On-chip Surfing Interconnect

by

Suwen Yang

B.Eng., Huazhong University of Science and Technology, 1995
M.Sc., The University of Washington, 2001
M.Sc., The University of British Columbia, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

April, 2010

© Suwen Yang 2010

Abstract

With growing chip sizes and operating frequencies, on-chip global interconnect has become a critical bottleneck for CMOS technology. With processes scaling into deep submicron scales, the gap between gate delay and global-interconnect delay increases with each technology generation.

Bandwidth is also important for on-chip interconnect and is limited by skew and jitter. Due to temperature variation, crosstalk noise, power supply variation and parameter variation, timing variation increases with the length of global interconnect lines. Jitter and skew in the transmitter and receiver's clocks add timing variation to on-chip interconnect communication. Repeaters in a buffering technique amplify clock jitter and drop pulses due to intersymbol interference. Latches can be inserted in place of some of the buffers to control the timing variation. However, these latches increase latency and power consumption.

In 2002, a novel circuit technique called "surfing" was proposed to bound the timing uncertainty in wave pipelines [57]. This thesis extends the application of surfing to on-chip interconnects and introduces surfing RC interconnect and surfing LC interconnect techniques.

For RC interconnects, we present a jitter attenuating buffer. This buffer uses inverters with variable output strength to implement a simple, low-gain DLL. Chains of these surfing buffers attenuate jitter making them well suited for source-synchronous interconnect. Furthermore, our chains can be used to reliably transmit handshaking signals and support sliding-window protocols to improve the throughput of asynchronous communication.

We use distributed varactors to dynamically vary the latency of LC interconnects and thus effect surfing. Different from RC signaling, signals on LC interconnect propagate at nearly the speed-of-light. The varactors not only modulate the line latency, but also sharpen the edges of signals. We present both a full-swing and a low-swing LC interconnect designs. In both interconnects, the jitter and skew are attenuated along the line due to the surfing effect. In the low swing interconnect, the surfing effect also helps to reshape the pulses to increase the eye height. To demonstrate these techniques in real silicon, we designed, fabricated and tested a chip. The testing

results show that surfing LC interconnects are promising for deep submicron technology.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Without extensive support, discussion and endless encouragement from Dr. Mark Greenstreet, this work would not have been possible. Most importantly, Dr. Mark Greenstreet tries every possible way to help me to enjoy this work and make my stay at Vancouver to be such a wonderful time. He and his family, Susan Greenstreet, Laura Greenstreet and Jenny Greenstreet impress me a lot by their sense of humor to the world.

I would not have completed the chip design without the support and help from the member of VLSI research group in SUN Microsystems Laboratory. Special thanks goes to Dr. Robert Drost, who triggers the idea of varactors used in the surfing interconnect, Alex Chow for a lot of discussions on Varactors and line parameters and helps on chip testing, Justin Shauer, Jon Lexau and Jon Gainsley who helps me a lot on the chip layout. I owe the students and staff in the UBC SOC lab a debt of gratitude, especially Roozbeh Mehrabadi and Roberto Rosales for helping me with the chip simulation and testing. Without the technical support from Dave Brent and Sean Godel, it would be impossible to complete the layout at the University of British Columbia.

Finally, I would like to thank my husband Zhenyu Zhang who encourages me a lot to make this work interesting.

Dedication

To my husband, Zhenyu Zhang for his endless support and patience.

Chapter 1

Introduction

Ongoing innovations in desk top computers, graphics processors, high-speed networks, mobile computation and communication devices, and embedded processors in everything from automobiles to medical devices to children's toys have contributed to nearly all aspects of our lives. This innovation is largely driven by relentless advances in fabrication technology. Smaller transistors and wires enable chips with ever increasing performance and functionality. For example, the first microprocessor, the Intel 4004 was implemented in a process with 10μm gate lengths; today, Intel produces processors in a 45nm process [27]. As transistors have gotten smaller, dies have grown larger. The compounded effect of these two trends is an increase in the number of transistors on a chip by nearly a factor of a million from the first microprocessors to today. The same improvements in fabrication technology that enable new electronic projects also create immense design challenges including managing the complexity of large designs, power dissipation and interconnect. This thesis addresses issues of designing efficient on-chip global interconnect. This focus is driven by the observation that chips are mostly wires: while a chip has one layer of transistors, it may have ten or more layers of wiring. Furthermore, when transistors shrink, logic gates get faster, but when wires shrink, the interconnect gets slower. This growing gap between logic gate and interconnect performance has made interconnect a critical limiter of chip performance. This chapter starts with an overview of on-chip interconnect methods. Section 1.2 describes surfing pipelines, a novel pipelining technique that I will apply to on-chip interconnect in this thesis. From these, Section 1.3 identifies controlling timing uncertainty as a critical issue for interconnect, and Section 1.4 gives my thesis statement and list of contributions describing my new approach based on surfing to address the problems of timing uncertainty.

1.1 Overview of On-chip Interconnect

This section gives an overview of on-chip interconnects. We first examine how line delay, power consumption and other performance scale with

technologies in section 1.1.1. Section 1.1.2 describes the impact of timing uncertainty on interconnect. Section 1.1.3 presents different timing conventions: synchronous, self and source synchronous and asynchronous. At low frequencies, the delay of a wire is determined by the product of the resistance and capacitance of the wire (the RC mode). At higher frequencies, the delay is dominated by the square root of the product of the inductance and capacitance of the wire (the LC mode). The LC mode can be *much* faster than the RC mode. Section 1.1.4 compares RC and LC mode signaling for on-chip communication.

1.1.1 Scaling Issues

An interconnect is characterized by line resistance, capacitance and inductance. Figure 1.1 shows the trend of line resistance at each generation. In this figure, we present the line resistances R_{wire} with conservative and aggressive scaling parameters. Table 1.1 only lists the conservative scaling parameters. The line resistance depends on the wire thickness and width which decrease as the technology scales down. Although the thickness for the global interconnect decreases more slowly than the width, the line resistance continues to increase at each generation. As the technology scales below 65nm, wires width becomes a small multiple of the mean free path of electrons which is roughly 40nm [48] for copper at room temperature and the scattering effect exacerbates the trend of growing wire resistance. In Table 1.1 and Figure 1.1, we see a significant increase in the line resistance beyond 65nm. Line capacitance and inductance do not change much at each generation. Although the spacing between metal layers decreases, capacitance per micron is dropping but will hit a low limit of low-k dielectrics as shown in Figure 1.2. Note that in fabrication processes for processors, the effective gate length has been about half of the drawn gate length for processes down to 65nm. For these processes, the FO4 delay is about half of the value given in Table 1.1 and the clock frequency is doubled. For processes smaller than 65nm, the gates are being undercut less or not at all for reasons of leakage, manufacturing capability and power. Furthermore, power dissipation constraints have also limited increases in clock frequency. Thus, the clock frequency for microprocessors has stayed roughly the same from the 90nm process down to 45nm process.

Current designs use RC-mode signaling where interconnect delays are proportional to the product of line resistance and capacitance and thus proportional to square of line length. With continuously decreasing transistor sizes, more functional units can be incorporated into a single chip and

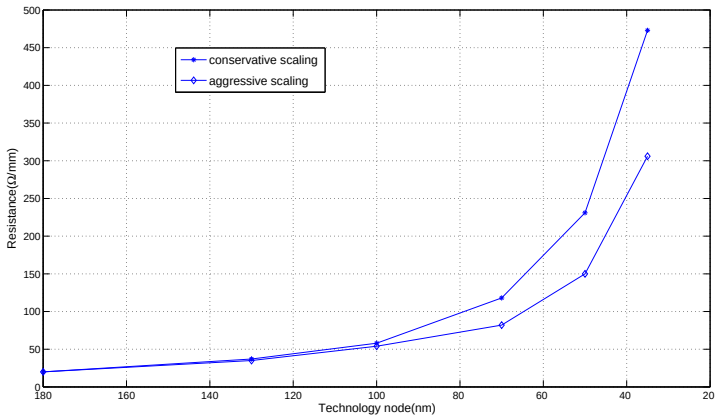


Figure 1.1: Resistance scaling of interconnects (from [26], Figure 14).

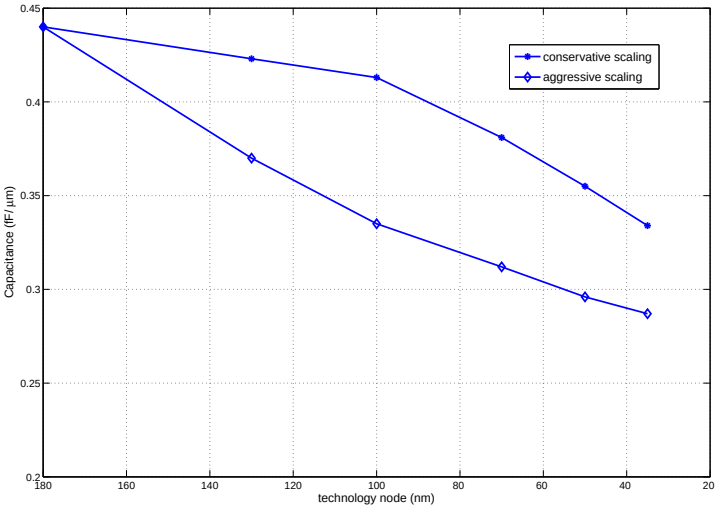


Figure 1.2: Capacitance scaling of interconnects (from [26], Figure 15).

Table 1.1: Interconnect parameters under technology scaling [26].

Technology node (nm)	180	130	100	70	50	35
Chip edge (mm)	19	20.7	22.8	24.9	27.4	30.1
frequency (GHz)	0.7	1	1.25	1.8	2.5	3.6
FO4 (ps)	90	65	50	35	25	17.5
latency(ps/mm)	50	56	61	70	80	93
global aspect ratio	2.2	2.3	2.5	2.5	2.5	2.5
sideway ϵ_r	3.75	3.375	3.038	2.734	2.460	2.214
resistance (Ω/mm)	20	37	58	118	231	473
capacitance (fF/mm)	440	423	413	381	355	334

the number of modules per chip increases exponentially. The chip size as shown in Table 1.1 increases around 10% with each fabrication generation. Although local (within functional units) and semi-global (between a functional unit and its neighbors) wires scale with the feature size [26, 52], global wires (crossing several functional units) with increasing length do not. Figure 1.3 [68] shows the delay trend of global interconnects in microprocessors based on data from the ITRS roadmap [28]. Figure 1.3 also demonstrates that ignoring the scattering effect under 60nm will cause an underestimation of the delay. The gap between the interconnect latency and FO4 delay (fan-out-of-four inverter delay, a "typical" gate delay) increases with each technology generation. The FO4 delay continues to decrease with each technology node. However, as shown in Figure 1.4, the delay of a 10mm long interconnect relative to inverter FO4 delay increases by roughly 2.5X with each technology generation. Breaking long wires into many short segments with repeaters between segments, *i.e.* repeated wires, mitigates the gap between the interconnect latency and FO4 delay. As shown in Figure 1.4, the repeated propagation delay over FO4 delay increases by 1.5X with each generation. In other words, the propagation speed of repeated global wires is roughly 60ps/mm and is roughly constant across technology generations. As illustrated by Figure 1.5, the reachable distance during one clock cycle decreases significantly when the technology moves into the deep submicron region. This decreasing reachable distance was mainly due to the increasing clock frequency. As noted above, clock frequencies have stayed roughly constant since the 90nm generation (after [26] was written). Thus, the reachable distance is also leveling off for sub 90nm processes. However, due to the increasing chip sizes, the gap between the chip size and reachable dis-

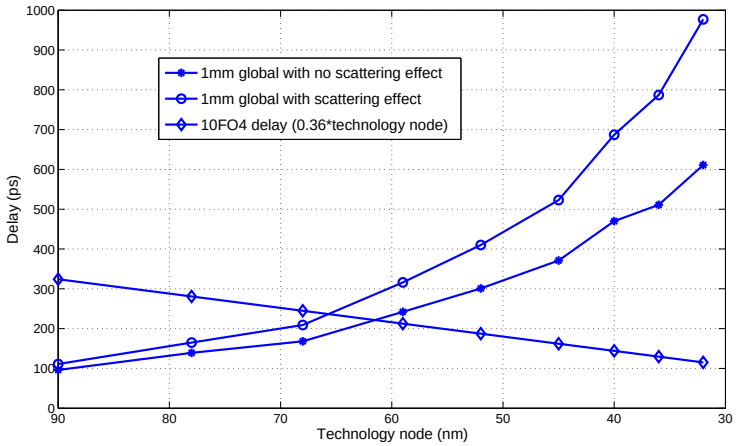


Figure 1.3: Delay trend of global interconnects.

tance increases slightly at each generation node. Thus, interconnects are an ongoing, critical bottleneck for modern integrated circuit design.

Buffers are commonly inserted into the interconnect to reduce interconnect delay. It is well-known [6, 19, 32, page 359] that minimal delay is achieved when wire delay matches buffer delay and wire capacitance matches buffer capacitance. Because power consumption grows linearly with total capacitance, using delay minimizing buffering roughly doubles the power consumption of global interconnect [33]. While this can be improved by modest factors by sizing buffers and wires to minimize the energy-delay product or the energy with a delay constraint [17, 33], power consumption remains a critical issue for long-wire interconnect. The power consumption contains three parts: dynamic power, static power and short circuit power. Dynamic power consumption occurs when charging the capacitive loads of the gates. The dynamic power consumption increases proportionally with chip area [52]. As shown in Figure 1.5, the distance that a signal can propagate within one clock cycle decreases with technology scaling and chip size increases. Thus deeper and deeper pipelining is needed. This further increases the dynamic power consumption. Static power consumption is due to the leakage current which is an exponential function of threshold voltage. As the technology moves into deep submicron region, leakage power consumption becomes a larger component of the total power consumption. Short circuit power occurs when the pull-up stack and pull-down stack of a

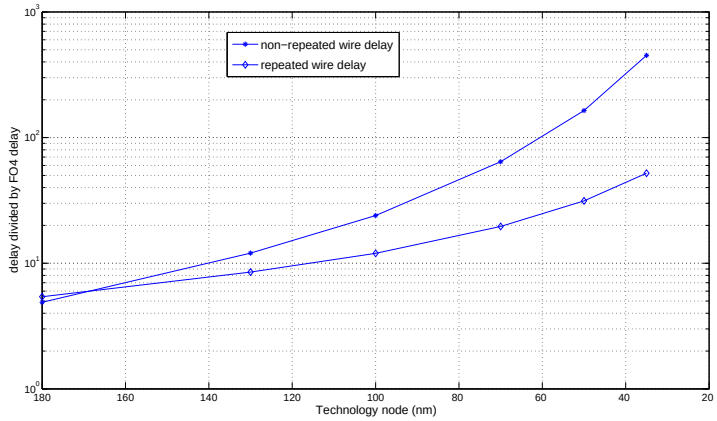


Figure 1.4: Non-repeated and repeated wire delays for 10mm long wires (from [26], Figure 18).

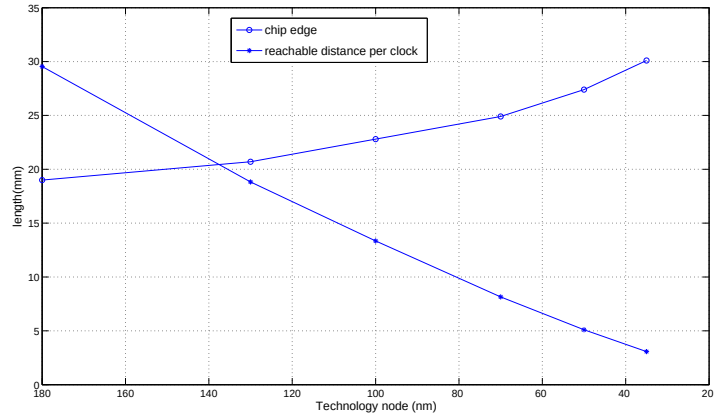


Figure 1.5: Reachable distance within one clock for repeated wires (from [26], Figure 20).

CMOS gate are on simultaneously. Research in this area demonstrates that a well-designed wire buffer has short-circuit power that is about 20% of the dynamic power [54].

1.1.2 The Impact of Timing Uncertainty

Bandwidth is an important figure of merit for on-chip interconnect and is limited by skew and jitter, which are caused by temperature variation, crosstalk noise, power supply variation and parameter variation. These variations can be classified as intra-die, inter-die systematic or random as shown in Table 1.2. Inter-die systematic variation includes channel length variation, mean threshold voltage and mean metal capacitance and resistance. The mean values of these parameters vary within the normal manufacturing tolerances. Intra-die systematic variation is caused by layout specific variations due to semiconductor process methods or environmental differences. Optical proximity effects cause effective gate lengths to change as a function of local layout. Local wire densities affect the inter-layer dielectric thickness. Stepper-induced illumination and imaging nonuniformity due to lens aberration [50, 67] result in systematic spatial intrachip variability of the effective gate length which affects the distribution of the maximum frequency at which a VLSI chip can run [9, 47]. Environmentally introduced variation includes power supply and temperature variations across the chip. This variation is related to the switching activity of the chip. Cross-talk noise due to coupling capacitance between wires results in large delay variation. Coupling capacitance has been kept about 75% percent of the total capacitance across technology generations. In the worst case where neighboring wires switch in opposite directions, the coupling capacitance can cause an 80% increase in the wire delay. Designing for the worst case delay severely decreases the bandwidth of the interconnect. Furthermore, repeaters amplify jitter [7, 64] due to the intersymbol interference effect (aka "drafting" [56]). Jitter and skew in the transmitter and receiver's clocks further add timing variation to on-chip interconnect communication. All of these contribute to the timing variation for global interconnect.

The impact of timing uncertainty can be characterized using an eye diagram [66, chapter 2.8.2]. An eye-diagram is obtained by overlaying the signal for multiple bit-periods to show the high and low portions along with the possible transistions as shown in Figure 1.6. A robust design requires enough eye height H to distinguish different symbols and enough eye width W to accommodate timing uncertainties in the system. Timing uncertainty shrinks the eye width and limits the bandwidth of the system. More impor-

Table 1.2: Components of delay variation [69].

component	form of variation
channel length	inter-die systematic
	intra-die systematic
	Intra-die random
mean threshold voltage difference between device types	inter-die systematic
threshold voltage	inter-die systematic
	intra-die random
mean metal R and C differences between metal levels	inter-die systematic
voltage and temperature	intra-die systematic
NBTI, hot-e	intra-die systematic
cross-talk noise (Miller capacitance)	intra-die random

tantly, timing variation increases with the process technology. With smaller feature size, more switching activity occurs per mm^2 at each generation. To reduce leakage current, the power supply voltage does not scale by the same factor as the transistor gate length. Thus, current densities and cross-talk increase. This reduces eye height.

Global wires are ones whose lengths are comparable to the diameter of the chip. Various studies have shown that the number of global wires grow slowly across technology generations. Because the delay per mm for wires is growing due to smaller wire cross section and thus increased resistance, the latency of global interconnect is increasing and the bandwidth of global wires drops unless pipelining is used. Thus, global wires make increasing use of repeaters and pipeline latches or flip-flops. To achieve bandwidths that scale with chip clock frequencies, extra design effort can be expanded. To reduce the timing uncertainty of the global interconnect, extra design effort is needed to reduce clock skew and jitter. This can lead to a complex clock network with high power consumption. Latches can be inserted in place of some of the buffers to control the timing variation. However, these latches increase latency and consume extra power.

In conclusion, timing uncertainty affects latency, power consumption and bandwidth which are major concerns for interconnect design. In this thesis, we propose a novel interconnect design to achieve enough eye height and width.

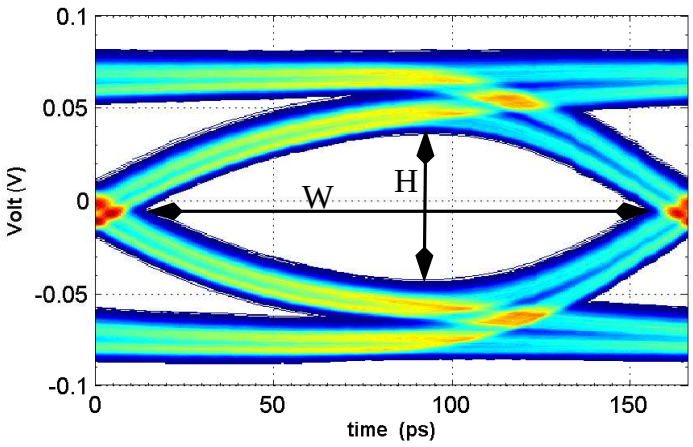
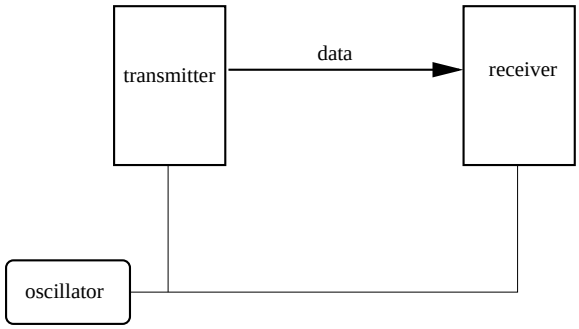


Figure 1.6: An example of an eye diagram [46].

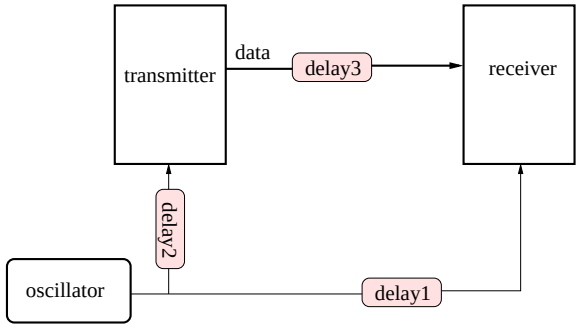
1.1.3 Synchronous and Asynchronous Signaling

Global interconnect can have latencies greater than a clock period. Thus, some method is required so that the receiver can distinguish successive data values sent by the transmitter. These signaling methods can be synchronous or asynchronous. This section describes commonly used signaling methods with an emphasis on their suitability for on-chip, global interconnect.

Part a of Figure 1.7 shows the block diagram for a typical implementation of globally synchronous communication. Both the transmitter and receiver are clocked by a global clock, and the transmitter only sends data to the receiver without any requirement for a reverse channel (although such a channel may be used for high-level flow control). Part b of Figure 1.7 shows the timing uncertainty that we must consider to ensure reliable operation. The skew in the global clock network and timing variation in the interconnect between transmitter and receiver both contribute to the timing uncertainty of the system. For large chips, it becomes difficult to design a low skew clock network. Thus, this method can be very expensive. Furthermore, SoC designs use many IP blocks which may operate at different frequencies. This makes the globally synchronous approach impractical for many designs.



(a) Block diagram

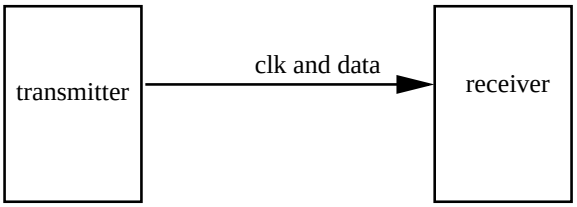


(b) Timing details

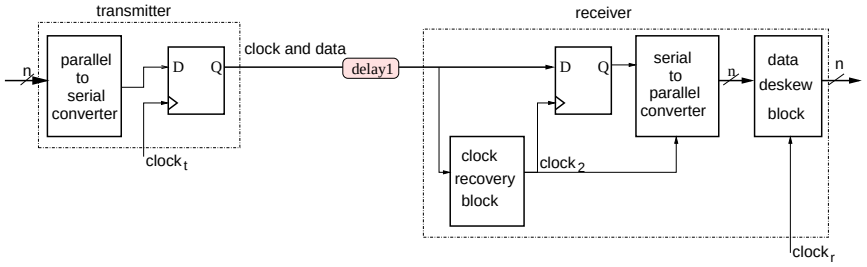
Figure 1.7: Globally synchronous communication.

In a self-synchronous design as shown in Figure 1.8, the data stream contains data and clock information, for example, 1-out-of 4 [5] and 8-out-of-10 [19, page 378]. Part b of Figure 1.8 shows a more detailed schematic for this method. This approach is often used for serial communication with limited area. Accordingly, the transmitter has a parallel-to-serial converter and the receiver has a serial-to-parallel converter [29, 37, 39, 44]. The receiver has a clock recovery block to generate a clock, $clock_2$, with frequency and phase matching those of the received data. Normally, the clock recovery block is implemented with phase-locked loops, and such circuits are relatively large, power intensive and require sophisticated analog designs [12, 34, 63]. The clock recovery block uses the historical clock information of the receiver to generate $clock_2$. Therefore, $clock_2$ may be out of phase with the transmitter clock $clock_t$ due to jitter in the clock oscillators, network and link. The coding scheme for embedding the clock must provide enough phase information

for the clock recovery circuit to adequately track the sender's clock. This coding lowers the throughput of the link. After deserialization, the data is sent through the deskewing block to resynchronize to receiver's clock $clock_r$.



(a) Block diagram

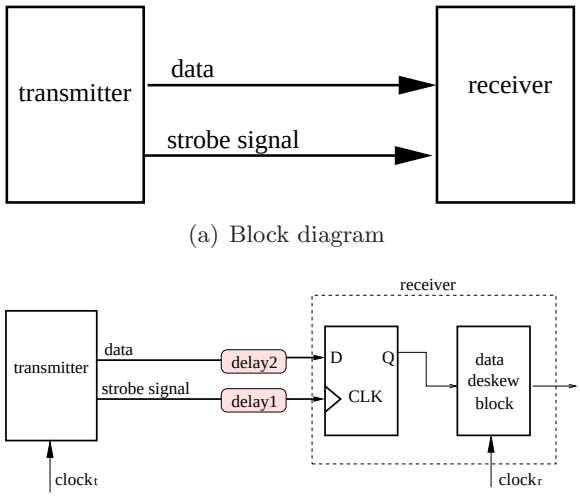


(b) Timing details

Figure 1.8: Self-synchronous communication.

Source-synchronous signaling (see Figure 1.9) is better suited for parallel communication. The transmitter sends a strobe signal along with the data to the receiver. The sampler uses this strobe signal to process the data, and the data deskew block conveys the data from the received clock domain to its own clock domain $clock_r$. As shown in part b of Figure 1.9, the sampler design on the receiver side only needs to consider the timing variation between the data and strobe signal. Source-synchronous design simplifies global timing closure because it does not require bounding the relative skew between the transmitter and receiver. However, the routing must ensure that the delays for the data and strobe are well-matched. By sending these signals on adjacent wires, this delay matching can be achieved by local analysis of each segment of the line. The jitter of $clock_r$ adds extra timing uncertainty to the source synchronous communication system which must be handled by the deskew circuitry.

Several researchers have examined asynchronous handshaking [5, 25, 40] for on-chip communication. The transmitter sends a **request** to the receiver



(a) Block diagram

(b) Timing details

Figure 1.9: Source synchronous communication.

either embedded in the data [5, 40] or along a delay-matched line with the data [25]. The receiver sends an **acknowledgment** back to the transmitter to confirm receipt of each datum. The transmitter does not send the next datum until it gets an **acknowledgment** from the receiver. Asynchronous handshaking relies on the handshake protocol instead of the clock signal used by the synchronous methods. Thus, this approach does not need to consider the jitter in the clock network or global wires, and issues of global timing closure are avoided. The major disadvantages are the overhead of the handshaking protocol where the round trip delay limits the throughput of the system. The need to interface these asynchronous circuits with functional units on the chip which are typically designed using synchronous techniques also hinders its widespread usage. Due to the wide support from the CAD tools, synchronous communication is easier to design than the asynchronous communication. Thus, synchronous communication remains the main stream for on-chip communication.

This thesis focuses on source-synchronous communication which only requires local matching of delays between the strobe line and data line. As we will discuss later, surfing is very well suited for this communication method.

1.1.4 Signaling Methods

At low frequencies, the wire delay is dominated by the RC mode. However at high frequencies, the wire delay is determined by the LC mode. This subsection introduces the advantages and disadvantages of RC and LC signaling.

The conventional approach to reducing the interconnect delay is buffer insertion [19]. Without buffers, the delay of a RC line grows quadratically with the wire length. Buffers can be inserted to divide a line into several segments. Thus the total delay is the sum of delays on each segment and grows linearly with the total line length. Figure 1.10 plots the delay and power consumption versus the number of stages for a given wire which is $0.44\mu m$ wide, $0.46\mu m$ spacing and 20mm long. I performed HSPICE simulations using parameters for a 0.18μm technology. These show that the rise time decreases with more repeaters in the wire. As noted in [15], the power consumption decreases slightly at first because the short circuit power decreases as the rise time decreases. The power consumption increases dramatically as the number of repeaters increases above the number where dynamic power dominates. Furthermore, the delay climbs when the inverter intrinsic delay dominates the wire delay. Thus, this design has optima for minimizing power consumption and delay. With RC signaling, jitter accumulates along the chain. Even if all of the inverters are of the same design and all of the wires are of the same length, width and spacing, random variations due to power-supply noise, crosstalk, temperature variation and intra-chip parameter variation add jitter at each stage, and this jitter is cumulative. Furthermore, intersymbol interference (ISI) effects amplify jitter [7]. These two effects, the random walk of edge timing combined with the jitter amplification of intersymbol interference, will cause a sufficiently long buffer chain to drop pulses even when operating at low clock frequencies. Phase-locked loops (PLLs) and delay-locked loops (DLLs) overcome the limitations of simple buffer chains by actively compensating for jitter. Unfortunately, these circuits require much more power and layout area than simple inverters. Thus, they are not practical in most on-chip applications, to forward a clock for a source-synchronous link.

An attractive alternative, LC signaling using transmission lines has received considerable attention in recent years [15, 16, 18, 55, 60, 68]. On-chip wires are distributed, rlc lines (interline conductance is negligible) where r, l and c are line resistance, inductance and capacitance per unit length. At low frequencies, the effects of wire resistance dominate those of wire inductance, and the wire can be accurately approximated as an rc transmission

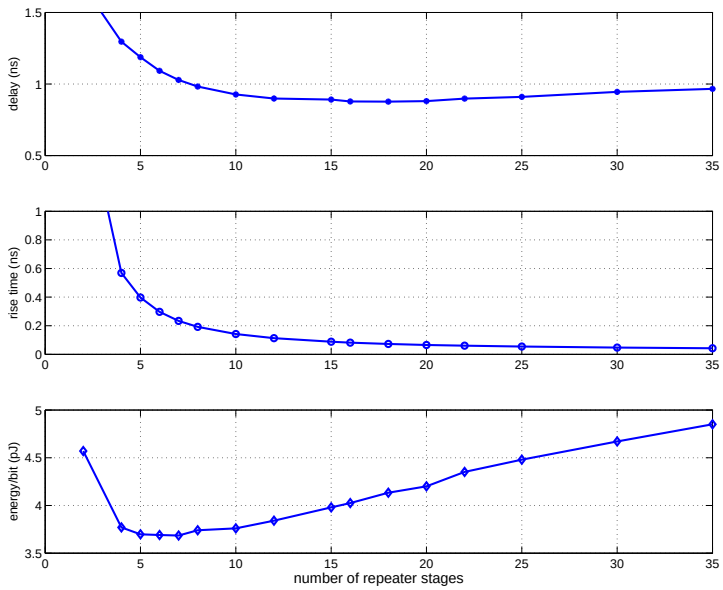


Figure 1.10: Performance of a 20mm long interconnect with repeaters.

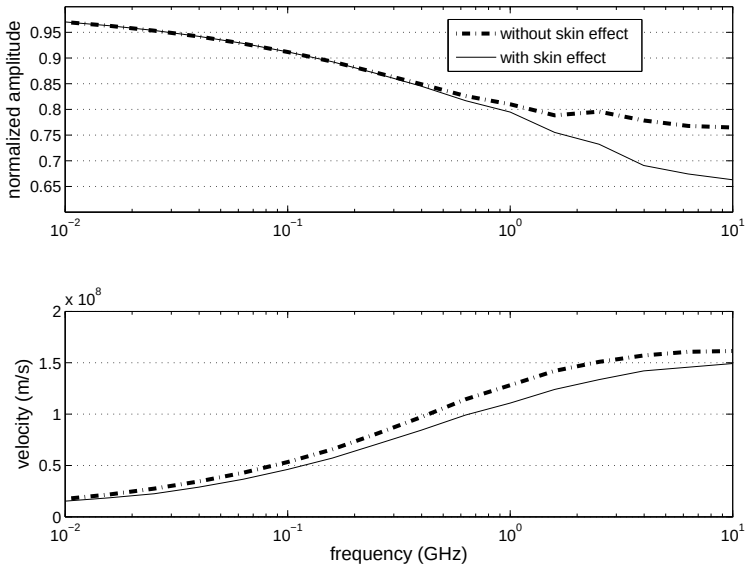


Figure 1.11: Velocity and attenuation for a 10mm long, 8μm wide transmission line.

line. Data is propagated in such lines according to the diffusive mode of the line, and wire delay grows quadratically with length. At high frequencies, the effects of wire inductance become dominant, and the wire can be approximated as a lossy transmission line where the velocity is approximately $1/\sqrt{lc}$, the speed-of-light in the dielectric, and attenuation is exponential at a rate of $\frac{r}{2Z_0}$ where $Z_0 = \sqrt{l/c}$ is the characteristic impedance of the corresponding lossless line. LC transmission lines are attractive for two reasons. First, wave propagation in the high frequency LC mode is faster than the diffusive propagation in the low frequency RC mode. Second, wave propagation can consume much less power, because there is no need for the driver to charge the entire wire during the transmission of the symbol. To achieve lc behaviours, the high frequency component must dominate the low frequency ones and designers have proposed various schemes to exploit the high-frequency, near speed-of-light response of the lines. However, the line resistance introduces distortion into the transmitted symbols. As shown in Figure 1.11, the velocity and attenuation rate depend strongly on frequency. The velocity dispersion causes the intersymbol interference and limits the bandwidth of the interconnect. The high attenuation rate in the high frequency regime limits the length of the interconnect and complicates the design of the receiver. The skin effect resistance reduces the propagation speed and makes the attenuation even worse. Techniques such as nonlinear transmission lines [2], the "surfliner" technique [68] and short pulse signaling [29, 55] have been proposed to tackle the velocity dispersion problem. For these kind of designs, repeaters in series must be added in for long distance communication. However, this introduces extra delay and power consumption that are not considered in the paper cited above. Distributed amplifiers in a shunt configuration [30, 60] can provide active compensation to reduce the attenuation rate. In the dielectric, the propagation speed of LC signaling is relatively insensitive to power supply noise and temperature variation. Although it is reasonable to expect that the relative jitter will be smaller than that in RC signaling, jitter still exists due to random variations caused by line mismatches and crosstalk noise.

RC signaling is simple. However, it suffers from intersymbol interference and jitter accumulation. More importantly, RC lines may become the performance bottleneck for many designs because of their high latency. On the other hand, LC signaling needs distributed amplifiers for long distance communication to overcome the attenuation of the interconnect. LC signaling also suffers from intersymbol interference because of velocity dispersion. The big advantage of LC signaling is that it can in theory reach the speed-of-light in the medium. LC methods typically use wide and thick interconnect

to reduce the resistance on the line. LC signaling also requires wide spacing between wires to increase the inductance in the line. Larger inductances increase the characteristic impedance which reduce the attenuation rate. However, the use of wide wires with large separation means that LC signaling uses more area than RC signaling and lowers the bandwidth per unit crosssection.

Optical interconnect [43] and RF/wireless interconnect [14] have also been proposed to tackle the problem of interconnect performance in deep-submicron technologies. Optical designs require conversion of signals between the electrical and optical domains which is cumbersome because silicon is not an optically active material. RF/wireless interconnect requires RF/wireless carrier. The overhead for these methods is more complicated transmitter and receiver designs. This thesis focuses on wire (i.e. RC and LC) signaling.

1.2 Surfing

Surfing pipelining [57] is a novel technique to address the timing variations in the pipeline to improve its throughput. This thesis explores the application of this technique to address the timing issues of on-chip interconnects. This section gives a brief introduction of surfing pipelines.

The key idea in surfing is to implement a pipeline of stages that have variable delays. A timing pulse is propagated along with data, and each stage has a lower delay when the timing pulse is asserted than when the timing pulse is not asserted. Under conditions described in Section 2.1, this ensures that events in the data path are attracted to coincide with the rising edge of the timing pulse. Surfing provides an active mechanism for mitigating jitter and other timing uncertainty. I explore the applications of surfing pipelines for on-chip, global interconnect in the remainder of this thesis.

1.3 Problem Statement

For both RC and LC signaling, jitter in the interconnect is inevitable due to noise disturbances. Jitter brings timing uncertainty into the system and is difficult to quantify. Underestimation of the jitter will lead to unreliable designs with narrow eye width. Overestimation of the jitter results in a conservative design which sacrifices the throughput.

For RC interconnect, different edges may propagate at different rates due to noise, ISI, and other sources of timing uncertainty. If this causes a rising and falling edge to become too close to each other, then the pulse may be dropped. Thus, timing uncertainty causes a loss of eye-width and eye-height for RC interconnect. For LC interconnect, the main concern is high-frequency loss and dispersion. Losses cause a loss of eye-height, and dispersion causes ISI and degrades eye width. For both the RC and LC designs, further improvements in on-chip interconnect will require finding new methods to reduce timing uncertainty to maintain eye width. LC methods also require techniques to compensate for high frequency losses to maintain eye height.

1.4 Thesis Statement

Surfing provides an efficient solution for the timing problems of on-chip interconnect. This approach reduces the complexity of the receiver and is robust to noise and other disturbances.

In particular, this thesis explores the surfing technique for on-chip interconnect for both RC and LC signaling. This thesis makes contributions in the following areas:

- For the surfing RC Line, we proposed a novel and simple surfing delay-locked loop in Chapter 3. This surfing delay-locked loop (DLL) consists of surfing buffers. The timing chain, a chain of surfing delay-locked loops, is able to attenuate jitter and is robust to process, power supply and temperature variation.

- We show how this surfing DLL can be used to implement robust and high performance source-synchronous and asynchronous on-chip interconnects. These designs were validated by analytical models that we derive and by HSPICE simulations.

- In Chapter 4, we proposed surfing LC interconnect which has much higher velocity than RC signaling.

- Varactors presented in Section 4.3 can be used efficiently to implement surfing LC lines.

- Varactor can also increase the eye-height of the received signal by reshaping pulses to partially compensate for dispersive losses of long lines.

- In Chapter 5, we demonstrated the surfing LC line with a test chip. This shows the effectiveness of varactors for reshaping pulses. The jitter reduction effects were less compelling. Thus, we see that this work shows the potential of using surfing and varactors for global interconnect, but we recognize that further design work will be needed to bring these methods into practical application.

1.5 Thesis Organization

The thesis is organized as follows:

- Chapter 2 summarizes related work for interconnect signaling techniques.
- Chapter 3 presents surfing RC interconnect.
- Chapter 4 presents surfing LC interconnect.
- Chapter 5 presents the chip to demonstrate that surfing LC interconnect works in real silicon.
- Chapter 6 concludes the thesis.

Chapter 2

Background

In this chapter, I provide an overview of related work for RC signaling, circuits applicable to surfing techniques and LC signaling and a summary of my M.Sc. work.

2.1 Related Work for RC Signaling

Buffer insertion is well understood [1, 6, 8, 17, 33, 45]. The design task can be represented as an optimization problem with power, delay, and bandwidth being three common objectives. The research literature in this area is extensive, and this section describes a few representative papers. Typically, on-chip interconnect has stringent bit-error-rate requirements, so we describe previous work that examines the trade-offs between throughput and robustness for source synchronous interconnect. This identifies skew and jitter as key concerns which using surfing methods that are examined in more detail at the end of this section.

In Chen and Friedman's paper [17], the optimization variables are the number and size of buffers assuming a fixed ratio between PMOS and NMOS transistor widths. By using empirical equations for power, delay and bandwidth, the authors developed closed-form solutions to minimize power with delay and/or bandwidth constraints. They showed that the best solution was always at the edge of the design space. The authors also compared the optimized solution with that from SPICE. With only delay constraints and an objective of minizing power consumption, the average error between analytic results and HSPICE simulation was less than 7%. With only bandwidth constraints, power consumption is minimized by a solution that only uses minimum sized inverters. The authors also investigated the effect of inductance. They showed that the line delay and power consumption are non-monotonic with inductance. In this paper, the authors assumed that the bandwidth is at least 0.4 times of the inverse of the 10% to 90% signal transition time. However, as described in Chapter 1, intersymbol interference, crosstalk and other disturbances limit bandwidth to values below their approximation. The paper does not investigate the robustness of the

optimization results to intersymbol interference, crosstalk noise, power supply variation, parameter variation, temperature variation and so on. These introduce timing uncertainty into the system. Latches may be added to control these timing uncertainties. However, such latches would increase latency and power consumption.

While buffer insertion is well-understood and widely used, timing uncertainty remains a critical issue which limits the throughput of the design. In this thesis, I will use a variant of wave pipelining, the surfing technique, to address timing issues. Accordingly, the following methods that I examine describe wave pipelining, surfing pipelining and related circuits applicable for surfing pipelining.

For source-synchronous communication, latches are inserted into the data paths to synchronize the data with the strobe signal. Wave pipelining [10] is a technique to improve the throughput of a system by allowing multiple values to be in flight between registers. The throughput improvement is achieved by closely matching the delay between all paths. Figure 2.1 illustrates an example of a wave pipeline. In a wave pipeline, buffers are often added to match the delay of various paths. In this figure, buffers in the 2^{nd} data path are used to match the delays in the other three paths. Let $\delta_{\min}$ and $\delta_{\max}$ be the minimum and maximum delay among all the data paths. Theoretically, this approach allows k waves to propagate through the logic without latches (see [10]) as long as

$$\frac{\delta_{\max}}{k} \quad < \quad P \quad < \quad \frac{\delta_{\min}}{k-1} \tag{2.1}$$

where P is the period of the clock signal. The above equation implies that P must be greater than $\delta_{\max} - \delta_{\min}$. Thus, timing uncertainty limits the throughput of wave pipelining, and this timing uncertainty accumulates along the data paths. In practice, applications of wave pipelining have been largely limited to specialized structures (e.g. on-chip caches) and small numbers of waves (i.e. two or three).

Teehan and others [53] investigated the reliability and throughput of source-synchronous wave-pipelined interconnect. The authors classified the noise disturbances as static and dynamic variation. The static variation includes die-to-die, cross-chip and device-to-device parameter variation. Low frequency power supply noise and temperature variation are also considered as static variation because they change slowly with regard to the bit period. Dynamic variation includes crosstalk noise, inter-symbol-interference (ISI), high frequency power supply noise and other disturbances on a time scale of one to a few bits. This is because static variation only affects the alignment

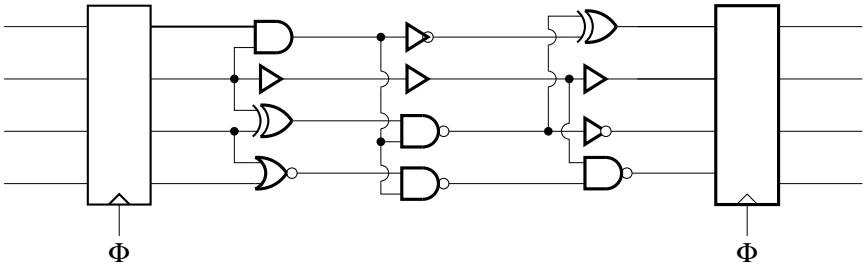


Figure 2.1: A wave pipeline.

of the data and strobe signals. This skew can be controlled by inserting latches in the data path. Dynamic variation contributes to the accumulated edge-to-edge jitter for both the strobe and data lines. Because the strobe line has no latches or other circuits to bound jitter, this leads to a loss of strobe pulses at high data rates. Thus, dynamic timing variation for the strobe line was identified as the most critical limitation of throughput.

The authors developed a statistical timing model to quantify the reliability and throughput of source-synchronous on-chip interconnect. Real designs need bit-error-rates (BERs) in the order of 10^{-20} or less. However, using circuit simulation (*e.g.* HSPICE), it is only practical to establish BERs of 10^{-3} using a single computer for a few hours or a day to 10^{-6} using a compute farm for several days or weeks. These are much larger than the BERs required for real designs, and number of simulation runs required to establish such BERs is prohibitive. The authors of [53] used a statistical method that allows the computation of BERs in the range needed for real designs. The authors observed that with practical assumptions about power supply and cross-talk noise traditional global synchronous communication offers higher throughput than wave pipelining when a BER less than 10^{-20} is required. Thus, ignoring dynamic variation in wave pipelining will result in an optimistic design which is vulnerable to the accumulated jitter.

Adding latches in wave pipelining helps to bound the accumulated timing uncertainty. However, latches add extra delay into a design and consume extra power. Overcoming the disadvantages of latches in a pipeline becomes especially important at very high operating frequencies where the overhead from latches can be a large fraction of the total clock period. In 2002, Winters and Greenstreet proposed surfing pipeline to bound the timing uncertainty in data paths [57]. The key idea is that a timing pulse called fast is propagated along the pipeline as shown in Figure 2.2. Logic elements in

the pipeline are modified so that their delay is lower when fast is asserted than when it is not. In particular, if the data path is faster than the timing path when fast is asserted, and the data path is slower than the timing path when fast is not asserted, then switching events in the data path will be attracted to coincide with the rising edge of the fast signal. In other words, data path events move forward with the rising edge of fast like a surfer rides the leading edge of a wave.

Winters and Greenstreet [57, 58] showed that this approach can achieve negative overhead: the latency of a surfing pipeline is less than the combinational logic delay of the corresponding non-surfing circuit. They also analyzed the timing constraints for reliable surfing which are captured by the following two requirements:

1. When the timing pulse is asserted, the maximum delay of the gate $\delta_{1,max}$ must be less than the minimum propagation delay of the timing pulse $\delta_{f,min}$, where $\delta_{1,max}$ is the maximum delay of the logic gate with fast asserted, and $\delta_{f,min}$ is the minimum stage-to-stage delay of the fast pulse. Thus a logic event that lags the fast signal can eventually catch up.

2. If the timing pulse is not asserted, the minimum delay of the gate $\delta_{0,min}$ must be greater than the maximum delay of the timing pulse $\delta_{f,max}$, where $\delta_{0,min}$ is the minimum delay of the logic gate with fast not asserted. Thus, a logic event that arrives before the fast pulse will be slowed down to eventually coincide with the rising edge of the fast signal as desired.

In conclusion, the timing variation induced by the timing pulse must be greater than the timing variation caused by noise and other disturbances such as power supply variation, thermal noise and so on. These two requirements can be summarized with the following inequality:

$$\delta_{1,max} < \delta_{f,min} < \delta_{f,max} < \delta_{0,min} \tag{2.2}$$

If Inequality 2.2 is satisfied, the arrival time of logic events in the data paths will be attracted to the rising edge of the timing pulse. As shown in Figure 2.3, the timing pulse divides the period of the timing pulse into four intervals. The interval $[t_1, t_4]$ is the capture region. In interval $[t_1, t_2]$, the input event arrives earlier than the timing pulse, and the logic delay is greater than the propagation delay of the timing pulse. Conversely, in interval $[t_3, t_4]$, the input event arrives after the timing pulse, and the logic

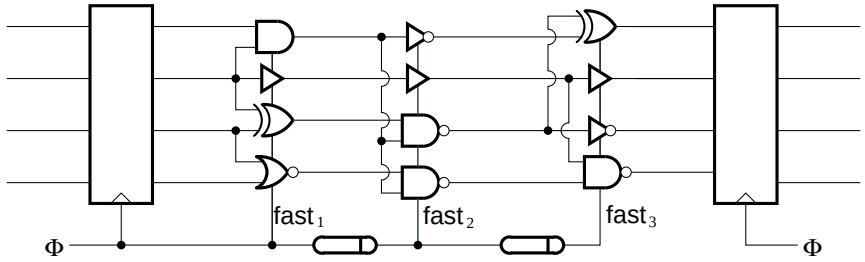


Figure 2.2: Surfing pipelining.

delay is less than the delay of the timing pulse. Whenever the input event comes in interval $[t_1, t_2]$ or $[t_3, t_4]$, the input event at the next stage will arrive closer to the timing pulse. Hence, after several stages, the input events will converge to arrive in the steady-state surfing interval $[t_2, t_3]$. The interval $[t_4, t_5]$ is the metastability interval. Events in this interval will eventually exit to surf with either the preceding wave or the following wave. Surfing creates an event attractor such that the delay spread in the data path is kept small regardless of the pipeline length.

