A Novel Interleaved and Distributed FIFO

by

Santosh Sood

B.E. University of Delhi, 2001

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

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

November 2005

© Santosh Sood, 2005
Abstract

In deep submicron technologies, the delays of metal lines continue to increase in spite of an increasing number of metal layers and the use of low-k dielectrics. Thus, some form of interconnect pipelining is required in throughput intensive designs. Various approaches have been used for interconnect pipelining, e.g., synchronous, asynchronous, GALS and source-synchronous, and each presents a trade-off between the throughput and latency that can be achieved.

This work provides an evaluation of the synchronous and the source-synchronous methods of interconnect pipelining. Reference designs for various synchronous and source-synchronous signalling methods are presented. The source-synchronous method entails the forwarding of a clock along with data; this forwarded clock suffers from skew due to process, voltage and temperature variations along the forwarded path. A FIFO is used to compensate for the skew between the forwarded clock and the local clock at the receiver end. We present a novel, interleaved and distributed FIFO that implements wave pipelining between the FIFO stages. This FIFO design helps to lower the latency of the source-synchronous interconnect. A metric of comparison called velocity is introduced, and a comparison of the performance of synchronous and source-synchronous signalling is presented on the metrics of throughput, velocity and power.
Contents

Abstract ii

Contents iii

List of Tables vi

List of Figures vii

Acknowledgements x

1 Introduction 1
   1.1 Interconnect Overview ............................................. 1
   1.2 Thesis Contributions ............................................. 5
   1.3 Thesis Outline .................................................. 5

2 Background 7
   2.1 Interconnect Structures ........................................... 7
      2.1.1 Point-to-Point Links ........................................ 8
   2.2 Interconnect Components ......................................... 8
      2.2.1 Wires and Buffers .......................................... 8
      2.2.2 Latches and Flip-Flops .................................... 10
2.3 Other Interconnect Issues ........................................... 12
   2.3.1 Crosstalk ....................................................... 12
   2.3.2 PVT Variations ................................................ 13
2.4 Interconnect Timing Schemes ..................................... 14
   2.4.1 Synchronous Design for Interconnect ...................... 14
   2.4.2 Two-Phase Clocking .......................................... 15
   2.4.3 Multi-Phase Clocking ....................................... 16
   2.4.4 Asynchronous Design for Interconnect .................... 17
   2.4.5 GALS .......................................................... 18
2.5 Source-Synchronous Design ....................................... 19
2.6 FIFOs ............................................................. 21
   2.6.1 FIFO Control ................................................... 21
   2.6.2 Linear FIFO ................................................... 23
   2.6.3 Twin-Control FIFO ........................................... 24

3 Novel Interleaved and Distributed FIFO .......................... 28
   3.1 Introduction ..................................................... 28
   3.2 Linear FIFO ...................................................... 30
      3.2.1 Latency of a Linear FIFO ................................. 31
   3.3 Novel FIFO 1: Buffered Control Wire ......................... 36
   3.4 Novel FIFO 2: Interleaved Control ............................ 38
      3.4.1 Latency of Interleaved and Distributed FIFO .......... 41

4 Comparison of Interconnect Schemes ............................. 45
   4.1 Circuit Components Variations ................................. 45
   4.2 The Velocity Metric ............................................ 49
   4.3 Two-Phase Clocking ............................................. 50
Contents

4.4 Multi-Phase Clocking ........................................... 53
4.5 Source-Synchronous with the Novel FIFO ..................... 56
   4.5.1 Forwarded Clock Path .................................... 57
   4.5.2 FIFO Design .............................................. 63
   4.5.3 FIFO Initialization ....................................... 70
4.6 Comparison ..................................................... 75
   4.6.1 Throughput vs. Velocity ................................ 75
   4.6.2 Power Dissipation ....................................... 77

5 Conclusion ...................................................... 79
   5.1 Future Work .................................................. 81
   5.2 Thesis Contributions ....................................... 82

Bibliography ..................................................... 83
   A-1 Repeater Sizing ............................................ 89
   A-2 Latch Design ............................................... 90
   A-3 Latch Characterization .................................... 92
   A-4 Crosstalk Characterization ................................ 93
   A-5 Placement Sensitivity ..................................... 96
   B-1 Effect of Multiple Control on Latch Delay ............... 98
   C-1 Distributed and Interleaved FIFO ......................... 100
   D-1 Dynamic FIFO Initialization .............................. 102
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Delays of various circuit components at typical process corner</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>Effect of PVT variations</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Skew, jitter and crosstalk budget</td>
<td>48</td>
</tr>
<tr>
<td>4.4</td>
<td>Worst-case delays of various circuit components</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Source-synchronous tracking margins</td>
<td>49</td>
</tr>
<tr>
<td>4.6</td>
<td>Two-phase clocking</td>
<td>53</td>
</tr>
<tr>
<td>4.7</td>
<td>Multi-phase clocking</td>
<td>56</td>
</tr>
<tr>
<td>4.8</td>
<td>Forwarded path: source-synchronous tracking margin 20%</td>
<td>62</td>
</tr>
<tr>
<td>4.9</td>
<td>Forwarded path: source-synchronous tracking margin 40%</td>
<td>62</td>
</tr>
<tr>
<td>4.10</td>
<td>Characteristics forwarded path</td>
<td>66</td>
</tr>
<tr>
<td>4.11</td>
<td>Result table for design in Table 4.10 (P=442 ps)</td>
<td>70</td>
</tr>
<tr>
<td>4.12</td>
<td>FIFO design: 20% tracking margin</td>
<td>74</td>
</tr>
<tr>
<td>4.13</td>
<td>FIFO design: 40% tracking margin</td>
<td>74</td>
</tr>
<tr>
<td>4.14</td>
<td>Power: minimum pitch and minimum spacing</td>
<td>78</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Buffer insertion helps with latency ........................................... 3  
1.2 Latching wire helps with throughput ......................................... 3  
1.3 Source-synchronous gives high throughput with low design overhead ... 4  
2.1 Interconnect RC model for long wire ......................................... 9  
2.2 Interconnect buffering ................................................................ 9  
2.3 Interconnect pipelining ............................................................... 10  
2.4 Transmission-gate latch ............................................................. 12  
2.5 Two-phase clocking ................................................................. 15  
2.6 Multiple-phase clocking ............................................................ 17  
2.7 Asynchronous protocol ................................................................ 18  
2.8 Source-synchronous forwarded clock and data path ..................... 20  
2.9 FIFO control circuit ................................................................. 22  
2.10 Control circuit: two GasP stages driving single control wire ........ 23  
2.11 Linear FIFO ............................................................................ 24  
2.12 Source-synchronous with lumped FIFO ..................................... 24  
2.13 Source-synchronous with distributed FIFO ................................. 25  
2.14 Twin control with shared handshake wire ................................... 26  
3.1 Source-synchronous with FIFO .................................................. 29
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>Timing: source-synchronous with FIFO</td>
<td>30</td>
</tr>
<tr>
<td>3.3</td>
<td>Linear FIFO</td>
<td>31</td>
</tr>
<tr>
<td>3.4</td>
<td>Latency of the FIFO</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>Twin control with buffered handshake wire</td>
<td>37</td>
</tr>
<tr>
<td>3.6</td>
<td>Single stage of top control path</td>
<td>38</td>
</tr>
<tr>
<td>3.7</td>
<td>Multiple control with buffered handshake wire</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>A pipelined link</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>Two-phase synchronous interconnect</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Propagation constraint two-phase synchronous interconnect</td>
<td>51</td>
</tr>
<tr>
<td>4.4</td>
<td>Multi-phase clocking</td>
<td>54</td>
</tr>
<tr>
<td>4.5</td>
<td>Constraints multi-phase interconnect</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>Analysis flow for designing the forwarded pipeline stages</td>
<td>58</td>
</tr>
<tr>
<td>4.7</td>
<td>Forwarded clock derived from global clock</td>
<td>59</td>
</tr>
<tr>
<td>4.8</td>
<td>Source-synchronous: forwarded clock and data path</td>
<td>59</td>
</tr>
<tr>
<td>4.9</td>
<td>Edge detector generate pulses on both clock edges</td>
<td>60</td>
</tr>
<tr>
<td>4.10</td>
<td>Flow for deciding the optimal FIFO design</td>
<td>64</td>
</tr>
<tr>
<td>4.11</td>
<td>Velocity of the FIFO as a function of $k$ and $n_{buf,fifo}$ ($P = 442ps$)</td>
<td>67</td>
</tr>
<tr>
<td>4.12</td>
<td>FIFO forward delay vs. $k$</td>
<td>68</td>
</tr>
<tr>
<td>4.13</td>
<td>Procedure for initializing the FIFO</td>
<td>71</td>
</tr>
<tr>
<td>4.14</td>
<td>Clock distribution network</td>
<td>72</td>
</tr>
<tr>
<td>4.15</td>
<td>Buffer with bundled reset</td>
<td>72</td>
</tr>
<tr>
<td>4.16</td>
<td>Throughput vs. velocity: synchronous and source-synchronous</td>
<td>75</td>
</tr>
<tr>
<td>A-1</td>
<td>Repeater sizing with crosstalk</td>
<td>89</td>
</tr>
<tr>
<td>A-2</td>
<td>Latch topology</td>
<td>91</td>
</tr>
<tr>
<td>A-3</td>
<td>Elmore delay model for latch</td>
<td>91</td>
</tr>
</tbody>
</table>
List of Figures

A-4 Set-up time and set-up time optimization ........................................ 92
A-5 Hold time characteristics of latch ..................................................... 93
A-6 16-bit bus in field solver .................................................................... 93
A-7 Delay variations due to crosstalk in 16-bit minimum width bus .......... 94
A-8 Staggered repeaters .......................................................................... 94
A-9 Crosstalk due to first neighbour ......................................................... 96
A-10 Crosstalk due to second neighbour ................................................... 96
A-11 Delay vs. drift (optimal repeater) ...................................................... 97
A-12 Percentage variations in delay vs. drift from optimal repeater .......... 97
B-1 Latch followed by mux - logical effort ............................................... 98
C-1 Control path of a multiple-control scheme ....................................... 101
C-2 Latency profile of multiple-control scheme ....................................... 101
D-1 2-stage FIFO .................................................................................... 102
D-2 Limit on $\delta_{\text{min}}$ ......................................................................... 104
D-3 Limit on $\delta_{\text{max}}$ ........................................................................... 105
Acknowledgements

I would like to take the opportunity to thank all the people who have supported me through my MS experience. I would sincerely like to thank my supervisor Dr. Mark Greenstreet for his guidance throughout my research. This thesis would not have been possible without the great courses and thought provoking discussions I had with my co-supervisor Dr. Resve Saleh. Several of the concepts I learnt during interactions with him helped me throughout my research work. I would also like to thank my colleagues in System-on-Chip lab who helped me out with my questions and various tools especially Victor and Roozbeh. I am very greatful to Royal Commonwealth Society for providing funding for my Masters.

Santosh Kumar Sood

University of British Columbia

November 2005
Chapter 1

Introduction

In deep submicron designs, interconnect delays for long (i.e., cross-chip) wires are greater than gate delays. In spite of the introduction of copper interconnect and the use of low-k dielectrics [1], the delay of metal lines continues to increase with each successive technology [2]. As a result, for high-performance designs, the time to transmit a signal across a chip can be several clock cycles [3] [4]. This thesis explores effective methods for signal transmission across a chip over multiple clock cycles.

1.1 Interconnect Overview

Most commercial chips today are designed using a synchronous design flow comprised of two stages: logic synthesis and physical design. The assumption that the delay of each combinational path (including wires) is less than a clock period simplifies synchronous design. In particular, any long wire within a synchronous design or between blocks must have delays shorter than a clock cycle. Thus, any combinational path that presents a delay longer than a clock period is treated as an exception [5], and paths with very short delays can also be problematic. Ensuring correct timing operation becomes a
Chapter 1. Introduction

major challenge under these circumstances.

There are a number of techniques that can mitigate the delay of long wires. One of the options for reducing wire delay is to use better interconnect materials. The introduction of copper [1] reduces the resistance of metal wires, while the adoption of low-k dielectrics [1] lowers the wire capacitance. Copper and low-k dielectrics provided a respite for a technology generation or so, but they do not solve the basic problem of delay of long wires. Innovative circuit design techniques are essential to properly manage the delay of long wires.

Strategies that reduce wire length help reduce wire delays. The addition of extra metal layers with successive fabrication generations helps reduce average wire length by relieving the routing congestion. The pervasive “Manhattan-style” [6] wiring approach, where interconnect is routed along two orthogonal directions, adds a substantial overhead to wire lengths. Routing at 45° was used in early days of integrated circuit design, but it has fallen into disuse because of its impact on tools and masking complexity. A recently introduced X-architecture [7] demonstrated effective use of diagonal routing. The impact on wire length is quite significant, reducing it on average by 20 to 30%. However, all of these methods can not completely avoid the presence of long wires.

The most popular and simple way to reduce propagation delay of long wires is to introduce wire buffering [8]. Figure 1.1 shows a long buffered interconnect between two modules operating at the same global clock frequency. Buffered interconnect reduces the wire delay from a quadratic function of length to a linear one by breaking long wires into short ones. However, even buffering does not guarantee that the signal can travel from one end to other in a single clock cycle. This effectively limits the throughput of the interconnect.

If high throughput is required, then memory elements such as flip-flops or latches can be used as well as simple buffers. This is shown in Figure 1.2. With the introduction
of memory elements in a long interconnect, we effectively pipeline the interconnect. The path delay for the signal to cross in a single clock cycle is reduced to that of the delay of buffers and wires between the two memory elements. Thus, we obtain a lower clock period and higher throughput at the cost of additional latency. To realize the benefits of pipelining, we need fast local clocks that increase the design complexity and increase power consumption. Global clock routing and power for clock distribution are major design concerns [9].

Source-synchronous pipelining, as shown in Figure 1.3, can overcome this drawback of synchronous pipelining, by eliminating the global clock along the signal path. In source-synchronous communication, the sender forwards a clock along with the data, and the receiver uses a first-in-first-out (FIFO) buffer to compensate for the skew\(^1\) between the forwarded clock and the local clock. This FIFO adds extra latency, but we can use various FIFO designs and initialization techniques to limit the overhead. An overview of various FIFO designs is provided in Chapter 2.

**Problem Statement**: The goal of this work is to develop a high-throughput and

\(^1\)Skew is defined as spatial variation in the arrival time of clock transitions. In this work, we are concerned with peak-to-peak skew between the forwarded and the local clock.
low-latency method for cross-chip communication. Given various methods of interconnect pipelining, i.e., synchronous and source-synchronous, there is trade-off between the throughput and latency that can be achieved for each approach. Synchronous methods of interconnect pipelining simplify the design. However, the maximum possible throughput is limited by the worst-case delay of the path between two pipeline stages. We can further improve the throughput of synchronous interconnects by introducing wave-pipelining [10] [11] between the pipeline stages. Wave-pipelining is a technique that allows the circuits between pipeline registers to accept new inputs before the effect of the previous inputs have completely propagated through the circuit.

Source-synchronous methods are extensively used in off-chip interconnects. They have the advantage of high throughput with low design overhead, i.e., they avoid the latency incurred for synchronization that is required in designs with independent clocks. In source-synchronous communication, the sender forwards a clock along with the data, and the receiver uses a FIFO [12] [13] [14] [15] to compensate for the skew between the forwarded clock and the local clock. This FIFO introduces extra latency compared with a synchronous design where FIFOs are not needed. Thus, the central issue for synchronous and source-synchronous on-chip communication is whether or not the savings in relaxed requirements for the global clock network justify the overhead of the FIFO.

Figure 1.3: Source-synchronous gives high throughput with low design overhead

![Diagram of source-synchronous communication](image-url)
Chapter 1. Introduction

1.2 Thesis Contributions

The main contributions of this thesis are:

- A novel, interleaved and distributed FIFO is developed that hides the control latency and enables wave-pipelining between the FIFO stages. An implementation of source-synchronous interconnect using this interleaved and distributed FIFO for the skew compensation is presented.

- A metric of comparison called velocity is defined, and a comparison of the performance of synchronous and source-synchronous signalling is presented on the metrics of throughput, velocity and power.

1.3 Thesis Outline

This thesis is organized as follows:

Chapter 2 begins with an assessment of the challenges posed by global wires in high-performance designs. A summary of related work addressing the problems of long wire interconnect is given. It also gives a description of various first-order issues affecting the performance of long wire interconnect. Then, a brief overview of various synchronous methods, namely two-phase pipelining, multi-phase pipelining and edge-triggered pipelining, is presented.

Chapter 3 describes the source-synchronous approach for interconnect pipelining in greater detail. We examine the effects of the FIFO design on the latency of source-synchronous interconnect. A novel interleaved and distributed FIFO is presented that enables wave-pipelining between the FIFO stages. This FIFO reduces the latency penalty for source-synchronous interconnect.

Chapter 4 defines a metric called velocity for comparing interconnect techniques.
Chapter 1. Introduction

In order to use this metric to evaluate trade-offs between synchronous and source-synchronous signalling, we present the implementation details of the various methods of interconnect pipelining. We then compare the performance of synchronous and source-synchronous interconnects on the metrics of throughput, velocity and power.

Chapter 5 summarize the performance, advantages and limitations of the source-synchronous scheme with an interleaved and distributed FIFO.
Chapter 2

Background

This chapter provides a brief description of the background and challenges posed by global wires in high performance designs. It discusses point-to-point links used for global on-chip interconnect. We describe various circuit components used in this work and first-order variations affecting the timing properties of these components. Finally, an overview of various point-to-point interconnect topologies, i.e., synchronous, asynchronous, source-synchronous and Globally Asynchronous Locally Synchronous (GALS) is given.

