Towards an Ultra-Low Energy Computation with Asynchronous Circuits

Tsung-Te Liu

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-36
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-36.html

May 1, 2014

Towards an Ultra-Low Energy Computation with Asynchronous Circuits

by

Tsung-Te Liu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jan M. Rabaey, Chair
Professor Elad Alon
Professor Paul K. Wright

Spring 2012

Towards an Ultra-Low Energy Computation with Asynchronous Circuits

Abstract

Towards an Ultra-Low Energy Computation with Asynchronous Circuits

by

Tsung-Te Liu

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Jan M. Rabaey, Chair

Emerging biomedical applications would benefit from the availability of digital processors with substantially improved energy-efficiency. One approach to realize ultra-low energy processors is to scale the supply voltage aggressively to near or below the transistor threshold, yet the major increase in delay variability under process, voltage and temperature variations combined with the dominance of leakage power makes robust near- and sub-threshold computations and further voltage scaling extremely challenging.

This research focuses on the design and implementation of robust and energy-efficient computation architectures by employing an asynchronous self-timed design methodology. A statistical framework is first presented to analyze the energy and delay of CMOS digital circuits considering a variety of timing methodologies. The proposed analysis framework combines variability and statistical performance models, which enables designers to efficiently evaluate circuit performance, and determine the optimal timing strategy, pipeline depth and supply voltage in the presence of variability.

Two asynchronous self-timed designs are then implemented. First, a low-energy asynchronous logic topology using sense amplifier-based pass transistor logic (SAPTL) is presented. The SAPTL structure can realize very low energy computation by using low-leakage pass transistor networks at low supply voltages. The introduction of asynchronous operation in SAPTL further improves energy-delay performance without a significant increase in hardware complexity. The proposed self-timed SAPTL architectures provide robust and efficient asynchronous computation using a glitch-free protocol to avoid possible dynamic timing hazards.

Second, an asynchronous neural signal processor is presented to dynamically minimize leakage and to self-adapt to process variations and different operating conditions. The self-timed processor demonstrates robust sub-threshold operation down to 0.25V, while consuming only 460nW in a 65nm CMOS technology, representing a 4.4X reduction in power compared to the state-of-the-art designs. The proposed asynchronous design approach enables CMOS

circuits to fully benefit from continued technology scaling and realize ultra-low voltage operation, without incurring the leakage and variability issues associated with conventional synchronous implementations.

To my beloved family and Chiang-Min

Contents

List of Figures

List of Tables

Acknowledgments

The past seven years have been a wonderful and unforgettable journey. A lot of people helped me along the road, and I am extremely grateful to them. Here, I would like to take this opportunity to show my appreciation to some of them.

First, I would like to thank my advisor Professor Jan Rabaey for his guidance and support for my research. With his passion and vision, Jan shows me what is needed to be a researcher. I would also like to thank Professor Elad Alon for his valuable feedback, and Professor John Wawrzynek and Professor Paul Wright for being my qualifying and dissertation committee.

I had a valuable opportunity to work with Marly Roncken and Ivan Sutherland at the time I was looking for my research topic. They led me into this fantabulous asynchronous world and have been inspiring me ever since. They also showed me how to effectively present and deliver a research idea. I really appreciate their encouragement and help with my research work.

I would like to thank Louis Alarcon for being my mentor as well as tapeout partner. He was always the first one I talked to whenever I had a research idea or problem. I would also like to thank everyone at Berkeley Wireless Research Center for the help during my Ph.D. study, especially during the tapeout time. Special thank goes to Jun-Chau Chien for his valuable assistance to my processor tapeout.

Apart from the research life, hanging out with my friends in Berkeley Association of Taiwanese Students has been always enjoyable. I especially want to thank Darsen Lu for being my partner of various activities and the help on all kinds of occasions.

Hsiu-Yu Fan has been an important part of my life during the later stages of my Ph.D. study. I want to thank her for the love and support, especially staying up with me during numerous late-night research works. Her knowledge and philosophy truly enrich my life, and I really treasure her company.

I would like to thank Chiang-Min Cheng for the love, support and understanding. Without her, I would not be able to start this journey at Berkeley, or to accomplish anything near significant. She will be always inside my heart, and be the reason that keeps me fighting for the rest of my life.

Last, but not the least, I would like to thank my family, especially my parents and sister. Their love and unconditional support get me through each challenge, and keep me warm at windy Berkeley. I love them from the bottom of my heart.

Chapter 1

Introduction

Researches on energy-efficient circuit and system design have been drawing a lot of attentions from both the industry and the academia since the last decade for the following reasons: First, although the speed of CMOS devices keeps improving due to the efforts on continued technology scaling, further power and energy reduction via technology scaling have become extremely difficult due to issues of leakage and variability [Nowak02]. Second, technology scaling combined with advances in system integration and wireless communication has enabled a new set of electronic devices that used to only exist in science fictions, including ubiquitous sensor swam and miniature medical devices [Rabaey08]. However, these emerging devices can only work on an extremely stringent energy budget. As a result, in order to accommodate the demands from these application and future technology scaling, it is essential to develop alternate devices and novel circuit architectures for ultra-low energy computation.

1.1 Overview

This work presents the design and implementation of robust and energy-efficient computation architectures by employing an asynchronous self-timed design methodology. A statistical analysis framework is first presented to evaluate the energy and delay of CMOS circuits considering a variety of timing methodologies in the presence of variability. Two asynchronous self-timed designs are then implemented. First, a low-energy asynchronous circuit topology using sense amplifier-based pass transistor logic is presented. Second, an ultra-low power asynchronous neural signal processor for brain-machine interface applications is presented.

1.2 Dissertation Outline

Chapter 2 provides an overview of ultra-low energy digital circuit design in the context of voltage scaling. Leakage and variability issues associated with voltage and technology scaling

are discussed. Various circuit techniques for ultra-low voltage operation are also summarized in this chapter.

In chapter 3 an asynchronous self-timed design methodology is introduced as an attractive alternative for the realization of robust and energy-efficient computation. Both the advantage and challenge of design of an asynchronous system are discussed. The energy and delay of synchronous and asynchronous circuits in the presence of process variability are evaluated within a statistical analysis framework.

Chapter 4 presents an asynchronous circuit topology using the sense amplifier-based pass transistor logic (SAPTL). The bundled-data and the dual-rail self-timed protocols are employed to realize asynchronous self-timed SAPTLs, respectively. Low-energy handshaking protocol design and circuit implementation are presented. Results of simulation and measurement on various self-timed SAPTL circuits are compared in 90nm CMOS technology.

Chapter 5 presents an asynchronous neural signal processor for brain-machine interface applications. The background and motivation of on-line neural signal processing are first introduced. Design methodology and circuit implementation of an ultra-low power asynchronous neural signal processor are then presented. Results of measurement on both synchronous and asynchronous designs from a 65nm test chip are compared with the state-of-the-art processors.

Chapter 6 summarizes and concludes this research. Future research directions are also suggested.

Chapter 2

Ultra-Low Energy Digital Circuit Design

Voltage scaling has been demonstrated as the most powerful way of reducing digital computation power and energy [Chandrasakan92]. However, as the supply is scaled near or below the device threshold, dramatic increases in leakage and variability severely limit the minimum power and energy that can be achieved. The research goal of this work is therefore to develop circuit techniques and design methodology addressing these two issues, and thus further minimizing power and energy consumption of digital circuits.

In this chapter, an overview of ultra-low energy digital circuit design via aggressive voltage scaling is presented. Section 2.1 first introduces the background and principle of voltage scaling. Leakage and variability, two major challenges of low-voltage operation, are then discussed in section 2.2. Section 2.3 presents and compares the state-of-the-art system and circuit design techniques for low-voltage operation. Section 2.4 concludes this chapter.

2.1 Opportunity of Low-Voltage Design

Historically, technology scaling is usually accompanied by voltage scaling. Fig. 2.1 shows the supply voltages used in different CMOS technology nodes for both high-performance and low-power applications [ITRS09]. For long-channel CMOS device, ideally, both transistor density and speed can be increased from technology scaling while still keeping the total power density constant, if the device threshold and supply voltage are also scaled by the same ratio [Dennard74]. However, for short-channel devices in deep-submicron CMOS technology, voltage scaling has been slowing down for the recent CMOS technology nodes due to leakage and variation issues. This results in dramatic increase in power density [Nowak02]. To address this problem, on the one hand, researchers have been looking for "new device" that has better performance characteristics to extend CMOS technology scaling. On the other hand, developing "new circuit" architectures that exhibit lower leakage and better reliability is essential in order to take the full advantage of technology scaling.

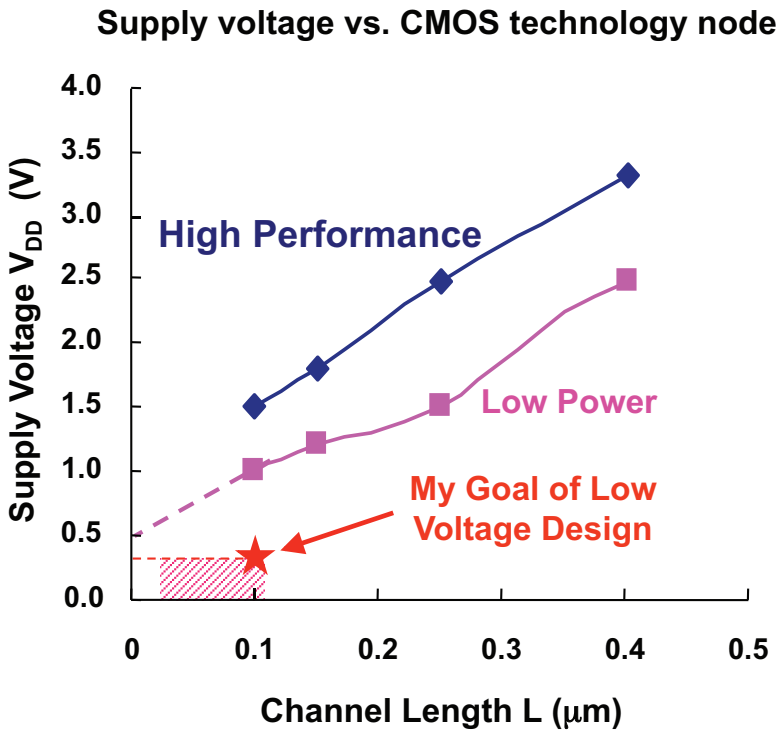

Figure 2.1: Supply voltages used in different CMOS technology nodes [ITRS09].

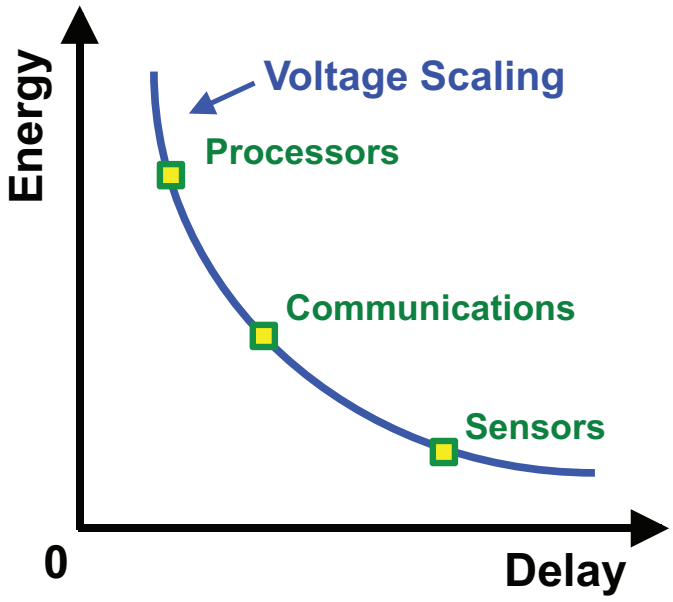

Figure 2.2: Energy-delay characteristics of different electronic applications.

On the application side, emerging sensor applications such as miniature medical devices have drawn huge attention in recent years. These applications have the potential to completely change people's life [Rabaey08]. Since these sensor devices have a very small form factor, they usually operate with a tiny battery, or even rely only on the energy scavenged directly from the environment. As a result, the energy budget of the entire device is severely limited, but the speed performance requirement is usually much relaxed compared to the high-end processors, as shown in Fig. 2.2. As a result, minimizing the energy consumption while meeting the application-specific constraint is the most important design issue for these ultra-low energy applications.

If the ultimate design goal is to minimize digital energy and power, and circuit leakage can be neglected, digital circuits should operate at a supply voltage below the nominal supply to reduce dynamic power and energy consumption [Chandrasakan92]. The dynamic power $P_{dynamic}$ and energy $E_{dynamic}$ of a CMOS gate can be expressed as [Rabaey03]:

$$P_{dynamic} = \alpha C_{switch} V_{DD}^2 f \tag{2.1}$$

$$E_{dynamic} = \alpha C_{switch} V_{DD}^2 \tag{2.2}$$

where α is the activity factor, C_{switch} is the switched capacitance, and f is the operating frequency. Eqs. 2.1 and 2.2 show that voltage scaling has a dramatic impact on dynamic energy and power consumption due to its quadratic dependency. This characteristic is therefore very attractive to these emerging sensor applications that demand for extremely low energy consumption.

While Eq. 2.2 suggests that supply voltage should be lowered as much as possible to minimize the dynamic energy, the total energy consumption, however, may actually increase at low supply voltages due to circuit leakage and variability. Unfortunately, as CMOS technology continues to scale, these two issues become even severe, which significantly limits the energy reduction that can be realized from voltage scaling. In the next section, leakage and variability, two main challenges of low-voltage design, are discussed.

2.2 Challenge of Low-Voltage Design

2.2.1 Leakage

For portable devices, leakage plays a crucial role since it determines the standby power and thus the standby time. At low supply voltages, the dominant leakage source is the weak inversion, or sub-threshold conduction current between source and drain even if the gate voltage is smaller than the threshold voltage [Chandrakasan01]. This off-state leakage current can be expressed as [Taur98]:

$$I_{leak} = \mu_{eff} C_{ox} \frac{W}{L_{eff}} (m-1) V_T^2 e^{\frac{-V_{TH}}{mV_T}} \left(1 - e^{-\frac{V_{DD}}{V_T}}\right) \tag{2.3}$$

where

$$m = \left(1 + \frac{C_d}{C_{ox}}\right), \tag{2.4}$$

C_{ox} is the gate-oxide capacitance per unit area, C_d is the depletion region per unit area, W is the transistor width, L_{eff} is the effective channel length, and $V_T = \frac{kT}{q}$. Note that the sub-threshold leakage current also reduces as supply voltage is lowered due to the effect of drain induced barrier lowering (DIBL).

The power and energy per operation of a CMOS gate, P_{total} and E_{total}, including both active and leakage components can thus be expressed as

$$P_{leak} = V_{DD} I_{leak} \tag{2.5}$$

$$E_{leak} = V_{DD} I_{leak} T_{delay} \tag{2.6}$$

$$P_{total} = P_{dynamic} + P_{leak} = \alpha C_{switch} V_{DD}^2 f + V_{DD} I_{leak} \tag{2.7}$$

$$E_{total} = E_{dynamic} + E_{leak} = \alpha C_{switch} V_{DD}^2 + V_{DD} I_{leak} T_{delay} \tag{2.8}$$

where T_{delay} is the computation delay of a CMOS gate. Eqs. 2.5 and 2.6 show that by scaling down supply voltage, both leakage power and total power consumption can be reduced. However, when the supply voltage approaches the threshold voltage, leakage energy and total energy consumption actually start to increase. This is because the computation delay increases exponentially when the supply voltage is scaled below the threshold voltage, which offsets the leakage power reduction from voltage scaling. In general, because of the exponential increase in delay, the minimum energy point of CMOS digital system exists in sub-threshold or near-threshold regions [Wang05].

To estimate the energy consumption characteristics of low-activity sensor circuit, the energy consumption of a 423-stage fanout-of-four (423FO4) inverter-based ring oscillator for different supply voltages is simulated in 90nm CMOS technology, as shown in Fig. 2.3. The minimum energy point of this ring oscillator circuit is around 300mV. Below it, further voltage scaling is useless because the leakage energy dominates the total energy consumption.

2.2.2 Variability

The second issue of low-voltage digital computation is variability. Process variations continue to increase dramatically with CMOS technology scaling. The transistors will have different die-to-die (inter-die; DTD) and within-die (intra-die; WID) variation behaviors after actual manufacturing process [Bowman02]. DTD variations consisting of parametric variations between different runs, lots, and wafers, affect each transistor equally on the same die. On

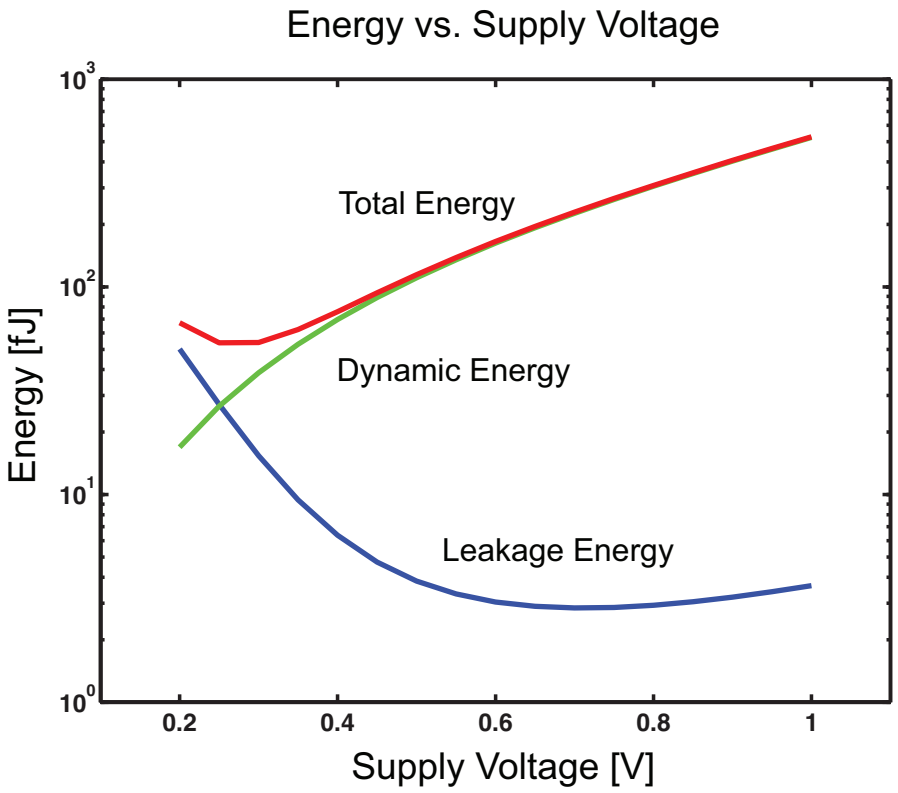

Figure 2.3: Energy consumption of an inverter-based ring oscillator as a function of supply voltage.

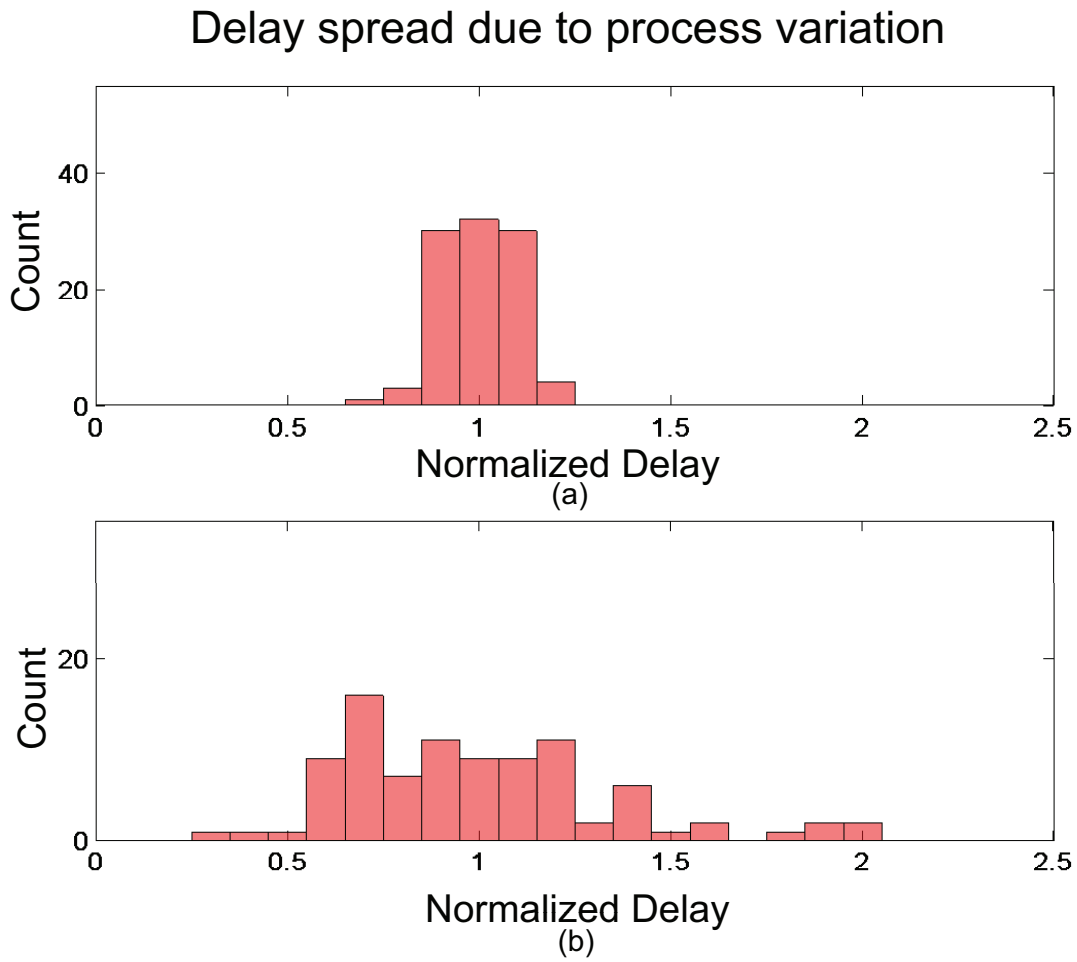

Figure 2.4: Delay distribution of a 4FO4 inverter chain under process variations (a) at 1V and (b) at 300mV.

the other hand, WID variations result in random fluctuations of transistor characteristics within the same die. These variations severely increase the performance variability of CMOS digital circuits, and their impacts are especially significant at low supply voltages. Fig. 2.4 shows the delay distribution of a 4FO4 inverter chain under process variations in 90nm CMOS technology using Monte Carlo SPICE simulator. A delay spread of 70% at 1V and of 190% at 0.3V is observed. This demonstrates that at low supply voltages, circuit delay becomes highly sensitive to device parameters due to the lower circuit overdrive voltage.

In actual operating environment, unfortunately, other variation sources must also be considered, such as supply voltage and temperature variations. Fig. 2.5 shows the computation delay of a 16FO4 inverter chain in 65nm CMOS technology on the blue curve for supply voltages ranging from 300mV down to 200mV. The red curve illustrates the worst-case delay

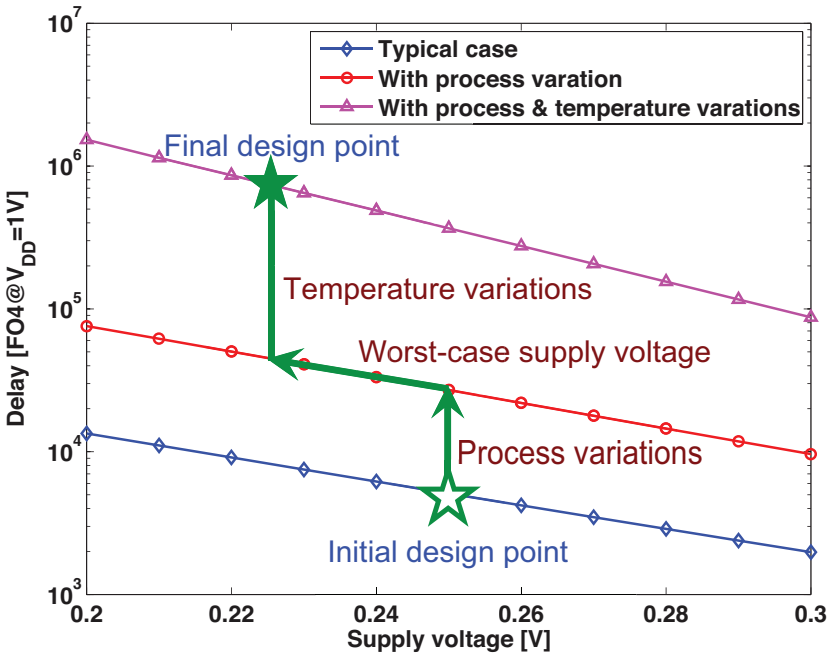

Figure 2.5: Delay penalty of the worst-case design methodology under process, voltage and temperature variations.

considering process variations, while the purple curve shows the worst-case delay considering both process and temperature variations. Assuming that the initial design point is to operate at 250mV, in order to guarantee the circuit functionality for the worst-case scenario, the circuit speed must be slowed down by 5X for process variations, by 9X if 10% supply change is further considered, and by up to 147X for process, voltage and temperature variations. As a result, in the presence of severe delay variability, using traditional worst-case design methodology to ensure reliability is very expensive at low supply voltages.

Furthermore, this extra delay margin to guarantee the circuit functionality for the worst-case scenario not only slows down the overall circuit speed during typical operation, but also translates into extra leakage waste because circuits must be idle longer. The total leakage energy including this extra idle leakage can be rewritten as

$$E_{leak} = V_{DD} I_{leak} \left(T_{delay} + T_{margin} \right) \tag{2.9}$$

Fig. 2.6 shows that both leakage energy and total energy increase after extra timing margin is included for delay uncertainty. The amount of total energy increase is especially significant at low supply voltages where leakage dominates.

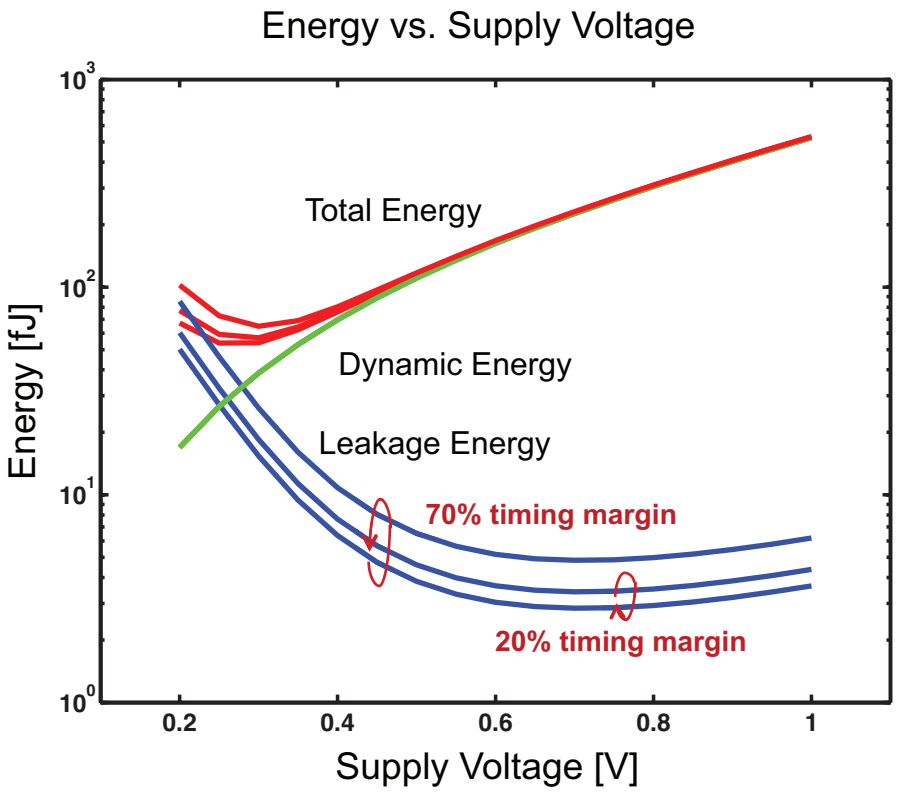

Figure 2.6: Leakage and total energy increase as more timing margin is reserved.