Surfing may increase throughput and decrease latency simultaneously, because surfing logic elements may achieve less delay than their non-surfing counterparts. Winters and Greenstreet modified self-resetting domino logic to obtain a surfing gate as shown in Figure 2.4. When fast is high, the keeper transistor $m3$ is off and the voltage on node out is pulled a little bit higher by transistor $m5$. The voltage level on node out is determined by the voltage divider formed by the NMOS pull-down transistor in the inverter lout and pull-up transistor $m5$. This speeds up the rising transition on node out. If fast is not asserted, the keeper transistor $m3$ is enabled and this slows down the rising transition on node out. The main disadvantage of this circuit is that surfing is achieved by creating a short circuit with both the transistor $m5$ and the NMOS device in the inverter lout being on. Thus, power consumption is a concern for Winters and Greenstreet's design.

Greenstreet and Ren applied surfing to interconnect in 2006 [24]. They used the traditional buffer insertion technique. However, they replaced the inverter with a "soft-latch" shown in Figure 2.5. This "soft-latch" is a static surfing inverter. Without the left part, the "soft-latch" would be a traditional inverter. The left part is used to modulate the driving strength of the inverter. If fast is asserted, the right part and left part are both enabled. Thus the delay of the surfing inverter decreases. Otherwise, only the

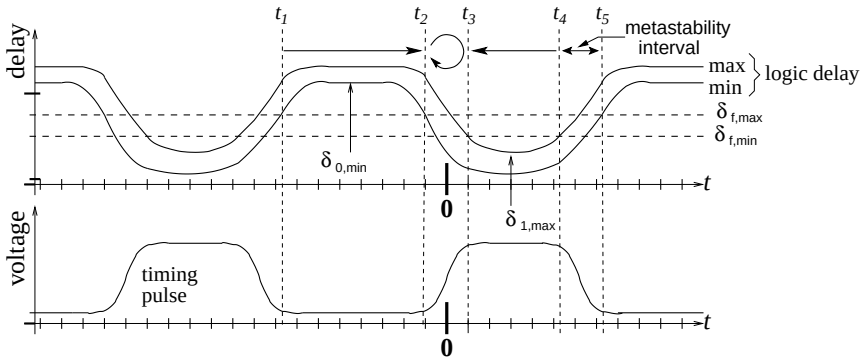


Figure 2.3: Timing requirement of surfing pipeline.

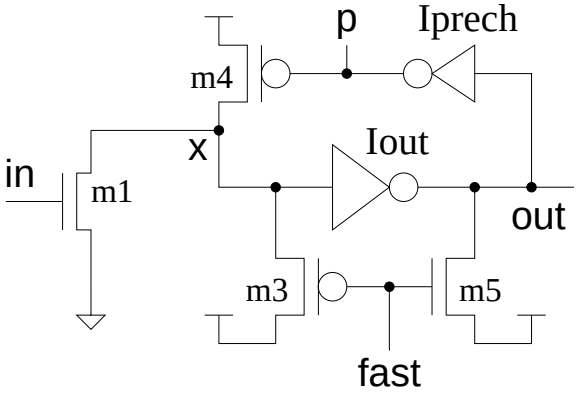


Figure 2.4: A self-resetting preswitching buffer.

right part is enabled, and the delay increases. Figures 2.6 and 2.7 compare surfing interconnect with tradition buffer insertion for an implementation using the TSMC 180nm process. Without considering variations of power supply, process and temperature and other disturbances, for low throughput requirements, tradition buffer insertion technique provides lower energy consumption and less delay than surfing. When throughput is greater than 1.8GHz, the surfing interconnect performs better than the traditional technique under ideal conditions. Unlike the surfing circuits in [57, 58], the delay of a surfing inverter is greater than that of a non-surfing inverter. Hence the surfing interconnect does not obtain negative overhead. With low throughput, the velocity of the traditional buffer technique is mainly determined by the inverters between the latches. This explains why with slow throughput, the traditional buffer technique performs better than the surfing interconnect. However, when parameter, power supply and temperature (PVT) variations and other disturbances are taken into account, extra latches must be added to the synchronous interconnect to ensure correct operation under all operating conditions. These disturbances seriously degrade the performance of traditional buffer technique as indicated by the "Synchronous derated" curves in Figures 2.6 and 2.7. Surfing interconnect is designed for the average case, and the delays of the data and strobe paths track well over PVT variations. The delay of the inverters is dynamically adjusted according to the disturbances. Due to these properties, surfing interconnect provides higher velocity and lower power consumption than the traditional buffer insertion when the realistic variations are considered and the throughput is higher than 0.8GHz.

The authors of [24] used a simple RC model for buffers and wires and only considered dynamic power consumption. To better understand the trade-offs in this design, I performed HSPICE simulations for various line lengths and inverter sizes and compared the results with the model from [24]. Figure 2.8 shows the results. The power estimates from [24] are 44% to 60% of those observed from HSPICE simulations. There are two causes for this discrepancy: the model from [24] underestimates gate capacitance for the purposes of computing energy consumption and it only considers dynamic power and neglects short-circuit power consumption.

From Figure 2.8, we can see that power consumption grows much faster with transistor gate width than predicted by the model from [24]. This implies that the gate capacitance used in [24] is an underestimate. Our experiments show that to match the delays between the RC model and HSPICE, 2fF/μm(as in [24]) is a good approximation. However, to match the energy consumption 3fF/μm provides a much better estimate. We have not deter-

mined what aspect(s) of transistor non-linearity lead to this discrepancy but plan to investigate this in future research.

Short-circuit power is the other cause of the discrepancies. Vendrik [54] observed that if the transition time of the output of a buffer is longer than the transition time of the input, then the short circuit power consumption is less than 20% of the total power dissipation. However, if the wire segments between buffers are too long, then Vendrik's condition is violated, and the short-circuit power becomes a significant fraction of the total power. This short-circuit energy accounts for the remaining discrepancy between the model from [24] and HSPICE. I believe that the impact of short-circuit current and the error in the capacitance model should be similar for synchronous, asynchronous and surfing interconnect. Thus although the absolute energy-per-bit will be higher for all schemes than reported in [24], the relative ordering of the different techniques should remain roughly the same.

Typical designers avoid wave pipelining of interconnect because there is little CAD tool support, and the timing constraints for wave pipelining are difficult to satisfy. Surfing simplifies this timing analysis by bounding the difference between the data and strobe. Greenstreet and Ren also compared the response of surfing interconnect and traditional techniques to input jitter. They showed that at the output side, surfing interconnect can keep a fixed phase relative to the timing signal. The timing spread between the data signal and the timing signal for synchronous communication increases along the stages until failure occurs. Hence surfing interconnect is more robust to temperature variation, power supply variation and input jitter. Although surfing interconnect is more robust to noise, the design in [24] used a timing chain consisting of a chain of inverters. It is not surfing, and jitter accumulates. Due to the drafting effect [56] and dynamic variations [53], long chains of inverters drop pulses. Thus, the timing chain is the bottleneck for the surfing design for high throughput and long distance communication. A careful analysis of power consumption and reliable timing chains are needed for feasible surfing communication. I address these issues in Chapter 3.

Fairbanks and Moore described an analog micropipeline ring for high precision timing [21]. They replaced the digital C-element in a traditional micropipeline [51] with an analog C-element as shown in Figure 2.9. They used this analog C-element in a closed-loop micropipeline ring as shown in Figure 2.10 to generate precise timing signals. Figure 2.11 presents three simulations that I ran with different separation time between $\overline{S_1}$ and S_8. Although the separation time between $\overline{S_1}$ and S_8 vary in a wide range, the stage delay between the last triggering event to S_0 is roughly the same. We use the second simulation (2.11 (b)) to explain this. In the second

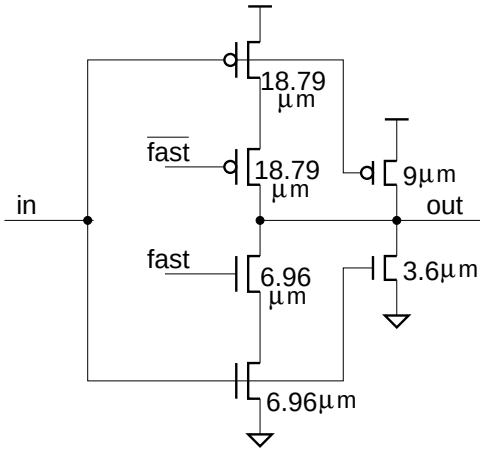


Figure 2.5: A surfing inverter ([24]).

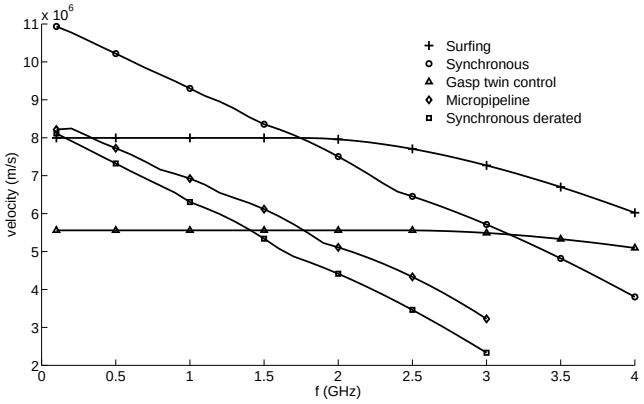


Figure 2.6: Velocity comparison of different interconnects [24].

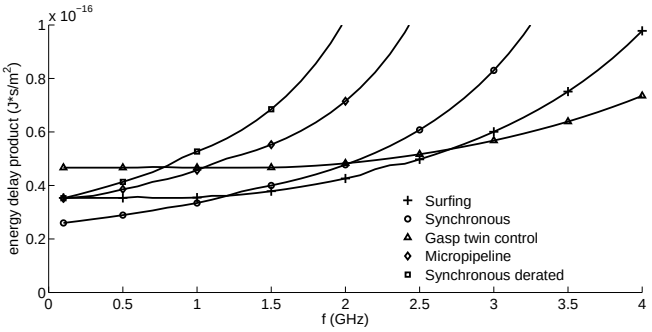


Figure 2.7: Energy delay product comparison of different interconnects [24].

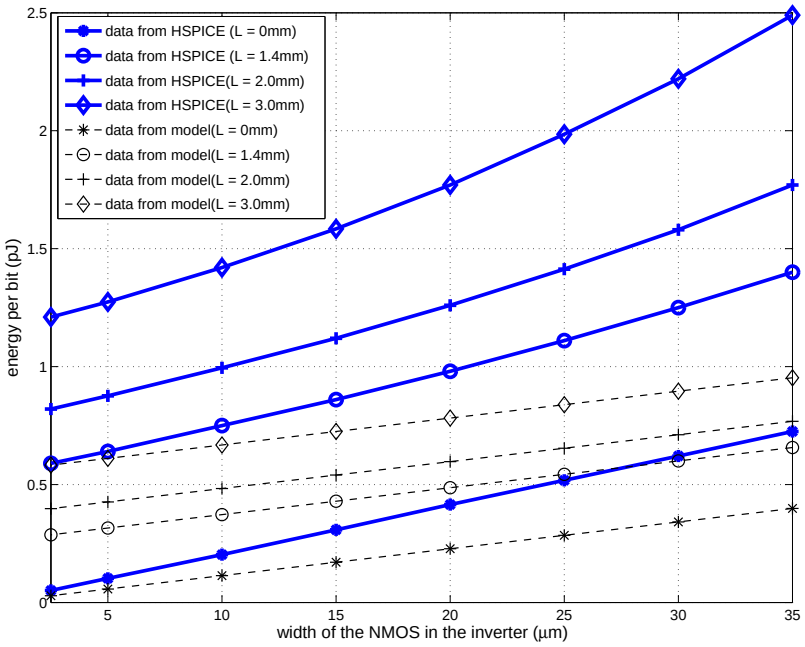


Figure 2.8: Energy comparison from the model in [24] and HSPICE simulation.

simulation, $\overline{S_1}$ comes earlier than S_8 by 126ps. Observing the waveform of S_0, before the rising transition, S_0 first dips to roughly 0.05V. Then S_0 climbs up to 0.15V and we see some tiny oscillation after that. This oscillation is due to the three-inverter ring formed between the neighboring stages. For example, the loop connecting S_0, S_1, $\overline{S_1}$ and S_0 is a three-inverter ring. If S_8 comes when S_0 is at 0.05V, the delay will increase. If S_8 comes when S_0 oscillates to its highest voltage, the delay will decrease. In all three simulations, the last triggering event rises after the voltage dip of S_0; thus, the stage delays are close to each other. The upper figure in Figure 2.12 shows the closed-loop stage delay as a function of separation time between the two inputs, S_8 and S_1 where $T(S_8)$ and $T(S_1)$ are the arrival times of the two inputs. The bottom figure illustrates the voltage on the rising transition of S_0 at $T(S_8)$. The delay curve mirrors the voltage curve of S_0. The delay variation is around 40% and the slope is roughly -0.4. These two factors bring the ring into lock and generate evenly spaced timing signals. If two events are too close to each other, the stage delay increases to push them back apart but does not increase as much as the variation of the separation of the two events. Along the ring, the stage delay increase becomes smaller and smaller and finally the variation goes to zero. Likewise, if an input event is late, the stage delay decreases to compensate. As the authors pointed out, the delay of analog-C element is linear in power supply voltage. This will produce evenly spaced phase in the ring in the ideal scenario where there is no power supply or other noise.

Maneatis and Horowitz [41] split a single-input inverter into a dual-input inverter in a shunt configuration as shown in Figure 2.13. This circuit's delay is dependent on the arrival time of the two inputs. Similar to the surfing buffer from Figure 2.5, this circuit also demonstrates the surfing property. However, in this circuit, the surfing property is achieved by the short circuit current between the two inverters. Assume that in1 and in2 are initially high and out is low. If in1 goes to low earlier than in2, then the upper inverter will drive out to be high while the bottom inverter continues to drive out low. As a result, the voltage on out will rise to an intermediate voltage level between ground and V_{DD} and then go the rest of the way to V_{DD} once in2 goes to low. Figure 2.14 shows the array they constructed to obtain high precision timing signals. They used one inverter of the dual-input inverter to form a ring and used another such inverter to couple the rings together and adjust the phase relationship between the rings. This structure forms an array oscillator which produces a precise delay resolution equal to the buffer delay divided by the number of rings. As with Fairbank's analog C-element oscillators, their design is sensitive to power supply noise which

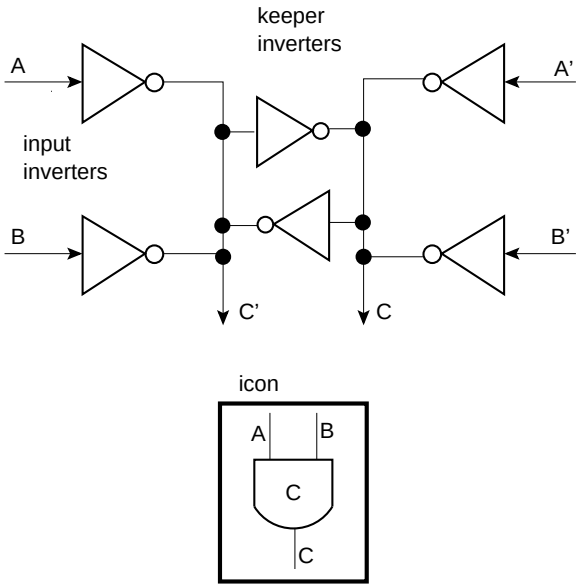


Figure 2.9: Analog C-element.

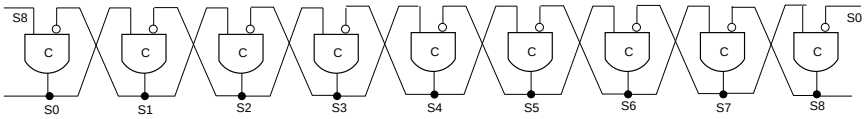


Figure 2.10: A micropipeline.

will limit the resolution in practice. Furthermore, this array structure has several stable operation modes and the actual mode obtained depends on the initial conditions. Thus, the ring needs auxiliary circuits to set the ring in the desired stable operating mode.

In one contribution of this thesis, we extend the application of the surfing inverter as shown in Figure 2.5. We use this surfing inverter to build a delay-locked loop. Instead of using an inverter chain to propagate the timing signal, we use a delay-locked loop chain to transmit the clock signal. The surfing effect not only bounds the timing difference between the clock signal and data signal, it also cancels the drafting effect in the timing chain and avoids jitter peaking. This guarantees reliable surfing synchronous and asynchronous communication for long distance, on-chip interconnect.

2.2 Related Work for LC Signaling

This section presents some related work for LC signaling which offers much higher propagation speed than RC signaling. The key challenges are handling attenuation and dispersion due to line resistance.

Afshari and Hajimiri [2] added variable capacitance into the transmission line using MOSFET varactors. This helps to narrow pulse width and sharpen edges. In contrast, the line resistance causes the pulse to spread out. Thus, the variable capacitance helps to cancel the dispersive property of lossy lines. However, if the input pulse is wide enough, the line cannot concentrate the input pulse into one signal pulse, instead, the wide input pulse will be spread into multiple, narrow soliton pulses. The authors introduced a gradually scaled, nonlinear transmission line to gradually narrow the pulse and avoid the splitting into several pulses. The authors observe that due to polysilicon depletion and short channel charge quantization effects, the capacitance of an accumulation-mode MOS varactor drops for voltages on A above a voltage they called V_2 as shown in Figure 2.15. When the voltage on A increases from 0 to V_2, the NMOS varactor works in cut-off region first and linear region later, and the capacitance grows. This non-monotonic capacitance curve allows the varactor shunted line to sharpen both the rising and falling edges of a signal. This technique requires biasing the interconnect near voltage V_2, the local maximum. This voltage level may vary with power supply variation, temperature variation and parameter variation. They did not address how this bias voltage could be determined or applied in a practical design. In addition, the capacitance curve is not symmetric around V_2.

Thus the sharpening strength of the varactor is not the same for the rising edge and falling edge. The authors only demonstrated that they could send pulses down the line. However, they did not show how the line works in a real on-chip communication system.

Surfliner interconnect was introduced in [18] and [68]. In this method, distributed shunt conductance is intentionally added to the line to cancel dispersion. In theory, if $g = rc/l$, there will be no velocity dispersion along the line and all frequency components will attenuate at the same rate and propagate at the speed-of-light in the medium(*i.e.* $\frac{1}{\sqrt{lc}}$). The disadvantage of this approach is that the attenuation constant increases from $r/(2Z_0)$ without shunt conductance to r/Z_0 where $Z_0 = \sqrt{\frac{l}{c}}$. Hence, the surfliner interconnect consumes more power than the typical interconnect without shunt conductance. The high attenuation rate limits the length of the interconnect. As in paper [2], the authors did not present a complete on-chip communication circuit.

Jose and others [29] used pulsed current-mode signaling to achieve nearly speed-of-light for intra-chip communication. Due to the attenuation of the line, the voltage swing at the far-end of a differential, 3mm long transmission line implemented in TSMC 180nm process is 120mv for a 0.21V input pulse which requires careful design of the receiver. Although the driver only consumes 0.29pJ/bit, the DLLs, skewing and deskewing latches consume 3.1pJ/bit. Hence more than 90% of power consumption is spent on deskewing compensation. The clock phase recovery was done manually off-chip and was not included in the power budget. The authors extended their work to include distributed amplifiers in the line to compensate for the energy loss in [30]. They could transmit signal on a $14mm$ long wire. However, the latency is almost doubled due to the capacitance of the distributed amplifier.

Chang and Talwakar proposed converting a baseband signal to an RF signal to transmit symbols at the speed-of-light [15]. The whole circuit includes the following parts: transmitter, receiver, an oscillator to generate the high frequency carrier and differential interconnect. A 1GHz signal is modulated onto a 7.5GHz carrier to ensure LC regime transmission. The advantage of this design is that the transmitted signal only contains a narrow range of frequency components. Thus, frequency dispersion is not a problem in this design. The authors apply this technique to a $16\mu m$ wide, 20mm long differential line using a $0.18\mu m$ CMOS technology. The delay is 300ps. The whole circuit including the receiver, transmitter and the oscillator to generate the high frequency carrier consumes 16.1mW and supports a throughput of 2Gb/s. From the paper, it appears that the receiver and transmitter use

the same oscillator. Thus the high frequency carriers for the receiver and transmitter have the same phase and frequency. However, this will not hold for real designs. Phase differences between the transmitter and receiver will further limit the throughput and increase the power consumption of the receiver. Thus the authors underestimated the power consumption of the whole circuit by using only one oscillator for the transmitter and receiver.

In 2001, Wood *et al.* proposed a traveling-wave oscillator [60] as shown in Figure 2.16. We are interested in using this technique to make a communication link. The traveling-wave oscillator is a closed-loop differential LC line. If there were no resistance on the line, a wave would propagate around the ring forever. The authors used distributed amplifiers, cross-coupled inverters, to compensate for the energy loss arising from the wire resistance. The cross-coupled inverters have two functions. During a transition, the cross-coupled inverters work as an amplifier to sharpen the edges. After the transition, the cross-coupled inverters are latches that maintain the state on their segments. The traveling-wave oscillator is attractive for two reasons:

- Edges are very sharp. Figure 2.17 shows the operation of the traveling-wave oscillator in a 90nm TSMC process. The rise and fall times are each around 10ps each.

- Low power consumption. The distributed amplifiers only provide the energy consumed by the line resistance.

For use as a clock generator, Wood *et al.* needed to increase the capacitance of the transmission line to obtain an oscillation frequency that was not higher than can be used by digital circuits. Thus they used large inverters which provide plenty of gain and simplify the design of the circuit. In Chapter 4, I provide an open-loop version of this circuit for data transmission. In this case, reducing the line capacitance is desirable, and I provide a detailed analysis of the oscillator operation that was not given in [60].

LC signaling techniques can be divided into two categories: passive and active. Active compensation provides energy along the wire for long distance interconnect such as [11, 23, 30, 60]. Passive methods provide no energy compensation and focus on velocity dispersion only, such as [18], [68], [2] and [15]. The non-linear transmission line proposed in [2] does not provide energy compensation because the varactor is connected between the signal line and a DC voltage supply.

In this thesis, we propose a LC signaling technique. To obtain the surfing effect, we added distributed varactors into the transmission line. Varactors not only produce the surfing effect; their non-linear capacitance curve also

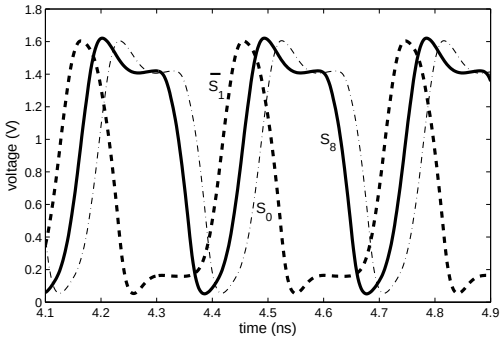
helps to narrow the pulses. This is the basis for our low-swing surfing LC line. For our full-swing LC line, cross-coupled inverters are added to restore the energy loss due to wire resistance.

2.3 Summary of My Master's Work

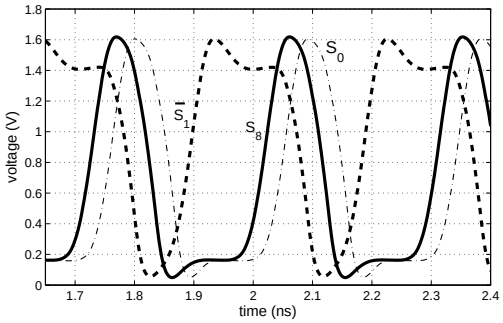
In my master's research [64], I designed and tested a working surfing chip, invented new surfing circuits that greatly reduced the power penalties associated with surfing, and developed a novel noise-margin analysis approach. The surfing chip implemented a simple, pseudo-random sequence generator. It supported two independent waves of computation in a 12 stage ring without any latches or other storage elements. I operated the ring for over 48 hours and $2.6 * 10^{15}$ surfing transfers of data between stages without error. The chip operated correctly over a wide range of power supply voltages. This chip demonstrated that surfing pipelines are possible in the real world. The robustness of the surfing circuits was quantified and compared with the static gate, dynamic logic gate and output prediction logic using the novel noise margin analysis.

My master's research focused on demonstrating the surfing technique in practice and analyzing the robustness of surfing circuits compared to static and dynamic CMOS logic families. The test chip demonstrated that the surfing technique works in real silicon. The noise margin analysis confirmed the timing stability of the surfing circuits. This timing stability bounds the timing difference between the data and fast signals and makes latchless pipelines possible.

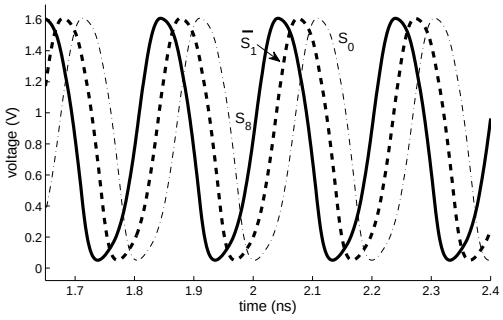
This thesis extends the application of surfing technique to on-chip communication and develops novel surfing circuits for this application.



(a) Simulation run 1: separation time = 37ps and stage delay = 32.4ps.



(b) Simulation run 2: separation time = 126ps and stage delay = 32.3ps.



(c) Simulation run 3: separation time = 33ps and stage delay = 32.8ps.

Figure 2.11: Simulation of an 9-stage micropipeline(see Figure 2.10)

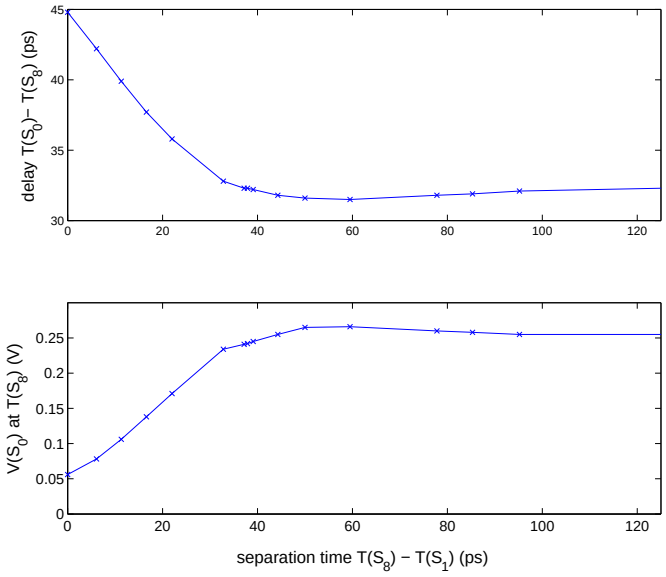


Figure 2.12: Delay curve of the analog C-element.

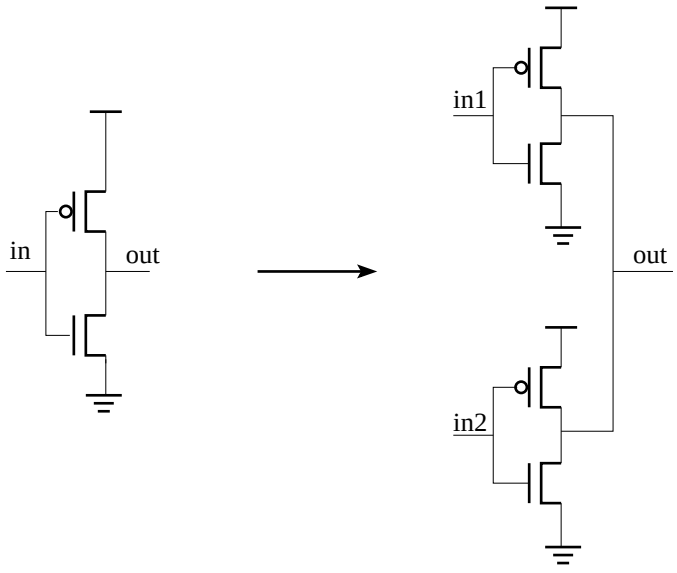


Figure 2.13: Another surfing inverter.

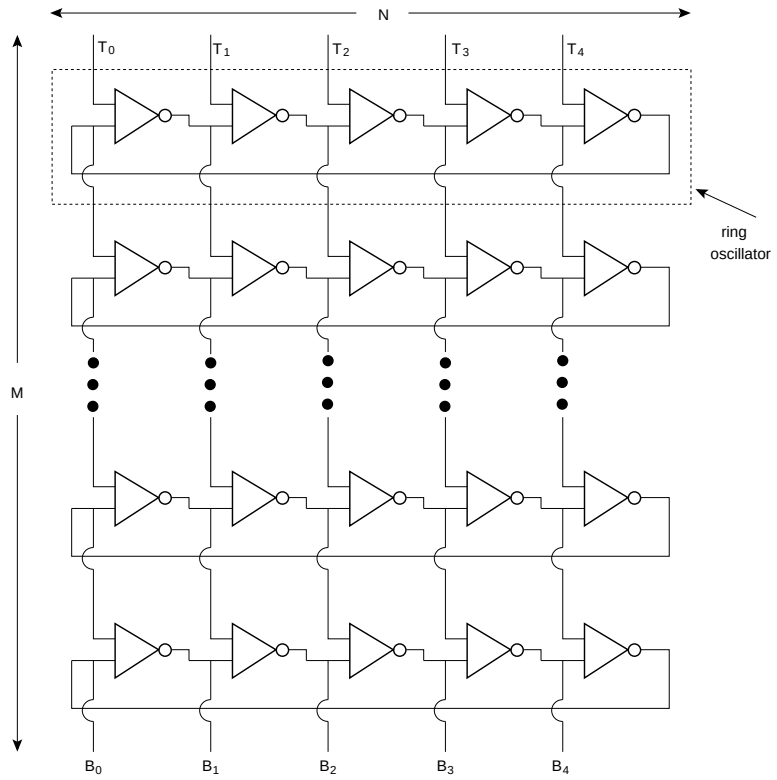


Figure 2.14: The ring for high precision timing (from [41]).

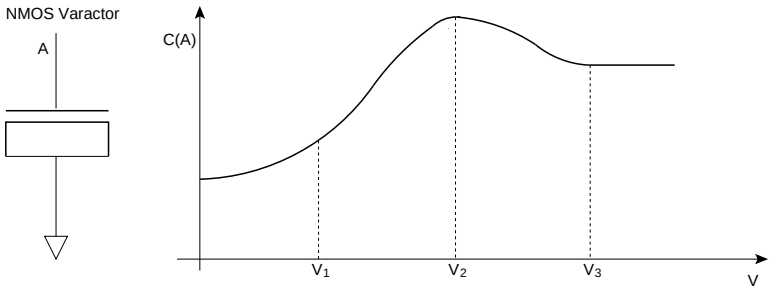


Figure 2.15: The capacitance curve for a varactor (from [2]).

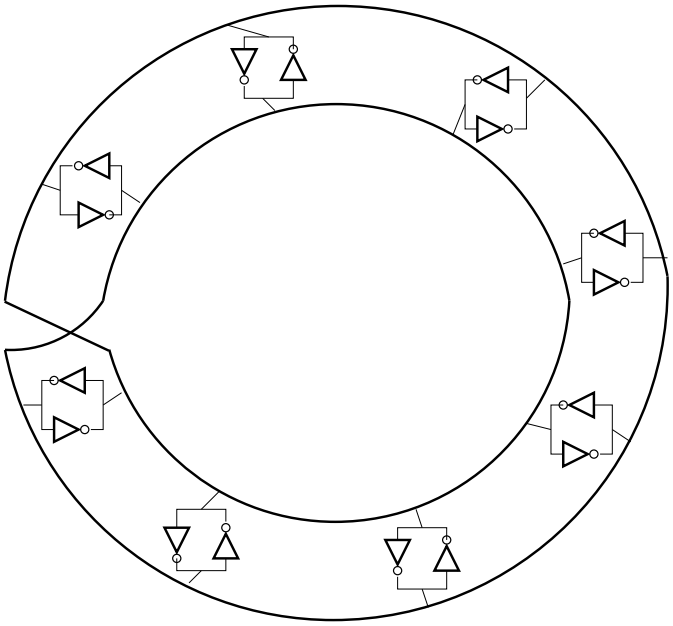


Figure 2.16: The traveling-wave oscillator.

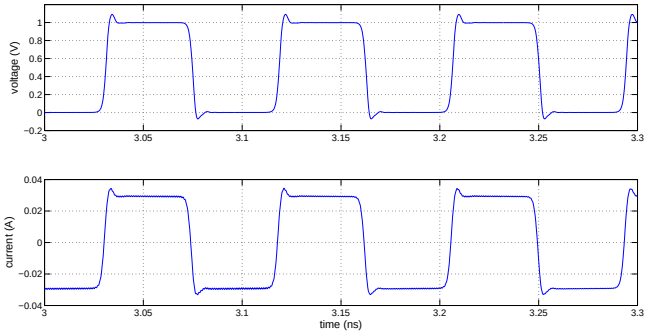


Figure 2.17: Simulation of the traveling-wave oscillator.

Chapter 3

Surfing RC Interconnect

My investigation of surfing RC interconnect starts from a surfing source synchronous communication scheme as proposed by Greenstreet and Ren [24] shown in Figure 3.1. The data path uses surfing inverters as shown in Figure 2.5 to keep the data signal aligned with the strobe signal. Their design used an inverter chain to send the strobe signal. As described in Section 3.1, the jitter on the strobe signal increases at each inverter. This limits the total length of on-chip source synchronous interconnect when operating at high data rates.

This chapter starts with a detailed explanation of the disadvantages of using an inverter chain to send the clock signal. Next, we show how a very simple modification to the inverter chain design produces a low-gain DLL at each stage. We examined applications of this DLL in source synchronous and asynchronous communication.

3.1 Jitter in Inverter Chains

Consider the problem of forwarding a clock signal through a chain of buffers and long wire segments as shown in Figure 3.2. Such chains can be used in clock distribution networks or for clock forwarding for source-synchronous communication. Here, our focus is on source synchronous designs. A fundamental problem for such a design is jitter accumulation along the chain. Even if all of the inverters are of the same design and all of the wires are of the same length, random variations due to power-supply noise, crosstalk,

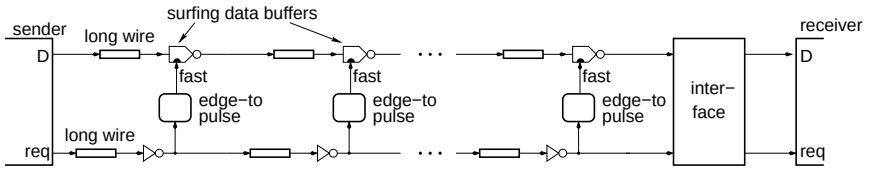


Figure 3.1: Surfing source synchronous communication (from [24], Figure 1)

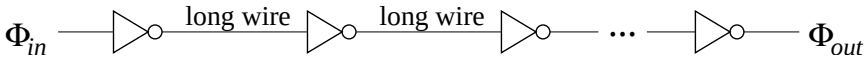


Figure 3.2: Forwarding a clock through a chain of inverters

temperature variation and intra-chip parameter variation add jitter at each stage, and this jitter is cumulative. Furthermore, intersymbol interference (ISI) effects (aka "drafting" [56]) amplify jitter [7]. These two effects, the random walk of edge timing combined with the jitter amplification of intersymbol interference, will cause a sufficiently long buffer chain to drop clock pulses even when operating at low clock frequencies. Figure 3.3 shows the maximum length chain through which a clock signal can propagate reliably as a function of the clock period. The data in this figure is from HSPICE simulations for inverters driving long wires optimized for minimum energy delay product in the TSMC 0.18μm process: one run was performed at each target frequency, and we noted the first stage at which pulses were missing. While the chains considered in Figure 3.3 are much longer than those used in typical designs, the problems of jitter amplification are concerns for designs with shorter chains as well – identifying the point at which pulses are completely dropped is an extreme failure criterion.

Similar problems occur when using asynchronous signaling. Simple handshaking protocols incur large penalties in cycle time due to the round trip delay for sending the data forward and an acknowledgment back. To avoid these disadvantages, one can use credit-based protocols where the sender can transmit up to K values before receiving an acknowledgment [20]. With these designs, multiple request (or acknowledge) events can be in flight at the same time, and are vulnerable to jitter accumulation and ISI just as in the synchronous case described earlier. Events can be dropped. Using an inverter chain to send clock signal is length-limited. In the following, we will use a delay-locked loop chain to send clock signal instead.

3.2 Surfing DLLs

Figure 3.4 shows a simple delay-locked loop (DLL). It is composed of three parts: a variable delay line, a phase detector and a loop filter. The DLL operates as a simple, feedback-control loop that seeks to set the delay of the adjustable delay element to the period of the incoming clock. Each clock event is compared with the delayed event from the previous clock period. If the delayed version occurs before the arrival of the new event, then the

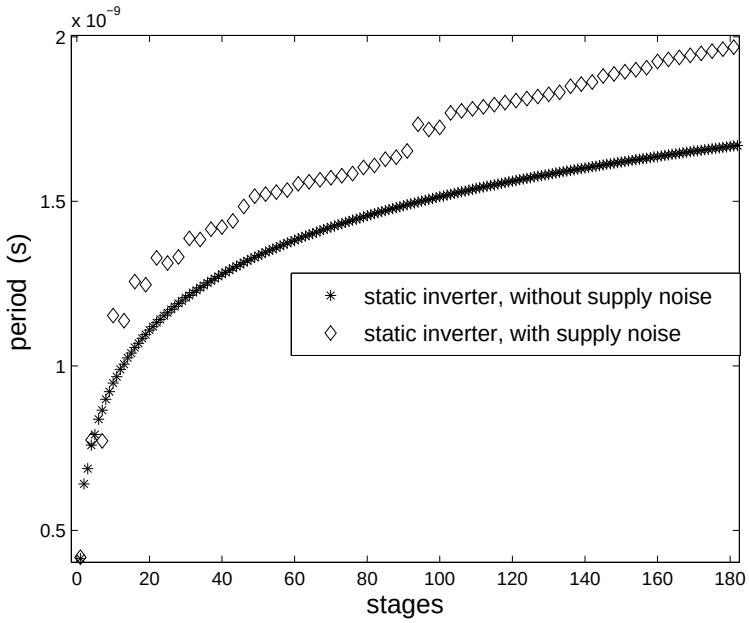


Figure 3.3: Maximum reliable chain length vs. clock period

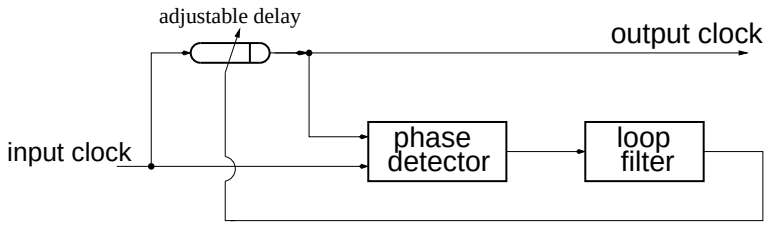


Figure 3.4: A simple DLL (from [38], Figure 1)

adjustable delay is increased. Conversely, if the delayed version is late, the delay is decreased. This style of DLL is often used to generate multiphase clocks and to deskew clocks in large designs. This design exhibits jitter peaking [38] because the phase comparator cannot distinguish between an early arrival of an input clock event (i.e. input jitter) and an excessive delay of the delay element. Thus, if the input event arrives early, the DLL will *decrease* the delay of adjustable delay element, and the output will occur even earlier than it would from the input jitter alone. This jitter peaking can be mitigated by lowering the loop bandwidth or by using a different DLL architecture with a separate, low-jitter clock reference. Phase-locked loops (PLLs) can also be used to attenuate jitter. In the next section, we present a novel DLL design based on surfing that has an particularly simple implementation and avoids jitter amplification.

3.2.1 Basic Operation

We use a surfing inverter to develop a novel digital DLL to avoid the jitter peaking problem. Our surfing inverter is a simple modification of the design shown in Figure 2.5 from [24]. Figure 3.5 shows our design. All transistors have a length of 0.18μm. The large transistor sizes reflect our intended application of driving long-wire interconnect. The fast signal in Figure 2.5 is now called predict as it is set to accelerate the next transition at its *predicted* time. We simulated this surfing inverter driving a 2.1mm long wire with HSPICE using parameters for the TSMC 0.18μm process and plotted the delay curve as shown in the upper part of Figure 3.6. The in and predict signals were generated by using "pulse" waveforms with 300ps rise and fall times. The the in signal buffered by a chain of four copies of the surfing inverter circuit where the in and predict inputs of each such buffer are connected together. The predict signal was buffered with four inverters with 1.2μm pull-ups and 0.48μm pull-downs. Thus, time separation of in

and predict was varied by changing the relative times of the transitions from the pulse sources driving these inverter chains, and the actual separation was measured at the $V_{dd}/2$ point on the in and predict signals for the surfing inverter. With this arrangement, the slopes of the transitions for the surfing inverter's inputs were nearly independent of the arrival times. Section 3.3 examines the operation of surfing inverters with power-supply noise and shows that the qualitative behaviour matches the predictions based on these simple simulations. The delay curve for falling input edges is similar. For simplicity, we assume symmetric delays for rising and falling edges in our derivations; the generalizations to handle asymmetric delays are straightforward. Let t_{in} denote the time of a transition on signal in; let $t_{predict}$ denote the time of a transition on signal predict; and let t_{out} denote the time of the resulting transition on the out output. The vertical axis is the delay, d:

$$d = t_{out} - t_{in}$$

The horizontal axis is the separation time, t_s, from the arrival of the predict signal to the arrival of the in signal:

$$t_s = t_{in} - t_{predict}$$

$[t_{s,\min}, t_{s,\max}]$ defines the stable surfing interval as shown in Figure 3.6. If the time from a rising (resp. falling) edge of in at stage i to the rising (resp. falling) edge of predict at stage i is between $t_{s,min}$ and $t_{s,max}$, each stage will have its input and output events converge to a fixed delay relative to those of its predict signal [57].

This surfing inverter can be used to implement a simple delay-locked loop as shown in Figure 3.7. For the results presented in this thesis, we implemented the delay line using a simple chain of inverters. Because the clock's rising and falling events alternate, we use a delayed version of the output to predict when the next input event should happen. Surfing occurs when the next input clock transitions relative to the predict signal so as to achieve an event separation in the high-slope part of the surfing timing curve as shown in Figure 3.6. Thus, the surfing DLL will lock if the input period P satisfies the following inequality:

$$d_{max} + t_{s,min} + D \leq \tfrac{P}{2} \leq d_{min} + t_{s,max} + D \tag{3.1}$$

where D is the delay of the delay element, d_{min} and d_{max} are the minimum and maximum delay of the surfing inverter. Taking the delay curve as shown in Figure 3.6 as an example, the surfing DLL can operate with clock periods

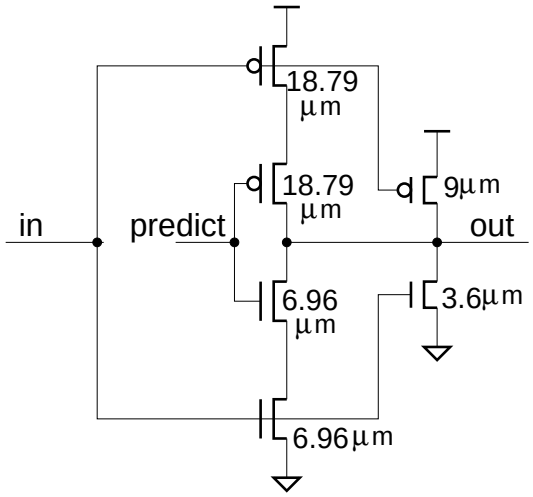


Figure 3.5: The surfing inverter

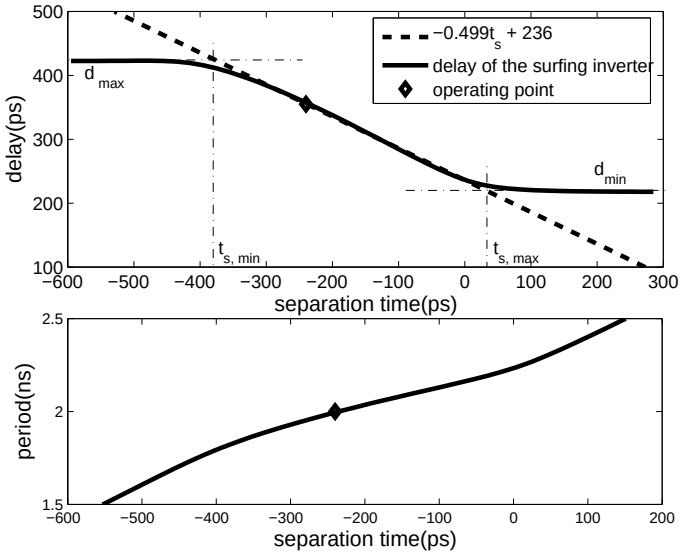


Figure 3.6: The delay of the surfing inverter

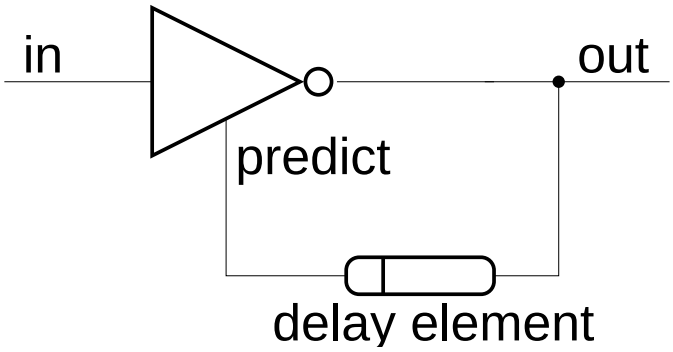


Figure 3.7: A surfing DLL

ranging from $2D + 52$ps to $2D + 572$ps. As an example, the bottom plot in Figure 3.6 shows the period corresponding to the separation time when D is 880ps.

The surfing DLL has important simplifications when compared with a traditional DLL shown in Figure 3.4. The surfing DLL combines the functions of the adjustable delay, phase comparator and loop filter into a single surfing inverter. Rather than using a traditional voltage or current controlled delay, the surfing inverter effects a *weighted average* of the times of the input events on the in and predict signals. This removes the need for the phase-detector which, in a traditional DLL, translates timing differences into voltages or currents.

3.2.2 Jitter Propagation

We now analyze the jitter-propagation characteristics of our design. Assume that the circuit is operating near the point labeled by the diamond in Figure 3.6. Then for $t_{s,\min} \leq t_s \leq t_{s,\max}$, we can approximate the delay curve with a linear function

$$d \approx -\alpha t_s + \tau_0. \tag{3.2}$$

where $\alpha = \frac{d_{max} - d_{min}}{t_{s,max} - t_{s,min}}$. For the delay curve from Figure 3.6, $\alpha = 0.499$, $\tau_0 = 236$ps, $t_{s,\min} = -374$ps, and $t_{s,\max} = 38$ps. In response to a small perturbation of the timing of the input clock, the circuit is characterized with the following equation:

$$\Delta t_{out} = \alpha * \Delta t_{predict} + (1 - \alpha) * \Delta t_{in} \tag{3.3}$$

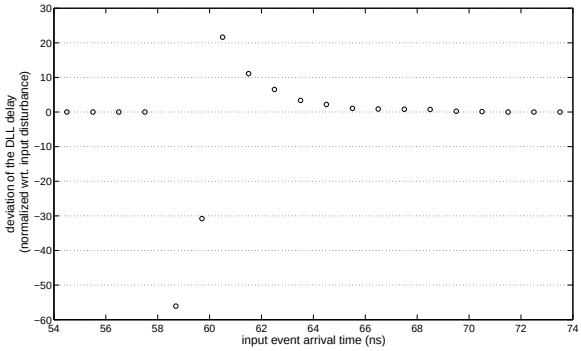


Figure 3.8: Jitter attenuation of the surfing DLL

Assume that the first event of in is disturbed by Δt_{in} and all other events are undisturbed. We use s_i to denote the i^{th} event on signal s, and we number the events with the output generated from the disturbed input as event 0. The input disturbance propagates along the in to out path once and the predict to out path i times to disturb the i^{th} output event. Thus,

$$\Delta t_{i,out} = (1 - \alpha) * \alpha^i * \Delta t_{in} \tag{3.4}$$

The summation of the sequence is Δt_{in}. However, the disturbance is spread over the subsequent events. The jitter in the circuit decays by a factor of α for each successive clock edge. Figure 3.8 shows the operation of a surfing DLL operating with a 2.0ns period. One pulse (i.e. a rising and falling edge) comes 200ps later than the expected time. For these two edges, the in-to-out delay of the surfing inverter *decreases* because of the increased separation time from predict to in. After these two events, input events arrive at the jitter-free time, which *decreases* the separation from predict to in because predict has been delayed by the lateness of the earlier pulse. This causes the delay of the surfing inverter to *increase* for the events following the delayed pulse. The disturbance decays exponentially as predicted by Equation 3.4 and is barely discernible after seven events.

