Pattern-Placement-Error Detection for Spatial-Phase-Locked E-Beam Lithography (SPLEBL)

by

Cynthia L. Caramana

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

© Cynthia L. Caramana, MMIV. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUL 2 6 2004
LIBRARIES

Author . . . .
Department of Electrical Engineering and Computer Science
May 20, 2004

Certified by . . . .
Henry I. Smith
Keithley Professor of Electrical Engineering
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

BARKER

Pattern-Placement-Error Detection for Spatial-Phase-Locked E-Beam Lithography (SPLEBL)

by

Cynthia L. Caramana

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Spatial-phase-locked electron-beam lithography (SPLEBL) is a new paradigm for scanning electron-beam lithography (SEBL) that permits nanometer-level pattern placement accuracy. Unlike conventional SEBL systems which run in an open-loop fashion, SPLEBL uses continuous feedback to directly monitor and correct the beam's position, eliminating the need for expensive shielding equipment and costly isolation techniques. When compared to the most advanced and sophisticated SEBL systems, SPLEBL exceeds all of them in the areas of pattern-placement accuracy and affordability. However, much improvement is needed to increase the throughput of SPLEBL to a level on par with its commercial counterparts.

As SPLEBL is further optimized for throughput and affordability, the placement-error detection and correction subsystem will need to be upgraded with a custom hardware solution. The work presented in this thesis describes the design of an efficient error detection and correction mechanism for SPLEBL and how it could be implemented as a digital circuit. An error-detection algorithm, well suited for digital hardware, has been developed and characterized. A digital circuit design to implement the algorithm has been created, optimized, and verified using the MathWorks Simulink^{TM} and the Xilinx System Generator^{TM} hardware design tools.

Thesis Supervisor: Henry I. Smith
Title: Keithley Professor of Electrical Engineering

Acknowledgements

I would like to express my appreciation to my advisor, Professor Henry Smith, for all of his guidance and support. I want to thank him for his encouragement and for allowing me to work independently. Working in the NanoStructures Laboratory has taught me that research can be frustrating and demanding but also extremely rewarding. My work on this thesis has helped me to improve my writing ability, organization of thoughts, and presentation skills, all of which will be invaluable to my future career.

The members of the NanoStructures Laboratory are a superb group of talented individuals to whom I owe much gratitude. I would like to thank Todd Hasings and Feng Zheng for their helpful explanations and advice with regard to every aspect of SPLEBL, and Dr. Schattenburg for his guidance and insight when I was having difficulty with developing a proper phase-detection scheme. I appreciate the time Tymon Barwicz spent answering my questions and for graciously providing me with his vibrational measurements on the Raith. It has been a pleasure sharing an office with Lynn Chen. She has been a great friend and I have enjoyed having another woman in the group to relate to. I would like to wish all my other NSL friends and colleagues, Mike Walsh, Euclid Moon, Amil Patel, Minghao Qi, David Chao, Will Arora, Ryan Tabone, and Thomas O'Reilly, the best of luck with their studies and research. I would also like to thank Lucent Technologies-Bell Labs for their financial support through their GRPW Fellowship award.

Outside of research, I would like to pay tribute to the 6.1 Girls, an extrodinary group of women who are some of my dearest friends. I also want to thank Ulas Ziyan for keeping my spirits up as I struggled to finish my thesis. Finally, I would like to express my gratitude to my parents, Edward and Christina Caramana, for all of their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past several decades, tremendous changes have occurred within the semiconductor industry. Driven by industry requirements for smaller, faster, and less expensive integrated circuits, the critical dimensions for semiconductor devices have been decreasing at an accelerated rate. More layers, new materials, larger wafer sizes, and most importantly, smaller line width, have enabled the semiconductor industry to keep pace with Moore's Law. By providing the capability to continuously reduce the size of features patterned on wafers, each new generation of lithography has enabled faster microprocessor and smaller, less-expensive integrated circuits. The demand for smaller feature sizes pushes the developers of lithography technology to improve resolution. As resolution is improved, however, an often overlooked yet equally important issue arises — pattern placement.

Accurate pattern placement is crucial to the planar fabrication process, with which integrated circuits and various devices are fabricated layer by layer. Mask patterns must overlay to within a fraction of the minimum feature size. As linewidth decreases, this tolerance is also reduced, resulting in a great need for highly accurate pattern placement on masks, many of which are fabricated by electron-beam lithography.

According to the International Technology Roadmap for Semiconductors (shown in Figure 1-1), by 2016 mask image placement error must be below 6 nm on a 152 mm^2 wafer for a 22 nm node [1]. No manufacturing solution is known to produce such a mask; however,