2.3 Low-Voltage Design Techniques

Various circuit techniques have been proposed to reduce leakage consumption at design time. Header and footer switches with multiple threshold voltage have already been widely employed in today's mobile devices to minimize standby leakage [Sakurai03]. Other popular leakage control techniques including the use of non-minimum channel length devices, stacked transistors and body-biasing, are discussed and well summarized in [Chatterjee03]. Novel circuit topologies exhibiting better effective on-to-off operating current ratio than traditional static CMOS structure have also been proposed for low-voltage operation, such as sense amplifier-based pass transistor logic [Alarcón07] and Schmitt-Trigger logic [Lotze12]. Most low-voltage designs today employ several aforementioned leakage reduction techniques simultaneously to minimize leakage.

In order to reduce the impact of process variations, designers can use regular layout structures [Pang09] and variation-aware design methodologies [Ickes12] at design time to minimize performance variability due to process variations. After chips are fabricated, post-silicon tuning is widely used to recover performance and thus increase yield. Post-silicon tuning is usually accomplished by first measuring performance characteristics of replica process monitor, and then re-adjusting supply voltage and body bias to meet performance and yield specifications [Hanson08]. Since this approach estimates circuit variability indirectly using replica process-control circuits, it is only effective to recover performance and yield loss from DTD variations, but incapable of capturing characteristics of WID variations and dynamic variations such as supply ripples and real-time changes in circuit activity.

To deal with dynamic variations, most approaches requires additional tracking and feedback loops [Kurd09], which are usually very costly or difficult to deploy at fine-grained levels at low supply voltages. System architecture employing inherent error detection and correction circuitry has been proposed to deal with performance variability [Bul111, Bowman11]. This approach uses additional latch and error detector along the circuit critical path to identify system failure due to insufficient timing margin. The error events are then either corrected with correction circuitry, or fixed at the system level. Since the error detection and correction are implemented in the main datapath, this approach is effective to reduce the impact of DTD, WID and dynamic variations. However, since only the error event happening before the end of next clock cycle can be detected, the error-recovery range of this technique is very limited, up to 2X delay variations reported in literature. Therefore, it is not sufficient for ultra-low voltage operation where delay variations can easily exceed 10X.

2.4 Summary

Voltage scaling is the most effective way of reducing digital power and energy consumption. However, dramatic increase in leakage and variability resulted from both technology and voltage scaling makes further energy reduction extemely difficult. Leakage energy dominates

the total energy consumption at low supply voltages. Variability slows down the circuit operating speed, which futher increases idle leakage and total enegy consumption.

Various design techniques have been proposed to minimize circuit leakage and variability at the circuit and system levels. However, most techniques either require significant implementation cost, or have restricted application or exhibit performance limit at low supply voltages. In the next chapter, an asychornous self-timed design apporach is introduced to realize robust and energy-efficient computation at low supply voltages.

Chapter 3

Design and Analysis of Asynchronous Circuits

Traditionally, asynchronous timing strategies have been advocated to enhance the speed and ease the global clocking problem in high-performance digital systems [Sparsø01]. However, the built-in timing mechanisms that automatically adapt to different operating conditions make asynchronous design an attractive alternative for the realization of robust and energy-efficient computation at low supply voltages. In this chapter, energy and delay performances of CMOS circuits with different timing methodologies in the presence of variability are evaluated with a statistical analysis framework. The sweet spots of different timing schemes are then demonstrated. Section 3.1 presents the opportunities and implementation challenges of asynchronous self-timed circuits. Section 3.2 introduces a delay variability model of CMOS digital circuit across a wide range of supply voltage and logic depth. Section 3.3 presents a statistical performance model to analyze the minimum delay overheads required to implement different timing schemes under process variations. Section 3.4 evaluates energy and delay performances of synchronous and asynchronous CMOS digital circuits, based on the variability and statistical performance models developed in section 3.2 and section 3.3. Section 3.5 concludes this chapter.

3.1 Opportunity and Challenge

Process variations continue to increase dramatically with CMOS technology scaling. These variations severely increase the performance variability of CMOS digital circuits [Bowman02]. Fig. 3.1 shows the typical delay distribution of digital circuit in the presence of process variations on the black curve. Traditional synchronous approach uses a very conservative timing margin to meet a certain reliability requirement. As a result, a synchronous design must slow down for the "worst-case" scenario, which causes a design to fail to exercise the whole capacity after actual fabrication. The red curve shown in Fig. 3.1 illustrates the performance loss using synchronous timing scheme after including extra timing margin

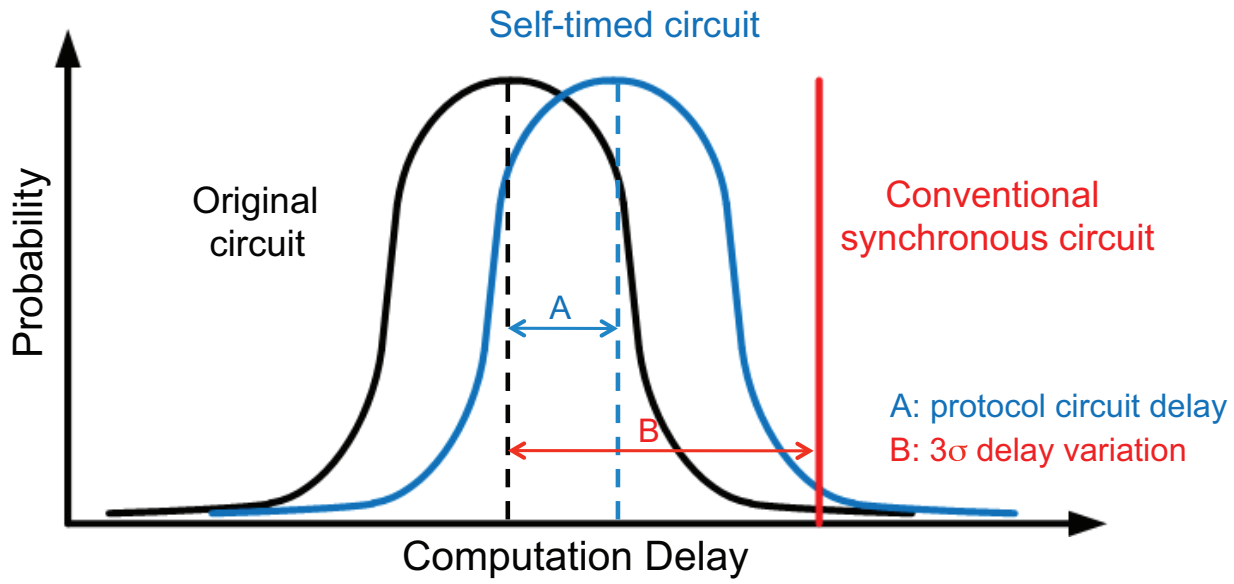

Figure 3.1: Performance characteristics of original (black), synchronous (red) and asynchronous (blue) digital circuits.

for delay variability. Furthermore, as the variation keeps increasing, this performance loss becomes larger and might offset all the performance enhancements from technology scaling.

Asynchronous self-timed approach, on the other hand, can get the best performance in the presence of variability while still guaranteeing circuit reliability. Fig. 3.1 also shows the performance characteristic of asynchronous circuit on the blue curve. Ideally, the statistical performance of asynchronous circuit can closely follow the statistical profile of actual circuit behavior and achieve "average-case" performance.

In addition to speed enhancement, asynchronous computing can also achieve better power performance. Since the accurate timing information can be immediately obtained for an asynchronous system, circuit computation can still be realized efficiently even in the presence of variations. Fig. 3.2 shows the operating power profiles of synchronous and asynchronous systems. An asynchronous design can response and minimize the standby leakage immediately after an operation completes, instead of being limited by the worst-case delay. On the other hand, a traditional synchronous design must slow down for the worst-case variation because it has no information of real-time circuit behavior. This causes extra timing slack most of time, which translates into extra leakage waste.

Despite a better statistical performance of asynchronous operation, the overhead cost to implement a handshaking protocol might be nontrivial and may offset all the statistical performance advantages. A statistical analysis framework is therefore necessary to compare different timing schemes and determine the optimal approach in the presence of variability.

In summary, the major implementation challenge of self-timed circuit is the additional energy, delay and hardware cost from the protocol circuitry. The self-timed approach only

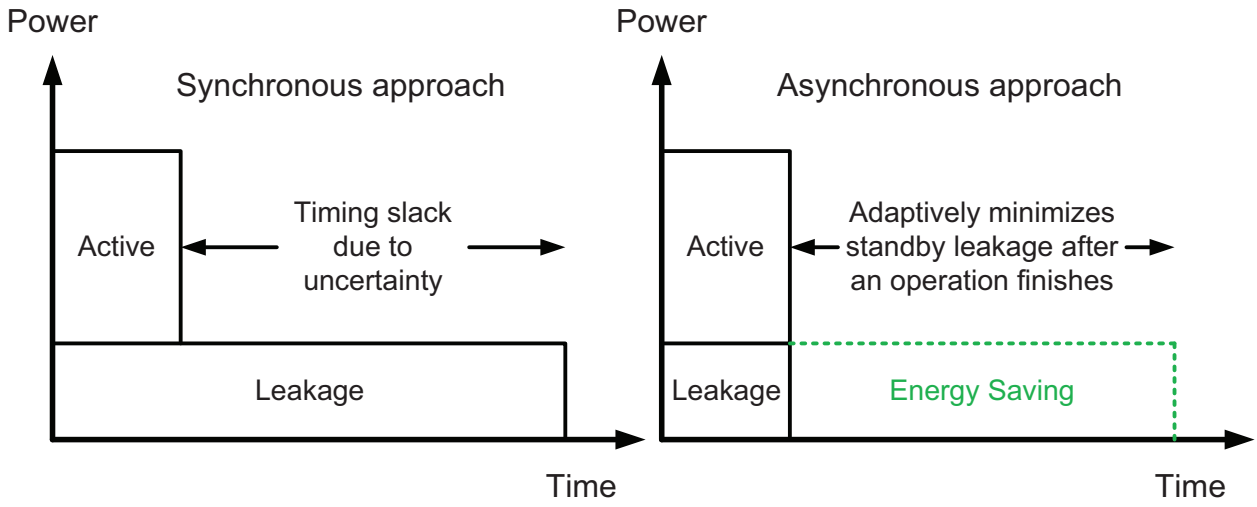

Figure 3.2: Power profiles of synchronous and asynchronous systems.

makes sense when this protocol circuit overhead is smaller than the circuit delay variability. On the other hand, self-timing is very attractive to realize ultra-low power digital circuits, since variations is relatively large at low supply voltages. A self-timed design can achieve average-case performance in the presence of variability. Moreover, it can dynamically control leakage paths, realizing real-time power management at fine-grain level and minimizing leakage as much as possible. This leakage minimization techinique is demonstrated in two design examples introdued in chapter 4 and chapter 5. Finally, since a self-timed design operates based on request-acknowledgement handshaking control protocols, it is the most robust solution for ultra-low voltage operation.

In the following sections, a statistical analysis framework is presented to efficiently evaluate energy and delay performances of CMOS circuits with different timing schemes under process variations. The communication costs of implementing different self-timed pipeline protocols have been modeled and compared in [Stevens11], and are not discussed here. Instead, a statistical model is presented to estimate the computation costs and performance upper bounds of generic synchronous, asynchronous bundled-data and dual-rail self-timed approaches for various operating points. This is accomplished by evaluating the minimum delay overhead required to guarantee circuit functionality in the presence of variability across a wide range of supply voltage and logic depth. Note that the specific variation-tolerant circuit techniques in synchronous design such as post-silicon adaptive tuning [Hanson08], or other hybrid-timing methodologies like globally-asynchronous locally-synchronous (GALS) approach [Muttersbach00], are not discussed. The proposed analysis framework, however, can be easily extended or modified to evaluate the statistical performances of the aforementioned approaches.

3.2 Modeling of CMOS Digital Circuit

A MOSFET transistor exhibits dramatically different performance characteristics in the strong-inversion and sub-threshold regions. In order to efficiently estimate the delay variability across a wide range of supply voltage, it is essential to have a generic analytical model that describes transistor behaviors across different operating regions. A unified current model that captures the most essential physical characteristics of MOSFET transistor and exhibits excellent model scalability as derived in is used in this work [Cao07]. The device current I is described as:

$$I \propto \frac{\left\{\ln\left[1+\exp\left(\frac{V_{DD}-V_{th}}{2S}\right)\right]\right\}^2}{\left\{1+\ln\left[1+\exp\left(\frac{V_{DD}-V_{th}}{m}\right)\right]\right\}} \tag{3.1}$$

where S is the sub-threshold swing parameter and m is the parameter modeling the effect of velocity saturation.

In the strong-inversion region where the exponential term dominates, Eq. 3.1 becomes the well-known square-law formula with velocity saturation effect

$$I \propto \frac{(V_{DD}-V_{th})^2}{1+\left(\frac{V_{DD}-V_{th}}{m}\right)} \tag{3.2}$$

Similarly, in the sub-threshold region, the exponential term is much smaller than the constant 1 term. Eq. 3.1 can be simplified as

$$I \propto \exp\left(\frac{V_{DD}-V_{th}}{S}\right) \tag{3.3}$$

3.2.1 Delay Model

The delay model of CMOS gate across a wide range of supply voltage can now be derived based on Eq. 3.1. After substituting Eq. 3.1 into the Alpha-power law model for device current, the gate delay T_d can be expressed as [Sakurai90, Cao07]:

$$T_d \propto \frac{CV_{DD}}{I} = \frac{K \cdot V_{DD} \cdot \left\{1+\ln\left[1+\exp\left(\frac{V_{DD}-V_{th}}{m}\right)\right]\right\}}{\left\{\ln\left[1+\exp\left(\frac{V_{DD}-V_{th}}{2S}\right)\right]\right\}^2} \tag{3.4}$$

where k is the delay-fitting parameter.

To extract delay model parameters in Eq. 3.2, supply voltages from 300mV to 1V are swept for an industrial 65nm CMOS technology. Fig. 3.3 shows the nominal delay predicted by Eq. 3.4 and simulated results as a function of V_{DD} for a four-stage fanout-of-four (4FO4)

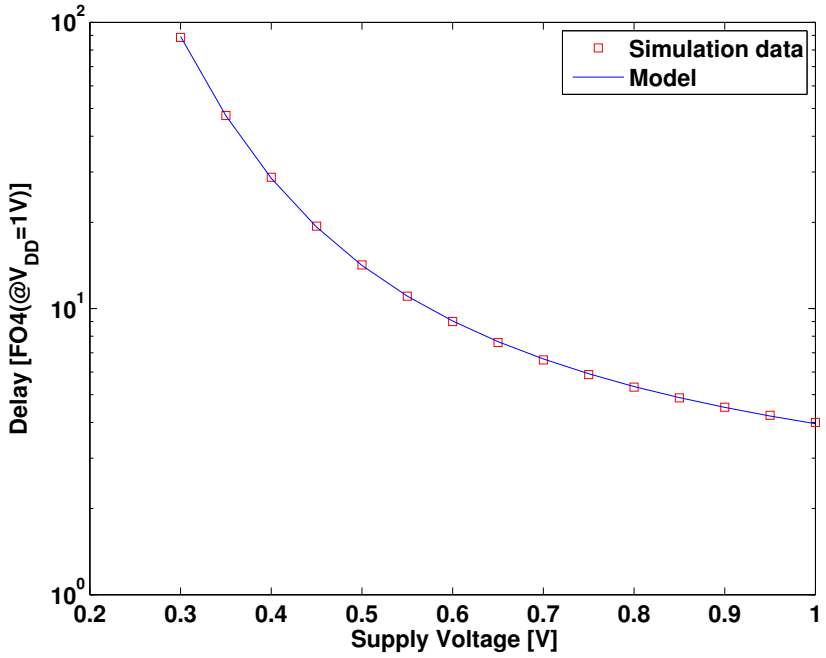

Figure 3.3: Simulated and modeled delay of a 4FO4 inverter chain in 65nm CMOS technology.

inverter chain. The model error is smaller than 5% across the entire range of supply voltage, which demonstrates the accurate prediction of CMOS gate delay in both the strong-inversion and sub-threshold regions. The unified delay model Eq. 3.4 is used in the next section as a baseline to derive a delay variability model.

3.2.2 Variability Model

To estimate the delay variability of a CMOS gate, two major parametric variation components are considered: threshold voltage variations due to random dopant fluctuations, and geometric variations due to fluctuations of device length, width and oxide thickness [Ghosh10]. If the variations in devices parameters follow the Gaussian statistical distribution, the delay variability of CMOS gate can be estimated directly from Eq. 3.4. The normalized variation of gate delay $\frac{\sigma_{T_d}}{\mu_{T_d}}$ due to parametric variations can be thus given by

$$\frac{\sigma_{T_d}}{\mu_{T_d}} = \sqrt{\left(S_{T_d}^{v_{th}}\right)^2 \cdot \left(\frac{\sigma_{V_{th}}}{\mu_{V_{th}}}\right)^2 + \left(S_{T_d}^{K}\right)^2 \cdot \left(\frac{\sigma_K}{\mu_K}\right)^2} \tag{3.5}$$

where

$$S_{T_d}^{V_{th}} = \frac{\partial V_{th}/V_{th}}{\partial T_d/T_d} \quad and \quad S_{T_d}^{K} = \frac{\partial K/K}{\partial T_d/T_d} \tag{3.6}$$

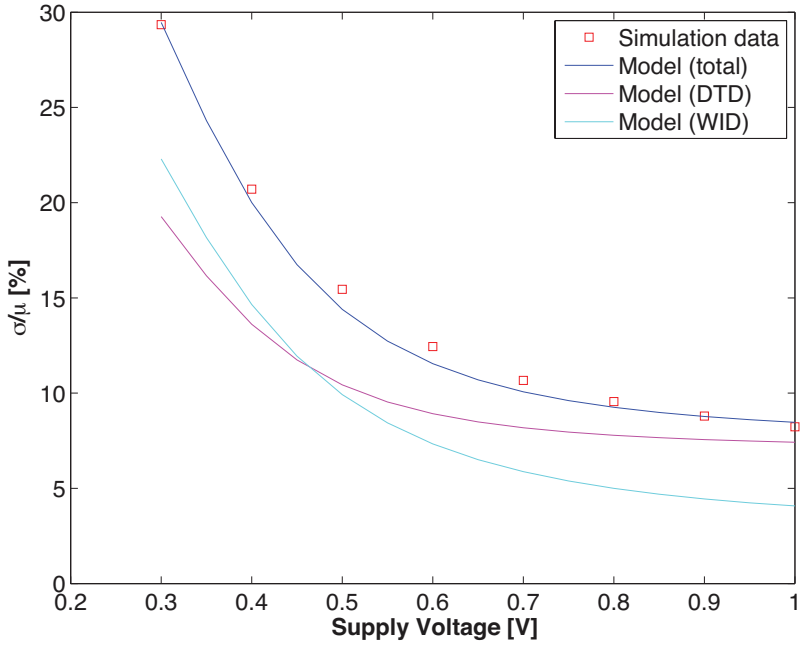

Figure 3.4: Simulated and modeled delay variability results.

After actual manufacturing process, the transistors will have different die-to-die (inter-die; DTD) and within-die (intra-die; WID) variation behaviors [Bowman02]. Both DTD and WID variation behaviors can be estimated using Eqs. 3.5 and 3.6. However, since DTD and WID variations affect circuit performances in completely different ways, they must be estimated separately to derive the total delay variability model accounting for both variations. If DTD and WID variations are completely independent, the combined effects on delay variability of CMOS gate can be expressed as

$$\frac{\sigma_{T_{d,total}}}{\mu_{T_{d,total}}} = \sqrt{\left(\frac{\sigma_{T_{d,DTD}}}{\mu_{T_{d,DTD}}}\right)^2 + \left(\frac{\sigma_{T_{d,WID}}}{\mu_{T_{d,WID}}}\right)^2} \tag{3.7}$$

Fig. 3.4 shows the estimated delay variation results using Eqs. 3.5 to 3.7 as a function of V_{DD} for a 4FO4 inverter chain, as well as the simulation results using Monte Carlo SPICE simulator. The simulation results show that the model accurately predicts delay variability due to DTD and WID variations. DTD variation is the dominant delay variation source at high supply voltages, while the impact of WID variation grows rapidly as the supply voltage is lowered. The model error is smaller than 8% across the entire range of supply voltage. In this work, Eqs. 3.5 to 3.7 are used as delay variability models to estimate the delay variability of CMOS circuit under process variations. More parametric variation components can be easily incorporated in Eqs. 3.5 and 3.6 to increase model accuracy.

3.2.3 Circuit Delay Variability with Different Logic Depths

If WID variations are completely random, it has been shown that a longer logic path with more gate stages is expected to have less variability from WID variations since timing variations are averaged out [Eisele96]. However, circuits with longer logic path cannot reduce the impact of DTD variations. If each state delay in a logic path is completely independent and random, Eq. 3.7 can be modified to include the effect of logic depth on delay variability as

$$\frac{\sigma_{T_{d,total\_n}}}{\mu_{T_{d,total\_n}}} = \sqrt{\left(\frac{\sigma_{T_{d,DTD\_4}}}{\mu_{T_{d,DTD\_4}}}\right)^2 + \left(\frac{4}{n}\right) \cdot \left(\frac{\sigma_{T_{d,WID\_4}}}{\mu_{T_{d,WID\_4}}}\right)^2} \tag{3.8}$$

where n is the logic depth, $\frac{\sigma_{T_{d,DTD\_4}}}{\mu_{T_{d,DTD\_4}}}$ and $\frac{\sigma_{T_{d,WID\_4}}}{\mu_{T_{d,WID\_4}}}$ are the normalized DTD and WID delay variability of a 4FO4 inverter chain, respectively. The delay variability model of a 4FO4 inverter chain is employed as a baseline model to estimate the delay variability of an nFO4 inverter chain. To verify the scalability of Eq. 3.8, the delay variability results predicted by Eq. 3.8 are compared with the Monte Carlo SPICE simulation results for $n = 8$ and 24, as shown in Fig. 3.5. The model error is smaller than 13% for $n = 8$, and smaller than 15% for $n = 24$ across the entire range of supply voltage. Since delay variability models Eqs. 3.5 to 3.8 are derived directly from a closed-form formula, the impact of process variations on circuit performance across a wide range of supply voltage and logic depth can be efficiently estimated, instead of performing extensive Monte Carlo SPICE simulations. This enables designers to perform design exploration and system optimization much more efficiently at early design stage.

In the next section, a statistical model is introduced to evaluate performance characteristics of synchronous and asynchronous digital circuits in the presence of delay variability.

3.3 Statistical Analysis of Synchronous and Asynchronous Timing Schemes

In order to deal with the delay variability due to process variations, traditional synchronous digital circuits must slow down to meet a certain reliability requirement. Asynchronous circuits, on the other hand, can exploit local timing information to achieve both reliability and "average-case" performance in the presence of variability. Although the advantages of asynchronous timing are well known, the actual statistical performance gain and implementation overhead of asynchronous circuit under process variations have not been investigated quantitatively. In this section, a statistical model is introduced to analyze the performances of synchronous and asynchronous circuits in the presence of delay variability.

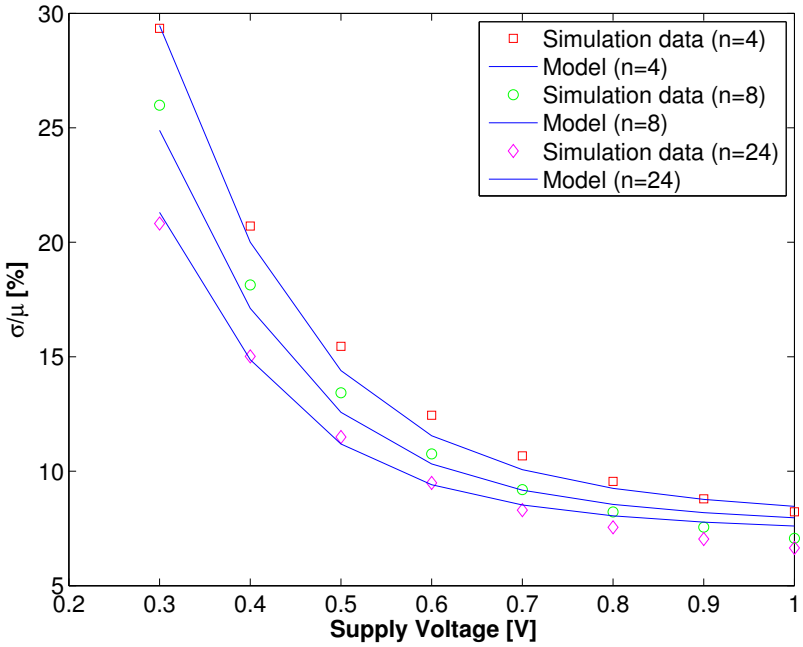

Figure 3.5: Simulated and modeled delay variability results for different logic depths.