If the j^{th} input event experiences jitter $\Delta t_{in,j}$, then the disturbance that the j^{th} event, $\Delta t_{in,j}$, contributes to the i^{th} output event is $(1-\alpha)\alpha^{i-j}\Delta t_{in,j}$. Thus, the disturbance of the i^{th} event of the output is the summation of the disturbances contributed by the input disturbances from the 0^{th} event to

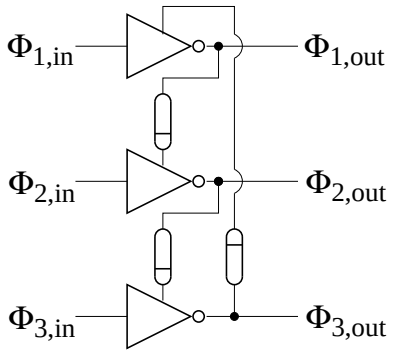


Figure 3.9: A multiphase surfing DLL

the i^{th} event:

$$\Delta t_{i,out} = \sum_{j=0}^{i} (1-\alpha) * \alpha^{i-j} * \Delta t_{in,j} \tag{3.5}$$

If the input jitter is independent for each input event, the variance of the output will be less than that of the input. Thus a chain of surfing DLLs is jitter attenuating. For example, the jitter attenuation factor is roughly 0.5 for the example shown in Figure 3.7.

3.2.3 Multiphase Designs

From Equation 3.1, the locking range of the surfing DLL is determined by the delay of the surfing inverter which is in the interval $[d_{\max}, d_{\min}]$ and the delay of the feedback path, D. If a ring oscillator has a half-period less than three inverter delays, then the oscillator output will not have enough time to approach the power rail and its waveform will appear roughly sinusoidal. To make a reliable ring oscillator, D should be greater than $2d_{\max}$. Arbitrarily long periods can be achieved by making D sufficiently large, but the relative locking range is:

$$range \quad = \quad \frac{d_{min}+t_{s,max}+D}{\max(d_{max}+t_{s,min}+D,0)} - 1 \tag{3.6}$$

which diminishes with increasing D. We can extend the operating range of the surfing DLL by connecting the predict signals of multiple surfing inverters into a ring as shown in Figure 3.9. The three channels receive three, evenly spaced clock signals, and generate three evenly spaced clocks as well. Like the single-phase design shown in Figure 3.7, this design is jitter attenuating.

A multiphase, surfing DLL with k phases works for periods ranging from $2k(d_{max}+D+t_{s,min})$ to $2k(d_{min}+D+t_{s,max})$. Thus, the multiphase DLL can achieve a large tracking bandwidth when operating at low clock frequencies. Conversely, the multiphase DLL can operate with very small values of D because the loop of surfing inverters provides enough total delay to ensure stable oscillation.

The multiphase design can also be used to generate closely spaced clock phases as required in various precharged logic families such as OPL [42] and surfing gates [65]. It is difficult and power intensive to generate these phases globally and distribute them through a separate clock network for each phase. This motivates developing ways to locally generate the required clock phases for these logic families. Our multiphase DLL can do just this. For example, a surfing DLL with three channels will divide each clock period into three evenly spaced phases. Our simulations show that even when are connected to the same input source, after several DLL stages, the phases of the three channels are evenly distributed. Using both the rising and falling edges for each channel provides six phases for a three-channel, surfing DLL, and surfing DLL's with more channels can achieve even finer divisions.

Other researchers have proposed ring-oscillators for generating closely spaced clock phases as described in Chapter 2.1 [21, 22, 35, 41]. These prior methods produced free-running oscillators. To the best of our knowledge, our use of surfing to implement a DLL is novel.

We summarize the advantages of the surfing DLL as follows:

1. The surfing inverter combines the function of the variable delay element and phase detector. It is very simple.

2. The surfing design makes use of the fact that for a clock signal, 1s and 0s are interleaving to accurately estimate when the next event should happen.

3. It avoids jitter peaking by event-time averaging.

3.3 Pipelined Clock Forwarding

As noted in section 3.1, simple inverter chains are jitter amplifying and ill-suited for forwarding timing signals such as clocks or asynchronous handshake signals across long distances. DLLs and PLLs are often used to regenerate timing signals for inter-chip communication. However, these circuits require substantial power and area which limits their use for on-chip,

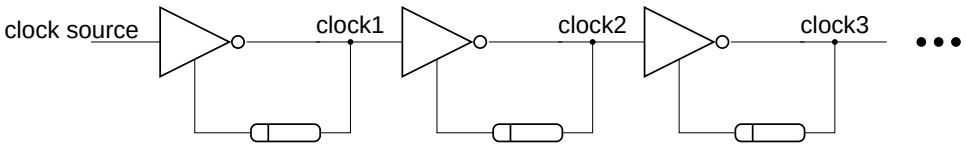


Figure 3.10: The single-phase surfing pipeline timing chain

global interconnect. For example, the DLLs and other phase-recovery circuits in [29] accounted for 90% of the power consumption for a 1Gb/s cross-chip link in a 0.18μm CMOS process. Our simple design uses much less area and power than a traditional DLL and the surfing DLL for the forwarded clock provides jitter attenuation. We now analyze the jitter transfer of this design. In Equation 3.3, $\Delta t_{predict}$ is a delayed version of Δt_{out}. Thus, we can rewrite Equation 3.3 in the Z-transform as follows:

$$\frac{\Delta t_{out}(z)}{\Delta t_{in}(z)} = (1-\alpha) * \frac{z}{1-z} \tag{3.7}$$

This shows that our DLL is a first order stable circuit when $\|\alpha\|$ is less than 1. Although its gain is also less than that of a traditional DLL, it is sufficient for many on-chip applications. These features make surfing DLLs ideal for on-chip clock-forwarding.

We can use the surfing DLL to implement each stage of a clock forwarding network, such as the one shown in Figure 3.10. The minimum period of the clock is limited by the left side of the inequality of Equation 3.1: $P \geq 2 * (d_{max} + t_{s,min} + D)$. If P satisfies that constraint and D is large enough, then unlike the inverter chain, this timing chain can propagate timing pulses through an arbitrarily large number of stages without ever dropping one – the surfing effect works to maintain uniform separation of edges. To obtain a periodic output, the clock's period should not exceed $2 * (d_{min} + t_{s,max} + D)$. At lower frequencies, the chain will propagate clock events without dropping any, but it no longer preserves uniform spacing. Thus, jitter will grow with pipeline length if the clock period is too large. We exploit this in Section 3.5 where we use our surfing design to forward asynchronous handshaking signals for which jitter is not a critical issue.

Due to the surfing effect, the surfing inverter chain is less sensitive to power supply noise than a simple inverter chain. We simulated the inverter chain with a PMOS transistor width of 18.45μm and an NMOS width of 7.1μm. In Figure 3.11, the solid curve is the output of the 200^{th} stage with no power supply noise and the dashed curve is the output of the same stage

but with V_{DD} oscillate in the range of $[1.62V, 1.8V]$ ($1.8V$ is the nominal V_{DD} for the TSMC 0.18μm process). With no power supply noise, the inverter chain can propagate the pulses through 200 stages at 500MHz without losing pulses. However, with V_{DD} varying randomly in $[1.62V, 1.8V]$, the chain loses pulses. We applied the same power supply noise to the surfing inverter chain. Figure 3.12 shows the output of the 200^{th} stage with and without power supply noise. We further simulated a 200-stage chain for 200ns with V_{DD} varying randomly in $[1.62V, 1.98V]$. At each stage of the chain, we measured the cycle-to-cycle variation in the period (the relative jitter). For simplicity, we did not include any branching loads to the data path or for a clock tree for either the surfing or non-surfing chains. Section 3.4 describes a complete surfing link including these branching loads. Figure 3.13 shows the RMS relative jitter along the chain. At the first several stages, the RMS relative jitter is smaller than for the later stages. This is because the input's relative jitter is 0 and the first several stages are mainly affected by the power supply noise. The later stages are not only affected by the power supply noise but also by the input disturbance. After 50 stages, the RMS relative jitter varies in a small range within 10ps. 50 stages later, the RMS relative jitter has a maximum of 7.2% and the standard deviation is 3% of the clock period. The chain shows no jitter accumulation: at each stage, the power supply noise injects new jitter, but the surfing inverter also attenuates its input jitter. These two processes interact to produce a bounded, steady-state jitter throughout the chain.

We now consider jitter propagation in a chain of surfing DLLs. For a single-phase chain, let $t(i, j)$ be the time that stage j outputs the i^{th} clock event. Let $\Delta t(i, j)$ be a disturbance applied to this output, and let α be defined as in Equation 3.2. We note that this disturbance is attenuated by a factor of $(1 - \alpha)$ by the next stage. Thus, $\Delta t(i, j + 1) = (1 - \alpha)\Delta t(i, j)$. Furthermore, this disturbance also affects the predict signal for stage j, and we get $\Delta t(i + 1, j) = \alpha\Delta t(i, j)$. To determine the impact at an arbitrary downstream stage and event, we must account for all paths from $t(i, j)$ to the downstream event. This disturbance affects the $(i + m)^{th}$ event of the $(j+n)^{th}$ stage by propagating forward through n stages and along the out to predict loop m times. Thus, there are $\begin{pmatrix} m+n \\ n \end{pmatrix}$ paths for the disturbance to take, and all of these paths contribute to perturbing $t(i+m, j+n)$. This

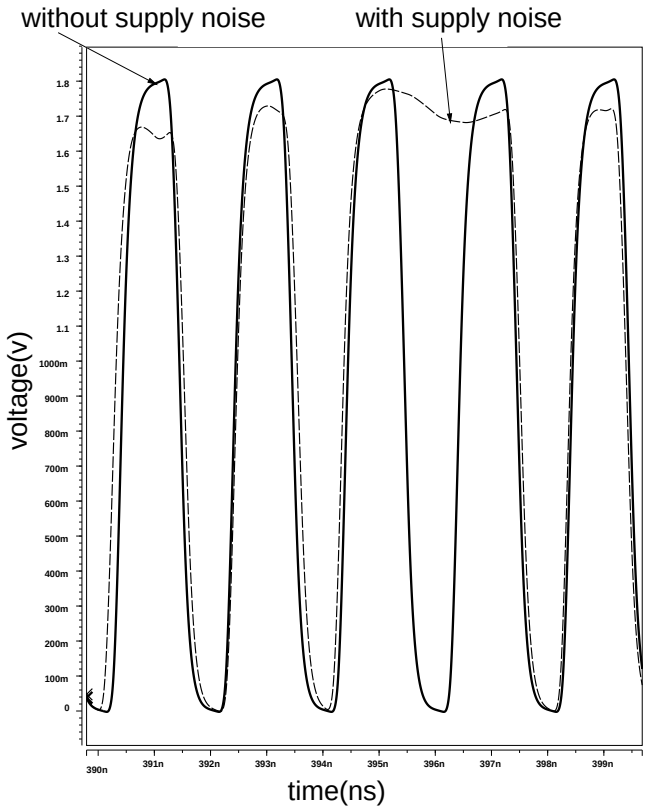


Figure 3.11: The impact of power supply noise on an inverter chain

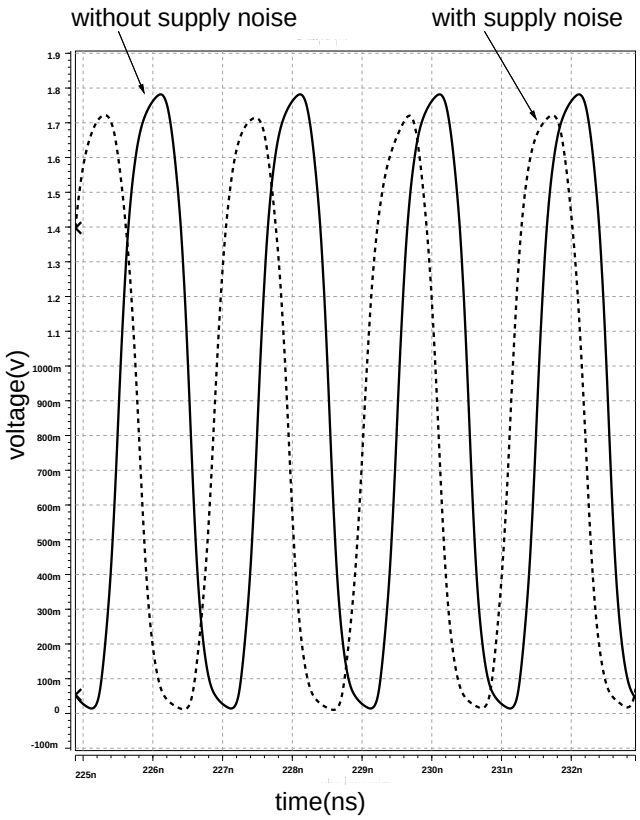


Figure 3.12: The impact of power supply noise on a single-phase surfing pipeline timing chain

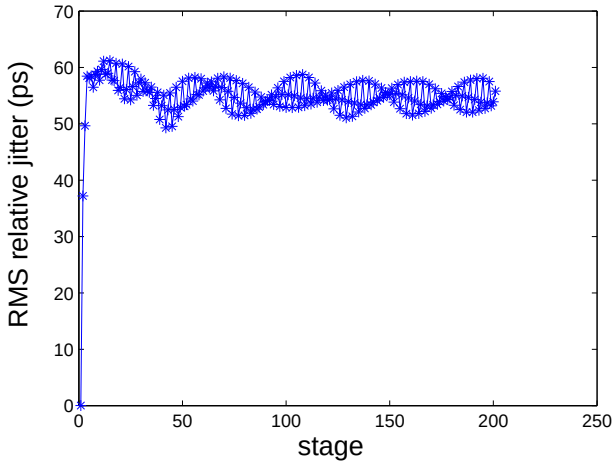


Figure 3.13: RMS relative jitter of a single-phase surfing pipeline timing chain with ±10% power supply noise

yields:

$$\begin{aligned}\Delta t(i+m, j+n) \\ = \quad \begin{pmatrix} m+n \\ n \end{pmatrix} \alpha^m (1-\alpha)^n \Delta t(i,j) \ .\end{aligned} \tag{3.8}$$

We note that for any fixed $m \geq 0$,

$$\sum_{n=0}^{\infty} \Delta t(i+m, j+n) \quad = \quad \Delta t(i,j) \ . \tag{3.9}$$

In words, the sum of the disturbances caused by the disturbance $\Delta t(i,j)$ after m time steps is exactly equal to the original disturbance. However, the disturbance is now spread over $m+1$ stages of the pipeline.

We examined the jitter propagation of a chain of multiphase DLLs using simulations as well. As with the single phase design, we applied a single event disturbance to the chain with the magnitude of the disturbance equal to 10% of the period. With the period equal to 2.3ns, we simulated a 200-stage surfing chain for 250ns. Figure 3.14 shows the relative jitter at every stage. In Figure 3.15 we plot the maximum absolute jitter by comparing the disturbed chain with the response of an undisturbed chain. We simulated the design at the circuit level with HSPICE and using the linearized timing model from Equation 3.2 with Matlab. Both methods show how the jitter

dies out in the pipeline. For the first 100 stages, the circuit and linearized-timing models produce nearly identical results. For longer pipelines, the linearized model shows continuing decrease in the jitter while the HSPICE simulation reaches a floor. We believe that this "floor" simply reflects the quantization errors arising from the size of the HSPICE time steps.

Figure 3.16 plots the maximum absolute jitter against the stage number on a log-log plot. From Equation 3.8, we conclude that as the stage number, n grows large, the peak impact of the input disturbance should occur at time $1 + n * \alpha/(1 - \alpha)$. Using Stirling's approximation, we conclude that the magnitude of the peak disturbance should drop as $n^{-1/2}$. Fitting our simulation data to a curve of the form $a*n^b$, we find that we get an excellent fit with $b = -0.503$. This matches very well with the analytical prediction.

Now, consider the cumulative effect of jitter introduced by each input clock. For simplicity, we assume that each input disturbance is an independent random variable with variance σ_0^2. The total disturbance at stage j has a variance, σ_j^2 and the square root of this variance is the mean jitter at that stage. We applied random jitter on each event of the input clock for a single phase DLL chain. By rewriting Equation 3.3 as follows:

$$\Delta t_{out,i} = \alpha * \Delta t_{out,i-1} + (1 - \alpha) * \Delta t_{in,i} \tag{3.10}$$

we can use this linear model for the delays of the surfing inverter to calculate the disturbance and the mean jitter at each stage. We also estimated the mean jitter by simulating the chain using HSPICE for 250ns. Figure 3.17 and Figure 3.18 shows the attenuation of mean relative and absolute jitter when a surfing DLL chain is driven from a clock with a 2.1ns period and 10% relative jitter. For the relative jitter, the simulation data match very well with the analytical prediction. However, the absolute jitter from the analytical prediction drops more slowly along the stages than the simulation data. We took a closer look at the waveforms of the in, out and predict for the first stage. In Figure 3.19, the thin curves are the nondisturbed signals and the thick curves are the disturbed signals. No signal is disturbed in the 1^{st} event. In the 2^{nd} event, the input is disturbed to come late. The predict for that event however comes earlier. This is because the late coming input will cause the output to come late. Due to this late coming output, the inverter driving the predict signal experiences less Miller capacitance. Thus, the predict signals comes earlier and the delay of this inverter decreases more. This early predict signal reduces the disturbance of the output signal. Conversely, an early coming input will cause a late coming predict signal which also helps to reduce the output disturbance. However, in the linear

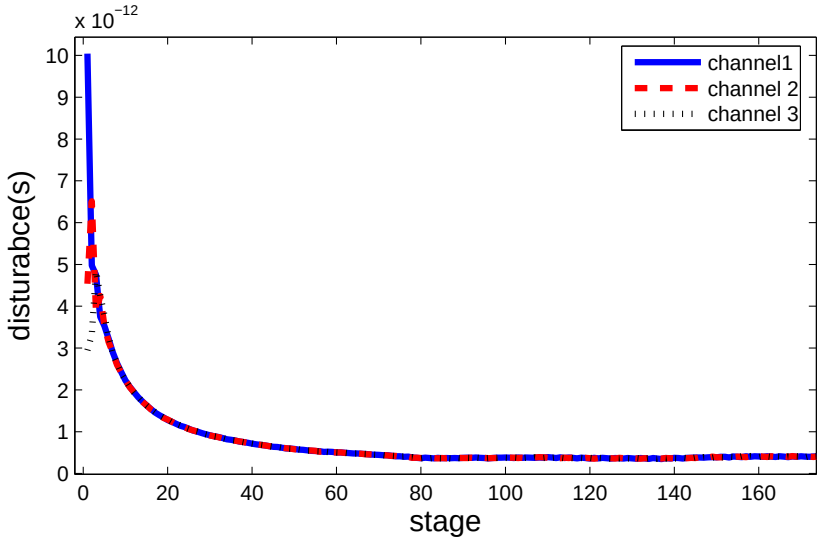


Figure 3.14: The relative jitter with one single event disturbance

model given by Equation 3.10, we assume that $\Delta t_{predict,i}$ equals $\Delta t_{out,i-1}$ and $\Delta t_{predict,i}$ is only affected by $\Delta t_{predict,i-1}$ and $\Delta t_{in,i-1}$. Thus the disturbance of the predict signal in the 2^{nd} event should be zero. Figure 3.19 shows that $\Delta t_{predict,i}$ is also a function of $\Delta t_{in,i}$. However, the linear model in Equation 3.10 does not characterize this second order effect. We revise the linear model as follows:

$$\Delta t_{out,i} = \alpha * \Delta t_{out,i-1} + (1-\alpha) * \Delta t_{in,i} + \gamma * \Delta t_{in,i} \tag{3.11}$$

where the last term on the right side represents the effect of the i^{th} input disturbance on the i^{th} event of the predict signal. Figure 3.20 plots the effect of the disturbance of i^{th} input event on the arrival time of the i^{th} predict signal using HSPICE simulation. Although the arrival time of the predict varies in a narrow range, γ varies between -0.25 to $+0.04$. It is difficult to set a constant value for γ. Instead, we use a small value for γ to show the importance of this term. In Figure 3.18 we also plot the mean jitter using the revised linear model with γ being -0.01. This absolute jitter drops much faster compared with the original linear model. The second order effect contributes significantly to the attenuation of the absolute jitter.

We further compared the inverter chain with the surfing DLL chain with respect to throughput and power consumption. We replaced the surfing DLL

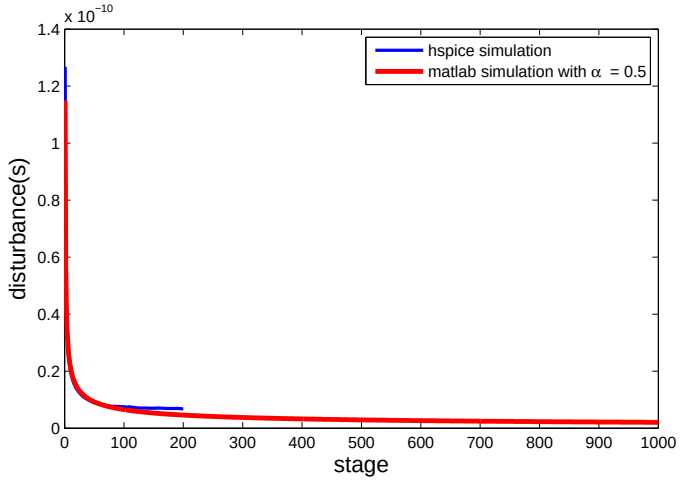


Figure 3.15: The maximum absolute jitter with one single event disturbance (I)

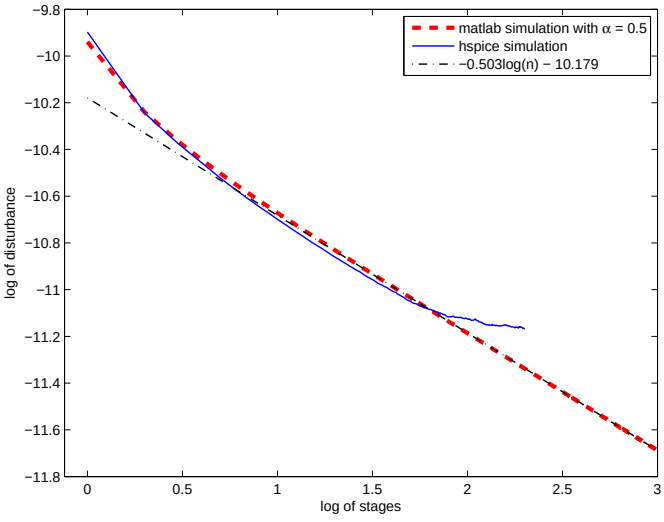


Figure 3.16: The maximum absolute error with one single event disturbance (II)

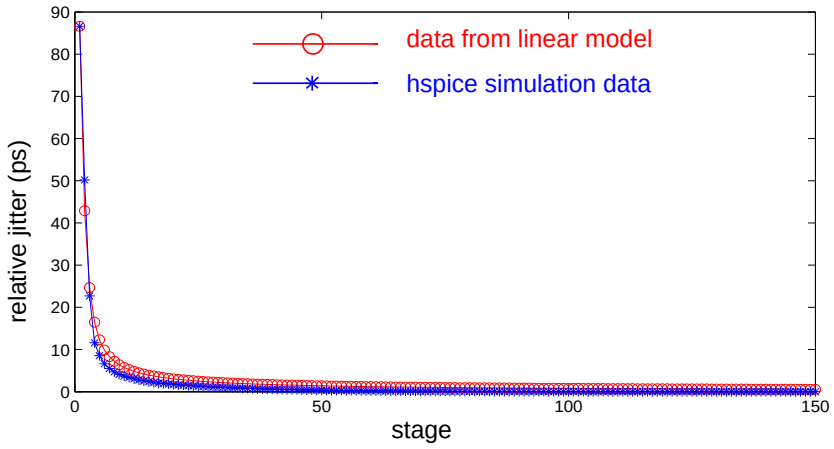


Figure 3.17: Attenuation of relative jitter for a single-phase timing chain

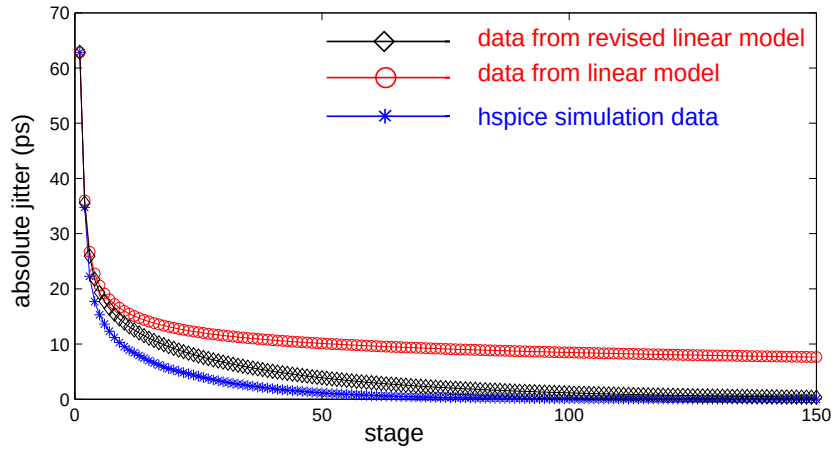


Figure 3.18: Attenuation of absolute jitter for a single-phase timing chain

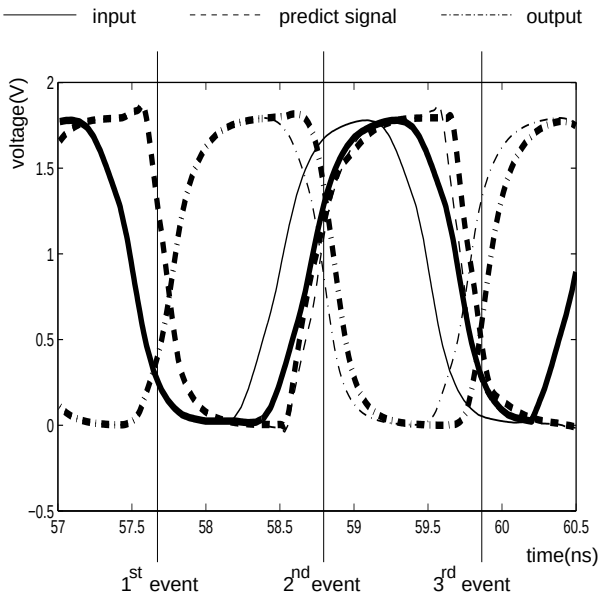


Figure 3.19: Waveform of the disturbed and nondisturbed predict signal

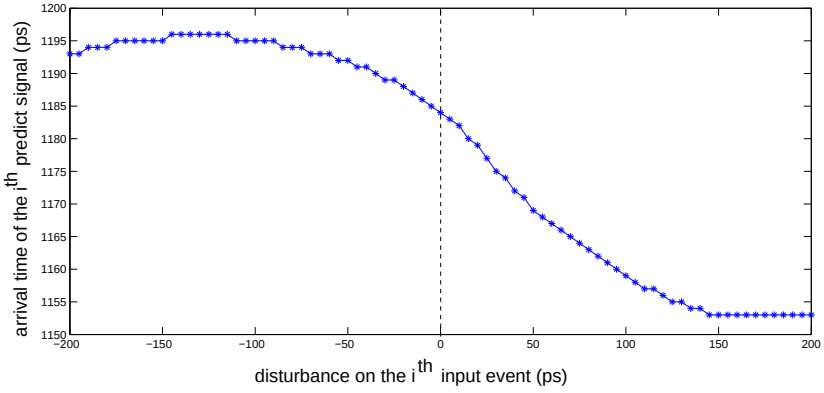


Figure 3.20: Effect of the i^{th} input disturbance on the i^{th} predict signal

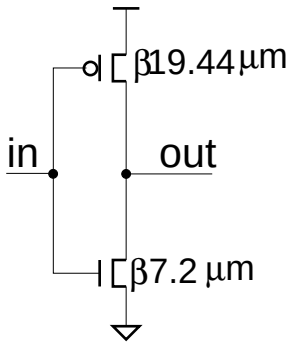


Figure 3.21: Inverter in the inverter line

in Figure 3.10 with inverters as shown in Figure 3.21 to send the strobe signal where β is a coefficient from 0 to 1. The line length between inverters is the same as in Figure 3.10. By varying β, we can trade-off delay and power consumption.

Chains of surfing DLLs can reliably propagate a clock signal through an arbitrarily long chain. On real chips, a strobe chain will have some fixed, finite length. If the chain is short enough, an ordinary inverter chain should suffice. To examine this trade-off, I simulated an approximation of a long chain of inverters connected by long wires using a loop-structure as shown in Figure 3.22. Inverters 1 through 2N form a loop. The idea is to initialize this loop with an even number of transitions (i.e. an equal number of rising and falling transitions) to get the circuit to generate a clock with a desired frequency. If the ring is long enough, this should model a long chain using a much smaller number of inverters, thus making the simulation practical. Delay element d is an HSPICE voltage controlled voltage source that allows the total loop delay to be adjusted to a multiple of the clock period. The delay element does not present a load to the ring; instead inverters L1 through Lm model a terminating load on the ring. Note that inverters 1 through 2N form a ring with an even number of inverters. Thus, for any initial conditions, the ring will *eventually* lock-up in a stable state. I simulated the circuit from Figure 3.22 for 200ns of simulation time for various clock periods with $\pm 10\%$ power supply noise. If the ring could sustain oscillation without dropping any events for this much time, I conclude that an on-chip inverter chain should be a reliable way to propagate a clock of that frequency – it would be very unusual to have an on-chip strobe line with a total latency of 200ns or greater.

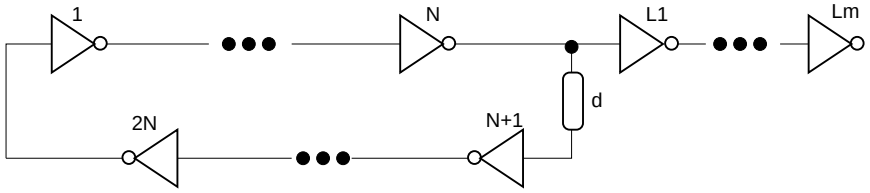


Figure 3.22: Setup for measuring minimum period

Table 3.1: Power consumption of surfing DLL chain and inverter chain

design	delay ps	minimum period, P ns	energy of clock, E pJ/mm/bit	$E * P$ pJ*ns/mm/bit
surfing DLL chain	311	2.1	0.68	1.43
inverter chain ($\beta = 1$)	250	2.5	0.51	1.28
inverter chain ($\beta = 0.79$)	280	2.8	0.48	1.34
inverter chain ($\beta = 0.58$)	340	3.5	0.43	1.50

Our experiments shows that for a simple inverter chain to work reliably, the minimum period should be at least around than 10 times of its stage delay. For an inverter chain with β equal to 1, the delay of that inverter is 250ps and the minimum period it can work is 2.7ns with $\pm 10\%$ power supply variation. With β being 0.58 and 0.79, the inverter's delay is 340ps and 280ps respectively and the minimum period is 3.5ns and 3.0ns. Table 3.1 summarizes the energy consumption of these chains. For the inverter chain, the energy consumption drops as the delay increases. The throughput is proportional to its inverter delay. However, the energy consumption drops slowly due to the increased rising time. Thus the Et (*i.e.* $(E * P)$ metric goes up as β goes down. Due to the surfing effect, the surfing DLL chain can achieve higher throughput. The surfing DLL chain consumes more energy per bit than the inverter chain due to the extra feedback loop. Thus, when comparing the EP metric, surfing DLL chain is not the best among the four designs.

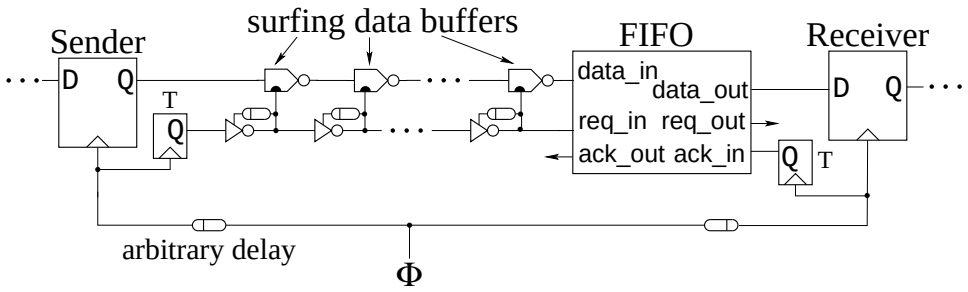


Figure 3.23: Source synchronous surfing

3.4 Source Synchronous Surfing

The jitter attenuating properties of our clock buffer make it ideal for forwarding clock signals in source synchronous interconnect. Figure 3.23 shows such a link, and Figure 3.24 shows the surfing data buffer from [24] that we use in this design. The transistor widths are the same as the corresponding transistors in Figure 3.5. Only the connection of the surfing signal is different.

Our design uses a double-pumped clocking: separate data values are transferred on rising and falling edges of the strobe signal. This allows the strobe to operate at the same transition rate as the data path, thereby raising the maximum throughput and decreasing power consumption. The surfing data buffer requires pulses to enable its tri-state inverter. Thus, we use the self-resetting edge-to-pulse conversion circuit from [24] as shown in Figure 3.25. The numbers on the nodes denote the gate delays from the input. Output nodes are labelled p/q with p being the gate delay from a rising edge of req and q the gate delay from a falling edge of req. This edge-to-pulse converter generates a pulse on fast and $\overline{\text{fast}}$ whenever it sees a rising or falling edge of req. Transistors m1 and m2 are used to detect a rising edge. Conversely, m4 and m5 detect falling edges. Transistors m3 and m6 provides the self-reset. Due to the delay of this circuit, data values surf behind the strobe edges. Like transparent latches, the surfing data path time borrows. Thus, jitter from the edge-to-pulse converter is relatively benign in our design.

The surfing inverter for the data path is very similar to the one used in the strobe path (see Figure 3.5). This similarity makes it straightforward to match the delays of the strobe and data paths. Furthermore, this tracking is preserved extremely well over changes in device parameters, operating

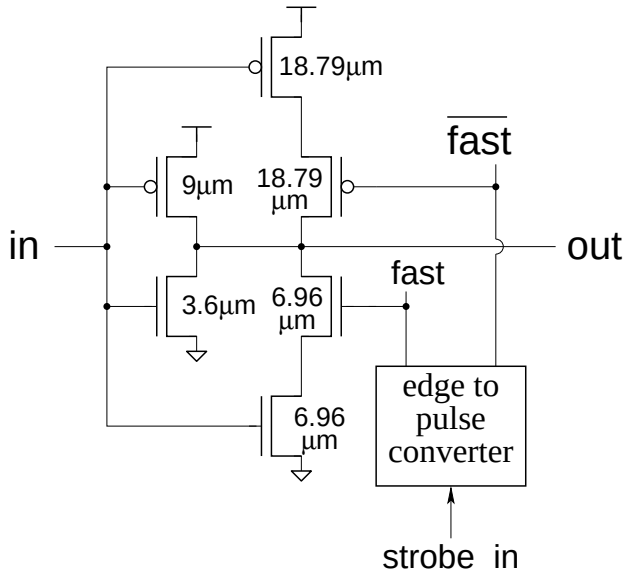


Figure 3.24: Surfing data-path buffer (from [24], Figure 3)

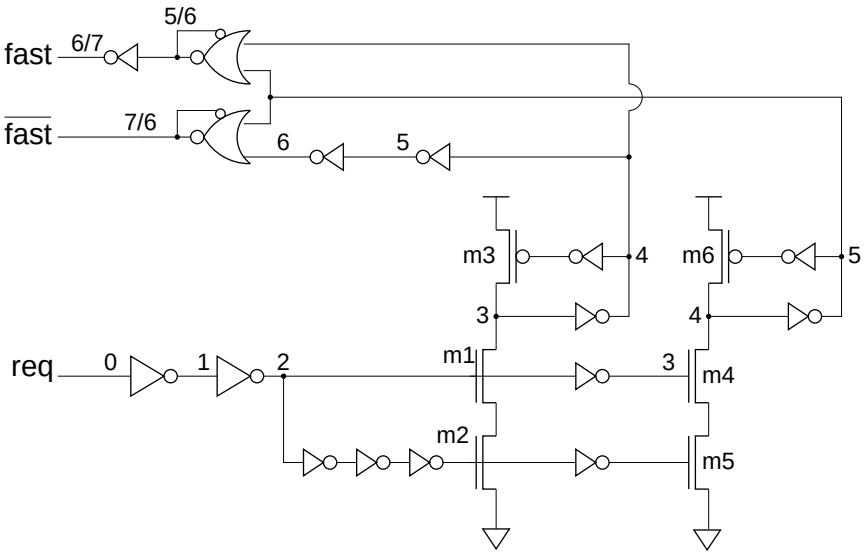


Figure 3.25: Edge-to-pulse converter (from [24], Figure 4)

temperature and V_{DD}. In fact, a desirable feature of our design is that it can be used with V_{DD} scaling in designs that dynamically optimize power versus speed trade-offs.

The design in [24] used a chain of simple inverters to forward the strobe. This technique was limited by the number of stages that could be used before strobe events would be lost due to intersymbol interference. The surfing buffers in the strobe path of our design overcome this limitation. The jitter-attenuation of the surfing buffer ensures that successive edges of the strobe signal remain well-separated. Thus, our design provides reliable communication through an arbitrary number of repeater stages.

We compared the surfing source synchronous with the traditional source synchronous design. For the surfing design, we placed a surfing DLL and a surfing inverter at 2.1mm intervals. For simplicity, we replace the edge-to-pulse converter with two inverters to generate the required fast and $\overline{\text{fast}}$ signals for the data path. Thus the design is not double-pumped; *i.e.* , the bit rate is the same as the frequency of the strobe line. We use the equivalent single-pumped design for the traditional source synchronous communication circuit; so this does not affect the comparison. The DLL chain can work at 2.0ns with ±10% power supply variation. To send one bit, with the DLL chain working at 2.1ns, the strobe line consumes 0.68pJ/mm and data line consumes 0.31pJ/mm per bit.

In the traditional source synchronous design, we use inverters as shown in Figure 3.21 to send the strobe signal where β is a coefficient from 0 to 1. Latches are inserted in the data line to align the data signal with the strobe signal. In the data line as shown in Figure 3.26 and 3.27, every 4.2mm, a latch is added into the data line to synchronize the data signal with the strobe signal. The delay variation of the data line in the surfing source synchronous communication is ±18%. In the traditional source synchronous design, the transistors in the data line are sized to minimize the energy consumption such that the delay of the data line is less than or equal to 82% of the delay of the strobe line. We use the same transistor delay and energy model as in [24]. Based on the sizing offered by the optimization model, we adjusted the transistor sizing with HSPICE simulation to further reduce the energy consumption. With β being 0.58 or 0.79, only two inverters N1 and N2 (Figure 3.26) is needed to meet the delay constraint. T1, N1 and N2 are 2.7μm, 7.5μm and 9.1μm given β equal to 0.58. T1, N1 and N2 are 8μ m, 10.4μm and 12μm respectively when β is 0.79. When β is equal to 1, one more transistor is needed to reduce the delay and the transistor sizes are given in Figure 3.27.

Table 3.2 extends Table 3.1 and summarizes the energy consumption of

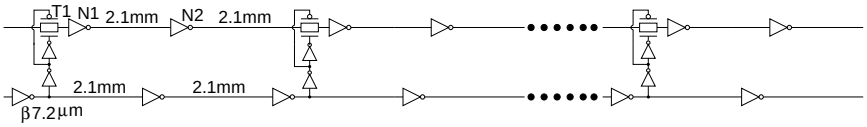


Figure 3.26: Source synchronous communication (β being 0.58 and 0.79)

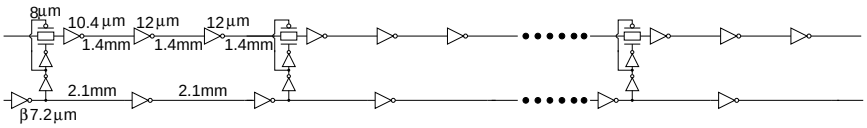


Figure 3.27: Source synchronous communication ($\beta = 1$)

different schemes. In The DLL chain consumes more energy per bit than all the traditional schemes. This is also true for the surfing data line. However, surfing makes the design of the data line simple and the power consumption of the data line is less than that of the surfing DLL chain. In the traditional source synchronous communication scheme, the design of the data line is more complex than the clock line which results in more power consumption in the data line than in the clock line. Thus when comparing the Et (*i.e.* ($E_s + E_d$)*P) metric, surfing communication is close to the best design of the traditional scheme.

3.5 Surfing Handshakes

Figure 3.28 shows a typical asynchronous interface with bundled completion. In standard implementations, each request event from the producer

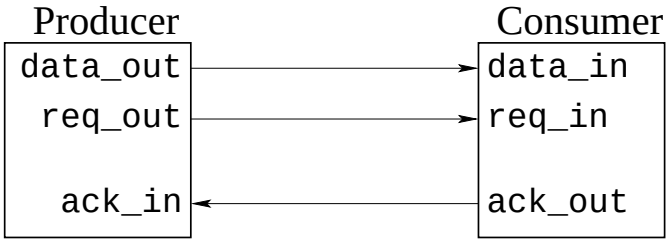


Figure 3.28: Asynchronous handshaking

Table 3.2: Power consumption of source synchronous communication schemes

design	delay ps	minimum period, P ns	Energy of strobe, E_s pJ/mm/bit	Energy of data, E_d pJ/mm/bit	$(E_s + E_d)$*P pJ*ns/mm/bit
surfing	311	2.1	0.68	0.31	2.08
traditional scheme 1 ($\beta = 1$)	250	2.5	0.51	0.31	2.05
traditional scheme 2 ($\beta = 0.79$)	280	2.8	0.48	0.28	2.13
traditional scheme 3 ($\beta = 0.58$)	340	3.5	0.43	0.25	2.38

must be acknowledged by the consumer before the next data value can be sent. The throughput of such a link is constrained by the round-trip time for the producer, the consumer and the wire delays between them. For long-wire communication, these delays can be large, seriously degrading the performance of the interface. These overheads can be mitigated somewhat by breaking the long wires into shorter segments and placing a handshaking buffer between each pair of successive segments [40]. This reduces latency by avoiding the quadratic delay growth of long wires. Throughput also increases because the asynchronous buffers provide data storage and pipelining; many values can be in flight between the producer and consumer at the same time. The disadvantage of this approach is that the asynchronous buffers introduce a latch at each stage, increasing the area and power consumption of the design.

We now show how a credit-based flow control (aka "sliding window" [36, p. 217]) scheme can be implemented with surfing buffers to overcome the limitations of asynchronous, global signaling [20]. The basic idea behind credit-based flow control is simple. Initially, the consumer has the buffer capacity to receive k data values. The producer starts with k credits. Each time the producer transmits a value, it uses a credit and decrements its credit count accordingly. Conversely, when the producer receives an acknowledgment from the consumer, the producer increments its credit count. Thus, the producer may send up to k values before it receives an acknowl-

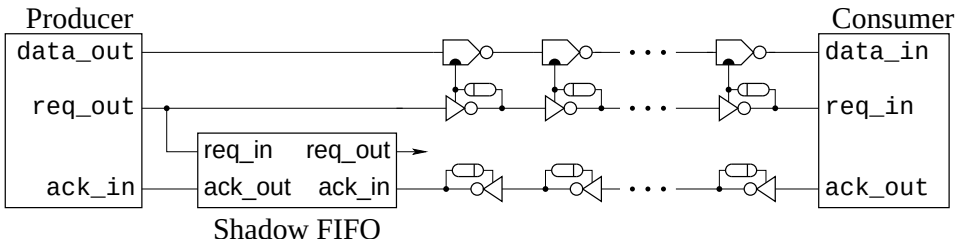
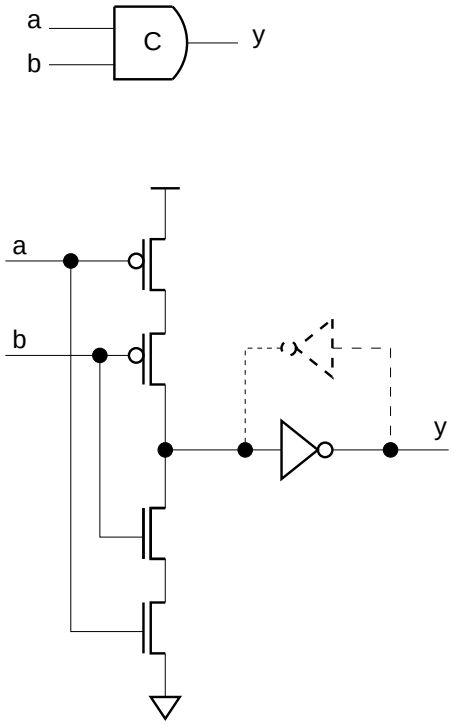


Figure 3.29: Credit-based surfing

edgment from the consumer; the consumer is guaranteed to have space to receive them. If k is sufficiently large, the link can operate at the maximum throughput of the producer and consumer without limitations from the wire delay. In our design, the "shadow FIFO" is an asynchronous ripple FIFO that holds no data. Instead, it is simply a chain of handshaking stages. Such a "FIFO" can be implemented as a chain of Muller-C elements as shown in Figure 3.30. As shown in the truth table, when the inputs to the Muller-C element agree, the output will transit to the same value as the input. Figure 3.31 gives an example of a FIFO using the Muller-C element. If $X_i = X_{i+1}$, we say that stage i is *empty* and that it holds a *bubble*. Conversely, if $X_i \neq X_{i+1}$, then we say that stage i is *full* and that it holds a *token*. Note that tokens propagate from req_in to req_out while bubbles propagate in the opposite direction, *i.e.* from ack_in to ack_out. If the consumer has an initial buffer capacity to receive k data values, then the shadow FIFO should have at least k stages and be initialized to hold k bubbles.

Figure 3.29 shows our surfing implementation of a credit-based scheme. The "shadow FIFO" holds no data. If the consumer has an initial buffer capacity to receive k data values, then the shadow FIFO is initialized to hold k bubbles. Thus, the producer may transmit up to k values before it receives an acknowledgment from the consumer. Each time the producer sends a value, it inserts a token into the shadow FIFO and thereby removes a bubble. Conversely, receiving an acknowledgment removes a token from the shadow FIFO and inserts a bubble. Because the producer consumes a credit from the shadow FIFO before the corresponding data value arrives at the consumer, the number of bubbles in the shadow FIFO is always less than or equal to the remaining capacity of the consumer to accept data from the producer. This ensures that the link neither drops nor duplicates data.