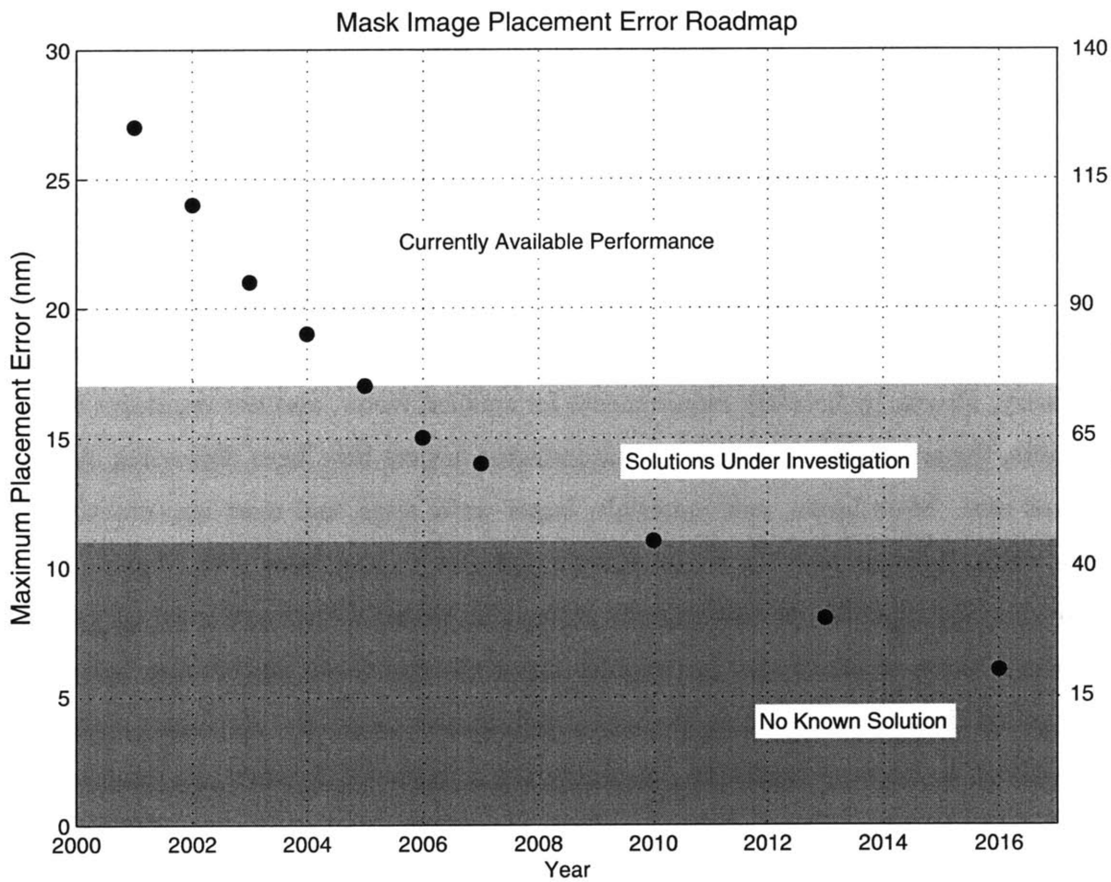


Figure 1-1: Mask image placement requirements for a 152 mm^2 wafer, adapted from the International Technology Roadmap for Semiconductors, [1].

Company:	Raith	Lepton	Etech Systems	Leica
Model:	150 (with SPLEBL)	EBES4	MEBES 5500	VB-6
Stitching Error:	1 nm	40nm	30 nm	20 nm
Speed (MHz):	2.6	320	320	50
Price (Millions of U.S. Dollars):	1	3	14	5

Table 1.1: Comparison of Gaussian spot, raster-scan mask making SEBL systems.

the effort involved would be equivalent to placing each road, street, and highway in the continental United States to within 3 inches of its intended position. How can this degree of accuracy be achieved within the next decade when even the most sophisticated systems can barely achieve placement accuracy to within 10 nm?

Spatial-phase-locked electron-beam lithography (SPLEBL) introduces a novel approach to improving pattern placement accuracy by adding a feedback loop to continuously monitor and correct the beam position in real time. The locations of the patterns are directly registered to a fiducial grid that produces a detectable signal. The signal is then processed to detect any deviation of the beam from its desired location on the substrate, and the correction signals are fed back to the beam-control electronics to cancel the errors in the beam's position. SPLEBL has been implemented on a Raith 150 scanning electron-beam lithography system; moreover, preliminary results show that SPLEBL is a highly effective means to achieving placement accuracy to within 1 nm [2]. SPLEBL eliminates the need for expensive shielding equipment, temperature control, and vibration isolation, required by contemporary commercial SEBL systems running open loop. Table 1.1 compares SPLEBL with some of the the most advanced commerical SEBL tools [3], [4], [5], [6] . The pattern-placement accuracy of SPLEBL exceeds that of its commercial counterparts by more than a factor of ten, and SPLEBL is far more affordable. Yet the speed of SPLEBL needs to be improved to increase its throughput to a level on par with commercial SEBL systems.

The current SPLEBL implementation on a Raith 150 uses a general-purpose processor to calculate the phase error and the correction signals; however, the processing becomes computationally intensive as the number of samples and bandwidth increases. As the system is further optimized for throughput, the need for very fast phase locking and error-correction computation arises. A custom hardware solution in the form of a dedicated IC chip is more efficient than a processor because the control logic can be implemented entirely in the hardware. In a dedicated chip, the hardware architecture can be tailored for speed, saving precious clock cycles. Moreover, a general-purpose processor is expensive, when considered from a commercial development point of view, compared to a custom chip that can be mass manufactured at a much lower price-per-unit cost.

This thesis will describe the design of a time-efficient phase-error and position-correction processing mechanism for SPLEBL, and how the algorithm could be acheived entirely with a digital circuit.

1.1 Scanning Electron-Beam Lithography Overview

Electron-beam lithography has been used for many years to write features at linewidths below the capability of optical lithography. An electron beam can create linewidths as fine as 10 nm using magnetic lenses to direct electrons onto the surface of a resist-coated substrate. Although the resolution of electron-beam lithography systems greatly exceeds that of optical systems, patterns are written in a serial process. Writing an entire chip using a scanning electron-beam would take hours, whereas mask-based optical projection requires about one minute. The development of photomasks is the most widespread application of electron-beam lithography, though some other applications include fabricating integrated optical components and writing special features such as magnetic heads.

1.1.1 System Components

A scanning-electron-beam lithography system (SEBL) is composed of an electron optical column, a laser-interferometer-controlled mechanical stage, and a computer to control the various machine subsystems and to transfer pattern information to the beam deflection coils. The electron-optical column is equipped with an electron gun to produce an electron beam, an anode to set the beam energy, a beam blanker that modulates the beam, and various elec-

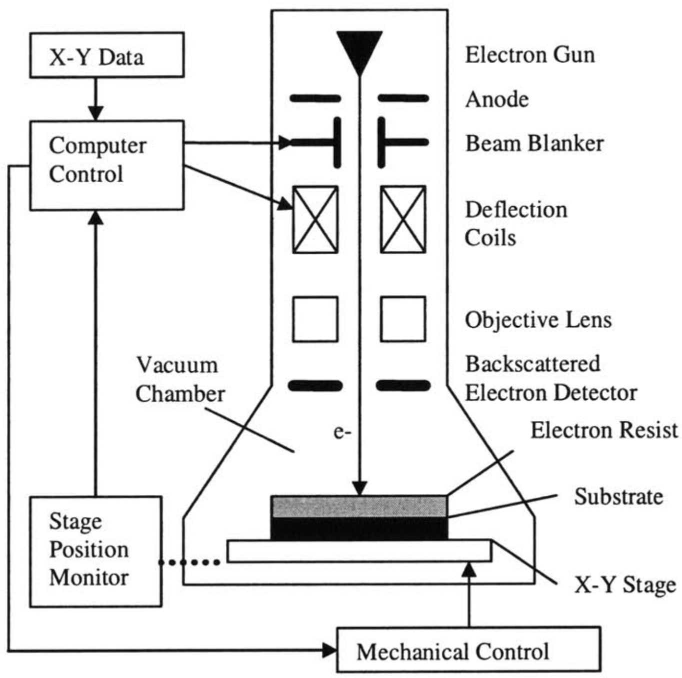


Figure 1-2: Schematic of a scanning-electron-beam lithography system: electrons are emitted from an electron gun and accelerated down the electron optical column. A set of electromagnetic lenses focus the beam onto the substrate. The beam scans along the substrate, while the blanker turns it on and off to produce a pattern.

tromagnetic lenses and coils to adjust the beam's shape and position. An overall schematic of a scanning electron-beam lithography system and its main components are shown in Figure 1-2.

