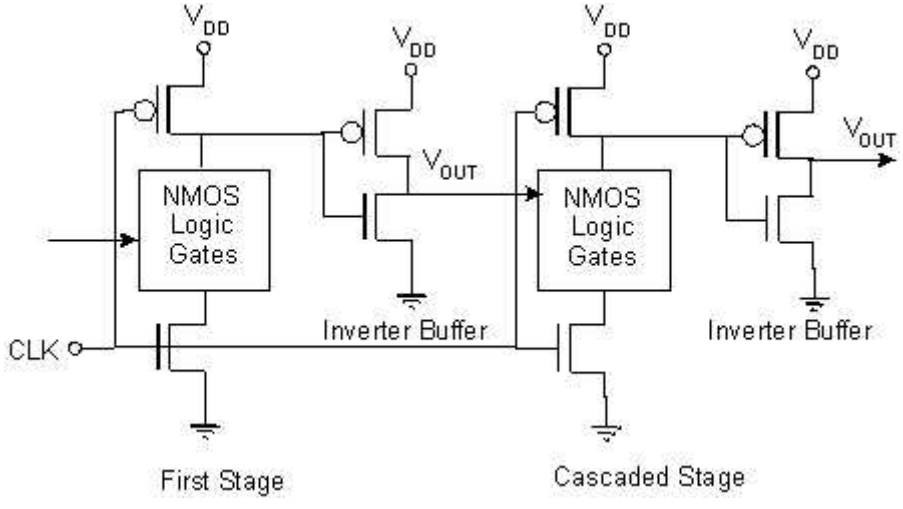


Fig. 2. 9 Dynamic CMOS domino logic structure

This inverter then would start out low at the very beginning of the evaluation phase. The output low state of the inverter would cutoff the NMOS logic gates in the next stage preventing any erroneous pull-down path. If a pull-down path is formed by the NMOS logic block of the first stage, the output of the inverter buffer would conditionally charge from low to high. Only if the inputs to the first stage NMOS logic block warrant a discharge of the output node would the output inverter make the low to high transition. When the output of this inverter buffer goes high, the following stage of NMOS logic would conditionally form a pull-down path to ground. In this way, the addition of the inverter buffer eliminates any logic errors caused by the finite pull-down time of the NMOS logic block. This kind of design is referred to as Domino Logic since the pull-down of one stage can conditionally cause the pull-down of succeeding stages and so on like falling dominoes [26].

The number of Domino logic stages that may be cascaded is limited only by the sum of the total pull-down times in all cascaded logic blocks, which must be contained within the evaluation clock phase. Drawbacks to this design are of course the addition of two additional components to each dynamic block. Extra design consideration must also be observed when using dynamic CMOS logic blocks in conjunction with static CMOS logic blocks. Since the final output to the Domino logic blocks is the inverted form of the original output due to the additional inverter buffer stage, only non-inverting logic may be used between the output and input of dynamic logic. That is, since the inverter must make only one conditional state change from logic low to high (not high to low) during the evaluation phase only an

even number of static logic blocks may be used in between dynamic logic blocks [24].

2.2.2 NORA Logic

An alternative to Domino Logic is NORA or Domino-Zipper Logic. NORA stands for 'no-race,' indicating another method to eliminate the 'racing' problem of directly cascaded dynamic logic blocks. Fig. 2.10 depicts the basic structure of NORA logic which is characterized by alternating the MOSFETs in the logic block from PMOS to NMOS logic gates and so on. Note that the function of the clocked n- and p- FETs in the PMOS logic stage are reversed compared to the NMOS logic stage [23].

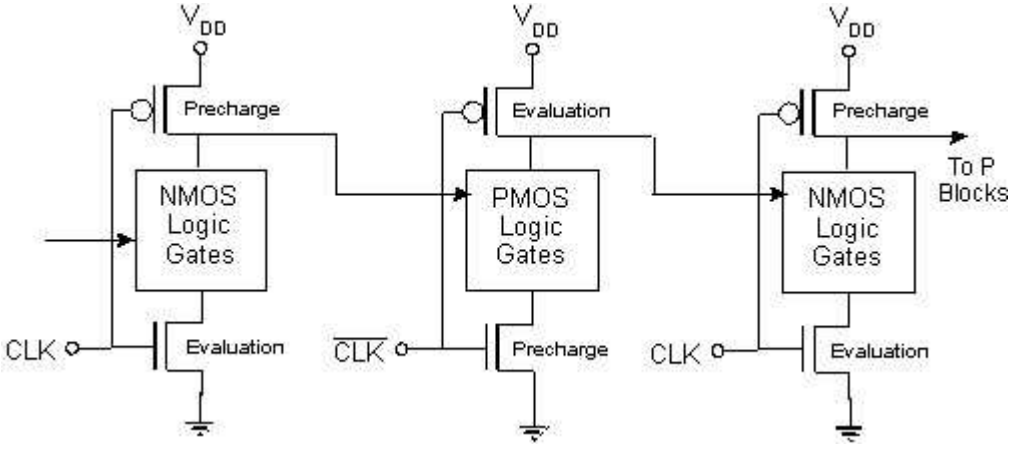


Fig. 2.10 Dynamic CMOS NORA logic structure

Although this structure eliminates the cascading problem, the excess use of PMOS in forming the logic gates reduces the maximum clocking speed and increases the surface area of the system. For this reason, it is preferable to use only the NMOS for the logic gates and leave the PMOS as precharge elements. Further design considerations for NORA logic are needed when combining the dynamic NORA

blocks with static blocks. As observed with Domino Logic, the output may only be allowed to change from low to high once during the evaluation phase of the NMOS logic and visa-versa for the PMOS, so only an even number of static blocks may be used in between two of the dynamic blocks.

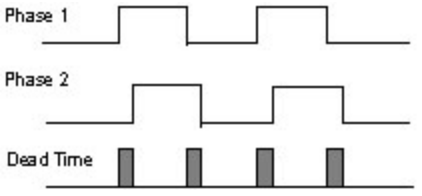


Fig. 2.11 clock skew representation

Another significant drawback to this configuration is the use of the two-phase clock. For a circuit operating at high speeds, the clock characteristics become increasingly important. The signals of both clock phases must be delivered at nearly the same instant for the circuit to operate correctly. Routing a one phase clock to the millions of circuit elements such that the delay is minimized is a challenging design issue in and of itself. Routing a second clock phase to a similar quantity of circuit elements such that the delay is minimized compared not only to itself, but to the first clock phase becomes a serious problem. The time delay between the first and second clock phase is known as clock skew. The presence of clock skew in a circuit reduces the maximum operation speed of that circuit, since the logic cannot be correctly evaluated during this delay time [8]. Clock skew, as shown in Fig. 2.11, can be eliminated by using only one clock phase.

2.3 Handshake Basics

In Fig 1.2(c) one particular handshake basic protocol has been explained, known as the 4-phase bundled-data protocol also called as return-to-zero handshake protocol [4]. The below sections explain the basic principles for designing the handshaking (asynchronous) circuits.

2.3.1 Principles of Bundled-data protocols

In bundled data protocols the data signals use normal Boolean levels to encode information and separate request and acknowledge wires are bundled with the data signals. In Fig. 2.12, a bundled data channel is shown in which data is bundled with request and acknowledge wires.

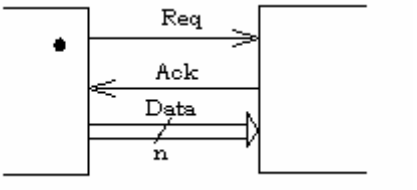


Fig.2.12 A bundled-data channel

The 4-phase protocol is illustrated in Fig. 2.13. Here the request and acknowledge wires use normal Boolean levels to encode information, and the term 4-phase refers to the number of communication actions [4]:

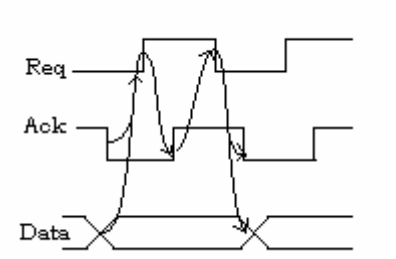


Fig.2.13 A 4-phase bundled-data protocol

(1) the sender issues/sends Data and sets Req High,

(2) the receiver absorbs/receives the Data and sets Ack High,

(3) the sender responds by taking Req Low and

(4) the receiver acknowledges this by taking Ack Low.

The sender may initiate the next communication cycle once all above 4 phases are over.

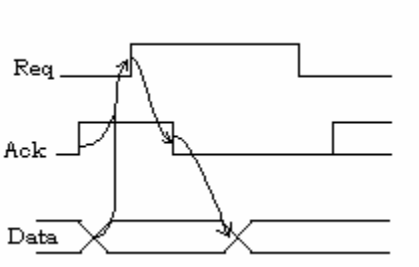


Fig.2.14 A 2-phase bundled data protocol.

The 2-phase bundled data protocol is shown in Fig. 2.14. The information on the request and acknowledge wires is now encoded as signal transitions on the wires and there is no difference between a 0 to 1 and a 1 to 0 transition, they both represent a “signal event”. Ideally the 2-phase bundled-data protocol should lead to faster circuits than the 4-phase bundled-data protocol, but often the implementation of circuits responding to events is complex [54].

The term ‘bundled data’ hints at the timing relationship between the data signals, whereas the term ‘single rail’ hints at the use of one wire to carry one bit of data [58]. The protocols introduced above all assume that the sender is the activity party that initiates the data transfer over the channel. This is known as a push channel. The opposite, the receiver asking for new data, is also possible and is called a pull channel. In this case the directions of the request and acknowledge signals are reversed, and the validity of the data is indicated in the acknowledge signal going from the sender to receiver. In the abstract diagram, the active end of the channel is marked with a dot.

2.3.2 4-phase dual-rail protocol

The 4-phase dual rail channel is shown in Fig.2.15.In this handshake protocol, it encodes the request signal into the data signal by using 2 wires per bit of information that is to be communicated. This 4-phase dual-rail protocol uses two request wires per bit of information a;

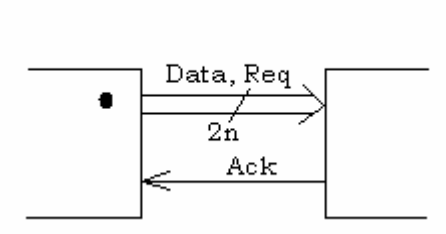
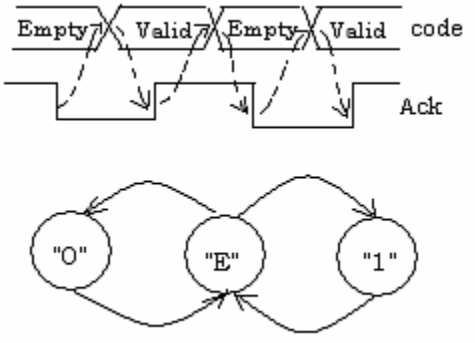


Fig.2.15 The 4-phase dual rail channel

one wire a.t is used for signaling a logic 1(true), and another wire a.f is used for signaling logic 0(false). Thus in this 1 bit channel one can understand a cycle of 4-phase handshakes in which the “request” signal in any handshake cycle can always be at either a.t or a.f. Due to

its robustness and delay insensitive nature, two parties thus can reliably do communication regardless of delays in the wires connecting the two parties [49].



code	a.t	a.f
Empty("E")	0	0
Valid "0"	0	1
Valid "1"	1	0
Not used	1	1

Fig.2.16 The 4-phase dual rail protocol handshaking

For an information a, the {a.f,a.t} wire pair is a codeword; {a.f,a.t}={1,0} and {a.f,a.t}={0,1} represents “valid data” (logic Low and logic High respectively) and {a.f,a.t}={0,0} represents “empty” (“no data” or “NULL”). The codeword {x.f,x.t}={1,1} is not used , and here the important is a transition from one valid codeword to another valid codeword is invalid and not allowed. Fig. 2.16 illustrates this protocol operations. Now, one can understand the 4-phase handshaking [56]:

(1) the sender issues/sends a valid,

(2) the receiver absorbs/receives the codeword and sets the Ack. High,

(3) the sender responds by issuing the empty codeword, and

(4) the receiver acknowledges this by taking acknowledge Low.