Surfing serves two functions in this design. First, we note that several



a	b	y
0	0	0
0	1	no change
1	0	no change
1	1	1

Figure 3.30: The Muller-C element: symbol, implementation and truth table (from [49], Figure 2.7)

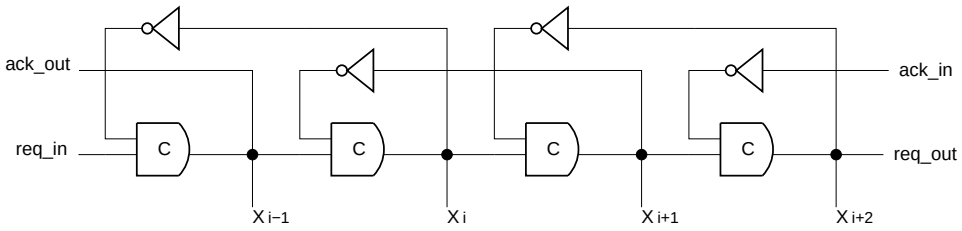


Figure 3.31: A four-stage Muller pipeline (from [49], Figure 2.7)

requests can be in flight from the producer to the consumer at the same time and likewise for acknowledgments. If ordinary inverters were used for repeaters, then consecutive edges of these signals could propagate at different rates. For example, consider what happens if the producer sends a burst of data values after a relatively long pause. Due to drafting, the edges for the later request events will propagate faster than the first edge. Thus, the second edge could catch up with the first edge and cause the link to loose both edges. Ordinary inverters cannot provide reliable forwarding when multiple handshaking events are simultaneously in flight.

The surfing design maintains a minimum separation between edges at the point where the propagation delay is minimized. If an edge occurs later than this separation, the surfing effect will be strengthened, and that edge will be accelerated at subsequent stages. Conversely, if an edge occurs earlier than this separation, the surfing effect will be weaker, and the early edge will be retarded at subsequent stages. Thus, surfing ensures that a *minimum* edge separation is maintained as events propagate through chains of repeaters. This ensures that no edges are lost even though the asynchronous design may operate with successive request or acknowledge events separated by more than the surfing DLL lock limit. Therefore, our surfing design can forward a strobe through an arbitrarily large number of stages and is guaranteed to deliver all edges.

The second function of surfing is to maintain the bundling relationship between the request signal and the data. Here, we use the surfing data buffer originally proposed in [24]. The design described in [24] only considered source synchronous designs and used ordinary inverters to buffer the strobe signal. Our present design extends this to asynchronous communication.

We tested this approach with HSPICE simulations. We implemented producer and consumer modules that can vary their delays to allow the link to operate at full bandwidth or to be limited by handshakes at either end. The data and request paths from the producer to the consumer consist of 32 wire segments and thus 31 surfing repeaters each. The acknowledgment path consists of 32 wire segments and 31 surfing repeaters. The consumer has an input FIFO that is initially empty with a capacity to hold 17 values; accordingly, the shadow FIFO can hold up to 17 tokens. We included simulations where the producer and consumer each occasionally stall for a prolonged period and then resume full-speed operation. In this way, we showed that our link can operate at full speed, with varying handshake cycle times and with bursts. Figure 3.32 shows waveforms from one of these simulations where the response time of the producer and consumer varied randomly from roughly 100ps to about 5ns. In the figure, traces labeled $\mathsf{P}.x$ denote

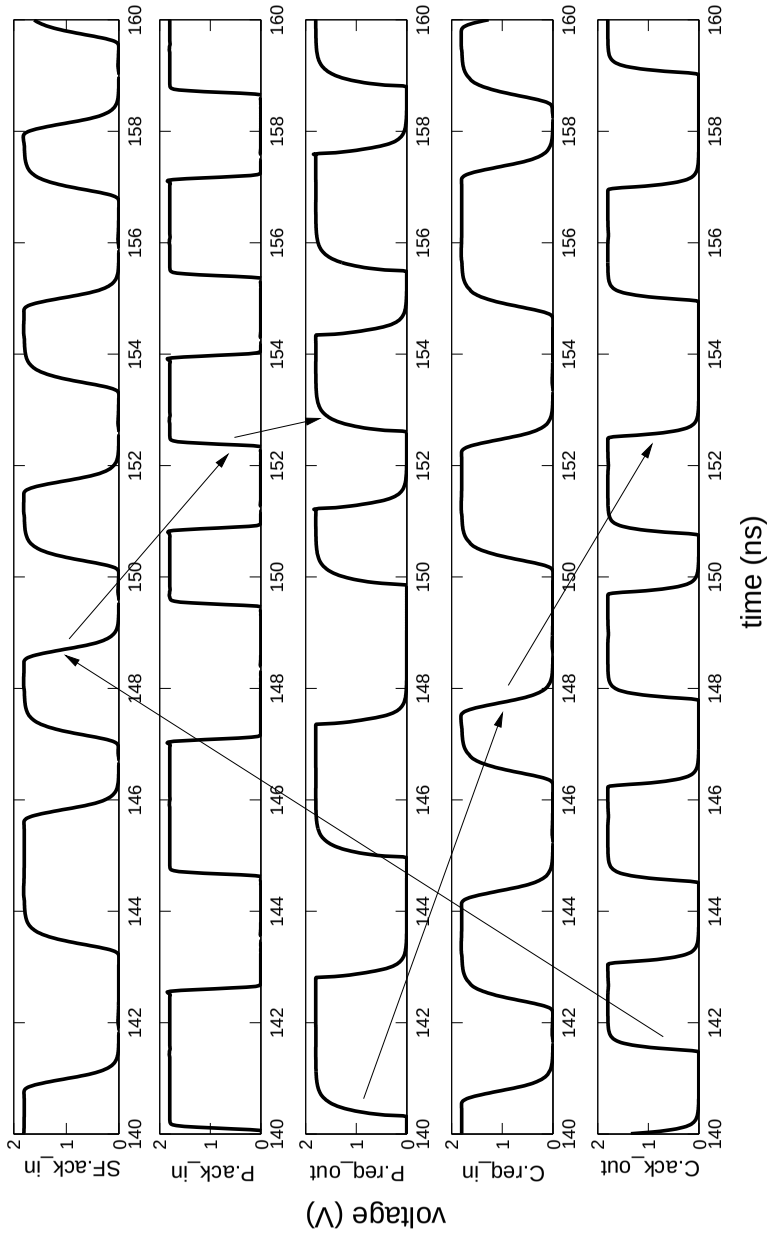


Figure 3.32: Simulation of asynchronous link with aperiodic handshaking.

signals at the *producer's* end of the link; traces labeled C.x denote signals at the *consumer*'s end; and SF denotes the shadow FIFO. These traces show that the surfing control path operates reliably with highly aperiodic signals.

In Figure 3.33, we set the producer and consumer delays to be at their minimums but occasionally stall. Here, we see 15 events at C.req_in after C.ack_out stalls, showing that the link supported 15 simultaneous data and acknowledgments in flight. This is two less than the capacity of the consumer and shadow FIFOs; the remaining two credits cover the forward latency of the consumer's FIFO, thus maximizing the links throughput. When neither the producer nor consumer are stalled, the link transfers data at 1.02ns per data value.

In our simulations, we used a simple C-element chain as shown in Figure 3.30 to implement the shadow FIFO. We modeled wire-segments of length 2.1mm between surfing stages with three RC segments. We used two-phase handshaking to minimize the number of events transmitted on the request and acknowledge wires. We simulated our design using parameters for the TSMC 0.18μm process. The shadow FIFO has 17 stages, and we initialize it to be empty (i.e. it initially holds 17 bubbles). Furthermore, we included extra delay in the first stage (closest to the producer), to ensure that successive request events have adequate separation to allow reliable surfing – 915ps for our design.

We modeled the consumer with another FIFO. Note that the consumer must be ready to quickly accept incoming data for which it has claimed to have capacity. We model varying response times for the consumer by the time it takes to *remove* values from the consumer's FIFO. Thus, if the consumer is slow to remove data, the link will continue to operate at full-speed until the outstanding credits are consumed. In this arrangement, the consumer outputs an acknowledge event each time it removes a value from its input FIFO.

The best previous asynchronous communication method that we know of is the twin-control path design reported by Ho *et al* [25]. For 2.1mm wires, they report a throughput of 1GHz, the same as our design. However, the latency of our design is roughly the same as that of the source-synchronous design described in [24] and therefore about 30% lower than the twin-control path approach.

Finally, we note that using a sliding window protocol offers an additional opportunity for reducing power consumption. It is no longer necessary to acknowledge individual data transfers. Instead, the consumer can acknowledge every third, fourth, or greater transfer. The producer treats each acknowledgment as multiple credits. This reduces the power consumption of the

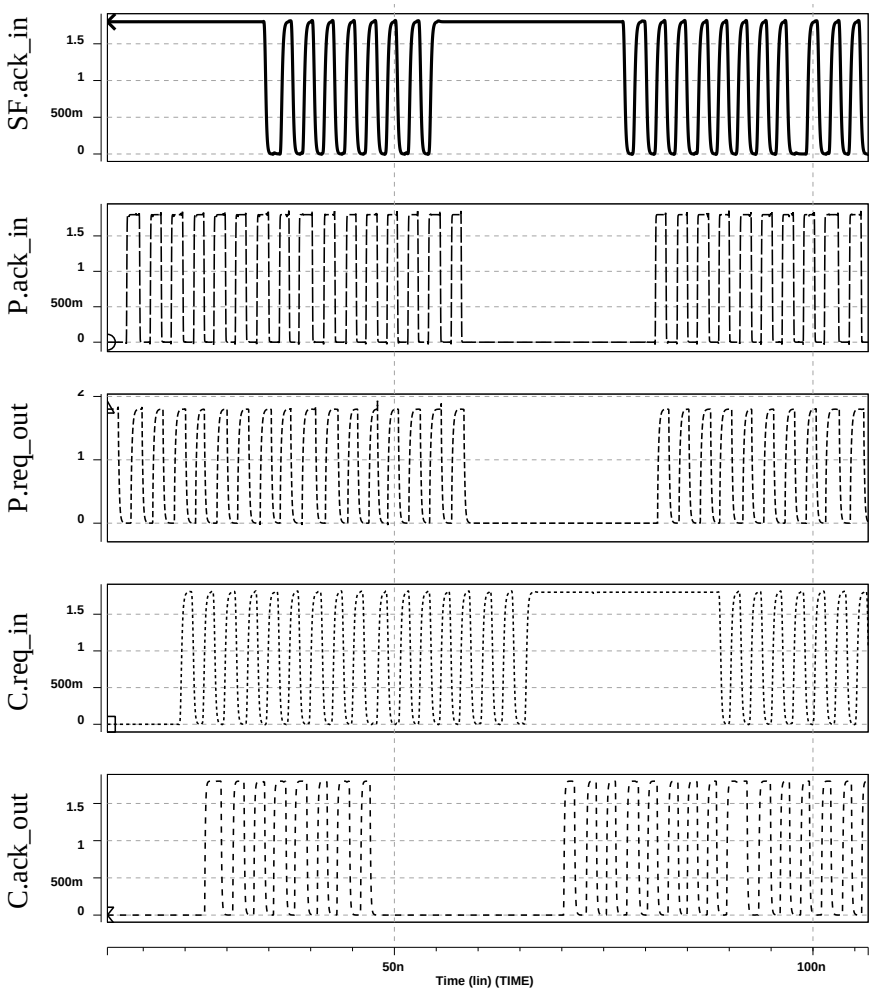


Figure 3.33: Simulation of asynchronous link with bursts

acknowledge path by the same factor. Likewise, the shadow FIFO becomes smaller, but the consumer needs slightly greater buffering capacity to support the same throughput. Exploring the details of these trade-offs is a topic for future work.

3.6 Summary

We have shown a jitter attenuating buffer. Unlike simple inverters that amplify jitter due to intersymbol interference, our circuit implements a low-gain DLL that reduces jitter. This makes our design well-suited for conveying timing signals for cross-chip communication.

The jitter attenuating buffer consists of an inverter with variable drive strength. The variable strength is used to implement the controlled delay variations required for surfing circuits. When used in a DLL configuration, the output time of this surfing inverter is a weighted average of the arrival time of the input clock and the predicted time for the next event. In Section 3.2, we showed that this averaging avoids the problems of "jitter peaking" that are typically associated with DLLs.

We then showed in Section 3.3 how these surfing DLLs can drive long wires to build chains connected to propagate timing signals for cross-chip communication. An analytical model based on a linear approximation of the timing shows that disturbances are spread out over the pipeline, and this results in an attenuation of random input jitter. Simulation results confirmed this analysis and showed that these timing chains are robust in the presence of other disturbances such as power supply noise.

We demonstrated the applications of these timing chains for source-synchronous and asynchronous communication. For the latter, we showed how our surfing timing chains can be used with surfing repeaters for data to provide robust, asynchronous, wave pipelining. In particular, we showed that long-distance communication can be implemented using our techniques to implement a sliding window protocol for handshaking. This allows multiple data transfers to be simultaneously in flight. Our design achieves high throughputs without the high latency overhead that other asynchronous methods incur from using latches in every repeater.

The designs that we presented use surfing inverters where the surfing effect provides a delay variation of roughly $\pm 18\%$ around the nominal delay. This ensures that our surfing designs can compensate intra-chip variations in device parameters, V_{DD} and temperature (i.e. on-chip PVT). Greater range of operation can be obtained by increasing the size of the tri-state inverter

relative to the simple inverter in Figure 3.5 at a cost of an increase in the overall delay of each surfing inverter. Alternatively, one could incorporate a single, traditional DLL onto the chip to set a reference voltage or current for the delay elements of all of the surfing inverters. Although the various delay chains will not be exactly matched, the surfing design should provide enough tolerance to compensate for on-chip PVT variations. The surfing DLLs have enough range to compensate for on-chip PVT variation, and the traditional DLL would compensate for global variations.

Chapter 4

Surfing LC Interconnect

This chapter presents the surfing transmission line where the line inductance (L) plays an important role in signal propagation. Different from the RC mode interconnect, the propagation speed of the LC mode interconnect is determined by $\frac{1}{\sqrt{LC}}$, the speed-of-light in the media. Line resistance both attenuates and distorts the signal. Active compensation is needed for long distance communication. Previous published active compensation methods uses negative impedance [11, 23, 60] to compensate the energy loss in the line. In 2001, Wood *et al.* [60] proposed a traveling-wave oscillator which uses cross-coupled inverters to compensate for energy loss. We open the loop in this traveling-wave oscillator to transmit data signals. By opening the loop, we produce a line that transmits data at the speed-of-light where the cross-coupled inverters contribute energy to transitions, compensating for the resistive losses of the line.

This chapter starts with an introduction of the transmission line. Section 4.2 analyzes the traveling-wave oscillator. We apply the surfing technique to the transmission line and present two kinds of surfing interconnect: a full-swing design and a low-swing design. Surfing performs two important functions in these designs. First, it compensates for the high frequency losses of the line. Second, surfing keeps data transitions aligned with a strobe which allows simple timing recovery at the consumer without any need for PLLs or DLLS.

4.1 Introduction to Transmission Line

A transmission line can be modeled with a distributed RGLC model as shown in Figure 4.1 which is characterized with line conductance G, inductance L, resistance R and capacitance C. Denote the voltage at distance x along the transmission line as $V(\omega, x)$ and the current as $I(\omega, x)$ where ω is the radian frequency. The voltage and current drop along the line are characterized by the following partial differential equation:

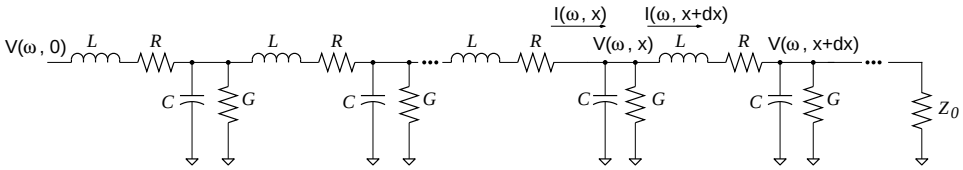


Figure 4.1: Model of a transmission line.

$$\begin{aligned} -\frac{\partial I(\omega,x)}{\partial x} &= V(\omega,x)(G+j\omega C) \\ -\frac{\partial V(\omega,x)}{\partial x} &= I(\omega,x)(R+j\omega L) \end{aligned} \tag{4.1}$$

From Equation 4.1, we get $\frac{\partial^2 V(\omega,x)}{\partial x^2} = V(\omega,x)(G+j\omega C)(R+j\omega L)$ which yields:

$$V_{\omega,x} = V(\omega,0)e^{-Ax} \tag{4.2}$$

where

$$A = [(G+j\omega C)(R+j\omega L)]^{\frac{1}{2}} = \alpha(\omega) + j\beta(\omega) \tag{4.3}$$

The real part of A, $\alpha(\omega)$, is the attenuation rate of a data signal with radian frequency ω, and β, is the angular propagation velocity. The propagation velocity of a signal at radian frequency w is given by

$$v(\omega) = \frac{\omega}{\beta(\omega)} \tag{4.4}$$

For digital signaling, we are most concerned about the attenuation rate and propagation velocity as functions of frequency. Squaring the middle and right terms of Equation 4.3 yields:

$$\begin{aligned} \alpha(\omega)^2 - \beta(\omega)^2 &= RG - \omega^2 LC \\ 2\alpha(\omega)\beta(\omega) &= \omega(RC+GL) \end{aligned} \tag{4.5}$$

Solving for β according to the second part of Equation 4.5 and substituting into Equation 4.4 yields:

$$v(\omega) = \frac{2}{RC+GL}\alpha(\omega) \tag{4.6}$$

Thus, we see that the attenuation rate, α, and velocity, v remain in a fixed relationship as frequency changes.

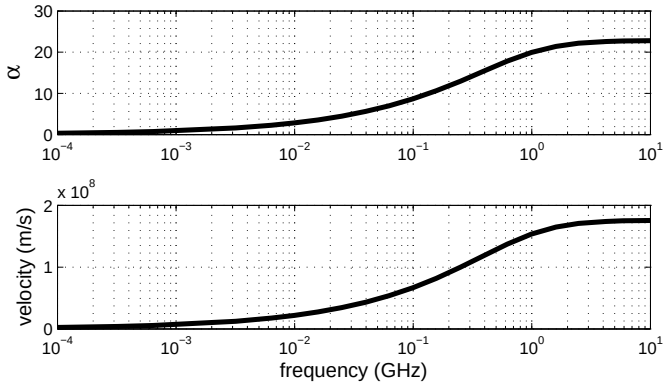


Figure 4.2: Attenuation rate and velocity of a transmission line.

The dependence of attenuation and velocity on frequency can be understood by considering the limiting cases of $\omega = 0$ and $\omega = \infty$. For $\omega = 0$ (the DC limit), we define:

$$\begin{array}{rcccl} \alpha_0 & = & \alpha(0) & = & \sqrt{RG} \\ v_0 & = & v(0) & = & \frac{2}{C/\gamma + L\gamma} \end{array} \tag{4.7}$$

where $\gamma = \sqrt{G/R}$. At low frequencies, attenuation is determined by the ratio of line resistance to conductance. Velocity is determined by the time constants formed by the capacitance and inductance with γ, a sort of geometric mean of the line resistance and conductance.

For the limit with $\omega \to \infty$, we define:

$$\begin{array}{rcccl} \alpha_\infty & = & \lim_{\omega\to\infty} \alpha(\omega) & = & \frac{1}{2}\left(\frac{R}{Z_0} + GZ_0\right) \\ v_\infty & = & \frac{1}{\sqrt{LC}} & & \end{array} \tag{4.8}$$

where $Z_0 = \sqrt{L/C}$ is the characteristic impedance of the corresponding lossless transmission line (i.e. R and G both zero). Thus, at high frequencies, velocity is determined by the line inductance and capacitance, in other words, it approaches the speed-of-light in the dielectric. Attenuation is determined by the ratio of the characteristic LC impedance of the line to the resistive and conductive losses. Between these two limit cases, the attenuation rate monotonically increases from α_0 to α_∞, and likewise the velocity monotonically increases from v_0 to v_∞.

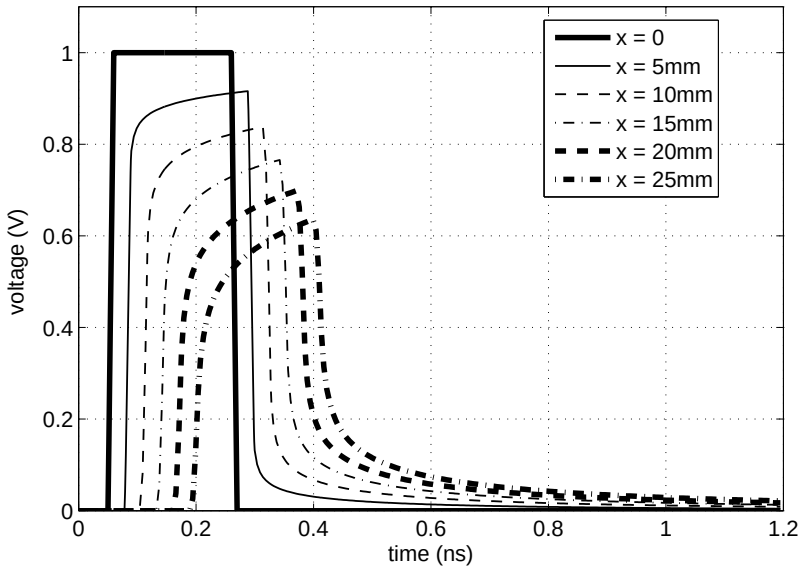


Figure 4.3: Propagating a pulse along a transmission line.

As an example, Figure 4.2 plots the relationship between α, v and ω for a line with $L = 311$nH/m, $C = 104$pF/m, $R = 2.5$kΩ/m, and $G = 0$. Note that in reality, the resistance R will increase as the frequency increases due to skin effect and proximity effect. We use a constant R to simplify the analysis. Observe that high frequency signals propagate faster than low frequency ones, but high frequency signals also suffer greater attenuation. The curves for attenuation and velocity have the same shape as noted following Equation 4.6. The frequency dependence of velocity and attenuation distort high data-rate signals propagating on lossy transmission lines. Figure 4.3 illustrates this by plotting the waveform for a narrow pulse propagating along the same lines as from Figure 4.2. Successive curves of the plot show the waveform at 5mm intervals along the line. As the pulse propagates, it becomes wider and shorter, thus decreasing both the eye-width and eye-height at the far end of the line. If skin effect is included in the model, these distortions become even more severe. Thus, preserving the shape of data pulses is a primary concern for designing high bandwidth, on-chip interconnect.

One approach to preserving pulse shape is to deliberately introduce shunt

conductances to make the propagation velocity independent of frequency. In particular, if $G = RC/L$, then Equation 4.5 is solved by

$$\begin{array}{rcccl} \alpha & = & \sqrt{RG} & = & \frac{R}{Z_0} \\ \beta(\omega) & = & \omega\sqrt{LC} & & \end{array} \tag{4.9}$$

and thus

$$v = \frac{1}{\sqrt{LC}} \tag{4.10}$$

Note that α and v are independent of ω which means that pulse shape is preserved along the line. This is the idea behind the surfliner method [18, 68] (see Chapter 2.2). Observe that α is *twice* the value of α_∞ for the corresponding line without the shunt conductances. This means that the data signal is attenuated at twice the rate of that for a line without shunts. Thus, this method is only useful for relatively short lines unless repeaters are used which would greatly increase the total latency of the interconnect.

If active circuitry is used for the shunts, then the effective value of G can be negative. We can again make the propagation velocity independent of frequency. In particular, if the following constraint is satisfied:

$$G = -RC/L \tag{4.11}$$

then Equation 4.5 is solved by

$$\begin{array}{rcl} \alpha & = & 0 \\ \beta(\omega) & = & \omega\sqrt{\frac{R^2C}{\omega^2 L} + LC} \end{array} \tag{4.12}$$

which yields

$$\begin{array}{rcl} v(\omega) & = & \left(\frac{R^2C}{\omega^2 L} + LC\right)^{-1/2} \\ v_\infty & = & \frac{1}{\sqrt{LC}} \end{array} \tag{4.13}$$

Signals at all frequencies propagate without attenuation, and the velocity approaches $\frac{1}{\sqrt{LC}}$ for $\omega \gg \frac{R}{L}$. We can generalize the condition that $G = -RC/L$ as an energy balance requirement and obtain:

$$\frac{G \int V(t)^2 dt}{C \int V(t)^2 dt} = \frac{R \int I(t)^2 dt}{L \int I(t)^2 dt} \tag{4.14}$$

where $V(t)$ is the voltage across G and C and $I(t)$ is the current through L and R. If $V(t)$ and $I(t)$ are high frequency signals, V(t) can be approximated with $\sqrt{\frac{L}{C}} I(t)$. Then the constraint becomes:

$$-G \int V(t)^2 dt \geq R \int I(t)^2 dt \tag{4.15}$$

In English, this says that in order to propagate high frequency signals, the energy produced by the active conductance must be greater than or equal to the energy consumed by line resistance.

4.2 Analysis of the Traveling-Wave Oscillator

Wood *et al.* [60] added distributed cross-coupled inverters into a transmission line as negative conductance shunts to compensate for energy loss. Figure 2.16 illustrates such a traveling-wave oscillator. For their application, large inverters were used to obtain practical operating frequencies and they did not provide a detailed analysis of the conductance needed for correct operation. Previously published work only address the negative conductance [11, 23] required for small signal analysis. We will demonstrate later that small signal analysis is not enough for the traveling-wave oscillator. For our design, smaller inverters are desirable. This section provides an analysis of the traveling-wave oscillator. After analyzing the closed-loop traveling-wave oscillator, we demonstrate that the same analysis can also be applied to open-loop transmission lines with negative conductance shunts. We use the open-loop transmission line to transmit data signal where the cross-coupled inverters are used to restore data signals.

Let C_{inv} denote the capacitive load added by the inverters every Δx units of length. The line parameters are R_w, C_w and L_w respectively. We start by rewriting Equation 4.11 as

$$\frac{\mid (G_p + G_n) \mid}{C_{inv}/\Delta x + C_w} \geq \frac{R_w}{L_w} \tag{4.16}$$

Where G_p and G_n are the conductances introduced by the NMOS and PMOS devices in the cross-coupled inverters. It is important to note that for the cross-coupled inverter pair, G_p and G_n are both functions of the terminal voltages of transistors. When the cross-coupled inverter pair is not in the amplifying mode, the pair resists changes of its state, and the line must provide energy in order to flip the state of cross-coupled pairs. Once the inverter pair enters its amplifying mode (gain mode), it injects energy back into the line. The energy injected by the inverters is predominantly at high frequencies. Because the conductance of cross-coupled inverters, G, the summation of G_p and G_n, is a function of the line voltage, $V(t)$, and it is time varying. For simplification, we denote $G(V(t))$ as $G(t)$. Multiplication of *G(t)V(t)* in the time domain becomes convolution between $G(\omega)$ and $V(\omega)$ in the frequency domain. The convolution enables the energy transformation

between low frequency and high frequency components of the signal. The simple inequality 4.11 cannot be applied to the transmission line with time varying conductance. This complicates the frequency domain analysis.

To overcome these problems, I developed an empirical approach for modeling transmission line with regeneration shunts. Section 4.2.1 presents the results of simulation experiments that motivate this method. Section 4.2.2 compares the difference between small signal analysis and large signal analysis. With full voltage swing, small signal analysis is not enough to characterize the behaviour of cross-coupled inverters in the transmission line. Instead, we use the average conductance and capacitance to characterize the cross-coupled inverters. Section 4.2.3 introduces our measurement of the effective capacitance presented by the cross-coupled inverters and Section 4.2.4 presents our model to calculate the effective conductance of the cross-coupled inverters.

To provide a concrete example, the following analysis used parameters from the TSMC $90nm$ CMOS process. V_{DD} is $1V$. $L_w = 311$nH/m, $C_w = 104$fF/mm and segment length Δx is 0.1mm. The nominal resistance of the copper line is $2.5k\Omega$ per meter without considering skin effect which corresponds to an $8\mu m$ wide wire on top-level metal. For simplicity, we ignore the skin effect resistance in this chapter to derive a simple model to calculate the effective conductance presented by the cross-coupled inverters. We ran additional simulations with other values for the wire resistance to get a better understanding of this circuit. In this section, we define a basic sized inverter which has a 2μm wide NMOS and a $5\mu m$ wide PMOS transistors. Both transistors are of minimum length (*i.e.* 90nm).

4.2.1 Behaviour of the Traveling-Wave Oscillator

In this section, I use 5 sets of basic inverter pairs and the loop is 2.4mm long. With the line parameters, L_w and C_w fixed, I increased the resistance per meter R_w gradually to show how line resistance affects the behaviour of traveling-wave oscillator.

Figure 4.4 shows the voltage and current waveform at one point of the loop with R_w equal to $0.1k\Omega/m$. After the wavefront, the line becomes stable quite quickly. Taking the rising edge as an example, the edge overshoots V_{DD} slightly. After the edge, the voltage dips a little bit and then settles to V_{DD} quickly. The behaviour at falling edges is similar. The period of the oscillator is 88ps.

Figure 4.5 shows the voltage and current waveform with R_w equal to $9k\Omega$/m. The oscillator's period remains 88ps showing that the propagation

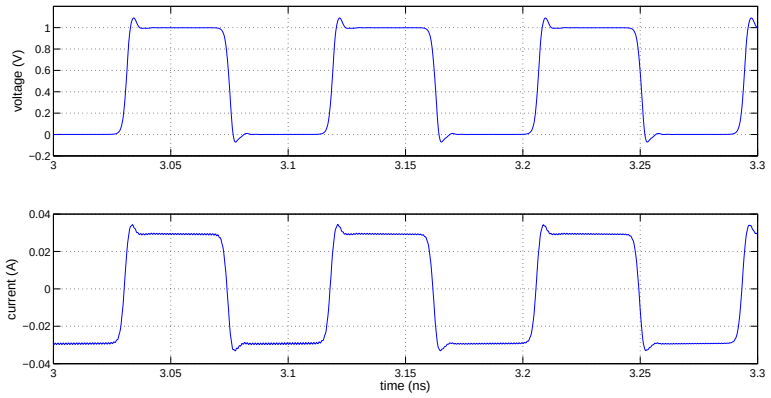


Figure 4.4: Waveform with $R_w = 0.1\text{k}\Omega/\text{m}$ and loop length of 2.4mm.

velocity is determined by $\frac{1}{\sqrt{LC}}$ and largely independent of R. The maximum voltage is 0.99V, very close to V_{DD}. After the dip, the voltage is pulled up to V_{DD} and the current drops. We increase the loop's length to 19.2mm, and the oscillator still works with clean edges.

We continue to increase R_w to $11k\Omega/m$. The behaviour of the oscillator now is dependent on the length of the loop. In this case, I fixed R_w to be $11k\Omega/m$ and varied the length of the loop. Figure 4.6 shows the waveforms when the loop is 2.4mm long. The oscillator has a behaviour that is very similar to that which was observed with R_w equal to $9k\Omega/m$. The period is 89ps and propagation delay is 18ns per meter. Figure 4.7 presents the case that the loop length is 4.8mm. In this case, the oscillator shows substantial ringing on the edges. This ringing occurs due to the constant resistance we use. Due to skin effect and proximity effect, the line resistance will increase dramatically which makes the ringing impossible to show up on a real transmission line. The period of the oscillator is 0.19ns and propagation delay is 19.8ns per meter. The loop in Figure 4.8 is 19.2mm long. We see extensive ringing on the edges. The oscillator's period is 0.97ns and propagation delay is 25.3ns per meter.

Now, I set R_w to be $14k\Omega$; the traveling-wave oscillator ceases oscillation when R_w is larger than $14k\Omega$. Some parts of the loop settle with DC voltages and some parts oscillate with reduced voltage magnitude. Figure 4.9 shows the voltage on one node in the loop. Though the ringing on that node seems to be stable, this ringing does not reach the full power rail and it diminishes

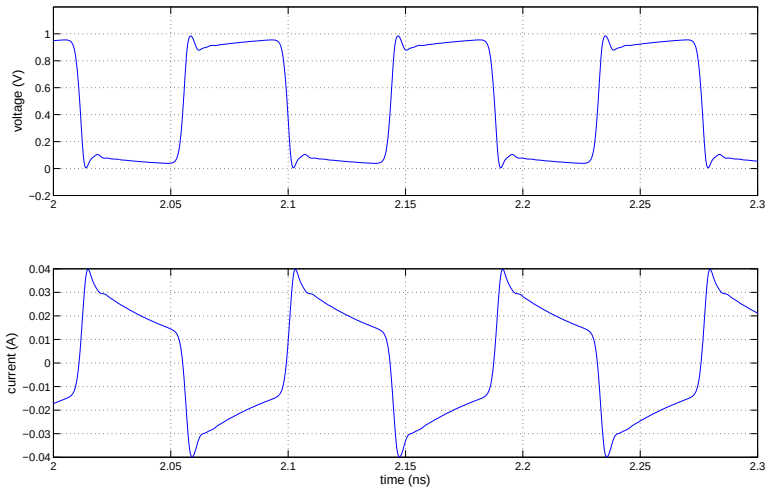


Figure 4.5: Waveform with $R_w = 9\text{k}\Omega/\text{m}$ and loop length of 2.4mm.

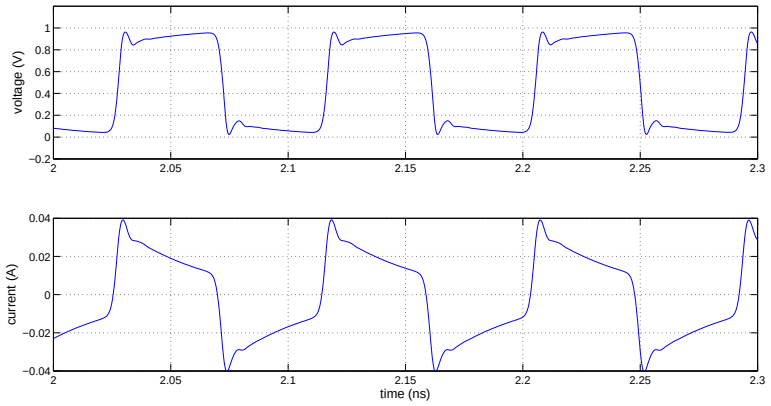


Figure 4.6: Waveform with $R_w = 11\text{k}\Omega/\text{m}$ and loop length of 2.4mm.

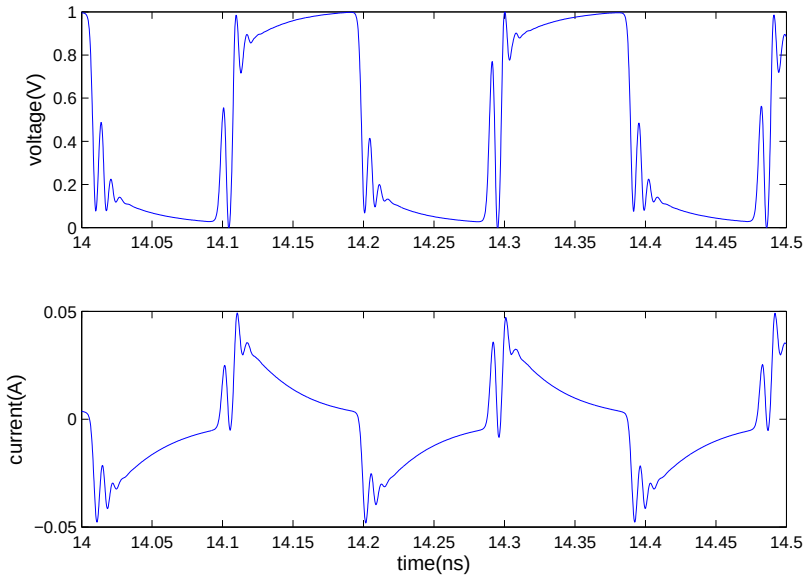


Figure 4.7: Waveform with $R_w = 11\text{k}\Omega/\text{m}$ and loop length of 4.8mm.

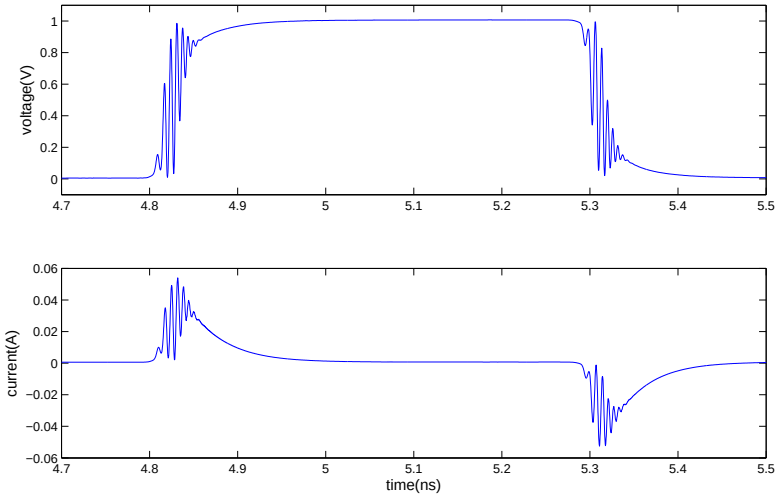


Figure 4.8: Waveform with $R_w = 11\text{k}\Omega/\text{m}$ and loop length of 19.2mm.

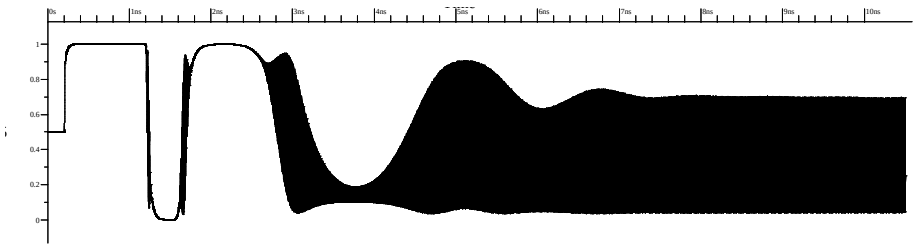


Figure 4.9: Waveform with $R_w = 14\text{k}\Omega/\text{m}$.

along the chain.

As line resistance increases, the cross-coupled inverters can not provide enough energy during the transition time. When line resistance exceeds a critical some value, the operation of traveling-wave oscillator is dependent on the loop length. When R_w is equal to $11k\Omega/m$ and loop is long enough, the circuit ceases to oscillate. Comparing the difference between Figure 4.6, 4.7 and 4.8, we see that as loop length increases to 19.2mm, the line inductance releases nearly all of its energies by the end of each half period. Thus, at each edge, the inductance absorbs energy. The cross-coupled inverters cannot provide enough energy to overcome the losses of the line resistance and restore the current flow through the line inductance to build up the edges. With a shorter loop, each segment of the line can retain enough energy in its inductance to provide a boost for successive edges in the line.

Wood's paper did not give explicit conditions that ensure proper operation of a distributed oscillator circuit. They note that if the total resistance of the line is less than twice its characteristic impedance, then "Transmission line characteristics dominate over RC characteristics." This condition does not consider the effects of shunt conductances, and we note that it does not determine whether or not a distributed oscillator will function as shown by two examples below. First we consider a distributed oscillator with inverters with $w_n = 4\mu\text{m}$ and $w_p = 10\mu\text{m}$ spaced at 0.1mm intervals. The transmission line we use has an inductance of $L_w = 310.6\text{nH}/\text{m}$, a capacitance of $C_w = 104\text{pF}/\text{m}$, a resistance of $R_w = 5\text{k}\Omega/\text{m}$ and a length of 48mm. The characteristic impedance of the line is roughly $Z_0 = 25\Omega$, and the total resistance of the line is $R = 240\Omega$. The resistance is much larger than twice the characteristic impedance; the regime where RC characteristics dominate for an unshunted line. We ran the oscillator for 200ns and the loop still oscillates with clean edges and the loop's period is 1.19ns. Although the inductance releases nearly all of its energy before the next edge comes, the

inverters are big enough to generate enough energy to compensate for the energy loss from the line resistance and provide sufficient energy to allow the line inductance to build the edges.

Likewise, we tried ran simulation experiments using the same values for L_w, C_w and R_w as described above, but we used smaller inverters with $w_n = 0.5\mu$m and $w_p = 0.5\mu$m every 0.2mm along the line and shortened the loop to a length of 4.8mm. This increased the characteristic impedance of the line to roughly 50Ω and decreased the total wire resistance to 24Ω. Thus the total resistance is much clearly less than twice the characteristic impedance, the regime where LC characteristics dominate for an unshunted line. The circuit does not oscillate because the inverter is too weak to provide enough energy to compensate for the energy loss from the line resistance. Obviously, we could make the inverters arbitrarily small and still satisfy $R < 2Z_0$. These examples show that we need a new condition to determine when a distributed amplifier or oscillator has enough gain to restore signals propagating on a long line. We derive such a condition later in this section.

Our goal is to use the open-loop transmission line with cross-coupled inverters to transmit data. Thus, we compared the open-loop transmission line with the closed-loop oscillator. For the open-loop transmission line, we use $1m$ long differential transmission line without cross-coupled inverters to terminate the transmission line in investigation. The lines with cross-coupled inverter pairs are 80mm long. We observe the signals 40mm away from the input source where the forward wave is stable and the reflected wave is absorbed by the cross-coupled inverters. In Figure 4.10, 4.11 and 4.12, the periods are set to be the same as in Figure 4.6, 4.7 and 4.8 respectively and R_w is $11k\Omega/m$. The voltage waveforms are similar as those shown in the above section. As the period increases, the ringing on the edges also increases. According to our simulations, the open-loop and closed-loop transmission lines both fail to propagate edges when R_w is larger than $12k\Omega/m$. Figure 4.13 plots the rising edge of the voltage waveform with different values of R_w for an open-loop transmission line. As R_w increases, the maximum voltage on the rising edge decreases and the ringing after the edge becomes more severe. If R_w is big enough, extended ringing happens on the rising edge and the time for a transition to settle increases dramatically.

We conclude that if the initial condition for the open-loop transmission lines is set as the same as that for the closed-loop traveling-wave oscillator and the period of the voltage source is the same as the period of the oscillator, then the two circuits will behave the same. In the following sections, we will only simulate the open-loop transmission line, and the analysis also applies to closed-loop transmission line.

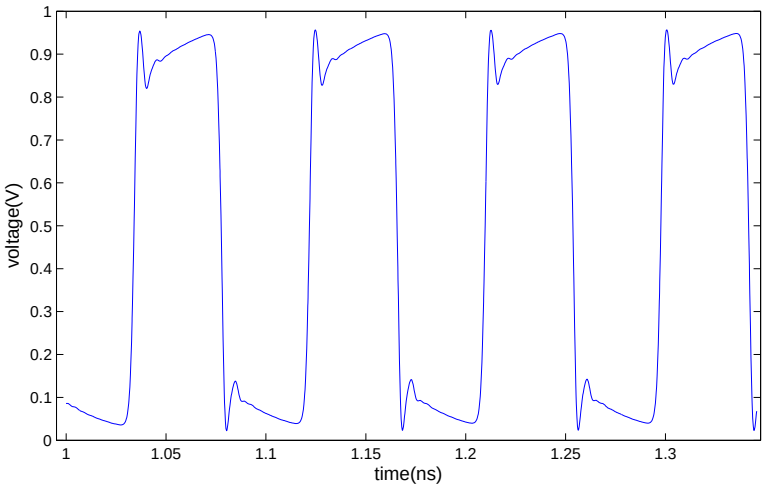


Figure 4.10: Voltage waveform for an open-loop transmission line with $R_w = 11\text{k}\Omega/\text{m}$ and loop length of 2.4mm.

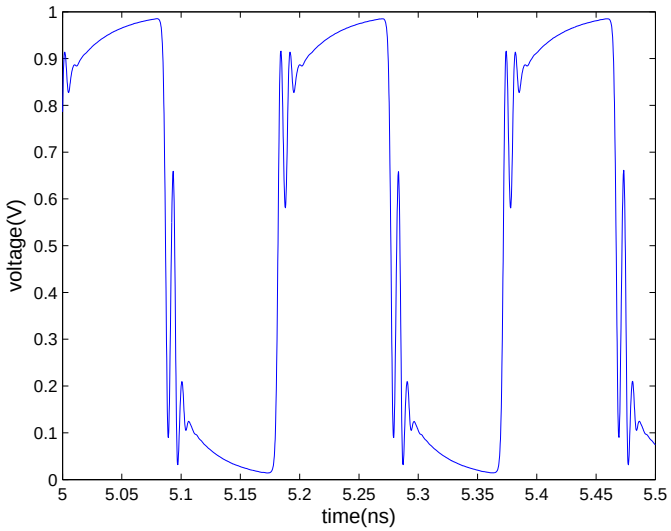


Figure 4.11: Voltage waveform for an open-loop transmission line with $R_w = 11\text{k}\Omega/\text{m}$ and loop length of 4.8mm.

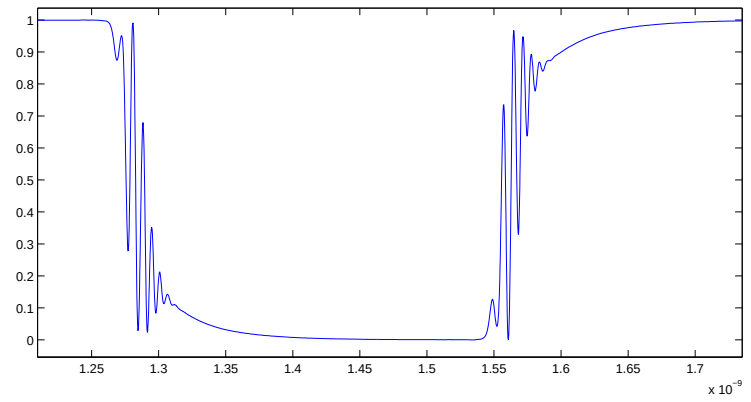


Figure 4.12: Voltage waveform for an open-loop transmission line with $R_w = 11\text{k}\Omega/\text{m}$ and loop length of 19.2mm.

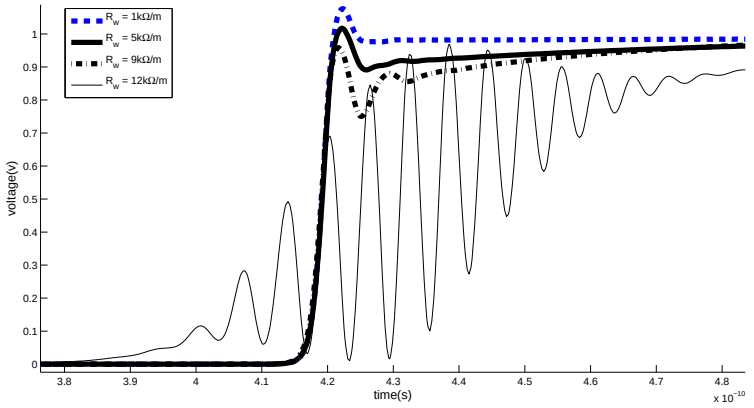


Figure 4.13: Voltage waveforms with regard to different values of R_w.

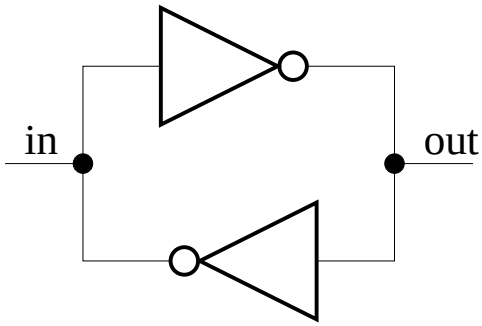


Figure 4.14: A cross-coupled inverter pair.