3.3.1 Synchronous Approach

If process variations in a digital circuit follow the Gaussian statistical distribution, the delay variability of a critical path can be modeled as a normal distribution with a probability density function

$$f_{logic}(t) = N\left(\mu_{logic}, \sigma^2_{logic}\right) \tag{3.9}$$

where μ_{logic}, and σ_{logic} are the mean and standard deviation of the critical path delay, respectively. The probability of a critical path satisfying a specified period of synchronous clock t_{sync} can be given by

$$P\left(t_{logic} \leq t_{sync}\right) = \int_0^{t_{sync}} f_{logic}(t)\, dt \tag{3.10}$$

In order to achieve a yield of 99.7%, t_{sync} is set to $(\mu_{logic} + 3\sigma_{logic})$ in conventional synchronous designs. As a result, a "3σ worst-case" synchronous design will fail to exercise the whole circuit capacity after actual fabrication most of time, which indicates a statistical performance loss. In this work, this statistical performance loss of synchronous design under process variations is defined as a delay overhead required to implement a synchronous

timing scheme. Both DTD and WID variations contribute to delay overhead in conventional synchronous designs. The normalized delay overhead D_{sync} of synchronous design is given by

$$D_{sync} = \frac{3\sigma_{logic,total}}{\mu_{logic,total}} \tag{3.11}$$

3.3.2 Bundled-Data Self-Timed Approach

Bundled-data self-timed approach employs an additional delay line to generate local control signals instead of using a global clock [Sparsø01]. The delay line is designed to exhibit similar delay characteristics to the circuit critical path. Therefore, a bundled-data approach is effective to eliminate the impact of DTD variations. WID variations, however, would still cause delay mismatches between a critical path and a delay line, which poses a major performance limitation in a bundled-data self-timed design. The statistical performance of bundled-data self-timed scheme is evaluated by assuming that the delay variability of delay line can also be modeled as a normal distribution

$$f_{delay-line}(t) = N\left(\mu_{delay-line}, \sigma^2_{delay-line}\right) \tag{3.12}$$

In order to ensure a reliable timing generation, the delay of delay line must be always larger than the critical path delay in the presence of process variations. This timing constraint is specified by

$$P\left(t_{logic} \leq t_{delay-line}\right) \approx 1 \tag{3.13}$$

In actual circuit implementation, a delay line is usually just a replica of main circuit critical path with more gate stages. Therefore, it is reasonable to further assume that a replica delay line and a circuit critical path exhibit similar statistical characteristics that can be expressed as

$$\mu_{delay-line} - \mu_{logic} = D_{bundled-data} \cdot \mu_{logic} \tag{3.14}$$

$$\sigma^2_{delay-line} = (D_{bundled-data} + 1) \cdot \sigma^2_{logic} \tag{3.15}$$

where $D_{bundled-data}$ is the normalized delay overhead of bundled-data self-timed scheme, which indicates the number of additional gate stage required for a replica delay line to satisfy Eq. 3.13. If the final yield requirement is still 99.7%, $D_{bundled-data}$ can be calculated by solving Eqs 3.12 to 3.15:

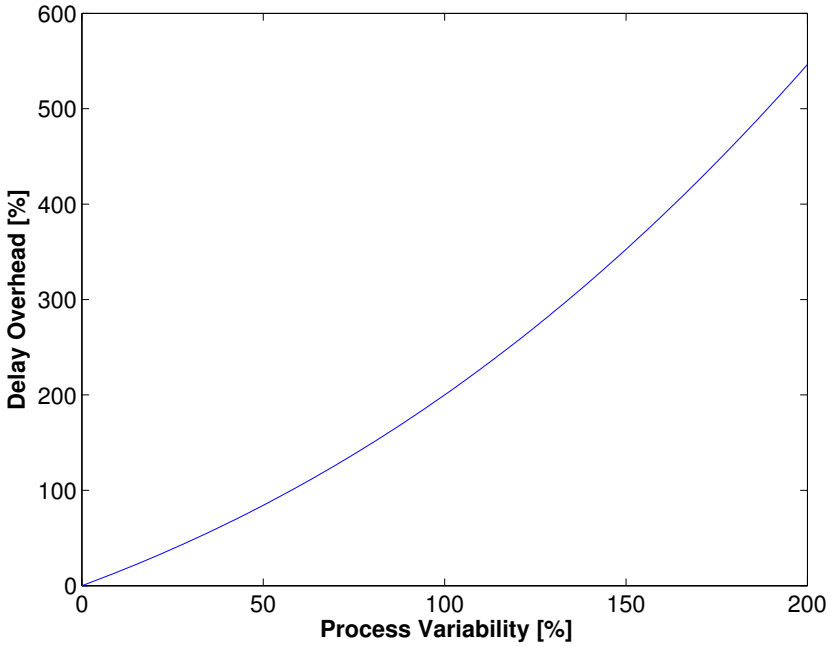

Figure 3.6: Delay overhead characteristics of bundled-data self-timed scheme.

$$D_{bundled-data} = D_{variation}^2 \cdot \left(\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{2}{D_{variation}^2}} \right) \tag{3.16}$$

where

$$D_{variation} = \frac{3\sigma_{logic,WID}}{\mu_{logic,WID}} \tag{3.17}$$

$$D_{bundled-data} \propto \begin{cases} \sqrt{2} \cdot D_{variation} & when\ D_{variation} \to 0 \\ D_{variation}^2 & when\ D_{variation} \to \infty \end{cases} \tag{3.18}$$

Note that only WID variation contributes to $D_{variation}$ in a bundled-data design. Fig. 3.6 shows the normalized delay overhead $D_{bundled-data}$ as a function of process variability $D_{variation}$. It is clear that the delay overhead of bundled-data self-timed scheme has different characteristics as the amount of process variation changes. While $D_{bundled-data}$ increases linearly with small $D_{variation}$, its dependency on $D_{variation}$ becomes quadratic as $D_{variation}$ becomes large.

3.3.3 Dual-Rail Self-Timed Approach

Another popular asynchronous self-timed scheme, dual-rail self-timed approach, uses dual-rail encoding to represent the data, and four-phase handshaking protocol to ensure a reliable operation [Sparsø01]. Since a dual-rail self-timed design responses immediately after a computation, its performance can closely follow the actual circuit delay statistics, as shown in Fig. 3.1. As a result, a dual-rail self-timed design would not have delay overhead to compensate the timing variability of critical path. Despite its better statistical performance, a dual-rail self-timed design must employ additional circuitry to implement dual-rail handshaking protocol and completion detection. This extra delay overhead from protocol circuit may not be trivial, and must be considered in final performance evaluation and optimization. To account for this protocol overhead, the delay of the protocol circuitry is estimated and added to the final critical path delay. Note that a bundled-data self-timed design would also require additional protocol circuitry, but the corresponding protocol overhead is much smaller than a dual-rail design since it can be utilized as part of matched delay line without slowing down system speed.

In the next section, the statistical performance models developed in this section, combined with the variability model of CMOS circuit developed in section 3.2, is employed to evaluate energy and delay performances of CMOS digital circuits with different timing methodologies.

3.4 Performance Comparison

In the previous section, a statistical model to analyze the performances of synchronous and asynchronous circuits under process variations is presented. Together with the delay variability model of CMOS circuit developed in section 3.2, the impacts of process variations on synchronous and asynchronous CMOS digital circuits across a wide range of supply voltage and logic depth can be evaluated. In this section, the delay and energy performances of different timing schemes are first evaluated. The proposed statistical analysis framework is then used to determine the optimal timing strategy, pipeline depth and supply voltage under process variations for a low-energy application example.

3.4.1 Speed Performance

Table 3.1 summaries the delay overheads required to implement different timing schemes. $P_{bundled-data}$ and $P_{dual-rail}$ represent the normalized delay overheads of protocol circuits associated with bundled-data and dual-rail self-timed designs, respectively. The total delay here represents the time required to correctly evaluate a logic function in the presence of process variations. The latency and cycle time performances, which must be evaluated with respect to the actual implementation of handshaking control protocol, are not discussed here.

Eqs. 3.5 to 3.8, Eq. 3.11 and Eq. 3.16 are first used to estimate D_{sync} and $D_{bundled-data}$ associated with CMOS digital circuits under process variations. As discussed in section

Table 3.1: Speed Performance Analysis Model

Timing Scheme	Delay Overhead (D)	Protocol Overhead (P)	Total Delay
Synchronous	D_{sync}	-	$t_{logic}\,(1 + D_{sync})$
Bundled-Data	$D_{bundled-data}$	$P_{bundled-data}$	$t_{logic}\,(1 + D_{bundled-data} + P_{bundled-data})$
Dual-Rail	-	$P_{dual-rail}$	$t_{logic}\,(1 + P_{dual-rail})$

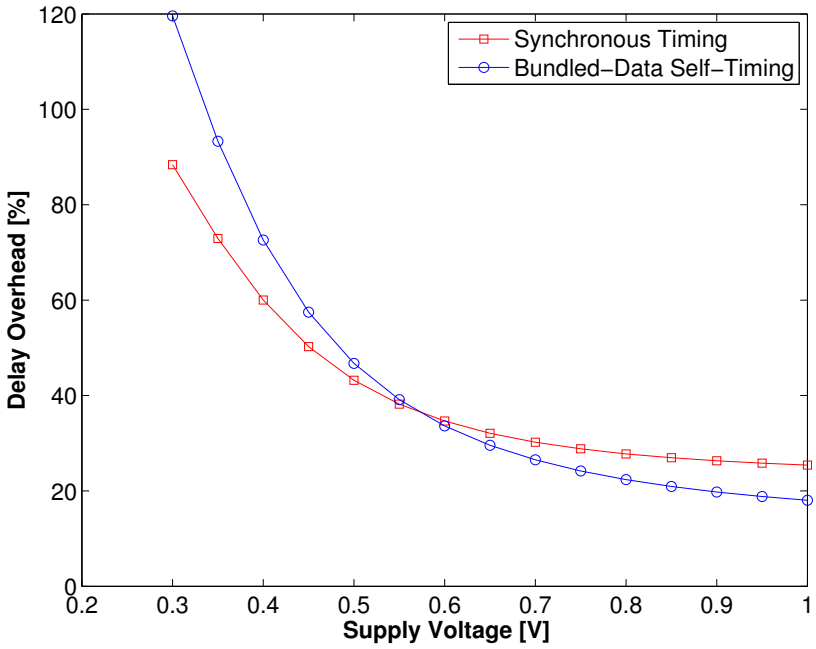

Figure 3.7: Delay overhead characteristics of synchronous and bundled-data schemes for a critical path delay of 4FO4.

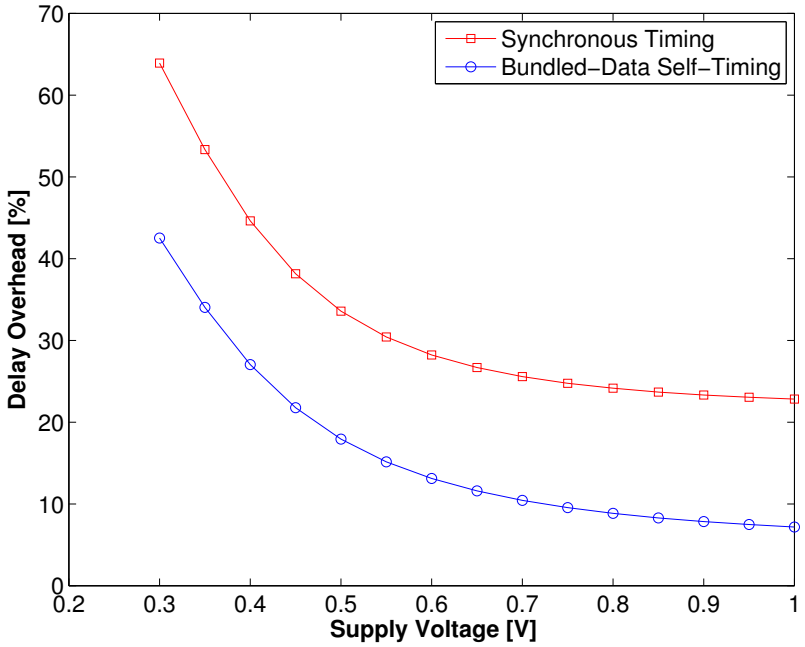

Figure 3.8: Delay overhead characteristics of synchronous and bundled-data schemes for a critical path delay of 24FO4.

3.2, DTD variations is assumed to affect only synchronous approach, while WID variations influence both synchronous and bundled-data designs. Fig. 3.7 shows the delay overhead results of synchronous and bundled-data schemes for a critical path delay of 4FO4 as the supply voltage varies from 300mV to 1V. As the supply voltage is lowered, $D_{bundled-data}$ becomes larger and eventually greater than D_{sync}. This is because WID variations become larger and dominate at low supply voltages as shown in Fig. 3.4. This causes $D_{bundled-data}$ to increase rapidly as predicted by Eq. 3.16. Therefore, a bundled-data design operating below 600mV would suffer more delay overheads than a synchronous approach. Fig. 3.8 shows the delay overheads required for a larger critical path delay of 24FO4. Since a longer critical path with more gate stages can greatly reduce the impact of WID variations, a bundled-data design now has less delay overheads than a synchronous approach across the entire supply range.

To compare the total delay required to perform synchronous and asynchronous computations, both delay overhead D and protocol overhead P are considered as summarized in table 3.1. In actual circuit implementations, a bundled-data design can utilize its protocol overhead as part of matched delay line to improve the speed. As a result, $P_{bundled-data}$ is set to 1FO4 first assuming that the overall protocol delay can be less than one-stage delay after optimization. A dual-rail design, on the other hand, requires additional completion detection circuitry and cannot utilize it as part of matched delay line. $P_{dual-rail}$ is set to 2FO4 to account for this extra delay of completion detection since the completion detector is usually

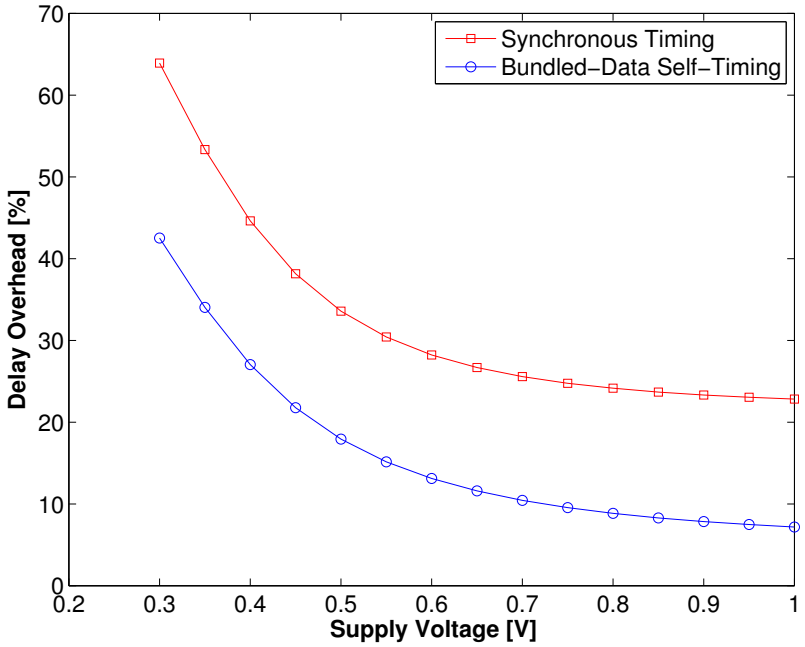

Figure 3.9: Speed performances of bundled-data and dual-rail schemes for a critical path delay of 4FO4. The delay is normalized to the synchronous design.

a single-stage NOR/NAND gate. Note that the protocol overhead here is independent of the number of logic stages, and is simply estimated to the first order to demonstrate the design tradeoff and its impact on the overall speed performance. The actual protocol overhead of an asynchronous scheme can be determined after actual circuit implementation.

Fig. 3.9 shows the estimated speed performances of bundled-data and dual-rail asynchronous circuits for a critical path delay of 4FO4 as the supply voltage varies from 300mV to 1V. The performances of both asynchronous approaches are normalized to the synchronous design. For this specific design example, a synchronous design achieves better speed performance at high supply voltages, while a dual-rail approach performs better below 450mV where circuit delay variability becomes larger. Fig. 3.10 shows another example of larger critical path delay of 24FO4. Both asynchronous designs now have better speed performances than synchronous approach. This is because for circuits with larger critical path delay, protocol overhead becomes relatively smaller than delay overhead required to guarantee reliability under process variations. In summary, for a circuit topology with fixed critical path delay, a synchronous timing scheme is a better choice for high-speed applications with smaller critical path delay operating at higher supply voltages, while a dual-rail self-timed scheme performs better when the protocol overhead is relatively small compared to the circuit critical path delay.

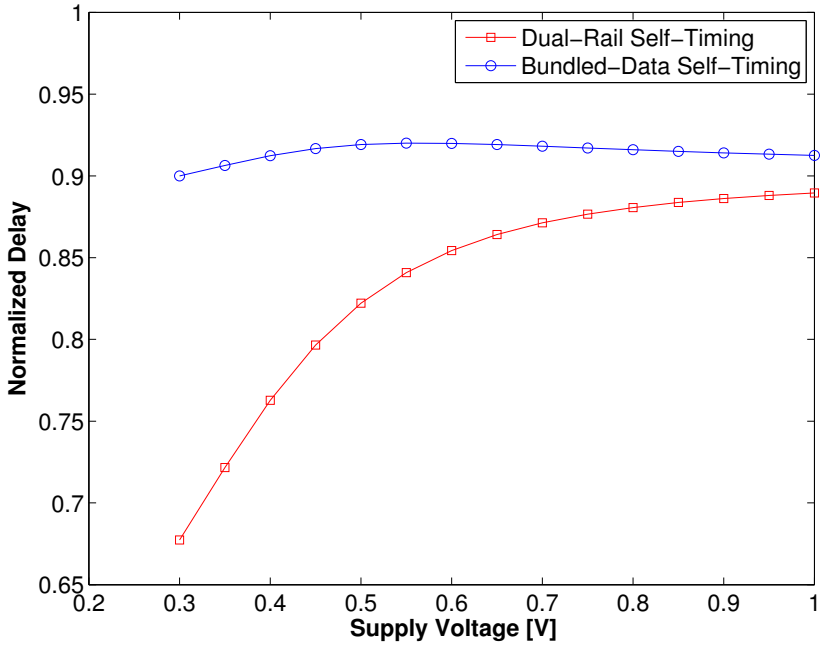

Figure 3.10: Speed performances of bundled-data and dual-rail schemes for a critical path delay of 24FO4. The delay is normalized to the synchronous design.

3.4.2 Energy Performance

The energy per operation of a CMOS gate E_{total} contains both active and leakage components, as shown in Eq. 2.8. An asynchronous design has a larger C_{switch} and I_{leak} than a synchronous design due to additional protocol circuit overheads, but may have small T_d under process variations as discussed in the previous section. Assuming that the extra leakage and switched capacitance from implementing an asynchronous self-timed scheme is also proportional to the protocol overhead P in table 3.1, the computation energy of a datapath under process variations can be estimated as

$$E_{total} = \alpha C_{switch} (1 + P) V_{DD}^2 + V_{DD} I_{leak} (1 + P) T_d (1 + P + D) \tag{3.19}$$

Note that Eq. 3.19 evaluates only the average computation energy of a datapath in the presence of variability. The corresponding communication energy with different timing schemes, such as clocking network in synchronous design and delay line in bundled-data design, is not included in Eq. 3.19. Fig. 3.11 and Fig. 3.12 show the estimated energy consumption of a 24FO4 inverter chain using different timing schemes as a function of supply voltage for $\alpha = 0.1$ and $\alpha = 0.01$, respectively. The simulation results show that if the circuit activity is high and active energy dominates, a synchronous approach generally consumes less computation energy. However, if a circuit has a low activity factor operating at low supply

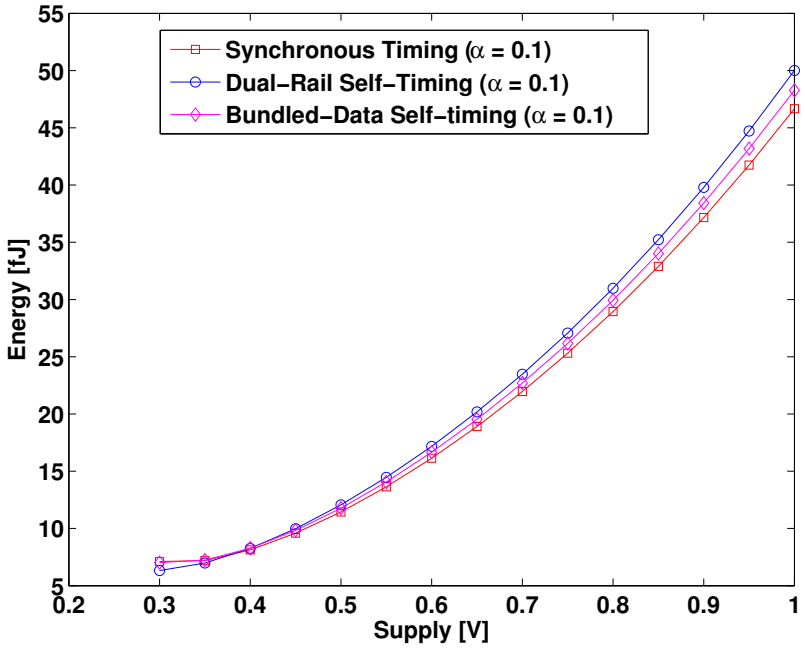

Figure 3.11: Energy performances of a 24FO4 inverter chain with different timing schemes for $\alpha = 0.1$.

voltages, leakage energy becomes the dominant component of total energy consumption. An asynchronous design now demonstrates better energy performances especially at low supply voltages, as shown in Fig. 3.12. This is because both bundled-data and dual-rail designs have smaller idle time than a synchronous design in the presence of variability, and therefore consume less leakage energy and achieve better overall energy performance.

3.5 Summary

Asynchronous self-timed schemes allow for an adaptive adjustment to delay variations and support for an inherent leakage minimization for both static and dynamic variations, indicating a robust and energy-efficient alternative for ultra-low voltage operation. A statistical analysis framework is presented to evaluate energy and delay performances of CMOS digital circuits with different timing methodologies in the presence of variability. The proposed analysis framework consists of two parts: a delay variability model of CMOS circuit and a statistical performance model of synchronous and asynchronous timing schemes. The variability model accurately estimates the impact of process variations on speed performance of CMOS circuits operating in both the strong-inversion and subthreshold regions with different logic depths. The statistical performance model efficiently predicts the delay overheads required to implement synchronous, bundled-data and dual-rail self-timed schemes under

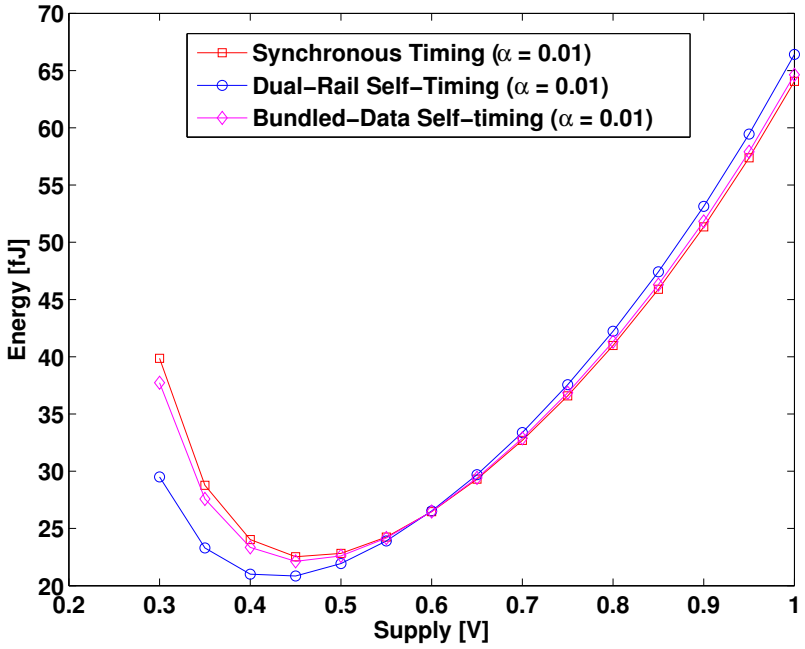

Figure 3.12: Energy performances of a 24FO4 inverter chain with different timing schemes for $\alpha = 0.01$.

process variations. By using the proposed statistical analysis framework, designers can efficiently evaluate energy and delay performances of CMOS digital circuit under process variations, and perform system optimization considering a variety of timing methodologies and design parameters such as logic depth and supply voltage, instead of performing extensive Monte Carlo SPICE simulations.

Based on the proposed statistical models and inverter chain simulation results in 65nm CMOS technology, the analysis shows that in the presence of variability, a synchronous design generally demonstrates better delay performance for high-speed applications and consumes lower computation energy for circuits with high activity. On the other hand, an asynchronous design exhibits better energy and delay characteristics for circuits with low activity and larger critical path delay. Note that since the delay variability of CMOS transistor actually follows the log-normal distribution in sub-threshold region, an asynchronous design is expected to perform even better at low supply voltages. For future technology nodes with smaller transistor dimensions, both process variations and their impact on circuit performance would become even significant. As a result, a timing methodology that is capable of achieving better statistical performance in the presence of variability, such as asynchronous dual-rail self-timed scheme, would be a promising approach for future CMOS digital circuit design.

In the next two sections, two implementation examples using asynchronous self-timed circuits for ultra-low energy applications are presented. Self-timed circuit designs using sense

amplifier-based pass transistor logic are first introduced. After that, a self-timed neural signal processor for brain-machine interfaces is presented.

Chapter 4

Case Study I: Asynchronous SAPTL Design

Lowering the supply voltage effectively reduces dynamic energy consumption but is accompanied by a dramatic increase in leakage energy due to the lower device threshold voltage needed to maintain performance. As a result, for low-energy applications, the leakage energy that the system can tolerate ultimately sets the minimum total energy point and limits the minimum device threshold voltage. Speed, therefore, benefits little from technology scaling. The sense amplifier-based pass transistor logic (SAPTL) [Alarcón07] is a novel circuit topology that breaks this tradeoff to realize very low energy logic without sacrificing speed. SAPTL modules were initially designed to operate synchronously. In this chapter, asynchronous operation is introduced in SAPTL to further improve energy-delay performance.

This chapter presents two different approaches to realize self-timed SAPTL: the bundled-data and the dual-rail handshaking protocol. Section 4.1 introduces the basic operation and circuit architecture of SAPTL. Section 4.2 defines the handshaking protocol between two SAPTL modules to realize self-timed operation. A self-timed SAPTL architecture with bundled-data protocol is presented including the circuit architecture and a detailed performance analysis. Section 4.3 introduces a glitch-free adaptation to the handshaking protocol between self-timed SAPTL modules that can further improve robustness and performance. Section 4.4 presents the design and implementation of self-timed SAPTL with a dual-rail protocol. Section 4.5 shows the test chip implementation of self-timed SAPTL in 90nm CMOS technology. Section 4.6 shows the simulated and measured energy, delay, and leakage of various self-timed SAPTL circuits with different handshake protocols. Section 4.7 concludes this chapter.

4.1 SAPTL Architecture

Fig. 4.1 shows the basic architecture of a SAPTL circuit, which is composed of a stack, a driver, and a sense amplifier [Alarcón07].

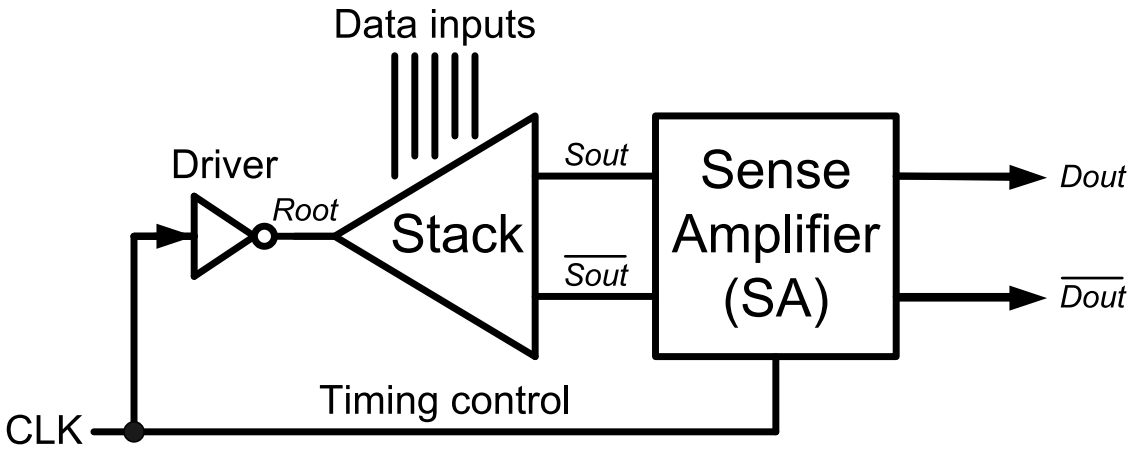

Figure 4.1: Architecture of SAPTL module with synchronous timing control.