Next communication cycle starts after this above cycle. In the abstract view, the channel is a data stream of valid codewords separated by empty codewords.

Thus, an N-bit data channel is done by concatenating N wire pairs, each using the encoding description above and a receiver which always be able to detect when all bits are valid i.e by it responds by giving acknowledge high. Also the receiver responds by giving acknowledge low for the condition when all bits are empty. The dual rail code has following unique properties [59]:

(1) concatenation of dual-rail codewords is also a dual-rail codeword.
(2) the set of all possible codewords can be disjointly divided into 3 sets, for a given N number of bit,

(a) the empty or null codeword where (all N)wire pairs are {0,0}.

(b) the intermediate codewords where some wire-pairs assume the empty state and some wire pairs assume valid data(as protocol definition).

(c) 2^N different valid codewords.

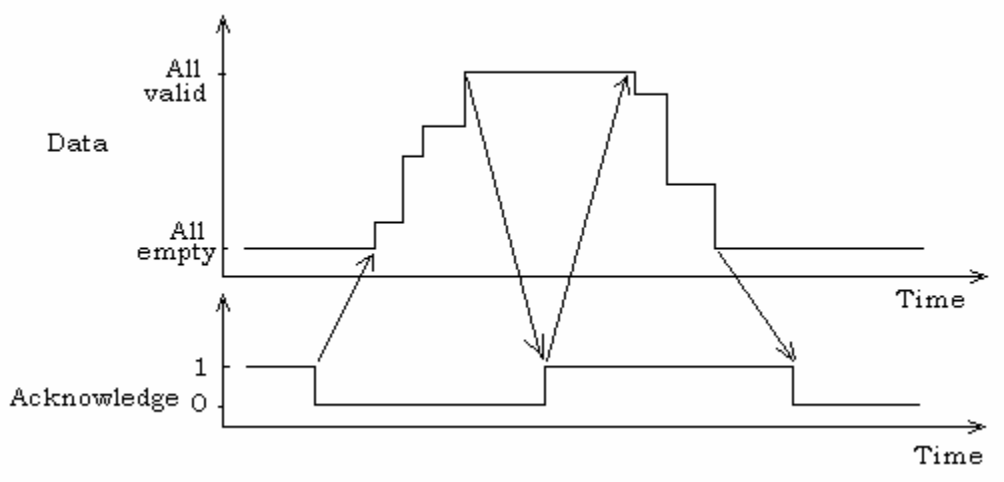


Fig. 2.17. Illustration of 4-phase dual-rail channel handshaking

Simple illustration of the handshaking of an N-bit channel is shown in Fig. 2.17. At receiver end one will see the empty codeword,

a sequence of intermediate codewords and eventually a valid codeword. On reception and acknowledging the codeword, the receiver will see a sequence of intermediate codewords, and eventually the empty codeword to which the receiver responds by setting acknowledge low and the cycle will move on.

2.3.3 2-phase dual-rail protocol

In this 2-phase dual-rail protocol just like 4-phase dual-rail protocol, it also uses 2 wires {a.t,a.f }per bit to communicate, but the information is encoded as transitions(events).In an N-bit channel a new codeword will be received if exactly one wire in each of the N wire pairs has made a transition(event). If there is no empty value then a valid message is acknowledged and followed by another message that is acknowledged. The below Fig. 2.18. shows simple illustration of the signal waveforms on a 2-bit channel for the 2-phase dual-rail protocol [4].

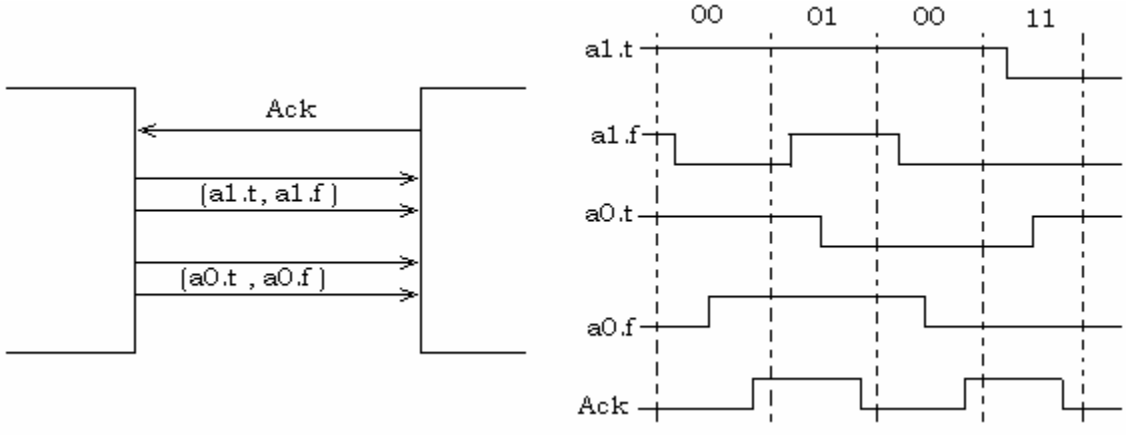


Fig. 2.18. Illustration of 2-phase dual-rail protocol handshaking

2.4 Indication Principle and the Muller C-element

In synchronous circuits, valid and stable signals found at every clocking points. But in between the clock-ticks, the signals may not be stable and usually exhibits hazards and may make multiple transitions as the combinational circuits stabilize. But in the case of asynchronous circuits there is no clock.

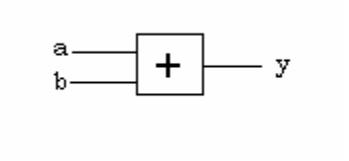


Fig. 2.19 OR gate

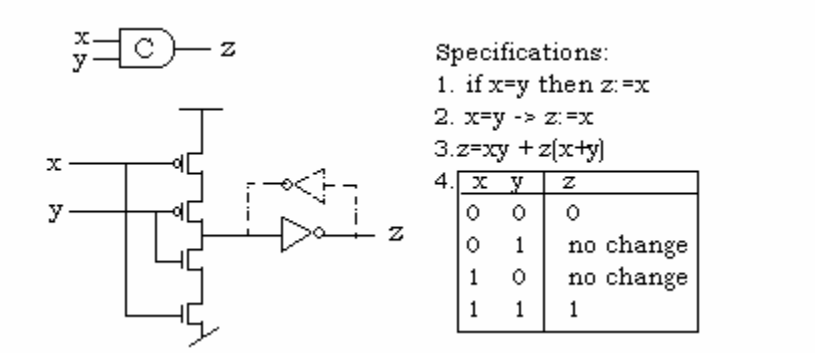
a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

Table 2.1. Truth table-OR gate

Thus the signals should be valid all the time and every signal transition should have a meaning. Consequently those hazards and races during the transitions must be avoided. Hence the concept of indication or acknowledgement plays vital role in designing asynchronous circuits. The simple 2-input OR gate and truth table are shown in Fig.2.19 and Table 2.1, respectively.

On observing the output change from 1 to 0 one may conclude that both inputs are now at 0. However, when seeing the output change from 0 to 1 the observer is not able to make conclusions about both inputs. The observer only knows that at least one input is 1, but it does not know which. So the OR gate only indicates or acknowledges when both inputs are at 0. Through similar arguments it can be seen that an AND gate only indicates when both inputs are 1. Signal transitions that are not indicated or acknowledged in other signal transitions are the

source of hazards and should be avoided. A circuit that is better in this respect is the Muller C-element [53].



x	y	z
0	0	0
0	1	no change
1	0	no change
1	1	1

Fig. 2.20 Muller C-element symbol and implementation with specification

The Muller C-element is a state-holding element much like an asynchronous set-reset latch [4] shown in Fig. 2.20. When both inputs are 0 the output is set to 0, and when both inputs are 1 the output is set to 1. For other input combinations the output does not change. Consequently on observing one can see the output change from 0 to 1 may conclude that both inputs are now at 1; and similarly, an observer seeing the output change from 1 to 0 may conclude that both inputs are now 0. Combining this with the observation that all asynchronous circuits rely on handshaking [30] that involves cyclic transitions between 0 and 1, it is understood that the *Muller C-element is indeed a fundamental component* that is extensively used in asynchronous circuits.

2.5 Design of Muller Pipeline

The Muller pipeline or Muller distributor is shown in Fig. 2.21. It is a circuit that is built from C-elements and inverters. Simple variations and extensions of this circuit form the (control) backbone of

almost every asynchronous circuits. It may not always be obvious at a first glance, but if one strips off the cluttering details, the Muller pipeline [34] is always there as the crux. This muller pipeline circuit has a peculiar beautiful and symmetric behaviour. The Muller pipeline in Fig 2.21 is a mechanism that relays handshakes. After all of the C-elements have been initialized to 0 the left environment may start handshaking. Lets consider the 'i'th C-element, C[i]: It will propagate (i.e input and store) a 1 from its predecessor, C[i-1], only if its successor, C[i+1], is 0. In the same way, it will propagate a 0 from its predecessor if its successor is at 1.

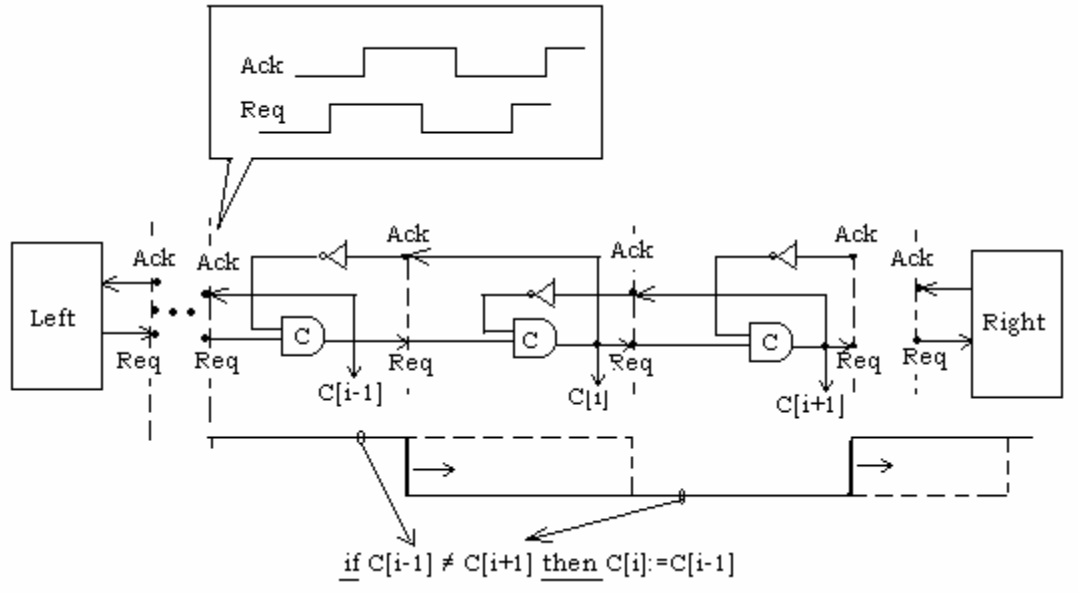


Fig 2.21 Illustration of Muller pipeline

It is often useful to think of the signals propagating in an asynchronous circuit as a sequence of waves, as illustrated at the bottom of Fig. 2.21. Viewed this way, the role of a C-element stage in the pipeline is to propagate crests and troughs of waves in a carefully controlled way that maintains the integrity of each wave [4].

In all interface between C-element pipeline stages one can see correct handshaking, but the timing may differ from the timing of the handshaking on the left hand environment; once a wave has been injected into the Muller pipeline it will propagate with a speed that is determined by actual delays in the circuit. Eventually the first handshake (request) injected by the left hand environment does not respond to the handshake, the pipeline will eventually fill. If this happens the pipeline will stop handshaking with the left environment-the Muller pipeline behaves like a ripple through FIFO[55] [58].