4.2.2 Small Signal versus Large Signal Analysis

Small signal analysis assumes that the conductance and capacitance presented by the inverters are constants. We did two sets of simulation experiments for basic inverters as in Figure 4.14. In the first set of simulations, V_{out} is set to be equal to $V_{DD} - V_{in}$. We refer these conditions as operating with a differential configuration. In the set of simulations, we only provided the voltage on in and the voltage on out is determined by the cross-coupled inverter pair. This is equivalent to opening the loop of the cross-coupled inverters and we refer to this as the open-loop configuration. The voltage on out is the solid thick curve in Figure 4.15. The solid thin curve is from the simulation of the differential transmission line (T-line). This curve matches very well with the dash-dotted curve where $V_{out} = V_{DD} - V_{in}$. Figure 4.16 shows the corresponding differential transconductance G_m and the drain-to-source conductance G_{ds} for the two sets of simulations. G_m acts as a negative conductance and restores energy to the line, whereas G_{ds} is a positive conductance that absorbs energy. Note that at the beginning and end of a transition, G_{ds} is larger than $-G_m$ and the inverters consume energy, and we refer to these two regions as the resistive mode. When $-G_m$ is larger than G_{ds}, the cross-coupled inverters enter the amplification mode. Notice that here G_m and G_{ds} are differential parameters. The single-ended conductance is twice the differential conductance. In this figure, $G_m = G_{m,p} + G_{m,n}$ and $G_{ds} = G_{ds,p} + G_{ds,n}$. For the open-loop configuration, $-G_m$ is quite flat when the differential input voltage, $V(t) = V_{in} - V_{out}$, is between $-0.4V$ and $0.4V$ and drops quite quickly to 0 outside of this voltage interval. However, if the input and output are differential signals, $-G_m$ reaches the maximum value at the balance point and then drops very rapidly. Again, the results for

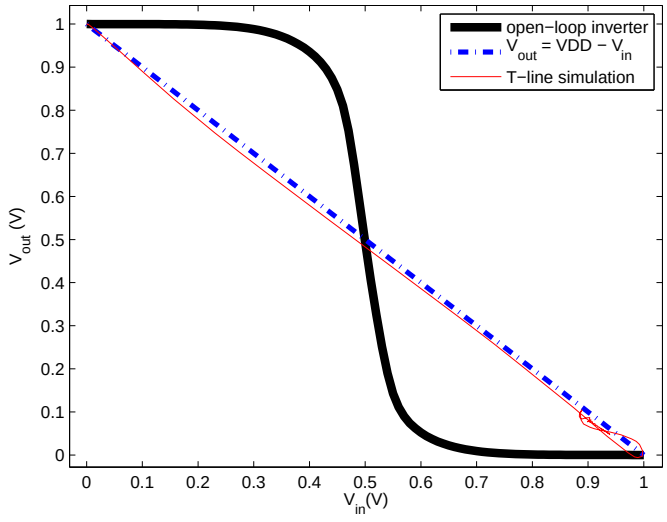


Figure 4.15: Input and output of the inverter.

the differential configuraion match quite well with observations from transmission line simulation. These simulations show that the difference between G_{ds} for the open- and closed-loop models is much smaller than that for G_m. In contrast with G_m, G_{ds} does not change much when $V(t)$ is between $-0.3V$ and $0.3V$.

Depending on $V(t)$, the NMOS and PMOS devices in the inverters may work in the linear, cut-off or saturation region. The difference between the transistor's maximum and minimum gate capacitance can vary by more than a factor of two. The capacitance presented by the transistors is also a function of $V(t)$. Thus for this traveling-wave oscillator, small signal analysis is not adequate because the conductance and capacitance vary widely during transitions. I found a simple and useful approach to analyzing these circuits by using an equivalent constant capacitance and conductance to replace the time varying parameters. I define G_{eff} as the effective conductance for a basic inverter. I do not define G_{eff} as effective conductance per μm for a transistor because it is also dependent on the ratio between the NMOS and PMOS devices in the inverter. To follow the definition of G_{eff}, C_{eff} is defined as the effective capacitance for a basic transistor.

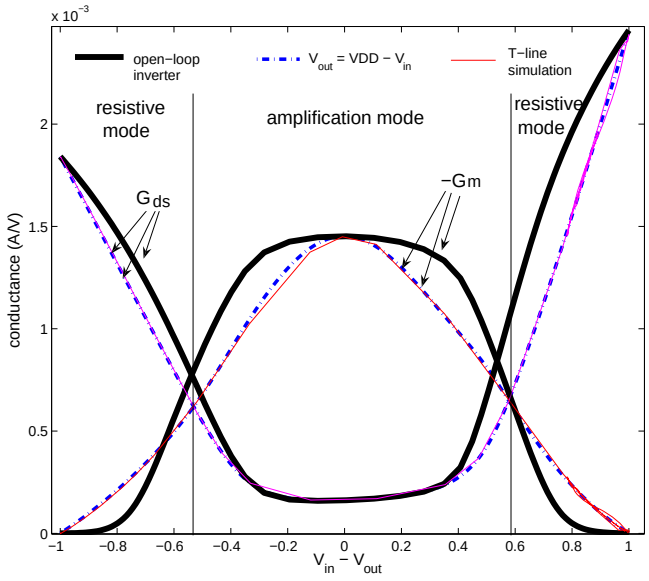


Figure 4.16: Transconductance and conductance of the inverter.

Table 4.1: Effective capacitance per micron of gate width.

number of basic inverters	propagation delay per segment (ps)	C_{eff} (fF/basic inverter)
1	0.953	18.837
2	1.231	19.215
3	1.447	19.334
4	1.648	19.257
5	1.833	19.558
6	2.006	19.866
7	2.152	19.810

4.2.3 Measurement of C_{eff}

The propagation delay d for one segment is equal to $\sqrt{L_s C_s}$ where $L_s = L_w * \Delta x$ and $C_s = C_w * \Delta x + C_{inv} = C_w * \Delta x + C_{eff} * N$ where N is the number of basic inverters in one segment. To measure C_{eff}, I fixed the resistance per meter R_w to be $5k\Omega/m$. Table 4.1 lists the propagation delay and C_{eff} with respect to different sizes of inverters. Table 4.1 shows that C_{eff} grows slowly with the number of inverters used in each cross-coupled inverter pair. I believe that this is because the shape of the waveform in the line depends on the number of inverters, and this affects the contribution of the non-linear capacitances to the line capacitance. However, these effects are relatively small, and I average value, $19.4fF$, in the remainder of this analysis.

4.2.4 Measurement of G_{eff}

Although most of the analysis of the transmission line with cross-coupled inverters models the line as a distributed transmission line, the inverters present lumped capacitance and conductance to the transmission line. The rise time t_r and fall time t_f of the edges are determined by the cutoff frequency F_{cutoff} of the individual segments:

$$F_{cutoff} = \frac{1}{2\pi\Delta x\sqrt{L_w * (C_w + C_{inv}/\Delta x)}} \tag{4.17}$$

Because the signals are differential, t_r is equal to t_f and $t_r = \frac{1}{2F_{cutoff}}$.

To simplify the analysis, I modeled rising and falling transitions as linear functions of time as shown in Figure 4.17. During a transition, the effective

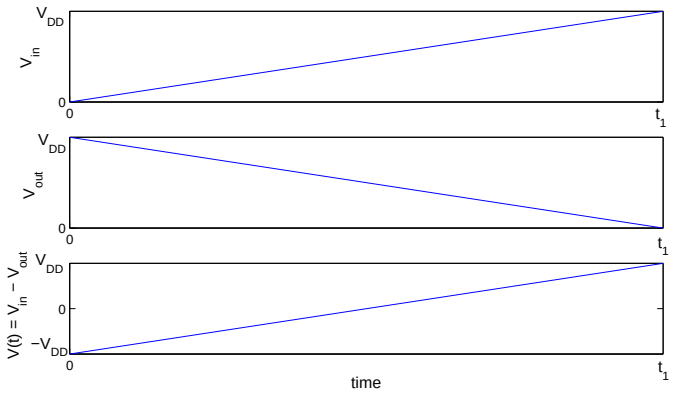


Figure 4.17: Inputs to the cross-coupled inverter.

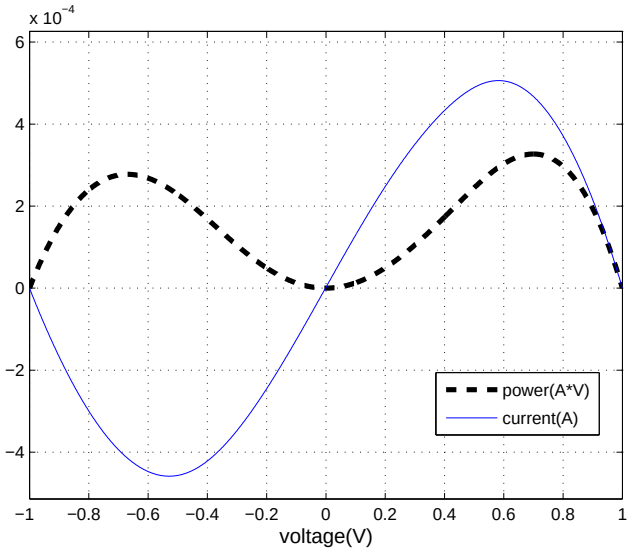


Figure 4.18: DC current of the cross-coupled basic inverter pair as a function of differential voltage input.

conductance of a cross-coupled inverter should inject the same amount of energy as that by the time varying conductance $G(t)$. I derive G_{eff} below using this assumption and then present some simulation results that indicate that the analytical approach produces a good approximation of the behaviour of the actual circuit. For a rising edge, the differential voltage $V(t)$ is $V_{DD}(2\frac{t}{t_1} - 1)$, and $V(t) = V_{DD}(1 - 2\frac{t}{t_1})$ for a falling edge. $I_G(t)$ is the DC current through the conductance. We ignore G_{ds} at first and assume that the cross-coupled inverters only present as negative conductance, the transconductance G_m of the NMOS and PMOS transistors, to the transmission line. Thus the power is positive. The total energy contributed by this transconductance during a transition can be calculated as follows:

$$E_G(t_1) = \int_0^{t_1} I_G(t)V(t)dt = \frac{t_1}{2V_{DD}} \int_{-V_{DD}}^{V_{DD}} I_G(V)VdV \tag{4.18}$$

If the conductance is a constant, $I_G(t) = \|G_{eff}\|V(t)$. At the end of the transition, the total energy E_{eff} provided by this constant conductance is:

$$E_{eff}(t_1) = \int_0^{t_1} \frac{1}{2}\|G_{eff}\|V(t)^2dt = \frac{t_1}{6}\|G_{eff}\|V_{DD}^2 \tag{4.19}$$

Here, G_{eff} is the single-ended effective conductance and the coefficient $\frac{1}{2}$ in the middle term of the above equation converts the single-ended conductance to a differential conductance.

Equating equations 4.18 and 4.19, G_{eff} can be written as follows:

$$\|G_{eff}\| = \frac{3}{V_{DD}^3} \int_{-V_{DD}}^{V_{DD}} I_G V dV \tag{4.20}$$

Figure 4.18 plots the current I_G and power corresponding to $V(t)$. The integral $\int_{-V_{DD}}^{V_{DD}} I_G V dV$ is the area under the power curve. Using Equation 4.20, the effective conductance of the basic inverter is found to be $9.7 \times 10^{-4}\Omega^{-1}$, which is roughly 35% of the maximum transconductance.

Define R_{crit} as the maximum wire resistance a traveling-wave oscillator can tolerate for a given number of basic inverters. Using G_{eff} in Equation 4.11, we predict that, for the transmission line, R_{crit} for 2, 5 and 7 basic inverters per 0.1mm segment are $12.2k\Omega/m$, $14k\Omega/m$ and $14.4k\Omega/m$ respectively. With HSPICE simulation, the maximum resistance that can propagate clean edges without ringing is $8k\Omega/m$, $9.5k\Omega/m$ and $10k\Omega/m$ respectively. With ringing, R_{crit} for 2, 5 and 7 basic inverters are $11.5k\Omega/m$, $12k\Omega/m$ and $12k\Omega/m$. This method gives an upper bound for R_{crit}. We have not yet considered the effect of G_{ds} in our model. When the cross-coupled

inverters are in transition, Figure 4.16 shows that G_{ds} does not change much. Deducting a value of G_{ds} equal to $0.32 \times 10^{-3}\Omega/\mu m$, $\|G_{\mathit{eff}}\|$ decreases to $6.5 \times 10^{-4}\Omega^{-1}$. R_{crit} becomes $8.2k\Omega/m$, $9.4k\Omega/m$ and $9.7k\Omega/m$ for 2, 5, and 7 sets of cross-coupled inverters respectively. These values match those observed from HSPICE simulation to within 3%. Using $G_{m,\mathit{eff}}$ to denote the effective transconductance calculated from Equation 4.20, the effective conductance of a basic inverter is:

$$-G_{\mathit{eff}} = -G_{m,\mathit{eff}} - G_{ds,\mathit{eff}} = \frac{3}{V_{DD}^3}\int_{-V_{DD}}^{V_{DD}} I_G V dV - G_{ds} \tag{4.21}$$

To further validate our model, we change L_w to be $207nf/m$ and C_w to be $0.156nF/m$. G_{eff} is unchanged because it is only related to the cross-coupled inverters. R_{crit} given by this effective conductance is $6.2k\Omega/m$ with 7 sets of cross-coupled inverters. HSPICE simulation shows that the transmission line can transmit a clean edge with R_w equal to $6.5k\Omega/m$ and fails with R_w being $7k\Omega/m$; the agreement is to within $5-15\%$. This gives some evidence that Equation 4.21 is a reasonable approximation of the effective conductance.

Now consider a line when the inverter capacitance C_{inv} dominates the line capacitance, Replacing C_{inv} with NC_{eff} where N is the number of basic inverters per segment, Equation 4.11 can be written as follows:

$$\frac{C_{\mathit{eff}}}{-G_{\mathit{eff}}} = \frac{L_w}{R_w} \tag{4.22}$$

The left term in Equation 4.22 is a parameter related to the process and the ratio of the widths of PMOS and NMOS transistors in the basic inverter. We can reorganize the above equation as follows:

$$R_w C_w \leq -\frac{L_w C_w * G_{\mathit{eff}}}{NC_{\mathit{eff}}} \tag{4.23}$$

where $L_w C_w$ is a constant equal to $\frac{c^2}{\epsilon_R^2}$ where c is the speed-of-light in vacuum and ϵ_R is the relative permittivity. We can see that for a given process, $R_w C_w$ must satisfy the above inequality so that the traveling-wave oscillator can oscillate.

The above analysis is based on a basic inverter where the PMOS transistor is two and half times wider than the NMOS transistor. We now examine the impact of varying this ratio. In particular, we kept the total width of the inverter the same and changed the percentage of NMOS width to the total width. The maximum effective conductance is $6.6 \times 10^{-4}\Omega^{-1}$ for a

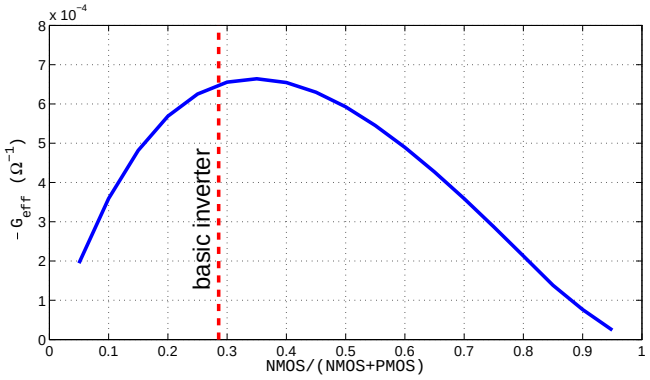


Figure 4.19: Effective conductance of an inverter.

ratio of 0.35 as shown in Figure 4.19. Our basic inverter's ratio is 0.286 and its effective conductance is $6.5 \times 10^{-4}\Omega^{-1}$, close to the maximum effective conductance.

4.3 Surfing LC Interconnect

To propagate data at near-speed-of-light-in-glass speed, we open the loop of the traveling-wave oscillator and use the transmission line to send data. To control uncertainties on the line, I developed a surfing technique for LC mode, on-chip communication. The key requirement for surfing is to make the delay of the interconnect a time varying function. LC signaling requires a new type of surfing circuit. Previous surfing circuits dynamically adjust the driving strength of drivers to modulate their delay. This is not effective for the LC mode signaling. Dynamically adjusting the driving strength of cross-coupled inverters only changes the transconductance; however the propagation velocity of transmission line is determined primarily by the line inductance and capacitance. Thus we need a new surfing technique. I now present a way to build surfing interconnects which use varactors to vary the line delay.

Figure 4.20 shows an implementation of a varactor having three terminals. This varactor consists of two long channel transistors of the same width: one NMOS transistor controlled by $\overline{\mathsf{ctrl}}$ and one PMOS transistor controlled by ctrl. When ctrl is low, depending on the voltage on node A, at least one transistor operates in the linear region. Figure 4.21 plots the

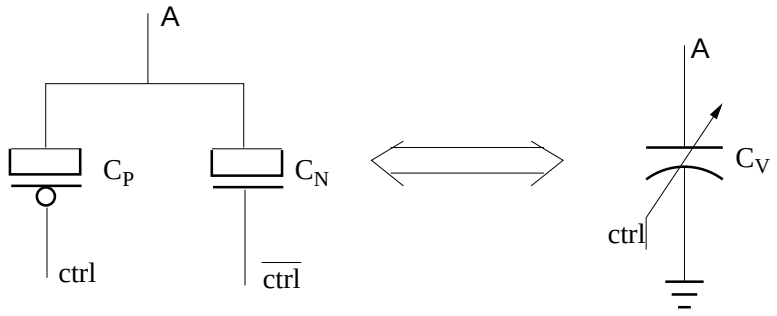


Figure 4.20: A varactor.

capacitance of the NMOS and PMOS devices as seen by node A when ctrl is low. In Figure 4.21 and 4.22, the NMOS and PMOS transistors are both $2.2\mu m$ long, $0.55\mu m$ wide. When voltage on node A is 1V, the channel of the NMOS is depleted and the capacitance for source and drain to gate, $C_{A,n}$ is very small. The channel of the PMOS is fully inverted and the capacitance for source and drain to gate, $C_{A,p}$, reaches its maximum value, close to 16fF. The maximum capacitance seen by node A is 29.5fF while both PMOS and NMOS devices are in inversion. Both transistors are depleted if ctrl is high and the maximum capacitance is 0.8fF as shown in Figure 4.22.

We use the symbol on the right side in Figure 4.20 to represent a varactor and place varactors along the transmission line as shown in Figure 4.23. The propagation speed of the input is dependent on the capacitance value of the varactor and thus the voltage of ctrl. If the varactors are placed every y units along the transmission line, the propagation speed of the transmission line is:

$$D(\mathsf{ctrl}) = \sqrt{L_w(C_w + \frac{C_v(\mathsf{ctrl}, V_\mathsf{A})}{y})} \tag{4.24}$$

where $C_v(\mathsf{ctrl}, V_\mathsf{A})$ is the capacitance presented by the varactor as a function of ctrl and voltage V_A of node A. In the following, we we use $C_v(\mathsf{ctrl})$ to denote the varactor's capacitance because the varactor's capacitance value is mainly dependent on ctrl.

The ratio between the minimum delay and maximum delay of the transmission line can be written as:

$$\frac{D(0)}{D(1)} = \sqrt{\frac{C_w + C_v(0)/y}{C_w + C_v(1)/y}} \tag{4.25}$$

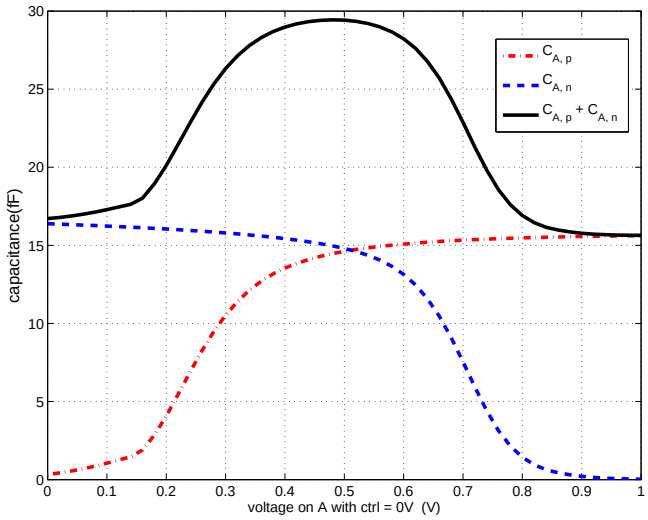


Figure 4.21: Capacitance to ground of node A (ctrl = 0).

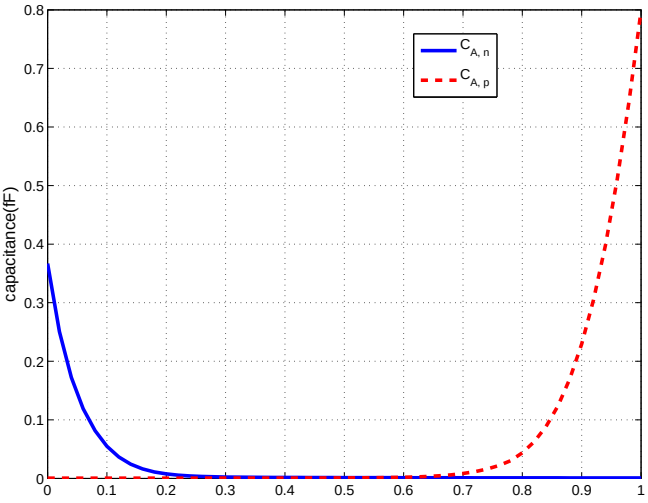


Figure 4.22: Capacitance to ground of node A (ctrl = 1).

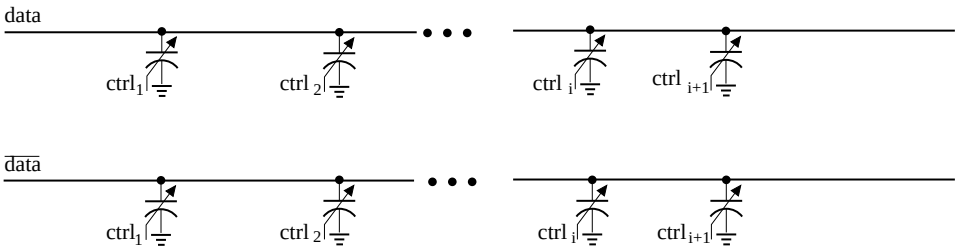


Figure 4.23: A surfing transmission line.

Maximizing the capacitance difference of the varactor and decreasing C_w help to increase the delay variation of the transmission line. We use long channel transistors such that the gate capacitance dominates parasitic capacitance and improves the capacitance difference of the varactor. As shown in Figure 4.21 and 4.22, $C_v(1)$ is much smaller than $C_v(0)$. To simplify the calculation, we can approximate $C_v(1)$ with 0. We only need to consider $C_v(0)/y$, which is chosen to meet the timing variation requirement of the system.

We could not find any estimation of the timing variation for on-chip, global wires in the literature. In the following design, we assume that the timing variation is at most $\pm 10\%$. Thus, we chose design targets of $\frac{D(0)}{D(1)}$ equal to 1.22 and $C_w + C_v(0)/y$ equal to $1.5C_w$. This means that with this timing variation requirement, the value of $C_v(0)/y$ is roughly half of the line capacitance C_w. Otherwise, the propagation speed is mainly dependent on the line parameters.

Figure 4.23 is not a complete surfing interconnect. The ctrl signal must propagate aligned with the data signal to dynamically adjust the capacitance of the varactor. Attenuation and intersymbol interference are critical concerns for long on-chip communication. We developed two surfing schemes to deal with these issues: a full-swing surfing and a low-swing surfing transmission line.

4.3.1 Full-swing Surfing Interconnect

Figure 4.24 shows the scheme for full-swing surfing interconnect. It includes three parts: a clock line, a data line and bridges. The bridge circuit in Figure 4.25 is used to forward the clock signal to the varactor. The outputs of the bridge circuit, ctrl and $\overline{\text{ctrl}}$, are delayed versions of inputs clk and $\overline{\text{clk}}$. We add cross-coupled inverters at the last two stages to align transitions of

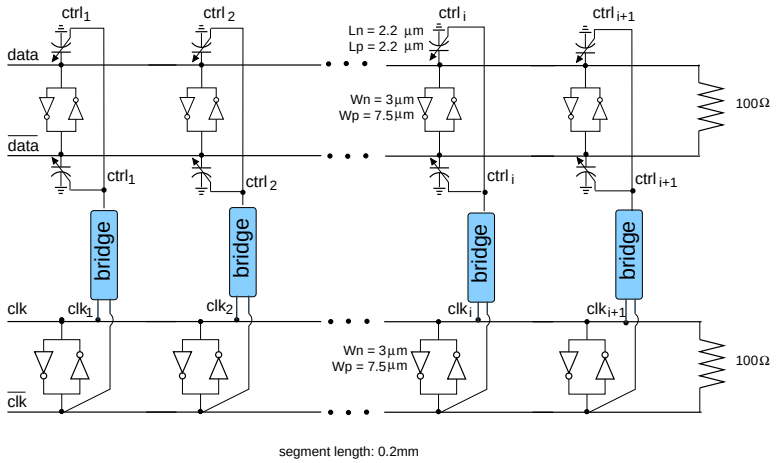


Figure 4.24: Surfing interconnect.

the ctrl and $\overline{\text{ctrl}}$ signals such that they arrive at the varactor at very nearly the same time to cancel their capacitive coupling effect on the data line. To simplify our prototype implementation, the varactors are not double-pumped. The clk signal completes a cycle (two edges) for each data bit sent. Thus, the frequency of the clk signal is at least twice that of the data signal, and propagating the clock signal is the main throughput limiter for this design. We use an inverter chain to implement the bridges between the clock and data lines. As in the traveling-wave oscillator, the cross-coupled inverters are added into the data line and clock line to provide negative conductance to compensate for the energy losses and consequent dispersion caused by the line resistance.

The requirement for surfing can be summarized with the following inequality (compare with Inequality 2.2):

$$D_{d,max}(1) < D_{c,min} < D_{c,max} < D_{d,min}(0) \tag{4.26}$$

where D_c is the delay of clk and D_d the delay of data signal. The delay interval from $D_{c,min}$ to $D_{c,max}$ on the clock line is caused by noise disturbances, parameter variation and so on. Without the surfing circuits, *i.e.* varactors, the data line also faces the same amount of delay variation. In another words, surfing must create a delay variation on the data signal which must be larger than that created by the noise, ISI, and any other disturbances.

Replacing the delay with line inductance and capacitance, Inequality 4.26

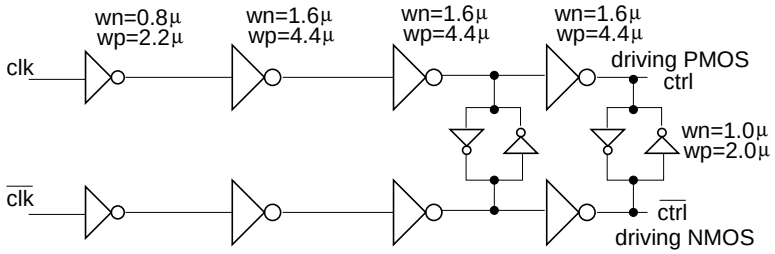


Figure 4.25: Bridge between the clock and data lines.

can be rewritten as:

$$\sqrt{\frac{L_{w,d}}{L_{w,c}}C_d(1)} < \sqrt{C_c} < \sqrt{\frac{L_{w,d}}{L_{w,c}}C_d(0)} \tag{4.27}$$

Where $C_d(\mathsf{clk})$ is the total capacitance per unit of the data line as a function of clk and C_c is the total capacitance per unit of the clock line. $L_{w,x}$ and $C_{w,x}$ are the data line ($x = d$) or clock line ($x = c$) inductance and capacitance. $C_d(\mathsf{clk})$ and C_c can be calculated with Equation 4.28 and 4.29 assuming that varactors and inverters are placed every y units along the interconnects.

$$C_c = C_{w,c} + \frac{C_{inv,c}}{y} \tag{4.28}$$

$$C_d(\mathsf{clk}) = C_{w,d} + \frac{C_{v,d}(\mathsf{clk}) + C_{inv,d}}{y} \tag{4.29}$$

where $C_{inv,c}$ and $C_{inv,d}$ are the capacitance presented by the cross-coupled inverters in the clock line and data line. $C_{v,d}(\mathsf{clk})$ is the capacitance of the varactors in the data line.

If the data line and clock line are of the same geometry, Inequality 4.27 can be simplified as:

$$C_{v,d}(1) + C_{inv,d} < C_{inv,c} < C_{v,d}(0) + C_{inv,d} \tag{4.30}$$

An easy way to satisfy this requirement is to set $C_{inv,d}$ equal to $C_{inv,c}$ and add varactors to the clock line as shown in Figure 4.26. This varactor differs from that in Figure 4.20 because the two MOSFETs in Figure 4.26 never operate with their channels depleted at the same time. Figure 4.27 shows an implementation of this surfing interconnect. As long as the varactor in the clock line is smaller than that in the data line and the cross-coupled

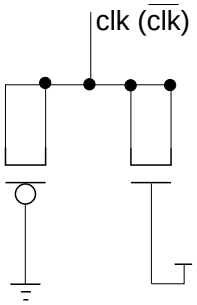


Figure 4.26: Varactors on the clock line.

inverters in the clock line and data line are of the same size, the surfing requirement is satisfied. A good choice of the varactor size is to make C_c equal to $\frac{1}{2}(C_{v,d}(0) + C_{v,d}(1))$ such that the delay of the clock line is close to the mean of the maximum and minimum delay of the data line. With varactors in the transmission line, Inequality 4.23 can be rewritten as follows:

$$-\frac{NG_{eff}/y}{\frac{NC_{eff}+C_v(0)}{y} + C_w} \geq \frac{R_w}{L_w} \tag{4.31}$$

Where N is the total number of basic inverters every y units along the transmission line and C_v is the varactor capacitance. The above inequality ensures that signals propagate without attenuation. In our implementation, every 0.2mm, we add inverters and varactors to the transmission line. For the data line and clock line, the NMOS transistor of the inverter is $3\mu m$ wide and the PMOS transistor 7.5μm wide. The transistors in the varactors for the data line are 2.2μm long and 0.55μm wide. The data line and clock line have the same geometry. The transistors in the varactors for the clock line are 0.8μm long and 0.55μm wide. Thus, the surfing requirement is satisfied. The line parameters, L_w, C_w and R_w are 212nH/m, 200pF/m and 1645Ω/m. We also take the skin effect resistance R_s equal to $1.4e^{-5}\Omega/\text{mm}/\text{Hz}^{-0.5}$ into account. The transmission lines are 19mm long.

Figure 4.28 plots the delay curve as a function of the separation time from the clk signal to the data signal for the design shown in Figure 4.27. If the data comes earlier than the clock transition, the capacitance of the varactors in the data line increases to increase the propagation delay of the data line. Thus, the delay of the data line becomes greater than that of the clock line. Conversely, if the data comes later than the clock transition, the capacitance of the varactor decreases such that the data signal can propagate

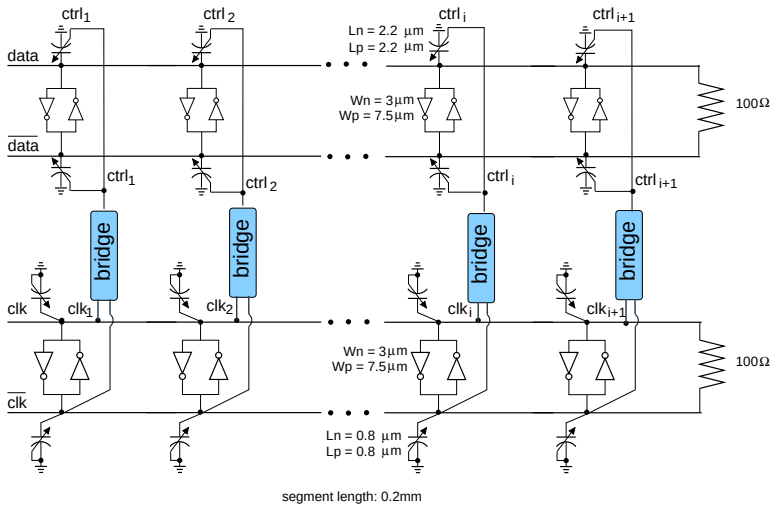


Figure 4.27: An implementation of a surfing interconnect.

faster than the clock signal. When the separation time between the data signal and clock signal is around 98ps, the propagation delay of the data line and clock line are the same. The delay variation is 2.2ps/mm with the maximum delay being 12.3ps/mm and the minimum delay 10.1ps/mm.

The clock signal propagates at 11.2ps/mm. With L_w being 212nH/m, the clock line's total capacitance is around 470fF/mm and the characteristic impedance of the differential line is around 40Ω. The terminators at the end of data line and clock line are chosen to be 100Ω/m. Because this is greater than the line impedance, a reflection will occur. The reflected wave helps to build up the clock and data signals at the far end of the transmission line. The cross-coupled inverters work as latches to hold their states. They can be modeled as resistors connecting the lines to the power rails. Thus, the cross-coupled inverters absorb the energy of the reflected wave.

Figure 4.29 shows waveforms from a simulation of this implementation. Although the arrival time of the data signal is disturbed at the near end of the interconnects, at the far end, the data signal still comes later than the clock signal by around 98ps. The 98ps delay is introduced by the bridge circuit between the clock line and data line. At the rising and falling edges, neither the data signal nor clock signal reach the full power rail voltage due to the skin effect resistance. Cross-coupled inverters sharpen the edges by consuming energy from the low frequency components and injecting energy

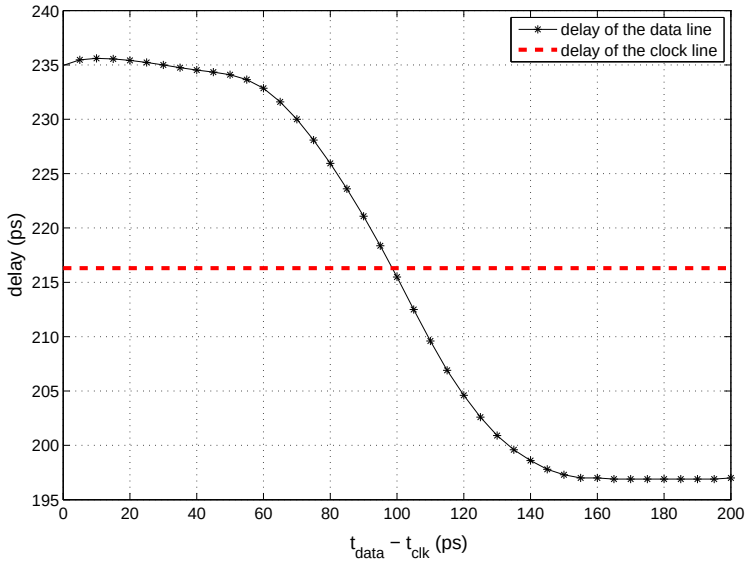


Figure 4.28: Delay curve of the surfing transmission line.

into the high frequency components into the line. At the sender's end of the interconnects, from $-0.6V$ to $0.6V$, the transition time is 30ps. At 12mm away, the transition time of the signals has decreased to less than 20ps.

4.3.2 Low-swing Surfing Interconnect

The cross-coupled inverters contribute a significant amount of capacitance to the line in the full-swing design. This lowers the propagation speed of the transmission line. If we remove the cross-coupled inverters in the data line, we may remove the varactors in the clock line to match the nominal delays of both paths. This increases the propagation speed of the interconnects. Driving the interconnects becomes easier because the characteristic impedance of the clock and data line increases. We propose our low-swing design to achieve these advantages.

Varactors not only reduce timing uncertainty, they also sharpen transitions. As shown in Figure 4.21, the capacitance of the varactor is not monotonic but has a global maximum at roughly half V_{DD}. If node A is biased at half V_{DD}, the varactor can sharpen both rising and falling edges. At the wavefront, the upper part of the edge propagates faster than the lower

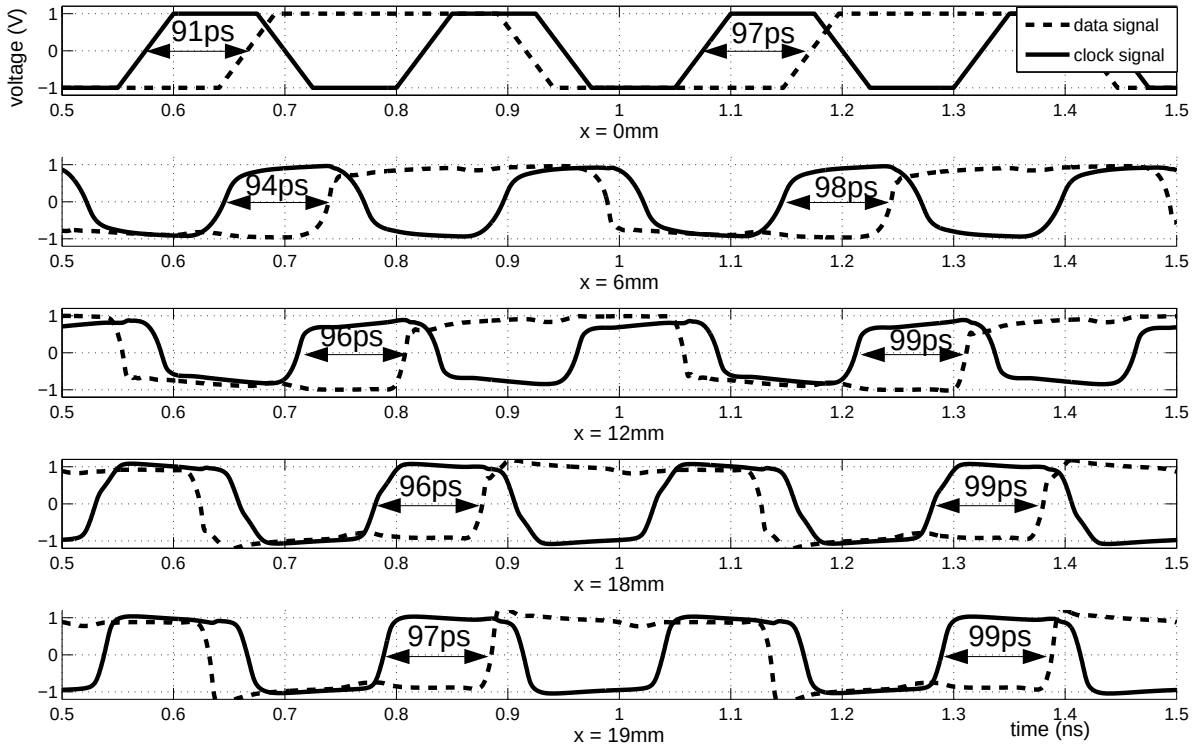


Figure 4.29: A simulation of full-swing surfing interconnect.

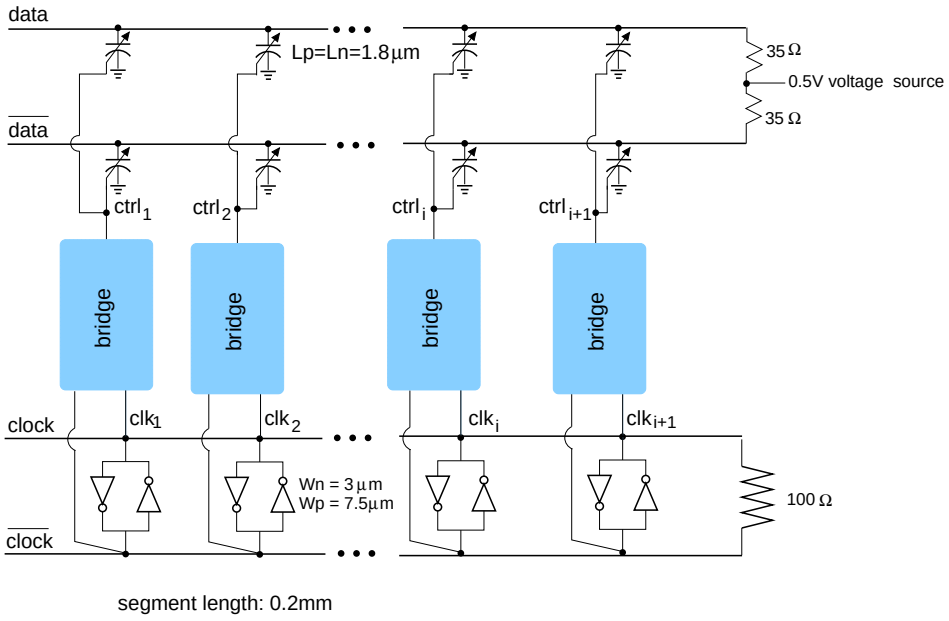


Figure 4.30: Low-swing surfing interconnect.

part of the edge with decreasing capacitance. In the full swing design, edge sharpening is achieved with the cross-coupled inverter pairs by injecting high frequency energy into the line. As we will describe later, surfing achieved with varactors further increase the varactor's capacitance variation range and can reshape a wide data pulse into a narrow pulse. Using the global capacitance maximum of the varactor and the surfing effect introduced by the varactor, we can remove the cross-coupled inverters from the data line.

Figure 4.30 presents a low-swing surfing interconnect where the data signal is low-swing, return-to-zero signal and the clock signal is still full-swing signal. The data line is biased at half V_{DD}, 0.5V, such that the varactor capacitance is close to its maximum value. The varactor capacitance decreases whenever the data line voltage changes. To meet the surfing requirement, Inequality 4.26 should be satisfied. There are some challenges in establishing this condition because the low-swing line does not restore the data signal as it propagates down the line. Instead, pulses change shape. If the clock is constantly low to get the slowest line, then pulses will spread out as they propagate down the line due to the dispersion caused by line resistance. In fact, the line can be close to operating in its RC mode. When the clock is

constantly high, to get the fastest line, this dispersion will be less, but still present. The desired pulse shaping and sharpening only occurs when the the clock is changing with the data. Noting that the data waveform changes shape along the line, I measured the delay of the data signal when the clock is a constant (high or low) as the time until the data signal crosses its mid-point value. When the clock is a square wave, I measured delays according to the time at which the data signal reached its peak. With these considerations in mind, a starting point for establishing the surfing condition is to compare the capacitance of the clock and data lines; the inductance and resistance of the lines should be roughly equal because the two lines have the same geometry. Likewise the wire capacitances, C_w should closely match for the two lines. For reasonable sized varactors and inverters, $C_{v,d}(1)$ will be less than $C_{inv,c}$ because the varactors in the data line contribute much less capacitance to the data line than the cross-coupled inverters to the clock line. Thus, as long as $C_{v,d}(0)$ is greater than $C_{inv,c}$, the surfing requirement should be satisfied. Because of the signal propagation is complicated by the mix of RC and LC behaviours described above, this condition should be confirmed by simulation, which is what I do with the design reported in Chapter 5.

The low-swing design in Figure 4.30 uses the same line geometry as the full-swing design in Figure 4.27. Due to the removal of the varactors in the clock line, the clock signal propagate at 10.2ps/mm, roughly 6% faster than that in the full-swing design. Figure 4.31 plots the data signal for such a design from three simulations: clk being 0 all the time, clk being 1 all the time and clk alternating between 0 and 1. The varactor in this simulation is 1.8μm long, 0.55μm long. When the clock signal is constant 0, the data signal propagates the slowest of the three cases because the total line capacitance is the largest of the three cases. Conversely, when the clock signal is constant 1, the data signal shows up the earliest at the far end of the line because the total line capacitance is the smallest. When the clock signal alternates between 0 and 1, the wide pulse at the sender's side of the line is reshaped into a narrow pulse at the far end of the line. Note that when clk is constant, either 0 or 1, at the far end of the line, the data signal becomes wider and shorter due to attenuation and dispersion caused by the line resistance. However, on the surfing transmission line, the pulse becomes narrower and the magnitude of the pulse increases. More importantly, the narrow pulse is developed at the rising edge of clk. Before the rising edge of clk, the signal propagates slower than clk. After the rising edge of clk, due to the smaller line capacitance on the data line, the pulse propagates earlier than clk. This produces a narrow, tall pulse at the rising edge of clk.

We now consider the optimal timing of the data pulse relative to the rising edge of the clock at the input side. We consider three scenarios: the clock edge occurs before the leading edge of the data pulse; the clock edge occurs after the trailing edge of the data pulse; and the clock edge occurs between the leading and trailing edges of the data pulse. If the rising edge of the clock occurs before the leading edge of the data pulse, then the data pulse will propagate faster than the clock. When the leading edge catches up with the clock, the pulse will start to narrow and grow in height. Likewise, if the data pulse occurs before the clock edge, then it will gradually fall back to the clock; at which point, the pulse will narrow and grow in height as before. On the other hand, if the clock edge occurs between the leading and trailing edges of the data pulse, then both edges of the data pulse will move towards the clock edge, narrowing the data pulse and increasing its height. By concentrating the charge of the pulse into a narrower time interval, the surfing line increases the height of the pulse, thereby counteracting the attenuation of the line. More importantly, the narrow pulse is developed at the rising edge of clk which makes the design of the deskew circuit easy as we will demonstrate in Section 5.5.3.

Note that this only works for a bounded length line – eventually all of the charge is in a narrow pulse, and that pulse will die out as it continues down the line. As the pulse get narrower and narrower, this pulse gets less and less charges from the leading and trailing parts of this pulse. However, the line resistance increases because the skin effect resistance increases as the pulse gets narrower, thus the attenuation rate increases and eventually, the concentrating effect induced by the nonlinear capacitance of the varactors is weaker than the attenuation effect due to the line resistance. Thus the low-swing design does not work for an infinite long wire.

4.4 Summary

In this chapter, we extended surfing to methods that can be used with LC transmission lines. In particular, we used varactors to modulate the delay of the transmission line such that the data signal can surf with the clock signal. We proposed two surfing schemes: a full-swing design and low-swing design. The full-swing design adds cross-coupled inverters to both the clock line and data line to compensate for the energy losses along the lines. The low-swing design only adds cross-coupled inverters to the clock line to compensate for energy losses. This reduces the power consumption of the data line. We will demonstrate this in Chapter 5. The data signal is low-swing, return-to-zero

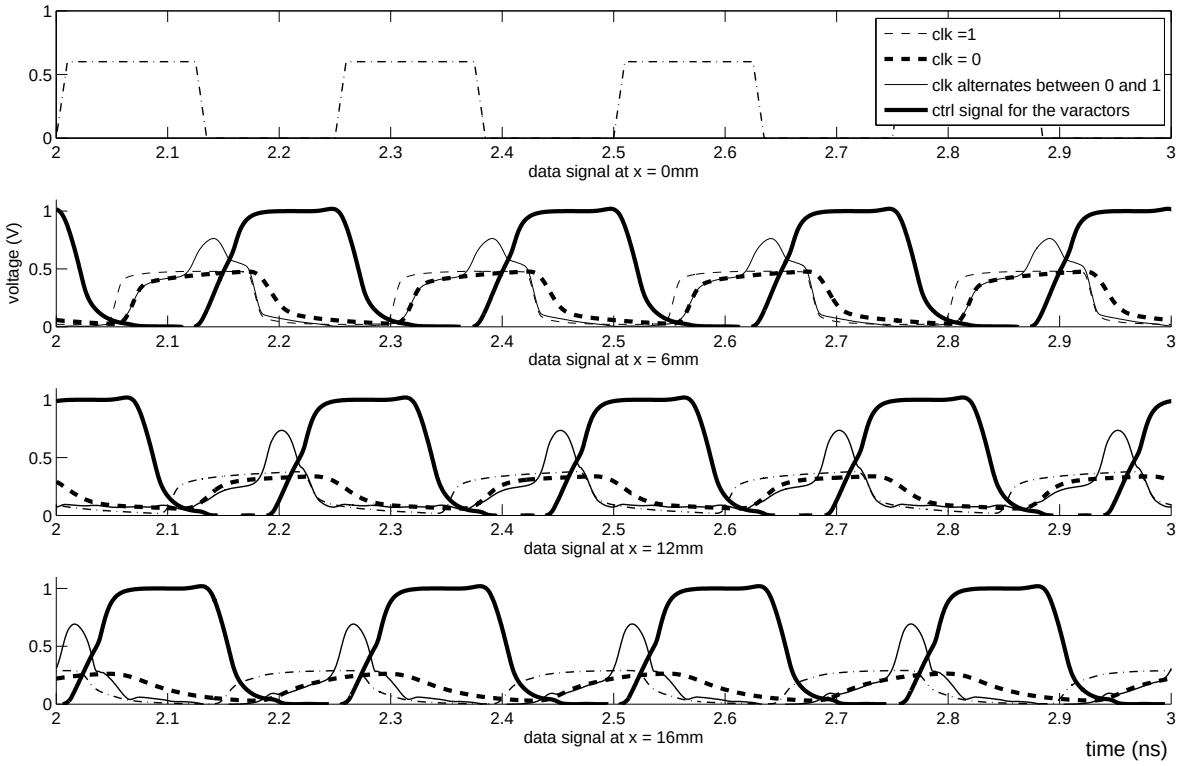


Figure 4.31: Simulation of low-swing interconnect.

differential signal. The surfing effect in this design not only bounds the delay spread between data signal and clock signal, but also helps to reshape the data signal into a narrow pulse to improve the eye height.