4.1.1 Stack and Driver

The stack consists of an NMOS-only pass transistor tree with full-swing inputs and low-swing pseudo-differential outputs to perform the required logic function, as shown in Fig. 4.2. The stack can implement a given Boolean expression by connecting the minterm branches of the tree to one output, and the maxterm branches to the other as illustrated by the programming switches in the diagram. In this implementation, the logic function of a SAPTL stack is determined and permanently fixed at fabrication by replacing the programming switches with hardwired connections. Because the stack has no supply rail connections, it does not contribute sub-threshold leakage current, and it also has no gain.

Because the stack has no supply rail connections, a driver, which is a simple inverter in this example, is placed at the root input of the stack to inject the evaluation current. In operation, either *Sout* or $\overline{Sout}$, but not both, is charged toward the supply rail when the driver energizes the selected path through the stack. After each computation and before every evaluation, both differential outputs are reset to ground (logical "0") to initialize the stack to a known state. This initialization is done by turning on all the transistors in the stack and draining the charges out through the root of the stack when the driver output is zero. The alternate charging and resetting of *Sout* or $\overline{Sout}$ realizes a standard dual-rail encoding [Sparsø01].

The speed of the SAPTL module depends strongly on the depth of the stack, N_{stack}, which is defined as the number of transistors in series from the root node to the differential outputs. Because the stack contributes no sub-threshold leakage current, the stack transistors can have a very low threshold voltage and still operate in the super-threshold region even with a very low supply voltage. Therefore, SAPTL is a promising candidate to realize ultra-low energy computation without soliciting sub-threshold operation [Alarcón07].

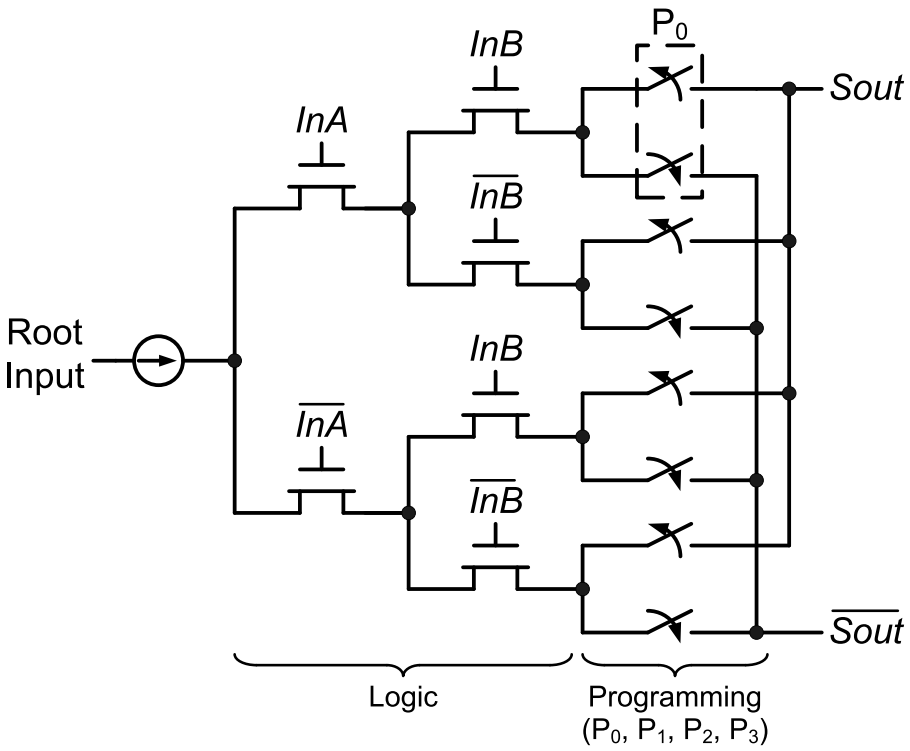

Figure 4.2: Schematic of a two-input stack with $N_{stack} = 5$.

4.1.2 Sense Amplifier

The sense amplifier shown in Fig. 4.3 serves two purposes. First, it amplifies the low voltage stack output, restoring the signal to full voltage. Second, it serves as a buffer stage to store the output of the stack, so as to improve overall speed. The sense amplifier consists of two stages; the first stage acts as a pre-amplifier to reduce the impact of mismatch in the actual technology environment, and the second stage acts as a cross-coupled latch that retains the processed data even after the stack is reset. The sense amplifier is designed to detect input voltages that are less than $(V_{DD} - V_{th})$, thus reducing the performance degradation due to the low stack voltage swings and the absence of gain in the pass transistor network. If the driver can be turned off as soon as the sense amplifier makes a decision, the stack voltage swings could be kept to a minimum, reducing the energy required to perform the desired logical operation. The leakage of the sense amplifier accounts for most of the leakage energy of the SAPTL module. It can be directly traded off against the input sensitivity of the sense amplifier to width, length, and threshold voltage mismatch.

The complete analysis and design of SAPTL and comparison with other logic styles can be found in [Alarcón10].

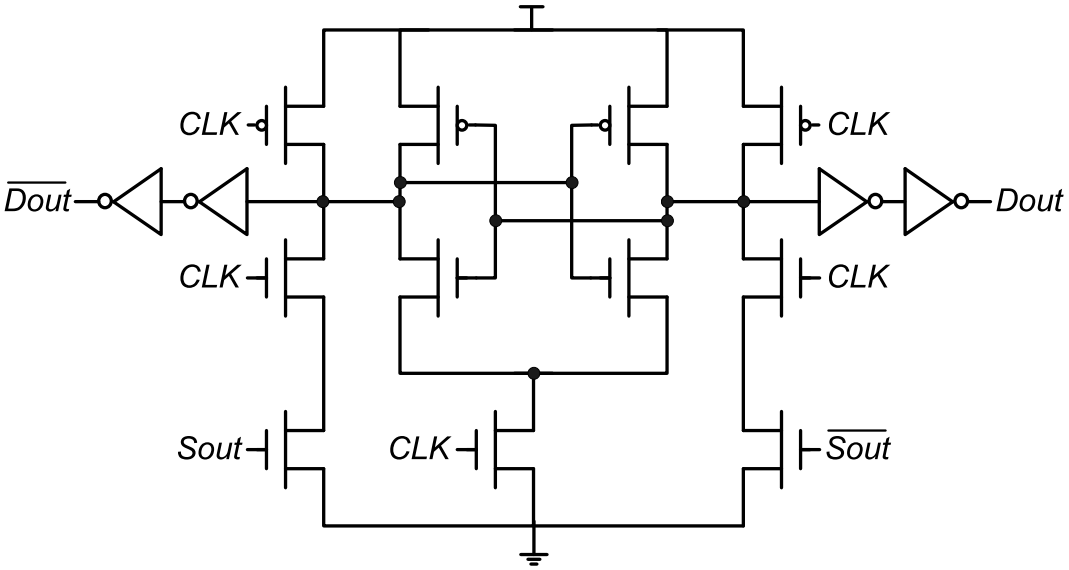

Figure 4.3: Sense amplifier circuit.

4.1.3 Synchronous Timing

In the synchronous SAPTL design in Fig. 4.1, the global clock signal CLK controls the timing of the evaluation-reset operation of the SAPTL module. In a synchronous pipeline, alternate SAPTL stages use different clock phases; while one evaluates, the next one resets the stack. This is similar to the operation of latch-based pipelines in synchronous static CMOS design [Rabaey03] and requires two-phase non-overlapping clock signals [Alarcón07].

In the following sections, the design and implementation of self-timed SAPTL modules and pipelines are presented.

4.2 Bundled-Data Self-Timed SAPTL Design

The communication between two self-timed SAPTL modules based on a request–acknowledge handshaking protocol is shown in Fig. 4.4. The operation of the self-timed SAPTL involves two parts: 1) data evaluation and 2) data reset, as shown in Fig. 4.5. The handshaking protocol of the self-timed SAPTL can be thought of as similar to the four-phase protocol in [Sparsø01]. Note that, during the reset cycle, both data inputs must be reset to a logical 1 level rather than the commonly used logical 0 [Sparsø01]. New valid data are present when either one, but not both, of the differential data input signals falls to the logical 0. Various relative timing assumptions (RTAs) [Stevens03] are presented to ensure that sufficient voltage levels are present at the stack outputs in order to guarantee correct SAPTL operation.

The circuit implementation of the self-timed SAPTL module using the bundled-data protocol is shown in Fig. 4.6. The main data path, composed of a driver and stack, evaluates data

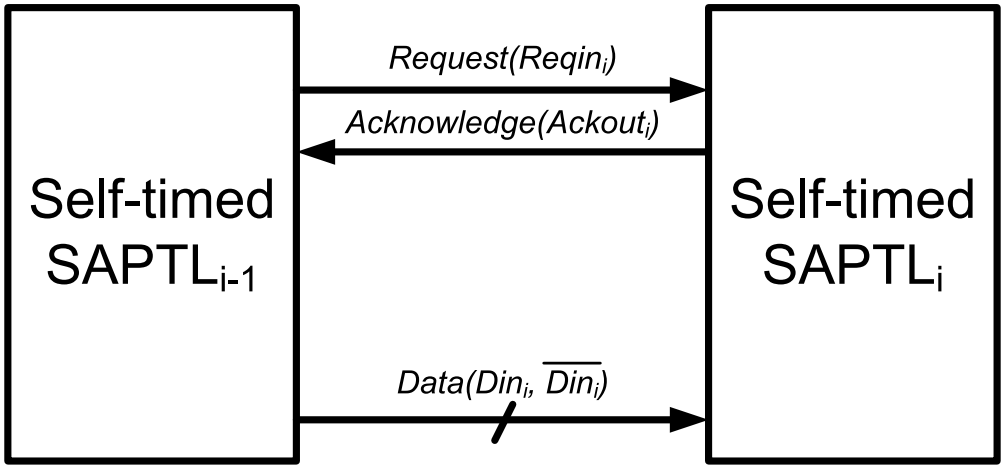

Figure 4.4: Communication between two self-timed SAPTL modules.

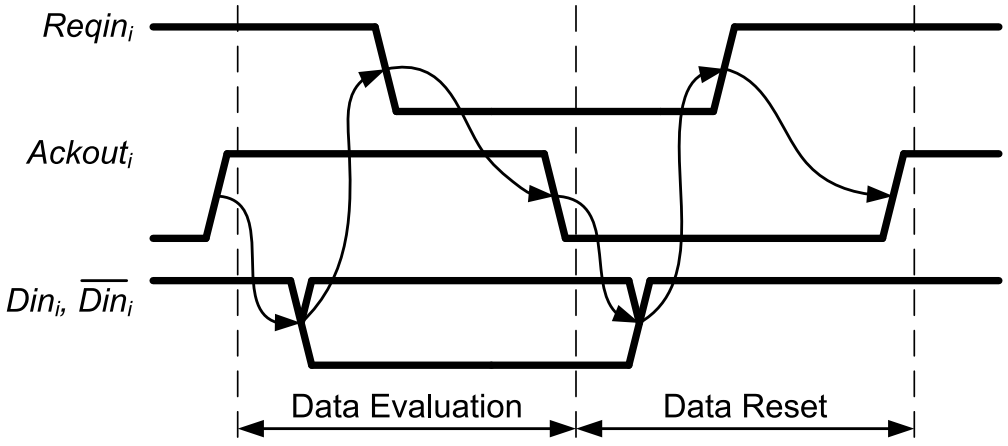

Figure 4.5: Two-cycle evaluation-reset operation for the self-timed SAPTL stage i in Fig. 4.4.

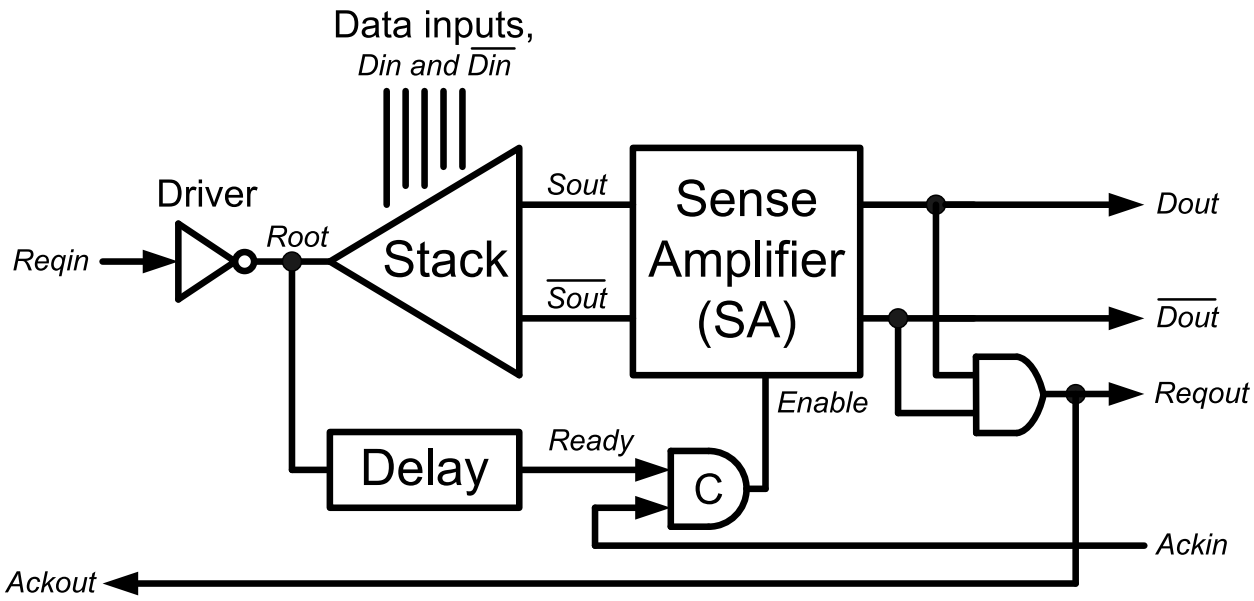

Figure 4.6: Architecture of self-timed SAPTL module with bundled-data protocol.

or resets after receiving the request signal *Reqin* and data input signals *Din* and $\overline{Din}$ from the previous SAPTL stage. The control path, which consists of a delay line and a C-element, produces the local clock signal *Enable* to trigger the sense amplifier. The delay line mimics the delay of the stack to generate the control signal *Ready* indicating that the stack has finished an operation. The C-element then produces *Enable* by collecting *Ready* and the acknowledge signal *Ackin* from the next SAPTL stage. In multiple fan-in and fan-out situations, additional C-element can be employed to re-converge multiple request and acknowledge events from the different fan-in and fan-out stages. When triggered by *Enable*, the sense amplifier latches the stack output data or resets depending on the logical state of *Enable*. The full-swing data output signals *Dout* and $\overline{Dout}$ are made available at the outputs of the sense amplifier. The AND gate serves as a completion detection circuit, generating the handshake signals *Ackout* and *Reqout* that indicate the completion of the current operation.

The relationship between the input and output signals of the i^{th} SAPTL stage can be summarized as

$$\begin{aligned} Dout_i, \overline{Dout_i} &= f\left(Enable_i, Sout_i, \overline{Sout_i}\right) \\ &= ff\left(Reqin_i, Ackin_i, Din_i, \overline{Din_i}\right) \end{aligned} \tag{4.1}$$

$$\begin{aligned} Ackout_i &= Reqout_i \\ &= g\left(Dout_i, \overline{Dout_i}\right) \end{aligned} \tag{4.2}$$

$$Enable_i = h\left(Reqin_i, Ackin_i\right) \tag{4.3}$$

where for the subsequent $(i+1)^{th}$ SAPTL stage

$$\begin{aligned} Din_{i+1} &= Dout_i \\ \overline{Din_{i+1}} &= \overline{Dout_i} \\ Reqin_{i+1} &= Reqout_i \\ Ackout_{i+1} &= Ackin_i \end{aligned}$$

4.2.1 Data Evaluation Cycle

When a self-timed SAPTL stage finishes a data reset cycle, it raises the acknowledge signal *Ackout*, (*Ackout* $\uparrow$), and waits for *Din*, $\overline{Din}$, and a falling *Reqin*, (*Reqin* $\downarrow$), before performing a data evaluation cycle. The delay between *Ackout* $\uparrow$ and *Reqin* $\downarrow$ is the reverse latency for the data reset cycle $T_{Lr,reset}$ [Williams94], and given by

$$T_{Lr,reset} = T_{C\text{-}element} + T_{SA\_data} + T_{AND} \tag{4.4}$$

Before the driver applies the evaluation current, which is controlled by *Reqin* $\downarrow$, the data input signals *Din* and $\overline{Din}$ must be set to correct logic levels for proper operation. Therefore, the first RTA of the SAPTL in the data evaluation cycle can be expressed as

$$Din \downarrow < Ackin \downarrow \tag{RTA 4.1}$$

where $DIN \downarrow$ indicates that a voltage difference has been developed between the two data input signals *Din* and $\overline{Din}$, which means that one of the two data input signals has been pulled down to logical 0, while the other stays at logical 1. In most cases, RTA 4.1 can easily be satisfied if the fan-in of the SAPTL is small and the wire delay is insignificant compared with the gate delay. However, if the SAPTL has a large fan-in or the wiring capacitance is significant, RTA 4.1 must be carefully verified upon completion of the layout design.

After the stack starts to perform data evaluation, either *Sout* or $\overline{Sout}$ will be pulled toward the voltage level $(V_{DD} - V_{th})$. The sense amplifier must be triggered after the outputs of the stack have developed a "sufficiently large" voltage difference. This voltage difference is set during the sense amplifier design time, and in the case of the SAPTL, it is set to 100mV, balancing the tradeoff between sense amplifier leakage and delay. For a valid bundled data protocol, the following RTA between data path and control path, therefore, must be satisfied to ensure correct operation of the sense amplifier

$$SOUT \uparrow < Enable \uparrow \quad \text{(RTA 4.2)}$$

or equivalently,

$$T_{Stack,data} < T_{Delay\text{-}line,data} + T_{C\text{-}element} \quad \text{(RTA 4.3)}$$

where $SOUT \uparrow$ represents the development of a sufficient voltage difference between the two stack output signals $Sout$ and $\overline{Sout}$, meaning that one of the output signals has been charged high enough, while the other stays at logical 0. Unlike RTA 4.1, which is easily satisfied, it is hard to guarantee RTA 4.2 in actual implementation. Process, voltage, and temperature variations make delay matching difficult between the stack data path and a separate controlling path. Moreover, the delay characteristic of the stack also depends on the input signal statistics, while the delay line has a fixed delay. As a result, the design of the delay line not only must track the "worst-case delay" of the stack but also requires extra margin to compensate for possible process, voltage and temperature variations. Because of the delay line, the speed performance of a single SAPTL stage is fixed and limited to the slowest possible case even if the computation finishes early. Failure to meet RTA 4.2 would result in either an incorrect sense amplifier decision, or an increase in sense amplifier delay due to metastability.

The forward latency of the data evaluation cycle $T_{Lf,data}$ [Williams94] can thus be determined as the delay between the incoming SAPTL request going down $Reqin \downarrow$ and the outgoing request and acknowledge going down $Reqout \downarrow$ and $Ackout \downarrow$, i.e.,

$$T_{Lf,data} = T_{Driver} + T_{Delay\text{-}line,data} + T_{C\text{-}element} + T_{SA,data} + T_{AND} \quad (4.5)$$

4.2.2 Data Reset Cycle

In the data reset cycle, similar timing assumptions can be derived, as well as forward and reverse latencies for the self-timed SAPTL module, giving

$$DIN \uparrow < Reqin \uparrow \quad \text{(RTA 4.4)}$$

$$SOUT \downarrow < Enable \downarrow \quad \text{(RTA 4.5)}$$

$$T_{Lf,reset} = T_{Driver} + T_{Delay\text{-}line,reset} + T_{C\text{-}element} + T_{SA,reset} + T_{AND} \quad (4.6)$$

$$T_{Lr,data} = T_{C\text{-}element} + T_{SA,reset} + T_{AND} \quad (4.7)$$

where $DIN \uparrow$ indicates that both data input signals have been reset to logical 1 and $SOUT \downarrow$ means that both stack outputs have been reset to logical 0. In contrast to the micropipeline two-phase handshaking protocol [Sutherland89] that is able to exploit both phases for data transmission in order to achieve maximum throughput, the self-timed SAPTL architecture can transmit data only during alternate phases because the stack must be reset after every data evaluation cycle. Thus, one handshake phase in the self-timed protocol is always dedicated to data reset. As a result, the minimum cycle time T_P required to transmit valid data in a self-timed SAPTL stage is given by

$$T_P = T_{Lf,data} + T_{Lr,data} + T_{Lf,reset} + T_{Lr,reset} \tag{4.8}$$

4.2.3 Speed Enhancement

It is possible to improve the performance of self-timed SAPTL by signal restructuring. By carefully inspecting Eqs. 4.1 to 4.3, it can be observed that *Reqout* is actually a delayed version of *Enable*, although they are not logically equivalent. A more aggressive timing strategy is to use *Enable* as the acknowledge signal *Ackout* for the current SAPTL stage. Thus, this new SAPTL design can achieve better performance with the same functionality under certain RTAs. It is safe to generate the acknowledge signal from Enable rather than from the output of the AND gate, provided the conversion time of the sense amplifier $T_{SA,data}$ is smaller than the sum of the reverse latency $T_{Lr,data}$ and the time to reset the stack $T_{Stack,reset}$ in the next data reset cycle, which can be written as

$$T_{SA,data} < T_{Lr,data} + T_{Stack,reset} \tag{RTA 4.6}$$

As a result, the SAPTL module can begin the handshaking process for the data reset operation of its next cycle immediately after the stack completes data evaluation and concurrently with the sense amplifier latching the evaluated data. This scheme shortens the reverse latency of an SAPTL stage and thus yields higher throughput than the original handshaking scheme. The new reverse latencies $T_{Lr,data,new}$ and $T_{Lr,reset,new}$ and the new minimum cycle time $T_{P,new}$ can be given by

$$T_{Lr,data,new} = T_{Lr,data} - T_{SA,data} - T_{AND} \tag{4.9}$$

$$T_{Lr,reset,new} = T_{Lr,reset} - T_{SA,reset} - T_{AND} \tag{4.10}$$

$$T_{P,new} = T_P - T_{SA,data} - T_{SA,reset} - 2T_{AND} \tag{4.11}$$

The new minimum cycle time $T_{P,new}$ can easily be verified by observing the loop behavior starting at the output of the C-element and remembering that one has to do this loop twice: once for data evaluation, and once for reset.

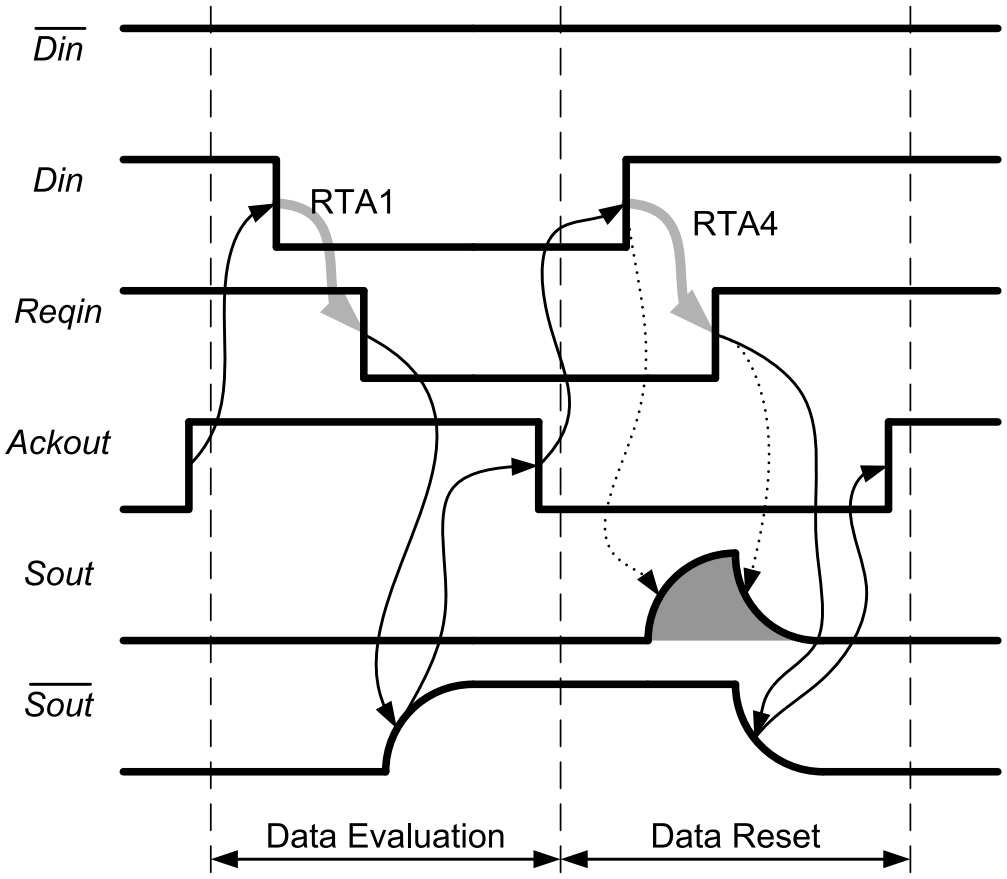

Figure 4.7: Timing diagram of self-timed SAPTL.

4.2.4 Glitch Problem

Fig. 4.7 shows the timing diagram of the self-timed and restructured SAPTL modules for two successive cycles. The solid arrows represent the control flow for normal operation, while the light-gray arrows represent the timing sequence governed by RTA 4.1 and RTA 4.4. The dotted arrows indicate events that may introduce unwanted glitches in the data reset cycle. Such a glitch can be harmless if its amplitude is less than the trigger threshold of the sense amplifier or, in other words, if the glitch takes a relatively long time to reach the sense amplifier trigger threshold voltage. The glitch will always happen in the interval between the two events $DIN \uparrow$ and $Reqin \uparrow$, and its duration is approximately equal to the sum of the driver delay T_{Driver} and AND gate T_{AND}. Therefore, the relative timing constraint needed to guarantee system functionality in the presence of a glitch can be expressed as

$$T_{Driver} + T_{AND} < T_{Stack,data} \qquad \text{(RTA 4.7)}$$

If the design of SAPTL fails to meet RTA 4.7, both outputs of the stack may simultaneously be in the logical 1 state, which violates the basic assumption of SAPTL operation and could

produce incorrect results. Typically, the sum of T_{Driver} and T_{AND} is much smaller than the time required by the stack to pull up its output nodes $T_{Stack,data}$. Therefore, RTA 4.7 is usually satisfied, and the glitch will not cause problems. However, the generation of a glitch increases the total energy dissipation of the self-timed SAPTL. The unwanted stray signal coupling induced by the glitch can also lead to erroneous operation. In the next section, the design and implementation of glitch-free handshaking protocol for the self-timed SAPTL to prevent this glitch problem is discussed.