The Muller pipeline has a set of beautiful symmetries. First, it's the same circuit and it does not matter whether one uses a 2-phase or 4-phase handshaking. The difference is in how one interprets the signals and uses the circuit. Second, the circuit operates equally well from right to left. One may reverse the definition of signal polarities, reverse the role of request and acknowledge signals, and operate the circuit from right to left. It is analogous to electrons and holes in a semiconductor; when current flows in one direction it may be carried by electrons flowing in one direction or by holes flowing in the opposite direction. Also, the circuit has the interesting property, the Muller-pipeline is delay-insensitive and it works correctly regardless of delays in gates and wires [52].

2.6 4-Phase Dual-rail Pipeline

The Muller pipeline is the basic for 4-phase dual rail pipeline, but elaborately it has to do with the combined encoding of data and request. The implementation of a 1-bit wide and three stage deep

pipeline without data processing is shown in fig. Fig.2.22. As per circuit, it can be understood as two Muller pipelines connected in parallel, using a common acknowledge signal (Ack) per stage to synchronize operation. The pair of C-elements in a pipeline stage can

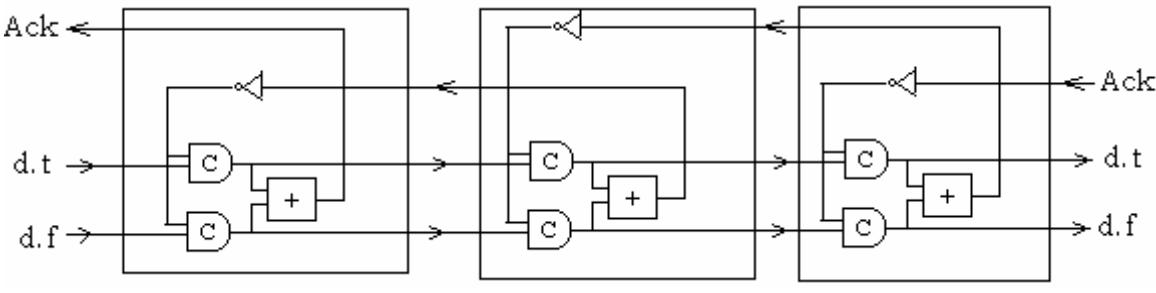


Fig 2.22 Simple 1-bit wide 3-stage 4-phase dual-rail pipeline

store the empty codeword {d.t,d.f}={0,0}, causing the acknowledge signal out of that stage to be 0, or it can store one of the two valid codewords {0,1} and {1,0}, causing the acknowledge signal out of that stage to be logic 1. Since the codeword {1,1} is illegal and does not occur, the acknowledge signal generated by the OR gate safely indicates the state of the pipeline stage as being “valid” or “empty” [4] [6]. So it is understood that an N-bit wide pipeline can be implemented by using a number of 1-bit pipelines in parallel. But this does not guarantee to a receiver that all bits in a word arrive at the same time and often the necessary synchronization is done in the function blocks. The individual acknowledge signals can be combined into one global acknowledge using a C-element for the need of bit-parallel synchronization.

Chapter 3

Architecture and Implementation of ALU

We have discussed so far about the importance efficient low power design by asynchronous technology for portable battery devices and motivation of our work in chapter one and the second chapter dealt about implementing styles of low power and asynchronous design methodologies. In this section, we are discussing about the fundamental features, building blocks and the completion detection of asynchronous circuits[16] architecture and implementing the architectures through the basic C-Muller element.

3.1 Speed-independence basics

On reviewing the basics of Muller's model of a circuit and the conditions for it being independent, a circuit is modeled along with its environment as a closed network of gates, closed meaning that all inputs are connected to outputs and vice versa. In this circuit design all the gates are modeled as Boolean operators with arbitrary non-zero delays with wires as ideal [31]. Here in this environment the circuit can be described as a set of concurrent Boolean functions, for every gate output. Hence the state of the circuit is the set of all gate outputs.

In Figure 3.1, it is illustrated Muller pipeline with an inverter and a buffer performing the handshaking behaviour of the left and right environments. A gate whose output is consistent with its inputs is defined to be stable; also its "next output" is the same as its "current output", $z_i' = z_i$. A gate whose inputs have changed in such a way that

an output change is called for is said to be excited; its 'next output" is different from its "current output", i.e $z_i' \neq z_i$. After an arbitrary delay an excited gate may spontaneously change its output and become stable with new output values, other gates in turn become excited etc.

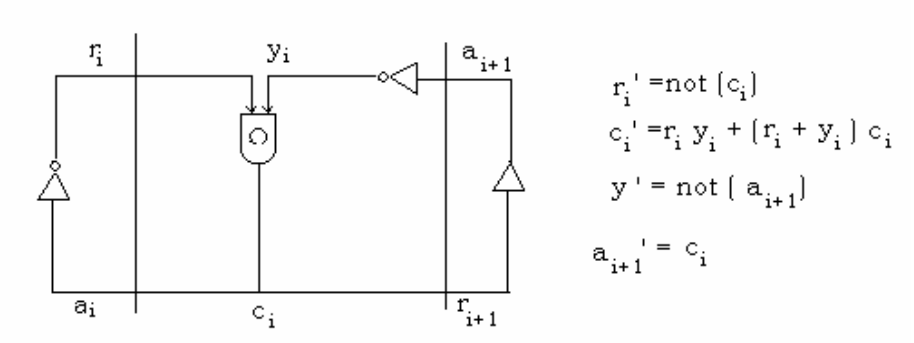


Fig 3.1 Muller Model of a Muller pipeline stage with dummy gates modelling

On illustrating this, with the circuit in Fig.3.1 is in state $(r_i,y_i,c_i,a_{i+1}) = (0,1,0,0)$. The inverter r_i gets excited corresponding to the left environment being about to take request high in this state. After the firing of $r_i \uparrow$ the circuit reaches state $(r_i,y_i,c_i,a_{i+1}) = (1,1,0,0)$ and c_i now becomes excited. In order to synthesis and analysis one can construct the complete state graph representing all possible sequences of gate firings. Generally it is possible that several gates are excited at the same time in a given state. If one of these gates, say z_i, fires the interesting thing is what happens to the other excited gates which may have z_i as one of their inputs: they may remain excited, or they may find themselves with a different set of input signals that no longer calls for an output change. Generally a circuit is said to be speed-independent if the latter never happens. The practical implication of an excited gate becoming stable without firing is a potential hazard. Since delays are unknown the gate may or may not have changed its output, or it may be in the middle of doing so when the 'counter order' comes

calling for the gate output to remain unchanged. Because of the model involves a Boolean state variable for every gate and for each wire segments in the case of delay-insensitive circuits even for very simple circuits the state space becomes very large. Now, we have a model for describing and reasoning about the behaviour of gate-level circuits in the following section.

3.2 Asynchronous Circuits- A brief classification with delay

Asynchronous circuits are classified as self-timed, speed-independent or delay-insensitive depending on the delay assumptions that made in the gate level. The Fig. 3.2 shows three gates: A, B, and C. The output signal from gate A is given as input signal on gates B and C. A speed-independent circuit as introduced above is a circuit that operates "correctly" assuming positive, bounded but unknown delays in gates and ideal zero-delay wires. On referring to this Fig.3.2 , this means arbitrary d_A , d_B , and d_C, but $d_1 = d_2 = d_3 = 0$. On assuming ideal zero-delay wires is not realistic in semiconductor processes. So by keeping arbitrary d_1 and d_2 and by requiring $d_2 = d_3$ the wire delays

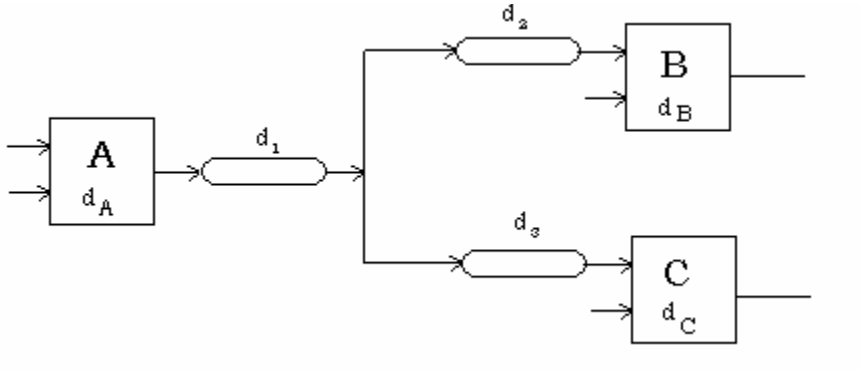


Fig. 3.2 A circuit part with gate and wired delays

with gates, but theoretically the circuit is speed-independent [52] one.

Asynchronous circuit which operates correctly with positive, bounded but with unknown delays in wires as well as in gates in delay-insensitive (DI), on referring to above Fig. 3.2 this means arbitrary d_A , d_B , d_C, d_1, d_2, and d_3. These kind of circuits are always very robust. One of the ways to show that any circuit is delay-insensitive is to use Muller model of the circuit where wire segments are modeled as buffer components. If this equivalent circuit model is speed-independent, then usually the circuit is delay-insensitive. But the class of delay-insensitive circuits is rather small unfortunately. As mentioned earlier, only circuits composed of C-element and inverters can be delay-insensitive and the Muller pipeline in chapter two and in Fig. 3.2 is one important example. The circuits that are delay-insensitive with the exception of some carefully identified wire forks where $d_2 = d_3$ are called often as quasi-delay-insensitive (QDI). Such wire *forks, where signal transitions occur at the same time at all end-points, are called isochronic* [55] [4].

These isochronic forks are used to found in gate-level implementations of basic building blocks for the designer can able to control wire delays. In high level of abstraction the composition of building blocks usually is delay-insensitive. Now it is clear that difference between DI, QDI and SI. Since the class of delay-insensitive [57] circuits is very small, basically excluding all circuits that compute, most circuits that are referred to in the literature as delay-insensitive are only quasi-delay-insensitive.

3.3 Isochronic forks

From the classification above one can understand that the specific difference between speed-independent circuits and delay-insensitive circuits relates to wire forks and, more specifically, to whether the delays to all end-points of a forking wire are identical or not. *If the delays are identical, the wire-fork is called isochronic*. The need for isochronic forks is related to the concept of indication principle defined in chapter 2 with Muller C-element and indication. On viewing the situation in Fig. 3.2 where gate A has changed its output. Eventually this change in gate A output is observed on the inputs of gates B and C, and after sometime gates B and C may respond to the new input by responding a new output. If this sequence happens one can say that the output change on gate A is indicated by output changes on gates B and C. On the other hand, if only gate B responds to the new input and it is not possible to establish whether gate C has seen the input change as well. Then in this particular case it is necessary to strengthen the assumptions to $d_2 = d_3$ (the fork is isochronic) and conclude that because the indication of changing output gate B and C for the input signal change at A.

3.4 Bundled-data circuit relations with speed-independence

Normally the 2-phase and 4-phase bundled-data approaches the control circuits are speed-independent or in some cases even delay-insensitive, but the data-path circuits with their matched delays are self-timed [11]. Circuits designed following the 4-phase dual-rail approach are generally quasi-delay-insensitive. In the 4-phase dual-rail pipeline circuits shown in chapter 2, the forks that connect to the

inputs of several C-elements must be isochronic, whereas the forks that connect to the inputs of several OR gates are delay-insensitive.

The different circuit classes, DI, QDI, SI and self-timed, are not mutually exclusive ways to build complete systems, but useful abstractions that can be used at different levels of design. In most practical designs they are mixed. The choice of handshake protocol and circuit implementation style is among the factors to consider when optimizing an asynchronous digital system [46]. It is important to stress that speed-independence and delay-insensitivity are mathematical properties that can be verified for a given implementation. If an abstract component – such as a C-element or a complex And-Or-Invert gate – is replaced by its implementation using simple gates and possibly some wire-forks, then the circuit may no longer be speed-independent or delay-insensitive [7]. As illustrated in Muller pipelines in chapter 2, it is no longer delay-insensitive if the C-element is replaced by the gate level implementation which uses simple AND and OR gates. Furthermore, even simple gates are abstractions; in CMOS the primitives are N and P transistors, and even the simplest gates include forks. As SI circuits ignore wire delays completely some care is needed when physically implementing these circuits [40].