Our surfing transmission line uses the idea of cross-coupled inverters to provide negative conductance shunts as in Wood's traveling-wave oscillator [60]. To understand the functionality of this style of design, we presented a detailed analysis of the traveling-wave oscillator. Furthermore, we developed a model to size the cross-coupled inverter pairs for a given transmission line. Our models match very well with simulation results. In this model, we do not take the skin effect resistance into account. We expect to use this simple model as a rough estimation for the transistor sizing.

Although we only provided examples of surfing transmission lines with matched geometry for the clock and data line, the clock and data lines do not necessarily have to have the same width and spacing. Taking the full-swing design as a example, assume that the sizes of cross-coupled inverters are the same and there are no varactors on the clock line. Because the two lines run at the same speed at the surfing operating point, the total capacitances of the two lines must be the same. The clock line could be widened to meet this requirement. Of course, there is a practical advantages to matching the lines as much as possible: by matching the wire capacitance of the data line with wire capacitance of the clock line and likewise matching varactors with varactors, and inverters with inverters, the design should be quite robust to PVT variations.

Chapter 5

The Test Chip

We designed a chip to demonstrate the LC-mode surfing interconnect techniques that were developed in the previous chapter. The chip has four groups of surfing interconnects: two for the full-swing design and two for the low-swing design. Each experiment includes three parts: a transmitter, a receiver and the surfing interconnect. To increase the line inductance and reduce line capacitance, we used top-level metal, metal 9, for the transmission line and metal 2 for the ground plane in all four experiments. We chose different widths and spacings for the experiments to get a better understanding of the impact of line parameters.

This chapter is organized as follows: Section 5.1 gives the big picture of the chip; Section 5.2 describes the design issues that are common to full-swing and low-swing designs; Section 5.3 and 5.4 present the design details for full-swing and low-swing designs. Section 5.5 presents the testing results.

5.1 Structure of the Whole Chip

Figure 5.1 shows the structure of the whole chip which includes: input and output pads, transmitters, receivers, surfing transmission lines, interfaces to a deskew block and a scan chain. The deskew block implements a variation of the simple, source-synchronous interface described in [13]. It was implemented by Tarik Ono of SUN Labs and UBC as part of her research. Thus, I will not describe the details of the design here. However, the deskew block is used in the testing to demonstrate that surfing can be used to implement on-chip, source-synchronous signaling without the use of the complicated clock-data-recovery circuits that are needed for the LC-mode, on-chip communication techniques that have been proposed by other researchers [29, 30, 37, 39]. The scan chain provides a JTAG interface and is implemented using standard library cells from Sun Microsystem Laboratories. The pads on the right side are the inputs and outputs for the scan chain only. The outputs of the scan chain control various chip functions such as choosing which experiment to run, configuring various adjustable delays on the clock lines and in the receiver circuits and so on. There are four groups

of surfing lines on the chip which serpentine through the chip: FS-8, FS-12, LS-12, LS-8. Each group consists of two differential pairs, one for data and the other for the forwarded clock. Together, the four groups consist of 16 long, snake wires. Figure 5.2 lists the order of these signals. FS-8 and FS-12 use the full-swing design and LS-8 and LS-12 use the low-swing design. The dashed lines are the center lines for every group. The numbers above the center lines are the length of the center lines. The solid filled boxes along the lines in Figure 5.1 are the cross-coupled inverter pairs and varactors along the transmission lines. We call each such unit with a cross-coupled inverter pair and varactor a *repeater*, and place a repeater every 0.2mm along the center line of every group. In the low-swing design, a repeater may include a cross-coupled inverter pair only or a varactors only. Note that the repeaters in this figure does not corresponds to the exact number and location of repeaters on the chip. Each group has its own transmitter and receiver. The leftmost column of pads are only for power and ground and the second column of pads on the left side provides the differential clock and data inputs and outputs.

The block for interface1 and interface2 in Figure 5.1 is shown in Figure 5.3. This interface determines which group of transmission lines to use and whether the deskew block is enabled or not. The major components in the interface block are the multiplexors. Table 5.1 gives the truth table for this block. The transmitters for the full-swing design or the low-swing design can be enabled at the same time to create cross-talk noise for transmission lines. In this figure, there are three clock domains: producer, receiver and consumer. After the transmission line or the inverter chain, the producer's data is in the receiver's clock domain. The deskew block transfers the receiver's data from the receiver's clock domain to the consumer's clock domain.

5.2 Layout Issues

In the process of designing this test chip, we encountered many issues that must be addressed to make LC-mode surfing work on a real chip. These issues included:

- The line geometry determines the line inductance, capacitance and resistance. These parameters affect the size of cross-coupled inverters and varactors.
- To make long wires on a chip of reasonable area and aspect ratio,

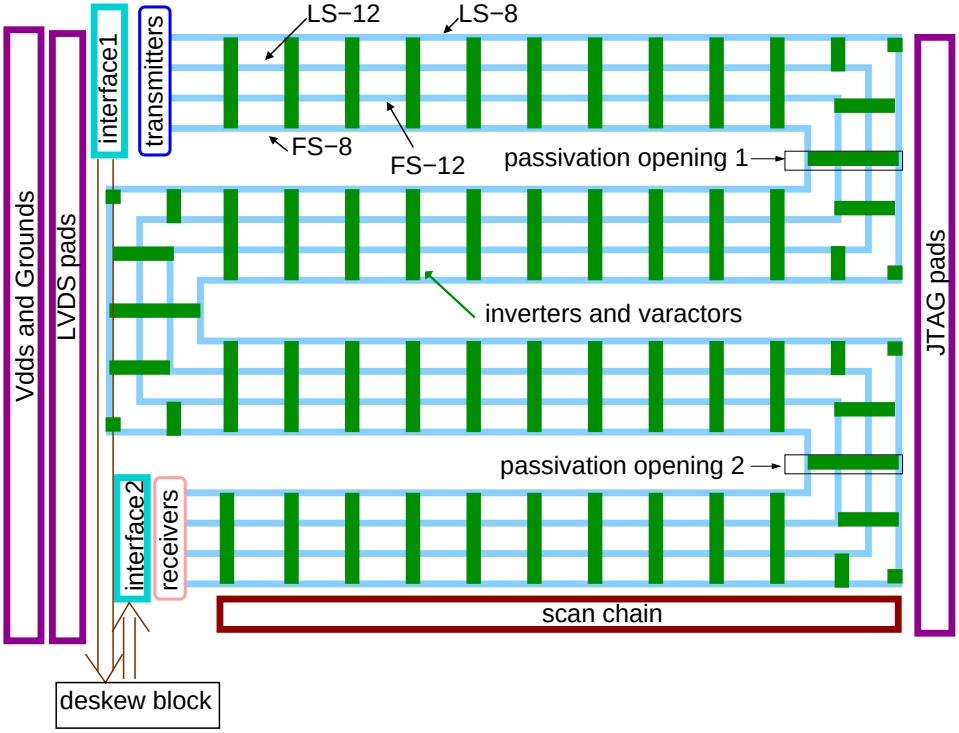


Figure 5.1: Schematic of the test chip.

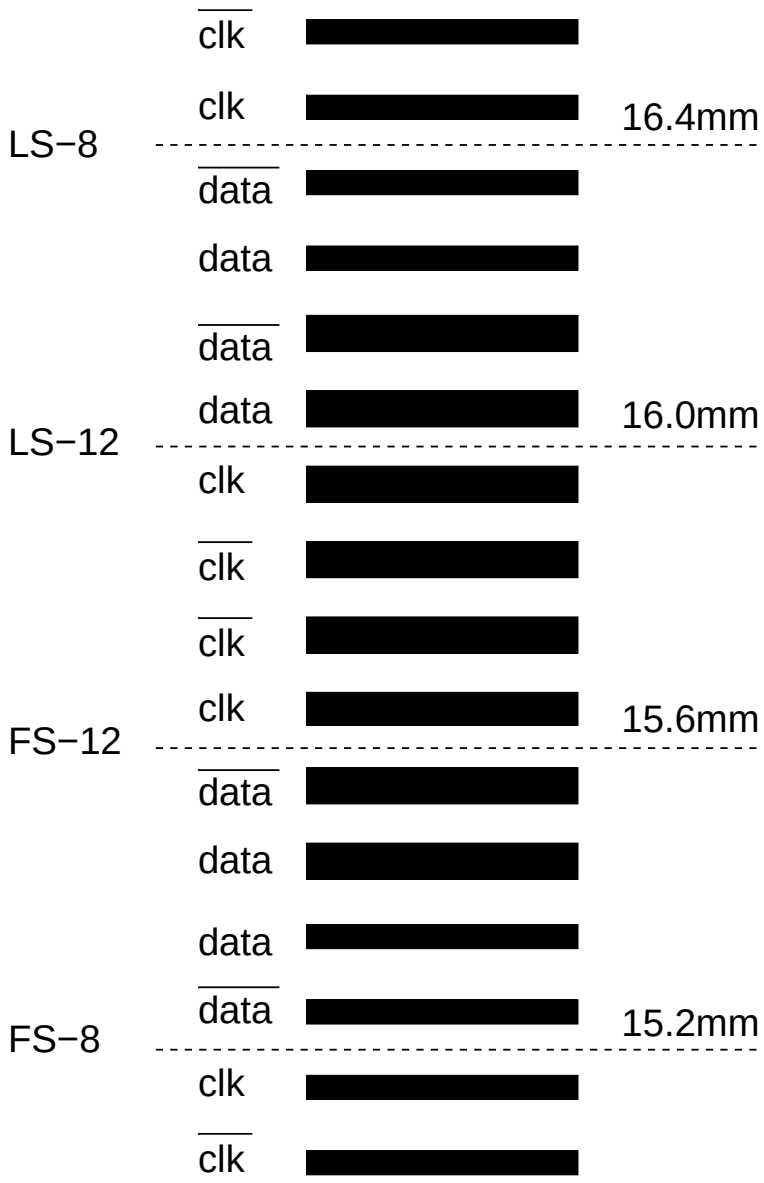


Figure 5.2: Order of signals.

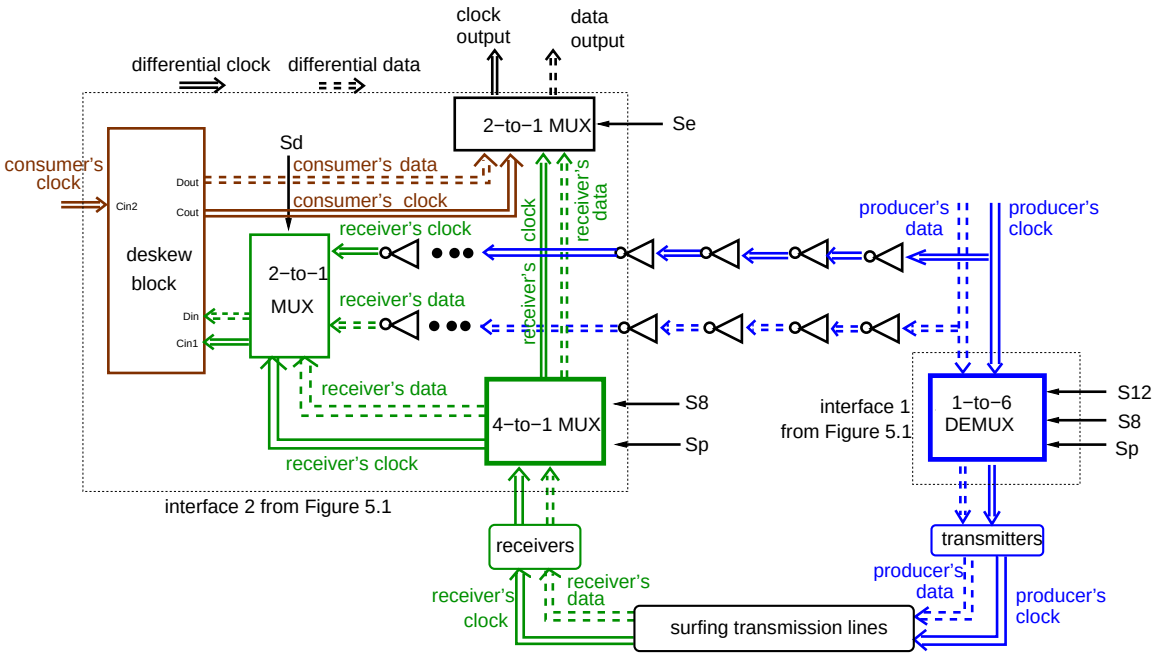


Figure 5.3: Interface to the deskew block.

Table 5.1: Truth table for the interface to deskew block.

Sp	S8	S12	Sd	Se	function
X	X	X	X	1	choose the data and clock outputs from deskew block
X	X	X	X	0	choose the data and clock outputs from surfing transmission lines
X	X	X	1	X	choose inputs for deskew block from inverter chains
X	X	X	0	X	choose inputs for deskew block from surfing transmission lines
1	X	X	X	X	enable transmitters and receivers for low-swing transmission lines
0	X	X	X	X	disable transmitters and receivers for low-swing transmission lines
X	1	X	X	X	enable transmitters and receivers for the 8μm wide transmission lines
X	0	X	X	X	enable receivers for the 12μm wide transmission lines
X	X	1	X	X	enable transmitters for the 12μm wide transmission lines
X	X	0	X	X	disable transmitters for the 12μm wide transmission lines

we used serpentine layouts for the transmission lines. We carefully considered the impact of the corners in these serpentine lines on the functioning of the design.

- To obtain high inductance and low capacitance transmission lines, we ran the surfing lines on top-level metal. The varactors and cross-coupled inverters are, of course, implemented with transistors at the bottom of the layer stack. The vias from top-level metal to the varactors and cross-coupled inverters added significant resistive and capacitive losses to the design. We had to design the via stacks carefully to minimize these parasitics.

- Mandatory metal fill increases the line capacitance above what one would expect for a metal 9 wire running over a metal 2 ground plane. We used the HSPICE field solver to guide a manual design of the metal fill layout to minimize this extra capacitance.

- The spacing between cross-coupled inverters and varactors is also a design concern. Placing the inverters and varactors too densely will introduce a large amount of parasitic capacitance. Conversely, sparsely distributed inverters and varactors will cause reflections which affects signal quality.

- The final inverters that drive the transmission lines use very wide transistors. We had to make careful use of a fingered layout to keep the RC delay of the gates comparable with the rise and fall times of signals on the transmission lines.

- For the low-swing design, the data signal at the end of the transmission line is reshaped into a narrow pulse. Careful design of the receiver is needed to detect such a narrow pulse.

We will describe the first five items in this section because they are common to both the full-swing and low-swing designs. We will describe the transmitter design in Section 5.3.2 and the receiver for the low-swing design in Section 5.4.3.

The full-swing design includes cross-coupled inverters to compensate for energy loss due to the line resistance. To satisfy Inequality 4.31, we need to maximize L_w and $-G_{eff}$ and minimize C_w, R_w and C_{inv}, the capacitance presented by the inverters. To increase L_w and decrease C_w, we use metal 9, the top-level metal in the TSMC 90nm process, to implement the transmission line and metal 1 and metal 2 as the ground plane. In the ground plane,

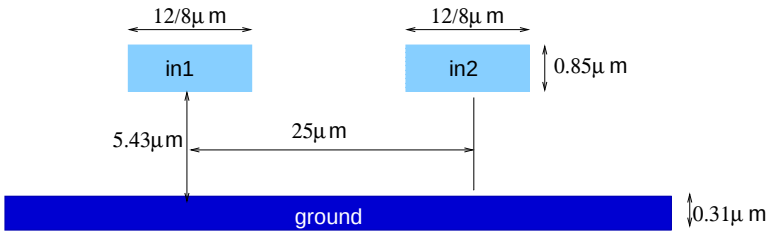


Figure 5.4: Geometry of the interconnect.

metal 2 wires run horizontally and metal 1 wires run vertically. The metal 2 and metal 1 wires are all 0.3μm wide with a 0.5μm pitch. Top-level metal layer provides the thickest wires which help to decrease R_w. To decrease R_wC_w, we use wide wires with wide separation. When increasing the width of the wire, R_w decreases faster than C_w increases because the increased wire width only contributes to the parallel-plate capacitance of the wire but does not affect the fringe capacitance. Thus, wide separation increases L_w. Using wide separation also decreases both the coupling capacitance and the mutual inductance between wires and thus improves signal quality.

The geometry of the lines is given in Figure 5.4. The distance to the ground plane is 5.43μm which is the distance from metal 9 to metal 2 in the TSMC 90nm process. The wire pitch is 25μm. The lines in FS-8 and LS-8 are 8μm wide and lines in FS-12 and LS-12 are 12μm wide. To improve the robustness to variations and disturbances, the clock lines and data lines share the same geometry in each group. Narrower wires would probably work as well. I used wide wires to maximize the design margins and thus to maximize the likelihood of the chip working on the first try. The line parameters are summarized in Table 5.2. The total line resistance $R_T = R_w + R_s\sqrt{f}$ where f is the frequency of the signal transmitted on the transmission line. As we will describe later, the transition time of the signals in this chip is roughly 15ps. We use f equal to 30GHz to calculate R_T. With wide separation between lines, the ratios of mutual inductance to self inductance and coupling capacitance to total capacitance are less than 0.1. We ignore these parameters in the table.

As illustrated in Figure 5.1, the transmission lines serpentine through the chip. At the corners, we add repeaters at both ends of the corner where the center line is 0.2mm long. As shown in Figure 5.5, the maximum length difference between the lines within one group reaches 0.15mm which is more than 50% of the normal separation between the varactors. In the chip, there

Table 5.2: Line parameters for straight lines.

		12μm wide and 13μm spacing	8μm wide and 17μm spacing
L_w	(pH/mm)	260.0	330.0
C_w	(fF/mm)	136.0	113.0
C_{fill}	(fF/mm)	45.0	60.0
C_{via}	(fF/0.2mm)	4.5	4.5
R_{via}	(Ω/0.2mm)	1.6	1.6
R_w	(Ω/mm)	1.6	2.5
R_s	($10^{-5}\Omega$/mm/Hz$^{-0.5}$)	1.4	2.0
R_T	(Ω/mm, $f = 30$GHz)	4.0	6.0
C_{inv}	(fF/0.2mm)	35.5	35.5

are 6 such corners for every group. Between two consecutive turns, the transmission lines are straight for 3.4mm. The minimum center line length among the four groups is 15.2mm. The total length of the corners is less than 10% of the total line length.

We used FastHenry [31] to study the impact of the serpentine lines on the line inductance. Compared with a straight line with the same geometry, the inductance of a serpentine line has much less inductance at low frequencies. At low frequencies, the current in the return path spread out on the ground plane. When the line serpentines, the currents in the return path are distributed in a smaller area which reduces the inductance. However, at frequencies higher than 1GHz, most of the current in the return path flows beneath the transmission line. If the separation between the forward part and backward part is wide enough, at high frequencies, the currents flowing in opposite directions in the return path for serpentine lines will not interfere with each other. In the chip, the closest distance between the section going left and the section going right is 50μm, twice of the line pitch. FastHenry indicates that at frequencies higher than 1GHz, the line inductance difference between a straight line and a serpentine line is less than 5%.

The lines at the corner have a 90-degree bend which results in extra width at the bend. This extra width will cause the instantaneous impedance to drop by at most 15% and a voltage dip of 10%. We do not care very much about the reflection and voltage dip at the corners because the varactor and cross-coupled inverters create a larger impedance change than the corners.

Instead we care more about the delay mismatches that are caused by different line length at the corners. The signal on the inner line propagates a shorter distance than the signal on the outer line. The cross-coupled inverters help to align the signals in a differential pair. Surfing helps to align the data signal with the clock signal. We expect that the corners will not affect the timing very much. We used HSPICE to confirm this. Based on the above observations, we modeled the corners as W-element [4] in HSPICE where every line is 0.2mm long and the total line inductances, capacitances and resistances of every line match those for the corners. We use FS-8 to compare straight lines with serpentine lines. As shown in Figure 5.1, the first turn in FS-8 has two consecutive corners. The second turn for FS-8 has 3 straight segments between the corners, where surfing and cross-coupled inverters help to align the signals. Thus the first turn has worse timing situation than the second turn. In this simulation, the clock signals run on the inner differential lines and data signals run on the outer differential lines. Figure 5.6 shows the waveforms for the clock and data signals. In this figure, the solid curves are signals transmitted on serpentine lines and dashed curves are signals on straight lines. The upper figure shows the waveforms of the three consecutive clock signals in the turn formed by two corners. The middle figure shows the waveforms of the three consecutive data signals in this turn. On straight lines, the clock and data signals propagate with delay roughly at 4.8ps per 0.4mm. The delays for the corners from the innermost line to the outermost line are: 3.25ps, 4.28ps, 5.06ps and 6.09ps. For one turn, the maximum delay difference among the four lines is roughly 2.8ps, or $\pm$1.4ps relative to the center line. Thus the mismatch is insignificant for frequencies lower than about $\frac{1}{2\pi 1.4ps}$, *i.e.* about 85GHz. In our simulations, the transition time is roughly 15ps. Thus, most of the energy in the signal is at frequencies of 33GHz or lower. Without surfing, we would expect the data signal at the outer lines to propagate with delays close to 5.32ps and 6.35ps. Because the clock signal at the turn propagates faster than that in the straight lines, the capacitance value of the varactor on the data line decreases to speed up the propagation of the data signal. The lower figure shows the clock and data signals 2mm away from the turn. The clock signal in serpentine lines comes earlier than that in straight lines because the serpentine clock lines are shorter than the straight clock line. However, due to the surfing effect, the data signal in serpentine lines comes at a similar time to the data signal in straight lines. Without surfing effect, the data signal in serpentine lines would come later than that in straight lines because the outer lines are longer than the straight line. In serpentine lines, after the corners, the data signal and clock signal still stay close to the stable oper-

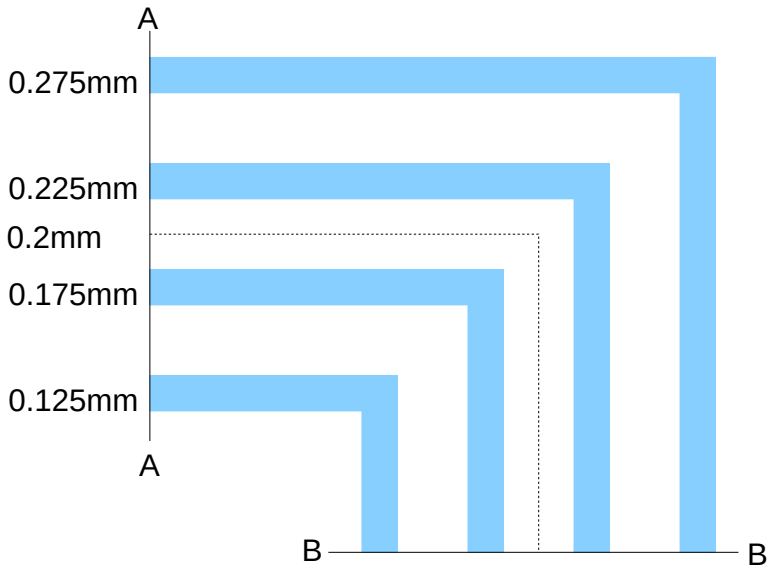


Figure 5.5: Corner of the interconnect.

ating point. This confirms our assumption that corners do not significantly affect the phase difference between clock and data signals in our design due to the surfing effect.

The vias connect the transmission line with devices at the bottom of the layer stack, such as cross-coupled inverters, varactors and the inputs to the bridges. The major concern for the via design is to reduce the added capacitances and resistances. The via's resistance and wire resistance connecting the vias and transistors in cross-coupled inverters, R_{via}, is in series with the resistance presented by the cross-coupled inverters. The conductance as seen at the metal 9 transmission line, $G_{eff,net}$, is:

$$G_{eff,net} = \frac{1}{1/(NG_{eff}) + R_{via}} \tag{5.1}$$

Note that G_{eff} is negative but R_{via} is positive. Thus, via resistance degrades $G_{eff,net}$. The obvious solution is to make a via stack with a large number of vias at every level. The problem with this solution is that the vias contribute to the total capacitance added by at each repeater stage. Figure 5.7 shows our solution. The connection from the metal 9 transmission lines to the repeater block consists of vias from metal 9 to metal 7, a metal 7 wire to get close to the repeater, vias from metal 7 to the metal 2, and a metal 2 wire

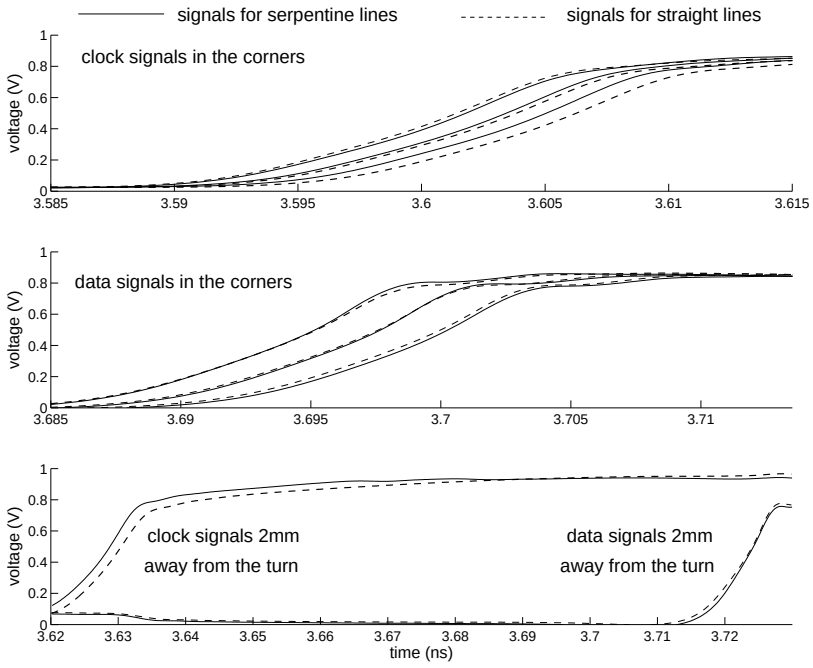


Figure 5.6: Comparison between serpentine lines and straight lines.

from the bottom of this via stack to the repeater itself. In Table 5.2, C_{via} is the total capacitance of these vias and wires. The definition of R_{via} is slightly more subtle. Each transistors in the cross-coupled inverters has six parallel fingers in the layout. R_{via} is the total resistance from the transmission line on metal 9 to one of these fingers – this is a slight overestimation, as the paths from the fingers to the bottom vias include some parallel paths. Thus, we include the resistance of the wiring inside the repeaters in R_{via}, but the capacitance of this wiring is included in C_{inv}. This distinction avoids the need to introduce a R_{inv} term in our equations and simplified analyzing various layout trade-offs. To minimize the impact of R_{via} and C_{via}, the vias from the metal 7 layer to the metal 9 layer are 5μm squares while the vias from metal 7 to metal 2 are 3μm squares. Because the upper level vias are further from the ground plane than the lower level ones, we can use larger vias for the upper levels without making C_{via} unacceptably larger. Furthermore, the glass between metal layers is thicker for the top metal layers. By using larger via arrays at the top layers, the via resistance is roughly 0.011Ω between any pair of adjacent layers for our via stack. We use a 2μm wide metal 7 wire to connect to the vias from metal 2 to metal 7, and a 0.3μm wide wire on metal 2 to connect the bottom of the via stack to the repeaters. By putting most of the length on high level metal (but away from the metal 9 transmission lines), we minimize the added capacitance for the total structure. With this design, C_{via} is roughly 4.5fF per 0.2mm and R_{via} is roughly 1.6Ω. Given $-G_{eff}$ equal to $6.5e^{-4}\Omega^{-1}$, $G_{eff,net}$ has not been degraded significantly from G_{eff} by the introduction of the via stack and wiring.

We must add fill layers for metal 3 to metal 8 to meet the design density rules. The density requirement in the TSMC 90nm process for metal 8 and metal 9 is at least 20% and 15% for the other layers for every 25μm by 25μm square. As shown in Figure 5.8, we put metal fills between the two transmission lines. To meet the requirements of the TSMC 90nm process, we have to split the metal fills for each layer into two pieces that are each 3μm wide, the maximum width allowed in this process. The spacing between fill rectangle in the same layer is 1μm, 0.2μm more than the minimum distance allowed in this process. The lengths of the fill rectangles are chosen to satisfy the density requirements. The metal 8 fill rectangles are the longest, and the metal 7 fill rectangles are a little bit longer than those on lower layers. In every 25μm by 25μm square, We added these fill rectangles. I'll now explain why we decided to place the fill rectangles between the wires of the transmission lines rather than under them. Placing the wires between the transmission lines reduces the parallel-plate capacitance to the metal 9

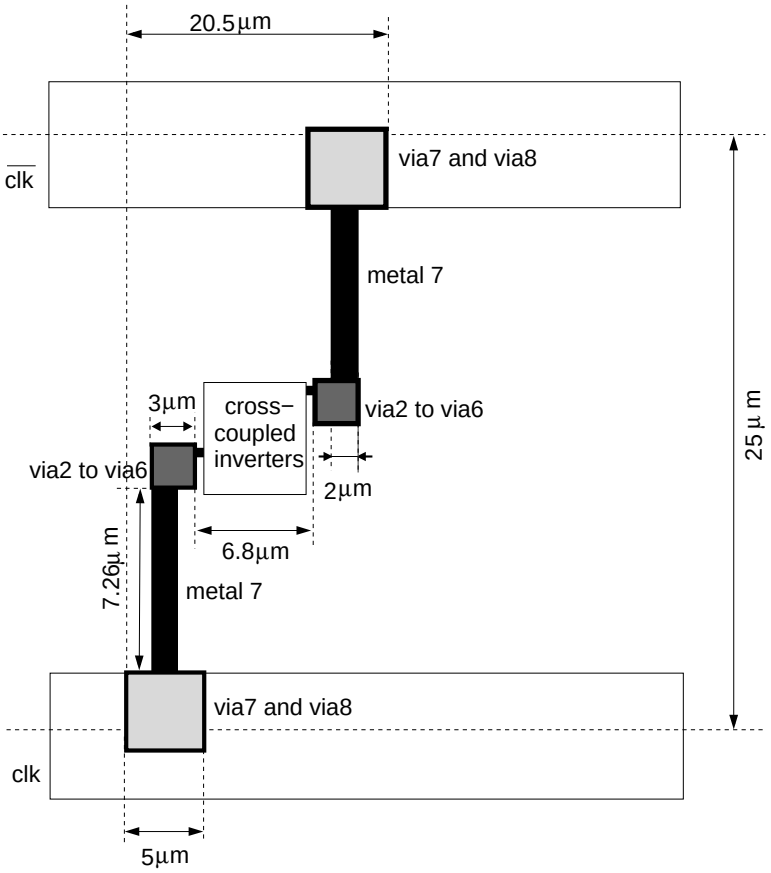


Figure 5.7: The via stack from the transmission lines (metal 9) to the cross-coupled pairs (metal 2).

wires. However, this increases the capacitive coupling between the two sides of the differential transmission lines. This doubles the effect of some fraction of the fringing capacitance and must be taken into account in the analysis. The next two paragraphs provide a quantitative comparison of the two fill placements.

Starting with the groups using 8μm wide wires, the capacitance between metal 9 line to the closer and further metal 8 fills are 24fF/mm and 3fF/mm respectively. For simplicity, we model the fills on all layers as 6μm wide continuous wires instead of small chunks. Thus for this design, the total capacitance per unit is close to the summation of the line capacitance and the peripheral capacitances. For the 8μm wide wire, the total capacitance is roughly 167fF/mm.

We now consider a design with the metal fill placed beneath the transmission line. Again, we used the 8μm wide transmission line as our example and model the fills on all layers as 6μm wide continuous wires. We use the formula in [59] to calculate the fringing capacitance. For a 6μm wide wires, the parallel-plate capacitances for layer metal 9 to layer metal 8 and layer metal 8 to layer metal 7 are both 312fF/mm. The parallel-plate capacitances between other pairs of adjacent layers are all roughly 697fF/mm. We ignore the fringing capacitance for layers metal 2 to layer metal 7 because they are small compared to the parallel-plate capacitance, noting that this underestimate the total capacitance. The fringing capacitance from layer metal 8 to the ground plane is roughly 78fF/mm. The total capacitance formed by the metal 9 and the metal fills is roughly 112fF/mm. For the 8μm wide wire without metal fills, the fringing capacitance to ground is roughly 59fF/mm and the area capacitance is 54fF/mm. With metal fill under the wires, the metal 9 fringing capacitance should stay about the same, and I'll estimate the parallel-plate capacitance as 112fF/mm for the 6μm of wire width that is over the metal fill, and $\frac{8\mu m}{2\mu m}$54fF/mm = 13.5fF/mm for the remaining 2μm of wire width. Again, this underestimates the total capacitance by ignoring the fringe capacitance from the sides of the metal fill rectangles to the metal 9 wires. With these approximations, the total capacitance is (59 + 112 + 13.5)fF/mm = 184.5fF/mm. Thus, placing the metal fills further away from the transmission line is better than placing them beneath the transmission line.

We extracted the capacitances of the metal fills and lines from the layout. We set all of the line resistances to be 0. Thus this circuit is a purely capacitive network. We drove a differential pair with differential signals and adjusted the phase of its neighboring differential pairs. The ratio between the input current to a line to the line voltage is the impedance of of the

line. We found that the coupling capacitance between neighboring differential pairs is less than 2% of line capacitance when the neighboring pairs are grounded. We will describe later that the cross-coupled inverters and varactors contribute a significant amount of capacitance to the line. The coupling capacitance between differential pairs due to the metal fills drops to less than 0.7% when those capacitances are taken into account. In the W-element for the transmission line, we model the capacitances contributed by the metal fills as line-to-ground capacitance in the worst case, the two adjacent lines in two different differential pairs transits in different directions. That's how we obtained the capacitance presented by metal fills, C_{fill} in Table 5.2. We note that metal fill increases the line capacitance by around 30%. These increase is greater for the 8μm wire than for the 12μm wide wire because the 12μm wide wire is closer to the metal fill and part of its line-to-ground capacitance is shielded by metal fill layers. We admit that our model for the metal fill layers is very simple. However, as we will show later, the total line capacitance is dominated by the capacitances contributed by cross-coupled inverter pairs and varactors and pure line capacitance. We expect that some deviation from the metal fill capacitances we used in the model will not lead to a big change in the propagation speed.

In the actual layout, the fill is a sequence of rectangles, rather than continuous wires. This means that the current in these rectangles is negligible, and they have very little impact on the inductance. Thus, adding fill increases the total line capacitance but it does not decrease L_w. As a result, the propagation velocity drops to less than the speed-of-light in glass.

To make $-G_{eff}$ big, we use large transistors in the cross-coupled inverters. However making these transistors too large increases C_{inv}, eventually pushing the design against the limit case where the capacitance contributed by the inverters dominates the wire capacitance. Furthermore, increasing C_{inv} decreases the propagation velocity of the line, and the reason for using LC-mode interconnect is the higher velocity. More importantly, increasing C_{inv} will make it more difficult to satisfy the surfing requirement for the full-swing design. As described in Section 4.3, to achieve 10% delay variation, the maximum varactor capacitance should be roughly 50% of the fixed total line capacitance which includes the line capacitance, capacitance presented by the cross-coupled inverters, capacitance introduced by the metal fills and parasitic capacitance of the varactors. Increasing C_{inv} means increasing $C_{v,d}(0)$ and further decreasing the propagation velocity for the full-swing design. In all four designs, the NMOS and PMOS devices in the cross-coupled inverters are 3μm and 7.5μm wide. The capacitance presented by the cross-coupled inverters, C_{inv} is roughly 35.5fF. This number is larger than what

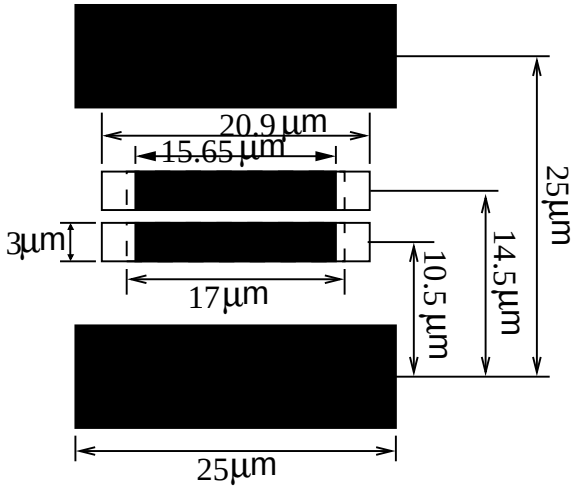


Figure 5.8: Metal fills for the transmission lines.

we get by using NC_{eff} which yields 29.1fF. The extra amount is mainly due to the wiring capacitance in the cross-coupled inverters. To reduce the parasitic capacitance, all the varactors are 0.55μm wide, the minimum width allowed in this process.

The spacing between consecutive repeaters affects the power consumption and propagation of reflected waves in the system. Densely distributed inverters and varactors reduce the reflection in the interconnect and thus reduce jitter. However, more densely distributed cross-coupled inverters and varactors will introduce more parasitic capacitances, such as via capacitance, the parasitic capacitance of the varactors and wire capacitance in the cross-coupled inverters. The parasitic capacitances reduce the characteristic impedance. The increased total capacitance increases power consumption if we use a voltage source to drive the transmission line. More importantly, the increased parasitic capacitance decreases the surfing variation. The design goal is to place the inverters and varactors as densely as possible and at the same time, in every segment, keep the parasitic capacitance much less than the maximum total capacitance of that segment. We placed repeaters every 0.2mm along the center lines for every group. In our design, the variable capacitances from FS-8, FS-12, LS-12 and LS-8 are roughly 117fF/mm, 145fF/mm, 177fF/mm and 192fF/mm. The parasitic capacitances in these groups are roughly 75fF/mm.

The total data line capacitance is the summation of $(C_{inv} + C_{via} + C_{v,d}(0))/y + C_{fill} + C_w$. We introduce another term C_{extra}, the extra capacitance that the cross-coupled inverters can tolerate per 0.2mm with C_{inv}, C_{via} and C_{fill} given. We replace $C_{v,d}(0)$ in Equation 4.31 with C_{extra} and rewrite this equation in as:

$$C_{extra} \leq \frac{-G_{eff,net} L_w}{R_T} - C_{inv} - C_{via} - (C_{fill} + C_w)y \tag{5.2}$$

where $-G_{eff,net}$ is $9.75e^{-4}\Omega^{-1}$ as calculated by Equation 5.1. $C_{v,d}(0)$ must be smaller than C_{extra} to send a full-swing signal without attenuation. For both 12μm wide and 8μm wide transmission lines, C_{extra} is less than 0. If we replace R_T with R_w in the above inequality, C_{extra} for the 8μm and 12μm wide transmission lines are 53fF and 86fF per 0.2mm. The skin effect resistance of the transmission lines affects the attenuation severely. However, as for the conductance presented by the cross-coupled inverter pairs, the line resistance is also a time varying function. Using R_w is optimistic and using R_T at 30GHz is too pessimistic. Note that in the calculation of G_{eff} in Section 4.2.4, we assumed that the differential voltage is from $-V_{DD}$ to V_{DD}. However, if the line resistance is too big, the signals on the LC line cannot reach full power rail, which helps to increase $-G_{eff}$. In the design process, we use $-G_{eff}$ and R_w to find the minimum size of the cross-coupled inverters provided with a rough estimation of the total line capacitance. We then increase the size of the cross-coupled inverters by 50% to compensate for the energy loss due to skin effect resistance. We increases $C_w + C_{fill}$ by 10% and decreases L_w by 80% and run 5 corner simulations to ensure that we still have enough design margins. We also simulated a 80mm long, 12μm wide wire in the full-swing design with the degraded inductance and capacitance. The clock and data signals can still propagate to the end of lines and the data signal surfs on the rising edge of the clock signal. The edges of the clock signals are roughly between 0.05V and 0.7V. We concluded that the size of the cross-coupled inverter are big enough in the four designs.

5.3 Full-swing Design

We started with the full-swing design as shown in Figure 5.9. This design includes three parts: surfing transmission line, transmitter and receiver. For the surfing transmission line, the main design constraint is to meet the surfing requirement. The transmitter and receiver design is described in Section 5.3.2 and 5.3.3. The transmitter must have enough driving strength.

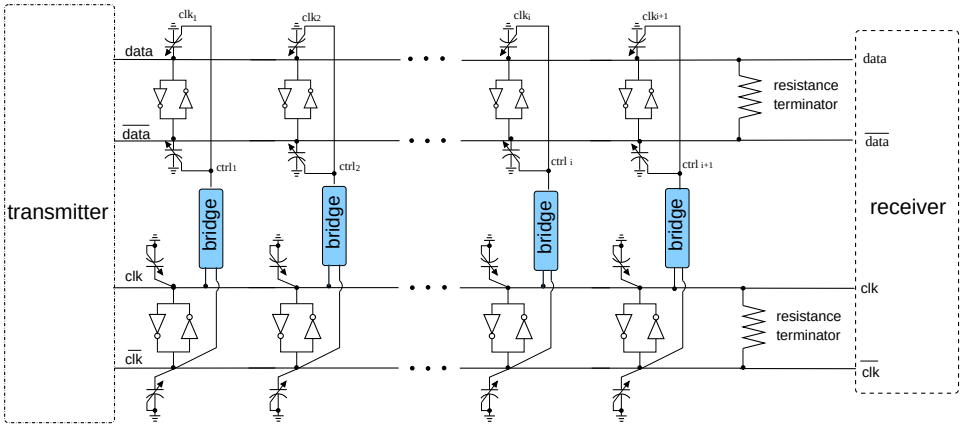


Figure 5.9: Schematic of the full-swing design.

Surfing makes the design of the receiver very easy: we use a Strong-Arm latch to detect the differential signals.

5.3.1 Surfing Interconnect

Figures 5.10 and 5.11 present the surfing interconnect for FS-8 and FS-12. The surfing interconnect includes 3 parts: transmission lines for the clock and data signals, bridge circuit and the terminating resistor.

The bridge circuit is the same as shown in Figure 4.25. This circuit is used to connect the clock signal to the varactors to control their capacitance values. The delay of the bridge circuit is roughly 98ps. Thus at a stable state, the data signal will propagate roughly 98ps behind the clock signal.

The cross-coupled inverters in the clock line and data line are the same size; they have NMOS transistors that are 3μm wide and PMOS transistors that are 7.5μm wide. The varactors for the data line are all 0.55μm wide, 1.75μm long for FS-8 and 2.2μm long for FS-12. By using long channel devices, the gate capacitance dominates the parasitic capacitance. The effect of the varactor's capacitance on the propagation speed of a LC line is equivalent to a capacitor of roughly 12fF for every micron of gate length in the TSMC 90nm process. Because the PMOS and NMOS varactor MOSFETs are of the same size, we use the size of the NMOS device to denote the size of the varactor. To meet the surfing requirement, we also include varactors in the clock lines in FS-8 and FS-12 as shown in Figures 5.12 and 5.13. The control signals s_0, s_1 and their complements are set by the scan chain.

The minimum capacitance of the clock line $C_{c,11}$ is achieved by setting s_0 and s_1 to be high. Correspondingly, setting s_0 and s_1 to be low sets the maximum capacitance value of the clock line, $C_{c,00}$. These varactors in the clock line allow us to vary the delay of the clock line during chip test to improve testing flexibility. The total width of the varactors in the clock line should be less than that in the data line to satisfy the surfing requirement. Table 5.3 summarizes the delays for FS-8 and FS-12 from HSPICE simulations. For both groups, the minimum and maximum capacitances of the clock lines are roughly the same because the total line capacitance consists of line capacitance, metal fill capacitance and the capacitance presented by the cross-coupled inverters. The delay difference in these two groups is mainly caused by the line inductance. For both groups, the minimum delay of the clock line is slightly slower than that of the data line. This is because the clock line drives the bridge circuit which presents some capacitive load to the clock line. The delay variation for the 8μm and 12μm wide wire is roughly 7.5% and 8.4%, less than 10%. For both groups, the variable capacitance of the data line is less than 50% of the fixed line capacitance, $C_{d,min}$. This is due to our underestimation of the metal fill capacitance in the early stage of design process. However, the propagation speed of the transmission line is robust to power supply variation and process parameter variation. We varied the power supply from 0.8V to 1.2V and the propagation speed of the clock line stays nearly the same. We also ran 5 corner HSPICE simulations and the variation of the propagation speed of the clock line is not noticeable. We examined timing variation due to coupling capacitances and mutual inductances. The added capacitance of the cross-coupled inverters and varactors decreases cross-talk noise due to coupling capacitance. Note that the coupling capacitance in this line geometry is roughly 10% of the pure line capacitance. With the capacitance presented by the inverters, metal fills and other parasitic capacitance, the coupling capacitance becomes less than 4% of the total line capacitance. Thus, cross-talk noise due to coupling capacitance will cause 2% timing variation. However, the added capacitance does not affect the line inductance. HSPICE simulations show that for these two groups, the delay variation caused by mutual inductance is less than 3.5%, which is less than half of the delay variation that the varactors can induce on the data lines in both groups. We conclude that the delay variation caused by the surfing effect is sufficient for these two line geometries.

We use polysilicon to implement the resistance terminators in the data lines. The polysilicon resistors for FS-8 and FS-12 exhibit roughly $\pm$30% variation in three corner HSPICE models relative to the typical-typical cor-

Table 5.3: Line parameters for straight lines in FS-8 and FS-12.

		FS-8	FS-12
$C_{fill} + C_w$	(fF/mm)	190	200
$C_{c,11}$	(fF/mm)	420	417
$C_{c,00}$	(fF/mm)	486	529
$C_{d,min}$	(fF/mm)	382	395
$C_{d,max}$	(fF/mm)	499	540
$d_{c,min}$	(ps/mm)	11.3	10.4
$d_{c,max}$	(ps/mm)	12.5	12.2
$d_{d,min}$	(ps/mm)	11.2	10.1
$d_{d,max}$	(ps/mm)	13.0	12.3

ner. Taking FS-8 as an example, the polysilicon resistor for the data line is 2μm long and 2.03μm wide. This resistor exhibits 120Ω, 88Ω and 148Ω at the typical-typical, fast-fast and slow-slow corners in HSPICE simulation. Because the clock line's period is half of the maximum data period, the clock signal has less time to settle and requires a more sophisticated design for the terminating resistor. In the clock line, we make the polysilicon resistor narrower to ensure that its minimum resistance is roughly 120Ω. The polysilicon resistor for the clock line is 2μm long and 1.4μm wide and exhibits 171Ω, 127Ω and 217Ω at the typical-typical, fast-fast and slow-slow corner HSPICE simulation. The resistance variation can be as much as 70%. To control the terminating resistance, for the clock line, we use NMOS and PMOS transistors as shown in Figure 5.14 in parallel with the polysilicon resistor. The sizing of the transistors in this figure is for FS-8. When the transistors are in their saturation regions, their resistances are very big compared to the resistance of the polysilicon resistor. When the transistors are in the linear region, the resistance for the bigger NMOS and PMOS transistors is roughly 400Ω at the typical-typical corner. For the smaller NMOS and PMOS devices, the resistance in the linear region is roughly 800Ω. Note that the clock signal is full-swing. Thus only the NMOS or PMOS transistors will work in the linear region. With these transistors, we can control the resistance variation to within 10% relative to typical-typical corner.

5.3.2 Transmitter

The main challenge for the transmitter is that it must be able to drive full-swing signals into a low impedance line. From Table 5.2, the nominal

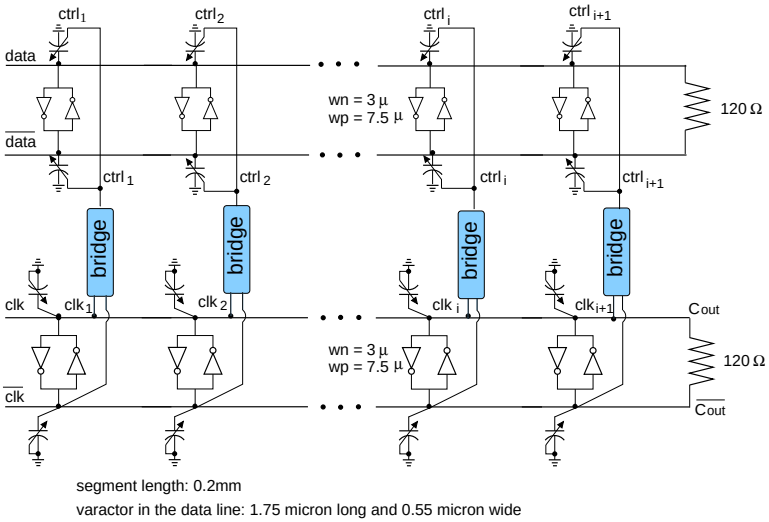


Figure 5.10: Surfing interconnect (wire width = 8um).