2.1 Interconnect Structures

Typically, the wires connecting inputs and outputs of circuits are called interconnect. Here, we use a broader definition since this work emphasizes interconnect for cross-chip communication. Some of the interconnect techniques used in current chips are described in the following sections.
Chapter 2. Background

2.1.1 Point-to-Point Links

Point-to-point links are generally preferred in designs with high throughput and low latency requirements. Using global dedicated interconnect increases design complexity, as it is difficult to characterize the electrical properties of unstructured wiring early in the design. In comparison to high level links like buses and NoCs [5], point-to-point links are easier to implement since they do not have overheads due to arbiters and multiplexers.

In this work, we emphasize dedicated point-to-point links as these are generally used for cross-chip communication in high performance designs. Furthermore, point-to-point links are essential components in many other interconnect schemes such as NoCs. The performance of point-to-point links is independent of special IP blocks such as switching fabrics, links and arbiters. Thus, it gives a basic framework for comparing the performance of various techniques of interconnect pipelining, i.e., two-phase pipelining, multi-phase pipelining, edge-triggered pipelining and source-synchronous. We study various circuits and timing techniques for point-to-point links in later sections.

2.2 Interconnect Components

This section examines the circuit components used in this work namely wire models, buffers and latches.

2.2.1 Wires and Buffers

Wires can be modeled using a lumped RC model [6]. When the wire is short and switching frequencies are in the low to medium range, it is meaningful to consider only the capacitive component of the wire. However, long metal wires have significant resis-
tance, and a resistive-capacitive model should be adopted. A single lumped RC model is pessimistic and inaccurate for long interconnect wires, which are more accurately represented by a "distributed" RC model as shown in Figure 2.1.

\[
\begin{align*}
\text{R}/4 & \quad \text{R}/4 & \quad \text{R}/4 & \quad \text{R}/4 \\
\text{C}/8^p & \quad \text{C}/4^p & \quad \text{C}/4^p & \quad \text{C}/4^p \\
 & \quad \text{C}/8^p & \quad \text{C}/4^p & \quad \text{C}/4^p & \quad \text{C}/8^p
\end{align*}
\]

**Figure 2.1:** Interconnect RC model for long wire

As discussed in Section 1.1, the delay of a wire grows quadratically with its length due to RC effects [16]. The most popular and simple way to reduce propagation delay of long wires is to introduce wire buffering. Figure 2.2 shows a buffered interconnect between two modules operating at the same global clock frequency.

\[
\begin{align*}
\text{Buffer} & \quad \text{R} & \quad \text{R} \\
\text{Module 1} & \quad \text{C}/2 & \quad \text{C}/2 & \quad \text{C}/2 & \quad \text{C}/2 & \quad \text{global clock} & \quad \text{Module 2}
\end{align*}
\]

**Figure 2.2:** Interconnect buffering

Breaking up the line into smaller segments and inserting a buffer between them reduces the wire delay and offsets extra delay due to buffers if the wire is sufficiently long. Because buffers introduce their own delay and load, there is an optimal number and size of buffers for minimizing delay. The basic analysis of inserting buffers in long wires was presented by Bakoglu in [16]. Simplified equations can be derived for optimal wire segment length and buffer size [8] [17] [18]. These equations are given in Appendix A-1.
2.2.2 Latches and Flip-Flops

It is well-understood that single cycle inter-module communication is getting more and more difficult due to the quadratically growing delay of long wires. Wire buffering helps to overcome the quadratic increase by making it a linear function of length. However, in high performance designs, it still does not guarantee single cycle communication. For example, with optimal buffer insertion and wire-sizing, five clock cycles are needed to go from one end of the chip to the other in 28.3 x 28.3 mm die in the 70nm technology generation [19]. Thus, we have to use some form of interconnect pipelining as shown in Figure 2.3. In interconnect pipelining, we insert clocked storage elements to store intermediate data waves. These memory elements can be latches or flip-flops.

![Interconnect pipelining](image)

Figure 2.3: Interconnect pipelining

A latch is level-sensitive, i.e., it copies its input to output during the period of time the clock is active. In contrast, the process of capturing data with a flip-flop is associated with a transition edge of the clock; in other words, flip-flops are edge-triggered. These clocked elements are characterized by: set-up time, hold time, D-to-Q delay (D is input, Q is output of the latch or flip-flop) and clock-to-Q delay.

These are defined as follows:

**Set-up time**: The minimum time that data (D) must be stable before the active clock edge of the latch or flip-flop to capture the data reliably.

**Hold time**: The minimum time that data (D) must be stable after the active clock edge of the latch or flip-flop to capture the data reliably.
**D-to-Q delay:** The delay from data (D) to output (Q), when the data arrives while the latch is transparent and the set-up constraint is not violated.

**Clock-to-Q delay of latch:** Propagation delay from a clock edge to the output (Q), assuming the data (D) has settled early enough relative to the leading clock edge.

**Clock-to-Q delay of flip-flop:** Propagation delay from a clock edge to the output (Q), assuming the data input satisfied the set-up and hold requirements.

Latches and flip-flops each have their own advantages and disadvantages. Because latches allow signals to pass through in an interval, the path delay between two latches can be greater than a clock cycle provided it is compensated by a short delay in some other path. This is known as **time-borrowing** [20]. Hiding the set-up overhead is a particular use of time-borrowing. Time borrowing can be used to avoid incurring set-up overhead in long interconnect, provided latches are inserted in a way that the signal always arrives at a latch in its active interval with sufficient margin for set-up and skew. Another advantage of latches is flexibility of timing constraints. Flip-flops being edge-triggered do not allow the path delay to be greater than a clock cycle. Thus, in a flip-flop based designs, one pipeline stage can not take advantage of the slack available in the other pipeline stages.

There is a trade-off between the set-up time and the D-to-Q delay of latches and flip-flops. Since the set-up time is not critical for latches, we can trade an increased set-up time for a lower D-to-Q delay, thus improving the overall latency. This trade-off is not possible with flip-flops. In case of flip-flops, we can optimize for the sum of set-up time and D-to-Q delay.

The latch model used in this work is a simple transmission gate latch shown in Figure 2.4 [21] [22]. This is a level-sensitive static latch, with a minimum sized keeper. When clock signal $\phi$ is high, the transmission gate is on and data at the input is captured at the output provided the set-up constraint is satisfied. When clock signal goes low the
transmission gate turns off and "keeper" maintains the data value stored in the latch. The latch sizing and characterization is described in Appendix A-2 and Appendix A-3 respectively.

![Transmission-gate latch](image)

**Figure 2.4:** Transmission-gate latch

### Chapter 2. Background

#### 2.3 Other Interconnect Issues

In this section, we look at how the variations due to capacitive coupling and PVT (process, voltage and temperature) variations affect the delay of the various circuit components in the pipelined interconnect.

#### 2.3.1 Crosstalk

The delay of a buffered wire between the two memory elements in a pipelined interconnect may vary as a result of crosstalk. *Crosstalk* [23] is defined as unwanted signal coupling from the neighbouring wires to a given wire. In high-performance designs, this inter-signal coupling can be both capacitive and inductive. Capacitive crosstalk is dominant at the switching frequencies in typical designs. The impact of crosstalk is a variation in the effective impedance of the signal line under examination. Since, the performance of both synchronous and source-synchronous techniques depends on delay variations in signal lines, crosstalk is one of the most important first-order design parameters affecting the performance of various interconnect topologies. We perform a

---

1Inductive coupling can be problem in higher frequency designs but will be ignored here.
brief study of crosstalk effects in TSMC 0.18\(\mu\)m technology in Appendix A-4. We also study a way to reduce the crosstalk using staggered repeaters.

2.3.2 PVT Variations

The timings of all the circuit elements after fabrication will deviate from those of ideal elements. These variations arise due to process, voltage supply and temperature variations.

Variations in process parameters such as impurity concentrations, oxide thickness and diffusion depth, are caused by non-uniform deposition, growth and diffusion steps. These result in varying values of transistor parameters such as threshold voltage, sheet resistance, etc. Variations in dimensions of devices occur due to the limited resolution of photolithographic processes. This causes a deviation in the \((W/L)\) ratio of transistor devices and geometries of the interconnect wires.

The supply voltage delivered to the circuit is by no means a constant [24]. For instance, the voltage supplied by a power supply can change \(\pm 10\%\). This must be taken into account when designing the circuits.

The temperature gradient across the chip further accentuates the variations in circuit timing. Specifically, CMOS circuits tend to slow down with an increase in temperature [18].

All of these variations manifest themselves as overheads in various schemes to drive long wires. The variations can be in the form of wire delay variations, buffer delay variations, clock skew, clock jitter and unmatched data and clock paths. To accurately capture these effects, the circuits must be simulated in SPICE at many process corners to ensure proper operation under PVT variations.
2.4 Interconnect Timing Schemes

We now present a variety of interconnect schemes that can be used to address the problem of multi-cycle communication.

2.4.1 Synchronous Design for Interconnect

The simplest synchronous pipelined approach is to insert latches along the buffered signal path. That is, when it is no longer possible to cross the signal path in a single clock cycle, we divide the signal path into smaller paths and use latches to store intermediate data waves. These latches are controlled by the global clock. To realize the full benefits of pipelining, we need fast local clocks. This tends to add to the design complexity and increases power consumption.

The best-case synchronous design is one that gives the lowest latency for a given throughput. For this design, we provide multiple clock phases available at any point on the chip. It makes extensive use of constructive skew\(^2\) to schedule the arrival of the clock at the latches.

We can further improve throughput of these interconnect topologies by using wave-pipelining. The concept of wave pipelining has existed since the 1960s [10] [11] [25]. Wave-pipelining allows the application of new inputs to a combinational logic block before the previous values have propagated to the output. Multiple waves corresponding to successive computations exist simultaneously in the same computational block, hence the name wave-pipelining. The same concept can be used for signal transmission in interconnect.

\(^2\)The term constructive clock skew refers to a clock skew that is intentionally created between two clock signals and that can be adjusted with predictable effects. This is in contrast to an uncontrolled clock-skew that exists in the circuit due to delay differences along the clock lines.
2.4.2 Two-Phase Clocking

Figure 2.5 illustrates a two-phase clocking scheme. This is a simple and conservative design that establishes a lower bound on the performance of latch based synchronous techniques, i.e., the throughput that can be achieved using synchronous techniques. The throughput in this case depends on the maximum delay between any two pipeline stages. Also, it does not make use of wave-pipelining, as we want to consider the simplest synchronous interconnect that does not require much design effort.

![Figure 2.5: Two-phase clocking](image)

In two-phase clocking, we have a global clock network supplying the clock at the required points on the chip. We use latches for pipelining, with a series of alternating active-high and active-low latches, as shown in Figure 2.5. Each pipeline stage has half a clock cycle to transmit data including the latch delay. Each pipeline stage consist of $N$ buffered wire segments. Latch-based pipelining provides flexibility on timing constraints, as latches allow signals to pass through during the interval when the clock is high. This allows the path delay between two latches to be greater than half a clock period, provided it is compensated by a short delay in some other path. As mentioned earlier, this is known as time-borrowing [20].

Latches are slower than the simple buffers. For a given throughput target, we want to have maximum number of buffered wire segments between two latches, i.e., a large $N$, before the set-up constraint becomes critical and we have to latch the data.

The timing requirements for this approach are relatively simple. A long, synchronous pipelined link will operate properly if the following three conditions are satisfied:
Set-up: Data arriving at latch $i$ coincident with a maximally retarded clock edge satisfies the set-up requirement of the latch $i+1$ when its clock is maximally advanced.

Hold: Data passing through latch $i$ with a maximally advanced clock satisfies the hold requirement of the latch $i+1$ when its clock is maximally retarded.

Propagation: Data arrives at each latch earlier than the latest possible arrival of the clock edge that makes the latch transparent.

The throughput of two-phase clocking depends on the maximum case path delay between the pipeline stages. For high target throughput, we have to reduce the delay between two pipeline stages. This can be done by increasing the granularity of the pipelining, i.e., by reducing the number of buffered wire stages $N$, between two memory elements. This increases the number of latches in the interconnect. The increased number of latches results in additional latch overheads and thus increases the latency of the interconnect. Clearly, there is trade-off between high throughput and low latency. This trade-off is further investigated in Chapter 4.

### 2.4.3 Multi-Phase Clocking

Multi-phase clocking, shown in Figure 2.6, is a more of a theoretical synchronous implementation than a practical one. This scheme provides an upper bound on the performance achievable with synchronous interconnects. It makes optimistic assumptions about the clock network.

It assumes that any required clock phase is available at any point on the chip, as indicated by the symbols at the clock input to the latches. It uses a global clock with constructive skew. Latches are used for the pipelining, because of their property of time borrowing. Each pipeline stage consists of $N$ buffered wires between two pipeline stages.
Chapter 2. Background

17

Figure 2.6: Multiple-phase clocking

It also makes "aggressive" use of wave-pipelining. Recall that wave-pipelining occurs when combinational logic is clocked faster than the latency through the logic would allow. Several data waves are active in the logic without being separated by storage elements. The same approach can be used in interconnects. The throughput of multiphase clocking depends on worst-case difference between the maximum and minimum case path delays.

The throughput of synchronous techniques suffers with increasing skew and jitter. *Skew* can be defined as spatial variation in the arrival time of clock transitions [26], while *jitter* is defined as temporal variation of the clock period at a given point on chip. Traditionally, designers target a skew of about 10% of the clock period [27] [28]. Long wire delays and variation in buffer delays make these targets challenging. Designers try to achieve these targets by using skew compensation techniques [29], but all these methods can cut into the overall power budget.

2.4.4 Asynchronous Design for Interconnect

An alternative to the well-understood synchronous methods are the so-called asynchronous design techniques. Asynchronous protocols are transactional in nature. These employ handshaking signals for communication between the pipeline stages. As shown in Figure 2.7, there is a full control cycle for each data transfer. It consists of a request token traveling from stage $i - 1$ to stage $i$ to indicate that a new data value is available,
and an acknowledgment token traveling from stage $i$ to stage $i-1$ to indicate the value has been accepted. The data path does useful work while the request is moving to the right, but no useful work is done while the acknowledgment is traveling back from stage $i$ to stage $i-1$, and this ultimately limits the throughput. The presence of long wires between the handshake stages further limits the throughput as wires add to both the forward and reverse delays.

Figure 2.7: Asynchronous protocol

A scheme proposed by Ho et al. [30] overcomes this reverse transactional latency of long wires. It uses a second control path in parallel with the first. The second control path sends a new data token on the data path while the first control signal is busy acknowledging the previous token. Further details of the scheme are presented in Section 2.6.3.

### 2.4.5 GALS

"Globally asynchronous, locally synchronous" (GALS) [31] [32] [33] [34] [35] [36] [37] design combines the synchronous design methods with asynchronous communication methods. In GALS, all modules in the system are designed in accordance with a traditional synchronous methodology. Each module operates with its own local clock; however, data exchange between modules follows a strict full-handshake protocol.
Latency insensitive design as proposed in [38] is an extension of GALS. It proposes the notion of islands of different frequency on the chip and channels between the islands which define the communication link between the islands. The distinction between a latency insensitive system and an asynchronous system is that, in latency-insensitive systems, communication is treated as though it was synchronous while it might actually be realized using hand-shake signalling.

2.5 Source-Synchronous Design

Source-synchronous designs give high throughput with low latency. In source-synchronous interconnect, we do not have a clock network along the long interconnect. In this type of design, the circuit that is producing the data pattern will create its own clock that is transferred along with the data.

Generating the clock in the same geographical location as the data, in addition to having the clock traverse the same media as the data bus, creates a much tighter timing correlation. The tighter timing means that the data valid windows of the data being latched are better aligned with the clock used to latch them. This reduces the channel-to-channel skew\(^3\) and allows data transfer rates to exceed that of a synchronous architecture.

Absolute path delays are not critical in the timing for source-synchronous interfaces. The throughput of a source-synchronous clocking scheme depends on tracking in the delays between the data and strobe paths. However, any process, voltage and temperature variations cause the mismatch in data and strobe path, resulting in a limit on maximum throughput.

Figure 2.8 shows details of the clock forwarding part of the source-synchronous in-

---

\(^3\)Channel-to-channel skew refers to the time difference of the data valid windows between various signal paths.
terconnect. Here, the sender forwards the clock along with the data and the receiver uses a FIFO (not shown) to compensate for phase differences between its local clock and the forwarded clock. In a proper design, the matching buffers match the latch delay over the full range of PVT variations [39]. The edge detector circuit detects both the rising and falling edge of the clock and produces pulses wide enough to satisfy the set-up requirement at the latch.

![Source-synchronous forwarded clock and data path](image)

**Figure 2.8:** Source-synchronous forwarded clock and data path

The following constraints must be satisfied to ensure proper operation of source-synchronous link.

**Set-up:** To satisfy the set-up timing, we have to ensure that, at each latch, there is enough set-up margin between the forwarded strobe and the corresponding data. This margin must be high enough to ensure that the set-up condition is not violated over all PVT variations. This is accomplished by designing the delay of the strobe path to be a little bit larger than that of the data path. In this work, we use a margin of around 20% [39], with process and temperature variations each accounting for 5% of the margin, while the voltage variations for the rest$^4$.

$^4$In source-synchronous the structure of the data and strobe paths are same, consisting of series of buffered wire stages. Since they are close together on a chip, their delays are expected to drift in the same direction due to PVT variations. More details on tracking are provided in Chapter 4.
Hold: To ensure that the hold-time constraint is not violated, we compare the timings of the current strobe and the next data pulse. To satisfy the hold condition at the latch, the next data must not appear until at least hold time after it latches the current data.

Prop: The propagation constraint is defined for the critical sequential path through a series of pipeline stages, where the set-up constraint is barely satisfied at the end of the path. The propagation constraint ensures that data arrives at each latch earlier than the latest possible arrival of the clock edge that latches the data.