In general one might think that the zero wire-delay assumption is trivially satisfied in small circuits involving 10-20 gates, but this need not be the case: a normal place and route CAD tool might spread the gates of a small controller all over the chip. Even if the gates are placed next to each other they have different logic thresholds on their inputs, which in combination with slowly rising or falling signals can

cause circuits to malfunction. For static CMOS and for circuits operating with low supply voltages this is less of a problem, but for dynamic circuits using a larger V_{DD} the logic thresholds can be very different [35].

3.5 Building Blocks of Asynchronous Circuits

The below is a set of minimum components that is sufficient to implement asynchronous circuits is shown in Fig. 3.3. These components can be grouped in four categories as explained below.

3.5.1 Latches

Latches are one of the primary component in building asynchronous circuits which provides storage for variables and implements the handshaking that supports the token flow. In addition to the normal latch a number of degenerate latches (sink) are often needed; a latch with only an output channel is a source that produce tokens with the same constant value, and a latch with only an input channel is a sink that consumes tokens [50]. Fig.2.22 in chapter 2, explain the implementation of a 4-phase dual-rail latch.

3.5.2 Function blocks

The function blocks are basically the asynchronous equivalent of combinatorial circuits and which is another main component for building asynchronous circuits. They are transparent/passive from a handshaking point of view. A function block will wait for tokens on its inputs (an implicit join), then perform the required combinatorial

function, and finally issue tokens on its outputs. Both empty and valid tokens are handled in this way [33]. It may be necessary to use an explicit join component. Where some implementations assume that the inputs have been synchronized.

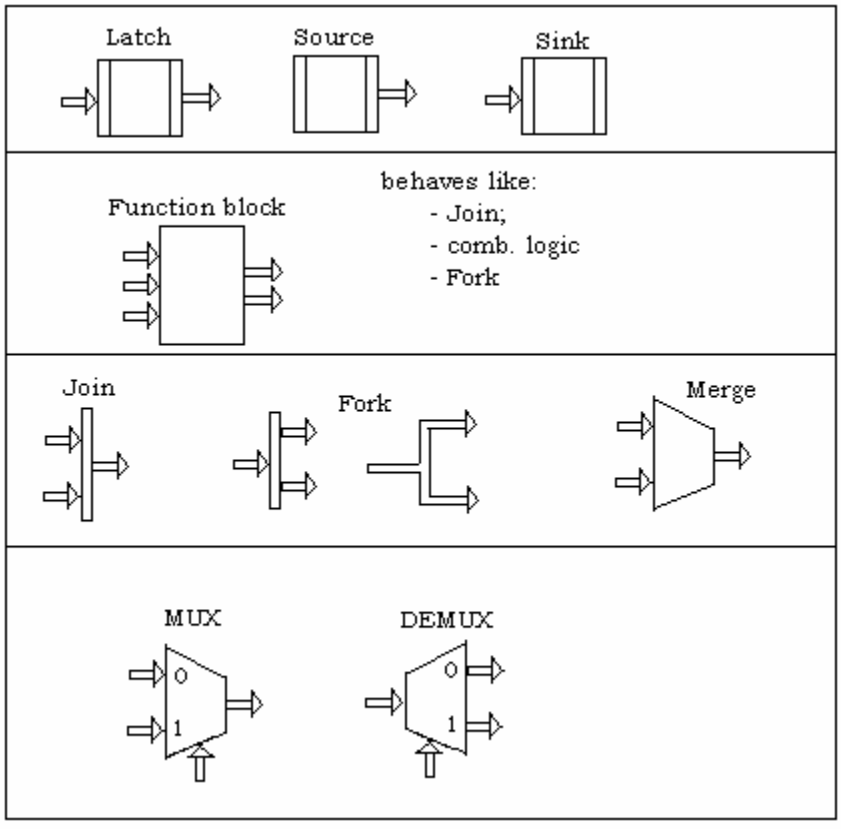


Fig. 3.3 Building blocks for asynchronous circuits

3.5.3 Unconditional flow control

The flow control will be done with Fork and Join components. Fork and join are used to handle parallel threads of computation. In technical terms, forks are used in situation where the output from one component is input to more components, and join are used where data from several independent channels need to be synchronized-typically because they are independent inputs to a circuit. But often it is omitted

joins and forks from a circuit diagrams: in short , the fan out of a channel implies the definition of a fork, and the fan-in of several channel implies the definition of a join. A merge component generally has two or more input channels and only one output channel. On the input channels handshakes are assumed to be mutually exclusive and the merge relays input tokens/handshakes to the output.

3.5.4 Conditional flow control

MUX and DEMUX are important components which perform the usual functions of selecting among several inputs or steering the input to one of several outputs. The control input is a channel just like the data inputs and outputs available. A MUX will synchronize the control channel and the relevant input channel and send the input data to the data output. The other input channel is ignored. Similarly a *DEMUX will synchronize the control and data input channels and steer the input to be selected output channel* [45].

3.6 4-phase dual-rail implementation of basic components

The 4-phase dual-rail implementations of the basic fork, join and merge components are important to build complex circuits are shown in Fig. 3.4. For easy understanding in this figure it shows a simple fork with two output channels and join and merge components with two input channels for simplicity. Also it is assumed that all channels are to be 1-bit channels. But of course, it is possible to generalize to three or more inputs and outputs respectively according to circuit requirements, and it is possible for extending to n-bit channels based on requirements [33].

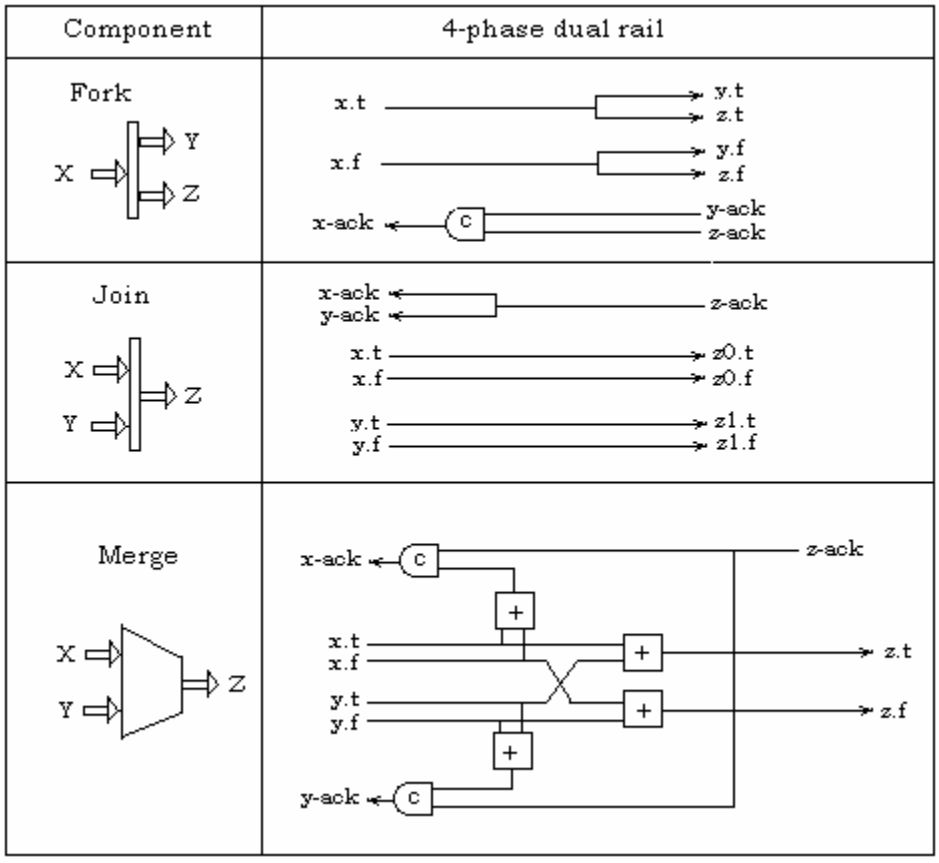


Fig. 3.4 The 4-phase dual rail implementation of fundamental components

The basic component fork includes a C-element for combining the acknowledge signals on the output channels into a single acknowledge signal on the input channel. The basic component join does not involve any active components as the request signal is encoded into the data in 4-phase dual-rail design. The particular fork in figure 4 duplicates the input data, and the join concatenates the input data. From a control point of view the different alternatives are identical: a join synchronizes several input channels and a fork

synchronizes several output channels. In the 4-phase dual rail merge implementation includes C-element and here the request is encoded into the data signals and an OR gate is used for each of the two-output signals z.t and z.f. Acknowledge on an input channel 4-phase dual-rail is produced in response to an acknowledge on the output channel provided where the input channel has valid data [44] [41].

3.7 Completion detection implementation with Muller C-elements

The Fig. 3.5 shows an N-bit wide latch. The OR gates and the C-element in the dashed box form a completion detector that indicates whether the N-bit dual-rail codeword stored in the latch is empty or valid. The figure also shows an implementation of a completion detector using only a 2-input C-element.

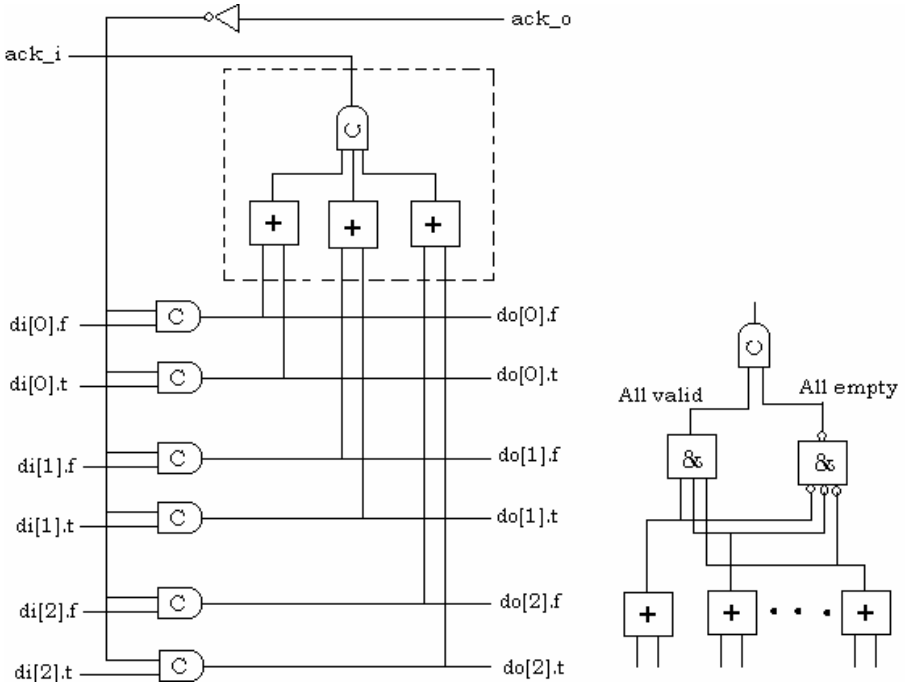


Fig. 3. 5 Implementation of N-bit latch with completion detection

In combinational circuits the 4-phase dual-rail pipeline is implemented in such a way that the circuits must be transparent to the handshaking between latches. Therefore, all outputs of a combinational circuit must not become valid until after all inputs have become valid. Otherwise the receiving latch may prematurely set acknowledge low before all signals from the sending latch have become empty. Consequently a combinational circuit for the 4-phase dual-rail approach involves state holding elements and it exhibits a hysteresis-like behaviour in the empty-to-valid and valid-to-empty transitions [14] [45].