4.3 Glitch-Free Handshaking Protocol

It is interesting to note that the glitch in Fig. 4.7 can occur only in a data reset cycle. The primary reason is the adoption of RTA 4.4. While RTA 4.4 is perfect for initializing the next data evaluation cycle in a micropipeline architecture, it is not appropriate for performing the data reset operation in the self-timed SAPTL architecture. In the data reset cycle, both the SAPTL data inputsDin and $\overline{Din}$, are charged to logical 1 as "reset signals" rather than as "data valid signals" for data evaluation. Consequently, if the $Reqin$ signal triggering the driver is still at logical 0, the stack will perform a false data evaluation and produce a glitch. Therefore, the revised relative timing constraint to avoid the glitch in the data reset cycle can be given by

$$Reqin \uparrow < DIN \uparrow \qquad \text{(RTA 4.8)}$$

4.3.1 Protocol Design

Fig. 4.8 shows two approaches to realize RTA 4.8, which are called the *early reset* and the *late reset* protocol, respectively. In the *early reset* protocol, another event $Reqin* \uparrow$ between the original $Ackout \downarrow$ and $DIN \uparrow$ events is introduced, as shown in Fig. 4.8(a). $Reqin*$ signal, instead of $Reqin$, can be used as a triggering signal to start the data reset operation earlier and avoid generating a glitch.

The other way to implement glitch-free operation, which is the *late reset* protocol, is shown in Fig. 4.8(a). The stack employs signals $Din*$ and $\overline{Din*}$, which are the replica delayed versions of Din and $\overline{Din}$, as reset input signals in the data reset cycle.

The only requirement for $Din*$ and $\overline{Din*}$ is that both signals be triggered later than $Reqin$. From Fig. 4.8, it can be observed that the timing slack needed to implement the early reset protocol is smaller than for the late reset protocol or, in other words, the implementation of the early reset protocol requires stricter RTAs. However, employing the early reset protocol will not affect the original latency of the data reset operation.

Therefore, the early reset protocol can achieve higher speed performance than the late reset protocol. Moreover, the early reset protocol can also minimize the leakage energy consumed

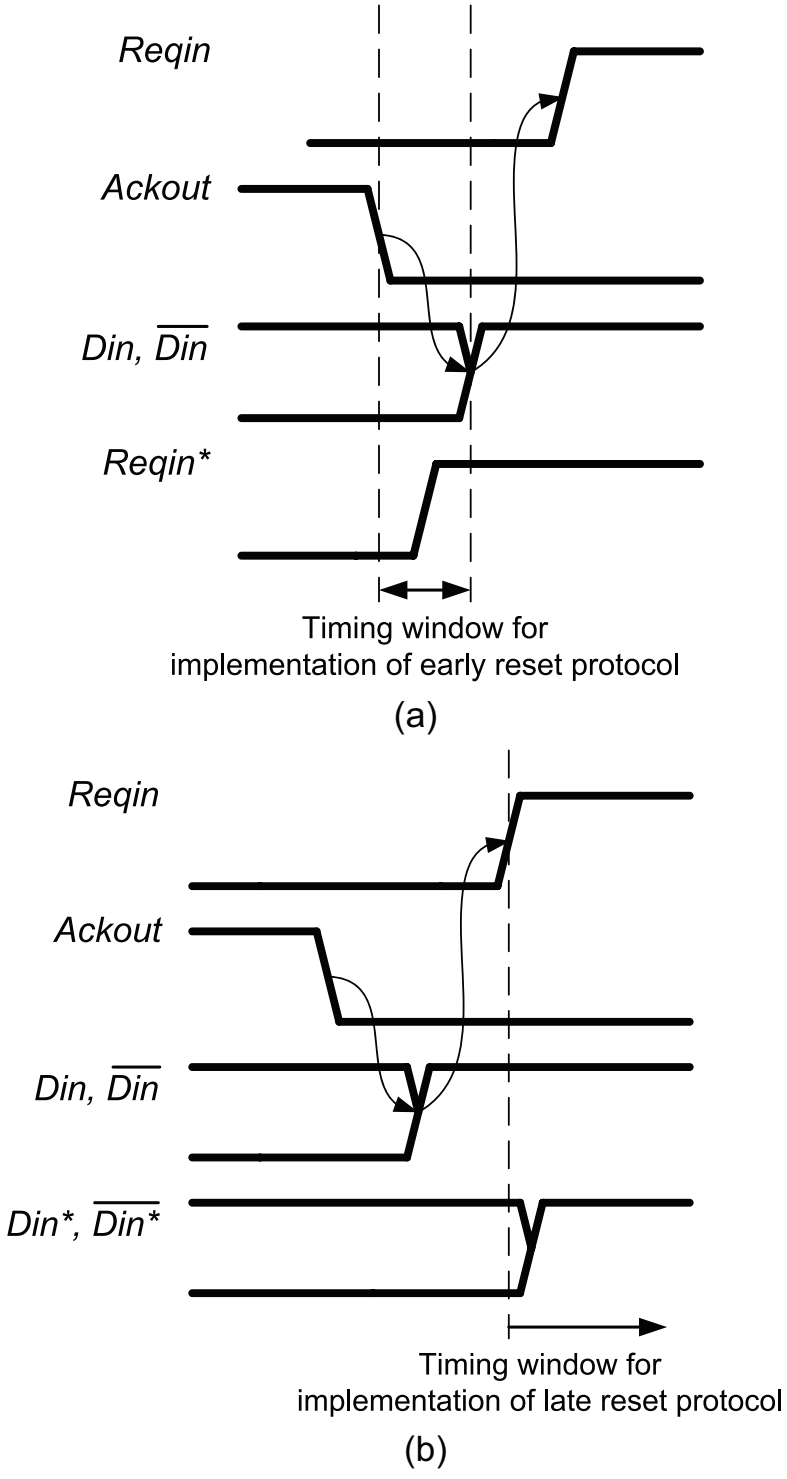

Figure 4.8: (a) Early reset and (b) late reset glitch-free handshaking protocol.

per handshaking operation of the SAPTL by keeping the root of the stack at logic 0 longer. This advantage of low-leakage operation makes the early reset protocol the preferred option.

4.3.2 Circuit Implementation

The circuit implementation of a self-timed SAPTL module with the early reset glitch-free handshaking protocol is shown in Fig. 4.9, and the corresponding timing diagram for two successive evaluation and reset cycles is shown in Fig. 4.10. The additional OR gate and extra input *Reqin* to the C-element do not change the functional behavior of self-timed SAPTL in the data evaluation cycle. In a data reset cycle, however, *Reqin*$*$ will be pulled up to a logical 1 and will start resetting the internal nodes of the stack immediately after the SAPTL module de-asserts the acknowledge signal *Ackout*. No glitch will be generated during the data reset cycle if *Reqin*$*$ is charged to a logical 1 and drives the stack root input to a logical 0 before *Din* and $\overline{Din}$ go high, which can be described by the following relative timing constraint for glitch-free operation

$$T_{Driver} + T_{OR} < T_{C\text{-}element} + T_{SA,reset} \qquad \text{(RTA 4.9)}$$

Because the OR gate and driver may be merged into one NOR gate, the total delay in the left-hand side of RTA 4.9 is really small and RTA 4.9 can thus easily be met. The extra input *Reqin* to the C-element is essential to maintain the reset state of the current SAPTL stage until the previous SAPTL raises *Reqin* for the next data evaluation cycle, thus the use of a three-input C-element. Note that the third C-element input signal *Reqin* is necessary only in the data reset cycle, but not in the data evaluation cycle. Therefore, an asymmetric C-element circuit can be used in this glitch-free self-timed SAPTL to minimize delay and energy consumption [Sparsø01].

By employing an additional NOR gate and a higher fan-in asymmetric C-element, the self-timed SAPTL architecture can avoid the glitch problem and consume lower energy, achieving more robust handshaking.

In addition, there are two main advantages to resetting the stack immediately after the SAPTL module sends the acknowledge signal. First, *Sout* and $\overline{Sout}$ will stay above logical 0 for only the short period required by the sense amplifier to latch the stack data. Once *Dout* or $\overline{Dout}$ has reached full swing, every internal node of the stack, as well as *Sout* and $\overline{Sout}$, will be reset to logical 0. Therefore, during the remainder of the cycle, the stack will stay in the data reset mode and consume minimum leakage energy.

Second, because the handshaking events with the previous SAPTL stage and the data reset events within the current SAPTL stack operate in parallel, the SAPTL will have a lower latency data reset cycle and thus achieve better performance. Thus, the delay line now becomes the major performance limiter in the self-timed SAPTL design using the bundled data protocol. In the next section, a self-timed SAPTL version with dual-rail protocol is introduced to address this issue.

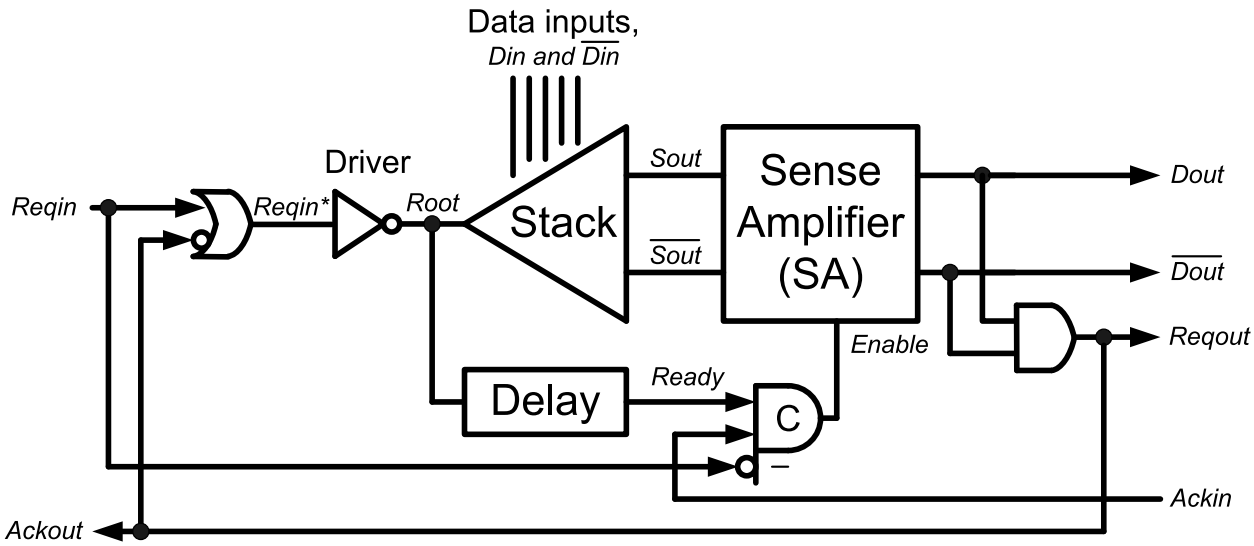

Figure 4.9: Self-timed SAPTL structure with early reset glitch-free protocol.

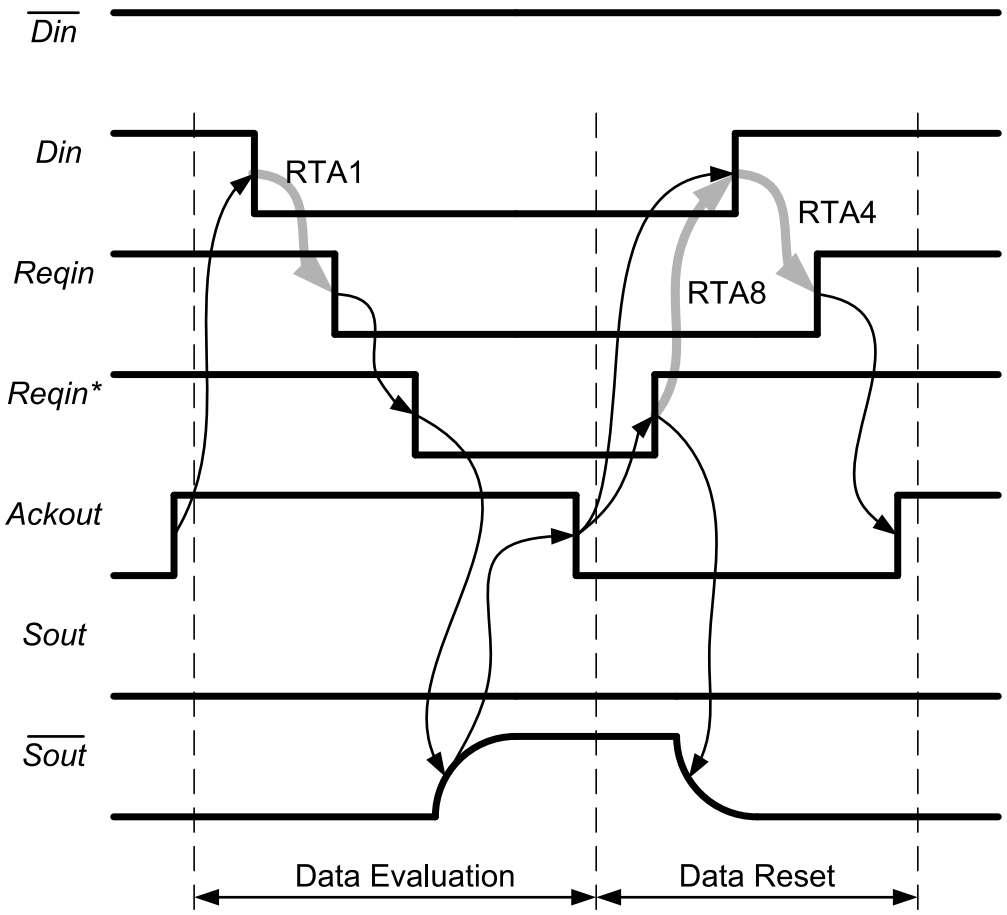

Figure 4.10: Timing diagram of glitch-free self-timed SAPTL.

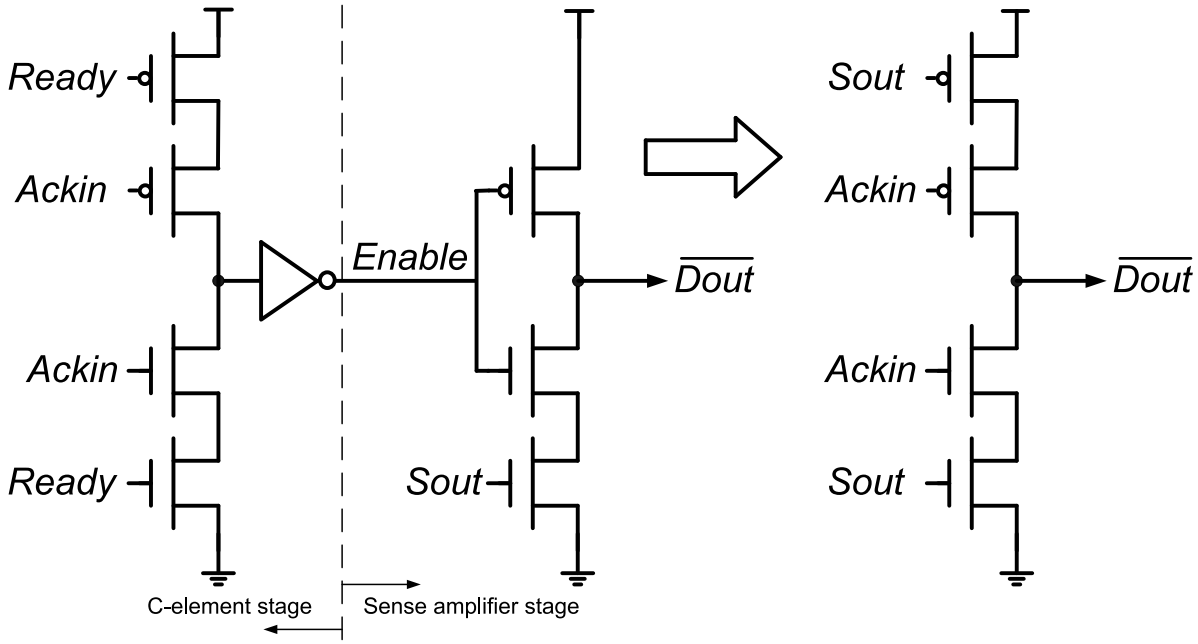

Figure 4.11: Logic combination of two-input C-element and sense amplifier circuits.

4.4 Dual-Rail Self-Timed SAPTL Design

In a self-timed SAPTL structure using the bundled-data protocol, RTA 4.2 is the most critical design constraint. In order to guarantee correct operation under process, voltage and temperature variations, the latency of the delay line can become very large and can severely limit the overall performance. Because the SAPTL uses dual-rail coding to represent data, the output signals of the stack *Sout* and $\overline{Sout}$, instead of *Ready* from the delay line, can be used to trigger the C-element. As a result, the delay line can be eliminated. Moreover, the C-element can respond immediately after the stack finishes operation, without being limited by RTA 4.2.

Furthermore, the sense amplifier and C-element circuits can be combined into a composite block through gate-level optimization, yielding a more energy-efficient architecture, as shown in Fig. 4.11. The optimized architecture with dual-rail protocol eliminates the traditional sense amplifier circuit and directly employs two C-element circuits as a complex gain stage at the outputs of the stack. The overall conversion speed, however, may be slower than the design with a sense amplifier due to the absence of a differential amplification and the loss of a positive-feedback mechanism between the two data paths. Fig. 4.12 shows the implementation of a glitch-free self-timed SAPTL architecture without the delay line.

The design and performance of the C-element circuits are particularly important in this architecture because the C-element not only plays the role of the gain stage but also serves as the handshaking element. The self-timed SAPTL with dual-rail protocol has latency and cycle time expressions similar to Eqs. 4.4 to 4.8. Note that the speed enhancement discussed

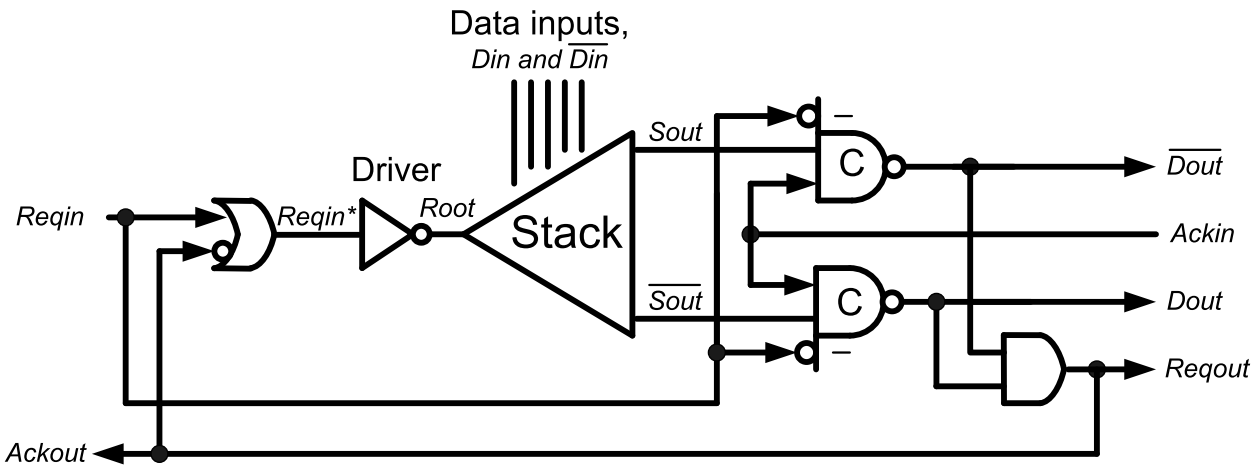

Figure 4.12: Architecture of glitch-free self-timed SAPTL module with dual-rail protocol.

in Section 4.2.3 does not apply to the dual-rail design in Fig. 4.12, because of the absence of the internal *Enable* signal. However, the single self-timed SAPTL stage is now elastic and able to achieve the best performance across process, voltage and temperature variations and different input characteristics. The self-timed SAPTL can thus exercise the full potential of asynchronous computation without the limitations of the delay line. It is interesting to note that the optimized self-timed SAPTL architecture in Fig. 4.12 has almost the same hardware complexity as the original synchronous SAPTL design in Fig. 4.1. This means that, with almost "zero cost," SAPTL is able to achieve both better performance and better robustness in the presence of variability by operating asynchronously.

The C-element design for the glitch-free handshaking protocol is shown in Fig. 4.13. In this circuit, two NMOS pass transistors, controlled by signals *Ackout* and *Reqin*, respectively, serve as the decision-making logic before the signal *Sout* can trigger the following two-input C-element. *Sout* can normally trigger the C-element except when *Ackout* and *Reqin* are both in the logical 0 state. This happens during a data reset cycle, as shown in Fig. 4.5, when the C-element is driven by the two back-to-back inverters in its feedback circuitry. Therefore, the state of the current SAPTL stage is maintained until the previous SAPTL stage raises the request signal *Reqin* for the next cycle.

4.5 Test Chip Implementation

The performance of the self-timed and synchronous SAPTL circuits are evaluated and compared using the Spectre circuit simulator. Monte Carlo simulations are also performed to ensure the correct operation of the SAPTL circuits at supply voltage down to 300mV even with 6σ process variations. Both bundled-data and dual-rail self-timed SAPTL designs were implemented in a 90nm CMOS test chip, as shown in Fig. 4.14. Fig. 4.15 shows the test circuit board. The implementation of synchronous SAPTL can be found in [Alarcón10].

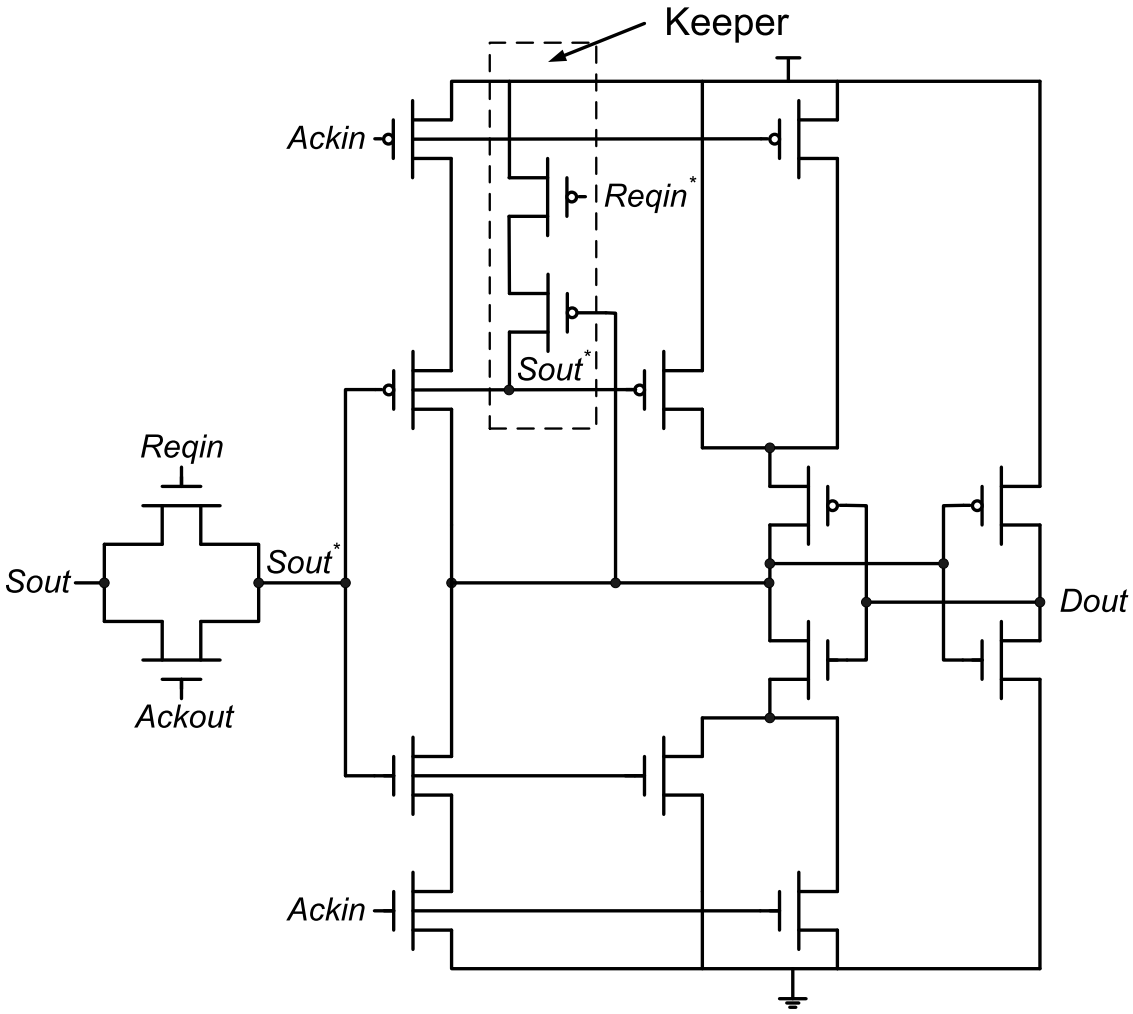

Figure 4.13: Two-input C-element circuit with additional decision-making logic for glitch-free self-timed SAPTL.

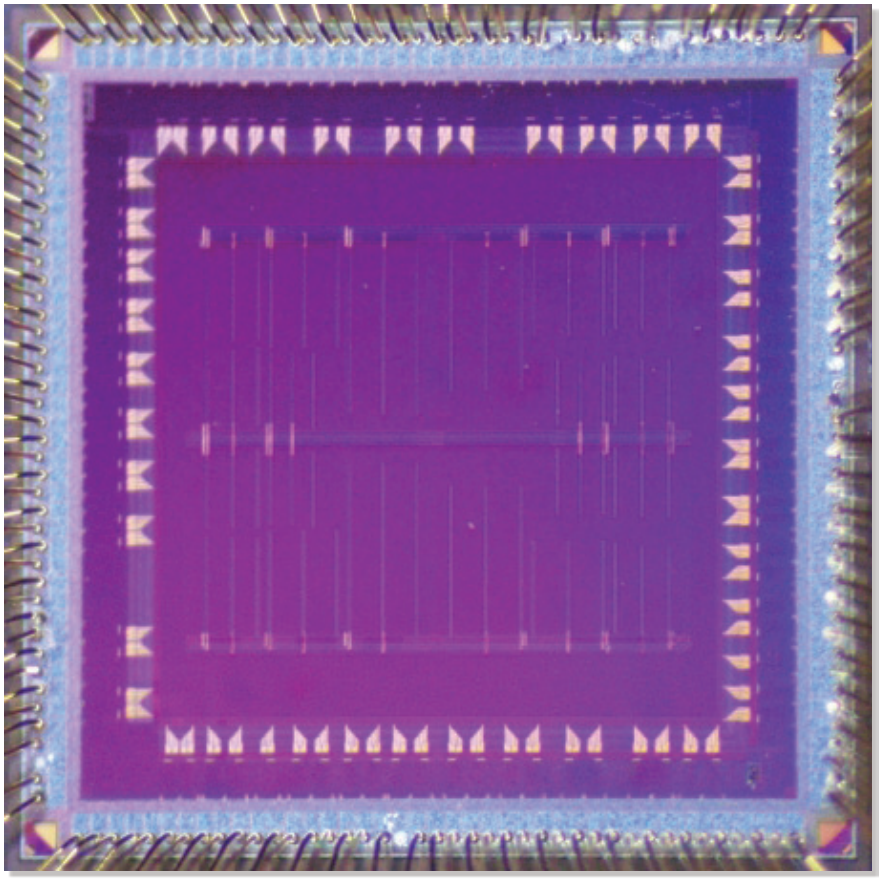

Figure 4.14: The self-timed SAPTL test chip.