A pattern is first designed on a computer and then sent to the pattern generator to be broken down into a series of lines or basic shapes which are then translated into a series of beam deflections and beam-blanker signals. The beam is turned off and on by a beam blanker that electrostatically deflects the beam away from its propagation path onto a beam stop. At high data rates, the beam blanker has a demanding task of modulating the beam without introducing distortion or shifting. A series of electromagnetic lenses and deflection coils focus the beam onto a desired location on the substrate. To minimize the distortion caused by stray magnetic fields, the column must be well shielded. Stray fields at 60 Hz and multiples thereof are most pervasive and can be difficult to screen out [7]. The deflection system must adhere to stringent requirements in order to achieve placement accuracy and repeatability to within a pixel. The field over which a beam can be deflected is limited to a small area (typically 100 μm to 250 μm); therefore, a precision stage is needed to move the substrate into the scan field. The pattern is built up from fields that have been "stitched" together.

1.2 Pattern Placement

Virtually all of the lithography used by the semiconductor industry is done by optical projection of a mask onto a substrate. Although electron-beam lithography lacks the throughput to print integrated circuits directly, its high resolution makes it ideal for mask fabrication. As the industry continues to push the achievable resolution of optical projection lithography, the accuracy of the image placement on the mask becomes ever more critical. One method employed to improve image-placement accuracy on masks has been to image a fiducial mark on the stage, away from the writing area [8]. However, this sporadic method of direct referencing is time consuming and marginally effective since the the beam will drift between

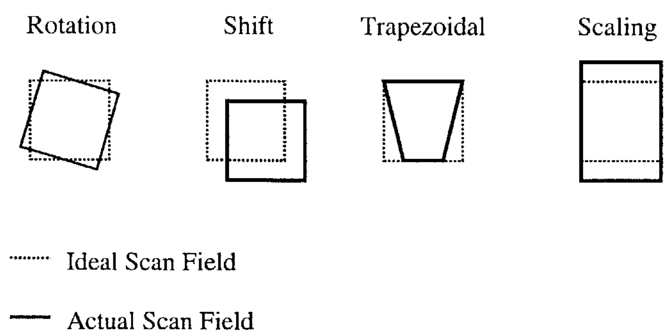


Figure 1-3: Four classes of scan field distortion.

registrations.

1.2.1 Pattern-Placement Errors

Pattern-placement error can be divided into two categories: interfield and intrafield distortion. Interfield distortion is the mismatch between two adjacent fields at their common boundary. This mismatch is often referred to as "stitching error" and arises from a combination of four main classifications of scan field distortion: rotation, shift, trapezoidal, and scaling [7]. Figure 1-3 illustrates these various errors. Some major sources of interfield distortion are the following: disparity in length scales between beam deflection and the laser interferometer, rotation of the deflection-field axes relative to the stage coordinate axes, electrical charging of sample and electron-beam system parts, temperature gradients, and mechanical strains and vibrations.

Intrafield distortion, on the other hand, is the deviation of the beam positioning within a scan field and is usually non-uniform and nonlinear across the field. Intrafield distortion arises primarily from electrical charging (of the resist, sample, and optical column), stray magnetic fields, thermal gradients, mechanical vibrations, lens distortion, deflection distortion, and D/A converter errors. The accumulation of both interfield and intrafield placement errors results in patterns being displaced from their intended positions.

In conventional electron-beam lithography systems, a laser interferometer controls the

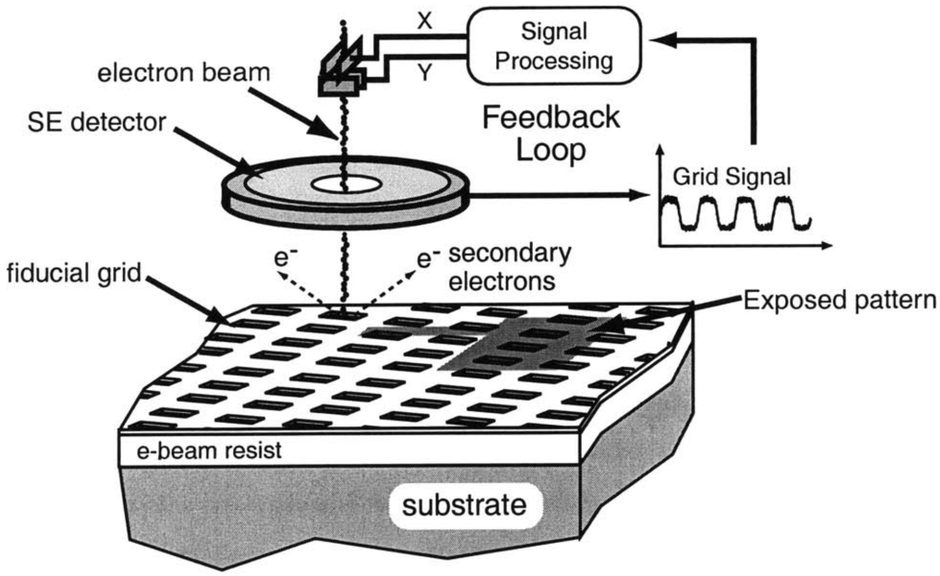


Figure 1-4: Schematic of spatial-phase locked electron-beam lithography.

stage that moves the substrate into the scan field. Though the positioning of the stage can be determined to within a fraction of a nanometer, the beam is never directly monitored and is free to drift. Only the stage position is directly monitored with no feedback to verify the actual position of the beam. For this type of open-loop system, small perturbations cause large distortions, especially over a long period of time when the perturbations add up.

1.3 Spatial-Phase-Locked Electron-Beam Lithography Overview

Spatial-phase-locked electron-beam lithography offers a novel approach to achieving nanometer pattern-placement accuracy. SPLEBL adds feedback to the control loop to directly monitor and correct the beam's position. A Raith 150 scanning electron-beam lithography system has been modified to employ spatial-phase locking [9]. As show in Figure 1-4, The substrate is covered with non-perturbing, aluminum global fiducial grid, used solely as a diagnostic reference. The grid is virtually electron transparent and does not interfere with pattern writing. The beam raster scans along the grid-covered substrate while a beam blanker modulates the beam to write a pattern.