3.8 Simple AND gate implementation with Muller C-element

The AND gate is implemented in dual-rail logic with the Muller C-elements in Fig. 3.6. The circuit may be understood as a direct mapping from sum-of-minterms expression for each of the two output wires into hardware and the truth table for this implementation is given in Table 3.1. The circuit waits for all inputs to become valid. When this happens exactly one of the four C-elements goes high. This again causes the relevant output wire to go high corresponding to the gate producing the desired valid output. When all inputs become empty the C-elements are all set low, and the output of the dual-rail AND gate becomes empty again. Note that the C-elements provide both the necessary ‘and’ operator and the hysteresis in the empty-to-valid and valid-to-empty transitions that is required for transparent handshaking. Note also that the OR gate is never exposed to more than one input signal being high [32].

a	b	y.f y.t
E	E	0 0
		No Change
F	F	1 0
F	T	1 0
T	F	1 0
T	T	0 1

Table 3.1 Truth table for AND gate implementation

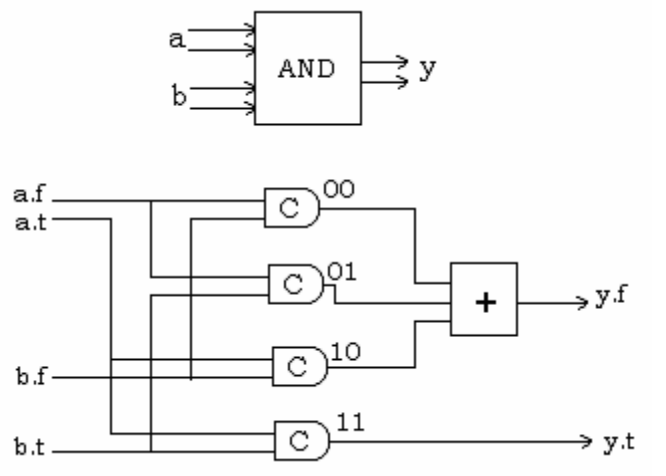


Fig. 3.6 Dual-rail implementation of AND gate with Muller C-element

Other dual-rail gates such as OR and EXOR can be implemented in a similar fashion, and a dual-rail inverter involves just a swap of the true and false wires. Given a set of basic dual-rail combinational circuits [28] [29] for arbitrary Boolean expressions using normal combinational circuit synthesis techniques. The transparency to handshaking is a property of the basic gates into larger combinational circuits. The transistor count in these basic dual-rail gates obviously high and it can be implemented efficiently with domino logic for reduced transistor count [53] [46].

3.9 Hybrid Adder Function

The basic structure of hybrid adder is shown in Fig. 3.7. Each full adder is composed of a carry circuit and a sum circuit. Here the concept is that the circuits precharged when signal Req_{in} =0, and in evaluation state when Req_{in} =1, able to detect when all carry signals are valid and use this information to indicate completion as 4-phase dual-rail protocol, i.e. Req_{out} ↑. There will be latency and if the latency of the completion detection does not exceed the latency in the sum circuit in a full adder then a matched delay element is needed to design as indicated in Fig. 3.7. Generally the latency of the completion detector may significantly exceed the latency of the sum circuit. With the use of basic Muller C-element this can be implemented and it is shown in Fig. 3.8 explained below section. The block diagram presented here shows the N-bit adder 4-phase bundled-data input output channels with internal dual-rail carry chain [17].

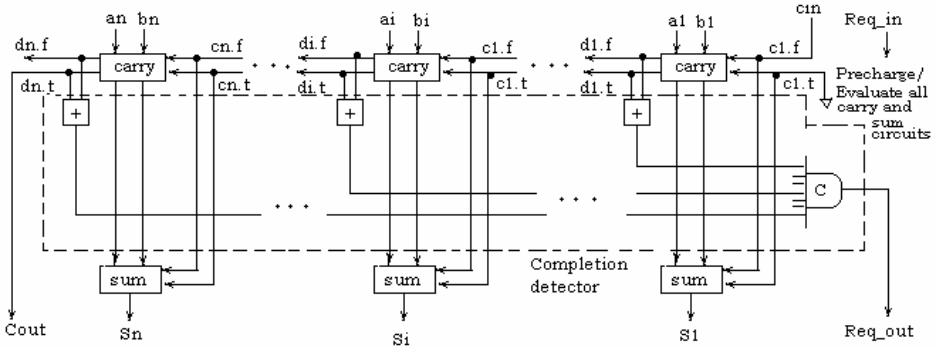


Fig. 3.7 N-bit Adder Block diagram dual-rail implementation

3.10 4-phase dual-rail adder implementation with Muller C-element

The implementation of an AND gate is explained in section 3.8. Using the same basic design it is possible to implement every other gates such as OR, EXOR, etc. An inverter involves no active circuitry, as it is just a swap of the two wires. By combining gates any arbitrary functions can be implemented exactly the same way as when one designs combinatorial circuits for a synchronous circuit. But essentially the handshaking is implicitly taken care of and can be ignored when composing gates and implementing Boolean functions. This has the important implication that existing logic synthesis techniques and tools may be used, but the basic gates are implemented differently that is the only difference here.

The 4-phase dual-rail AND gate implementation in section 3.8 is found rather inefficient: In this design there are 4 C-elements and 1 OR gate approximately has 30 transistors which is five greater than a normal AND gate whose implementation requires only 6 transistors. But if larger functions are implemented then the overhead can be reduced. This design is illustrated in fig 3.8. The circuit in fig 3.8 (b) is look like PLA design and it illustrates a principle for implementing arbitrary Boolean functions in general. This is implemented in DIMS-Delay-Insensitive Minterm Synthesis- since *the circuits are delay-insensitive and because the C-elements in the circuits generate all minterms of the input variables* [4]. The truth table have 3 set of rows

specifying the output when the input is different and they are as follows: set (1): the empty or null codeword to which the circuit responds by setting the output empty/ null, set (2): no change for an intermediate codeword which does not affect the data output, set (3): a valid codeword to which the circuit responds by setting the output to the proper value for each input.

a	b	c	s.t	s.f	d.t	d.f
E	E	E	0	0	0	0
			No change			
F	F	F	0	1	0	1
F	F	T	1	0	0	1
F	T	F	1	0	0	1
F	T	T	0	1	1	0
T	F	F	1	0	0	1
T	F	T	0	1	1	0
T	T	F	0	1	1	0
T	T	T	1	0	1	0

Table 3.2 Truth table for 4-phase dual-rail adder

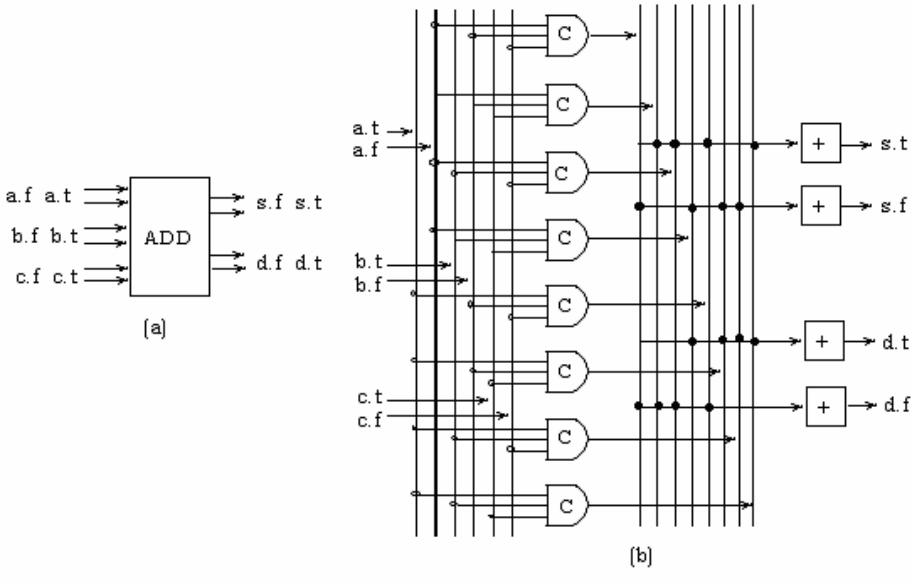


Fig. 3.8 4-phase dual-rail adder (a) symbol and (b) implementation

For designing the asynchronous ALU, the basic circuit is to design an adder. The above Fig. 3.8 shows the implementation of asynchronous 1-bit full adder circuit with C-elements. Fig.3.8 (a) shows the block level input and output signals. Since it is a 4-phase dual rail protocol, the information a, b and c uses 2 wires; a.t, b.t and c.t for signaling a logic 1 and other wires a.f, b.f and c.f for signaling logic 0. The Table 3.2 shows the truth table for this 4-phase dual rail full adder and the same implemented with C-elements and OR gates, which is shown in fig. 3.8(b).

3.11 4-phase Dual-rail Dynamic CMOS Asynchronous ALU Implementation

Using the logics and principles outlined in chapter 2 and above basic components implementation, we design an ALU at the transistor level for single bit operation as shown in the Fig. 3.9 to demonstrate our design concept. A single bit-slice ALU uses only 53 transistors and its range of operations in Table 3.3. Since we emphasize on the design of asynchronous component, there is no hardware implementation for 4-phase dual rail with Muller C. However, the proposed circuit assumes the signaling from such logic blocks. For example, C0out and C1out act as two wires of 4-phase logic, which makes reliable operation between its predecessor and successor blocks.

We extend our 1-bit asynchronous ALU to design a 32-bit ALU, which requires 1696 transistors. The basic principle of Bundled data – Bounded delay model of Sutherland's micro pipelines is used here [12]. The timing characteristics of all data busses of this architecture are bundled together. The statuses of the data busses are indicated

(acknowledged) by 4 phase-dual rail handshake signals. The clock power reduction at the architectural level is mainly due to pipeline technique. The dynamic logic of completion detection unit ensures precise internal operation, because of its 4-phase dual logic. It is also carrying the timing information because it uses common timing characteristics.

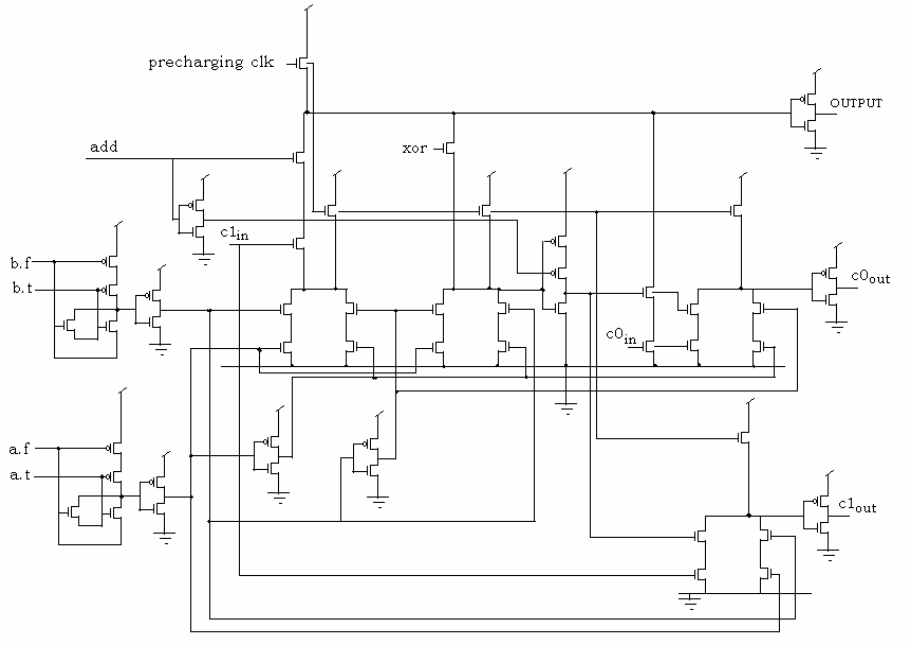


Fig. 3.9. Dynamic CMOS 4-phase dual-rail asynchronous ALU 1-bit circuit

In this circuit diagram fig.3.9, the data signals are a.t, a.f, b.t, b.f, C0in and C1in. The precharging clk signal is used to precharge the required nodes as per dynamic logic. Depending upon the required logic

function to be done by the ALU, the ADD and X-OR signals will be given for corresponding logic operation as stated in the Table 3.3.