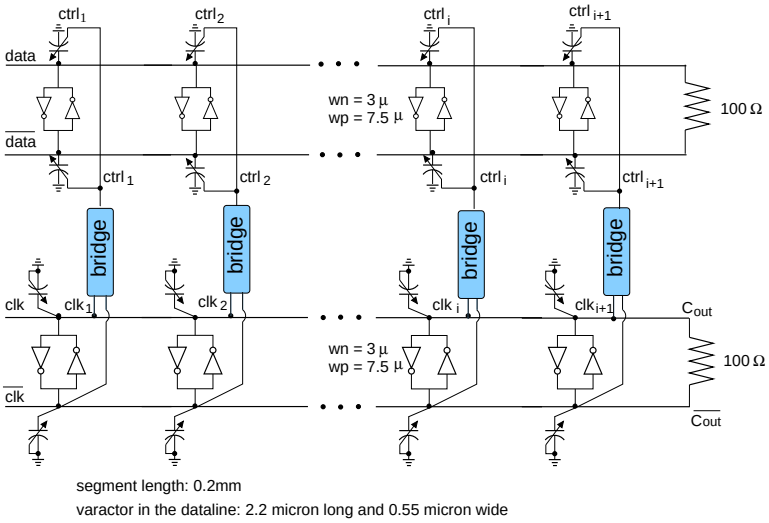


Figure 5.11: Surfing interconnect (wire width = 12um).

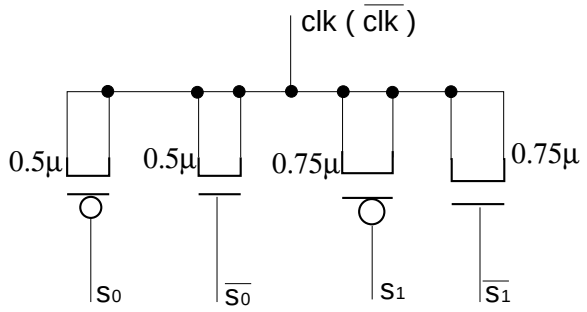


Figure 5.12: Varactor on the clock line (wire width = 8um).

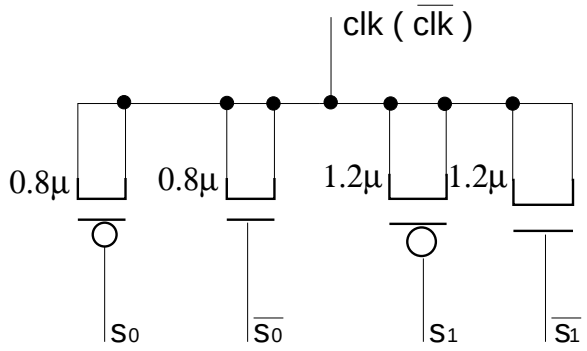


Figure 5.13: Varactor on the clock line (wire width = 12um).

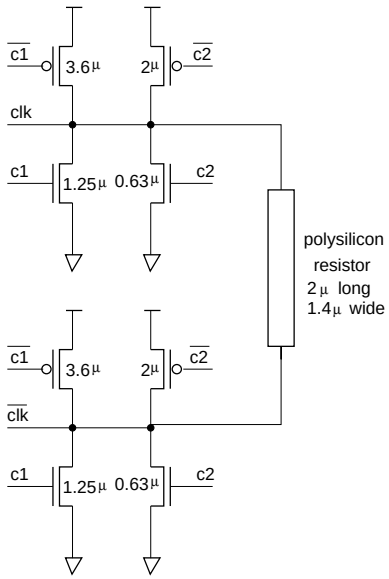


Figure 5.14: Implementation of the terminating resistors for the clock line.

impedance of the lines for FS-8 and FS-12 are 54Ω and 44Ω. However, when we include the capacitance of cross-coupled inverters, metal fill, varactors and other parasitic capacitances, both impedances drop to roughly 25Ω. To drive such a low impedance line, we need a transmitter with an output resistance less than 11Ω to drive the cross-coupled inverters into the amplification region. This requires a large transmitter and special layout techniques are needed for such a large transmitter. We use differential signals. From the input pads to the input of the transmitter, the differential signals may experience different delay. In the transmitter, the delay for the rising edge and falling edge of the signal will also be different. We need to compensate for the delay mismatch.

Figure 5.15 presents the transmitter for FS-8 and FS-12. The data line and clock line share the same transmitter design because their impedances are close to each other. The first two stages of the transmitter have cross-coupled inverters between the differential signals to compensate for the delay mismatch. Each stage reduces the delay mismatch by roughly 30%. Thus, after the first two stages, the delay mismatch will drop by 50%. Adding cross-coupled inverters to later stages would consume extra area and power and simulations indicated that cross-coupled inverters were not needed for

the later stages. The cross-coupled inverters along the transmission line remove any residual misalignment as the signal propagates down the line.

The last stage of the transmitter has a very large inverter with the NMOS and PMOS transistors being 110μm and 280μm wide. The last inverter can be modeled as a voltage source in series with a 8.6Ω resistance. Our simulations demonstrated that an inverter with a 72μm wide MOS transistor and 202μm wide PMOS transistor could also drive these low impedance wires. However, we intentionally overdesigned this inverter to ensure some design margin for this first, proof-of-concept chip. This extra margin comes at a cost of increased power consumption. This large transmitter can generate a voltage swing from 0.075V to 0.78V on the transmitter's side of the interconnect which is enough to flip the state of the cross-coupled inverters.

The large transistors in the output stage of the transmitter introduced another challenge: the RC delays of the gate and drain networks. If each inverter were implemented with a single, very wide NMOS transistor and a very wide PMOS transistor, then the RC delay of these polysilicon gates would be much larger than the desired rise and fall times of the inverter outputs. The solution to this problem is to use a fingered design where each of these transistors connected in parallel. Again, care must be taken. For example, if final inverter were broken into 110 inverters and every inverter has a 1μm wide NMOS and 2.8μm wide PMOS devices, the RC delay of the metal wires that connects the gates together and that for the drains would be larger than the desired rise and fall times. Note that these inverters can have output rise and fall times that are less than the unloaded output transition times. This is because the line acts like a distributed amplifier due to the line inductance. Here, I examine how to choose a feasible number of fingers for these transistors so as to optimize the RC delays of the gate and drain circuits. In the following, we will use the PMOS transistor as an example to demonstrate our solution.

Supposing that the finger number is m. We may model the PMOS transistor as illustrated in Figure 5.16. We use metal 2 layer to distribute the input and output. The pitch between the PMOS transistors is 0.4μm. The metal 2 wires are 0.3μm wide and the line resistance, R_2, is roughly 0.09Ω per finger. The resulting maximum line capacitances connecting the transistors is less than 0.1fF which is much smaller than the gate capacitance. Thus in the following calculation, we ignore the line capacitance between transistors. As illustrated in Figure 5.17, we model the gate of each transistor as a line with distributed resistance R and capacitance C which is around 2fF/μm. The polysilicon resistance, R, is 110Ω/μm which is the upper bound of the P+ polysilicon and N+ polysilicon silicide resistance provided by the TSMC

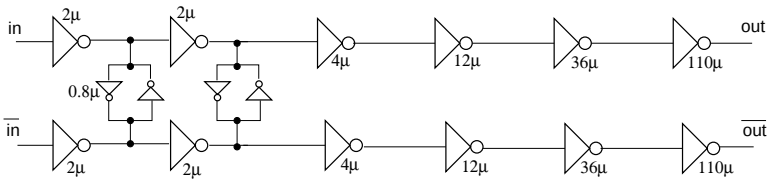


Figure 5.15: Transmitter of full-swing design.

device model manual. The total resistance of vias connecting metal 2 to polysilicon are 16Ω per transistor. We ignore this resistance in the following calculation because it is much smaller than R. We applied the Elmore delay model to this circuit. Assuming the total width of transistors are $W\mu$m, the delay to propagate the input to the end of the polysilicon gate is:

$$D_T = \frac{RCW^2}{2m^2} \tag{5.3}$$

The delay to reach the last transistor is:

$$D_2 = \frac{R_2CWm}{2} \tag{5.4}$$

D_2 is the time difference for the input to reach T_1 and T_m. The ratio of this time difference to D_T is:

$$\frac{D_2}{D_T} = \frac{R_2}{RW}m^3 \tag{5.5}$$

This ratio is proportional to the cube of m. For example, if W is 280μm and m is 70, the time difference D_2 is roughly equal to D_T. This means that while T_1 tries to pull the output high, T_m still drives the output low. Thus overfolding transistors will cause fighting between the smaller transistors. Our goal is to enable all of the transistors at the same time and we set m to 28 in which case D_2 is less than 10% of D_T. D_T is roughly 20ps. The rising and falling edge of the output is roughly 25ps. We admit that this is not an optimal design. For example, we can use a wide metal 2 wire to reduce R_2 and further decrease D_2.

5.3.3 Receiver

We use a Strong-Arm latch as the receiver shown in Figure 5.18 to synchronize the data signal to the forwarded clock signal. The setup and hold times of the Strong-Arm latch are −7ps and −40ps respectively. For FS-8,

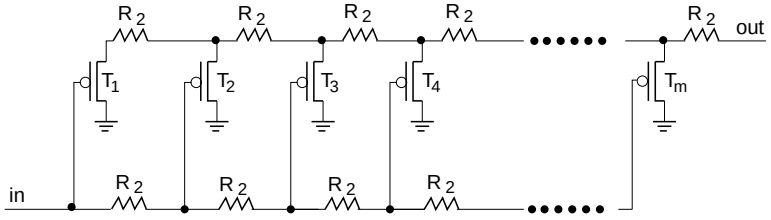


Figure 5.16: Model for a folded transistor.

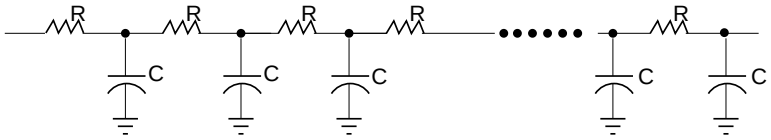


Figure 5.17: RC model for the gate of a transistor.

the sampling clock of the latch, ψ, is shifted by a fixed delay from the clock output of the clock line, C_{out}. Assume that the period of the clock signal is P. As we indicated previously, the data signal comes later than C_{out} by roughly 98ps. The fixed delay element delays C_{out} by roughly 80ps. The data signal comes earlier than the rising edge of the current sampling clock event by $P/2 - 18$ps and later than the the rising edge of the previous sampling clock event by $P/2 + 18$ps. Thus the setup and hold time of this latch is satisfied for a wide range of clock period.

To obtain a more robust design, for FS-12, we use a variable delay chain as shown in Figure 5.19 to replace the fixed delay element implemented with three inverters in series in Figure 5.18. In this figure, `s0` and `s1` are the control signals to select which delay path to choose. The thick line represents the positive part and the thin line the negated part of the differential signal. Notice that in the left section, the upper path has one inverter whereas the lower section has two; thus I introduced a twist in the lower path, exchanging the positive and the negative sides of the differential pair, to preserve the clock polarity for all settings of the delay. Table 5.4 summarizes the delay of this variable delay element, Δd. By selecting `s0` and `s1`, Δd can change from 151ps to 222ps. This delay variation range is larger than the vulnerable time window defined by the setup and hold times of the strong-arm latch. Thus this design is more robust to noise and disturbances.

We use a tri-state buffer as shown in Figure 5.20 in the variable delay

Table 5.4: Delay of the variable delay element (simulated).

s0	s1	Δd (ps)
0	0	222
1	0	198
0	1	175
1	1	151

element to improve the bandwidth compared with a design using a traditional tri-state inverter on the left side. This tri-state inverter design reduces the drain capacitance contributed by the transistors. Otherwise the output signal becomes nearly sinusoidal at 5GHz. We do not include keepers in the buffer because the periodic switching of the clock ensure that the MUX output never floats for an extended time. Omitting keepers further improves the bandwidth of the programmable clock delay line.

5.3.4 Simulation Results

We simulated the whole circuits for FS-8 and FS-12. They both work in HSPICE five corner simulations. Table 5.5 summarizes the power consumption for each major component in FS-8 and FS-12 with the clock's frequency equal to 4GHz and the data signal's frequency equal to 2GHz. The clock signal in FS-12 propagates faster than FS-8 mainly due to the lower line inductance in FS-12. The line resistance in FS-12 is less than that in FS-8. As we have expected, the clock line in FS-12 consumes less energy than that in FS-8. For both designs, the clock line consumes more energy than the data line. Note that the clock signal does not reach the full power rail and the bridge circuits consume some short circuit power. The short-circuit power in the bridge depends on the clock waveform, and this waveform is different at different repeater stages. Table 5.5 reports the bridge power for a full, rail-to-rail clock waveform, and the extra short-circuit power arising from the partial swing of the clock signal is attributed to the clock power. Even without the short circuit power, the bridge circuits still consumes more energy per bit than the cross-coupled inverters. In the future, we will investigate new designs for the bridge circuit to reduce its power consumption.

The transmitters in the two designs consume nearly the same amount of energy per bit. However, due to the the variable delay element in FS-12, the receiver in FS-12 consumes 0.5pJ per bit more than FS-8. In this table, we do not enable the transistors used as terminating resistance in both designs

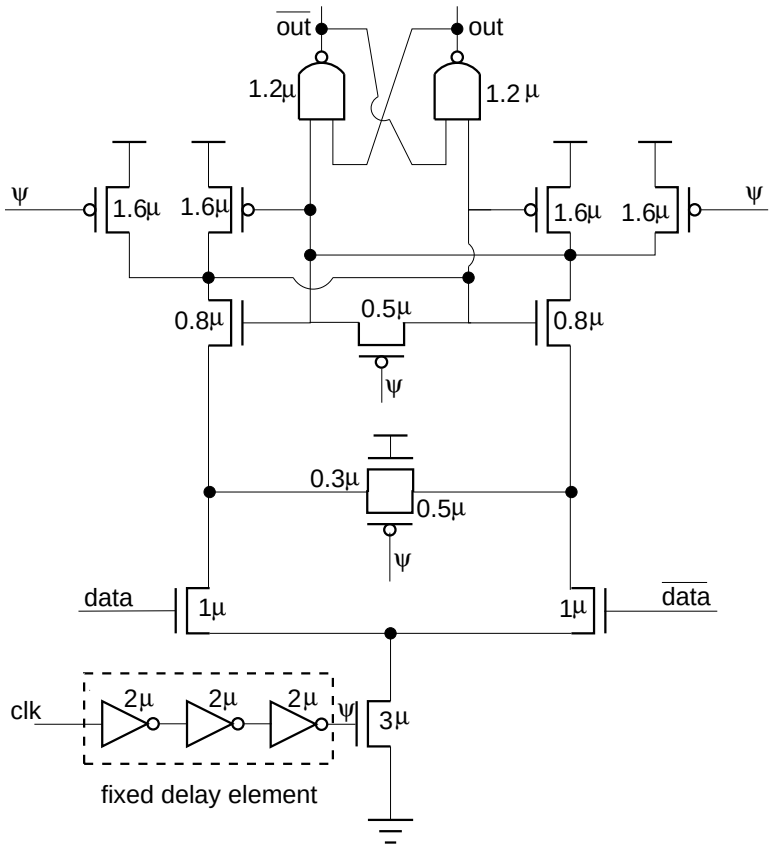


Figure 5.18: Receiver for the full-swing design.

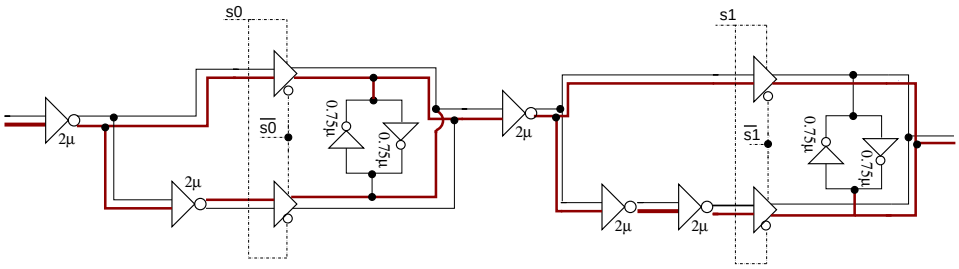


Figure 5.19: Variable delay element for the receiver in FS-12.

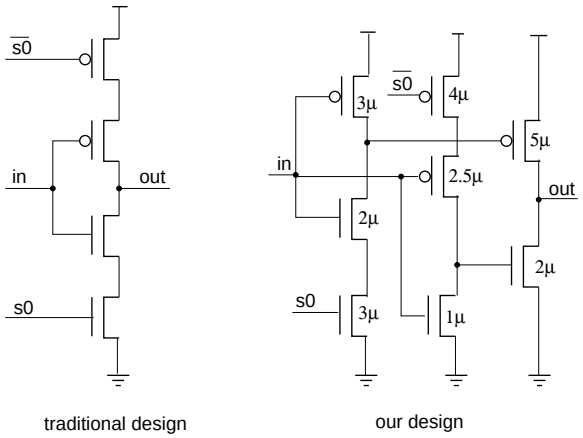


Figure 5.20: A tri-state buffer.

for a fair comparison. These transistors will contribute some short circuit power consumption to both groups.

We are not surprised that the power consumption of the two groups are nearly the same if the receiver's power consumption is not taken into account. As shown in Table 5.5, the added metal fill capacitance, cross-coupled inverter capacitance and varactor capacitance is roughly 70% of the average data line capacitance. These added capacitances are roughly the same for both groups which makes the line impedances roughly the same.

5.4 Low-swing Design

The schematic of low-swing design is shown in Figure 5.21. This design also includes three major components: a transmitter, surfing interconnects and a receiver. In the data line of the low-swing design, we removed the cross-coupled inverters to reduce the total line capacitance. The data signal in the low-swing design is a return-to-zero signal. We use transmission gates to drive the data line. The data signal at the end of the line is a pulse with roughly 40ps duration, a little less than 0.5 FO4 for the process. Careful receiver design is needed to detect such a narrow pulse. We will describe this in Section 5.4.3.

Table 5.5: Power consumption of major components in FS-8 and FS-12 (Simulated).

	FS-8	FS-12
delay of clock line (ps/mm)	12.0	11.2
line length (mm)	15.2	15.6
clock line (pJ/mm/bit)	0.42	0.37
bridge circuit (pJ/mm/bit)	0.98	1.01
power consumption for data line (pJ/mm/bit)	[0.25, 0.32]	[0.23, 0.32]
total clock+data+bridge (pJ/bit)	[25.1, 26.1]	[25.1, 26.5]
transmitter (pJ/bit)	5.5	5.4
receiver (pJ/bit)	0.6	1.1
total power (pJ/bit)	[31.2, 32.2]	[31.6, 33.0]

5.4.1 Low-swing Surfing Interconnect

Figures 5.22 and 5.23 present the surfing interconnects for LS-12 and LS-8. The bridge circuit is the same as in the full-swing design. The clock line still has cross-coupled inverters to compensate for the energy loss due to the line resistance. Although we removed the varactors in the clock line, we used the same size for the cross-coupled inverters for simplicity. Thus the clock signals in LS-12 and LS-8 propagate faster than those in FS-12 and FS-8. The data lines are biased at 0.5V, half V_{DD} in this process. We put multiple sets of varactors in the data line. The two groups use the same varactor design as presented in Figure 5.24 where the varactors controlled by s_0, s_1 and their complementary signals $\overline{\mathsf{s}_0}$ and $\overline{\mathsf{s}_1}$ are used to adjust the propagation speed of the data line to achieve testing flexibility. As given in Table 5.6, the minimum data line capacitances of both groups are much less than the clock line capacitances. When all of the varactors in the data line are enabled, the maximum data line capacitances are greater than those of the clock lines. The surfing requirement is satisfied and a narrow pulse will build up along the data line. When the varactors controlled by s_0 and s_1 are not enabled, for both groups, the data line capacitance with ctrl equal to 0 is nearly the same as the clock line capacitance. Note that the data line and clock lines are lossy, and the line resistance reduces the propagation speed of the signals. The line resistance slows down the data signal more than the clock signal because the data line does not have cross-coupled inverters to retore the high frequency components in the leading edge. The leading

Table 5.6: Line parameters for straight lines in LS-8 and LS-12.

		LS-12	LS-8
$C_{fill} + C_w$	(fF/mm)	200	190
C_c	(fF/mm)	410	403
$C_{d,min}$	(fF/mm)	271	263
$C_{d,max}$	(fF/mm)	448	455
d_c	(ps/mm)	10.2	11.0

edge of the data signal becomes smoother and smoother. Thus, the line resistance in the data line helps to extend the surfing interval. As described Section 4.3.2, the best timing of the rising edges of the ctrl signal at the transmitter side is between the leading and trailing edges of the data pulse. We intentionally design the varactors such that the maximum propagation speed of the high frequency component of the data signal is slightly slower than the propagation speed of the clock signal such that the clock signal gradually catch up with the data signal. In this way, at the end of the 17mm long wire, the clock signal still lags behind the data pulse and the data pulse still grows in height. Because surfing ensures that data pulses propagate at the same rate as clock edges, we report the propagation speed of clock edges in Table 5.6 to describe the performance of the design.

We use polysilicon to implement the terminating resistor for the data line. For the clock lines in LS-12 and LS-8, the terminating resistors are the same as in the clock lines in FS-12 and FS-8. The two designs work in HSPICE 5 corner simulations even when the varactors controlled by s_0 and s_1 are not enabled.

5.4.2 Transmitter

Because the clocks in LS-12 and LS-8 are still full-swing signal, we use the same transmitter as in FS-8 and FS-12 to drive the clock lines in the low-swing design. The clock signal at the transmitter's output swings between 0.06V to 0.81V for the edges. The data signals in the low-swing design are return-to-zero, differential signals. As described in the previous section, the rising edge of the ctrl signal for the varactors should occur between the leading and trailing edges of the data pulse. We use a simple design as shown in Figure 5.25 to meet these requirements. As shown in Table 5.6, the total data line capacitances for LS-12 and LS-8 are almost the same.

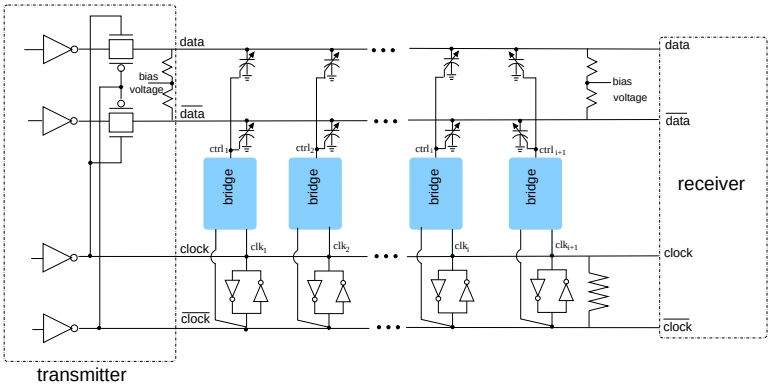


Figure 5.21: Schematic of low-swing design.

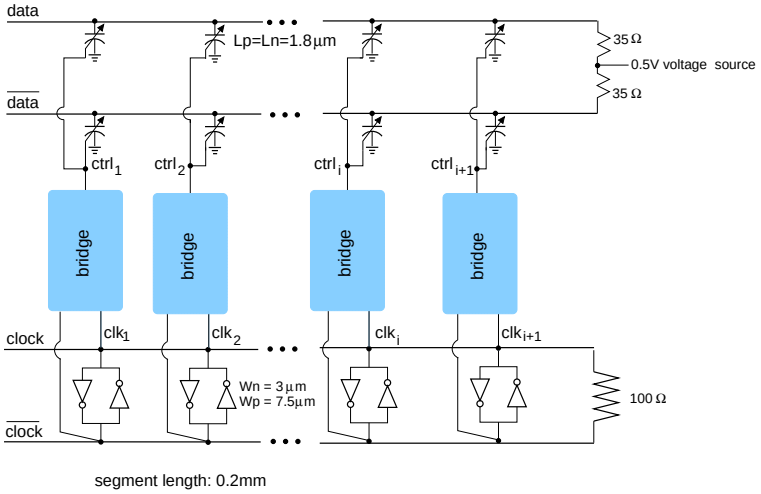


Figure 5.22: Surfing interconnect of low-swing design (width = 12um).

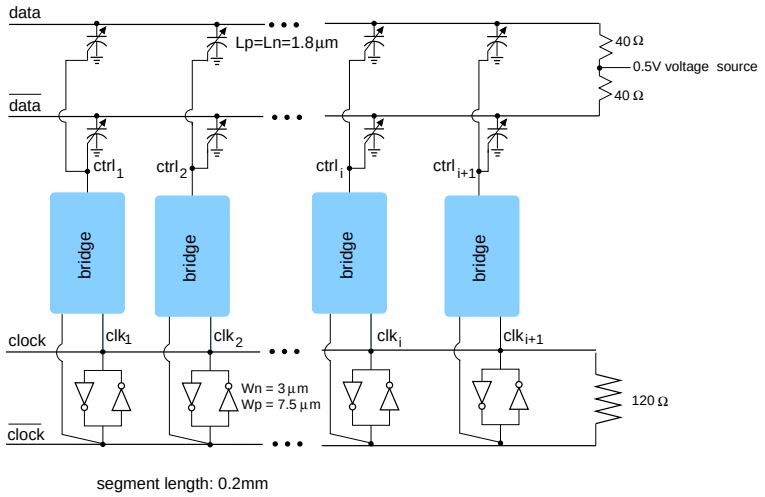


Figure 5.23: Surfing interconnect of low-swing design (width = 8um).

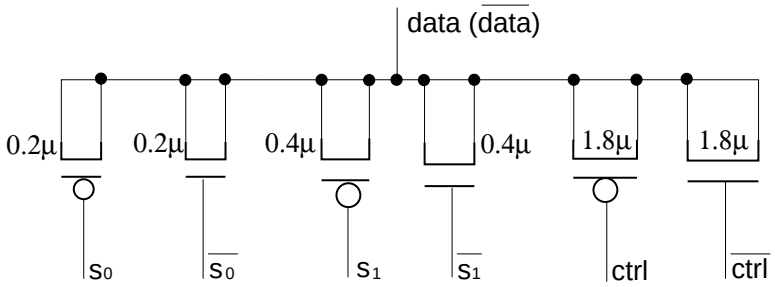


Figure 5.24: Varactor on the data line.

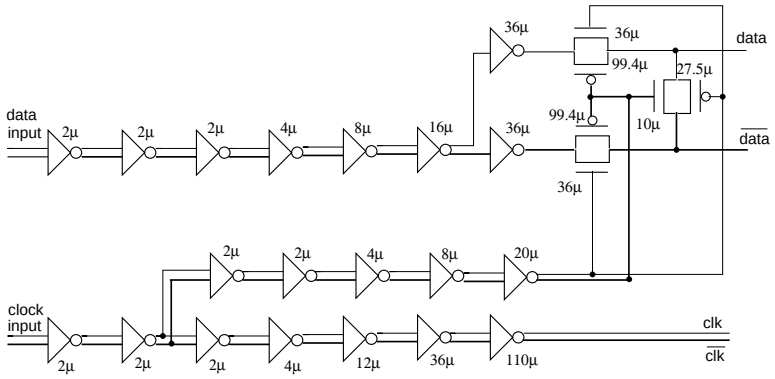


Figure 5.25: Transmitter for the data line in the low-swing design.

For simplicity, the transmitters driving the data line for LS-12 and LS-8 are of the same size. In this figure, we use the thin and thick lines to denote the differential signals. We use another inverter chain to duplicate the clock signal and use it to control the transmission gates for the data line. The data signal is launched when the clock signal is asserted. However, the bridge circuits delayed the clock signal by roughly 98ps. At the transmitter side, the data signal comes earlier than the ctrl signal for the varactors by roughly 90ps. This ensures that for a 17mm long wire, the clock signal never catches up with the data signal and the data pulse gets higher and higher.

The last inverter for the data line is only 36μm which is much smaller than the last inverter for the clock line. The transmitter for the data line has output resistance roughly equal to 100Ω. This produces a differential signal with roughly 0.3V voltage difference. The PMOS and MOS transistors between data and $\overline{\text{data}}$ are enabled when the clock signal goes to 0. These two transistors work as source terminating resistors and bring the data signal into the common-mode voltage. Thus, the data pulse is roughly half of the clock period.

5.4.3 Receiver

At the far end of the transmission line, the data signal becomes a 40ps wide narrow pulse. The magnitude of the differential voltage of the data signal is roughly 0.5V. Capturing such a narrow pulse is the main challenge for designing the receiver for the low-swing design. We added one edge detector before the Strong-Arm latch as shown in Figure 5.26. This edge detector

consists of two inverters and two cross-coupled PMOS transistors. All of the transistors in the edge detector are 1μm wide. As shown in Figure 5.27, when the input voltages are the same, the outputs of the edge detector maintain their states. Assume that in is high and $\overline{\text{in}}$ is low. The transistor, P2, is on to help the PMOS transistor in inverter I1 to maintain the high voltage of node in. The transistor, P1, is off. However, the NMOS transistor in inverter I1 is stronger than the PMOS transistor in this inverter and $\overline{\text{in}}$ is driven low. Once a pulse occurs on the inputs of the edge detector, the inverters, I1 and I2 can overwrite the states of the output nodes. The minimum height of a 40ps wide pulse that can trigger a transition on the outputs is 360mV. The edge detector extends this narrow pulse into a wide pulse for a full clock period. We use the variable delay element as shown in Figure 5.19 to adjust the arrival time of the sampling clock for the Strong-Arm latch. When s0 and s1 both are high, the input to the Strong-Arm latch, in, comes roughly 50ps early than the sampling clock at 4GHz. The setup and hold times of this latch are satisfied.

In Figures 5.22 and 5.23, the data lines are biased at 0.5V, half of V_{DD}. This bias voltage affects the operation of the receiver. Figure 5.28 plots the waveforms for in and $\overline{\text{in}}$ with the bias voltage equal to 0.6V, 0.5V and 0.3V. The top figure in Figure 5.28 is the differential input to the edge detector. This differential input is a 40ps wide, 0.4V high pulse. The receiver including the edge detector and Strong-Arm latch works when the bias voltage is from 0.38V to 0.53V in HSPICE five corner simulations. When the bias voltage is 0.3V, in the common mode, the voltages of outputs of the edge detector is 0.99V. Both P1 and P2 are off; the outputs of the edge detector are determined by inverters I1 and I2 and the edge detector is in a stable mode. The input pulse cannot push the outputs to diverge enough and the outputs of the edge detector goes back to 0.99V, the common mode voltage, very quickly which causes the failure of the receiver.

When the bias voltage is 0.6V, after the data pulse, the outputs of the edge detector stays at 0.009V and 0.69V respectively. However, the setup and hold times of the Strong-Arm latch is a function of the input voltage. Compared with the case that the bias voltage is 0.5V where the outputs of the edge detector are 0.075V and 0.877V, the reduced voltage difference on in and $\overline{\text{in}}$ prolonged the setup time of the Strong-Arm latch such that the setup time of the Strong-Arm latch is violated. For this kind of failure, we can adjust the delay of the variable delay element in the receiver to adjust the arrival time of the sampling clock to make the receiver work.

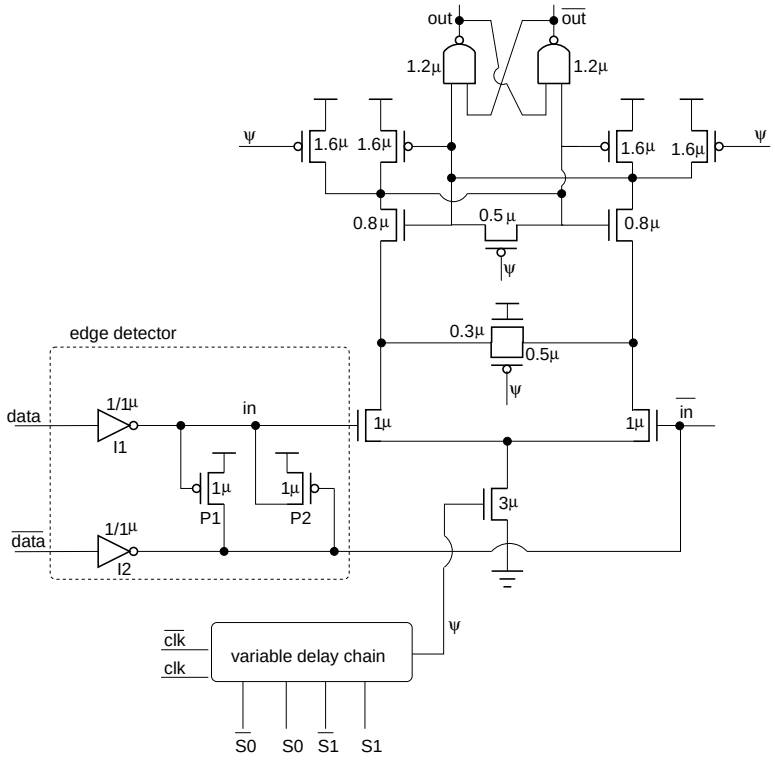


Figure 5.26: Receiver for low-swing design.

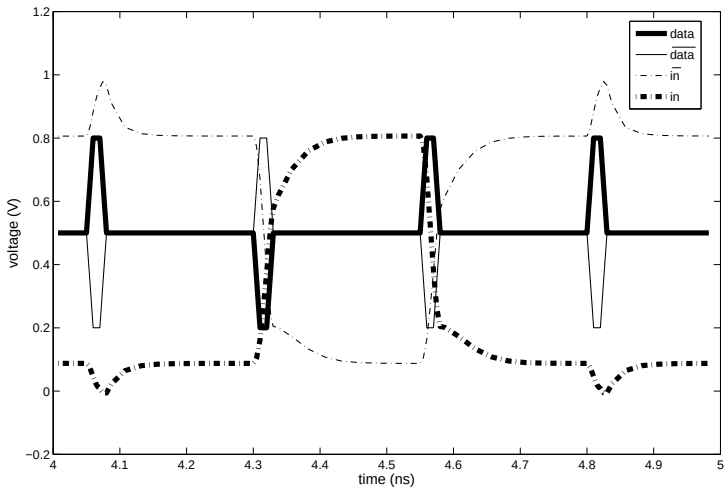


Figure 5.27: Simulation for the edge detector.

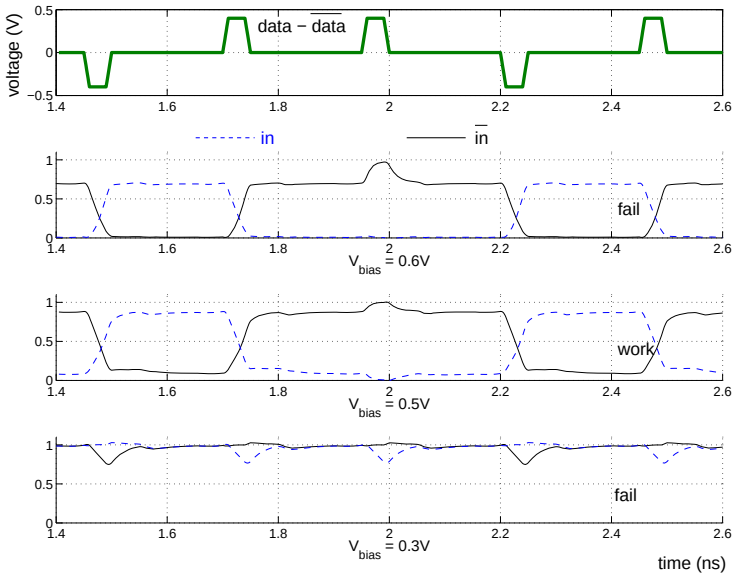


Figure 5.28: Failure modes for the edge detector.

5.4.4 Simulation Results

The two groups work correctly in HSPICE five corner simulations. Table 5.7 lists the power consumption for the major components in these two groups when the operating frequency for both groups is 4GHz. We disabled all the terminating resistance implemented by the transistors. As in Table 5.5, the power consumption for the bridges in Table 5.7 assumes inputs reach full power rail and their short circuit power consumption is included in the clock line.

The clock line in LS-12 consumes less energy than LS-8. The total line capacitance for both groups are nearly the same. The input signal in LS-12 attenuates less than that in LS-8 due to the smaller line resistance. Thus the inverters in LS-12 provides less energy into the transmission line. The transmitter for the data line consumes much less power than the transmitter for the clock line because the data signals are low-swing. The receivers for LS-12 and LS-8 consume a little bit more energy than FS-12 because the edge detector in front of the Strong-Arm latch consumes short-circuit power.

The main advantage of the low-swing design is that the capacitive load on the data lines is much smaller than that in the full-swing design. Thus, the low-swing design consumes less energy than the full-swing design. The total power of the data path is about 2pJ per bit. Because the capacitance of the clock lines is slightly lower than for the full-swing design, clock edges propagate 6% faster in the low-swing and consume less energy. However, the clock path remains full swing and still consumes a large amount of power. As in the full-swing design, we did not seek to minimize power consumption for this first, proof-of-concept test chip. We expect that by reducing the size of the transmitter and finding other improvements, we will be able to further reduce the power consumption in future research and thereby make the low-swing design even more attractive.

5.5 Test Results

To test the chip, we initially performed some simple experiments to make sure that the power consumption was close to the expected value (to avoid damaging the chips) and ensure that clock signals could be propagated across the long wires. As described in Section 5.5.1, these tests failed due to short circuits caused by unexpected metalization in the passivation cuts. Once these shorts were corrected by FIBbing, we had a working chip. Section 5.5.2 presents the test results for the full-swing design, and Section 5.5.3 presents the results for the low-swing lines.

Table 5.7: Power consumption of major components in LS-12 and LS-8 (Simulated).

	LS-12	LS-8
line length (mm)	16	16.4
clock line (pJ/mm/bit)	0.24	0.30
data line (pJ/mm/bit)	0.00	0.00
bridge circuit (pJ/mm/bit)	0.98	0.98
total clock+data+bridge (pJ/bit)	19.5	21.0
transmitter for data line (pJ/bit)	0.7	0.6
transmitter for clock line (pJ/bit)	3.7	3.5
receiver (pJ/bit)	1.2	1.2
total power (pJ/bit)	25.1	26.4

We used a Picoprobe model 35 probe to observe the signals in the passivation openings, an Agilent E8403 BERT to generate the clock and data signals and an Agilent Infiniium DSA80000B oscilloscope to display the outputs.

5.5.1 Initial Tests

We placed two passivation openings along the transmission lines as indicated in Figure 5.1. The center line distance from the transmitter to the first opening for FS-8, FS-12, LS-12, LS-8 are 3.6mm, 3.8mm, 4.0mm, 4.2mm respectively. The center line distance from the first opening to the second is 8mm for every group.

TSMC deposited a thin copper layer [3] in the entire area of the passivation openings such that the 16 wires for the four pairs of clock/data transmission lines were all shorted together. Without any AC input, we observed oscillations from the chip at roughly 5GHz; the power consumption was much higher than predicted by pre-fabrication HSPICE simulations; and none of the four experiments could successfully transfer data across the chip. Once we determined the cause of these behaviours, the SUN test lab at Sunnyvale helped to fib-edit the bare dies and removed the thin copper layer between the transmission lines. We also sent two boards to EAG [61] to get the shorts removed from dies that were already bonded to speed up the testing.

With the corrected chip, the oscillations stopped, the chip's power supply

current dropped to a value that was much closer to that predicted by pre-fabrication simulations, and we observed end-to-end transmission of data. The power supply current was still higher than what we had expected from pre-fabrication HSPICE simulations. We performed a set of measurements described below and determined that the extra current is consumed mainly by the I/O pads that were supplied by FORZA [62]. These pads had not been included in the pre-fabrication simulations. The chip uses several external voltage and current sources: the 1V power supply is used throughout the chip; the 1.8V supply and the current source are used only by the FORZA pads; and the 0.5V supply is used for terminating the low-swing lines. By selectively enabling or disabling the different power supplies, We performed the following experiments to determine where the power supply current was being consumed:

Measure the power consumption when the 1V supply was provided but the 1.8V supply was set to 0V. This disables the I/O pads and allowed me to measure the DC current of the surfing circuitry. This current is dominated by current flowing through the termination resistors for the lines.

Measure the power consumption when both the 1V and 1.8V supplies were provided but the input signals were held at fixed values. This measures the total static power consumption of the chip.

Measure the power consumption under normal operation. This measures the total static and dynamic power consumption of the chip.

When the 1V power supply is on and the 1.8V supply is turned off, the chip consumes between 47mA and 63mA depending on the selection of on-chip terminating resistors for the transmission lines. With the terminating resistors set to their largest values (thus, minimum current consumption, 47mA), we turned on the 1.8V and the off-chip current source. The current pulled from the 1V supply increased to 187mA. Thus, the FORZA pads consume 140mA DC. we then performed HSPICE simulations that show that each differential output pad consumes 0.3mA more dynamic current when its input is driven with a 2GHz square wave than under static operation. The differential input pads consume roughly 2.7mA more dynamic current when driven by 4GHz square waves. There are three differential input pads and two differential output pads on the chip. Thus, with the clock signal running at 4GHz and using a 2GHz square wave for the data signal, the dynamic current consumption of the FORZA pads should be roughly 5mA

(assuming that the deskew block is not used). We set the data line to be DC signal and ran the clock at 4GHz to get an estimation of the power consumption of the clock path plus bridge circuits. This clock path consists of the transmitter, the differential lines with cross-coupled inverters, circuits driven by the forwarded clock in the receiver and circuits from the end of the clock line to the inputs of FORZA clock output pads. By setting all of the varactors along the clock line to their minimum capacitance, the clock paths and bridge circuits in FS-8 and FS-12 consume roughly 102mA and 106mA. When setting all of the varactors to their maximum capacitance, the clock paths in FS-8 and FS-12 consume 5mA and 4mA more current at 4GHz. We set the clock line to be DC signal and ran the data line at 2GHz to get an estimation of the power consumption of the data path for the full-swing design. The data path consists of the transmitter, differential lines, cross-coupled inverters for full-swing design along differential lines, circuits driven by the data signal in the receiver. The dynamic current consumption of the data path in FS-8 varies in the range of 26.5mA to 36.5mA by changing the capacitances of the clock-controlled varactors from their minimum values to their maximum. With the corresponding change of varactor capacitance, the data path in FS-12 consumes roughly 29.5mA and 37.5mA respectively. When driving the clock signal with a 4GHz square wave, the 8μm and 12μm low-swing designs each consume about 8.5mA more current when the data signal is driven by a 2GHz square wave than when it is held to a constant value. The data path in the low-swing design consumes roughly one third of the current by the data lines in the full-swing design because of the absence of the cross-coupled inverters.

To determine the power consumption of the clock path separate from that of the bridge circuits, I used the fact that the clock path and full-swing data path are very similar. They use transmitters and cross-coupled inverters of the same size. There are also some differences that are relatively minor for the purpose of estimating power consumption. The clock line drives the bridge circuits along the line; however, the capacitance contributed by the bridge circuits to the clock line is a small percentage of the total clock line capacitance. The data line and clock line have different termination resistances at the receiver's end. The data line drives the data input of the Strong-Arm latch whereas the clock line drives the delay element in the receiver. Taking these differences into account, we believe that transmitters and cross-coupled inverters are the major power consumers in the clock lines, and they are the same as the circuits used for the data line. Thus, we drove the data line with 2GHz and 4GHz square waves to estimate the power consumption of the clock path. For this test, we set the clock signal to be fixed

Table 5.8: Propagation delay for the full-swing design (measured).

	FS-8 line delay (ps/mm)	FS-12 line delay (ps/mm)
$d_{c,min}$	12.8	12.3
$d_{c,max}$	14.6	14.4
$d_{d,min}$	11.8	11.2
$d_{d,max}$	13.6	12.6

at its high level such that the varactors for the data line were set to their minimum capacitance and the bridge circuits only consumed static power. We use design FS-8 as our example to estimate the power consumption of the bridge circuits. With a 4GHz square wave input to the data lines, the chip consumed 17mA more than with a 2GHz square-wave input. Adding this to the 26.5mA (minimum capacitance in FS-8) yields a minimum current consumption for the clock path of 43.5mA. If the varactors are set to their maximum values, this should increase by about 5mA to 48.5mA. As noted earlier, the total current consumption of the clock path and bridge in FS-8 circuits is about 102mA; thus, the bridge circuit in FS-8 consumes roughly 58.5mA at 4GHz if all of the scan-chain controlled varactors are set to their minimum values. Running at 4GHz, the bridge circuit consumes roughly 15pJ/bit which is close to the number predicted by simulations as shown in Table 5.5 and 5.7. The power consumption of the bridge circuits is comparable to that of the clock line and transmitter.

Operating at 4Gbps, design FS-8 and FS-12 consume roughly 33pJ/bit and 35pJ/bit. At 4Gbps, LS-8 and LS-12 consumes roughly 30pJ/bit. For all four designs, the measured power consumption is higher than the power consumption predicted by HSPICE simulations as shown in Table 5.5 and 5.7. This is because in HSPICE simulations, we did not include the interfacing circuits from the transmitters and receivers to the FORZA pads and the whole interface block as shown in Figure 5.3.

5.5.2 Full-swing Transmission Line

For both FS-8 and FS-12, we determined the minimum power supply voltage for which the chip would operate correctly with a 4GHz square wave for the clock signal and a 2 GHz square wave for the data. The FS-8 design operates correctly for power supply voltages down to 0.83V, and the FS-12 design operates down to 0.94V.

We probed the arrival time of the data and clock signals at the two

Table 5.9: Jitter in the clock and control signals (simulated).

distance to the transmitter	delay per 2mm of clock line	delay per 2mm of control signal
3mm	25.0	25.6
5mm	25.3	26.2
7mm	25.6	26.4
9mm	25.8	26.0
11mm	25.6	24.0
13mm	24.9	21.9
15mm	23.3	14.1

passivation openings to measure the propagation delay of the transmission lines. Because the active probes had fairly wide bodies and a very short tip, We could only probe one signal at a time. We set the clock line to be DC high or low to measure the minimum and maximum propagation delay of the data line. We set the clock line varactors to their maximum capacitances to measure the maximum propagation delay of the clock line and to their minimum capacitances to measure the minimum delay. Table 5.8 summarizes the propagation delay for the two groups. The measured delay variations of the clock lines and data lines are within 40% of the predicted simulation values. Given the measurement accuracy within 2ps, we see that 40% inconsistency is reasonable. We note that the delay range of the data line in FS-12 is smaller than the range predicted by HSPICE simulation as shown in Table 5.3. For both groups, the delay interval of the clock line overlap with the delay interval of the data line. When all of the varactors on the clock line are set to their maximum value, the propagation delay of the clock line is greater than that of the data line and surfing cannot occur. When all of the varactors on the clock line are set to their minimum value, the surfing requirement is satisfied. Thus, I used the minimum setting of the clock line varactors in the remaining full-swing experiments.