When struck by the electron beam, both the grid and the resist emit secondary electrons. However, the secondary-electron yield of the grid is much higher than that of the resist. The secondary-electron signal is periodic with fundamental temporal frequencies directly related to the x- and y-spatial frequencies of the grid. The phases corresponding to each of these fundamental frequencies are extracted via digital signal processing and used to compute the beam-placement error. The relationship between these phases and placement error is derived later in this thesis. The x- and y-position correction signals are then calculated and sent back to the Raith pattern generator. Any deviation of the beam from its desired location on the substrate is detected, and the correction signals are fed back to the beam control electronics to compensate the errors in the beam position. In this manner, the locations of the patterns are directly registered to the fiducial grid on the substrate.

1.4 Spatial-Phase Locking Subsytem

The spatial-phase locking (SPL) subsystem extracts the phase from the secondary-electron signal, and uses it to calculate the distance that the beam has deviated in both x- and y-directions. Once this error is known, correction signals are generated to steer the beam back to its intended location. The system block diagram in Figure 1-5 shows how the Raith 150 SEBL tool has been modified to implement SPLEBL. This same diagram can be generalized to show how any typical SEBL tool can be adapted for SPLEBL by adding a SPL subsytem. Once the secondary-electron signal is detected, it is amplified and sent to the SPL subsystem which generates x- and y-position correction signals. The pattern generator adds the correction signals to the intended x- and y-coordinates so that the x- and y-deflection coils will stear the beam to the correct location on the substrate. The actual pattern is produced when the beam blanker modulates the beam current to a level that exposes the resist on the substrate. In this manner patterns are written in a linear, serial process across a full field. The basic functional structure of the SPL subsystem is illustrated

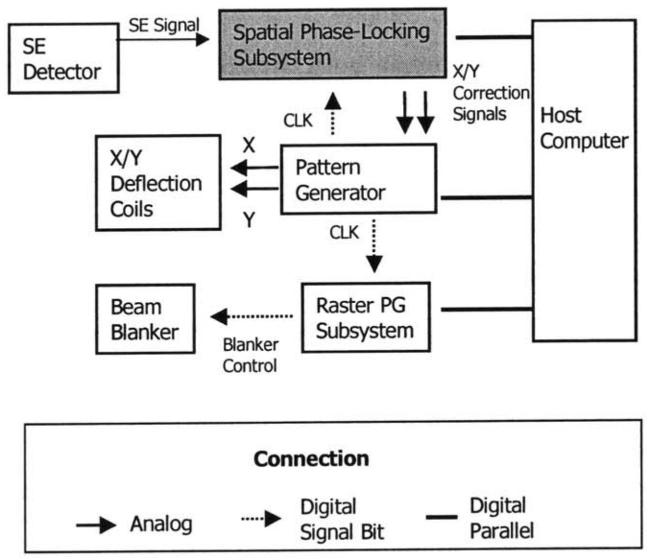


Figure 1-5: SEBL system block diagram adapted for SPLEBL.

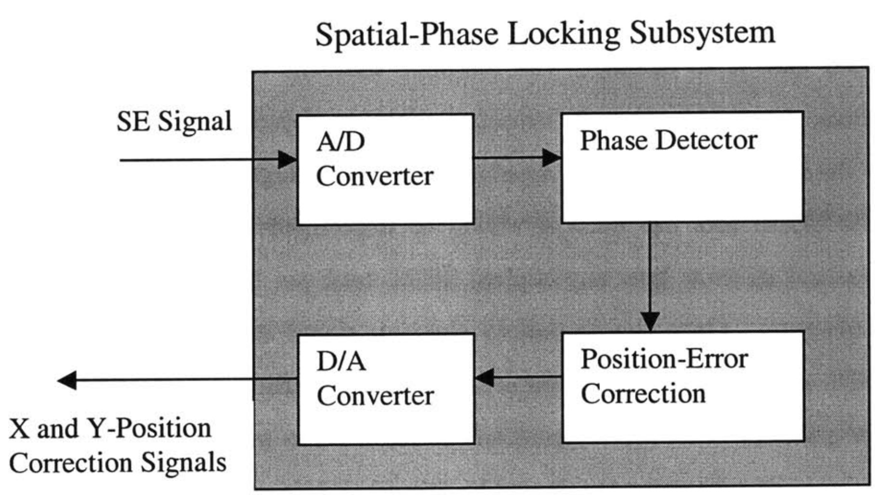


Figure 1-6: Spatial-phase locking subsystem functional block diagram.

in Figure 1-6. The SPL subsystem is a digital circuit that performs two main functions, phase detection and position-error correction. Because the secondary-electron signal and the x- and y-correction signals are inherently analog, A/D and D/A converters interface the SPL subsystem to the secondary-electron detector and pattern generator blocks.

1.4.1 Design Procedure

The design flow of the spatial-phase locking subsystem is illustrated in Figure 1-7. The design procedure begins with the algorithm development in which possible algorithms are designed and simulated. Based on the simulation results, the most suitable algorithm is chosen to be implemented with hardware. The hardware development begins with describing the algorithm at the system level that consists of the basic functional blocks of the algorithm and how they fit together. In the next stage, the design is modeled with the MathWorks Simulink^{TM} and Xilinx System Generator^{TM} design tools. The functional blocks from the system-level abstraction are reduced to primitive hardware building blocks, contained in the Simulink^{TM} and System Generator^{TM} libraries. At the schematic level, the circuit design can easily be optimized, simulated, and evaluated. The System Generator^{TM} tool converts the schematic design into a hardware description language (HDL) that could later be used to create dedicated hardware in the form of a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC).

The focus of this thesis does not include the programming of a physical FPGA or the fabrication of an ASIC. While these actions constitute the next step, they are beyond the scope of this thesis which focuses on the hardware development for a general SEBL system rather than the realization of the design for a specific system. Though the basic functionality of the design would remain the same, the hardware would need to be customized for each SEBL machine. This thesis will concentrate on the algorithm and the hardware fundamental to that algorithm, circumventing engineering issues that are specific to each SEBL machine.