2.6 FIFOs

The forwarded clock in source-synchronous interconnect suffers from skew problems due to process, voltage supply and temperature variations along the forwarded strobe path. Since this forwarded clock has to be synchronized with the local clock at the receiver, a FIFO is used to compensate for the skew between the two clocks.

In this section, we give a brief description of various FIFO designs used to compensate for the skew between the forwarded and the local clock. We also describe the advantages and limitations of each design.

2.6.1 FIFO Control

In this work, we consider a type of design called handshaking ripple FIFOs. These FIFOs use a ripple through or “flow-through” design that exploits the local communication between the stages as an alternative to the global clock. Individual FIFO stages alternate between the full and empty states as the data items move through the FIFO. Figure 2.9 shows a FIFO control with a common boundary between two stages of set-reset flip-
flop (SRFF). The control path consists of two control circuits driving a single common control wire that synchronize on two events: one, the arrival of a token from stage $i$ which indicates that the new data is available in stage $i$ and, second, the arrival of a bubble from stage $i + 1$ which indicates that stage $i + 1$ has passed on its previous data to the next stage and has an empty space. When both these events have occurred, the control moves data from stage $i$ to the next stage $i + 1$ and the bubble from stage $i + 1$ to stage $i$.

![Figure 2.9: FIFO control circuit](image)

We need simple and fast control circuits to implement this handshaking mechanism between the FIFO stages. GasP chains [40] provide a simple and low overhead design option for implementing these control circuits. GasP is a pulse-asynchronous control circuit that provides control for simple pipelines. It has the advantage of implementing a low cost handshake circuit; six gate delays are incurred for a full control cycle. Figure 2.10 shows a control path with two GasP stages driving a common control wire. Between GasP pipeline stages, a single control wire carries both request and acknowledge messages with the request event driving the control wire high, and an acknowledgement driving the wire low. A request traveling from stage $i$ to stage $i + 1$ incurs two gate delays through the gates labelled 1 and 2. An acknowledgement traveling from stage $i + 1$ to stage $i$ requires four gate delays through the gates labelled 3 to 6. Thus a single control cycle constitutes six gate delays through gates 1 to gate 6.
Ripple FIFOs have the following advantages over the more traditional pointer FIFOs: fast cycle time (6 gate delays, GasP), low control overhead, easy to embed in clocked systems and low latency for short FIFOs.

### 2.6.2 Linear FIFO

There has been extensive work on FIFOs and components to handle timing discrepancies between single-clock subsystems. One approach is to synchronize the signal using a synchronizer. Seitz [41] uses a two flip-flop synchronizer to handle clock uncertainty between two domains. In this case, the latency of the synchronizer is proportional to the number of flip-flops.

In another design, a single-stage linear FIFO [12] is placed between two communication domains. This FIFO shown in Figure 2.11, provides a skew tolerance of around two clock cycles. Worst-case skew across the long-wire interconnect is directly proportional to the length of the interconnect. Thus, we need skew tolerance of multiple clock cycles as we increase the length of the interconnect. The required FIFO depth increases proportionally to the required skew tolerance. These additional FIFO stages add latency to the data path.

---

5Pointer FIFOs are usually implemented with dual-port RAM, organized as a ring buffer structure addressed using separate read and write pointers.
2.6.3 Twin-Control FIFO

The use of a linear FIFO at the end of the forwarded clock path to compensate for the drift between the forwarded clock and receiver clock adds extra overhead to the latency of the forwarded path. Figure 2.12 shows a source-synchronous scheme with a lumped FIFO at the end of the signal path. However, this introduces additional latency due to the latches.

Instead of using a lumped FIFO of Figure 2.12, we can use an asynchronous FIFO with wire segments distributed between the FIFO stages, as shown in Figure 2.13. This helps to save on overhead of the delay due to latches by moving the pipeline stages to the FIFO, thus reducing the number of latches in the data path. Effectively, the distributed FIFO saves on the overhead of latches by distributing some of the interconnect stages in the FIFO.

As discussed in Section 2.4.4, in the asynchronous protocols during the reverse ac-
knowledgment stage, the data path is idle and waits for the acknowledgment from the subsequent stage before sending the new data. A scheme is proposed by Ho et al. [30] to overcome this reverse transactional latency. They propose to use the idle reverse transaction time to do useful work. This is done by replicating the control path.

The twin-control scheme [30] shown in Figure 2.14, uses a series of GasP stages at the transmitter and receiver end, which share the responsibility of driving a single control wire. To indicate a request, the driver GasP stage momentarily drives the control wire high; a pair of weak keepers maintains the state of control wire. Later the receiver acknowledges the request by generating a pulse signal that drives the control wire low.

It has two control paths working in parallel, namely, Top control and Bottom control. The second control path (Bottom control) sends a new data token on the data path while the first control path (Top control) is busy acknowledging the prior token. Later when Bottom control is acknowledging its token, Top control can move on yet another data token. Alternate transfer of the control between the two control paths is achieved by use of ping-pong (SR Latch) circuit with delay-matched alternation wires, which matches the data path delay. Thus, while one control path is busy acknowledging its previous data token, the other control path can send its data token over the shared data path. Thus, the data path is not idle at any time.

The tri-state data latch must be used for each FIFO stage. This is required when the FIFO is sink limited. This is because once the top stage $i$ has sent data to stage
Chapter 2. Background

Figure 2.14: Twin control with shared handshake wire

$i + 1$, we do not want the bottom stage $i$ to wait for the acknowledgment of the top stage. If we would have used a simple latch, data for the bottom stage would corrupt the data from the top stage. The twin-control scheme uses tri-state latches so that they can drive the data path in alternation.

We can determine the clock period, $P$, of the FIFO by examining the delays. Specifically, the clock period is limited by the sum of worst case forward\(^6\) FIFO delay $\delta_{f,\text{max}}$, and reverse\(^7\) FIFO delay $\delta_{r,\text{max}}$. Each FIFO transition occurs once every $2P$ time units thus the FIFO stage must be able to complete its worst-case forward and reverse transaction with in this time interval. Equation 2.1 gives the limit on the clock period of the

\(^6\)The forward delay of the FIFO is the time from arrival of a request at stages $i$ to its transfer to stage $i+1$.

\(^7\)The reverse delay of the FIFO is the time from the arrival of an acknowledgement at the stage $i+1$ to its transfer to stage $i$.  


twin-control scheme:

\[ 2P \geq \delta_{f,w}^{max} + \delta_{r,w}^{max} \]  

(2.1)

The forward and reverse FIFO delays include the wire delay distributed between the FIFO stages. Therefore, they increase quadratically with the wire length. Thus, the throughput degrades as we increase the per stage wire length.

In the next chapter, we propose a novel scheme to buffer the control wires to overcome the quadratic increase in the delay of long control wires. However, throughput is still limited by the delay of FIFO stages. We further propose a scheme that improves the throughput of the FIFO by introducing a novel interleaved FIFO that allows wave-pipelining between the FIFO stages.
Chapter 3

Novel Interleaved and Distributed FIFO

In this chapter, we study limitations of various FIFO designs used for compensating for the skew between forwarded and local clocks. A novel interleaved and distributed FIFO is presented that hides the control latency and enables wave-pipelining between FIFO stages.

3.1 Introduction

In source-synchronous interconnect, a forwarded clock propagates through a series of clock forwarding interfaces. This forwarded clock suffers from skew due to PVT variations along the forwarded strobe path. The receiver uses a FIFO to compensate for the skew between its local clock and the forwarded clock. We consider ripple FIFOs made of GasP [40] control circuits. As discussed in Section 2.6.1, these FIFOs have the advantages of short cycle times, low control overhead, ease for embedding in clocked systems and low latency for short FIFOs.
Figure 3.1 shows an implementation of source-synchronous interconnect. It consists of a clock forwarding network with a corresponding signal path, and a FIFO at the end. As shown, $\delta_t$ represents the delay of the forwarded clock $\phi_T$ from the common clock source to the transmitter end of the FIFO, while $\delta_r$ represents the delay of the receiver clock $\phi_R$ from the common clock source to the receiver end of the FIFO.

Figure 3.2 shows the timing diagram for this source-synchronous interconnect with the FIFO at the end. We have a global clock $\phi_o$ routed to both the stage 1 and stage 2 of the clock forwarding network, both of which operate at same frequency and communicate over the long interconnect. Stage 1 generates its local clock $\phi_t$ that is used to latch the data and forwards the clock along the clock forwarding network. As shown in Figure 3.2, we have multiple data and clock events active in a pipeline stage simultaneously. For example, in the interval from the event that data $D_i$ is transmitted by module 1 to the event that it is captured at next stage, we have multiple data items $D_{i-2}$, $D_{i-1}$ and $D_i$ existing in the pipeline stage simultaneously. We also have a FIFO at the end to provide adequate skew tolerance between the forwarded clock $\phi_T$ output by the final clock forwarding stage and receiver clock $\phi_R$. Note that the FIFO latency is $k$ cycles in this example.

![Figure 3.1: Source-synchronous with FIFO](image-url)
3.2 Linear FIFO

Figure 3.3 shows a linear FIFO [42] with $n_{fifo}$ stages. It combines synchronous and self-timed\(^1\) techniques to achieve high throughput. It uses the global clock to set the rate of data transfer, while the self-timed FIFO compensates for the skew between the

\(^1\)In self-timed circuits, the process of computation is controlled through local clocks and local handshaking and hand-off between adjacent units.
transmitter and receiver clock. Each FIFO stage consists of a control circuit and a corresponding memory element. In Figure 3.3, $U_0$ is the transmitter and $U_{n_{fifo}+1}$ is the receiver. The control stages are made of GasP stages driving common control wires.

This FIFO can be used to synchronize the $\phi_T$ and $\phi_R$ clocks, where both have the same clock period but an unknown phase drift\(^2\). The number of stages in the FIFO, $n_{fifo}$, is selected to provide sufficient skew tolerance between $\phi_T$ and $\phi_R$.

![Figure 3.3: Linear FIFO](image)

**Figure 3.4: Latency of the FIFO**

### 3.2.1 Latency of a Linear FIFO

The latency of the FIFO varies between some minimum and maximum value depending on the phase difference between the forwarded and the local clock. The following

\(^2\)The phase drift is the unknown timing relationship between $\phi_T$ and $\phi_R$ caused by the varying forwarded path delay.
notation is used in the analysis of the FIFO to determine the minimum and maximum values of latency and how it relates to the FIFO depth.

- We use $\phi(i)$ to denote the event of the $i^{th}$ rising edge of $\phi$.

- $P$ is the time period of the clock, where $P_{\text{max}}$ and $P_{\text{min}}$ denote the maximum and minimum values of the clock period$^3$.

- As shown in Figure 3.1, $\delta_t$ represents the (min or max) delay of the forwarded clock from the common clock point to the transmitter end of the FIFO.

- As shown in Figure 3.1, $\delta_r$ represents the (min or max) delay of the local receiver clock from the common clock point to the receiver end of the FIFO.

- $\Lambda$ denotes latency of the FIFO. It is defined as an integer number of clock cycles, taken from the instant when the transmitter outputs a value to the instant when the receiver acquires this value. In Figure 3.3, the latches $U_0$ and $U_{n_{\text{fifo}}+1}$ constitute the transmitter and receiver domains, respectively. As shown in Figure 3.4, the data launched at transmitter clock event $\phi_t(i)$ is captured at the $\phi_r(j^{th})$ event at the receiver, thus the latency of the FIFO is $(j - i)$ clock cycles.

- $\lambda$ denotes the latency, as the actual delay in seconds from the transmitter to the receiver end of the FIFO.

- $\delta_{f,\text{lin}}$ is the forward delay of the linear FIFO from the time from arrival of a request at stages $i$ to its transfer to stage $i+1$, when an acknowledgment is already present$^4$.

---

$^3$The clock period has short term variations due to jitter.
$^4$We use max and min superscript to denote the maximum and minimum value of the forward delay of a FIFO stage.
• $\delta_{r,lin}$ is the reverse delay of the linear FIFO from the time from arrival of an acknowledgment at stages $i+1$ to its transfer to stage $i$, when a request is already present\(^5\).

**Minimum FIFO latency:** The minimum latency of the FIFO corresponds to the time for a value to propagate unobstructed from the first stage of the FIFO until it is acquired by the receiver's input latch. In a FIFO with $n_{fifo}$ stages, $n_{fifo}$ transfers occur between the output of a value by a transmitter and the arrival of that value at $U_{n_{fifo}+1}$, the receiver latch and we allow $t_{set-up}$ for the set-up time of the latch. Thus, the worst-case minimum latency ($\lambda_{min}$) of the FIFO is given by:

$$\lambda_{min} = n_{fifo}\delta_{f,lin}^{max} + t_{set-up} \tag{3.1}$$

**Set-up constraint at latch $U_{n_{fifo}+1}$:** The data loaded in latch $U_0$ on event $\phi_T(i)$ is loaded into latch $U_{n_{fifo}+1}$ on event $\phi_R(i + \Delta)$. The time of the earliest clock event at latch $U_{n_{fifo}+1}$ is given by:

$$\delta_{r}^{min} + \Lambda P_{min} \tag{3.2}$$

The latest time that the corresponding data becomes available at the input of latch $U_{n_{fifo}+1}$ is given by:

$$\delta_{t}^{max} + n_{fifo}\delta_{f,lin}^{max} \tag{3.3}$$

This corresponds to sum of two delays: the time for forwarded clock to reach latch $U_0$, i.e., $\delta_{t}^{max}$, and delay through the FIFO stages.

The set-up requirement is satisfied for the latch $U_{n_{fifo}+1}$ if:

\(^5\)We use max and min superscript to denote the maximum and minimum value of the reverse delay of a FIFO stage.
Thus, the following condition must be satisfied for proper operation of the FIFO:

\[
\delta_r^{\text{min}} - \delta_t^{\text{max}} + \Lambda P_{\text{min}} \geq \lambda_{\text{min}} \tag{3.5}
\]

**Maximum FIFO latency:** The maximum latency of the FIFO corresponds to the time for a request to propagate from the first stage of the FIFO to the last stage when each FIFO stage is waiting for the acknowledgment of its previous request token. If \(\delta_{r,\text{lin}}^{\text{max}}\) is the maximum reverse delay of FIFO and each new request occurs once every \(P_{\text{min}}\) time units, the maximum delay between the arrival of a request signal at stage \(i\) and its transfer to stage \(i + 1\) is given by \((P_{\text{min}} - \delta_{r,\text{lin}}^{\text{max}})\). In a FIFO with \(n_{\text{fifo}}\) stages, \(n_{\text{fifo}}\) transfers occur between the removal of a value by the receiver and the arrival of the empty slot at the first stage of the FIFO and we allow \(t_{\text{hold}}\) for hold time at latch \(U_0\). Thus, the maximum latency of the FIFO is given by:

\[
\lambda_{\text{max}} = n_{\text{fifo}}(P_{\text{min}} - \delta_{r,\text{lin}}^{\text{max}}) + (P_{\text{min}} - t_{\text{hold}}) \tag{3.6}
\]

**Hold constraint at latch \(U_1\):** The data loaded in latch \(U_0\) on event \(\phi_T(i)\) is loaded into latch \(U_{n_{\text{fifo}}+1}\) on event \(\phi_R(i + \Lambda)\). This data was loaded into latch \(U_1\) at the instant \(n_{\text{fifo}}(P_{\text{min}} - \delta_{r,\text{lin}}^{\text{max}})\) time units before the event \(\phi_R(i + \Lambda)\). This data must be loaded into latch \(U_1\) before the output of \(U_0\) changes in response to the next clock event \(\phi_T(i + 1)\).

The earliest time of the next clock event \(\phi_T(i + 1)\) at latch \(U_1\) is no later:

\[
P_{\text{min}} + \delta_t^{\text{min}} \tag{3.7}
\]
Chapter 3. Novel Interleaved and Distributed FIFO

The latest time that the previous data becomes available at the input of latch \( U_1 \) no later than:

\[
\delta_r^{\text{max}} + \Lambda P_{\text{max}} - n_{\text{fifo}}(P_{\text{min}} - \delta_{r,\text{lin}}^{\text{max}})
\]

(3.8)

The hold requirement is satisfied for the latch \( U_1 \) if:

\[
P_{\text{min}} + \delta_{t,\text{min}}^{\text{min}} - t_{\text{hold}} \geq \delta_r^{\text{max}} + \Lambda P_{\text{max}} - n_{\text{fifo}}(P_{\text{min}} - \delta_{r,\text{lin}}^{\text{max}})
\]

\[
\delta_r^{\text{max}} - \delta_{t,\text{min}}^{\text{min}} + \Lambda P_{\text{max}} \leq n_{\text{fifo}}(P_{\text{min}} - \delta_{r,\text{lin}}^{\text{max}}) + P_{\text{min}} - t_{\text{hold}}
\]

(3.9)

Thus, the following condition must be satisfied for proper operation of the FIFO:

\[
\delta_r^{\text{max}} - \delta_{t,\text{min}}^{\text{min}} + \Lambda P_{\text{max}} \leq \lambda_{\text{max}}
\]

(3.10)

Skew tolerance, \( \sigma \), provided by the FIFO is given by the difference between the maximum and minimum latency.

\[
\sigma = \lambda_{\text{max}} - \lambda_{\text{min}}
\]

\[
\sigma = n_{\text{fifo}}(P_{\text{min}} - \delta_{f,\text{lin}}^{\text{max}} - \delta_{r,\text{lin}}^{\text{max}}) + (P_{\text{min}} - t_{\text{hold}} - t_{\text{set-up}})
\]

(3.11)

The skew tolerance provided by a FIFO of depth \( n_{\text{fifo}} \) is equal to difference between maximum and minimum latency. Thus, the required FIFO depth \( n_{\text{fifo}} \) is directly proportional to the skew between the forwarded strobe and the local clock at the receiver. Worst-case skew across a long wire interconnect is directly proportional to the interconnect length. Thus, as we increase the length of a cross-chip communication interconnect, the required FIFO depth increases proportionally. These additional FIFO stages add latency to the data path.