Logic Funcion	Basic Operation	a-input	b-input
and	AND	true	True
add	AND	true	true
add with carry	AND	true	true
subtract	AND	true	complement
reverse subtract	AND	complement	true
subtract with carry	AND	true	complement
rev. subtract with carry	AND	complement	true
test bits	AND	true	true
compare	AND	true	complement
compare negative	AND	true	true
bit clear	AND	true	complement
xor	XOR	true	true
test equal	XOR	true	true
or	OR	true	true
move	OR	zero	true
move NOT	OR	zero	complement

Table 3.3 Functions available with the ALU

Table 3.3 explains the operation available with the designed ALU where the data signals a and b requires 2 wires for each data bit. The data signals a.t, a.f and b.t, b.f are required as per the 4-phase dual-rail logic as explained in chapter 2. Here a.t and b.t used for representing logic 1 (true), similarly a.f and b.f represents logic 0 levels (complement) as shown in Table 3.3. For the arithmetic operation ‘add’ and for the logical operations ‘xor’ control signals will be given logic 1 for the listed above listed operations. The

implemented ALU design can carry out all the 16 operations listed in Table 3.3 and the simulated results, performance analysis are discussed in the next chapter.

Chapter 4

Performance Analysis and Simulation Results

The implementation and the operations of designed asynchronous 4-phase dual-rail domino logic ALU are discussed in the previous chapter. In this chapter, the simulation results verify the correct operation of the functions defined and the performance are analyzed with latest related published works to claim the improved throughput of our proposed circuit ALU design and the methodology proposed. Also the SPICE tool, which used for the simulation purpose and the technology library used are discussed briefly. The simulation results are plotted for 1-bit operation and performance analysis

4.1 SPICE Simulation tool

The cadence HSPICE and Tanner T-SPICE tools were used for the entire analysis and the results were tabulated and plotted in this chapter. The simulation results for the power consumption of typical addition operation with different supply voltages are analyzed. Since the circuit designed at CMOS transistor level and asynchronous, the SPICE tools are used worldwide to prove the efficiency of the circuit before lay out. i.e the variation between the simulation results by SPICE tools and post lay out level of the circuit will be comparable. Also the SPICE tools are the basic simulation tool and if any circuit performs well with the SPICE simulation performance can be moved to next lay out level with the same performance. The 0.18μm technology libraries were used for the simulations by the SPICE tools.

4.2 Analysis of Logic Operations

For a single bit addition operation three inputs: the two operand bits and a carry input from the previous stage. The addition operation is limited by the speed of the propagation of the carry signal across the word. However the carry output from a single bit addition does not always depend on the carry input and in half of the possible input cases it may be generated before the input carry state is known. It is therefore unlikely that a carry signal will have to propagate across many bit positions before it reaches one where its state has already been correctly predicted. In synchronous ALUs all operations must take the same amount of time; a considerable effort has been expended in schemes such as carry look ahead and carry steering in order to speed up the addition operation; these approaches require a large quantity of circuitry to accommodate a few worst cases of operation.

In an asynchronous ALU addition, operation may take different times depending on the input data, providing that some means of detecting completion is included. In our design the completion detection will be done x-or gate circuit as explained in previous chapter. If the cases with long carry propagation chains are relatively rare a simple adder may be used which-despite poor worst-case performance- can deliver typical results in in a reasonable time. This allows a reduction in size and complexity of the ALU, with a consequent reduction in power consumption, without radically altering the typical performance. The performance of our design is illustrated with different analysis in this section.

The simulated output waveform for the addition operation performed by this ALU presented in Fig. 4.1. It is performed with V_{DD}=1.8V, input sequence C1in=1111, C0in=0000, A=0011,B=0101 and the simulated output sequence is output=1001, C0out=1000, C1out=0111 which coincides the expected specification. This simulation is done using HSPICE tool with 10ns local clock period at room temperature 30˚C. The input and output specification for this addition operation is given in the Table 4.1.

Signals		Data			
Input	a.f	1	1	0	0
	b.f	1	0	1	0
	C1in	1	1	1	1
	add	1	1	1	1
	xor	0	0	0	0
	a.t	0	0	1	1
	b.t	0	1	0	1
Output	Cout (sum)	1	0	0	1
	C0out	1	0	0	0
	C1out	0	1	1	1

Table 4.1 Input/output logic specification for addition operation by ALU.

In Table 4.1, the input data sequence a.t, a.f, b.t, b.f and C1in are presented in logic levels and since it is addition operation the control bit ‘add’ is given high logic 1 and ‘xor’ being kept at low logic 0.As the operation is full addition, the carry bit signal C1in is given high logic 1. In the output signal ‘sum’ of the addition operation is given by ‘Cout’ and the carry bit will be represented by ‘C1out’ signal as shown in the simulated waveform in Fig. 4.1.

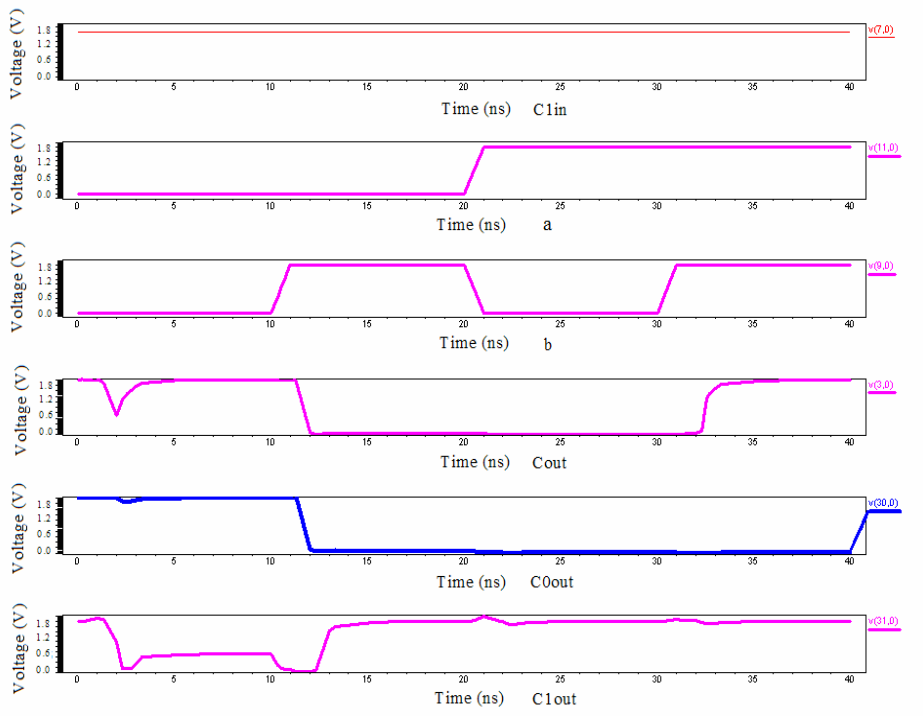


Fig. 4.1. Simulated waveforms for the ALU addition operation

The above Fig. 4.1 shows the SPICE simulated waveform for the one bit ALU addition operation, which follows the specification given in Table 4.1. The X-OR logic operation input output

signals		Data			
Input	a.f	1	1	0	0
	b.f	1	0	1	0
	C1in	0	0	0	0
	add	0	0	0	0
	xor	1	1	1	1
	a.t	0	0	1	1
	b.t	0	1	0	1
Output	OUT(xor)	0	1	1	0
	C0out	1	0	0	0
	C1out	0	0	0	1

Table 4.2 Input /output logic specification for x-or operation by ALU

specification is for single bit operation is given in Table 4.2 and the simulated SPICE waveform for this logic operation is given in Fig. 4.2.

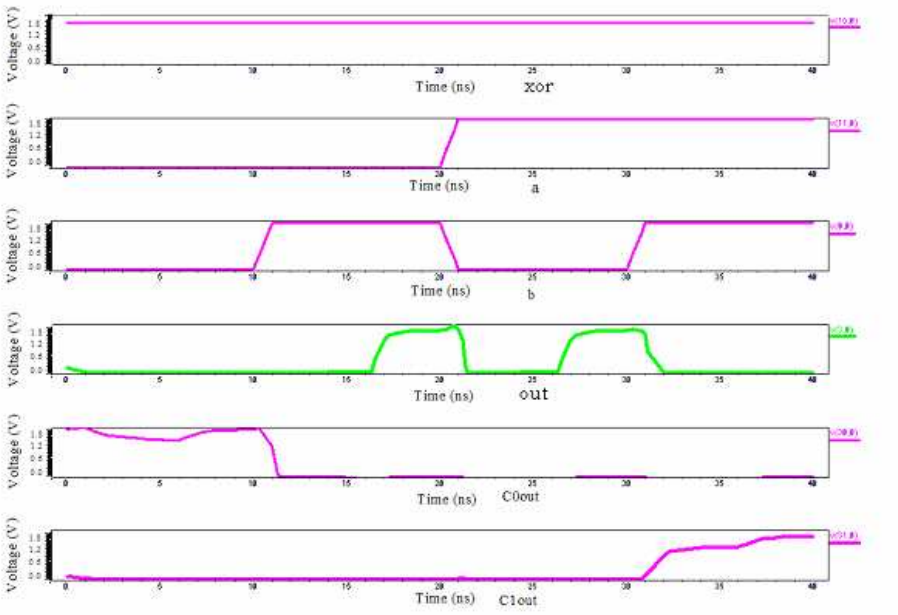


Fig. 4.2. Simulated waveforms for the X-OR operation

In Table 4.2, the input data sequence a.t, a.f, b.t, and b.f are presented in logic levels and operation to be done is XOR the control bit ‘add’ is given low logic 0 and ‘xor’ being kept at high logic 1. In the output signal ‘xor’ of the addition operation is given by ‘OUT’ signal as shown in the simulated waveform in Fig. 4.2.

4.3 Basic Addition Operation and Analysis

Addition is one of the fundamental functions of an ALU. We start by analyzing the number of transistors used in the addition. About 80%

of the operations require some form of addition [13]. If we improve the processing time of addition operation, the performance of complete ALU can also be improved. The latency required by our design is depended upon the operation, the input data at that incident and the carry flow across the whole word length, i.e. it needs to propagate carry until it has predicted by the completion detection stage. The average length of the mean carry propagation distance is varying according to input data. In this 32-bit operation, a sum of 140 transistors has used for precharging (domino logic) and buffer purposes to meet the specifications at the layout. The 4-phase dual rail logic with CMOS dynamic implementation so far discussed ensures very simple circuitry than existing designs and provides three benefits compared to size, power consumption and performance as discussed in previous chapters.

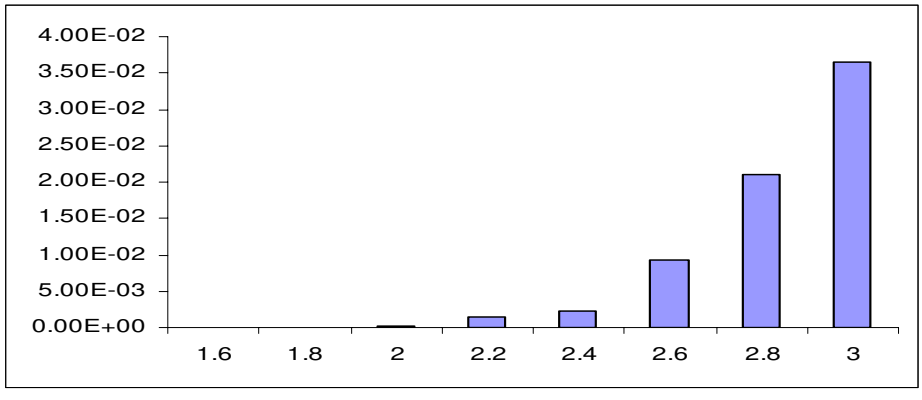


Fig. 4.3. Simulation Results for power consumption at different V_{DD}.

The Fig. 4.3 shows the bar diagram representation of power consumption by the same circuit with different V_{DD} ranging 1.6V to 3.0V and keeping the temperature as constant room temperature. The performance of the ALU is good and working correctly to a wide range of V_{DD} 1.6 V to 5V. The power consumption against different V_{DD} is

listed in the Table 4.3 and it plotted as a graph for easy analysis in Fig. 4.4.