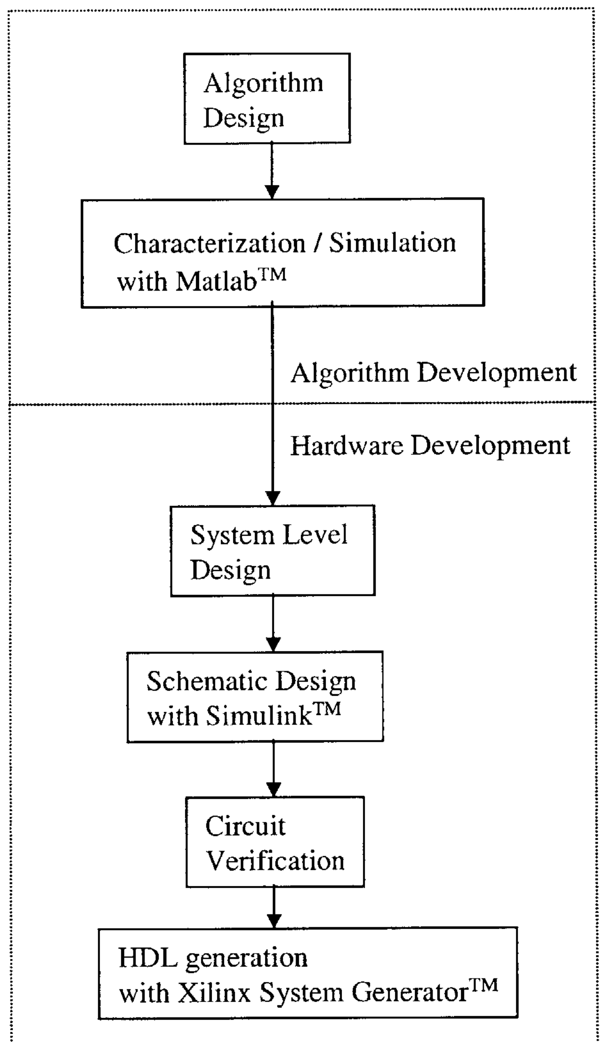


Figure 1-7: Design flow of the spatial-phase locking subsystem.

Chapter 2

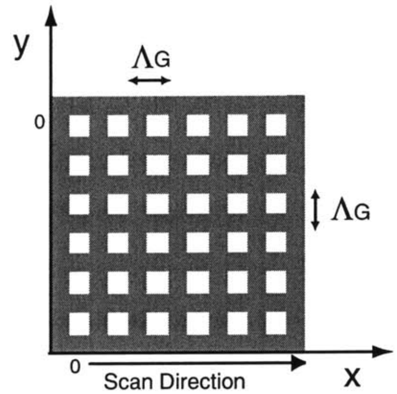
Error-Detection Algorithm

As the electron-beam is raster scanned along the grid-covered substrate, a secondary-electron signal is emitted with a temporal frequency that corresponds to the spatial frequency of the fiducial grid. The deviation of the beam from its intended position causes a change in phase of the secondary-electron signal. Once the phase of the signal is extracted, the x and y-position errors can be calculated from the phase-error values. Correction signals are then generated and sent to the pattern generator to cancel out the position errors. The whole system thus acts as a closed loop.

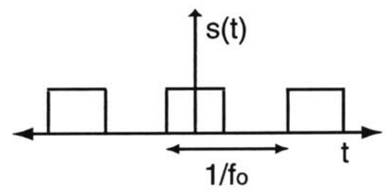
The speed and accuracy at which the phase can be extracted and processed relates directly to the loop bandwidth, which in turn determines the disturbance-rejection capabilities of SPLEBL. The loop bandwidth is ultimately determined by the computation speed of the error-correction and also by the signal-to-noise ratio (SNR) of the secondary-electron signal. The need for a very accurate, computationally efficient phase-detection algorithm arises as the SNR of the system is improved.

2.1 The Secondary-Electron Signal

As the high-energy electron beam impinges on the grid-covered substrate, a periodic secondary-electron signal is emitted. The phase of the signal directly corresponds to the deviation of



(a)



(b)

Figure 2-1: The reference grid pattern for SPLEBL shown in (a) with grid axes aligned with the scan field axes. (b) shows the secondary-electron signal modeled as a temporal square wave.

the electron beam's position. The relationship between this phase and placement error will now be derived.

Assume that the grid is perfectly square with spatial period Λ_G and that the grid axes align with the scan-field axes, as illustrated in Figure 2-1(a). As the beam scans left to right along the x-direction, the emitted secondary-electron signal s(t) will resemble a square wave, as shown in Figure 2-1(b), with temporal frequency f_o (in Hz) equal to beam velocity v_b divided by the grid period, Λ_G, so that

$$f_o = \frac{v_b}{\Lambda_G}. \tag{2.1}$$

This same signal can be mapped to the spatial domain by scaling f_o by $\frac{1}{v_b}$.

Let $s(x, y)$ represent the two-dimensional periodic signal in the spatial domain, and let k_o be the spatial frequency of the grid in radians such that $k_o = \frac{2\pi}{\Lambda_G}$. The signal has a Fourier series:

$$s(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} a_{m,n} e^{jk_o(mx+ny)}. \tag{2.2}$$

If the beam scans left to right in the x-direction at a fixed y-position (y = 0), the secondary-

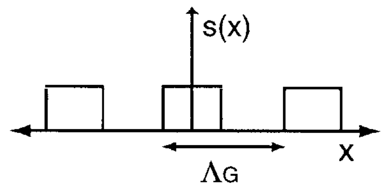


Figure 2-2: The secondary-electron signal modeled as a square wave in the spatial domain.

electron signal has the Fourier series:

$$s(x,0) = \sum_{m=-\infty}^{\infty} a_m e^{jk_o mx} \tag{2.3}$$

with Fourier coefficients, a_m, given by

$$a_m = \frac{1}{\Lambda_G} \int_{\Lambda_G} s(x) e^{-jk_o mx} dx. \tag{2.4}$$

As in the time domain, the signal will resemble a square wave in the spatial domain with periodicity Λ_G, shown in Figure 2-2. The Fourier series coefficients for s(x) are determined to be

$$a_m = \frac{1}{\Lambda_G} \int_{-\frac{\Lambda_G}{4}}^{\frac{\Lambda_G}{4}} e^{-jk_o mx} dx \tag{2.5}$$

$$= \frac{\sin(m\pi/2)}{\pi m}$$

$$= \frac{1}{2}\left(\frac{\sin(m\pi/2)}{m\pi/2}\right). \tag{2.6}$$

Using the property, $\lim_{x\to\infty} \frac{\sin(x)}{x} = 1$, the first few Fourier coefficients of s(x) are found to be $a_0 = 1/2$, $a_{\pm1} = 1/\pi$, $a_{\pm2} = 0$, and $a_{\pm3}$ = -1/(3π).