We can improve the performance of the source-synchronous interconnect by distributing the FIFO over the long interconnect. The twin-control scheme [30] discussed
in Section 2.6.3 can be used to distribute the wire segments between the FIFO stages. This helps to reduce the overhead of delay due to the latches by moving the pipeline stages to the FIFO. This effectively reduces the number of latches in the data path. However, the twin-control scheme suffers from quadratically increasing forward and reverse FIFO delays, limiting the throughput as we increase the wire length per pipeline stage. In the next section, we introduce a scheme to overcome the increasing control path delays.

3.3 Novel FIFO 1: Buffered Control Wire

The twin-control scheme proposed by Ho et al. [30] uses a bi-directional single control wire between adjacent FIFO stages which cannot be buffered. We can overcome quadratically increasing delays of the control wire by replicating and buffering the control wire. This means that separate wires are used for the request and acknowledgment. Since wires are no longer bi-directional, we can buffer the wires.

Figure 3.5 shows this new twin-control scheme with replicated and buffered handshake wires. Most of the implementation details are the same as for the twin-control FIFO. The only difference is in the control path. As shown in figure, we have two separate buffered wires between adjacent control stages, one carrying a request signal from stage $i$ to stage $i+1$, and a second carrying an acknowledgment signal from stage $i+1$ to stage $i$.

The buffers 1 and 3 in the control circuit are simple inverting buffers. There is an even number of buffers in the control path for preserving the logic of high-request and low-acknowledgment. Buffers 2 and 4 include only the NMOS or the PMOS part of the inverter. The stacked transistors at the end of request and acknowledgment paths prevent short-circuit current in the control paths.
Figure 3.5: Twin control with buffered handshake wire

Figure 3.6 shows the implementation details of the top control path. The control path consists of a series of GasP stages. The control circuit in stage \( i \) of the FIFO receives a request from its predecessor stage \( i-1 \) on its \( in \) port and sends a request to stage \( i+1 \) on its \( out \) port. The pulse control requires that the control wire be driven for a time long enough for its state to propagate to the stage \( i+1 \). This request travels along the \textit{req control path} and triggers the stage \( i+1 \) (provided stage \( i+1 \) has its acknowledgment from the previous request to stage \( i+2 \) and its \( sr \) input is enabled). Stage \( i+1 \) is triggered and it sends an acknowledgment back to stage \( i \) along the \textit{ack control path}. Each time a stage is triggered, a pulse is generated at the \( en \) port of the controller to latch the data in the corresponding memory element.

Equation (3.12) gives constraint on throughput of the twin-control buffered wire
Here, \((\delta_{f,rep} + \delta_{r,rep})\) is the sum of the forward and reverse delay of a FIFO stage and \(P\) is the clock period. This means that since each internal state transition of a FIFO stage occurs once every \(2P\) time units. The FIFO stage must complete its forward and reverse transaction within this time interval. Also, the clock period \(P\) must be long enough to ensure proper ping-pong enabling of the latch controller.

As we increase the length of wire between the two pipeline stages, the forward and reverse delays of the FIFO stage increases. This is because it includes the delay of many buffered wire stages. Thus, the throughput is still limited by the delay of a FIFO stage.

### 3.4 Novel FIFO 2: Interleaved Control

As shown in Equation (3.12), throughput is still limited by the sum of worst-case forward and reverse-delay of a FIFO stage. We can further improve upon the novel buffered control scheme by introducing multiple-control paths working in parallel. This enables wave-pipelining [10] [11] between the FIFO stages.
Figure 3.7 shows this multiple-control scheme with buffered handshake wires. Only three control stages are shown in the figure. In the multiple-control scheme, each FIFO stage consists of a number of control paths between the two pipeline stages. Each FIFO stage consists of a series of GasP [40] circuits and $SR$ latches. These GasP circuits work in a round-robin fashion controlled by the $SR$ latches. The circuit has to be properly initialized to ensure correct operation. The data path consist of latches with tri-state buffers. We need to use tri-state latches for the same reasons as in the twin-control scheme. The degree of multiple control, i.e., $k$, is defined as the number of controls working in parallel in a FIFO stage. If the degree of multiple control is high, we might need to use the latches followed by a multiplexer instead of using tri-state buffers. Note that as we increase $k$, this increases the load on the data path. Appendix B examines this trade-off in more detail.

After control path $j-1$ has sent its data and is waiting for the acknowledgment of its previous token, control path $j$ can send its data token (provided the multiple data waves from transactions on control path $j-1$ and control path $j$ do not overlap with each other), and likewise for the other control paths.

Thus, we have multiple waves traveling in the path between the FIFO stages. This is known as wave-pipelining. With the introduction of wave-pipelining, we do not need to have delay matched alternation wires (as required in twin-control scheme [30]). Instead, we use smaller delays that ensure there is enough time separation between the multiple waves. This is ensured by the inherent delays of $SR$ latches and the series of control buffers.

The forward and reverse delays of the FIFO increase linearly with the delay per pipeline stage. As we increase the delay per pipeline stage, the throughput is still limited by the sum of the forward and reverse delays of a FIFO stage. But unlike twin-control, we can increase the degree of multiple control, $k$, thus achieving the required
Figure 3.7: Multiple control with buffered handshake wire

As we increase the degree of multiple control, the load on the data path increases
as well. For large value of $k$, the increased value of latch delay $\delta_{dq}^{\max}$ will require a corresponding change to the control, i.e., increase $\delta_{f,rep}^{\max}$ which ultimately limits the throughput and latency of the link.

A further constraint on the clock period $P$ is as follows; it must be large enough to ensure that there is enough separation between successive data waves traveling on a pipeline between two FIFO stages, i.e., the clock period must be larger than the worst case difference between the maximum and minimum case path delay of a pipeline between two FIFO stages.

For each FIFO stage, we have a ring of control circuits. There is a delay from the instant a control circuit in this ring is triggered on a clock edge to the instant the next control circuit in the stage is triggered on next clock edge. The clock period must be larger than this delay between triggering of successive control stages to ensure proper operation of the local round-robin enabling in the latch controller.

### 3.4.1 Latency of Interleaved and Distributed FIFO

For the analysis of the FIFO, a similar notation is used as in Section 3.2.1:

- $\phi(i)$ to denotes the event of the $i^{th}$ rising edge of $\phi$.

- $P$ is the time-period of the clock, where $P_{\text{max}}$ and $P_{\text{min}}$ denote the maximum and minimum case clock period.

- $\delta_{t}$ and $\delta_{r}$ represent, the delay of the forwarded clock and the local receiver clock from the common clock point, respectively.

- $\lambda$ and $\Lambda$ denote the latency of the FIFO as defined in Section 3.2.1.

- $\delta_{f,rep}$ and $\delta_{r,rep}$ represent the forward and reverse delay of the FIFO with buffered control wires, respectively. These FIFO delays include the delay of long buffered
control wires\(^6\).

- \(k\) is the degree of multiple control.

**Minimum FIFO Latency:** The minimum latency of the FIFO corresponds to the time for a value to propagate unobstructed from the first stage of the FIFO to the last stage, with \(\delta_{f,rep}^{\text{max}}\) time units of set-up time for the receiver. Thus, in a FIFO with \(n_{\text{fifo}}\) stages, \(n_{\text{fifo}}\) transfers occur between output of a value by a transmitter and the arrival of that value at \(U_{n_{\text{fifo}}+1}\), the receiver latch and \(t_{\text{set-up}}\) time is required to satisfy the set-up requirement of \(U_{n_{\text{fifo}}+1}\). Thus, the worst-case minimum latency \((\lambda_{\text{min}})\) of the FIFO is given as follows:

\[
\lambda_{\text{min}} = n_{\text{fifo}}\delta_{f,rep}^{\text{max}} + t_{\text{set-up}}
\]  

\((3.14)\)

**Set-up constraint at the receiver:** If a FIFO has \(n_{\text{fifo}}\) stages, the set-up constraint at the receiver is given by:

\[
\begin{align*}
\delta_{t}^{\text{max}} + n_{\text{fifo}}\delta_{f,rep}^{\text{max}} + t_{\text{set-up}} & \leq \delta_{r}^{\text{min}} + \lambda P_{\text{min}} \\
\delta_{r}^{\text{min}} - \delta_{t}^{\text{max}} + \lambda P_{\text{min}} & \geq n_{\text{fifo}}\delta_{f,rep}^{\text{max}} + t_{\text{set-up}}
\end{align*}
\]  

\((3.15)\)

Thus, the following condition must be satisfied for proper operation of the FIFO:

\[
\delta_{r}^{\text{min}} - \delta_{t}^{\text{max}} + \lambda P_{\text{min}} \geq \lambda_{\text{min}}
\]  

\((3.16)\)

**Maximum FIFO latency:** The maximum latency of the FIFO corresponds to the time for a request to propagate from the first stage of the FIFO to the last stage, when each FIFO stage is waiting for the acknowledgement of its previous request token, with \(\delta_{r,rep}^{\text{max}}\) time units to spare. If \(\delta_{r,rep}^{\text{max}}\) is the reverse delay of FIFO and each new request

---

\(^6\)We use max and min superscript to denote the maximum and minimum value of the delays of a FIFO stage.
occurs once every $kP_{min}$ time units, the maximum delay between the arrival of a request signal at stage $i$ and its transfer to stage $i+1$ is given by $(kP_{min} - \delta_{r,rep}^{\text{max}})$. The latency of the FIFO is given by:

$$\lambda_{\text{max}} = n_{fifo}(kP_{min} - \delta_{r,rep}^{\text{max}}) + (P_{min} - t_{\text{hold}})$$  \hspace{1cm} (3.17)

**Hold constraint at the first stage of FIFO:** The hold requirement at the first stage of the FIFO, i.e., latch $U_1$ is satisfied if:

$$P_{min} + \delta_{t,\text{min}} - t_{\text{hold}} \geq \delta_{r,\text{max}} + \Lambda P_{\text{max}} - n_{fifo}(kP_{min} - \delta_{r,\text{rep}}^{\text{max}})$$

$$\delta_{r,\text{max}} - \delta_{t,\text{min}} + \Lambda P_{\text{max}} \leq n_{fifo}(kP_{min} - \delta_{r,\text{rep}}^{\text{max}}) + P_{min} - t_{\text{hold}}$$  \hspace{1cm} (3.18)

Thus, the following condition must be satisfied for proper operation of the FIFO:

$$\delta_{r,\text{max}} - \delta_{t,\text{min}} + \Lambda P_{\text{max}} \leq \lambda_{\text{max}}$$  \hspace{1cm} (3.19)

Skew tolerance, $\sigma$, provided by FIFO is given by the difference between the maximum and minimum latency.

$$\sigma = \lambda_{\text{max}} - \lambda_{\text{min}}$$

$$\sigma = n_{fifo}(kP_{min} - \delta_{r,\text{rep}}^{\text{max}} - \delta_{r,\text{rep}}^{\text{max}}) + (P_{min} - t_{\text{hold}} - t_{\text{set-up}})$$  \hspace{1cm} (3.20)

For a given FIFO depth $n_{fifo}$, forward delay and reverse delay of the FIFO, the skew tolerance is higher compared to that of linear FIFO (Equation 3.11) of same depth. We effectively have a square FIFO [43] with interconnect distributed between FIFO stages. Thus we can have increased skew tolerance by increasing the degree of multiple control. However, the increase in degree of multiple control is limited by the constraints discussed in Section 3.4.
Based on the latency vs. clock-period trade-offs presented in this chapter, the next chapter compare the performance of various synchronous interconnects, namely, two-phase and multi-phase with a source-synchronous interconnect implemented using the interleaved and distributed FIFO.
Chapter 4

Comparison of Interconnect Schemes

This chapter presents a number of comparisons of synchronous and source-synchronous interconnect techniques using throughput, velocity and power as performance metrics. The velocity metric is a new one that captures the latency aspect of pipelining. We first present details of various circuit components and deviations in their timing properties as a result of PVT variations. Designs for two-phase synchronous, multi-phase synchronous and source-synchronous interconnect with lumped and distributed FIFOs are then presented. Finally comparisons are provided using the three metrics.

4.1 Circuit Components Variations

The circuit components used in interconnect pipelining are latches, buffers, wire segments, tracking delays and FIFOs. In this section, we present various parameters affecting the timing of these components. They consist of process, supply voltage, temperature and cross-coupling variations. These variations cause deviations in delays of these com-
ponents and appear as overhead in various schemes for driving long wire interconnects. The overhead can be in form of wire delay variations, buffer delay variations, clock skew and jitter, or unmatched data and clock path delays.

Figure 4.1 shows a typical pipelined link. The long wire segments between consecutive registers are broken into smaller pieces using buffers or inverters to reduce the total delay. In this figure, \( \delta_{buf} \) denotes the delay of a buffer (inverter), \( \delta_w \) denotes delay of a wire segment, and \( \delta_{dq} \) denote the data-to-output delay of a register (transparent latch). Likewise, \( l_w \) denotes the length of a wire, and \( n_{buf} \) denotes the number of buffers between two consecutive registers.

![Figure 4.1: A pipelined link](image)

The parameters in Table 4.1 give delays of various circuit components shown in Figure 4.1. These parameters are based on the TSMC 0.18\( \mu \)m technology at typical process (TT), supply voltage (1.8v) and temperature conditions (\( 25^\circ \)C). Changing the values of these parameters will result in degradation or improvement in the performance of the various interconnect schemes.

The \( \delta_{buf} \) and \( \delta_w \) gives the delay of the buffer and wire, for a sequence of wire segments and buffers, sized for optimal repeater insertion. The buffer sizes and optimal length wire segments used in this work are given in Appendix A-1. We simulated the string of optimally sized wires (1346\( \mu \)m) and inverters (64X) at typical process (TT), supply voltage (1.8v) and temperature (\( 25^\circ \)C) conditions. The values in Table 4.1 reports these delays, where each delay parameter is average of the delay of the rising and the falling delays. The latch characteristics, \( t_{set-up} \), \( t_{hold} \) and \( \delta_{dq} \) are the set-up time, hold time and
latch delay respectively. The transistor sizing and delay characterization of the latch are given in Appendix A-2 and Appendix A-3 respectively.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay Latch + wire load</td>
<td>δ_{dq} + δ_{w}</td>
<td>168 ps</td>
</tr>
<tr>
<td>Delay inverter + wire load</td>
<td>δ_{buf} + δ_{w}</td>
<td>137 ps</td>
</tr>
<tr>
<td>Set-up time</td>
<td>t_{set-up}</td>
<td>67 ps</td>
</tr>
<tr>
<td>Hold time</td>
<td>t_{hold}</td>
<td>70 ps</td>
</tr>
</tbody>
</table>

**Table 4.1:** Delays of various circuit components at typical process corner

The throughput and velocity of the interconnects are affected by variations in the circuit components. The major sources of variations are differences in the fabrication conditions and environmental variations including supply voltage variations and temperature gradients. Run-to-run and die-to-die process variations can have significant effect on the values listed in Table 4.1.

The most significant device parameters affecting gate delays are effective channel length ($L_{eff}$), oxide thickness ($t_{ox}$), threshold voltage ($v_{th}$) and device dimensions ($W/L$) [44]. In the case of wires, the process parameters responsible for the variations are wire dimensions and dielectric thickness variations. The most common method of analyzing the implications of these inter-die process variations is the worst-case and best-case analysis of the circuit. Similarly, corner-case simulation of various circuit blocks can be performed for supply voltage and temperature variations.

Table 4.2 lists percentage variations from ideal delays for wires ($δ_{w}$), latches ($δ_{dq}$) and buffers ($δ_{buf}$) due to PVT variations. These values were obtained by simulating the circuit components in Hspice at various process corners, supply voltage and temperature conditions. Positive values represent a slow down from the ideal case delays and negative values represent a speed up from the typical delay values.

Table 4.3 provides peak-to-peak skew and jitter budgets typically used in the designs [28] and variations due to cross-coupling. We perform a brief study of crosstalk effects
Chapter 4. Comparison of Interconnect Schemes

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \delta_{dq} )</th>
<th>( \delta_{buf} )</th>
<th>( \delta_w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of volt. var.(2V/1.6V)</td>
<td>(-8% / +9%)</td>
<td>(-10% / +14%)</td>
<td>-</td>
</tr>
<tr>
<td>Range of temp. var.(10C/100C)</td>
<td>(-4% / +17%)</td>
<td>(-5% / +13%)</td>
<td>(-12% / +21%)</td>
</tr>
<tr>
<td>Range of process var.(FF/SS)</td>
<td>(-26% / +35%)</td>
<td>(-18% / +26%)</td>
<td>(-15% / +15%)</td>
</tr>
</tbody>
</table>

Table 4.2: Effect of PVT variations

in TSMC 0.18\( \mu \)m technology in Appendix A-4. We also study a way to reduce crosstalk, using staggered repeaters. Row 3 in Table 4.3 reports the variations in signal delay due to the crosstalk, as studied in Appendix A-4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock skew</td>
<td>( t_{sk} )</td>
<td>10%</td>
</tr>
<tr>
<td>Clock jitter</td>
<td>( t_j )</td>
<td>2%</td>
</tr>
<tr>
<td>Cross coupling var.</td>
<td>( \sigma_c )</td>
<td>(+37% / -5%)</td>
</tr>
</tbody>
</table>

Table 4.3: Skew, jitter and crosstalk budget

The worst-case latch delay with wire load and buffer delay with wire load after including the slow down due to worst-case process, supply voltage, temperature and crosstalk conditions are as given in Table 4.4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay latch + wire load</td>
<td>( \delta_{dq} + \delta_w )</td>
</tr>
<tr>
<td>Delay inverter + wire load</td>
<td>( \delta_{buf} + \delta_w )</td>
</tr>
</tbody>
</table>

Table 4.4: Worst-case delays of various circuit components

As shown in Figure 2.8, the structure of the data and strobe paths for source-synchronous interconnects are very similar, each consisting of series of buffered wire stages. Moreover, the data and strobe paths are close together on the chip, thus, we can have very good delay tracking between the two. With careful layout and design, we can ensure good process tracking between strobe and data paths [39]. Similarly, temperature variations across the chip do not cause major tracking variations, because of the close proximity of the paths. Supply voltage variations are the major concern, as
the voltage can change quite a bit over a single clock cycle. With ±10%\(^1\) variations in the supply voltage, we can get worst-case tracking of 20%.