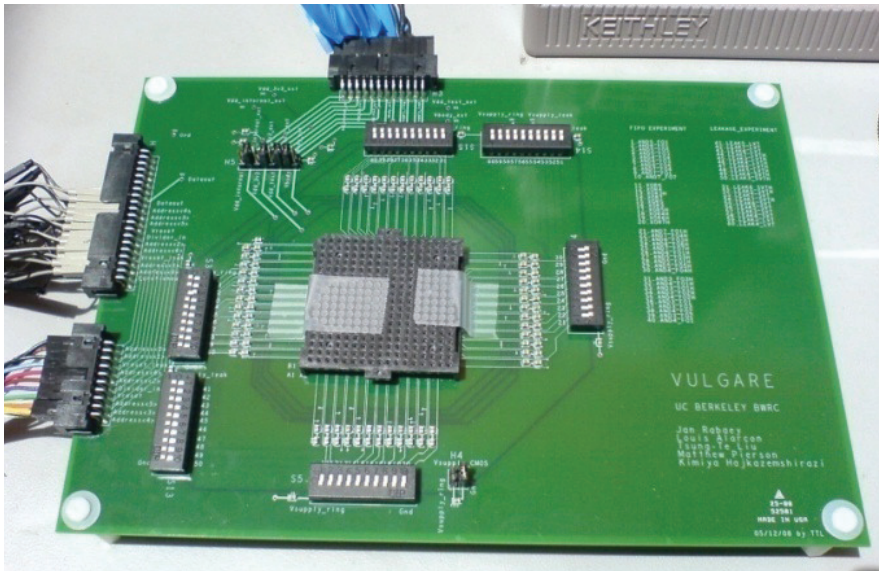

Figure 4.15: The self-timed SAPTL test board.

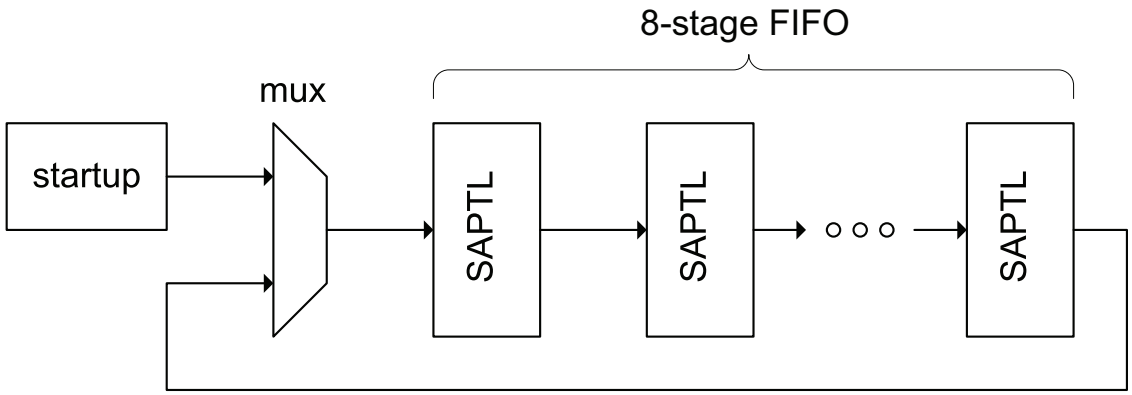

Figure 4.16: Test setup for energy and delay measurements.

The self-timed SAPTL test chip contains separate test structures with different fanin, fanout and logic topology for characterizing energy, delay and leakage currents. Fig. 4.16 shows the eight-stage first-in–first-out (FIFO) ring structure used to characterize the total energy and delay of self-timed SAPTL circuits with $N_{stack} = 5$ (SAPTL5). The oscillation frequency and supply current of self-timed SAPTL FIFO ring were then measured as the supply voltage is varied from 1V down to 300mV. Additional 100 replicas of each self-timed SAPTL circuit are also placed in the same test chip to measure the leakage current.

4.6 Results

This section presents the energy-delay and leakage comparisons of synchronous versus self-timed SAPTL. The simulation results exclude the parasitic contributions from the interconnect wires and the clock network. The effect of global parameters, such as clocks and long interconnect wires, should be done at the system level, in the context of an actual application. Comparisons between the synchronous SAPTL and other logic styles can be found in [Alarcón10].

4.6.1 Area Comparisons

The relative sizes and layout details of the synchronous, bundled-data and dual-rail SAPTLs are shown in Fig. 4.17. The stack occupies approximately half the SAPTL5 area, with the bundled-data SAPTL5 in Fig. 4.17(b) taking up 47% more area than the synchronous SAPTL5 due to the added delay line and handshaking circuitry. The removal of the delay line and the slightly larger sense amplifier found in the dual-rail SAPTL5 in Fig. 4.17(c) results in 30% more area when compared to the synchronous SAPTL5.

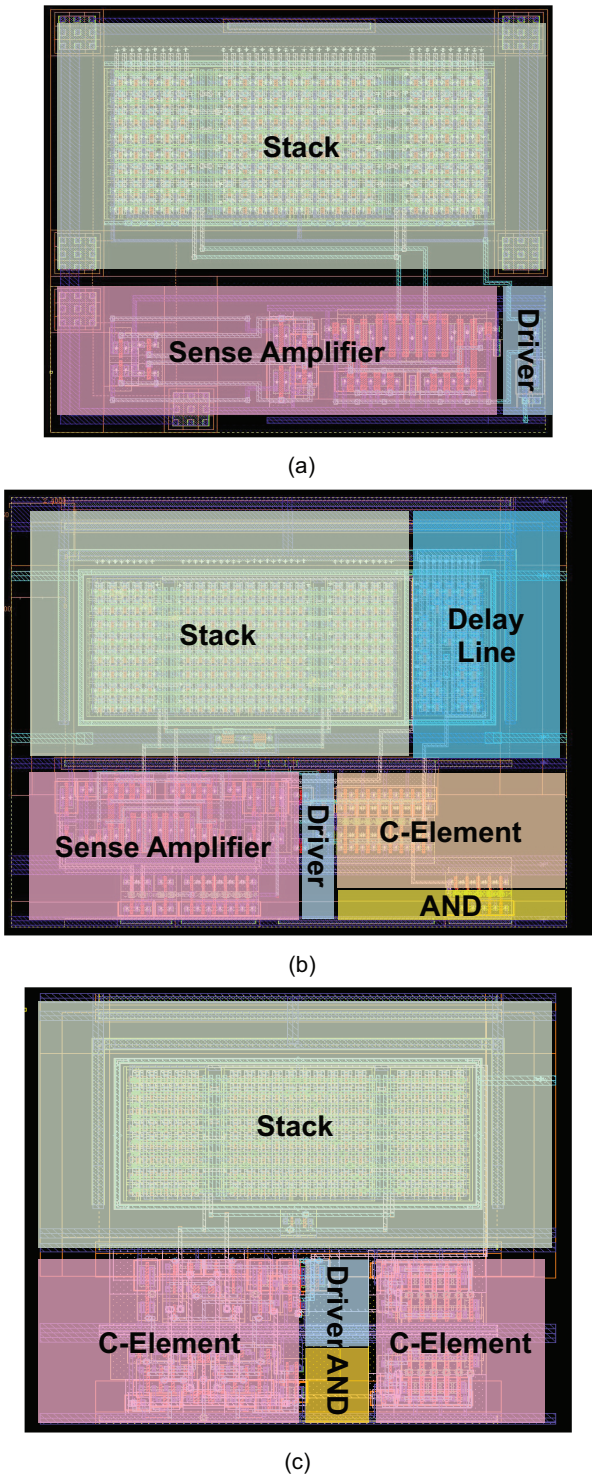

Figure 4.17: SAPTL5 layouts in 90nm CMOS technology: (a) The synchronous layout, (b) the bundled-data layout, including the built-in delay line immediately to the right of the stack, and (c) the dual-rail layout. The size of the stack occupies the top half of the figures and is the same for all three layouts.

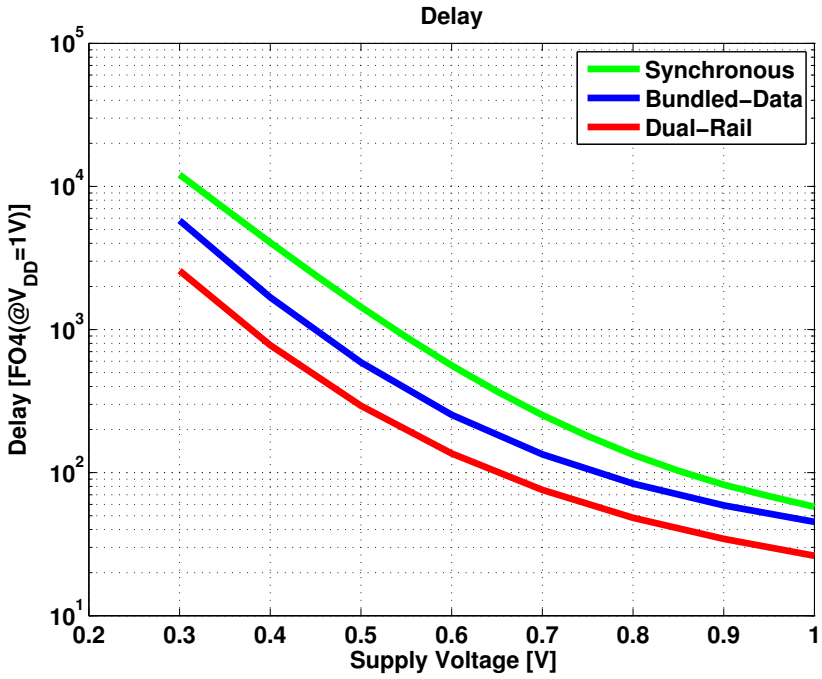

Figure 4.18: Simulated delay results of the SAPTL5 circuits as a function of supply voltage.

4.6.2 Energy-Delay Performance

Fig. 4.18 shows the pre-layout simulated delay results of the synchronous SAPTL5 and both versions of the self-timed SAPTL5 for supply voltages ranging from 1V down to 300mV. Both self-timed designs are better than the synchronous version and the performance advantage is especially larger at a low supply voltage. The dual-rail design can self-adapt to the delay variation and therefore is the fastest design.

The simulated energy and delay behaviors of the synchronous SAPTL5 and both versions of the self-timed SAPTL5 are shown in Fig. 4.19 as the supply voltage is varied from 300mV to 1 V. The simulation results show that both self-timed SAPTL configurations have better energy-delay characteristics than the synchronous design. This is because the sense amplifiers in self-timed SAPTL are able to respond earlier as the output swing of the stack starts to build up. Moreover, employing the early reset glitch-free protocol also reduces the delay and leakage energy. Although the bundled-data SAPTL5 can theoretically achieve its highest speed performance by overlapping the data evaluation and reset cycles, as shown in Eqs. 4.9 to 4.11, the delay line will require extra padding to compensate for process variations and create both delay and energy overhead, as discussed in chapter 3. As a result, dual-rail SAPTL5 emerges as the most efficient design. With a delay of around 3.5ns, the dual-rail SAPTL5 can achieve more than 7% and 32% energy saving compared against the bundled-data and synchronous SAPTL5 designs, respectively.

The leakage of the C-element circuit accounts for most of the overall energy consumption

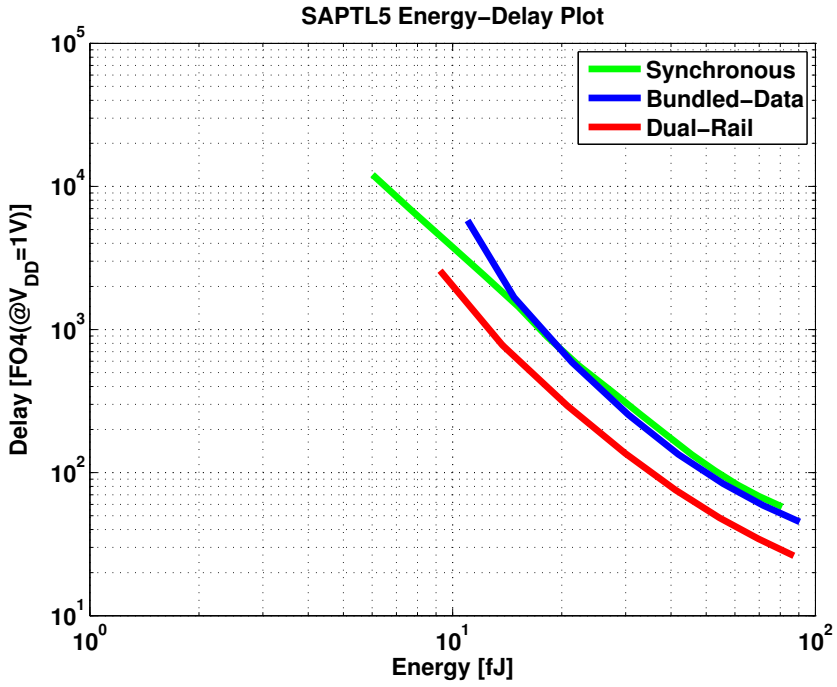

Figure 4.19: Simulated energy-delay plots of the SAPTL5 circuits for the supply voltage ranging from 300mV to 1 V.

when the self-timed SAPTL operates at very low supply voltages and thus prevents us from efficiently achieving any further energy reduction by scaling down the supply voltage. As a result, the synchronous SAPTL architecture appears to be capable of lower energy operation at low supply voltage, as shown in Fig. 4.19. This is misleading, because the energy simulation of the synchronous SAPTL omits the energy contribution from the clock distribution network, which corresponds to the energy cost needed to generate timing information in the self-timed designs locally.

The measured energy and delay plots are superimposed on their respective post-layout simulation results in Fig. 4.20. The measured energy-delay characteristics are very consistent with the simulation results.

4.6.3 Leakage Current Results

The pre-layout simulated leakage behavior of the different SAPTL5 designs as a function of supply voltage is shown in Fig. 4.21. The slight increase in the self-timed SAPTL5 leakage current is due to the added handshaking circuits needed for asynchronous operation. However, as shown in Fig. 4.22, both self-timed designs have lower leakage energy than the synchronous design due to less timing uncertainty. The dual-rail self-timed SAPTL has the least idle time and therefore has the lowest leakage energy and achieves the best energy

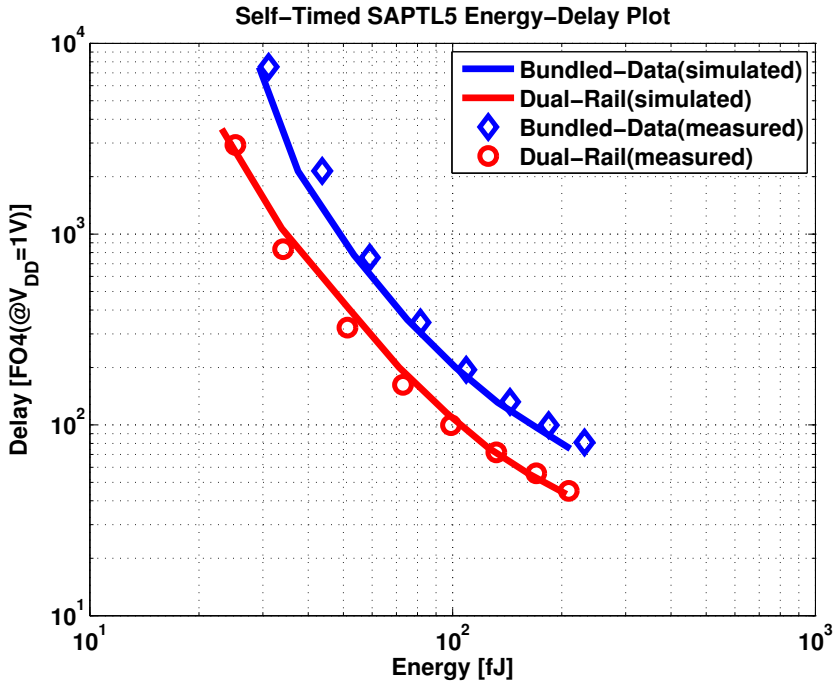

Figure 4.20: Measured versus simulated energy-delay plots for the 90nm CMOS self-timed SAPTL5, as the supply voltage is varied from 300 mV to 1 V.

efficiency.

The measured leakage currents as a function of supply voltage are superimposed on their respective post-layout simulation results in Fig. 4.23. Due to the measurement limit of the instrument, the measurement results at low supply voltages appear to be less accurate.

4.7 Summary

The introduction of asynchronous operation in SAPTL provides better robustness in the presence of variability, as well as performance advantages over synchronous operation. While the self-timed SAPTL using the bundled-data protocol can potentially achieve higher speed performance by overlapping the data evaluation and reset cycle, the self-timed design based on the dual-rail protocol has less rigid relative timing constraints, which leads to better energy and speed performance in technologies with increased process variations. The early reset operation of self-timed SAPTL not only prevents dynamic timing hazards from glitches but also improves both energy and speed performance.

The low implementation cost of the asynchronous operation makes the self-timed SAPTL family a very promising candidate to realize robust and low-energy computations. The dual-rail SAPTL architecture shown in Fig. 4.12 is a design that can be used to realize

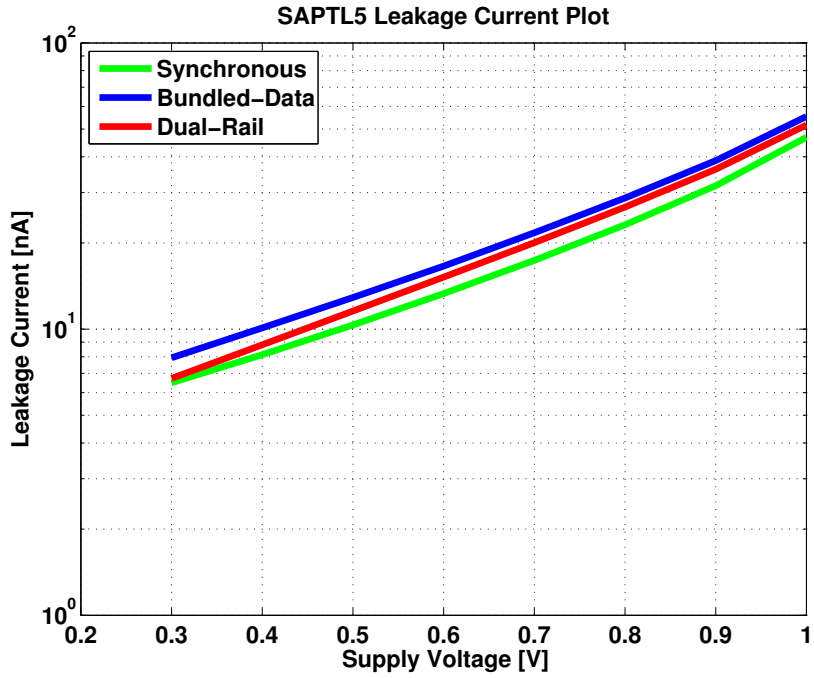

Figure 4.21: Simulated leakage current of the SAPTL5 circuits as a function of supply voltage.

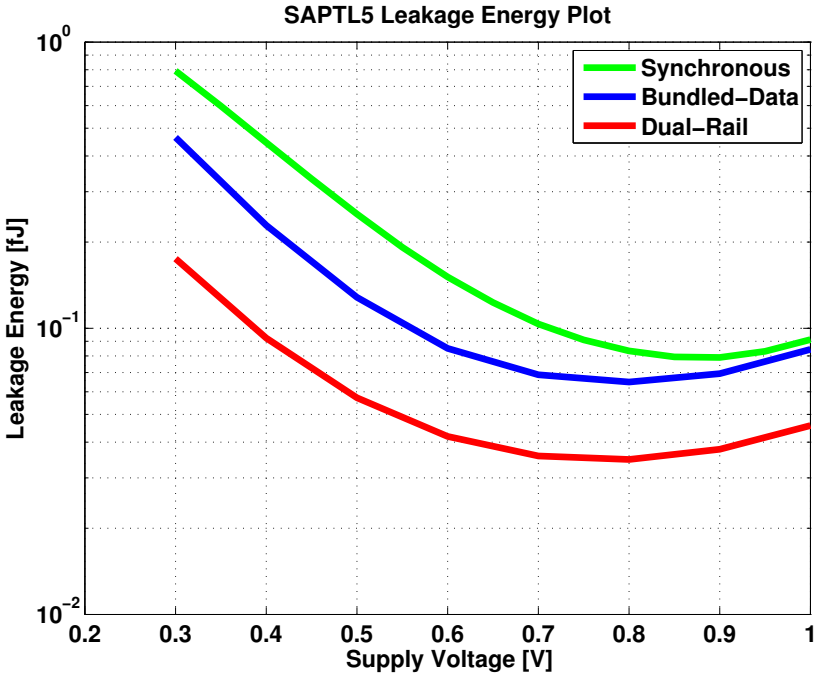

Figure 4.22: Simulated leakage energy of the SAPTL5 circuits as a function of supply voltage.

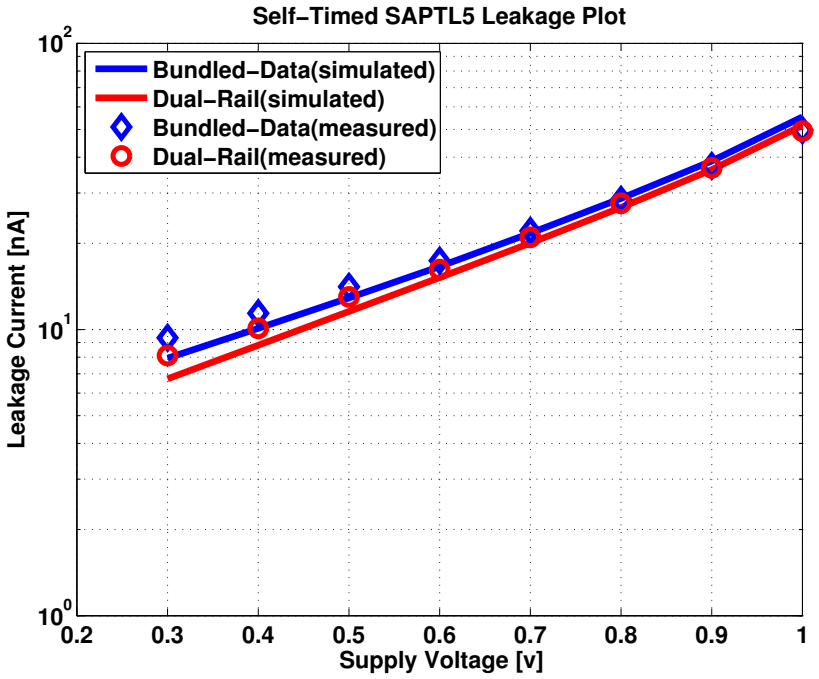

Figure 4.23: Measured versus simulated leakage current of the self-timed SAPTL as a function of supply voltage.

asynchronous computations with the least hardware complexity. It should be noted that the results presented here are local within the context of one or a few self-timed modules. Local results can be misleading. For instance, most of a typical synchronous system's power budget may be spent on the clock distribution, which could account for 7% to as high as 40% of the total system power [Bhunia04, Banerjee06]. In order to gain more insight into asynchronous operation, performance comparison between synchronous and asynchronous system should also be done at the system level in the context of actual application. An ultra-low power neural signal processor design using asynchronous self-timed circuits is therefore presented and compared with a synchronous timing based design in the next chapter.

Chapter 5

Case Study II: Asynchronous Processor Design

Biomedical sensor is one of the emerging applications that demand for ultra-low power and energy performance. Because of safety regulations and miniature size constraints, the electrical integrated system embedded in a biomedical sensor must be extremely energy-efficient and reliable, while speed performance requirement is usually much relaxed. In chapter 4, the energy and performance advantages of asynchronous computing in sense amplifier-based pass transistor logic have been shown. To further demonstrate the benefit of asynchronous operation at low supply voltages, an ultra-low power neural signal processor design using asynchronous self-timed circuits for brain-machine interfaces (BMIs) is presented in this chapter. Section 5.1 introduces the background and motivation of neural signal processing, as well as the major functions and performance requirements of an implantable neural signal processor. The design methodology and circuit implementation of an ultra-low power asynchronous neural signal processor are presented in section 5.2. Section 5.3 shows the test chip implementation of the proposed asynchronous neural signal processor in 65nm CMOS technology. Measured results of both synchronous and asynchronous designs are presented and compared to the state-of-the-art processors in section 5.4. Section 5.5 concludes this chapter.

5.1 Neural Signal Processing for Brain-Machine Interfaces

During the last decades, tremendous advances in neuroscience and engineering allow us to better understand the functions of human brain and to have the opportunities of improving the quality of life for patients. In particular, brain-machine interface (BMI) has made a huge progress toward directly decoding electronic signal from the human brain and simultaneously controlling the prosthesis for the physically impaired. One major implementation challenge of

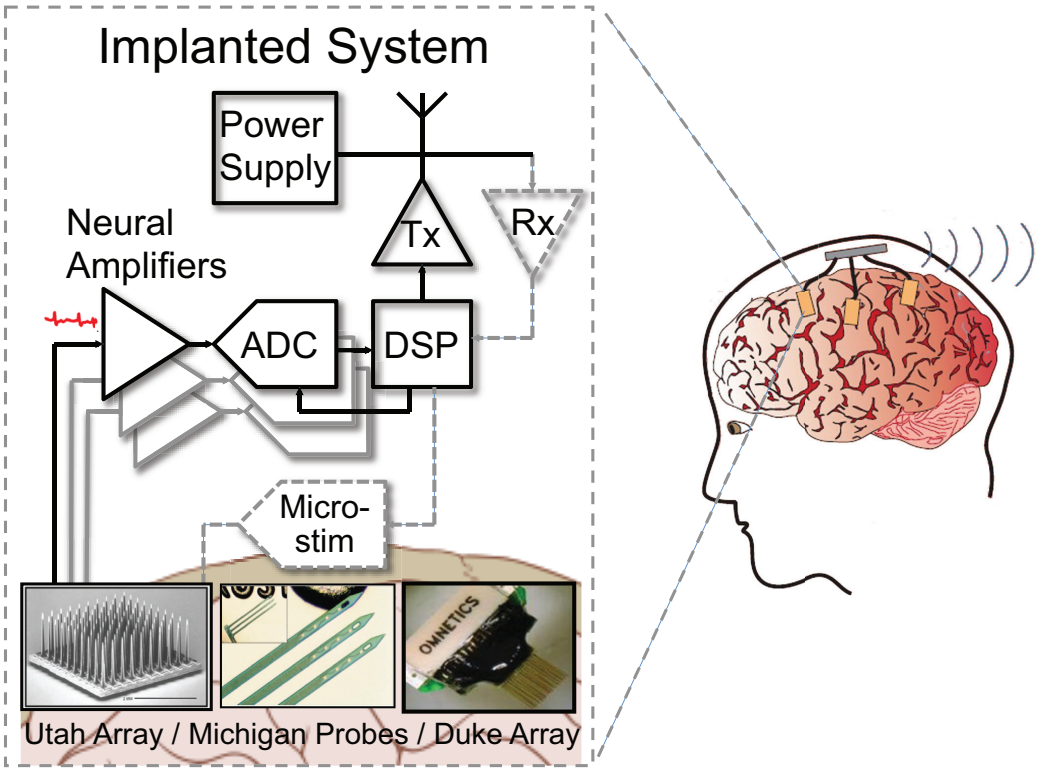

Figure 5.1: A generic integrated circuit system for brain-machine interfaces [Muller12].

a BMI system is to develop an mm-size implantable neural sensor with wireless connectivity [Rabaey11]. Fig. 5.1 shows a generic integrated circuit system for BMIs [Muller12]. The electrical integrated system for a neural implant is typically composed of a neural probe array, an analog neural acquisition front-end, a digital signal processor (DSP), an RF radio, and power delivery circuitry. The miniature size and safety regulations demand for an extremely low power operation of the whole system.