As the order m increases, the corresponding coefficients become much smaller. The higher order terms with $|m| > 1$ can be neglected because phase estimation on the first-order coefficients will be more accurate. Thus the signal of interest is a truncated version of the original signal:

$$s_{trunc}(x) = a_0 + a_1 cos(k_o x). \tag{2.7}$$

The actual position of the beam, given in (x, y) coordinates, will deviate by an amount Δx and Δy from the intended position, with coordinates (x_p, y_p):

$$x = x_p + \Delta x, \tag{2.8}$$

$$y = y_p + \Delta y. \tag{2.9}$$

Substituting (2.8) into (2.7), the signal is expressed in terms of x_p,

$$s_{trunc}(x_p) = a_0 + a_1 \cos(k_o x_p + k_o \Delta x). \tag{2.10}$$

Thus the position error in the x-direction, Δx, is contained in the phase of the first-order Fourier component of the signal at k_o, as the beam scans in the x-direction. Likewise, the position error in the y-direction, Δy, is contained in the phase of the first-order coefficient as the beam steps in the y-direction after finishing a scan along the x-direction. Following the same procedure above, but fixing the x-position (x = 0), the y-component of the secondary electron signal will resemble a square wave, with period N$\cdot\Lambda_G$ where N is the number of grid periods in a single scan field. A whole line would need to be scanned before any correction in the y-direction could be performed. Fixing x and truncating the signal, s(y) is written in terms of y_p,

$$s_{trunc}(y_p) = a_0 + a_1 \cos(N k_o y_p + N k_o \Delta y). \tag{2.11}$$

With the grid axes aligned to the scan field axes, multiple scans would be required to detect the phase in the y-direction. Before a single y-position data point could be collected, a full scan in the x-direction would need to be completed. In this scenario, error in the y-direction would accumulate considerably while waiting for each scan to finish. A more optimal scheme is to detect both x- and y-data simultaneously by rotating the grid axes with respect to the scan-field axes.

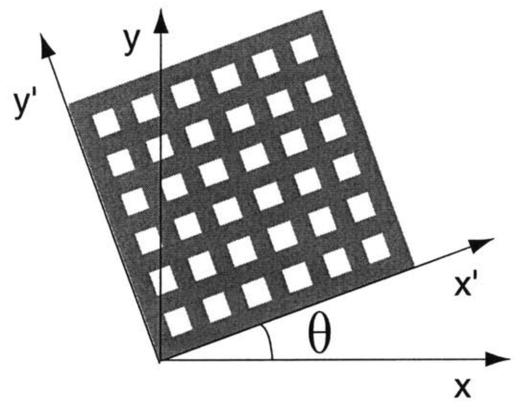


Figure 2-3: The reference grid axes (x', y') are rotated by θ with respect to the scan field axes (x, y).

2.1.1 Simultaneous Phase-Detection in X and Y

In Figure 2-3, the grid axes are rotated with respect to the scan-field axes by an angle θ (where $0 < \theta < \frac{\pi}{4}$) to enable the phases in both x and y-directions to be detected simultaneously. Once the phases are detected for each direction, the x and y-position errors are calculated from the phase error values. Let $\hat{\mathbf{a}} = \begin{bmatrix} x \\ y \end{bmatrix}$ be the original grid axes aligned with the scan-field axes. The new grid axes $\hat{\mathbf{a}}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ are formed by rotating $\hat{\mathbf{a}}$ counter-clockwise by an angle θ using rotation matrix R_θ:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

The rotated grid axes and the scan field axes are related by the following expression:

$$\hat{\mathbf{a}}' = R_\theta \hat{\mathbf{a}}. \tag{2.12}$$

Thus the rotated-grid axes are

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta && (2.13)\\ y' &= -x\sin\theta + y\cos\theta. && (2.14)\end{aligned}$$

The truncated signal in the rotated-grid coordinate system is

$$s_{rot}(x',y') = a_0 + a_{1y'}\cos(k_o y') + a_{1x'}\cos(k_o x'). \qquad (2.15)$$

For simplicity, $a_{1x'}$ is re-written as a_{HI} and $a_{1y'}$ as a_{LO}. Substituting the expressions for x' and y' into (2.15) and letting $k_{LO} = k_o\sin\theta$ and $k_{HI} = k_o\cos\theta$, the signal is expressed in terms of the scan-field coordinates (x, y) as

$$s_{rot}(x,y) = a_0 + a_{LO}\cos(-k_{LO}x + k_{HI}y) + a_{HI}\cos(k_{HI}x + k_{LO}y). \qquad (2.16)$$

The rotation angle θ cannot be too small or k_{LO} will approach zero. Furthermore, as θ approaches $\pi/4$, k_{Lo} and k_{HI} become indistinguishable because they both will equal the same value, $\sqrt{2}k_o/2$. A good choice for θ is a value between 20 and 30 degrees.

Substituting the equations (2.8) and (2.9) for x and y into (2.16), the signal is given in x_p and y_p coordinates by

$$\begin{aligned} s_{rot}(x_p,y_p) = a_0 &+ a_{LO}\cos(k_{LO}x_p + k_{LO}\Delta x - k_{HI}y_p - k_{HI}\Delta y)\\ &+ a_{HI}\cos(k_{HI}x_p + k_{HI}\Delta x + k_{LO}y_p + k_{LO}\Delta y). \qquad (2.17)\end{aligned}$$

By taking the Fourier transform of the signal with respect to x_p, it is apparent that the position errors, Δx and Δy, are directly related to the phase at the two fundamental frequencies, k_{LO} and k_{HI}. The Fourier transform for the general continuous-time function, $f(x) = \cos(k_o t + \phi)$, is

$$F(k) = \pi[\delta(k-k_o)e^{j\phi} + \delta(k+k_o)e^{-j\phi}]. \qquad (2.18)$$

Applying (2.18), the Fourier transform of (2.17) is found to be

$$\begin{aligned} S_{x_p}(k, y_p) = {} & 2\pi a_0 \delta(k) + \pi a_{LO}\left[\delta(k - k_{LO})e^{j(k_{LO}\Delta x - k_{HI}y_p - k_{HI}\Delta y)}\right] \\ & + \pi a_{LO}\left[\delta(k + k_{LO})e^{-j(k_{LO}\Delta x - k_{HI}y_p - k_{HI}\Delta y)}\right] \\ & + \pi a_{HI}\left[\delta(k - k_{HI})e^{j(k_{HI}\Delta x + k_{LO}y_p + k_{LO}\Delta y)}\right] \\ & + \pi a_{HI}\left[\delta(k + k_{HI})e^{-j(k_{HI}\Delta x + k_{LO}y_p + k_{LO}\Delta y)}\right]. \end{aligned} \tag{2.19}$$