In this work, we make reasonable assumptions about the tracking variations and assume that combined effects are additive. We use tracking variations of 40% [39] with process and temperature variations each accounting for 10% of the margin, while the voltage variations account for the rest. We also consider an aggressive\(^2\) implementation of the source-synchronous design with a tracking margin of 20%. Table 4.5 gives these assumed PVT mismatch variations.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Percentage variations</th>
<th>Percentage variations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>best case tracking</td>
<td>worst case tracking</td>
</tr>
<tr>
<td>Voltage variations</td>
<td>10%</td>
<td>20%</td>
</tr>
<tr>
<td>Temperature variations</td>
<td>5%</td>
<td>10%</td>
</tr>
<tr>
<td>Process variations</td>
<td>5%</td>
<td>10%</td>
</tr>
<tr>
<td>Total variations</td>
<td>20%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Table 4.5: Source-synchronous tracking margins

### 4.2 The Velocity Metric

Typically, latency is used as a performance metric for pipelines. However, it depends on the length of the pipeline. We need to have a normalized metric that captures the latency component. We derive a new metric called velocity, related to inverse of the latency. The advantage of using the normalized metric is that; we can generalize our comparisons of the various techniques of interconnect pipelining. Thus, given the throughput and velocity target for a point-to-point link, we can find the best technique of interconnect pipelining independent of actual length of the interconnect. This helps

\(^1\)Variations of ±10% in a single cycle is unlikely, but in the absence of validated data we use worst case values.

\(^2\)With careful design and layout we can have very good tracking between the data and the strobe path. Thus, considerably reducing tracking variations due to PVT variations.
to integrate our comparisons in a flow for interconnect synthesis, as we do not need to do know the routing information in advance.

The velocity of a link is the rate at which data propagates (e.g. in meters/second)\(^3\). It provides a normalized measure of the performance of the interconnects. The performance is no longer dependent on the length of the interconnect. For the link shown in Figure 4.1, the velocity is:

\[
v = \frac{n_{buf} \cdot l_w}{(\delta_{dq} + \delta_w) + n_{buf}(\delta_{buf} + \delta_w)}
\]

Substituting the values for latch \((\delta_{dq})\), inverter \((\delta_{buf})\) and wire delay \((\delta_w)\) from Table 4.4 and optimal wire length \((l_w)\) of 1346\(\mu\)m, for a very long pipeline \((n_{buf} \gg)\) consisting of large number of optimal sized buffered wire segments the velocity approaches an asymptotic value of 6.0mm/ns.

### 4.3 Two-Phase Clocking

In this section, we determine the velocity vs. throughput characteristics for a two-phase synchronous design. In the subsequent sections, we do the same for multi-phase synchronous and source-synchronous with lumped and distributed FIFOs. They provide the basis for the comparison that we present in Section 4.6.

Two-phase clocking is a simple and conservative implementation of synchronous interconnects. Because we can examine a practical implementation, it establishes a lower bound on the performance of the synchronous approach. Figure 4.2 shows a pipeline stage for two-phase synchronous interconnect. We have a global clock network that supplies a clock with guaranteed skew and jitter bounds. Each pipeline stage consist of alternating active-low and active-high latches, with \((n_{buf} + 1)\) buffered optimal length

\(^3\)In the limit that the length of link becomes very large.
interconnect segments distributed between the two pipeline stages. The equations for optimal wire segment \((l_w)\) and repeater sizes used in this work are given in Appendix A-1.

**Figure 4.2:** Two-phase synchronous interconnect

As shown in Figure 4.3, each pipeline stage has half a clock cycle to propagate data, including the latch delay. We also make use of the time borrowing property of latches, i.e., if the data arrives at the input of the latch in the interval that clock is high, we do not have to pay for skew and set-up overheads at each pipeline stage. The skew and set-up window is amortized over all of the pipeline stages. For optimal placement of latches, the data arrives at the latch input when the latch is transparent, i.e., the data ripples through the series of latches and pipeline stages. Thus, overheads due to set-up and clock uncertainty are paid at only the last pipeline stage.

For a large number of pipeline stages, the set-up and skew window overheads become
negligible, and throughput is limited by the maximum case path delay between two pipeline stages. Thus, given a target throughput, we have to ensure that the path delay between alternate latches is at most half a clock cycle. For a high target throughput, we have to reduce the delay between consecutive pipeline stages. This is achieved by reducing the number of buffered wire segments \((n_{buf} + 1)\) between two pipeline stages, which increases the number of pipeline stages.

The highest throughput is achieved for fine-grain pipelining which occurs, when each pipeline stage consists of a single wire segment. For very high throughput targets, it may not be possible to cross an optimal sized wire segment in half a clock cycle. In that case, we must reduce the size of wire segments between pipeline stages. However, the use of sub-optimal sized wire segments severely degrades the velocity of the interconnect.

We study the common clock scheme for varying granularities of the pipelined interconnect. In particular, the number of buffered wire segments per pipeline stage, \((n_{buf} + 1)\), is varied from eight to one. Throughput in this scheme depends on the maximum case path delay between two pipeline stages. The maximum case path delay between two pipeline stages is given by sum of latch delays, buffer delay, wire delay and, set-up and clock overheads divided by number of pipeline stages. Thus, for each of the above mentioned case, we perform corner case simulation in SPICE, at the SS (slow NMOS, slow PMOS) process corner with the worst case supply voltage (1.6V) and temperature (100°C) conditions. For each simulation, we measure the maximum possible throughput and the corresponding velocity of the pipelined interconnect.

Table 4.6 shows the performance of two-phase clocking. As we decrease the delay between two pipeline stages, i.e., decrease \(n_{buf}\), we obtain increasing throughput while the total pipeline overhead increases thus the decrease in the velocity. The first four rows in the table represent the designs where we use sub-optimal sized wire segments to achieve the target throughput. The use of sub-optimal wire segments degrades the
velocity of the link.

<table>
<thead>
<tr>
<th>Wire segments per pipeline stage ((n_{buf} + 1))</th>
<th>Distance per pipeline stage ((\text{um}))</th>
<th>Delay per pipeline stage (\text{(ps)})</th>
<th>Maximum throughput ((\text{GHz}))</th>
<th>Velocity ((\text{mm/ns}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>600</td>
<td>183</td>
<td>2.72</td>
<td>3.27</td>
</tr>
<tr>
<td>1</td>
<td>850</td>
<td>207</td>
<td>2.41</td>
<td>4.09</td>
</tr>
<tr>
<td>1</td>
<td>1170</td>
<td>250</td>
<td>1.99</td>
<td>4.66</td>
</tr>
<tr>
<td>1</td>
<td>1346</td>
<td>278</td>
<td>1.79</td>
<td>4.83</td>
</tr>
<tr>
<td>2</td>
<td>2692</td>
<td>501</td>
<td>1.00</td>
<td>5.36</td>
</tr>
<tr>
<td>3</td>
<td>4038</td>
<td>725</td>
<td>0.69</td>
<td>5.57</td>
</tr>
<tr>
<td>4</td>
<td>5384</td>
<td>948</td>
<td>0.53</td>
<td>5.68</td>
</tr>
<tr>
<td>5</td>
<td>6730</td>
<td>1171</td>
<td>0.43</td>
<td>5.74</td>
</tr>
<tr>
<td>6</td>
<td>8076</td>
<td>1384</td>
<td>0.36</td>
<td>5.79</td>
</tr>
<tr>
<td>7</td>
<td>9422</td>
<td>1617</td>
<td>0.31</td>
<td>5.82</td>
</tr>
<tr>
<td>8</td>
<td>10768</td>
<td>1841</td>
<td>0.27</td>
<td>5.85</td>
</tr>
</tbody>
</table>

Table 4.6: Two-phase clocking.

4.4 Multi-Phase Clocking

With multi-phase clocking as shown in Figure 4.4, we assume that any phase of the clock is available at any point on the chip. We also make use of wave-pipelining. These assumptions allow us to obtain an upper bound on the performance of synchronous methods with respect to other techniques. The multi-phase approach described in this section provides a valid upper bound on synchronous performance for the comparison. We note that higher performance may be possible by increasing wire pitch or using other latch designs. The parameters are fixed across all the designs that are considered to obtain a fair comparison.

Figure 4.4 shows our multi-phase, pipelined interconnect. It uses a global clock with constructive skew to provide any required clock phase at any point on the chip. It uses latches for the pipelining in order to exploit time borrowing. It also makes use of wave-pipelining. Each pipeline stage consists of \((n_{buf} + 1)\) buffered wire segments. The delay
characteristics of wire segments, latches and buffers are the same as given in Section 4.1.

![Multi-phase clocking](image)

**Figure 4.4:** Multi-phase clocking

Ordinary pipeline systems operate at a frequency that corresponds to the maximum logic path delay between any two stages. Rather than allowing data to propagate from a register through the combinational network to another register prior to initiating the subsequent data transfer, wave-pipelined designs apply subsequent data to the network as soon as it can be guaranteed that it will not interfere with the current data wave. The primary causes of data wave interference are the variation in the propagation delay due to difference in fabrication and environmental conditions. In wave-pipelined circuits, a wave must be set-up at the output register and the subsequent wave must not be less than a hold time away from the register when the register is clocked. Thus, the throughput depends on the worst-case difference between the delays of successive data waves. The double-sided timing constraints in multi-phase pipelining results in designs that operate at one frequency but might not operate at lower frequencies.

Figure 4.5 shows the timing constraints for the multi-phase interconnect with wave-pipelining. $T_L$ is the time from launch of data at stage $i$ to its capture at stage $i + 1$. To ensure proper operation of the multi-phase design, we have to ensure that the set-up and hold condition is satisfied at each latch. There is a trade-off between set-up and hold time due to the time borrowing property of latches. The maximum time borrowing along a sequential path is determined by $T_{high}$ ($T_{high}$ is the length of the transparent period of the clock signal). Better time borrowing can be obtained by widening $T_{high}$. 
However, if the transparency window is increased, the minimum path delay (i.e., hold time) constraint becomes harder to meet. We can increase $T_{\text{high}}$ and get some slack for the set-up which allows us to increase the operating frequency. We can continue doing this until the hold-constraint becomes tight. Thus, at the optimum point we get maximum throughput when both set-up and hold constraints are exactly satisfied (with no room to spare).

The throughput of multi-phase synchronous interconnect depends on the sum of worst case difference between the maximum and minimum path delays between two pipeline stages, the set-up and hold windows of the latches and twice the global peak-to-peak skew and jitter\textsuperscript{4}.

Table 4.7 gives the performance of multi-phase synchronous interconnect for various numbers of buffers between latches. The throughput for a given number of wire segments per pipeline stage ($n_{\text{buf}} + 1$) is given by the maximum variations in the path delay between two latches. The multi-phase interconnect is designed to operate at a particular frequency. Thus, we have to take the worst-case difference between maximum and minimum delay of the data path, at extreme process corners, to ensure that devices from different fabrication runs and different dies on same wafer will operate properly.

\textsuperscript{4}We have to account for the skew and jitter of each clock edge - one edge corresponds to the set-up constraint and the other corresponds to the hold constraint.
Chapter 4. Comparison of Interconnect Schemes

The maximum case path delay for a pipeline stage is measured at the SS process corner along with worst case temperature (100°C) and voltage (1.6V) conditions, while the minimum case path delay is measured at the FF process corner along with best case temperature of 0°C and supply voltage of 2.0V. The corresponding velocity for a given number of wire segments per pipeline stage \((n_{buf} + 1)\) is measured at the SS process corner with a worst case supply voltage (1.6V) and temperature (100°C) conditions, while the pipeline is operating at the throughput determined earlier.

<table>
<thead>
<tr>
<th>Wire segments per pipeline stage ((n_{buf} + 1))</th>
<th>Distance per pipeline stage ((\mu m))</th>
<th>Delay per pipeline stage ((ps))</th>
<th>Maximum throughput ((GHz))</th>
<th>Velocity ((mm/ns))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1346</td>
<td>278</td>
<td>2.80</td>
<td>4.83</td>
</tr>
<tr>
<td>2</td>
<td>2692</td>
<td>501</td>
<td>1.94</td>
<td>5.36</td>
</tr>
<tr>
<td>3</td>
<td>4038</td>
<td>725</td>
<td>1.50</td>
<td>5.57</td>
</tr>
<tr>
<td>4</td>
<td>5384</td>
<td>948</td>
<td>1.21</td>
<td>5.68</td>
</tr>
<tr>
<td>5</td>
<td>6730</td>
<td>1171</td>
<td>1.01</td>
<td>5.74</td>
</tr>
<tr>
<td>7</td>
<td>9422</td>
<td>1617</td>
<td>0.76</td>
<td>5.82</td>
</tr>
<tr>
<td>8</td>
<td>10768</td>
<td>1841</td>
<td>0.68</td>
<td>5.85</td>
</tr>
<tr>
<td>9</td>
<td>12114</td>
<td>2064</td>
<td>0.61</td>
<td>5.87</td>
</tr>
<tr>
<td>11</td>
<td>14806</td>
<td>2510</td>
<td>0.51</td>
<td>5.90</td>
</tr>
<tr>
<td>15</td>
<td>20190</td>
<td>3403</td>
<td>0.38</td>
<td>5.93</td>
</tr>
<tr>
<td>25</td>
<td>33650</td>
<td>5635</td>
<td>0.24</td>
<td>5.97</td>
</tr>
</tbody>
</table>

*Table 4.7: Multi-phase clocking*

### 4.5 Source-Synchronous with the Novel FIFO

The source-synchronous interconnect consists of forwarded pipeline stages and a FIFO at the end to compensate for the skew in the forwarded path. In this section we present analysis flows for design of forward path, FIFO design and FIFO initialization.
4.5.1 Forwarded Clock Path

We now describe the design flow for designing the data path and the strobe path of a source-synchronous interconnect. Given \( n_{buf,pipe} \) buffers per pipeline stage, the latch setup and hold characteristics and tracking variations between the data and clock path, we need to determine the maximum possible throughput while satisfying the timing constraints at the latch. We also need to determine the velocity of a forwarded pipeline stage, \( V_{pipe} \), and skew per pipeline stage in the forwarded path \( \sigma_{pipe} \). The flowchart in Figure 4.6 gives the design flow for the forwarded pipeline stages.

Source-synchronous designs give high throughput with high skew tolerance and low latency. As discussed in Section 2.5, source-synchronous clocking does not use the global clock along the long interconnect. Instead, the circuit that is producing the data pattern creates its own clock from its local version of the global clock. This clock is forwarded along with the data.

In a typical synchronous design, new data are produced at each rising edge (edges \( a \) and \( b \) as shown in Figure 4.7) of the global clock. If we forward this global clock, the clock forwarding network consumes power for the falling clock edges even though no data transfer occur with these events. We can save power by using a forwarded clock with period twice that of the global clock, and using both clock edges (edges \( c \) and \( d \) as shown in Figure 4.7) of the forwarded clock to latch the data.

Figure 4.8 shows the details of the pipeline stages of the source-synchronous interconnect. Each pipeline stage consists of a data path and a clock path. The data path consists of transparent latches separated by \((n_{buf,pipe} + 1)\) buffered optimal length interconnect segments. The optimal wire segment \((l_w)\) and repeater sizes used in this work are given in Appendix A-1. The latches are triggered by the pulses, produced by the edge detector circuits, on both edges of the forwarded clock.
Chapter 4. Comparison of Interconnect Schemes

Figure 4.6: Analysis flow for designing the forwarded pipeline stages
Figure 4.7: Forwarded clock derived from global clock

Figure 4.8: Source-synchronous: forwarded clock and data path

The clock path consists of matching buffers separated by \((n_{buf,pipe} + 1)\) buffered wire segments. The matching buffers are designed to match the delay of the latches over the full range PVT variations. Likewise, the buffered interconnect segments match the delay of the corresponding interconnect structure in the data path. We must ensure that the set-up constraint is satisfied at each latch. This is ensured by designing the clock path delay to be larger than that of the data path. For example in a design with a 20\% margin in the tracking between the data and the clock path, we need to ensure that the data arrives at least \(t_{set-up}\) earlier than the corresponding clock at the latch. However, due to tracking variations, the clock can be maximally advanced through the pipeline stage while the data path is maximally retarded, giving rise to possibility of set-up time violation. We design with a tracking margin of 20\%, i.e., we make clock path delay to be 1.2 times the data path delay so that we satisfy the set-up constraint over range of PVT variations. The throughput is given by worst case difference between the strobe and data path delays.

As seen in Figure 4.9, a falling clock edge has an extra delay of one inverter. Thus,
the margin between data and clock path differs by a gate delay for the falling and rising clock edges. However, time borrowing helps us here as we can trade-off between $T_{\text{high}}$ (the width of the pulse produced by the edge detector circuit) and the tolerance of jitter arising from the edge detector. However, increasing the width of the clock pulses can reduce the throughput of the source-synchronous interconnect, because the hold constraint becomes difficult to satisfy as transparency window becomes very wide. The optimal value of $T_{\text{high}}$ occurs when it is just wide enough to satisfy the set-up constraint at the latch in the presence of the worst-case jitter.