The DSP in Fig. 5.1 serves two functions in a BMI system. First, since each electrode in a probe array may pick up spike signals from multiple neurons, for a closed-loop BMI system, a DSP is necessary to perform real-time neural signal processing and classify different spike signals based on their source neurons. Second, the neural signals recoded from the front-end contain both spike signals and unwanted noise. Real-time neural signal processing can extract only useful spike information, which can substantially reduce the radio communication data rate. As a result, the radio can either consume less power with a lower output data rate, or support a larger number of communication channels simultaneously.

The process of mapping detected spike signals into their source neurons is called spike sorting, as shown in Fig 5.2 [Karkare11]. Spike sorting usually involves three steps: (a) Spike Detection, finding out the spike event from the raw recoded data; (b) Feature Extraction, transforming spikes into a feature space; (c) Clustering, classifying spikes into different groups

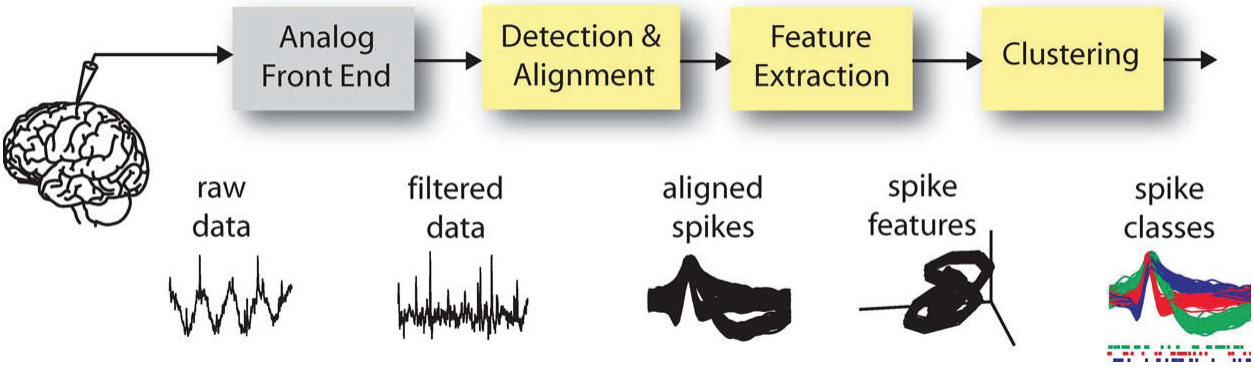

Figure 5.2: Spike-sorting process [Karkare11].

based on extracted features. Various spike-sorting algorithms designed for different applications have been discussed and summarized in [Gibson08] and [Gibson10]. In the next section, the spike detection and feature extraction algorithms that exhibit the power-density characteristics best suitable for real-time wireless neural signal processing is used to implement a low-power asynchronous neural signal processor.

5.2 Asynchronous Neural Signal Processor Design

In order to realize an ultra-low power computation, the neural signal processor is designed and implemented with asynchronous self-timed circuits. Fig. 5.3 shows the schematic of the proposed asynchronous neural signal processor. The prototype asynchronous processor implements a neural spike-sorting function in three steps: spike detection, spike alignment and feature extraction. The processor receives the 8b digitized data from a neural signal acquisition front-end running at 20kHz designed in [Muller12].

To lower the supply voltage as much as possible to reduce the leakage power, the differential dynamic logic and a 4-phase return-to-zero (RTZ) handshaking protocol is used to implement the asynchronous operation, instead of using self-timed SAPTL described in chapter 4. The differential dynamic logic is sized such that it consumes 40% less leakage in standby versus active mode, as shown in Fig. 5.4. The RTZ protocol further suppresses the leakage by another 10% by resetting each gate input during standby.

The input synchronous bit streams are first converted into dual-rail RTZ data with the necessary request-acknowledge handshaking control signals by a synchronous-to-asynchronous interface. The interface is composed of differential pre-charged dynamic logic buffers. The data processing thereafter is driven only by local handshaking events without a global synchronous clock, eliminating any timing uncertainty such as skew and jitter associated with clock distribution in conventional synchronous design. Each block operates at it own speed and achieves best-effort performance that adapts to the changing operating condition. The

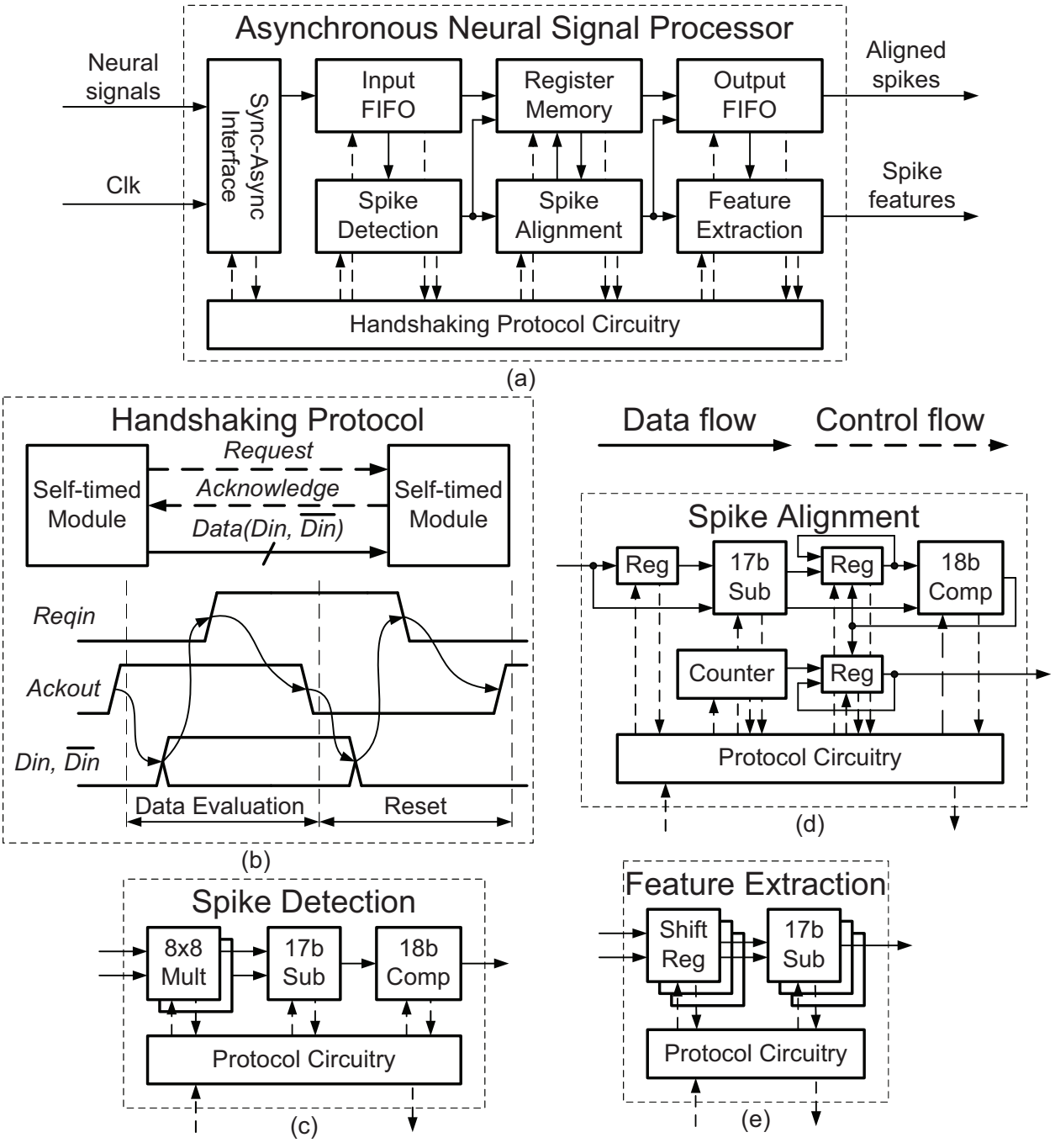

Figure 5.3: Schematic of the proposed asynchronous neural signal processor.

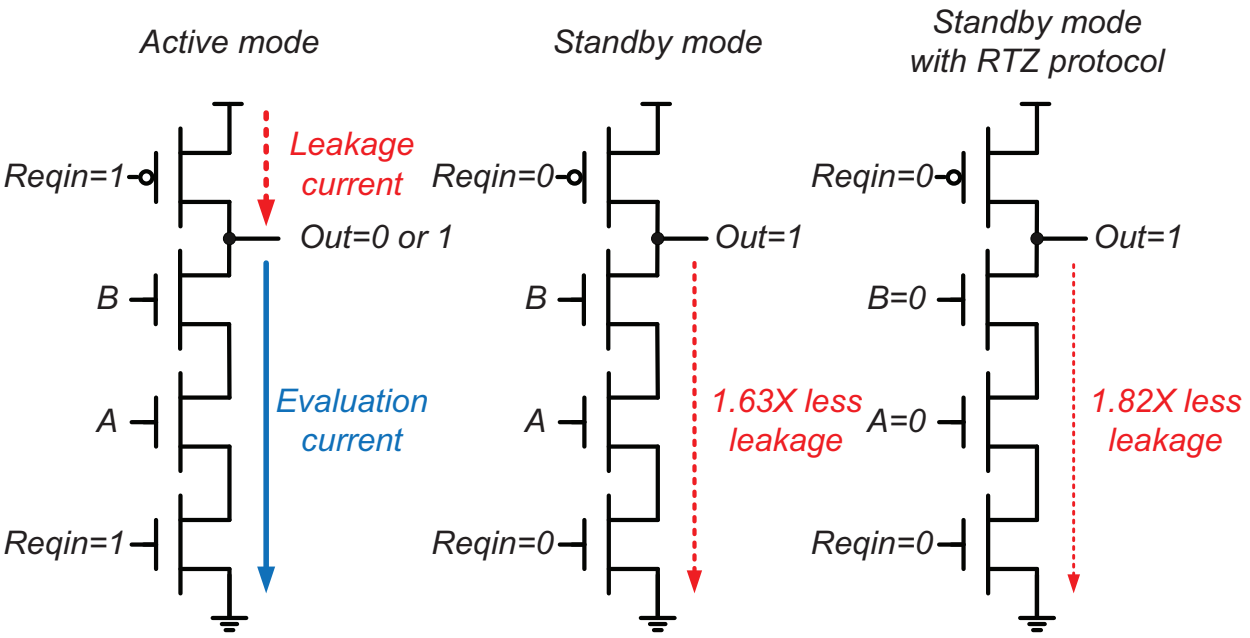

Figure 5.4: Leakage characteristics of a dynamic logic gate in different operating modes.

communication between each module is reliably governed by a self-timed 4-phase dual-rail handshaking protocol, as shown in Fig. 5.3(b), avoiding any setup- and hold-time violations.

The spike detector shown in Fig. 5.3(c) identifies the spike events by comparing the signal energy of the raw neural data to a threshold calculated during the initial training period. The spike detection process involves two 8b multiplications, a 17b subtraction and an 18b comparison, determining the critical path of the processor. Designed to operate at a 0.25V supply with 50mVpp supply noise, the detector exhibits switching activities from 0.2 to 0.01 depending on the operating condition and signal statistics, consuming mostly leakage power. Fig. 5.5 shows circuit implementations of a booth X2 decoder using static CMOS logic and differential dynamic CMOS logic. The differential dynamic CMOS logic topology used in this design can realize arithmetic functions with fewer logic gates than the static CMOS-based design, which effectively reduces the leakage paths. Most of logic gates in the datapath were selected to have fan-ins of four achieving better leakage characteristics while maintaining output signal integrity. Algorithms such as booth encoding are employed to minimize the overall gate count, further reducing the total number of leakage paths by 1.5X. The protocol circuitry regulates the dataflow and manages the transition between different operating modes. Therefore, immediately after a computation completes, the self-timed datapath can enter its standby mode to minimize leakage, as shown in Fig. 5.4. The simulation result shows that the employment of low-leakage logic topology together with asynchronous timing effectively reduces the leakage of the datapath by 10X.

The identified spikes are then aligned to the sample with the maximum derivative to improve the classification accuracy. The aligned spikes are finally transformed into a feature space

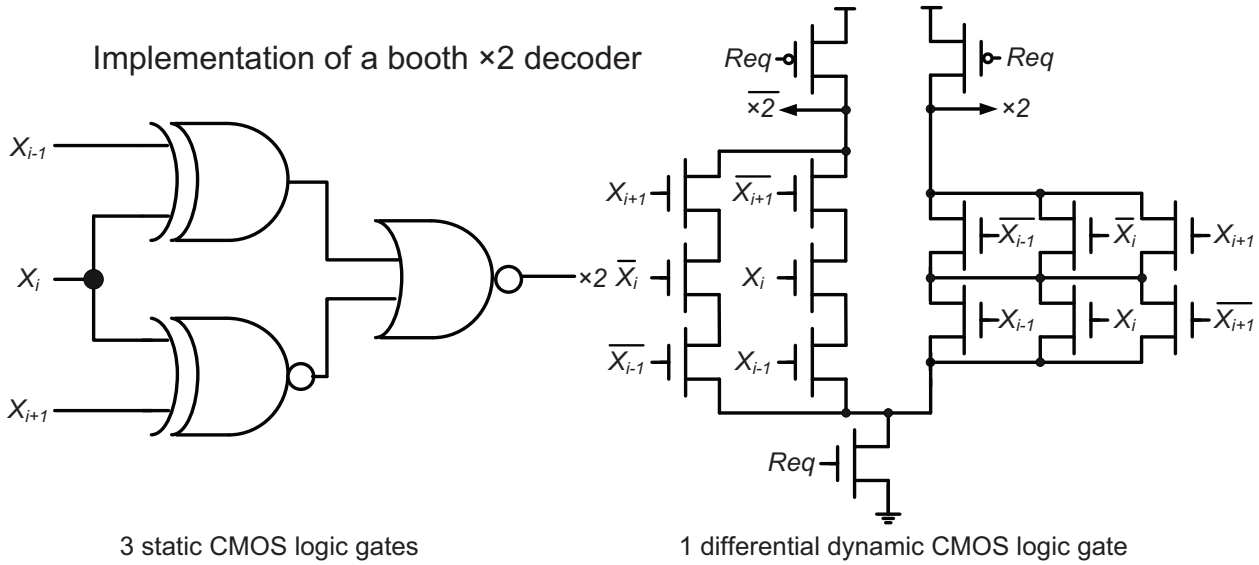

Figure 5.5: Circuit implementations of a booth X2 decoder using different logic styles.

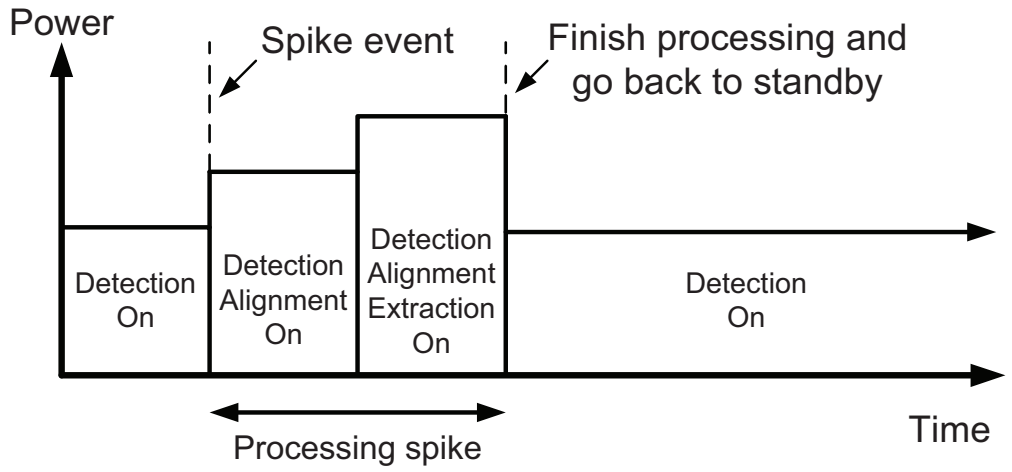

Figure 5.6: Power profile of neural signal processor during operation.

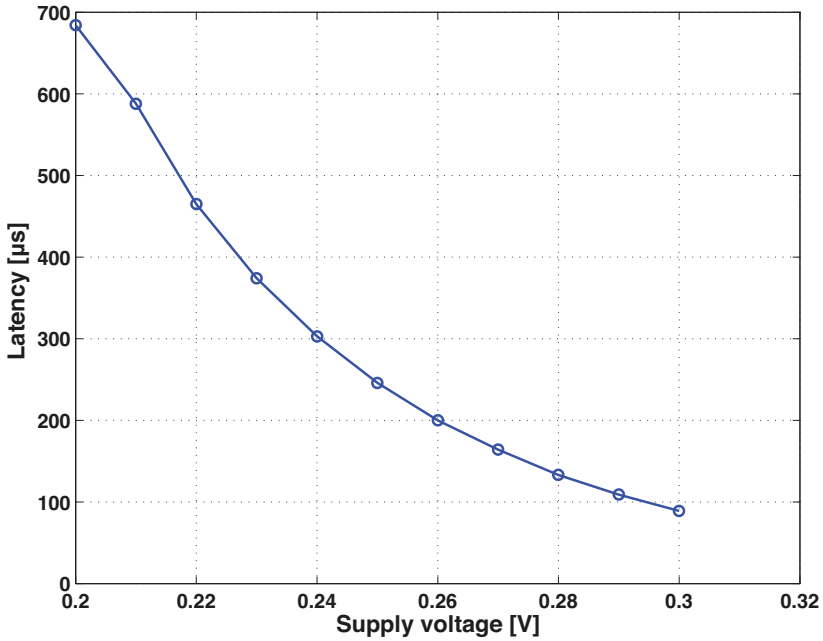

Figure 5.7: Processing latency as a function of supply voltage.

that better characterizes spikes from different neurons. The self-timed alignment and feature extraction (SAFE) module shown in Fig. 5.3(d) and 5.3(e) is activated only when a spike event is detected, therefore exhibiting a variable block activity ranging from 0.2 to 0.02 depending on the actual spike activity. The power consumption of the entire processor highly depends on the SAFE module activity, varying by 1.8X, as shown in Fig. 5.6. Asynchronous SAFE module processes data adaptively according to the different operating conditions, resulting in variable processing latencies rather than a fixed latency found in conventional synchronous designs. Fig. 5.7 shows the processing latency of asynchronous processor at different supply voltages. The capability of adjusting processing latency adaptively with changing operating conditions ensures that the SAFE module is active only for the shortest possible time in all conditions, avoiding any idle time and thus minimizing the power. The SAFE module demonstrates 4X less power consumption than a synthesized synchronous counterpart.

5.3 Test Chip Implementation

The layout details of the proposed asynchronous neural signal processor are shown in Fig. 5.8. Register memory (40%) and input/output FIFO (20%) occupy most of processor area. Both synchronous and asynchronous processors were fabricated in a 65nm CMOS technology using standard V_t devices for performance comparison. The devices with lower leakage

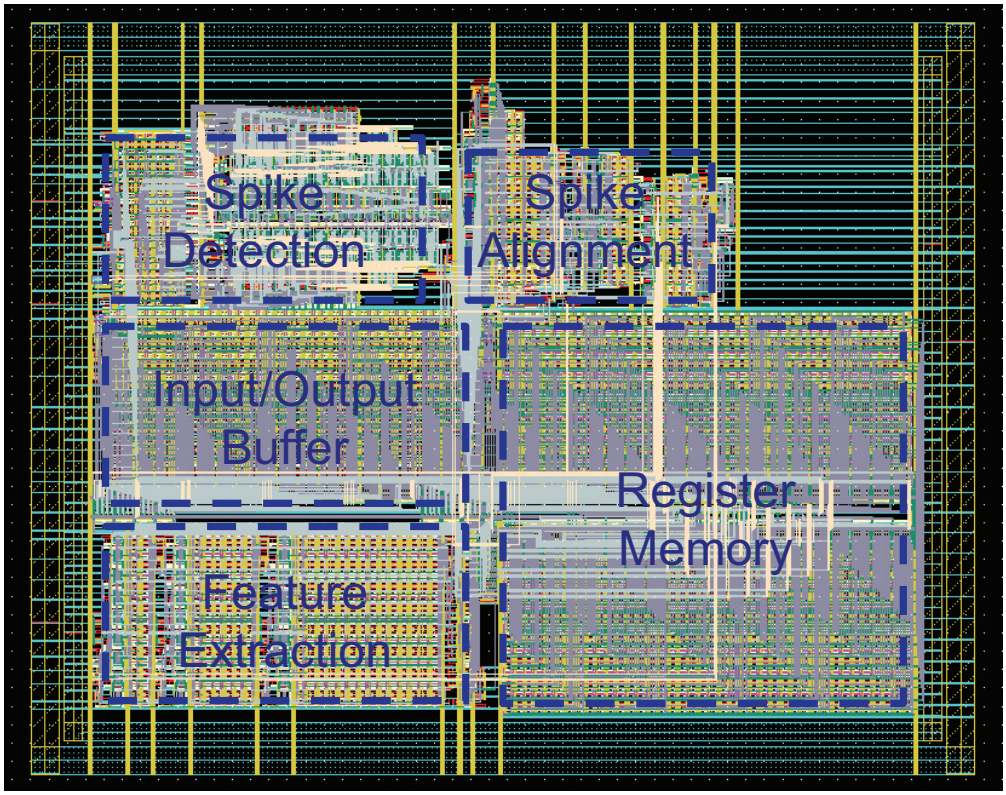

Figure 5.8: Layout of the proposed asynchronous neural signal processor.

characteristics, such as LP transistor, are not employed on purpose to better quantify the performance improvement resulted solely from the asynchronous operation. A microphotograph is shown in Fig. 5.9. The asynchronous processor core occupies an area of $0.03mm^2$, about 1.5X larger than the synchronous design due to additional protocol circuitry overhead and differential structure. Fig. 5.10 shows the test circuit board.

5.4 Results

Fig. 5.11 shows the measured dynamic behavior of asynchronous neural signal processor during operation at 0.25V. The top purple signal is processor wake-up signal; the green signal is aligned spike output signal; the yellow signal is internal handshaking control signal; the bottom blue signal is input synchronous clock from the acquisition front-end. The whole processing latency takes 40 handshaking cycles. The measured handshaking cycle time is much smaller than the input clock period, which indicates that the transistors are at the fast corner. This demonstrates that the asynchronous processor is able to self-adapt to process variations and automatically operate at a higher speed than the input clock rate. As shown in Fig. 5.11, once the operation starts, the asynchronous processor operates at a full speed and goes back to low-power standby mode immediately after the computation completes,

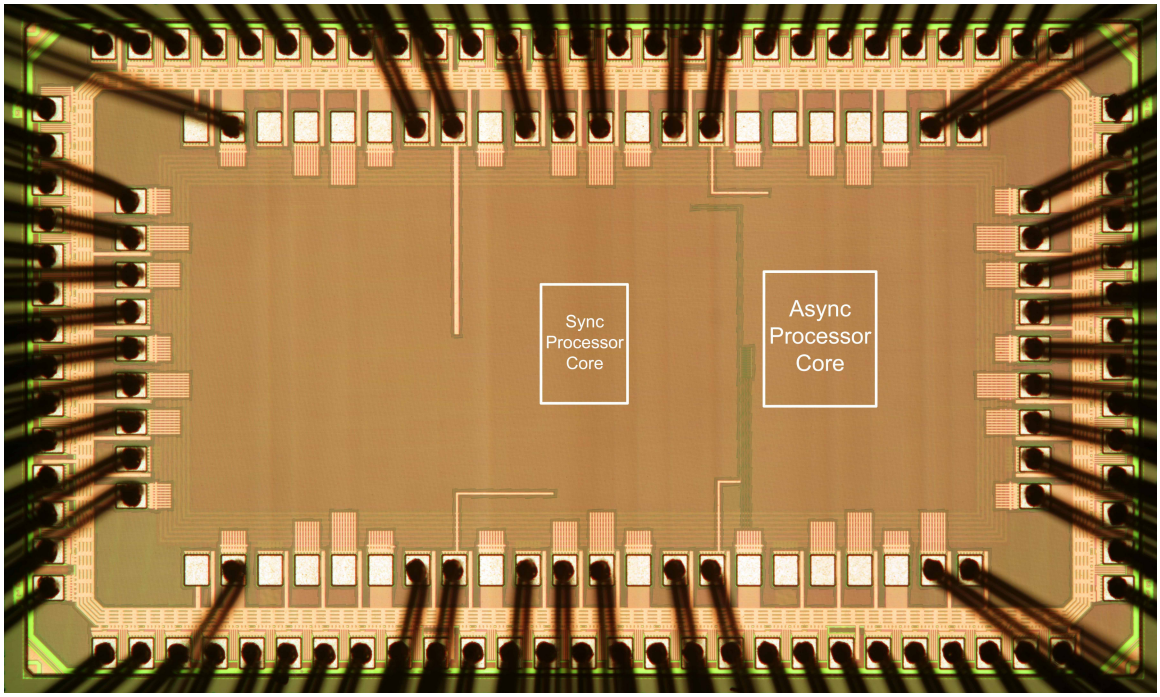

Figure 5.9: Die photo of test chip.

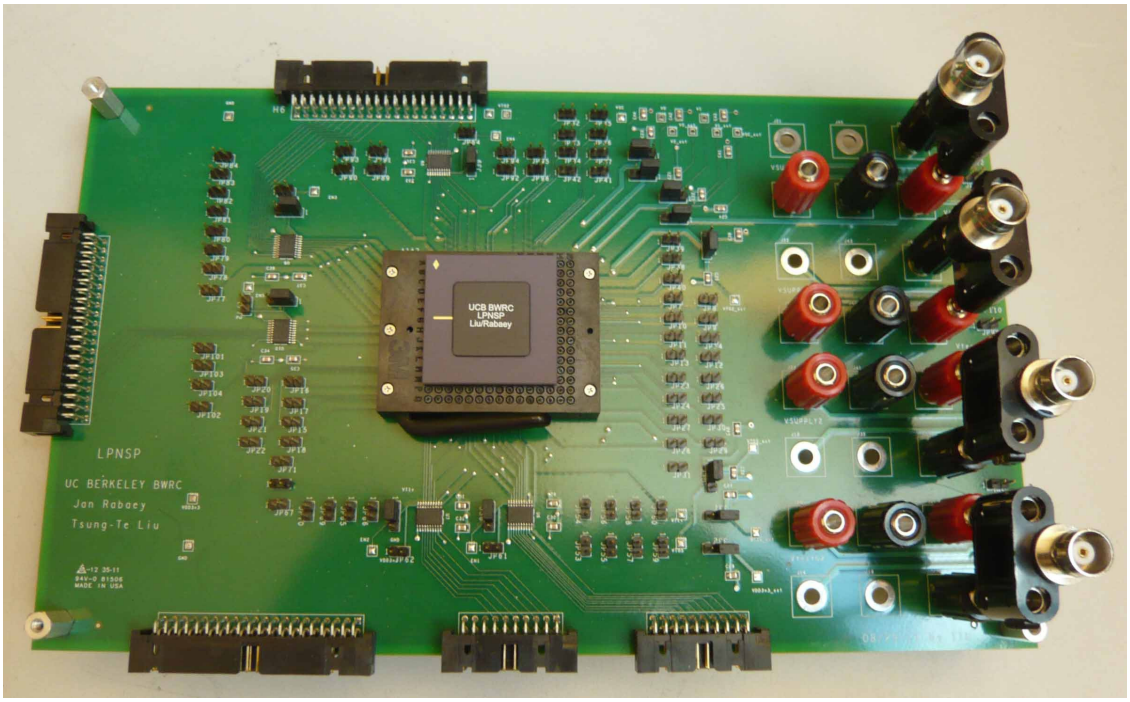

Figure 5.10: Asynchronous processor test board.