$S_{x_p}(k, y_p)$ contains a DC term and two fundamental frequency components at k_{LO} and k_{HI} with respective phases, ϕ_{LO} and ϕ_{HI}, where

$$\phi_{LO} = k_{LO}\Delta x - k_{HI}y_p - k_{HI}\Delta y, \tag{2.20}$$

$$\phi_{HI} = k_{HI}\Delta x + k_{LO}y_p + k_{LO}\Delta y. \tag{2.21}$$

The placement-error is contained in the phases, ϕ_{LO} and ϕ_{HI} at the two spatial frequencies, k_{LO} and k_{HI}. For any given phase estimate, the spatial frequencies and y_p are known; therefore, the error-placement contributions in both x and y-directions can be directly calculated as

$$\Delta x = \frac{k_{LO}\phi_{LO} + k_{HI}\phi_{HI}}{k_{LO}^2 + k_{HI}^2}, \tag{2.22}$$

$$\Delta y = \frac{k_{LO}\phi_{HI} - k_{HI}\phi_{LO} - y_p(k_{LO}^2 + k_{HI}^2)}{k_{LO}^2 + k_{HI}^2}. \tag{2.23}$$

2.2 Heterodyne Phase-Detection Technique

Now that the placement-error has been shown to be contained in the phase, an appropriate method for extracting the phase accurately and efficiently is needed. Because the signal is periodic, with known fundamental frequencies, a heterodyne phase-detection technique [10] is a good candidate. Figure 2-4 illustrates how a heterodyne phase detector extracts the phase ϕ from the f_o-frequency component of a continuous-time input s(t) with amplitude A,

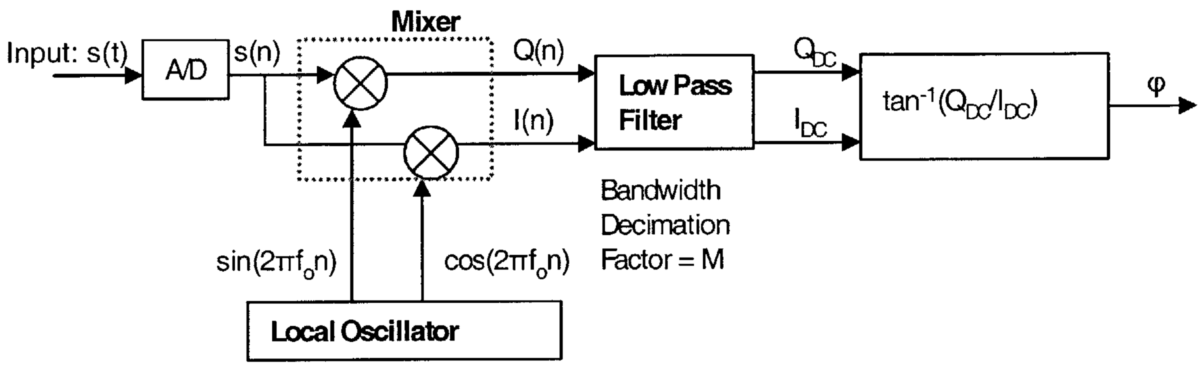


Figure 2-4: Block diagram of the heterodyne phase-detector.

where

$$s(t) = A\cos(2\pi f_o t + \phi). \tag{2.24}$$

The continuous-time signal is converted into samples by an A/D converter equipped with an anti-aliasing filter. Each sample of the digitized signal, s(n), is generated at a rate equal to the A/D clock frequency. The subsequent mixing, demodulation, and filtering functions of the phase detector are performed by three major functional blocks: a local oscillator, a mixer, and a low-pass filter.

2.2.1 Local Oscillator

The local oscillator is a direct digital frequency synthesizer that generates two outputs, a sine and a cosine, both with zero phase. The digital samples from the local oscillator are generated at a sampling rate equal to the A/D clock frequency. The digitized input signal, s(n), is then multiplied by each of the sine and cosine outputs of the local oscillator using a mixer.

2.2.2 Mixer

The mixer converts the f_o-frequency component of the input signal, $s(n)$, to baseband, as shown in Figure 2-5, by multiplying it by either of the local oscillator outputs. As an example

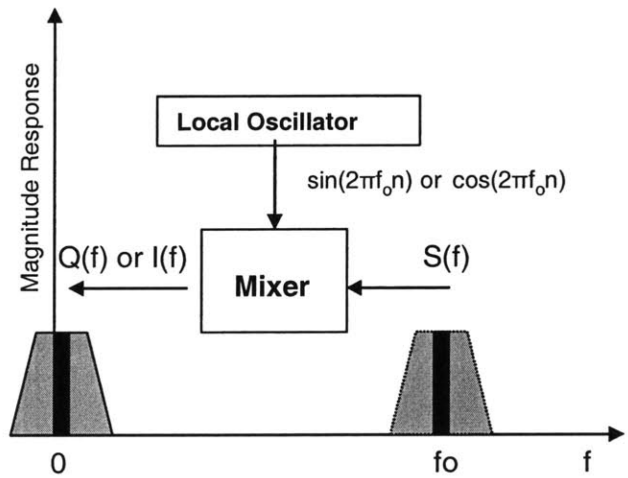


Figure 2-5: The mixer translates the input signal at f_o to baseband.

let s(n) = Acos($2\pi f_o n + \phi$), then the result of the multiplication with the sine and cosine outputs of the local oscillator is shown by the following trigonometric identities,

$$\cos\alpha\cos\beta = \frac{1}{2}\left[\cos(\alpha-\beta)+\cos(\alpha+\beta)\right],$$
$$\cos\alpha\sin\beta = \frac{1}{2}\left[-\sin(\alpha-\beta)+\sin(\alpha+\beta)\right].$$