As described in Chapter 2, absolute path delays are not critical in the operation of source-synchronous interconnect. The throughput of source-synchronous interconnect depends on the difference in the delays between the data and clock path. There is no global clock; instead, a local clock is used that is fed through the pipeline, i.e., data is associated with an event on the local clock. Several data and protocol events can propagate in the pipelined interconnect simultaneously. The source-synchronous interconnect replaces the constructive skew in multi-phase synchronous designs with a worst-case delay that tracks the data path delay over all range of PVT variations. This reduces the double-sided timing constraints, i.e., in contrast to multi-phase synchronous interconnect this design will operate at any clock rate lower than the one for which it is

\textbf{Figure 4.9:} Edge detector generate pulses on both clock edges
designed.

It is crucial that the data path and clock path delays track each other over all PVT variations. As discussed in Section 4.1, we consider two designs: one, using a conservative tracking margin of 40% and, second, an aggressive design with a tracking margin of 20%. For each design, we simulate the pipeline model at SS (slow NMOS, slow PMOS) corner along with worst-case supply voltage (1.6v) and temperature conditions (100°C). We measure the minimum possible clock period $P$ and velocity of a pipeline stage $V_{pipe}$.

The forwarded clock suffers from skew due to PVT variations along the forwarded clock path. The skew in the forwarded pipeline stage depends on the number of buffered wire stages in a pipeline stage. The skew increases as we increase the buffered wire stages per pipeline stage. We write $\sigma_{pipe}(n_{buf,pipe})$ for skew per pipeline stage between the forwarded clock and the local clock at the receiver, given $n_{buf,pipe}$ buffers per pipeline stage. We performed corner case simulations of the forwarded pipeline stage for all the PVT variables. We simulated the forwarded pipeline stage at SS process corner along with worst-case supply voltage (1.6V) and temperature conditions (100°C), to obtain the maximum case delay of the forwarded pipeline stage ($\delta^\text{max}_{t,pipe}$). Similarly, the best-case simulation at FF process corner along with best-case supply voltage of 2.0V and temperature of 0°C gives the minimum case delay ($\delta^\text{min}_{t,pipe}$). The drift in the local clock at the receiver end is due to global skew and jitter. Thus, we can find $\delta^\text{max}_{t,pipe}$ and $\delta^\text{min}_{t,pipe}$ given the global skew and jitter bounds. Given these values for variations in the delay of the forwarded and the local receiver clock, we can find the skew $\sigma_{pipe}(n_{buf,pipe}) = (\delta^\text{min}_{t,pipe} - \delta^\text{min}_{t,pipe}) + (\delta^\text{max}_{t,pipe} - \delta^\text{min}_{t,pipe})$ between the forwarded and the local receiver clock.

Table 4.8 and Table 4.9 give the minimum possible clock period $P$, velocity of a pipeline stage $V_{pipe}$ and $\sigma_{pipe}$, the skew per pipeline stage for different number of buffers ($n_{buf,pipe}$) per pipeline stage, for two design options: one, with a tracking margin of 20%
and other with a tracking margin of 40%.

<table>
<thead>
<tr>
<th>$(n_{buf,pipe} + 1)$</th>
<th>Distance per pipeline stage (um)</th>
<th>Delay per pipeline stage (ps)</th>
<th>Maximum throughput (GHz)</th>
<th>$V_{pipe}$ (mm/ns)</th>
<th>$\sigma_{pipe}(n_{buf,pipe})$ (ps/stage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2692</td>
<td>870</td>
<td>2.70</td>
<td>4.64</td>
<td>518</td>
</tr>
<tr>
<td>5</td>
<td>5384</td>
<td>1405</td>
<td>2.15</td>
<td>4.79</td>
<td>832</td>
</tr>
<tr>
<td>7</td>
<td>8076</td>
<td>1941</td>
<td>1.69</td>
<td>4.85</td>
<td>1145</td>
</tr>
<tr>
<td>11</td>
<td>13460</td>
<td>3012</td>
<td>1.19</td>
<td>4.91</td>
<td>1723</td>
</tr>
<tr>
<td>15</td>
<td>18844</td>
<td>4084</td>
<td>0.92</td>
<td>4.94</td>
<td>2400</td>
</tr>
<tr>
<td>25</td>
<td>32304</td>
<td>6762</td>
<td>0.58</td>
<td>4.98</td>
<td>3969</td>
</tr>
<tr>
<td>45</td>
<td>59224</td>
<td>12119</td>
<td>0.34</td>
<td>5.00</td>
<td>7107</td>
</tr>
</tbody>
</table>

**Table 4.8:** Clock period, pipeline velocity and skew per pipeline stage for tracking margin of 20%

<table>
<thead>
<tr>
<th>$(n_{buf,pipe} + 1)$</th>
<th>Distance per pipeline stage (um)</th>
<th>Delay per pipeline stage (ps)</th>
<th>Maximum throughput (GHz)</th>
<th>$V_{pipe}$ (mm/ns)</th>
<th>$\sigma_{pipe}(n_{buf,pipe})$ (ps/stage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1346</td>
<td>390</td>
<td>2.70</td>
<td>3.45</td>
<td>238</td>
</tr>
<tr>
<td>3</td>
<td>2692</td>
<td>1015</td>
<td>2.26</td>
<td>3.98</td>
<td>604</td>
</tr>
<tr>
<td>5</td>
<td>5384</td>
<td>1640</td>
<td>1.59</td>
<td>4.10</td>
<td>970</td>
</tr>
<tr>
<td>7</td>
<td>8076</td>
<td>2265</td>
<td>1.22</td>
<td>4.16</td>
<td>1337</td>
</tr>
<tr>
<td>11</td>
<td>13460</td>
<td>3515</td>
<td>0.84</td>
<td>4.21</td>
<td>2069</td>
</tr>
<tr>
<td>15</td>
<td>18844</td>
<td>4765</td>
<td>0.64</td>
<td>4.24</td>
<td>2801</td>
</tr>
<tr>
<td>25</td>
<td>32304</td>
<td>7889</td>
<td>0.40</td>
<td>4.26</td>
<td>4631</td>
</tr>
<tr>
<td>45</td>
<td>59224</td>
<td>14139</td>
<td>0.23</td>
<td>4.28</td>
<td>8392</td>
</tr>
</tbody>
</table>

**Table 4.9:** Clock period, pipeline velocity and skew per pipeline stage for tracking margin of 40%

As shown in Table 4.8 and Table 4.9, throughput is limited by the tracking between the data and the strobe path. Thus, the design with tracking margin of 20% gives higher throughput as compared to the design with tracking margin of 40% for same number of buffers per pipeline stage. As we increase the buffered wire segments per pipeline stage we get decreasing throughput because of increased variations in tracking between data and strobe paths. The velocity per pipeline stage, $V_{pipe}$, increases as we increase wire segments per pipeline stage due to reduced pipeline overheads as compared to actual
interconnect delay. The velocity is limited by the delay of strobe path rather than the delay of the data path. For a tracking margin of 40%, we design the strobe path to be 1.4 times the data path delay. Thus, for a very long pipeline consisting of large number of buffered wire segments, the velocity \( V_{pipe} \), approaches the asymptotic value, 0.7 times that of synchronous pipeline of same length.

### 4.5.2 FIFO Design

In this section, we give details of the procedure used for designing the FIFO. We give constraints for the proper operation of the FIFO. We determine the skew tolerance provided by the FIFO. Finally, we study the trade-offs between the various parameters for the FIFO design and give details of procedure used for deciding the optimal FIFO design to obtain maximum velocity for the interconnect.

Since the forwarded clock suffers from skew relative to the local clock due to PVT variations along the forwarded clock path, a distributed and interleaved FIFO is used to compensate for the skew. Design of a forwarded pipeline stage is characterized by clock period \( P \), \( n_{buf,pipe} \) buffers per pipeline stage, \( V_{pipe} \) the velocity per pipeline stage and \( \sigma_{pipe}(n_{buf,pipe}) \) the skew per pipeline stage. The FIFO design is characterized by the degree of interleaving \( k \), the number of buffers per pipeline stage in the FIFO, \( n_{buf,fifo} \), and the velocity of a FIFO stage, \( V_{fifo} \). We need to determine the FIFO design parameters \( (k, n_{buf,fifo}) \) to obtain optimal value for the velocity of the interconnect \( V_{link} \).

The flowchart in Figure 4.10 gives the flow for deciding the optimal FIFO design.

As discussed in Section 3.4.1, the skew tolerance, \( \sigma_{fifo}(P, k, n_{buf,fifo}, n_{fifo}) \), provided by the interleaved and distributed FIFO is given by the difference between the maximum and minimum latency of the FIFO. It is a function of clock period \( P \), buffers per pipeline stage in the FIFO, \( n_{buf,fifo} \), degree of interleaving \( k \) and the number of FIFO stages, \( n_{fifo} \).

Thus, if we have \( n_{fifo} \) stages, the skew tolerance of the FIFO, \( \sigma_{fifo} \), is given as follows:
Chapter 4. Comparison of Interconnect Schemes

Given clock period $P$, skew per pipeline stage $\sigma_{pipe}$, buffers per forwarded stage $n_{buf}$, and velocity of a forwarded pipeline stage $V_{pipe}$, design the FIFO.

1. Given $P$, $k$ and $n_{buf}$, find skew tolerance per FIFO stage, $\sigma_{fifo} = (kP_{min} - \delta_{r,rep} - \delta_{f,rep}) + (P_{min} - t_{set-up} - t_{hold})$.

2. Find velocity per FIFO stage $V_{fifo}$.

3. Determine the maximum number of pipeline stages in the forwarded path, so that skew tolerance provided by the FIFO is adequate to compensate skew in the forwarded path.

4. Find ratio $\alpha$, of the interconnect in the forwarded path to that in the distributed FIFO.

5. Given $V_{fifo}$, $V_{pipe}$ and $\alpha$, find velocity of the link $V_{link}$.

6. Note the values of $k$, $n_{buf}$, $V_{fifo}$, and $V_{link}$ in the result table.

7. Increase $n_{buf}$, buffers per pipeline stage in the FIFO.

\[ \sigma_{fifo} = \lambda_{max} - \lambda_{min} \]
\[ \sigma_{fifo} = n_{fifo}(kP_{min} - \delta_{r,rep} - \delta_{f,rep}) + (P_{min} - t_{set-up} - t_{hold}) \]
\[ \sigma_{fifo} = (kP_{min} - \delta_{r,rep} - \delta_{f,rep}) + (P_{min} - t_{set-up} - t_{hold}) \]
where \( \delta_{f, \text{rep}}^{\text{max}} \) and \( \delta_{r, \text{rep}}^{\text{max}} \) are the maximum case forward and reverse delays of the FIFO, and \( n_{fifo} \) is the number of FIFO stages. We need to determine the skew tolerance per FIFO stage \( \sigma_{fifo}(P, k, n_{buf, fifo}) \), which occurs when \( n_{fifo} = 1 \).

The constraints on clock period, \( P \), are as follows: \( P \) must be large enough to ensure that there is enough separation between successive data waves traveling on a pipeline between two FIFO stages; that is, the clock period must be larger than the worst-case difference between the maximum and minimum path delays of a pipeline between two FIFO stages.

For each FIFO stage, we have a ring of control circuits. There is a delay from the instant a control circuit in this ring is triggered on a clock edge to the instant the next control circuit in the stage is triggered on next clock edge. The clock period, \( P \), must be larger than this delay between triggering of successive control stages to ensure proper operation of the local round-robin enabling in the latch controller.

The clock period, \( P \), is also limited by the sum of the worst-case forward FIFO delay \( \delta_{f, \text{rep}}^{\text{max}} \) and reverse FIFO delay \( \delta_{r, \text{rep}}^{\text{max}} \). Because each internal state transition of a FIFO stage occurs once every \( kP \) time units, the FIFO stage must complete its forward and reverse transaction within this time interval. Equation (4.3) below gives a limit on the clock period \( P \), at which the FIFO can operate:

\[
kP \geq \delta_{f, \text{rep}}^{\text{max}} + \delta_{r, \text{rep}}^{\text{max}}
\]  

(4.3)

As discussed in Section 4.5.1, given the forwarded interconnect with \( n_{buf, \text{pipe}} \) buffers per pipeline stage, the velocity of a forwarded pipeline stage is given by \( V_{\text{pipe}} \). The period \( P \) of the forwarded clock is limited by mismatch between the data and clock paths. We have a skew to be compensated per pipeline stage, given by \( \sigma_{\text{pipe}}(n_{buf, \text{pipe}}) \). We use a

---

5The forward and reverse delays of the FIFO include the delay of interconnect in the distributed FIFO, which is function of \( n_{buf, fifo} \) buffers per pipeline stage in the FIFO.
FIFO to compensate for the skew. Each FIFO design is characterized by $n_{buf,fifo}$ buffers per pipeline stage in a FIFO, $k$, the degree of multiple control and, $V_{fifo}$, the velocity of a FIFO stage. We get a skew tolerance of $\sigma_{fifo}(P, k, n_{buf,fifo})$ per FIFO stage as given in Equation (4.2).

Given various choices of FIFO designs for different values of $(k, n_{buf,fifo})$ and skew requirement per pipeline stage $\sigma_{pipe}(n_{buf,pipe})$, we can determine the maximum number of pipeline stages in the forwarded path so that the skew tolerance $\sigma_{fifo}(P, k, n_{buf,fifo})$ per FIFO stage is enough to compensate for the skew in the forwarded interconnect. This gives us the ratio of interconnect in the forwarded path and that distributed in the FIFO. We write $\alpha l$ to denote the length of interconnect in forwarded path and $(1 - \alpha) l$ to denote the length of interconnect distributed in the FIFO. Thus, the velocity of the link is given by:

$$V_{link} = \frac{V_{fifo} V_{pipe}}{V_{fifo} \alpha + V_{pipe} (1-\alpha)}$$ (4.4)

As an example, we consider the design point in row 2 of Table 4.9 where $n_{buf,pipe} = 2$, i.e., we have three buffered wire segments per pipeline stage in the forwarded path, giving a period of 442ps, velocity $V_{pipe} = 3.98 mm/ns$ for the forwarded pipeline stages and a required skew tolerance of $\sigma_{pipe} = 604 ps$ per pipeline stage. This information is repeated in Table 4.10 for convenience.

<table>
<thead>
<tr>
<th>$n_{buf,pipe}$</th>
<th>$P$ (ps)</th>
<th>$V_{pipe}$ (mm/ns)</th>
<th>$\sigma_{pipe}$ (ps/pipeline stage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>442</td>
<td>3.98</td>
<td>604</td>
</tr>
</tbody>
</table>

Table 4.10: Characteristics forwarded path

Figure 4.11 shows the variations in $V_{fifo}$, the velocity of a FIFO stage for different values of $n_{buf,pipe}$ and $k$, at the given target clock period $P = 442 ps$. For a given value
of $k$, the velocity of FIFO stages increases as we increase the number of buffers ($n_{buf,fifo}$) per pipeline stage. However, for a given degree of interleaving $k$ the number of possible $n_{buf,fifo}$ choices are limited by the constraint that the sum of the forward and reverse delays of the FIFO must not exceed $kP$ (Equation 4.3). For example, at a throughput of 442ps we can have a ($k = 4, n_{buf,fifo} = 1$) FIFO design but we violate the constraint in Equation (4.3) for a ($k = 4, n_{buf,fifo} = 2$) design.

![Velocity of FIFO vs. various choices of FIFO design](image)

**Figure 4.11:** Velocity of the FIFO as a function of $k$ and $n_{buf,fifo}$ ($P = 442ps$)

As we increase $k$, this slows down the data path. The total capacitive load on the data path increases for large values of $k$. This increase in latch delay will require corresponding changes to the control path, i.e., we have to increase the stage delay of the FIFO which limits the throughput and latency of the link. Figure 4.12 shows the effect of increasing $k$ on the worst-case forward delay of a FIFO stage ($n_{buf,fifo} = 0$). This increase in the forward FIFO delay of the FIFO stage explains the dip in the velocity in the Figure 4.11 as we increase $k$. The maximum possible value of $k$ is limited by the constraint that the clock period, $P$, must be larger than the delay between triggering of successive control stages to ensure proper operation of the local round-robin enabling in the latch controller. Thus, at design point $P = 442ps$, we cannot have degree of multiple control greater than 6 (Figure 4.11), as we violate the above condition.
Chapter 4. Comparison of Interconnect Schemes

Figure 4.12: FIFO forward delay vs. $k$

For the design in Table 4.10, we consider the various choices of FIFO designs as given in Table 4.11. Given a skew requirement $\sigma_{\text{pipe}}(n_{\text{buf,pipe}})$ per pipeline stage and skew tolerance provided per FIFO stage $\sigma_{\text{fifo}}(P, k, n_{\text{buf,fifo}})$, we can find the maximum length of interconnect in the forward path for each FIFO design. For example, for the design in Table 4.10, with $(k = 4$ and $n_{\text{buf,fifo}} = 0)$, we have $V_{\text{fifo}} = 1.93\text{mm/\text{ns}}$ and we get a skew tolerance $\sigma_{\text{fifo}} = 1594\text{ps}$ per FIFO stage. Thus, the ratio $\frac{\sigma}{1-\alpha}$ of interconnect in the forwarded path to interconnect distributed in the FIFO is given by $6 \frac{(n_{\text{buf,pipe}}+1)^2 \sigma_{\text{fifo}}(P, k, n_{\text{buf,fifo}})}{(n_{\text{buf,fifo}}+1)^2 \sigma_{\text{pipe}}(n_{\text{buf,pipe}})}$, resulting in $\alpha = 0.72$ and resultant velocity of the link is $3.07\text{mm/\text{ns}}$. 