Figure 5.29 shows the arrival time of the data signal at the second passivation opening when we changed the arrival time of the clock signal and fixed the arrival time of the data signal at the output of the BERT for design FS-8. The maximum delay difference is roughly 20ps which matches the number given in Table 5.8. However, with roughly 250ps phase shift of the clock signal, the arrival time of the data signal only changes 20ps. Thus surfing is only eliminating about 8% of the timing variation which is not enough to strongly "lock" the data events to the rising edge of the clock. We would prefer to see a curve with a slope close to -1 which would indicate

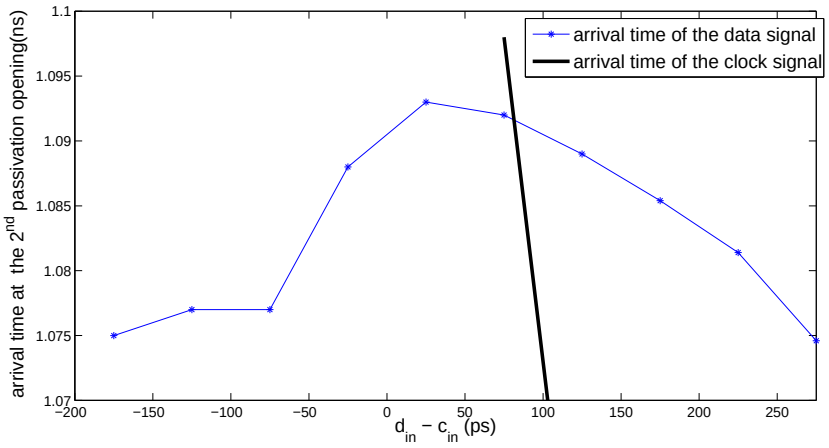


Figure 5.29: Arrival time of the signals at the 2^{nd} passivation opening of FS-8 (measured).

that data events are pulled close to the clock edge. As shown in Figure 5.30, the surfing effect is even smaller for design FS-12. We see a small surfing effect, but not enough to be useful in practice.

Based on these measurements, I reexamined the pre-fabrication simulations (see Figure 4.29). These also showed that the impact of surfing in the full-swing design is small. Furthermore, these simulations show that the rising time of the signal that drives the varactor is roughly 40ps which means that we cannot expect a strong surfing effect for timing variations on the order of 10ps. Both measurements and simulations indicate that the maximum delay variation is roughly 2ps per repeater. The fabricated lines are roughly 15mm; thus, the maximum jitter that the transmission line can correct is ±15ps. This is much smaller than the jitter tolerance of the Strong-Arm latch in the receiver; so, the benefit of surfing for the full-swing design is negligible.

While the test measurements do not show compelling surfing for the full-swing line, the measurements are consistent with the results from HSPICE simulations. This validates our line models, and motivated performing some additional simulations. The full-swing transmission lines do not behave like infinitely long transmission lines because neither the transmitter circuitry nor the receiver with its terminating resistors provides ideal termination. Furthermore, with an active line, it is not clear exactly what an ideal ter-

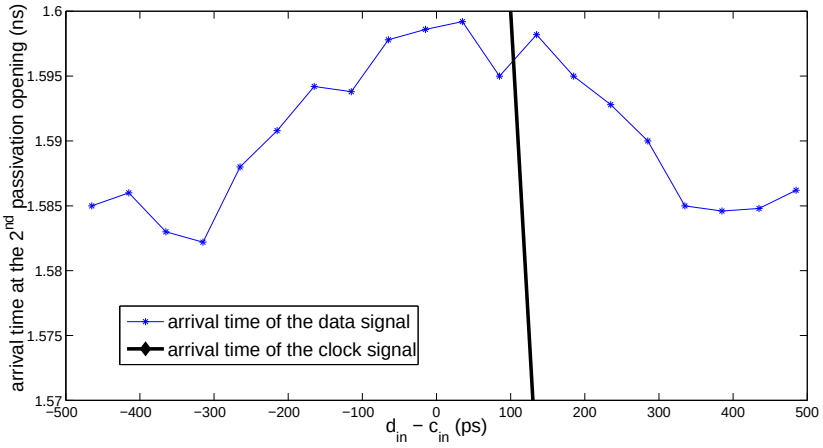


Figure 5.30: Arrival time of the signals at the 2^{nd} passivation opening of FS-12 (measured).

mination would be. Motivated by the testing measurement, we performed additional simulations to better understand the propagation of the clock on the transmission line. HSPICE simulations showed that the clock signal changes shape and propagation velocity along the line. After leaving the transmitter, the maximum slope of the transitions become larger, but the transition to the more gentle slope happens closer to $V_{DD}/2$. This shows that the cross-coupled pairs are sharpening the transition in the region where their transconductance is large. The maximum and minimum values of the waveform remain roughly the same. Near the end of the line, the transitions show a small amount of overshoot. This makes the initial edge both sharper and taller, and then the waveform droops slightly for the rest of each half-period. This overshoot is caused by the reflected clock edge for each transition. The bridge circuit forwarding the clock signal to the data line to control the varactors further amplifies variations in the clock signal timing. We simulated the 8μm wide, 15.2mm long transmission line in HSPICE using the non-degraded parameters for line inductance and capacitance. Table 5.9 lists the propagation delay of the clock signals and the control signals to drive the varactors. For the row labeled d mm, the element in the clock-line column gives the time from the arrival of a clock edge at the $(d-2)$ mm point to the d mm point on the line. Likewise, the element in the control-signal column gives the time from a rising edge of the output

of the bridge circuit at the $(d-2)$ mm point to the output of the bridge at the d mm point along the line. At the 11mm point which, is close to the 2^{nd} passivation opening, the clock signal arrives 0.2ps earlier than the nominal point, but the reflected clock signal causes the output of the bridge circuit to arrive 2ps earlier. The closer a bridge circuit is to the end of the transmission line, the earlier the control signal is output by the bridge. Thus, even if the data signal was locked to the clock signal before the 11mm point, at the 13mm point, the data signal would propagate faster than the clock signal because the control signals for the varactors arrive much earlier than expected and surfing effects on the data line are too weak to track these variations in the clock and control timing.

5.5.3 Low-swing Transmission Line

Designs LS-12 and LS-8 operate with a 4GHz clock for power supply voltages down to 0.86V. Likewise, both designs work for bias voltages for the data lines terminating between 0.3V and 0.7V with clock lines and data lines operating at 4GHz and 2GHz respectively. The data lines in the low-swing design include varactors controlled by the clock signal and varactors that are statically set by the scan-chain. The scan-chain controlled varactors were included to allow the nominal propagation rate of the data lines to be adjusted to roughly correspond to that of the clock lines. By holding the clock signal high, pulses along the data line propagate at the maximum possible velocity. The minimum propagation delays of the data line in LS-8 and LS-12 are both 7.9ps/mm with all of the scan-chain controlled varactors set to their minimum values. In LS-8 and LS-12, the propagation delay of the data lines increases to 8.9ps/mm and 8.7ps/mm respectively by setting all of the scan-chain controlled varactors to their maximum capacitance. The delay difference induced by the scan-chain controlled varactors are roughly 1ps/mm in LS-8. Thus, at the second passivation opening, we saw a pulse arriving later by roughly 13ps in LS-8 by setting all of the scan-chain controlled varactors to their maximum value. Changing the scan-chain controlled varactors from their minimum value to their maximum value had no visible effect on the magnitude of the data signal. In the following tests, we set all of the scan-line controlled varactors to their minimum capacitance values unless otherwise noted.

Unlike the full-swing design, the clock lines in the low-swing design do not include any varactors for static delay adjustments. The propagation delay of the clock line is 12ps/mm for design LS-8 and 11ps/mm for LS-12. Figure 5.31 shows the waveform of the data signal when the clock signal runs

at 3GHz. In Figure 5.32 is the waveform of the data signal when the clock signal is set to be DC high and the highest frequency of the data signal is 1.5GHz. We see that the low-swing transmission line reshapes a wide pulse into a narrow pulse as expected. The peak-to-peak voltage of the data signal with a DC clock is roughly 200mV. With an oscillating clock signal, the peak-to-peak voltage is more than 300mV. The peak-to-peak voltage we observed in HSPICE simulation using degraded line parameters is roughly 500mV at the 2^{nd} passivation opening. The observed peak-to-peak voltage with surfing is smaller than what simulations predicted. Simulations suggest that the pulse-widths are close to the bandwidth limit of the oscilloscope. Thus, we believe that the discrepancy is due to high-frequency losses in the probing set-up and oscilloscope. Although the data signal does not show a clear eye-opening, this does not indicate a failure of the design. The edge-detector circuit in the receiver (see Figure 5.26) functions as an integrating receiver, and it is the output of the edge-detector that determines the correct operation of the link. We report bit-error-rates shortly, and they show that this design is quite robust.

As described above, the arrival time of the data pulse changes by 13ps when the scan-chain controlled varactors are changed from their minimum to their maximum setting when the clock is held at a constant high value. When the clock is a square wave, surfing should attenuate this timing variation. To test this, we measured the arrival time of the peak of the data pulse at the second passivation cut measured from the rising edge of the clock output by the BERT. We made sixteen such measurements with all scan-line varactors set to their maximum capacitance, and sixteen with these varactors set to their minimum capacitance. Table 5.10 shows the distribution of timing shifts where $T_{\max,C}$ and $T_{\min,C}$ are the arrival time of the peak of the data when all of the scan-chain controlled varactors are set to their maximum and minimum capacitances. The mean and deviation of the differences for sixteen pairs are roughly -0.85ps and 1.66ps. This indicates that surfing removed $(1 - (0.85/13)) * 100\% = 93.4\%$ of the delay variation.

The clock signals in the low-swing design experience the same end-of-line effects as in the full-swing design. However, the data signal is much more robust to the jitter of the bridge circuit. The critical design difference between the low-swing and full-swing designs is that varactors for the low-swing line effect a much wider range of delay variation than those in the full-swing line. This is because the low-swing data lines do not include the capacitive shunts of the cross-coupled inverters. Thus, the varactors account for a much larger fraction of the total capacitance of the low-swing lines than they do for the full-swing lines. The wider delay range for surfing with the

Table 5.10: Distribution of the arrival time of the peak of the data signal.

$T_{\max,C} - T_{\min,C}$	# examples
< -2ps	2
$[-2\text{ps}, -1\text{ps})$	3
$[-1\text{ps}, 0\text{ps})$	6
$[0\text{ps}, 1\text{ps})$	3
$[1\text{ps}, 2\text{ps})$	1
> 2ps	1

low-swing design allows the data path to readily track the delay variations of the clock line and bridge circuits.

We sent the differential data signals to the Agilent BERT, ran the low-swing design at different frequencies for 10^{13} events, and counted the number of errors in this duration. In the following experiment, we set all of the scan-chain controlled varactors to their minimum capacitances. Table 5.11 shows the bit-error-rates(BER) for design LS-8 operating as several different frequencies. Each trial consisted of 10^{13} events. Operating with a 3GHz clock, we observed no errors. We enabled the deskew block and operated the clock line at 3GHz where the deskew block can work safely. Again we observed no errors in 10^{13} events. This clearly demonstrates that this simple deskew block does not sacrifice the performance of the transmission line.

Starting from 4.2GHz, we started to see errors in the received data streams. After 4.3GHz, the BER rises dramatically and at 4.4GHz, the BER is so high that the BERT's PLL fails to synchronize to the data stream. We also ran design LS-12 at 4GHz and we did not observe any errors in 10^{12} events. However, when running for 10^{13} events, we observed over 2000 errors. These results were repeatable, both for the 10^{12} and the 10^{13} event trials. This indicates that the errors are not independent events. We do not know if this is because of the BERT losing synchronization with the data stream or the chip parameters drifting over time (perhaps due to warming), or some other effect. At this point, we note this anomaly in the error rate but do not have an explanation.

5.6 Comparison with Other Techniques

This section compares the low-swing design with other previously published techniques [2, 15, 29, 30, 68]. These papers have been discussed in Chapter 2.

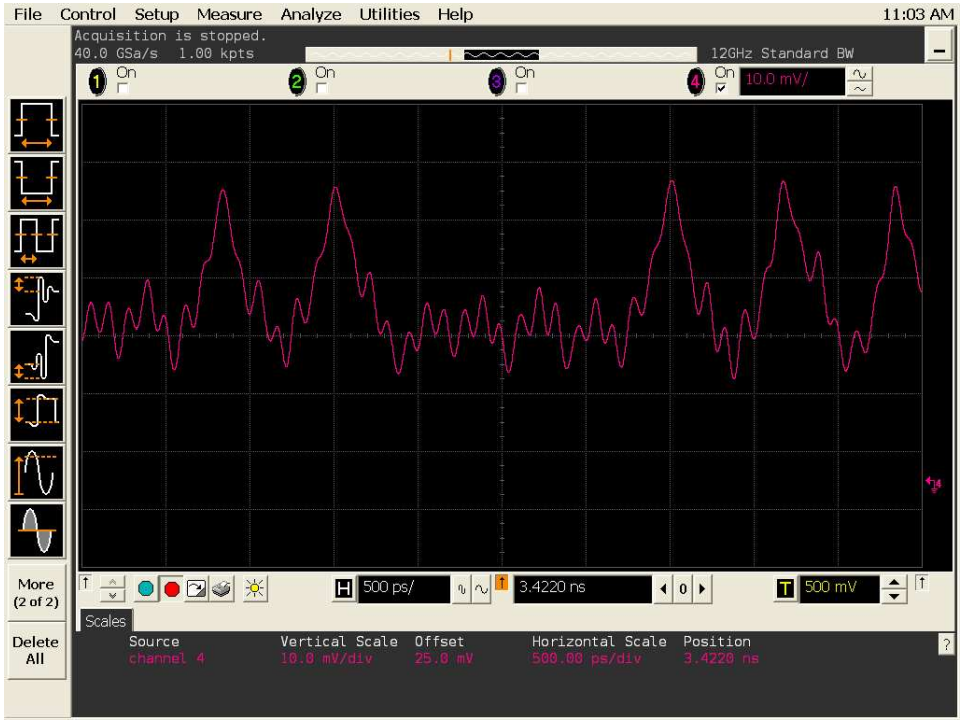


Figure 5.31: Waveform of the data signal (measured).

Table 5.11: BERs at different data rates for LS-8 (measured, 10^{13} events).

frequency (GHz)	BER
2.0	$< 10^{-13}$
3.0	$< 10^{-13}$
4.0	$< 10^{-13}$
4.1	$< 10^{-13}$
4.2	$\leq 3 \times 10^{-13}$
4.3	$\leq 2.2 \times 10^{-9}$

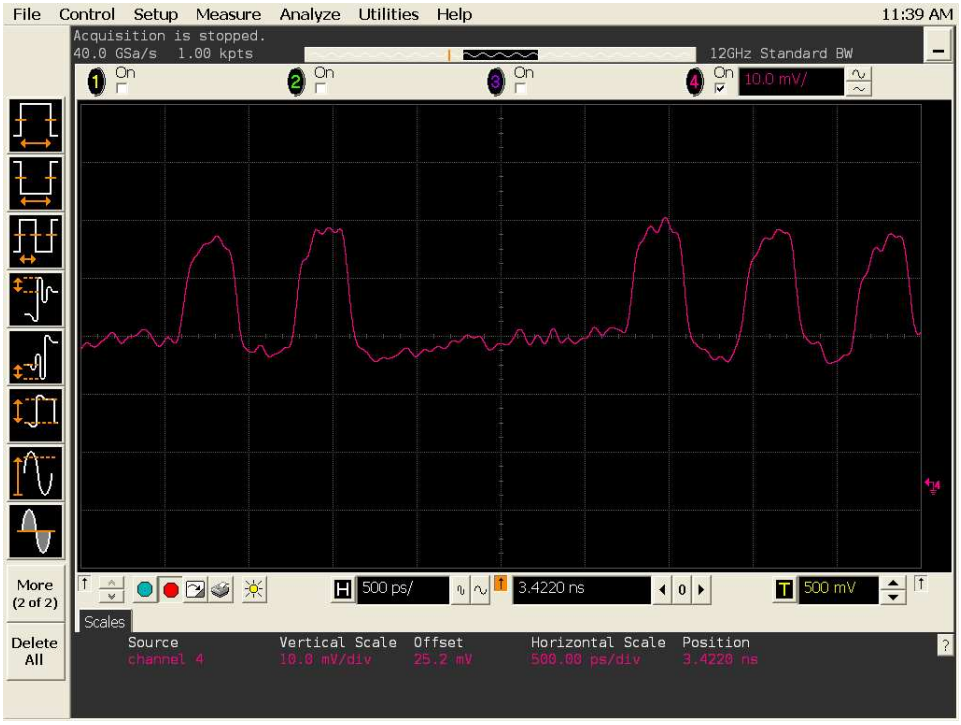


Figure 5.32: Waveform of the data signal by setting clock to be DC high (measured).

Here, I provide a comparison with our surfing designs based on the measured data reported in the previous section. The most directly related work is [2] which also uses varactors for pulse shaping and [29, 30] which presents a complete LC-mode line. We also noted that the surfliner technique from [68] and the Chang *et al.*'s method [15] based on transmitting data on a high-frequency carrier also achieve LC-mode latencies and we briefly compare our work with these two methods as well.

The nonlinear transmission line in [2] also uses varactors to reshape the pulse and concentrate the energy into a narrow pulse. This reshaping effect is achieved by biasing the varactors to a local maximum for their capacitance. Thus this design is sensitive to finding the correct bias voltage. Our design uses clock line to control the nonlinear behaviour of the varactors and can therefore work with a wide range of bias voltages. The authors in [2] use non-uniformly distributed varactors to concentrate all the energy into a single pulse. They demonstrate their ideas with a 3mm long wire without transmitters and receivers on the chip; pulses were injected directly onto the transmission lines from probes, and the solitons at the end of the lines were likewise observed by probing the chip. Our design uses uniformly distributed varactors; HSPICE simulation and testing show that all the energy is concentrated into a single pulse.

In 2006, Jose and others built a passive transmission link [29] and an active compensation link [30] one year later. Table 5.12 summarized the two techniques(passive link and active link) and the surfing low-swing transmission line(LS-8). The propagation velocity of the passive transmission line is twice that of the active transmission line and the surfing low-swing transmission line. The passive transmission line uses much less power than the active transmission line and the surfing low-swing transmission line. However, the passive link is only 3mm long; thus, repeaters would be needed for longer wires, and these would substantially increase the latency and power consumption of a long link. Compared to the active and passive transmission line, our surfing low-swing transmission line consumes much more power which includes the power consumption of the transmitters, receivers, clock and data paths. The deskew circuits only consumes 0.5pJ/bit at 3Gbps. However, the numbers in the 7^{th} row of this table for the links from [29, 30] only include the power consumption of the drivers and the link. Note that the power consumption of the two data deskew blocks, phase-locked loop and clock skew calibration block on the ends of the serial link is not included in [30]. This could significantly increase the power consumption. The pitch of our transmission line is 25μm. We chose this pitch to make the layout easy and believe that the design should work with smaller pitches as well.

Table 5.12: Comparison of different transmission lines.

row #	item	passive link in [29]	active link in [30]	LS-8
1	technology	0.18μm CMOS	0.18μm CMOS	TSMC 0.09μm
2	throughput(Gbps)	8.0	3.0	4.0
3	link length(mm)	3.0	14.0	16.0
4	link latency(ps/mm)	6.5	12.1	12.0
5	line width (μm)	4.0	8.0	8.0
6	line spacing (μm)	4.0	8.0	17.0
7	power consumption (pJ/bit/mm)	0.1	0.14	1.81
8	power consumption for deskew circuits (pJ/bit)	3.1	not given	0.5 at 3Gbs

The surfliner technique in [68] uses positive conductance shunts in the line such that all frequency components of a signal propagate with the same speed. They showed in simulation that they could send data signal on 10mm long, 2μm wide transmission line. Chang *et al.* [15] modulated a 2Gbps data stream to a 7.5GHz carrier and sent the high frequency signal through a 20mm long, 16μm wide transmission line. The transmission lines in [15, 68] do not have extra capacitive loads from repeaters, varactors or other such circuits. Thus their lines can propagate signals much faster than our surfing low-swing lines. However, the modulator and demodulator used in Chang's method add significant delay which would almost certainly exceed the link delay. Furthermore, these approaches do not address the problem of the deskew circuit at the receiver side which is necessary to allow the data to cross the clock domain safely. In our low-swing design, the data pulse is locked to the clock system, which makes the source-synchronous deskew interface design easy.

5.7 Summary

In this chapter, we presented a physical chip to demonstrate that surfing works in real silicon. The chip has four independent designs. Two of them are full-swing and the other two are low-swing. We choose two kinds of line geometries to study the effect of line parameters.

In all four designs, the signals in the transmission line propagate with delay less than 13 ps/mm which is roughly 5 times faster than the RC delay. This makes these designs attractive. In the full-swing design, the metal fill capacitance, varactor capacitance and capacitance presented by the cross-coupled inverters dominate the total line capacitance. The pure line capacitance is roughly 30% of the total line capacitance. The most critical consequence of this extra capacitance is that it lowers the characteristic impedance of the transmission lines which in turn exacerbates the losses from wire resistance. The skin-effect further increases the resistive losses of the lines, especially at high frequencies. Thus, we had to be careful to ensure that the cross-coupled pairs in our designs would provide enough energy to restore the clock and data signals. We did extensive analysis and simulation to validate our current design. Because of the low line impedance, the drivers for the lines must be very large, and these drivers consume a lot of power. Of course, the added line capacitance also reduces the propagation speed of the surfing interconnect.

To ensure robust surfing, the varactors must be able to change the total

line capacitance by a fairly large amount. We designed our full-swing lines to have a surfing delay variation of ±7.5% which is much more than required to compensate for crosstalk. Achieving this range requires the varactor range to be roughly 30% of the total fixed line capacitance. Thus, the large, total capacitance of the surfing transmission lines creates a need for large varactors as well. However, during the testing, we found that the delay variations induced by the varactors in the full-swing data lines were too small to produce robust surfing. There were three main causes to this outcome. First, the range of varactor induced delay variation was too small to compensate for any large jitter on the data line. Second, the slope of the transition on the bridge circuit output makes the delay transition on the line less sharp than would be ideal. Finally, reflections on the clock line produce jitter that is exacerbated by the bridge circuit to produce a variation that is larger than what the varactors can correct. While these problems surfaced during testing, further HSPICE simulations agree with the testing results confirming the models, but showing that a different design approach is needed. For the full-swing design, it's not clear how to make the varactors large enough to achieve effective surfing while keeping the line impedance high enough to achieve robust signal restoration. A new full-swing design is needed to meet these requirements.

In the low-swing design, we removed the cross-coupled inverters from the data line and changed the data signal from full-swing, non-return-to-zero signal to low-swing, return-to-zero signals. With cross-coupled inverters of the same size as in the full-swing design, the clock signal in the low-swing design propagates roughly 6% faster than the clock signal in the full-swing design. By removing the cross-coupled inverters, with the same size varactors, the delay variation of the low-swing design is much larger than the full-swing design. In this design, surfing not only aligns the data signal with the clock signal, but also reshapes the data signal from a wide pulse into a narrow pulse that surfs with the rising edge of the clock signal. Low-swing design will not work for extremely long distance communication because eventually the data pulse will lose amplitude. We do not yet have a good model to estimate how long we can propagate data pulses on a low-swing line. We will investigate this in the future.

The low-swing design clearly demonstrates surfing. Surfing enhances the non-linear behaviour of the varactors. The "peaking" of data transitions by the non-linearities of the varactors that transfer low-frequency energy to higher-frequency components is clearly shown in the waveforms. For both designs, we observed no errors in 10^{12} events at 4GHz. The design includes a deskew block transfers the data signal from the receiver's clock domain to the

consumer's clock domain. This block uses a self-timed FIFO to compensate for the clock skew between the transmitter and receiver. The FIFO has one latch and a latch controller which generates the clock signal for this latch according to the phase relationship between the transmitter and receiver. When the deskew block is enabled, we continued to observe no errors, but this time, the data is delivered to the BERT with a clock phase determined by the consumer's clock and not the delay of the data lines. In previous work, the deskew block is implemented with phase-locked loops or delay-locked loops which require sophisticated analog circuit design, and such circuits tend to be power intensive. However, the FIFO in our deskew block is purely digital resulting in a very simple deskew design. This confirms that our surfing interconnect only requires a very simple deskew circuit design.

From these experiments, I see that there are surfing and circuit effects that I had not previously considered. The chip demonstrates that low-swing surfing is promising. Of course, future work will have to address various issues including power consumption and wiring density. I see that there is still much to learn.

Chapter 6

Conclusions and Future Work

In this thesis, we have demonstrated that surfing techniques can be used to mitigate timing challenges for on-chip, source-synchronous interconnect. We developed new circuits for RC-mode and LC-mode on-chip interconnects. Surfing can bound the timing variation between the data signal and strobe signal which reduces the complexity of the receiver design.

For the RC-mode surfing interconnect, we invented a new DLL structure based on surfing and demonstrated how it can be used with on-chip asynchronous and source synchronous global interconnect. The surfing DLL is based on the surfing buffer proposed in [24]. A traditional delay-locked loop consists of three parts: a variable delay element, a phase detector and a loop filter. However, our surfing delayed-locked loop only has a surfing buffer and a fixed delay element. The surfing buffer combines the functionality of the variable delay element and phase detector. The fixed delay element generates the predicted time for the next event. The surfing buffer uses the predicted time to adjust the delay for the current clock event. Thus the output time of this surfing DLL is the weighted average of the arrival time of the current clock event and the predicted time for the next event. We showed that this averaging avoids the "jitter peaking" problems that are typically associated with traditional DLLs in Figure 3.4. Our surfing DLL is purely digital and simple.

We presented an analytical model for the jitter-transfer response of our surfing DLL based on a linear approximation of the timing. HSPICE simulations confirm that our analytical model is quite accurate. From this, we used a combination of analysis and simulation to show the jitter attenuating properties of the surfing DLL and chains of such DLLs. We showed how multiple chains of surfing DLLs can be connected in a ring structure to generate multiple-evenly spaced clock phases. Furthermore, simulation results showed that the chains of surfing DLLs are more robust to power supply noise than chains buffered with simpe inverters because the surfing DLLs preserve pulses and filter out jitter.

Using HSPICE simulations we demonstrated our surfing timing chain for both source-synchronous and asynchronous communication. The former design is a revision of the source-synchronous design from [24]. Compared with the traditional wave pipelining, our surfing technique is more robust to power supply voltage, parameter and temperature variation. Compared with a two-phase, latched pipeline, our design eliminates the need for distributing a separate, global communication clock and has lower latency at high throughputs by avoiding the need for latches to keep the data signal synchronous to such a global clock. Compared with other asynchronous interconnect methods, surfing allows sliding window protocols to be used to increase throughput. The surfing buffers for the request and acknowledgement signals allow multiple events to be in flight without risk of dropping or duplicating any pulses. The surfing buffers in the data path keep the data aligned with the request signals to provide a robust implementation of a bundled-completion signaling.

We presented the first applications of surfing to LC interconnect. The propagation speed of LC-mode interconnect corresponds to the speed-of-light for the wire as determined by the line inductance and capacitance. To achieve surfing, we need to vary the delay of the line. Unlike RC-mode signaling where the delay can be modulated by varying the strength of the line drivers, the delay of LC-mode signaling is nearly independent of driver strength. We achieved required delay modulation by using varactors to control the line capacitance.

Our surfing LC-mode interconnect uses cross-coupled inverter shunts to compensate for the energy loss due to the line resistance. This design was inspired by the traveling-wave oscillator proposed by Wood *et al.* [60]. The nonlinear behaviour of the conductance and capacitance presented by the cross-coupled inverters required a novel analysis of these lines. We developed a line model based on a heuristically defined "effective conductance" that is chosen so as to inject roughly the same amount of energy into the line during a transition as the actual inverters. We then used this effective conductance model to perform a detailed analysis of the traveling-wave oscillator and derive empirical conditions that ensure pulse height and shape can be maintained for an arbitrarily long line. HSPICE simulations using parameters for the TSMC 90nm process demonstrated that our empirical formulas provide an accurate characterization of the line.

We proposed two different surfing LC-mode interconnects: full-swing and low-swing surfing. In the full-swing design, both the data and clock lines include cross-coupled inverter shunts at regular intervals to maintain full-swing signals. Varactors on the data line modulate the delay of the data

line to keep the data aligned with the forwarded clock.

The low-swing design also uses cross-coupled inverter shunts to maintain a full-swing signal for the clock but omits these inverters from the data lines. This lowers the total capacitance of the lines to reduce power consumption and achieve a larger, surfing-induced delay variation. In addition to keeping data events aligned with the strobe, the varactors in the low swing design also perform a pulse shaping function. The low-swing design uses return-to-zero signaling, and the varactors cause the charge of each pulse to accumulate at the transition of the strobe signal. Pulses on the data line become higher and narrower as they propagate down the line.

We designed a chip to demonstrate that the LC-mode surfing in real silicon. A major concern in the design is the extra capacitance that is added to the lines due to the cross-coupled inverters shunts, varactors, metal-fill, vias and other parasitics. All of these lower the effective impedance of the line which exacerbates the losses from wire resistance and necessitates the use of large drivers to inject clock and data signals into the lines. As a result, in both the full-swing and low-swing designs, the transmitters consume more than 15% of the total energy. The full-swing and low-swing designs were both implemented by using 8μm wires with 17μm spacing and also by using 12μm wires with 13μm spacing; thus the chip has a total of four sets of surfing lines. Each such surfing line was 15-16mm long.

The chip was fabricated using the TSMC 90nm process. Initially, all chips were non-functional because of a thin layer of copper deposited in the passivation oxide cuts that we had made for test purposes. This extra metal shorted out the transmission lines and turned the chip into a 5GHz oscillator. These short circuits were corrected by FIB editing the chip, which produced working parts.

The full-swing lines operate at data rates up to 4Gbps consuming roughly 35pJ/bit. Measurements showed that the bridge circuits that take the strobe signal as input and drive the data line varactors at their outputs are the largest single power consumer on the chip. This is consistent with pre-fabrication HSPICE simulations. We measured a surfing-induced delay variation of the full-swing data line of about 20ps for the 8μm wide data lines and about 17ps for the 12μm wide lines. Over the length of the line, this could eliminate about 10% of a small applied jitter which is not sufficient to make a compelling case for surfing. After making this observations in the lab, I determined that pre-fabrication HSPICE simulations are consistent with these measurements, confirming the line and device models, but suggesting that a different design should have been fabricated. Further simuations revealed that timing variations for the strobe signal caused by

reflections at the receiver's end of the line create more timing variation than the entire range of surfing-controlled timing variation. In conclusion about the full-swing designs: they did not provide a compelling demonstration of surfing. On the other hand, they did validate many of the modeling techniques used in the design process including the use of effective-conductance in the line analysis.

The low-swing lines operate at data rates up to 4.2Gbps consuming roughly 30pJ/bit. The clock line in the low-swing design consumes roughly the same amount of power as in the full-swing design, the data line in the low-swing design consumes only one third the power of its full-swing counterpart. We observed a clear formation of sharp, narrow pulses as intended from the non-linear operation of the varactors. The arrival time of these narrow pulse is locked to the rising edge of the clock signal. In particular, we can vary the capacitance of the low-swing data line to effect a 13ps change in the delay from the transmitter to the passivation opening at the 12mm point on the line when the clock is at a constant high-level (no surfing). Performing the same experiment with a 3GHz clock enabling the surfing action, we measured the arrival time of pulses at the 12mm point on the line which is nearly independent of the line capacitance.

The low-swing line produced observable surfing when the full-swing line did not for several reasons. First of all, the varactor capacitance is a larger fraction of the total capacitance in the low swing line. Second, the transmission gates between the final inverter of the low-swing transmitter and the transmission line ensure that transitions enter the line with a controlled timing relationship relative to the clock. This allowed us to make more precise observations of surfing in the low-swing line than were possible in the full-swing design. Most importantly, the low-swing line has the peak-forming behaviour which is catalyzed by the transition of the clock signal. Thus, the peak *must* form at the clock edge. Of course, if this peak propagated at a different rate than the clock, it would be spread over a wider time interval, and would be neither as narrow or as tall as what we observed. For these reasons, the low-swing line produced a clear demonstration of a surfing LC-mode that we were unable to obtain from the full-swing line. Furthermore, the low-swing design has higher propagation velocity, larger surfing interval and consumes less power consumption than the full-swing design. These make the low-swing design more attractive than the full-swing version.

By providing new circuit design and a physical implementation, we demonstrated that surfing provides an efficient solution for the timing problems of on-chip interconnect. This approach reduces the complexity of the receiver and is robust to noise and other disturbances. We explored the

surfing RC line using surfing DLL for robust source synchronous and asynchronous communication. We proposed the surfing LC line using varactors to adjust the propagation speed and reshape pulses by moving the energy in the low frequency components to high frequency components. Comparing to the surfing RC line, the jitter reduction effect of the surfing LC line were less compelling. Thus, we see that this work shows the potential of using surfing and varactors for global interconnect, but we recognize that further design work will be needed to bring these methods into practical application.

Future Research

Based on our design experience, chip testing, and further simulations, we see many areas for future research:

1. A more accurate delay model is needed for the surfing DLL. During the rising edge of the predict signal, we modeled the delay as a linear function of the separation time between the input event and the predict signal. The second-order effect caused by Miller capacitance forms a positive feedback which further improves the jitter attenuation capability of the surfing DLL. We had not anticipated this positive feedback, and discovered it when examining the different jitter propagation behaviour predicted by our simple linear model and that from HSPICE simulations. We do not yet have a simple mathematical abstraction that accounts for this second-order effect.

2. The surfing inverters we presented can provide a delay variation of roughly $\pm 18\%$ around the nominal delay. A greater range of operation can be obtained by increasing the ratio of the driving strength of the tri-state inverter to that of the simple inverter in the surfing buffer. However, this would come at a cost of increasing in the overall delay and power consumption of each surfing buffer. Another approach would be to incorporate a single traditional DLL onto the chip to set a reference voltage or current for the delay elements of all of the the surfing inverters. This traditional DLL would be used to compensate for the static and low frequency variations such as parameter and temperature variation, changes of operating frequency, and changes in operating voltage and low-frequency power supply noise. The surfing DLLs and surfing buffers provide the compensation for dynamic variations due to high-frequency power supply noise, cross-talk noise, inter-symbol interference and so on. In the future, we will investigate

this approach.

Currently we use an inverter chain to implement the fixed delay element in the surfing DLL. The jitter in the inverter chain will degrade the jitter attenuation capability of the surfing DLL. In the future, we will investigate new designs for the delay element. We believe that our jitter attenuating buffers can be used to address the challenges of long-distance, on-chip communication. We are also looking at possibilities for incorporating low voltage-swing techniques to reduce power consumption and gain greater immunity to V_{DD} variations.

3. We used HSPICE simulations to demonstrate that surfing DLL chain is more robust to power supply noise than the inverter chain. An alternative approach is to use BERs to quantitatively characterize the robustness to power supply noise. Our next step is to implement a surfing DLL chain and inverter chain in real silicon to demonstrate this.

4. The data signal in the full-swing design cannot strongly lock to the clock signal. One reason is that the surfing delay variation interval is too small. Our current design only provides at most 20ps delay variation which is not enough to attenuate the jitter introduced by the bridge circuits. A new design for the full-swing design is needed to increase the surfing delay variation and reduce the jitter in the bridge circuits.

5. The low-swing method cannot be used with arbitrarily long transmission lines. As the pulse propagates down the line, the non-linear effects of the varactors make the pulse taller and narrower. However, energy is still lost due to the resistance of the line. Thus, eventually, the pulse will lose height and fall below the threshold of reliable detection by the receiver. In the future, we hope to develop quantitative models for how the shape and magnitude of the pulse change as a function of the line length.

6. From the testing, we observed that in the full-swing and low-swing design, the 12μm wide wires do not perform better than the 8μm wide wires. Due to metal fill, cross-coupled inverters and other devices, the total capacitances for both lines are about the same. This suggests that we may use smaller pitch for the transmission line to both reduce the energy per bit transmitted and increase the cross-sectional bandwidth of the interconnect.

7. The transmitters in the full-swing and low-swing design are overdesigned, and the bridge circuits consume large amounts of power. We see significant opportunity to reduce the power consumption in these designs.

Bibliography

[1] V. Adler and E. G. Friedman. Repeater design to reduce delay and power in resistive interconnect. *IEEE transactions on Circuit and Systems II: Analog and Digital Signal Processing*, 45(5):606–617, May 1998.

[2] E. Afshari and A. Hajimiri. Nonlinear transmission lines for pulse shaping in silicon. *IEEE Journal of Solid-State Circuits*, 40(3):744–752, March 2005.

[3] Shahid Ansari. private communication, 2009.

[4] Avant! Star-Hspice manual, 1998.

[5] W.J. Bainbridge and S.B. Furber. Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. In *Proceedings of the Seventh International Symposium on Asynchronous Circuits and Systems*, pages 118–126. IEEE, March 2001.

[6] H. B. Bakoglu and J. D. Meindl. Optimal interconnection circuits for VLSI. *IEEE Transactions on Electron Devices*, ED-32:903–909, May 1985.

[7] G. Balamurugan and N. Shanbhag. Modeling and mitigation of jitter in multi-Gbps source-synchronous I/O links. In *21st International Conference on Computer Design*, pages 254–260, 2003.

[8] K. Banerjee and A. Mehrotra. A power-optimal repeater insertion methodology for global interconnects in nanometer designs. *IEEE Transactions on Electron Devices*, 49(11):2001–2007, November 2002.

[9] Keith A. Bowman, Steven G. Duvall, and James 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*, 37(2):183–190, February 2002.

[10] Wayne P. Burleson, Maciej Ciesielski, et al. Wave-pipelining: a tutorial and research survey. *IEEE Transactions on VLSI Systems*, 6(3):464–474, September 1998.

[11] Matthias Bussmann and Ulrich Langmann. Active compensation of interconnect losses for multi-GHz clock distribution networks. *IEEE transactions on Circuit and Systems II: Analog and Digital Signal Processing*, 39(11):790–798, November 1992.

[12] M. Cases, D. N. De Araujo, and E. Matoglu. Electrical design and specification challenges for high speed serial links. In *Proceedings of 7th Conference on Electronic Packaging Technology*, pages 29–33, December 2005.

[13] Ajanta Chakraborty and Mark R. Greenstreet. A minimalist source-synchronous interface. In *Proceedings of the 15th IEEE ASIC/SOC Conference*, pages 443–447, September 2002.

[14] M.F. Chang, V.P. Roychowdhury, L. Zhang, et al. RF/wireless interconnect for inter-and intra- chip communications. *Proceedings of the IEEE*, 89(4):456–466, April 2001.

[15] R.T. Chang, N. Talwalkar, C.P. Yue, and S.S. Wong. Near speed-of-light signaling over on-chip electrical interconnects. *IEEE Journal of Solid-State Circuits*, 38(5):834–838, May 2003.

[16] R.T. Chang, C.P. Yue, and S.S. Wong. Near speed-of-light on-chip electrical interconnectt. In *Symposium on VLSI Circuits Digest of Technical Papers*, pages 13 – 15, June 2002.

[17] Guoqing Chen and E.G. Friedman. Low-power repeaters driving RC and RLC interconnects with delay and bandwidth constraints. *IEEE Transactions on VLSI Systems*, 14(2):161–172, February 2006.

[18] Hongyu Chen, Rui Shi, Chung-Kuan Cheng, and D.M. Harris. Surfliner: a distortionless electrical signaling scheme for speed of light on-chip communications. In *Proceedings of the 2005 International Conference on Computer Design*, pages 497–502, October 2005.

[19] William J. Dally and John W. Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[20] R. Dobkin, R. Ginosar, and A. Kolodny. Fast asynchronous shift register for bit-serial communication. In *Proceedings of the* 12^{th} *Symposium on Asynchronous Circuits and Systems*, pages 117–126, Grenoble, France, 2006.

[21] Scott Fairbanks and Simon Moore. Analog micropipeline rings for high precision timing. In *Proceedings of the Tenth International Symposium on Asynchronous Circuits and Systems*, pages 41–50, April 2004.

[22] Scott Fairbanks and Simon Moore. Self-timed circuitry for global clocking. In *Proceedings of the Eleventh International Symposium on Asynchronous Circuits and Systems*, pages 86–96, March 2005.

[23] Frank O'Mahony, C. Patrick Yue, Mark A. Horowitz, and S. Simon Wong. A 10GHz clock distribution using coupled standing-wave oscillators. *IEEE Journal of Solid-State Circuits*, 38(11):1813–1820, November 2003.

[24] Mark R. Greenstreet and Jihong Ren. Surfing interconnect. In *Proceedings of the Twelfth International Symposium on Asynchronous Circuits and Systems*, pages 98–106, April 2006.

[25] Ron Ho, Jonathan Gainsley, and Robert Drost. Long wires and asynchronous control. In *Proceedings of the Tenth International Symposium on Asynchronous Circuits and Systems*, pages 240–249, April 2004.

[26] Ron Ho, Kenneth W. Mai, and Mark Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, April 2004.

[27] Intel. Intel microprocessor quick reference guide, 2009.

[28] ITRS. International technology roadmap for semiconductors, 2005.

[29] A.P. Jose, G. Patounakis, and K.L. Shepard. Pulse current-mode signalling for nearly speed-of-light intrachip communications. *IEEE Journal of Solid-State Circuits*, 41(4):772–780, April 2006.

[30] A.P. Jose and K.L. Shepard. Distributed loss-compensation techniques for energy-efficient low-latency on-chip communication. *IEEE Journal of Solid-State Circuits*, 42(6):1415–1424, June 2007.

[31] M. Kamon, M. Tsuk, and J. White. FastHenry: A multiple-accerlerated 3-D inductance extraction. *IEEE Transactions on Microwave Theory and Technique*, 42(9):1750–1758, September 1994.

[32] Pawan Kapur, Gaurav Chandra, James P. McVittie, and Krishna C. Saraswat. Technology and reliability constrained future copper interconnects–part II: performance implications. *IEEE Transactions on Electron Devices*, 49(4):598–604, April 2002.

[33] Pawan Kapur, Gaurav Chandra, and Krishna C. Saraswat. Power estimation in global interconnects and its reduction using a novel repeater optimization methodology. In *Proceedings of the 39th ACM/IEEE Design Automation Conference*, pages 461–466, New Orleans, USA, 2002.

[34] Jaeha Kim and Mark A. Horowitz. Adaptive supply serial links with sub-1V operation and per-pin clock recovery. *IEEE Journal of Solid-State Circuits*, 37(11):1403–1413, November 2002.

[35] Kyu-hyoun Kim, P.W. Coteus, D. Dreps, et al. A 2.6mW 370MHz-to-2.5GHz open-loop quadrature clock generator. *IEEE Journal of Solid-State Circuits*, pages 458–459, February 2008.

[36] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, second edition, 2003.

[37] Kyeongho Lee, Sungjoon Kim, Gijung Ahn, and Deog-Kyoon Jeong. A CMOS serial link for fully duplexed data communication. *IEEE Journal of Solid-State Circuits*, 30(4):353–364, April 1995.

[38] M.-J. Edward Lee, William J. Dally, et al. Jitter transfer characteristics of delay-locked loops – theories and design techniques. *IEEE Journal of Solid-State Circuits*, 38(4):614–621, April 2003.

[39] Sang-Hyun Lee, Moon-Sang Hwang, Youngdon Choi, Sungjoon Kim, et al. A 5Gb/s 0.25μm CMOS jitter-tolerant variable-interval oversampling clock/data recovery circuit. *IEEE Journal of Solid-State Circuits*, 37(12):1822–1830, December 2002.

[40] Andrew Lines. Nexus: an asynchronous crossbar interconnect for synchronous system-on-chip designs. In *Proceedings of the 11th Symposium on High Performance Interconnects*, pages 2–9, August 2003.

[41] John G. Maneatis and Mark A. Horowitz. Precise delay generation using coupled oscillators. *IEEE Journal of Solid-State Circuits*, 28(12):1273–1282, December 1993.

[42] Larry McMurchie, Su Kio, et al. Output prediction logic: a high-performance CMOS design technique. In *Proceedings of the 2000 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 247–254. IEEE Computer Society, 2000.

[43] D.A.B. Miller. Rationale and challenges for optical interconnects to electronic chips. In *Proceedings of the IEEE*, volume 88, pages 728–749, 2000.

[44] A. Morgenshtein, I. Cidon, and A. Kolody anf R. Ginosar. Comparative analysis of serial vs parallel links in NoC. In *Proceedings of International Symposium on System-on-Chip*, pages 16–18, November 2004.

[45] A. Nalamalpu and W. Burleson. A practical approach to DSM repeater insertion: satisfying delay constraints while minimizing area and power. In *The 14th ASIC/SOC Conference*, pages 152–156, 2001.

[46] Jihong Ren. Private communication, 2009.

[47] Samie B. Samman. The impact of device parameter variations on the frequency and performance of VLSI chips. In *Proceedings of the 2004 International Conference on Computer Aided Design*, November 2004.

[48] Sean X. Shi and David Z. Pan. Wire sizing with scattering effect for nanoscale interconnection. In *the* 11^{th} *Asia and South Pacific Conference on Design Automation Conference*, pages 504–508, January 2006.

[49] Jens Sparsø. Asynchronous circuit design - a tutorial, 2004.

[50] B. Stine, D. S. Boning, and J. E. Chung. Analysis and decomposition of spatial variation in integrated circuit processes and devices. *IEEE Transactions on Semiconductor Manufacturing*, 10(4):24–41, February 1997.

[51] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989. Turing Award lecture.

[52] Dennis Sylvester and Kurt Keutzer. Getting to the bottom of deep submicron. In *Proceedings of the 1998 International Conference on Computer Aided Design*, pages 203–211, San Jose, USA, 1998.

[53] Paul Teehan, Guy Lemieux, and Mark R. Greenstreet. Estimating reliability and throughput of source-synchronous wave-pipelined interconnect. In *The 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pages 234–243, 2009.

[54] Harry J. M. Veendrick. Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. *IEEE Journal of Solid-State Circuits*, 19(4):468–473, August 1984.

[55] Pingshan Wang, G. Pei, and E.C.-C. Kan. Pulsed wave interconnect. *IEEE Transactions on VLSI Systems*, 12(5):453–463, May 2004.

[56] Anthony J. Winstanley, Aurelien Garivier, and Mark R. Greenstreet. An event spacing experiment. In *Proceedings of the Eigth International Symposium on Asynchronous Circuits and Systems*, pages 42–51, Manchester, UK, April 2002.

[57] Brian D. Winters and Mark R. Greenstreet. A negative-overhead, self-timed pipeline. In *Proceedings of the Eigth International Symposium on Asynchronous Circuits and Systems*, pages 32–41, Manchester, UK, April 2002.

[58] Brian D. Winters and Mark R. Greenstreet. Surfing: A robust form of wave pipelining using self-timed circuit techniques. *Microprocessors and Microsystems*, 27(9):409–419, October 2003.

[59] Shyh-Chyi Wong, Gwo-Yann Lee, and Dye-Jyun Ma. Modelling of interconnect capacitance, delay and crosstalk in VLSI. *IEEE Transactions on Semiconductor Manufacturing*, 13(1):108–111, February 2000.

[60] J. Wood, T.C. Edwards, and S. Lipa. Rotary traveling-wave oscillator arrays: a new clock technology. *IEEE Journal of Solid-State Circuits*, 36(11):1654–1665, November 2001.

[61] www.cea.co.

[62] www.forzasilicon.com.

[63] C. K. K. Yang and Mark A. Horowitz. A 0.8μm CMOS 2.5Gb/s oversampling recerver and transmitter for serial links. *IEEE Journal of Solid-State Circuits*, 31(12):2015–2023, December 1996.

[64] Suwen Yang. Implementation and analysis of surfing pipeline. Master's thesis, University of British Columbia, Vancouver, Canada, August 2003.

[65] Suwen Yang, Brian D. Winters, and Mark R. Greenstreet. Energy efficient surfing. In *Proceedings of the Eleventh International Symposium on Asynchronous Circuits and Systems*, pages 2–11, New York, USA, 2005.

[66] Brian Young. *Digital Signal Integrity: Modeling and Simulation with Interconnects and Packages*. Prentice Hall, 2001.

[67] Crid Yu, Tinaung Maung, C. J. Spanos, et al. Use of short-loop electrical measurements for yield improvement. *IEEE Transactions on Semiconductor Manufacturing*, 8(2):150–159, May 1995.

[68] Haikun Zhu, Rui Shi, Chung-Kuan Cheng, and Hongyu Chen. Approaching speed-of-light distortionless communication for on-chip interconnect. In *the* 12^{th} *Asia and South Pacific Conference on Design Automation Conference*, pages 684–689, January 2007.

[69] Paul S. Zuchowski, Pater A. Habitz, et al. Process and environmental variation impacts on ASIC timing. In *Proceedings of the 2004 International Conference on Computer Aided Design*, November 2004.