Sl. No	V_{DD} (V)	Power consumption (watts)
1	1.6	4.92E-05
2	1.8	1.02E-04
3	2	1.93E-04
4	2.2	1.48E-03
5	2.4	2.24E-03
6	2.6	9.27E-03
7	2.8	2.11E-02
8	3	3.65E-02

Table 4.3 Power consumption for addition operation at different V_{DD}

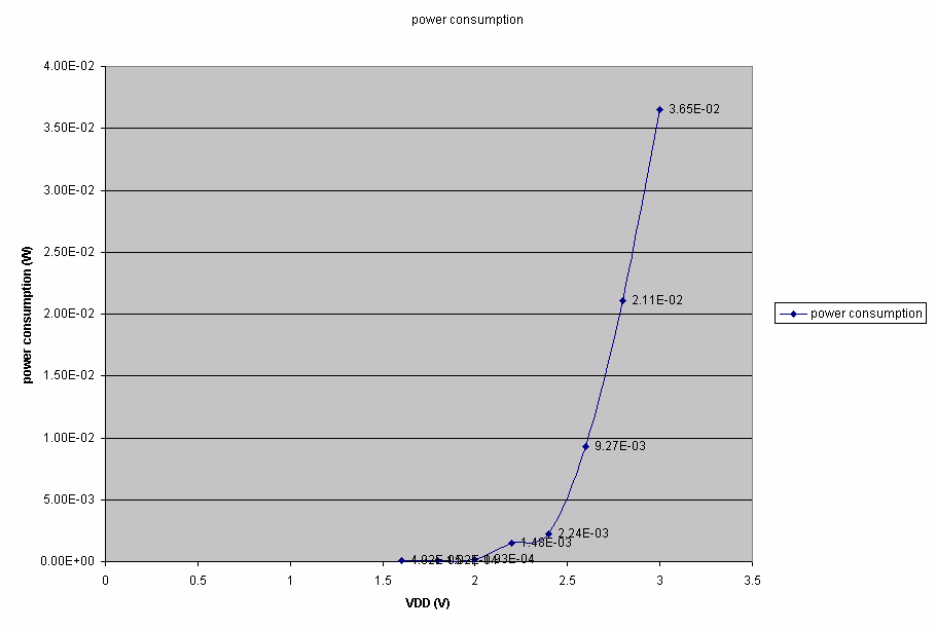


Fig. 4.4. V_{DD} Vs Power consumption of ALU for Adder operation

The simulation results of HSPICE-0.18μm technology shows, the average power consumption for typical addition operation is 1.02e-4w under 1.8V supply with 1000 sample inputs at room temperature and average time delay is 3.5ns. The Fig. 4.5 shows the graph representation of power consumption by the same circuit with different temperatures and keeping the V_{DD}=1.8V. The power consumption in different temperature is listed in the Table 4.4.

Sl. No	Temperature (˚C)	Power consumption (watts)
1	-15	9.44E-05
2	0	9.68E-05
3	15	9.91E-05
4	30	1.02E-04
5	45	1.04E-04

Table 4.4 Power consumption for adder operation at different Temperatures

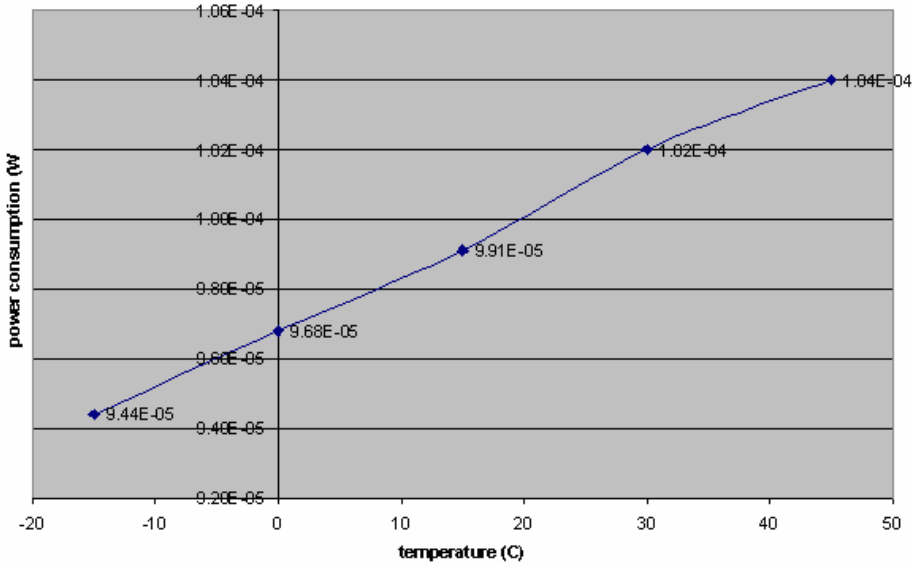


Fig. 4.5. Temperature Vs Power consumption of ALU for Adder operation.

The Fig. 4.6 shows the graphical representation of delay performance by the ALU circuit with different V_{DD} ranging 1.2V to 2.6V and keeping the temperature as constant room temperature.

Sl. No	V_{DD} (V)	Delay (ps)
1	1.2	12250
2	1.4	7200
3	1.6	5250
4	1.8	3500
5	2	2000
6	2.2	1600
7	2.4	1280
8	2.6	1150
9	2.8	1000

Table 4.5 Delay for addition operation with different supply voltages V_{DD}.

The delay get increased on reduction with V_{DD} and it is found that the correct performance assured on reducing the V_{DD} for a wide range of 5V to 1.6V with increasing delay and the data are listed in Table 4.5.

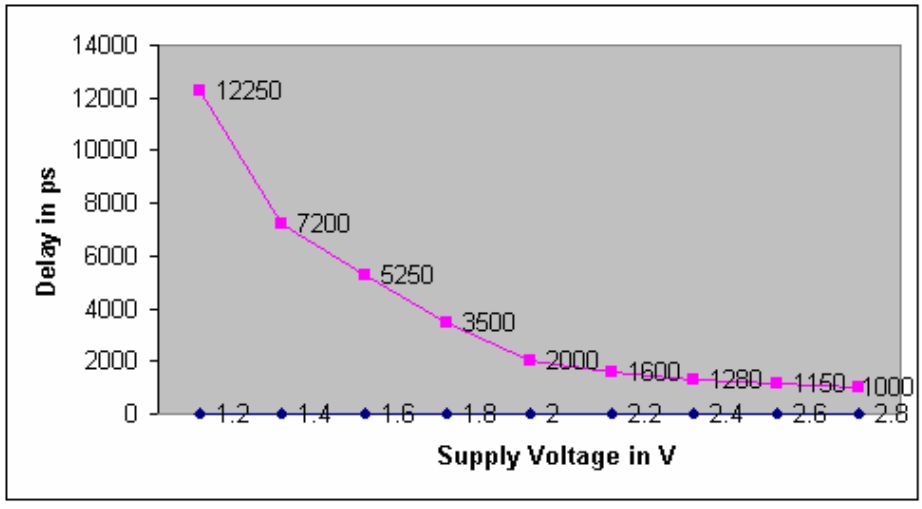


Fig.4.6 Supply Voltages (V_{DD}) Vs Delay performance for ALU addition operation

Simulation has been performed with our design to find the best and worst case performance, and the results are listed in table 4.6 A comparison of the simulated time performance of our design with other published works given in the below table which proves the better performance on worst case of 4ns and average case of 3.5ns.

Comparison with other related works	**Time(ns)**		
	Best case	***Worst case***	***Average***
Ref [14]	2.5	7.5	5
Ref [15]	3	6	4.5
Ref [16]	3	6	4.5
Our design	3	4	3.5
Synch	-	-	4

Table 4.6 Simulation Results for ALU operation

Comparison	Existing Designs and Discussed Design		
	Synchronous ARM ALU [14]	***Asynchronous ARM ALU [14]***	***Our Design***
Technology	1.2um CMOS	1.2um CMOS	0.18um CMOS
Supply voltage	~ 5V (CO/SS)	~ 5V (CO/SS)	~ 1.8V (CO/MS)
Self Time Unit	3000 (# transistors)	2300 (# transistors)	1696 (# transistors)
Timing Purpose	--	140 (# transistors)	140 (# transistors)
Data width	32 bit	32 bit	32 bit

CO-Correct Operation, SS-Slow Speed, MS-Medium Speed

Table 4.7 Performance Comparison with other published works

A comparison of the simulated time performance and transistor count of this design with other published alternatives is shown in table 4.7. Our design has much reduction in silicon area. In addition, this architecture enables to have reduced switching capacitance because of absence of master clock in the ALU circuit design. It gives reduced switching actions for every arithmetic operation. In summary, our proposed design gives a better throughput with minimum number of transistors.

Chapter 5

Conclusions

The Asynchronous chips are presently research prototypes and are not about to replace synchronous ARM cores in commercial production. However, there is a worldwide resurgence of interest in the potential of asynchronous design techniques to save power and to offer a more modular approach to the design of computing hardware.

Our investigation shows that the proposed asynchronous ALU, which reduce both average-case and worst case operation delays over that previous asynchronous designs, can be designed in such a way that with less power and less area. Since we used the 4-phase dual-rail pipelines in designing our 32-bit ALU circuit, the complexity of implementation is reduced when compared to the 2-phase dual-rail and bundled-data protocol designs. Also this ALU design has following unique asynchronous design advantages.

- Use little average power
- Show small current peaks, and
- Operate over a wide range of the power supply.

Measurements and simulations showed the following advantages of this design when compared to a conventional synchronous one:

- This asynchronous ALU design gives the maximum performance for the power received. This comes mainly from the fact that the asynchronous design needs less of what is the main limiting factor for the performance, namely power. Compared to a synchronous design, the asynchronous circuits

needs about 40% of the power for less. The proposed design having reduced transistor count in comparison with earlier designs on comparison and better power-delay characteristics.

- This asynchronous design is more resilient to voltage drops, since it still operates correctly for voltages down to 1.8 V.
- The current peaks of an asynchronous circuits are less pronounced , making the requirements with respect to the buffer capacitor more modest.

The power savings which result from removing the global clock, leaving each subsystem to perform its own timing functions whenever it has useful work to perform, are clear in theory but there are few demonstrations that the benefits can be realized in practice with circuits of sufficient complexity to be commercially interesting. Our ALU design work is aimed directly at adding to the body of convincing simulation demonstrations of the merits of asynchronous technology. It is also clear that, should asynchronous technology gain acceptance as a low power design style, the ALU work places the ARM architecture in the vanguard of the asynchronous assault on the stronghold of the global clock.

The objective of our work is to demonstrate that a self-timed delay insensitive processing system can deliver competitive performance in a very flexible way, simplifying power-efficient design and minimizing electromagnetic interference. Asynchronous designs are naturally miserly with power, since they are inactive until presented with work to do. The power benefits are expected to be particularly manifesting in systems with highly variable workloads, hence the emphasis on embedded applications. Additional reasons for looking at this

asynchronous ALU design include its lower emission of electromagnetic radiation due to less coherent internal activity, and its potential to deliver typical rather than worst-case performance since its timing adjusts to actual conditions whereas clocked circuit must be tolerance for worst-case conditions.

5.1 Future Directions

There is considerable resistance amongst industrial designers to the use of asynchronous design techniques, in part because of obsolete prospective that view asynchronous design as unreliable which has been now largely been overcome by new, more rigorous techniques and in part because of genuine difficulties with production testability which are only now beginning to be addressed by the asynchronous research community.

However, clocked design is becoming ever more difficult as manufacturing process technology shrink. Wiring delay increasingly dominate logic delays on high performance chips, causing the global clock wiring to compromise the local logic performance. Clock generators consume an increasing share of the silicon resource and the power budget, and increasing clock frequencies cause worsening radio emissions and power consumption.

The choice facing manufacturers of portable digital equipment will therefore be either to sacrifice performance or to abandon fully synchronous design. The pressure will be most apparent in physically small systems that include both radio receivers and high performance digital electronics, such as digital mobile telephones and pagers. PDAs, mobile email terminals and portable multimedia terminals will

also soon benefit from asynchronous design. In the longer term the modularity, composability and synthesis potential of asynchronous circuits will make them attractive to a wide range of applications, when dealing with a global clock signal on a 1000,000,000 transistor chip will become simply too difficult to manage.