For a given design, there is trade-off between the degree of multiple control $k$ and buffer stages $n_{\text{buf,fifo}}$ per FIFO stage. As we increase $k$, we get higher skew tolerance and thus the ratio of interconnect in the forward path to that of distributed in the FIFO increases but the FIFO velocity decreases due to increased forward delay of the FIFO stage (Figure 4.12). However, for given $k$ as we increase the buffers per FIFO stage, we get increased velocity per FIFO stage but the skew tolerance per FIFO stage

---

This is valid, when we use optimal sized wire segments in the forwarded pipeline stages and in the pipeline stages distributed in the FIFO.
decreases (because of increase in forward and reverse FIFO delays). Thus, we need to have more FIFO stages to compensate for the skew in the forward path resulting in decrease in ratio $\frac{a}{1-a}$. There is an optimal design point for $(k, n_{buf,fifo})$ at which we get the maximum velocity for the link. We consider various options of FIFO design. As seen from Table 4.11, the most efficient design occurs when $k = 5$ and $n_{buf,fifo} = 2$ giving velocity $V_{link} = 3.52mm/ns$.  


Chapter 4. Comparison of Interconnect Schemes

<table>
<thead>
<tr>
<th>Buffers per FIFO stage</th>
<th>Degree of interleaving k</th>
<th>Velocity of FIFO stage $V_{f_{f_{0}}}$ (mm/ns)</th>
<th>Skew tolerance per FIFO stage $\sigma_{f_{f_{0}}}$ $FIFO_{stage}$</th>
<th>(a)</th>
<th>Velocity of the link $V_{link}$ (mm/ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>1.93</td>
<td>1594</td>
<td>0.72</td>
<td>3.07</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2.66</td>
<td>751</td>
<td>0.55</td>
<td>3.25</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>1.93</td>
<td>2474</td>
<td>0.80</td>
<td>3.28</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2.66</td>
<td>1635</td>
<td>0.73</td>
<td>3.51</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3.05</td>
<td>792</td>
<td>0.57</td>
<td>3.52</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>1.61</td>
<td>2962</td>
<td>0.83</td>
<td>3.18</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>2.34</td>
<td>2119</td>
<td>0.78</td>
<td>3.45</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>2.76</td>
<td>1276</td>
<td>0.68</td>
<td>3.49</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3.02</td>
<td>432</td>
<td>0.41</td>
<td>3.35</td>
</tr>
</tbody>
</table>

Table 4.11: Result table for design in Table 4.10 ($P=442$ ps)

4.5.3 FIFO Initialization

One requirement is to initialize the FIFO at a latency between its minimum and maximum value, such that there is adequate skew tolerance. In this section, the procedure to initialize the FIFO is described. The design uses static initialization [12] which is used when we know the exact relationship of latencies of various paths between the transmitter and receiver end of the FIFO. To use static initialization, the designer performs extensive timing analysis of the forwarded clock path and the clock distribution network \(^7\) to find the relationship between the delays of various paths. Given the worst-case skew in the forwarded clock and the receiver clock and a FIFO big enough to provide required skew tolerance, we select a value of $\Lambda_0$, the latency at which the FIFO is to be initialized, such that the $\lambda_{\text{min}}$ constraint for the FIFO is satisfied. Next, we adjust the FIFO size, if needed, to ensure the $\lambda_{\text{max}}$ constraint is satisfied until we converge on a value for $\Lambda_0$. The flowchart in Figure 4.13 gives the procedure for initializing the FIFO.

\(^7\)Variations in clock distribution network are given by the bounds global skew and jitter.
Chapter 4. Comparison of Interconnect Schemes

Given max. and min. delay per pipeline stage $\delta_{t,\text{pipe}}^{\text{max}}$ and $\delta_{t,\text{pipe}}^{\text{min}}$.
Given FIFO design characteristics $k_0$, $n_{\text{buf, fifo}}$.
Given max. and min. delay in the receiver clock $\delta_i^{\text{max}}$ and $\delta_i^{\text{min}}$.
Given ratio $\alpha$.

We want $N$ pipeline stages in the link, $N\alpha$ stages are in the forwarded path and $N(1-\alpha)$ stages are distributed in the FIFO.

Given maximum and minimum delay per forwarded pipeline stage and pipeline stages $N\alpha$ in forward path, we can find maximum and minimum case delay in the forwarded clock, $\delta_t^{\text{max}}$ and $\delta_t^{\text{min}}$, respectively.

Given $P$, global skew and jitter bounds. Find $P_{\text{max}}$ and $P_{\text{min}}$.

Chose $\Lambda_0$ such that we satisfy the constraint for minimum latency $\lambda_{\text{min}}$.

If constraint on maximum latency is satisfied at given $\Lambda_0$.

Increase FIFO size $N\alpha$.

Figure 4.13: Procedure for initializing the FIFO.

$\delta_{t,\text{pipe}}^{\text{max}}$, the minimum case delay per pipeline stage, $\delta_{t,\text{pipe}}^{\text{min}}$, the proportion of interconnect in the forwarded path, $\alpha$ and total number of pipeline stages, we can determine the minimum and maximum case delay through the forwarded path, $\delta_t^{\text{min}}$ and $\delta_t^{\text{max}}$, respectively. Similarly, given $P$ and global skew and jitter bounds, we can find maximum and minimum case delay in the receiver clock $\delta_r^{\text{max}}$ and $\delta_r^{\text{min}}$, respectively. Equations (4.5) derived in Section 3.4.1 give the constraints for the proper operation of an interleaved
and distributed FIFO, with degree $k$ of interleaving:

\[
\delta_r^{\text{min}} - \delta_t^{\text{max}} + \Lambda_0 P_{\text{min}} \geq \lambda_{\text{min}} \\
\delta_r^{\text{max}} - \delta_t^{\text{min}} + \Lambda_0 P_{\text{max}} \leq \lambda_{\text{max}}
\]  

(4.5)

We choose some value $\Lambda_0$ such that the constraint for $\lambda_{\text{min}}$ is satisfied\(^8\). We check for the $\lambda_{\text{max}}$ constraint for the $\Lambda_0$ chosen earlier. If the constraint is satisfied, we initialize the FIFO at $\Lambda_0$; otherwise, we increase the FIFO capacity\(^9\) and repeat the above process until we converge to a value for $\Lambda_0$.

In static initialization, a global reset signal is distributed to both transmitter and receiver clock domains. Figure 4.14 shows a typical clock distribution network. By including latches with clock buffers, as shown in Figure 4.15, we can distribute a global reset to all the domains.

---

\(^8\)We consider only the integral values of $\Lambda_0$ as it simplifies the distribution of the global reset signal.

\(^9\)The increase in the number of FIFO stages to satisfy the initialization constraint does not have much affect on the velocity of the link, as FIFO overheads are amortized over long interconnect.
are reset. When the reset signal is released, the receiver path waits for \( A_0 \) cycles of its local clock before it is enabled. This establishes the latency of \( A_0 \).

As an example, consider design point \( n_{buf,pipe} = 2 \), i.e., each forwarded pipeline stage consist of three buffered wire segments. From row 2 Table 4.13, the ratio of forwarded interconnect to that of interconnect distributed in the FIFO is 0.57. Thus, if we have 5 pipeline stages, 2 of them would be in the distributed FIFO. Given the number of distributed FIFO stages and \( k \), we can calculate the minimum and maximum latency \( \lambda_{min} \) and \( \lambda_{max} \) of the FIFO as discussed in Section 3.4.1. We measured the variations in the delay of the forwarded path, giving a minimum case delay of \( \delta_{t}^{min} = 1230ps \) and a maximum case delay of \( \delta_{t}^{max} = 3045ps \), and the period \( P \) of the forwarded clock is 442ps. The variations \( \Delta P \) in clock period \( P \) occur due to global skew and jitter. Given these values, we can solve Equation (4.5) so that both constraints are satisfied simultaneously. We get \( 8 \leq A_0 \leq 10 \), thus we chose \( A_0 = 9 \).

Table 4.12 and Table 4.13 gives the choice of FIFO designs and velocity of the link for various forward paths given in Table 4.8 and Table 4.9. We studied the source-synchronous interconnect while varying the number of buffered wire segments per pipeline stage. Given the tracking margin, the drift between the data and clock path increases as we increase the length of the buffered interconnect between two latches and this limits the throughput. For a given target clock period \( P \) and skew per pipeline stage, we design the FIFO so that we get the optimal velocity for the link. The pipelining overheads decrease as we increase the number of buffered wire segments per pipeline stage, resulting in increased velocity of the link.
### Table 4.12: FIFO design and link velocity for various design points: tracking margin of 20%
4.6 Comparison

We now compare various schemes of interconnect pipelining discussed in this chapter and study the trade-offs between their performance.

4.6.1 Throughput vs. Velocity

As discussed earlier, velocity can be used as the metric of comparison for different interconnect schemes. It is defined as the normalized inverse of latency, i.e., the velocity of a link is the rate at which data propagates (in meters/second).

![Velocity Vs. Throughput For Various Interconnect Techniques](image)

**Figure 4.16:** Throughput vs. velocity: synchronous and source-synchronous

Figure 4.16 plots the performance of various schemes of interconnect pipelining as
derived in Sections 4.3—4.6. The rightmost point on each curve occurs for any particular scheme, if each pipeline stage consists of single wire segment. The granularity of the pipeline keeps on decreasing (i.e., increased delay between two latches) as we move from right to left along the x-axis. For comparison, an optimally buffered wire in metal 5 gives velocity of 6.0mm/ns under worst-case crosstalk conditions.

Figure 4.16 shows that the multi-phase wave pipelined interconnect has the best performance as compared to all other designs. This is not surprising as it assumes that any required clock phase is available at any point on the chip and makes optimal use of wave-pipelining. However, the double-sided timing constraints in multi-phase pipelining results in design that are designed to operate at one frequency might not operate at lower frequencies.

The velocity for the two-phase time borrowing scheme falls off rapidly for throughput greater than 1.79GHz. For data rates below this value, each pipeline stage consists of single optimal length wire segments. For higher throughputs, it is no longer possible to traverse a single optimal length wire segment in half a clock cycle and leave enough time for the timing constraints at the latch. Thus, for higher target throughput it is necessary to use shorter segments which results in a drop in the velocity of the link.

For comparison, we also looked at a simple, flip-flop based interconnect with and without wavepipelining, where all flip-flops were triggered by the positive edge of the clock and wave pipelining was used whenever the throughput allowed. It has the lowest performance on throughput vs. velocity metric at high frequencies. This is because the flip-flop based pipelining can not exploit time-borrowing, thus incurs extra overheads due to clock uncertainty and set-up time constraints at each pipeline stage.

The velocity of source-synchronous interconnect with interleaved and distributed FIFO is limited by the delay of the strobe path, designed to be 1.4 times the data path delay for tracking margin of 40%. For throughput beyond 1.79GHz the performance
of two-phase interconnect drops due to increased latch overheads resultant of using sub-optimal sized wire segments. For target throughputs greater than 2.4GHz source-synchronous with interleaved and distributed FIFO has better performance as compared to two-phase interconnect.

Another way to look at this data is to compare the latencies of the various schemes for fixed throughput, or the throughputs achievable at fixed latency. For example, if the time budget to cross a 20mm wire is 4.4ns, i.e., the velocity target is 4.5mm/ns; the twin-phase scheme can give a throughput of around 2.3GHz, the source-synchronous with distributed FIFO and best tracking gives throughput of around 1.5GHz. Similarly, given a throughput target of 2.5GHz, the two-phase synchronous interconnect can cross a 20mm repeated wire in around 5.13ns while the source-synchronous approach with a distributed FIFO takes 5.0ns to go across a 20mm repeated wire.

### 4.6.2 Power Dissipation

This section gives the comparison of power consumption for synchronous and source-synchronous methods. Power is measured for transfer of single bit of data when both synchronous and source-synchronous interconnects operate at the same data rate. The control path power for synchronous interconnect does not include the clock network, but it does include the power consumed in buffers driving the latches. Source-synchronous design uses both the edges of the forwarded clock to transmit the data. Table 4.14 gives the power consumption for a single bit data path for minimum pitch wire, when both synchronous and source-synchronous interconnect operating at same data-rate of 1GHz.

As seen from tables, the data path consumes 86% of the total power in the synchronous while data path power is 44% of total power in the source-synchronous interface. The data path power in source-synchronous is higher as compared to synchronous because of overheads due to tri-state latches in the FIFO stages. The absolute power
Chapter 4. Comparison of Interconnect Schemes

<table>
<thead>
<tr>
<th>Topology</th>
<th>Synchronous (Two-phase)</th>
<th>Source-synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data path</td>
<td>17mw/bit</td>
<td>17.8mw/bit</td>
</tr>
<tr>
<td>Control path</td>
<td>3.4mw/bit + clock tree power</td>
<td>21mw + 4mw/bit</td>
</tr>
</tbody>
</table>

Table 4.14: Power: minimum pitch and minimum spacing

consumed in source-synchronous is high as compared to synchronous because of overheads due to the control path, but as we increase the numbers of bits in the data path the control path power is amortized over multiple data bits. In all the designs we assumed skew budget of 10%, we can further save on the power budget by relaxing the global skew constraint. The performance of synchronous interconnects is adversely affected by relaxed skew in the global clock network. However, the performance of source-synchronous would not be affected to a great extent as it generates its own local clock that relies on matching between data and clock path. Thus, we can have high-performance designs with lower power consumption.

Our comparison here shows that source-synchronous signaling is a promising option for interconnect pipelining in high-performance designs.
Chapter 5

Conclusion

In high performance designs, it can take multiple clock cycles to transmit a signal across a chip. Thus, single cycle communication over long interconnect is not possible and we must therefore use some form of interconnect pipelining. We considered options of synchronous and source-synchronous pipelined interconnects, which are used for global communication between modules running at the same frequency. There is a trade-off between throughput and latency that can be achieved for each method of interconnect pipelining. We introduced a velocity metric that provides a normalized measure of latency, and studied velocity vs. throughput trade-offs for each interconnect topology.

To evaluate synchronous interconnect techniques, we presented implementations of two-phase pipelining and multi-phase pipelining. Two-phase pipelining is a simple and conservative model that establishes a lower bound on what can be achieved using synchronous techniques. The multi-phase design makes optimistic assumptions about the clock network and makes maximal use of wave-pipelining. Though impractical in real world designs, the multi-phase designs can provide an upper bound on the performance of synchronous techniques.

We also examined source-synchronous methods of interconnect pipelining, where the
strobe is forwarded along with the data. We used a FIFO to compensate for the skew between the forwarded transmitter clock and the local receiver clock. We studied various FIFO schemes for providing the required skew tolerance between the forwarded and the local receiver clock. We can further improve the performance of source-synchronous interconnect by distributing the pipeline stages in the FIFO using a twin-control scheme presented in [30]. This scheme suffers from quadratically increasing control path latency as we increase the length of the wire between the FIFO stages. We improved upon this scheme by using replicated and buffered, request and acknowledgement paths. However, throughput is still limited as it does not allow wave-pipelining between the FIFO stages. We introduce a novel distributed multiple control FIFO, which enables wave-pipelining between the FIFO stages.

Finally, we compared the performance of synchronous and source-synchronous methods of interconnect pipelining on velocity and throughput metrics. We found that source-synchronous with an interleaved and distributed FIFO is competitive with two-phase design at higher target throughput. Though the performance of source synchronous with interleaved and distributed FIFO scheme is significantly below the multi-phase interconnect, it has the advantage over the multi-phase design that it can actually be implemented.

Reducing power is the major challenge in future designs. Clock distribution networks often consume around 50% of the total chip power budget. In a SoC based design methodology we have multiple clock domains for various reasons, the most important being that it is economical to break large system in to independent clock domains to save on area and power for precise clock distribution. On-chip interconnects among the different clock domains are not only difficult, they are also unsupported by most commercial design and verification tools. Source-synchronous interconnect can provide the framework for global communication without the power overhead of low skew clock
distribution network.

Long wires and power are going to be the major issues affecting the performance of future designs. In deep-submicron technologies, the delays of metal lines continue to increase in spite of the increasing number of metal layers and use of low-k dielectrics. Power will be a major parameter affecting interconnect design decisions. Source-synchronous methods of interconnect pipelining provide advantages over synchronous techniques in terms of interconnect performance and overall system power. There is an area overhead due to logic used in clock-forwarding and FIFOs, but this is small compared with total chip area. Based on the comparison in this thesis, we expect that source-synchronous signaling techniques like the distributed FIFO proposed here will be increasingly used in future deep-submicron designs.

5.1 Future Work

Long wires are certainly a big enough concern in high-performance designs that it calls for development and use of new designs for the interconnect. The interleaved and distributed FIFO presented in this thesis promises a significant improvement in the performance of long interconnects. We need to further evaluate other methods of pipelining and also study the robustness of these schemes for different applications.

One of the major concerns of design houses are shorter and competitive product design cycles, which are not possible without extensive CAD support for any methodology. Thus, the next step would be to develop CAD tool support for generating top-level, long wire interconnect from a specification. This would involve an environment for modeling, simulation and perhaps formal verification of designs and then synthesis of the global interconnect from such models.
5.2 Thesis Contributions

The contributions of this thesis can be summarized as:

- A novel interleaved and distributed FIFO that hides the control latency and enables wave-pipelining between the FIFO stages. An implementation of the source-synchronous interconnects using this interleaved and distributed FIFO for the skew compensation.