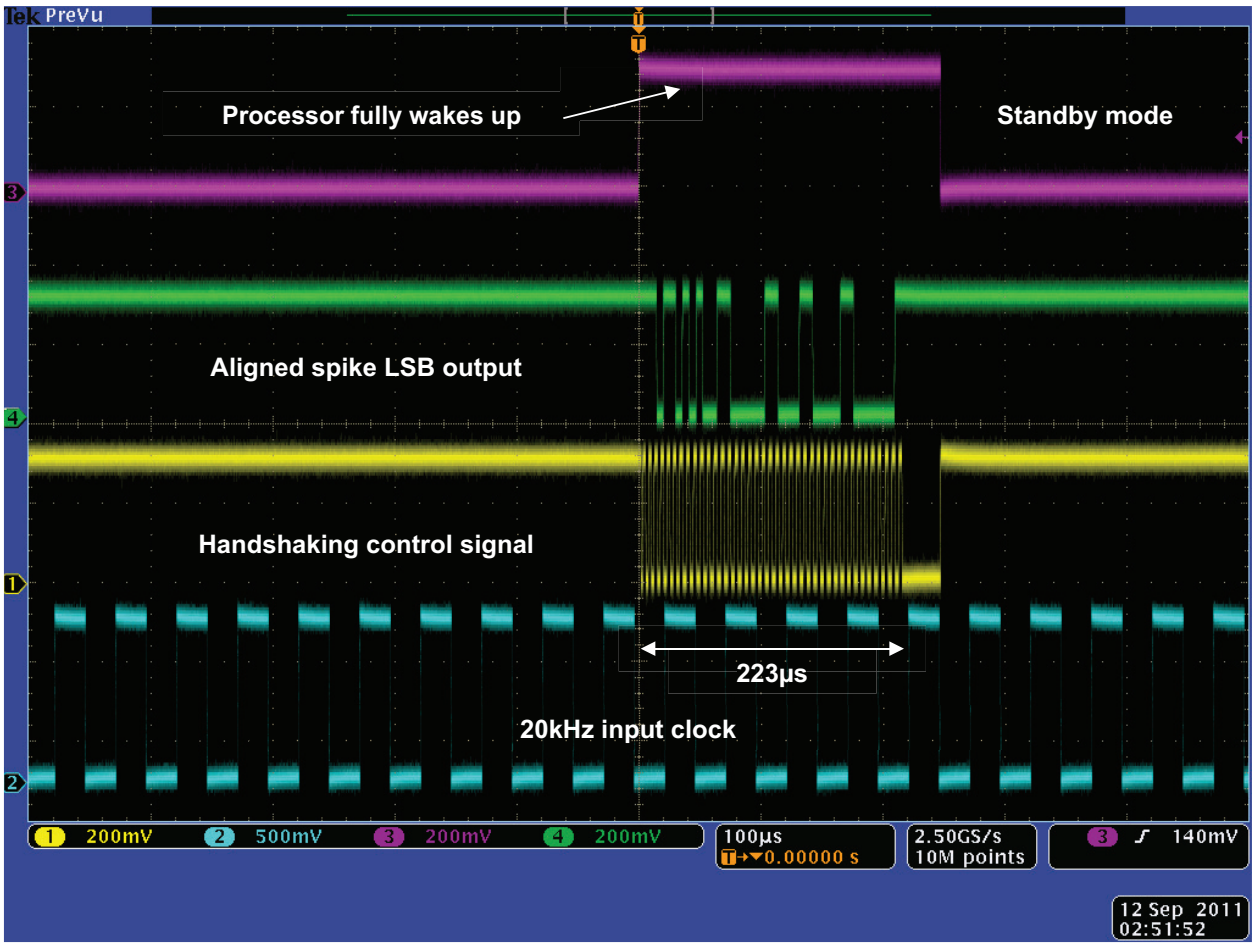

Figure 5.11: Dynamic behavior of asynchronous processor at 0.25V.

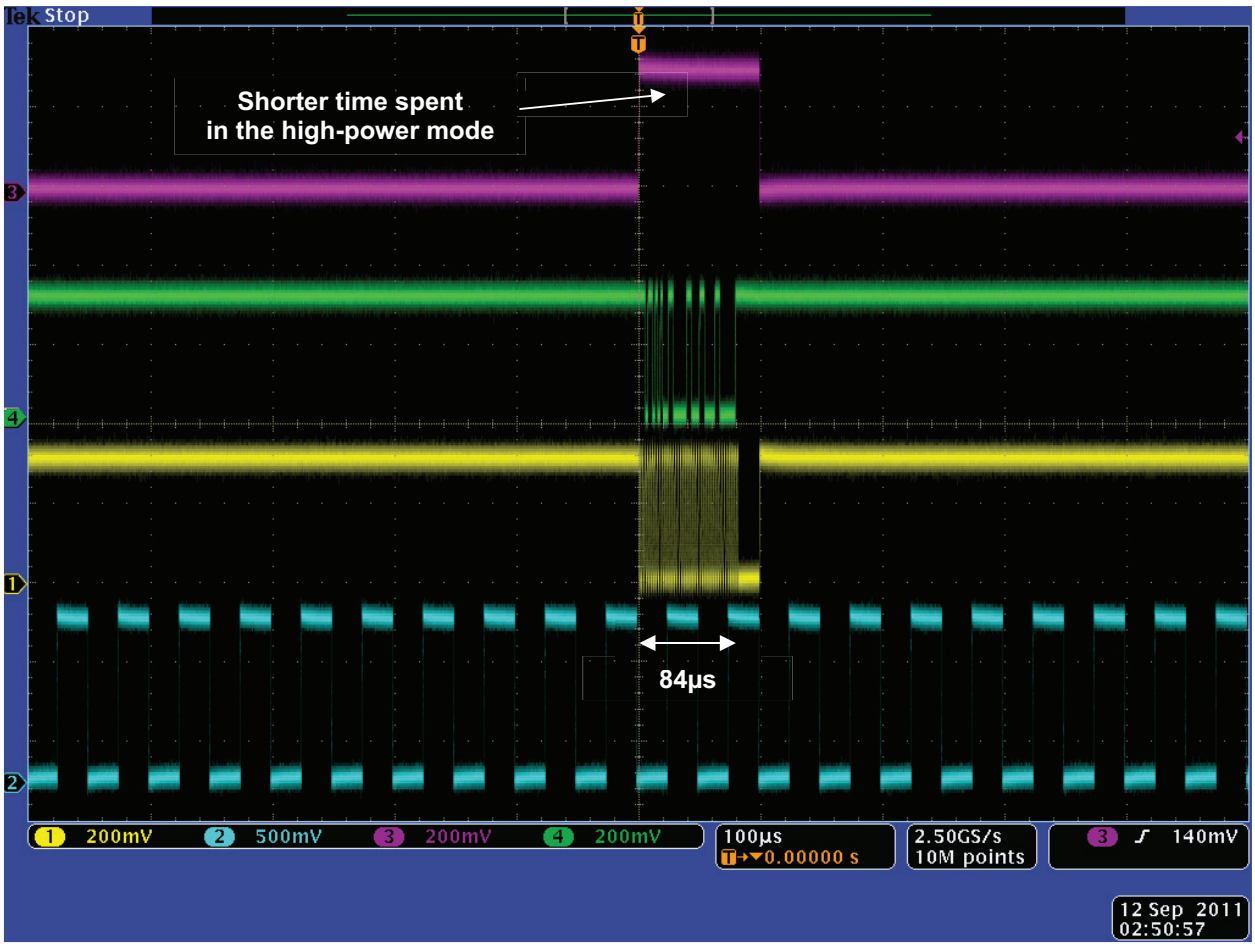

Figure 5.12: Dynamic behavior of asynchronous processor at 0.3V.

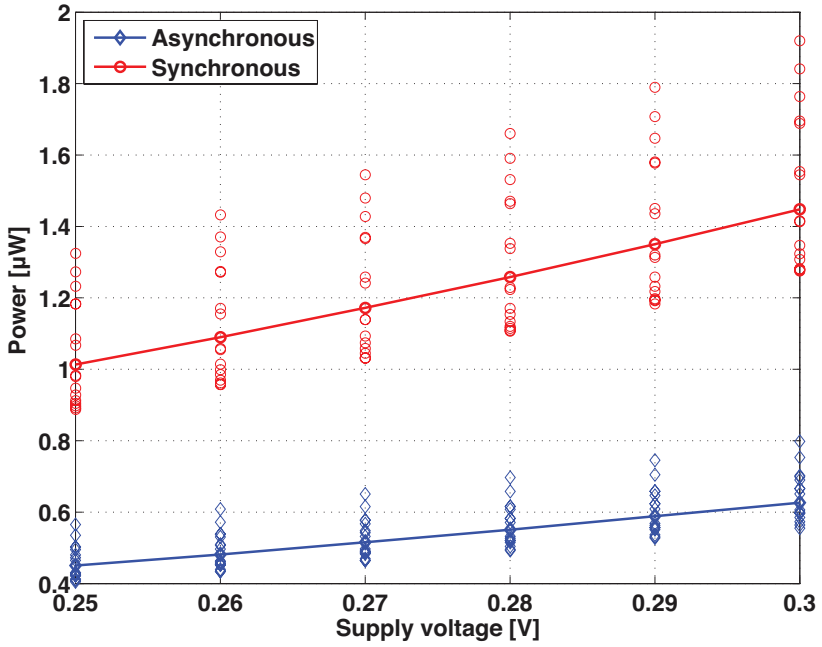

Figure 5.13: Measured power consumption results of 16 test chips.

avoiding any idle time and minimizing the power. Fig. 5.12 shows the dynamic behavior of asynchronous neural signal processor after the supply voltage is raised from 0.25V to 0.3V during operation. The measured handshaking cycle time is further reduced at 0.3V supply voltage, which indicates that the processor is also capable of adapting to supply voltage change and operating at an even higher speed acoordingly, without being limited by the fixed-rate input clock.

Fig. 5.13 shows the measured power results from 16 chips for supply voltage ranging from 0.25V to 0.3V. The synchronous results are on the top red curve, while the asynchronous results are on the lower blue curve. All chips are fully functional at 0.25V supply, while the best chip can operate down to 0.2V consuming 290nW. The malfunction of memory and sequential circuitry cause the asynchronous neural signal processor to fail to operate normally below 0.2V. The measurement results show that the asynchronous design reduces the average power by 2.3X compared to the traditional synchronous approach. Fig. 5.14 shows the measured latency-power plots of asynchronous processors at 0.25V. The processors show different performance characteristics due to process variations. Processors with higher speed performance consume more power. However, asynchronous self-timed operation ensures that faster processors can finish the operation earlier and immediately go back to low-power standby mode, which alleviates the impact of variations. As a result, an asynchronous design achieves better power statistical characteristic, as shown in Fig. 5.13. At 0.25V, the power spread (σ/μ) of 16 asynchronous designs is 0.1, 1.4X smaller than synchronous designs (0.14).

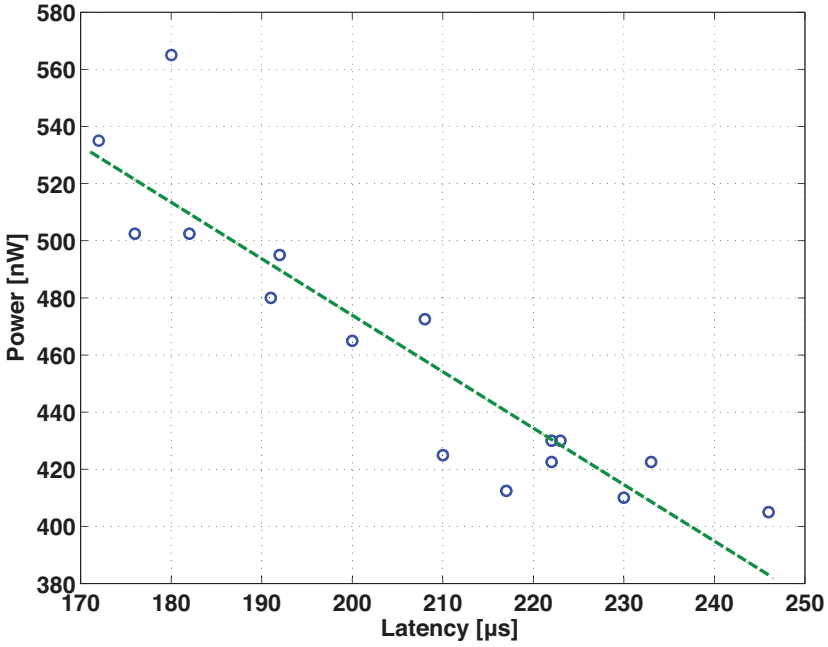

Figure 5.14: Measured power-latency results of 16 asynchronous processors at 0.25V.

Table 5.1 shows the performance summary of synchronous and asynchronous designs and comparison with the state-of-the-art identical neural signal processors. The proposed asynchronous neural signal processor occupies similar normalized silicon area to other processor designs. Note that this design was implemented in 65nm CMOS technology using only standard V_t devices. As a result, the technology itself actually exhibits worse transistor leakage characteristic than other technologies where processors [Chae08] and [Karkare11] were implemented. However, by employing the optimal circuit architecture and timing scheme to minimize the overall circuit leakage, the proposed asynchronous design demonstrates a 4.4X reduction in power, a 3.7X reduction in energy and a 2.2X reduction in power density, when compared to the state-of-the-art processors.

5.5 Summary

Low-power neural signal processor is an essential component to perform real-time spike sorting and to reduce the communication date rate in an implantable neural sensor. The proposed neural signal processor design exploits an asynchronous timing strategy to dynamically minimize leakage and to self-adapt to the process variations and different operating conditions. Based on a logic topology with built-in leakage suppression, the self-timed processor demonstrates robust sub-threshold operation down to 0.25V, while consuming only 460nW in a 65nm CMOS technology, representing a 4.4X reduction in power compared to

Table 5.1: Comparison to the state-of-the-art processors and performance summary

Reference	[Chae08]	[Karkare11]	Synchronous	Asynchronous
Power (μW/channel)	100	2.03	1.04	0.46
Power Spread (σ/μ) @0.25V	N/A	N/A	0.14	0.1
Power Density ($\mu W/mm^2$)	63.29	33.83	52	15.33
Spike Sampling Rate (kHz)	40	24	20	20
Processing Energy (pJ)	2500	84.58	52	23
Core Voltage (V)	3.3	0.55	0.25	0.25
CMOS Process (nm)	350	90	65	65
Normalized Area (to 65nm)	1.82	1.04	0.67	1

the state-of-the-art designs.

Chapter 6

Conclusion and Future Work

This research introduces an alternative approach to realize robust and energy-efficient computation using asynchronous self-timed circuits. It first presents the behavior modeling and performance analysis of asynchronous circuits, and identifies the sweet spot of self-timed operation. The design and implementation of energy-efficient self-timed circuits are demonstrated and verified in two test chips in 65nm and 90nm CMOS technology, respectively.

6.1 Contributions

The main contributions of this research are:

- The development of statistical analysis framework to evaluate energy and delay performances of CMOS circuits with different timing methodologies in the presense of variability.
- The analysis and optimization of robust and energy-efficient handshaking protocols for low-voltage operation.
- The design and implementation of self-timed sense amplifier-based pass transistor logic that demonstrates better energy-delay characteristics than synchronous approaches.
- The demonstration of an asynchronous neural signal processor for brain-machine interface with inherent leakage suppression, realizing energy-efficient and robust subthreshold computation.

6.2 Future Work

This work takes the first step in the realization of ultra-low energy computation using asynchronous self-timed circuits. The performance advantages of asynchronous computation at

low supply voltages are clearly demonstrated from a statistical analysis framework, as well as from two CMOS test chips that implements self-timed circuits and systems. Yet there is still a lot of work to be done to make asynchronous circuits a truly viable solution for ultra-low energy computation.

The statistical analysis models introduced in chapter 3 consider only the impact of process variations. In order to more accurately predict actual circuit performance, other variation sources such as input signal statistics, voltage variations and temperature changes, can be incorporated into the proposed analysis models. In addition, since the proposed analysis model considers only the delay overhead required to compensate the critical path delay variations, it estimates only the "lower bound" of delay overheads required for different timing approaches, i.e., the upper bound of performance capacity. For example, in order to reflect actual implementation cost of synchronous approach, the timing uncertainty resulted from global clock distribution, such as skew and jitter, must be included in future analysis model. This usually severely increases the delay overhead required for synchronous approach. The setup- and hold-time of latch and flip-flop under variations must also be modeled to accurately predict circuit performance in pipeline structure.

On the other hand, in order to characterize the behavior and performance of asynchronous circuits accurately, more elaborate estimation of communication overhead during the hand-shaking process is essential. The communication costs of implementing different pipeline protocols have been modeled and compared in [Stevens11] for a single operating point. The work in [Stevens11] can be extended to cover various operating points, and combined with the proposed analysis models, which would characterize both computation and communication overheads of asynchronous circuits accurately across a wide range of operation points.

The design examples introduced in chapter 4 and chapter 5 are designed and optimized to minimize both the area and energy performance in a standard CMOS technology. The design optimization in these examples is mainly performed at the logic and circuit levels. At the device level, various-threshold devices and body biasing techniques can be employed at the same time if the implementation technology supports these extra features. Architectural optimization and design techniques at system level, such as parallelism and multiplexing, can also be explored to enhance energy and delay performance [Marković10].

Many digital processors today are capable of scaling operating frequency and supply voltage dynamically with adaptive voltage-tuning and frequency-tuning circuitry to support a wide range of operating conditions and application loads [Bul111, Bol12]. The same techniques can easily apply to asynchronous self-timed processor with much less implementation overhead than synchronous timing based approach. Since an asynchronous system can automatically operate at a full speed and achieve full capacity at a given operating point, the use of feedback frequency-tuning loop becomes unnecessary. As a result, the capability of an asynchronous processor can be easily enhanced with additional voltage-tuning circuitry.

The malfunction of memory and sequential circuitry is the reason that the asynchronous neural signal processor presented in chapter 5 fails to operate normally below 0.2V, instead of the miss of speed performance. By employing memory and sequential circuits designed

specially for low-voltage operation, the supply voltage can be further scaled down to minimize energy consumption.

Finally, most of asynchronous circuits presented in this work are designed and implemented based on a semi-custom design flow [Pierson07, Alarcón10]. The continuous development of CAD tool for asynchronous circuit design will be essential for future large-scale system synthesis and optimization.

Bibliography

[Alarcón07] L. Alarcón, T.-T. Liu, M. Pierson, and J. Rabaey, "Exploring very low-energy logic: A case study," *Journal of Low Power Electronics*, vol. 3, no. 3, pp. 223–233, Dec. 2007.

[Alarcón10] L. P. Alarcón, *Sense amplifier-based pass transistor logic*, Dissertation, University of California, Berkeley, Dec. 2010.

[Banerjee06] N. Banerjee, K. Roy, H. Mahmoodi, and S. Bhunia, "Low power synthesis of dynamic logic circuits using fine-grained clock gating," in *Proceedings of Design, Automation and Test in Europe, DATE'06*, Mar. 2006, vol. 1, pp. 1–6.

[Bhunia04] H. Li, S. Bhunia, Y. Chen, K. Roy, and T. Vijaykumar, "DCG: deterministic clock-gating for low-power microprocessor design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 245–254, Mar. 2004.

[Bol12] D. Bol, J. De Vos, C. Hocquet, F. Botman, F. Durvaux, S. Boyd, D. Flandre, and J. Legat, "A 25MHz 7μW/MHz ultra-low-voltage microcontroller SoC in 65nm LP/GP CMOS for low-carbon wireless sensor nodes" in *2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2012, pp. 490-491.

[Bowman02] K. A. Bowman, S. G. Duvall, and J. D. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 183-190, Feb. 2002.

[Bowman11] K. A. Bowman, J. W. Tschanz, S.-L. L. Lu, P. A. Aseron, M. M. Khellah, A. Raychowdhury, B. M. Geuskens, C. Tokunaga, C. B. Wilkerson, T. Karnik, and V. K. De, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 194-208, Jan. 2011.

[Bul111] D. Bull, S. Das, K. Shivashankar, G. S. Dasika, K. Flautner, and D. Blaauw, "A power-efficient 32 bit ARM processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 18-31, Jan. 2011.

[Cao07] Y. Cao. and L. T. Clark, "Mapping statistical process variations toward circuit performance variability: an analytical modeling approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1866-1873, Feb. 2007.

[Chandrasakan92] A. P. Chandrasakan, S. Sheng, and R. W. Broderson, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–483, Apr. 1992.

[Chandrakasan01] A. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*, Wiley-IEEE Press, 2000.

[Chatterjee03] B. Chatterjee, M. Sachdev, S. Hsu, R. Krishnamurthy, and S. Borkar, "Effectiveness and scaling trends of leakage control techniques for sub-130 nm CMOS technologies," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, ISLPED'03*, Aug. 2003, pp. 122–127.

[Chae08] M. Chae, W. Liu, Z, Yang, T, Chen, J. Kim, M. Sivaprakasam1, and M. Yuce, "A 128-channel 6mW wireless neural recording IC with on-the-fly spike sorting and UWB transmitter," in *2008 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2008, pp. 146-147.

[Dennard74] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFETs with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 10, pp. 256–268, Oct. 1974.

[Eisele96] M. Eisele, J. Berthold, D. Schmitt-Landsiedel, and R. Mahnkopf, "The impact of intra-die device variations on path delays and on the design for yield of low voltage digital circuits," in *Proceedings of the 1996 International Symposium on Low Power Electronics and Design*, Aug. 1996, pp. 237-242.

[Ghosh10] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: new design paradigm for the nanoscale era," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1718-1751, Oct. 2010.

[Gibson08] S. Gibson, J. W. Judy, and D. Marković, "Comparison of spike-sorting algorithms for future hardware implementation," in *30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS 2008*, Aug. 2008, pp. 5015–5020.

[Gibson10] S. Gibson, J. W. Judy, and D. Marković, "Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 4, pp. 469–478, Oct. 2010.

[Hanson08] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, and D. Blaauw, "Exploring variability and performance in a sub-200mV processor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 881-891, Apr. 2008.

[Ickes12] N. Ickes, G. Gammie, M. E. Sinangil, R. Rithe, J. Gu, A. Wang, H. Mair, S. Datla, B. Rong, S. Honnavara-Prasad, L. Ho , G. Baldwin, D. Buss, A. P. Chandrakasan, and U. Ko, "A 28nm 0.6 V low power DSP for mobile applications," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 35-46, Jan. 2012.

[ITRS09] The International Technology Roadmap for Semiconductors (ITRS) 2009 Edition, Technical Report, http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[Jeon12] D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A super-pipelined energy efficient subthreshold 240 MS/s FFT core in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 23-34, Jan. 2012.

[Karkare11] V. Karkare, S. Gibson, and D. Marković, "A 130-μW, 64-channel neural spike-sorting DSP chip," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 5, pp. 1214-1222, May, 2011.

[Kurd09] N. Kurd, P. Mosalikanti, M. Neidengard, J. Douglas, and R. Kumar., "Next generation Intel core micro-architecture (Nehalem) clocking," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1121-1129, Apr. 2009.

[Lotze12] N. Lotze and Y. Manoli, "A 62 mv 0.13 um CMOS standard-cell-based design technique using schmitt-trigger logic," I*EEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 47-60, Jan. 2012.

[Marković10] D. Marković, C. C. Wang, L. Alarcón, T.-T. Liu, and J. Rabaey, "Ultra-low power design in near-threshold regime," *Proceedings of the IEEE*, vol.98, no.2, pp. 237–252, Feb. 2010.

[Muller12] R. Muller, S. Gambini, and J. M. Rabaey, "A 0.013mm2, 5μW, DC-coupled neural signal acquisition IC With 0.5 V Supply," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 232-243, Jan. 2012.

[Muttersbach00] J. Muttersbach, T. Villiger, and W. Fichtner, "Practical design of globally-asynchronous locally-synchronous systems," in *2000 16th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, Apr. 2000, pp. 52–59.

[Narendra01] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, "Scaling of stack effect and its application for leakage reduction," in *Proceedings of the 2001 International Symposium on Lower Power Electronics and Design, ISLPED'01*, Aug. 2001, pp. 195–200.

[Nowak02] E. J. Nowak, "Maintaining the benefits of CMOS scaling when scaling bogs down," *IBM Journal of Research and Development*, vol. 46, no. 2.3, pp. 169–180, Mar. 2002.

[Pang09] Liang-Teck Pang and B. Nikolic, "Measurements and analysis of process variability in 90 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 5, 1655–1663, May 2009.

[Pierson07] M. D. Pierson, *Automated design for current-mode pass-transistor logic blocks*, Master of Science, University of California, Berkeley, May 2007.

[Rabaey03] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed, Prentice-Hall, 2003.

[Rabaey08] J. Rabaey, "A brand new wireless day" in *Asia and South Pacific Design Automation Conference, ASPDAC 2008*, Mar. 2008, Keynote Address.

[Rabaey11] J, M Rabaey, M. Mark, D. Chen, C. Sutardja, C. Tang, S. Gowda, M. Wagner, and D. Werthimer, "Powering and communicating with mm-size Implants," in *2011 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Mar. 2011, pp. 1-6.

[Sakurai90] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584-594, Apr. 1990.

[Sakurai03] T. Sakurai, "Perspectives on power-aware electronics," in 2003 *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2003, pp. 26–29.

[Sparsø01] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design*, Springer, 2001.

[Stevens03] K. Stevens, R. Ginosar, and S. Rotem, "Relative timing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 129–140, Feb. 2003.

[Stevens11] K. Stevens, P. Golani, and P. Beerel, "Energy and performance models for synchronous and asynchronous communication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 3, pp. 369–382, Mar. 2011.

[Sutherland89] I. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.

[Taur98] Y. Taur and T. Ning, *Fundamentals of Modern VLSI Devices*, Cambridge University Press, 1998.

[Wang05] A. Wang and A. Chandrakasan, A 180-mV subthreshold FFT processor using a minimum energy design methodology, *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[Williams94] T. Williams, "Performance of iterative computation in self-timed rings," *The Journal of VLSI Signal Processing*, vol. 7, no. 1/2, pp. 17–31, Feb. 1994.