In addition to purely asynchronous technology, hybrid solutions appear to have future potential. Perhaps islands of synchronicity communicating asynchronously, or conventional data-path functions timed using a maximum delay philosophy within an asynchronously controlled system, could be employed. Key, large volume applications like general-purpose processors will have to drive the technology and associated tools development. Only when well-defined element libraries and high-level synthesizable design representations are available will asynchronous techniques challenge the contemporary synchronous approach.

5.2 Publications/ Communications out of this work

1) G. Sundar and C.R. Mandal, "A Design Technique for the Implementation of Asynchronous ALUs " in Proc. of VEDAS-2005, VLSI Society of India sponsored Conference, Salem, India, June1-2,2005.
2) G. Sundar and C.R. Mandal, "Design Methodologies for the Implementation of Low Power Asynchronous Designs Circuits " in Proc. of INCRUIS-2006 International Conference, Sona College of Technology, Erode, India, Jan7-8,2006.

3) G.Sundar, C. R. Mandal, IIT Kharagpur, India and P. Manikandan, B. D. Liu, L. Y. Chiou , NCKU, Taiwan " Asynchronous Design Methodology for an Efficient Implementation of Low Power ALU" communicated to Proc. Of APCCAS 2006 (IEEE) International Conference, Singapore Dec. 4-7, 2006 (http://www.apccas.org/)

Bibliography

[1] Inki Hong et. al. Power Optimization of Variable Voltage Core-Based Systems. IEEE Proceedings, 35th DAC, June 98, pp. 176-181.

[2] L.Benini and G. De Micheli. Transformations and Synthesis of FSM's for low power gated clock implementation. IEEE Trans. on CAD, Vol. 15, No. 6, June 1996.

[3] J. M. Rabaey, M. Pedram. Low Power Design Methodologies. Ch.1, Kluwer Academic Publishers, 1996, ISBNO-7923-9630-8.

[4] Jens Sparso and Steve Furber. Principles of Asynchronous Circuit Design: A Systems perspective. Kluwer Academic Publishers,2001.

[5] Scott Hauck. Asynchronous Design Methodologies: An overview, Proceedings of the IEEE, Vol.83,No. 1. January 1995.

[6] S.B. Furber, P. Day, J. D. Garside, N.C. Paver and J. V. Woods. A Micropipelined ARM. IEEE International Conference on Computer Design (1995).

[7] N. Paver, P. Day, S.B. Furber, J. D. Garside, J. V. Woods. Register locking in an Asyncronous Microprocessor, ICCD '92, IEEE International Conference on Computer Design (1992).

[8] Chris Hyung-il Kim, Jae-Joon Kim, , Saibal Mukhopadhyay, and Kaushik Roy. A Forward Body-Biased Low-Leakage SRAM Cache:

Device, Circuit and Architecture Considerations. IEEE Trans on VLSI Systems, Vol. 13, NO. 3, Mar 2005.

[9] V. Tiwari et. al.. Reducing power in High Performance Microprocessors. 35th DAC, June 1998.

[10] A.Hemani, T.Meincke , S.Kumar, A.Postula, T.01sson, P.Nilsson, J.Oberg , P.Ellervee',D.Lundqvist. Lowering power consumption in clock by using Globally Asynchronous Locally Synchronous design style, IEEE Proceedings, 35th DAC, Page(s) 873-878, 21-25 June 1999.

[11] G. M. Jacobs and R. W. Broaderson. A fully asynchronous digital signal processor using self-timed circuits. IEEE J. Solid State Circuits, vol.25,pp. 1526-1537, June 1990.

[12] P. Manikandan, B.D. Liu, V.K.P. Tripathi and G. Sundar. Design and Implementation of VLSI SoC for Remote Sensing Applications, The 2nd International Meeting on Microsensors and Microsystems, pp 168-169, Jan 15-18, 2006.

[13] J.F. Wakerly. Digital Design: Principles and Practices,3/e Prentice-Hall, 2001.

[14] J. D. Garside, A CMOS VLSI Implementation of an Asynchronous ALU, Proceedings of the IFIP Working conference on Asynchronous Design Methodologies, Manchester, M139PL, U.K.(1993).

[15] A. De Gloria and M Olivieri. Statistical Carry Lookahead Adders. IEEE Trans. on Computers, V 45 N 3, and March 1996 page 340-347.

[16] David Kearney, Neil W. Bergmann. Bounded data Asynchronous Multipliers with data Dependent Computation Times. IEEE Proceedings, Page(s):186 – 197, 7-10 April 1997.

[17] DeMassa, T. A. and Z. Ciccone. Digital Integrated Circuits. John Wiley & Sons Inc.: New York. 1996.

[18] Kang, S. and Y. Leblebici. CMOS Digital Integrated Circuits: Analysis and Design. The McGraw-Hill Companies, Inc.: New York. 1996. pp 322-372.

[19] Kurdahi, F. J. "Dynamic CMOS Circuits," Introduction to VLSI Design, University of California, Irvine, ECE151, Fall 1995. http://www.eng.uci.edu/ece/ece151/lec4/dynamic.html

[20] Pihl, J. Design Automation of High-Speed Digital Signal Processing in VLSI Design with Applications in Speech Recognition Systems Based on Hidden Markov Models, Ph.D. thesis defended November 1, 1996.
http://www.fysel.unit.no/People/Pihl/art_html/art_html.html

[21] Schindler, V. "Dynamic vs. Static CMOS Logic," High Speed RSA Hardware Based on Low-Power Pipelined Logic, Ph.D. thesis defended December 1996. http://www.iaik.tu-graz.ac.at/Lehre/Dissertationen/vschindl/node53.html

[22] Terman, C. "Precharge/Evaluate Logic," Introduction of VLSI Systems, Massachusetts Institute of Technology, Class 6.371, Fall 1996. http://cerberus.lcs.mit.edu/6.371/lectures/L9/

[23] Weste, N. H.E. and K. Eshraghian. Principles of CMOS VLSI Design: A System Perspective, 2nd Edition. Addison-Wesley Publishing Company.

[24] Yuan, J. and C. Svensson. "High-Speed CMOS Circuit Technique." IEEE Journal of Solid-State Circuits. vol. 24. pp 62-70. February 1989.

[25] Yuan, J. and C. Svensson. "New Single-Clock CMOS Latches and Flipflops with Improved Speed and Power Savings." IEEE Journal of Solid-State Circuits. vol 32. pp 62-69. January 1997.

[26] Yin-Kuan Lin and Ting-Chi Wang. "Chapter 5: CMOS Circuit and Logic Design," Introduction to VLSI Design, Chung Yuan Christian University.
http://www.ice.cycu.edu.tw/~vlsi/Chapter5/Chapter5.htm

[27 J. Rabaey. Digital Integrated Circuits: A Design Perspective. Prentice-Hall, 2003..

[28] W. J. Bainbridge and S. B Furber. Asynchronous macrocell interconnect using MARBLE. in “Async ‘98 “, pages 122-132. IEEE Computer Society Press, April 1998.

[29] W. J. Bainbridge and S. B. Furber. MARBLE: An asynchronous onchip macrocell bus. Microprocessors and Microsystems, 24(4):213-222, April 2000.

[30] A. Bardsley. Implementing Balsa handshake circuits. PhD thesis, Department of Computer Science, University of Manchester, 2000.

[31] P. A. Beerel, C. J. Myers, and T. H.-Y. Meng. Automatic synthesis of gate-level speed-independent circuits. Technical Report CSL-TR-94-648, Stanford University, November 1994.

[32] G. Birtwistle and A. Davis, editors. Proceedings of the Banff VIII Workshop: Asynchronous Digital Circuit Design, Banff, Alberta, Canada, Aug.28- Sept.3, 1993.

[33] I. Bogdan, M. Munteau, P. A. Ivey, N. L. Seed, and N. Powell. Power reduction techniques for viterbi decoder implementation. European Low Power Initiative for Electronic System Design (ESDLPD) Third International Workshop, pages 28-48, 2000.

[34] J. A. Brzozowsky and C.-J.H. Seager. Asynchronous Circuits. Springer Verlag, Monographs in computer Science, 1994.

[35] S. M. Burns. Performance Analysis and Optimization of Asynchronous Circuits. PhD thesis, Computer Science department, California Institute of Technology, 1991. Caltech-CS-TR-91-01.

[36] S. M. Burns and A. J. Martin. Syntax-directed translation of concurrent programs into self-timed circuits. In J. Allen and F.

Leighton, editors, Advanced Research in VLSI, pages 35-50, MIT Press, 1988.

[37] D. M. Chapiro. Globally-Asynchronous Locally-Synchronous Systems. Phd thesis, Stanford University, October 1984.

[38] T. –A. Chu. Synthesis of self-Timed VLSI Circuits from Graph-Theoretic Specifications. PhD thesis, MIT laboratory for Computer Science, June 1987.

[39] T.-A. Chu and R. K. Roy (editors). Special issue on on asynchronous circuits and systems, IEEE Design & Test, 11(2), 1994.

[40] T.-A. Chu and L. A. Glasser. Synthesis of self-timed control Circuits from graphs: An example. In Proc. International Conf. Computer Design (ICCD), pages 565-571. IEEE Computer Society Press, 1986.

[41] A. Davis. A data-driven machine architecture suitable for VLSI implementation. In Proceedings of the First Caltech Conference on VLSI, pages 479-494, Pasadena, CA, January 1979.

[42] A. Davis and S. M. Nowick. Asynchronous circuit Design: Motivation, background, and methods. In G. Birtwistle and A. davis, editors, Asynchronous Digital Circuit Design, Workshops in Computing, pages 1-49. Springer-verlag, 1995.

[43] A. davis and S. M. Nowick. An introduction to asynchronous circuit design. Technical Report UUCS-97-013, department of Computer Science, University of Utah, September 1997.

[44] S. B. Furber and P. day. Four-phase micropipeline latch control circuits . IEEE Transactions on VLSI Systems, 4(2):247-253, June 1996.

[45] J. D. Garside, The Asynchronous Logic Homepages. http://www.cs.man.ac.uk/async/.

[46] D. A. Gilbert. Dependency and Exception Handling in an Asynchronous Microprocessor. PhD thesis, Department of Computer Science, University of Manchester, 1997.

[47] D. A. Gilbert and J. D Garside. A result forwarding mechanism for asynchronous pipelined systems. In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems (Async'97), pages 2-11, IEEE Computer Society Press, April 1997.

[48] L. A. G;asser and D.W. Dobberpuhl. The design and analysis of VLSI Circuits. Addison-Wesley, 1985.

[49] S. Hauck. Asynchronous design methodologies: An overview. Proceedings of the IEEE, 83(1):69-93, January 1995.

[50] H. Jacobson , E. Brunvand, G. Gopalakrishnan, and P. Kudva. High-level asynchromous system design using the ACK framework. In Proc. International Symposium on Advanced Research in

Asynchronous Circuits and Systems, pages 93-103. IEEE Computer Society Press, April 2000.

[51] J. Liu. Arithmetic and control components for an asynchronous microprocessor. PhD thesis, Department of Computer Science, University of Manchester, 1997.

[52] A. J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. Distributed Computing, 1(4):226-234, 1986.

[53] A. J. Martin. Synthesis of Asynchronous VLSI circuits, 1991.

[54] C. J. Myers. Asynchronous Circuit Design. John Wiley & Sons, July 2001.

[55] C. D. Nielson. Evaluation of function blocks for asynchronous design. In Proc.EURO-DAC, pages 454-459. IEEE Computer Society Press, September 1994.

[56] C. D. Nielson. Low Power Asynchronous VLSI Design. PhD thesis, Department of Information technology, technical University of Denmark, 1997.

[57] L. S. Nielson, C.Niesson, J. Sparso, and C. H. Van Berkel. Low-power operation using self-timed circuits and adaptive scaling of the supply voltage. IEEE Transactions on VLSISystems, 2(4): 391-397, 1994.

[58] A. M. G. Peeters. The 'Asynchronous' Bibliography. http://www.win.tue.nl/~wsinap/pdf/peeters96.pdf.

[59] Philips Semiconductors. PCA5007 handshake-technology paper IC data sheet. http://www.semiconductors.philips.com/pip/PCA5007H.