- A metric called velocity is defined, and a comparison of the performance of synchronous and source-synchronous signaling is presented using the metrics of throughput, velocity and power.
Bibliography

dual PLLs in a 0.13um SOI technology with copper interconnect and low-k BEOL

plications for CAD. *Proceedings of the IEEE/ACM International Conference on


design: roadblocks and avenues. *Proceedings of the 1st IEEE/ACM/IFIP Inter­
national Conference on Hardware/Software Codesign and System Synthesis*, pages


[40] Ivan E. Sutherland and Scott Fairbanks. GasP: A minimal FIFO control. *Pro-


A-1 Repeater Sizing

Consider a long wire of length L, divided it in N equal sized segments and then insert buffers between segments to reduce wire delay [6]. We can find optimal number of wire segments N, and repeater size Mx (M times minimum sized inverter) by optimizing the delay of series of repeater-wire-repeater model (Figure A-1).

\[
\begin{align*}
T_{seg} &= \left[ \frac{R_{eqn}}{M} \right] [C_J M (1 + \beta) + \frac{(C_{int} + \pi C_0) L}{2N}] \\
&\quad + \left[ \frac{R_{eqn}}{M} + \frac{R_{int} L}{N} \right] \frac{(C_{int} + \pi C_0) L}{2N} + C_G M (1 + \beta)] \\
T_{total} &= N \times T_{seg} \\
\frac{\partial T_{total}}{\partial N} &= 0 \\
\frac{\partial T_{total}}{\partial M} &= 0
\end{align*}
\]

Where:
\( R_{int} \) = wire resistance per \( \mu m \)
\( C_{int} \) = wire capacitance per \( \mu m \)
\( R_{eqn} \) = resistance min sized buffer
\( C_J = C_j \times W \), where \( C_j \) is junction capacitance per \( \mu m \)
Appendix A

\[ C_G = C_g \times W, \text{ where } C_g \text{ is gate capacitance per } \mu m \]

\[ C_c = \text{Crosstalk capacitance per } \mu m \]

\[ B = 2 \text{(ratio of PMOS to NMOS in gate with equal rise and fall time)} \]

\[ W = \text{min gate width in a technology} \]

\[ x = 0, \text{ if neighbours switch in same direction} \]

\[ x = 2, \text{ if neighbours switch in opposite direction} \]

\[ x = 1, \text{ if neighbours switch in alternate direction} \]

Equations A-2 gives optimal value of repeater size \((M)\) and optimal number of segments \((N)\) as a function of technology parameters and wire length.

\[
N = \sqrt{\frac{R_{int}(C_{int} + 2C_c)L^2}{2R_{eqn}(C_G + C_c)(1 + \beta)}}
\]

\[
M = \sqrt{\frac{R_{eqn}(C_{int} + 2C_c)}{C_G(1 + \beta)R_{int}}}
\] (A-2)

For TSMC 0.18 \(\mu m\) Al process, assuming best case crosstalk \((x=0)\) gives \(L_{opt}=1346 \mu m\) and \(M_{opt}=65X\) for a 20mm minimum width minimum spacing interconnect in metal5. We simulated the string of optimally sized wires \((1346 \mu m)\) and inverters \((64X)\) at typical process (TT), supply voltage \((1.8v)\) and temperature \((25^\circ C)\) conditions. The delay of a buffer driving wire load was measured to be 137ps.

A-2 Latch Design

Latch shown in Figure A-2 is used as sequential element for this work. The keeper is minimum sized tri state driver. We find transistor sizes by minimizing Elmore-delay function (Figure A-3), in Matlab with optimal wire segment and repeater size, it being a convex function.
Figure A-2: Latch topology

Figure A-3: Elmore delay model for latch

\[
T_{elmore} = \frac{R_{en}}{M_3} \left[ C_J(1 + \beta)M3 + \frac{C_{wire}}{2} \right] + \left[ \frac{R_{en}}{M_3} + R_{wire} \right] \left[ \frac{C_{wire}}{2} + C_G(1 + \beta)M1 \right] \\
+ \frac{R_{en}}{M_1} \left[ C_J(1 + \beta)M1 + \frac{3WC_G}{2} \right] + \left[ \frac{R_{en}}{M_1} + R_{lg} \right] \left[ \frac{3WC_G}{2} + C_G(1 + \beta)M2 \right] \\
+ \frac{R_{en}}{M_2} \left[ C_J(1 + \beta)M2 + \frac{C_{wire}}{2} \right] + \left[ \frac{R_{en}}{M_2} + R_{wire} \right] \left[ \frac{C_{wire}}{2} + C_G(1 + \beta)M3 \right]
\] (A-3)

Which gives minimum delay at M1=36X, w=12\mu m, M2=60X.
A-3 Latch Characterization

The latch was characterized for its set-up time, hold time and data to q delay. The *set-up time* \((t_s)\) is the time that data input must be valid before falling edge of clock. The *hold-time* \((t_h)\) is defined as time that data input must be valid after falling edge of clock. When set-up and hold times are satisfied data at input is copied to output after worst case *propagation delay* denoted by \(\delta_{dq}\).

![Figure A-4: Set-up time and set-up time optimization](image)

As shown in Figure A-4 there is trade off between set-up time \((t_s)\) and latch-delay \((\delta_{dq})\). We can have low set-up time with extremely high \(\delta_{dq}\) delay in meta-stable region or we can trade-off high set-up-time with low latch delay. In above case (Figure A-4) we get minimum value of \(\delta_{dq} + t_s\) with \(t_s=70\)ps and corresponding \(\delta_{dq}=95\)ps. Similarly we can find out hold time for above latch. The Figure A-5 shows hold time constraint for the latch. We need to have input stable for at least \(t_{hold}=70\)ps after falling clock edge. The delay of a latch driving a wire and buffer load was measure to be 168ps, while the set-up and hold conditions are not violated.
In this section, we perform a brief study of crosstalk effects in TSMC 0.18 μm technology. We also study a way to reduce the crosstalk, using staggered repeaters [46], [23]. Figure A-6 shows the minimum width 16-bit bus configuration that is used for studying the crosstalk. This bus configuration was studied for cross coupling with 2mm wire segments. Figure A-7 shows variations in wire delay due to crosstalk noise from neighbours. It shows that, there are significant variations in the delay of a signal line due to cross coupling from neighbours. However, this cross-coupling is significant only till third neighbour.

One of the solutions for reducing crosstalk is to offset repeaters on adjacent lines, also known as staggered repeaters. With staggered repeaters any worst-case simultaneous switching on a neighbour line persists only for half of each period between consecutive inverters, and furthermore becomes best-case simultaneous switching for the other half.
Figure A-7: Delay variations due to crosstalk in 16-bit minimum width bus of the period. The setup in Figure A-8 was used to study crosstalk with staggered repeaters. The buffers are sized for optimal repeater insertion.

Figure A-8: Staggered repeaters

The delay across the middle wire segment was studied for various combinations
of neighbourhood switching patterns. As shown in Figure A-7, we have to consider
crosstalk till first three neighbours, however, with staggered repeaters this was reduced
to two immediate neighbours. Figure A-9 show variation in wire delays of the middle
segment due to various neighbourhood switching patterns. This shows that crosstalk due
to first neighbours always slows down. This is contradictory to the case when neighbours
switch in the same direction as victim, the victim delay is expected to decrease due to
crosstalk. This is true, if we just consider wire delay, but our wire is being driven by
buffers. Though we expect our wire delay to be decreased in case of aggressor switching
in same direction as victim, but the input edge of buffers is slowed by the aggressors, so
we have overall (negligible) increase in delay.

Thus, with staggered repeaters crosstalk is considerably reduced and we have sig-
nificant crosstalk effects only from the first two immediate neighbours. Maximum slow
down in the signal due to crosstalk is given by: Max slow down due to 1st neighbours
+ Max slow down due to 2nd neighbours = 37% Maximum speed up in the signal due
to crosstalk is given by: Max speed up due to 1st neighbours + Max speed up due to
2nd neighbours = 5%
A-5 Placement Sensitivity

The purpose of this section is to find out how sensitive is wire latency to drift in wire segment from the optimal solution.

Figure A-11 shows the variation in wire delay, repeater delay and total delay as
Appendix A

Figure A-11: Delay vs. drift (optimal repeater)

function of wire segment length, assuming that we have designed repeater for optimal wire segment. It shows that there is optimal point, when wire segment delay is equal to repeater delay. But as Figure A-12 shows that, if wire segment vary ±20% from optimal length, the overall latency vary by only around 7%.

Figure A-12: Percentage variations in delay vs. drift from optimal repeater
B-1  Effect of Multiple Control on Latch Delay

In multiple control scheme, for each path, we use a latch followed by multiplexer. But as we keep on increasing \((k)\) degree of multiple control, the load on datapath increases due to increase in parasitic delay of multiplexer. Thus datapath becomes limiting factor in setup and hold time constraints at last and first latch respectively. The following section determines the limit of multiple control \((k)\) at which the datapath constraint overtakes control path.

We can use multiplexer tree or a single multiplexer followed by series of buffer stages. We need to decide which configuration works best for given electrical effort requirements.

We can design an \(n\)-input multiplexer as an \(n\)-input multiplexer followed by series of amplifiers. Consider designing as \(n\) input multiplexer followed by amplifiers, where \(K\) is the degree of multiple controls, \(r\) gives the size of multiplexer. In this case \(r=k\). Thus total delay is given by

\[
D_{delay} = D_{mux} + T_{dq} = (n_a + 1 + 2)(2 + 2 \times H)^{(1/n_a + 1/2)} + n_a P_{inv} \\
+ 2rP_{inv} + 1.67P_{inv} + P_{inv}
\]

Figure B-1: Latch followed by mux - logical effort
Appendix B

Where total electrical effort (H) is dependent on degree of multiple control, as degree of multiple control increases the load on D input of latch increases. Thus designing for \( C_{tn} \) for that of minimum sized inverter, H is given by

\[
H = C_{\text{wire}}/2.4 * k
\]  

(B-2)

Minimizing B-1 for different multiplexer sizes \((r)\) we get.

<table>
<thead>
<tr>
<th>Multiplexer Type</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single 2 input multiplexer</td>
<td>19.2 ( \tau_{\text{inv}} )</td>
</tr>
<tr>
<td>Single 5 input multiplexer</td>
<td>19.71 ( \tau_{\text{inv}} )</td>
</tr>
<tr>
<td>Single 6 input multiplexer</td>
<td>20.07 ( \tau_{\text{inv}} )</td>
</tr>
<tr>
<td>Single 8 input multiplexer</td>
<td>21.62 ( \tau_{\text{inv}} )</td>
</tr>
<tr>
<td>Single 10 input multiplexer</td>
<td>23.11 ( \tau_{\text{inv}} )</td>
</tr>
</tbody>
</table>

Total Delay D is given by \( D_{\text{mux}} + T_{DQ} \). Thus degree of multiple control is limited when D exceeds forward cycle time \((\delta_{\text{f,ctrl}})\) of controller. Which happens at \( k \geq 5 \). Thus at this point the latch requirements set lower bound for \( \Delta_{RT,\text{min}} \) and control path set bound for \( \Delta_{RT,\text{max}} \).
C-1 Distributed and Interleaved FIFO

The FIFO with the distributed and interleaved control (see Figure 3.7) was introduced in Section 3.4. The details of control circuit are given in Figure C-1. It shows a single control path of a FIFO with $k$ degrees of multiple control. Note the buffers in the control path; the last buffer stage in both the request and acknowledgement control path is made of a single transistor either NMOS or PMOS. This helps to reduce load on control path. As we increase the degree of multiple control the load on datapath increases, thus for some value of $k$ the datapath constraint overtakes the control path constraint (see Appendix B-1).

Figure C-1 shows the controller for the FIFO. This is a 4-2 FIFO [40], the forward delay of the FIFO includes 4-gate delays and the delay of the wire and buffers. Similarly the reverse delay of the FIFO consists of 2-gate delays and the delay of the wire and buffers. The forward cycle time was measured to be $743\text{ps}$ and reverse cycle time was $588\text{ps}$. We take the average of the forward and reverse delay of a FIFO stage, while comparing the spice and hand calculations. The maximum and minimum case latency of the FIFO is given as below, the minimum case represents the time for request to ripple unobstructed through the FIFO, while maximum happens when each stage waits for previous acknowledgment.

Figure C-2 show variations in maximum and minimum latency of multiple control scheme for $k = 3$, $n_{fifo} = 2$ and $n_{buf,fifo} = 0$. The minimum latency is constant being equal to fall through time through FIFO, which happens when FIFO is request limited. While the maximum latency is given by number of FIFO stages time clock period minus the time, it takes for acknowledgment to travel back through FIFO. This
is what happens when FIFO is acknowledgment limited.

**Figure C-1**: Control path of a multiple-control scheme

**Figure C-2**: Latency profile of multiple-control scheme ($k=3$) with buffered control
D-1 Dynamic FIFO Initialization

FIFOs are memory elements used to pass wide data between asynchronous domains. In clock forwarding scheme we achieve high throughput rates by forwarding clock along with data. But due to various variations there is no definite phase relationship between forwarded clock and clock at receiver end. Thus to synchronize between two domains we need to have some synchronization mechanism. We have option of using ripple FIFO [47] or pointer FIFO [48]. The advantage of [47] is lower latency as compared to [48]. But as drift in signals increase we have to increase number of FIFO stages, thus adding overhead of each FIFO stage, thus we would like to use [Nowick] or RAM based synchronizer with write and read pointers. But in our case number of FIFO stages limited to 2-3 we can synchronize with [47] scheme. Let $\delta_{TR}$ denote time from when a clock event occur at transmitter until corresponding clock event occur at receiver. Then

\begin{equation}
\delta_{TR,_{min}} = \delta_{\phi_{t,_{min}}} - \delta_{\phi_{t,_{max}}}
\delta_{TR,_{max}} = \delta_{\phi_{t,_{max}}} - \delta_{\phi_{t,_{min}}}
\end{equation}

Figure D-1: 2-stage FIFO
Appendix D

Equations D-1 give limit of $\delta_{TR}$, for which correct operation of FIFO can be guaranteed. In case 1 FIFO is data limited, giving maximum latency $\lambda_{max}$. In case 2 FIFO is acknowledgement limited, giving minimum latency $\lambda_{min}$. Given initial $\delta_{TR0}$, FIFO can be initialized to operate at various possible latencies corresponding to $\delta_{TR0}$, $\delta_{TR0}+P$ and $\delta_{TR0}+2P$.

For given initial $\delta_{TR0}$ we want to initialize FIFO for maximum robustness. For two stage FIFO latency $\lambda$ varies between $\lambda_{min}$ to $\lambda_{max}$ corresponding to $\delta_{TR,min}$ and $\delta_{TR,max}$ respectively.

$$\delta_{TR,min} \approx \lambda_{min} \approx (n+1)\delta$$
$$\delta_{TR,max} \approx \lambda_{max} \approx (n+1)(P-\delta)$$

The idea behind maximum robustness initialization is to add adjustable delay to output of nand-gate in first gasp stage. Initially delay is set to large value ($\delta_{nand,\text{start}}$) and is gradually decreased to minimum value. To exclude initialization at $\delta_{TR0}$, we have limit on initial $\delta_{nand,\text{start}}$. We have to ensure that $\delta_{nand,\text{start}} \geq \delta_{min}$. To exclude initialization at $\delta_{TR0}+2P$, we have to ensure $\delta_{nand,\text{start}} \leq \delta_{max}$.

Figure below shows limit on $\delta_{min}$ as function of initial phase difference between transmitter and receiver, $\delta_{TR0}$. The nand gate of first Gasp stage has tuneable delay which is set to high value during initialization when init signal is high.

To exclude initialization at $\delta_{TR0}$ we want request on $\phi_t \uparrow$ to exclude pairing with acknowledgement on $\phi_r \uparrow$ which is $\delta_{TR0}$ after $\phi_t \uparrow$ edge. The low edge of node x3 disables pull down path for node c. So pull up of node c by node x3 being low, takes precedence over pull down of node c by virtue of node x5 being low.

The above figure shows minimum limit on $\delta_{min}$,
Figure D-2: Limit on $\delta_{\text{min}}$

\[
\delta_{\text{min}} = \delta_{TRO} + \delta_1 + \delta_2
\]

\[
\delta_1 = \text{cycle time 2}\text{nd stage} = \delta_{x_{21} - x_{51}} + \delta_{x_{51} - x_{51}}
\]  

$\delta_2$ = time $x_3$ must remain low after second stage stop driving node $c = \delta_{x_{31} - c}$

To exclude initialization at $\delta_{TRO} + 2P$ we have to ensure $\delta_{\text{nand,start}} \leq \delta_{\text{max}}$. To exclude initialization at $\delta_{TRO} + 2P$ we want request on $\phi_t \uparrow$ to exclude pairing with acknowledgement on $\phi_r \uparrow$ which is $\delta_{TRO} + 2P$ after $\phi_t \uparrow$ edge.

Figure below shows limit on $\delta_{\text{max}}$ as function of initial phase difference between transmitter and receiver, $\delta_{TRO}$. The maximum nand gate delay is limited by clock period. It should be short enough so that node $x_3$ resets to high before next rising edge of $\phi_T$.

The above figure shows minimum limit on $\delta_{\text{max}}$. 

\[
\begin{align*}
\delta_{\text{max}} &= \delta_{TRO} + \delta_1 + \delta_2 \\
\delta_1 &= \text{cycle time 2}\text{nd stage} = \delta_{x_{21} - x_{51}} + \delta_{x_{51} - x_{51}} \\
\delta_2 &= \text{time } x_3 \text{ must remain low after second stage stop driving node } c = \delta_{x_{31} - c}
\end{align*}
\]
Figure D-3: Limit on $\delta_{\text{max}}$

\[ \delta_{\text{max}} = P - (\delta_1 + \delta_2) \]
\[ \delta_1 = \phi_{T \uparrow \rightarrow x31} \]

$\delta_2$ = minimum time that x3 must rise before next rising edge of $\phi_T$