Setting $\alpha = 2\pi f_o$n + ϕ and $\beta = 2\pi f_o$n, the outputs of the mixer, I(n) and Q(n), are

$$I(n) = \frac{1}{2}A\left[\cos\phi+\cos(4\pi f_o n+\phi)\right], \tag{2.25}$$
$$Q(n) = \frac{1}{2}A\left[-\sin\phi+\sin(4\pi f_o n+\phi)\right]. \tag{2.26}$$

Outputs I(n) and Q(n), often termed "in-phase" and "quadrature," contain a DC component (directly related to ϕ) and higher-frequency terms. The in-phase and quadrature DC terms

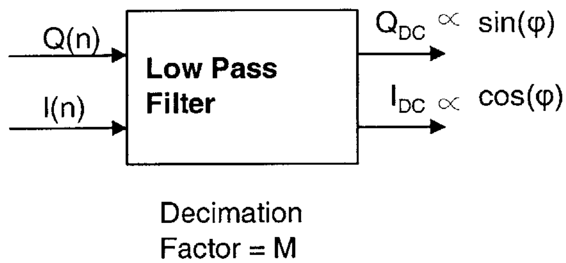


Figure 2-6: The low pass filter extracts the DC components of I(n) and Q(n). The DC components, I_{DC} and Q_{DC}, are proportional to the sine and cosine of the phase, ϕ.

are

$$
\begin{aligned}
I_{DC} &= \frac{A}{2}\cos(\phi), \\
Q_{DC} &= \frac{A}{2}\sin(\phi).
\end{aligned}
\tag{2.27}
$$

Once the DC components in (2.27) are extracted from I(n) and Q(n) by means of a low-pass filter, an inverse tangent operation can be applied to yield the phase,

$$\phi = -arctan\left(\frac{Q_{DC}}{I_{DC}}\right). \tag{2.28}$$

2.2.3 Low-Pass Filter

The output of the mixer consists of sum and difference frequencies in the sampled data spectrum. The higher order components must be removed to recover the baseband signal containing the phase information. The DC components of I(n) and Q(n) are extracted by passing them through a decimating low-pass digital filter, as shown in Figure 2-6. The filter reduces the original sampling rate of the input sequence to a lower rate, through a decimation process that averages data points. In SPLEBL, the input signal is periodic therefore its DC component can be extracted by averaging over a full period or multiple periods. Set M = kN where N is the period of the signal and k is an integer greater than zero. The in-phase and quadrature DC terms can be expressed as

$$I_{DC} = \frac{1}{M}\sum_{n=1}^{M} I(n), \tag{2.29}$$
$$Q_{DC} = \frac{1}{M}\sum_{n=1}^{M} Q(n).$$

The sampling rate of the filtered signal will be reduced to a rate equal to a sub-multiple of the pattern generator clock frequency. The sampling rate of the filtered output f_{out} is equal to the input sampling rate f_s (of the A/D converter) divided by the decimation factor M,

$$f_{out} = \frac{f_s}{M}. \tag{2.30}$$

The equation for Q_{DC} and I_{DC} in (2.29) is equivalent to

$$I_{DC} = \frac{1}{M}\sum_{n=<M>} s(n)\cos(2\pi f_o n), \tag{2.31}$$
$$Q_{DC} = \frac{1}{M}\sum_{n=<M>} s(n)\sin(2\pi f_o n). \tag{2.32}$$

Notice that I_{DC} and Q_{DC} in (2.31) and (2.32) are simply the real and imaginary parts of the discrete-time Fourier transform of the sampled input signal taken at a single frequency, f_o.

2.3 Heterodyne Phase Detection Applied to SPLEBL

When the x- and y-position errors are detected simultaneously by rotating the grid axes by an angle θ with respect to the scan field axes, the detected secondary-electron signal will contain two fundamental temporal frequencies, f_{LO} and f_{HI}. Recall that fundamental spatial frequencies, k_{LO} and k_{HI}, are defined as $k_{LO} = k_o \sin\theta$ and $k_{HI} = k_o \cos\theta$, where $k_o = 2\pi/\Lambda_G$ is the spatial period of the grid in radians per meter. Using the conversion relation (2.1), these temporal frequencies are expressed in terms of the fundamental spatial frequencies,

$$f_{LO} = \frac{k_{LO} v_b}{2\pi}, \qquad (2.33)$$
$$f_{HI} = \frac{k_{HI} v_b}{2\pi},$$

where v_b is a known beam velocity. The feedback signal is expressed in the time-domain as

$$s(t) = a_0 + a_{HI}\cos(2\pi f_{HI} t + \phi_{HI}) + a_{LO}\cos(2\pi f_{LO} t + \phi_{LO}), \qquad (2.34)$$

where ϕ_{LO} and ϕ_{HI} are given by (2.20) and (2.21), respectively.

The schematic in Figure 2-7 shows how heterodyne phase detection can be applied to SPLEBL to extract the phases, ϕ_{LO} and ϕ_{HI}, from the f_{LO}- and f_{HI}-frequency components of the secondary-electron signal. This scheme employs two heterodyne phase detectors, one to extract ϕ_{LO} and the second to extract ϕ_{HI}. The first branch that extracts ϕ_{LO} is now analyzed in further detail.

Suppose s(t) in (2.34) is the secondary-electron input to the A/D converter, and s(n) represents the digitized output. Also, set the frequency of the local oscillator to f_{LO} such that its sinusoidal outputs are $\sin(2\pi f_{LO} n)$ and $\cos(2\pi f_{LO} n)$. Each of the output products of the mixer, I_{LO}(n) and Q_{LO}(n), will contain a DC components, $\cos\phi_{LO}$ and $-\sin\phi_{LO}$, directly related to ϕ_{LO} plus higher-order terms,

$$I_{LO}(n) = \frac{1}{2}[\cos\phi_{LO} + \cos(2f_{LO}n + \phi_{LO}) \qquad (2.35)$$
$$+ \cos((f_{LO} - f_{HI})n + \phi_{LO} - \phi_{HI}) + \cos((f_{LO} + f_{HI})n + \phi_{LO} + \phi_{HI})],$$
$$Q_{LO}(n) = \frac{1}{2}[-\sin\phi_{LO} + \sin(2f_{LO}n + \phi_{LO}) \qquad (2.36)$$
$$- \sin((f_{LO} - f_{HI})n + \phi_{LO} - \phi_{HI}) + \sin((f_{LO} + f_{HI})n + \phi_{LO} + \phi_{HI})].$$

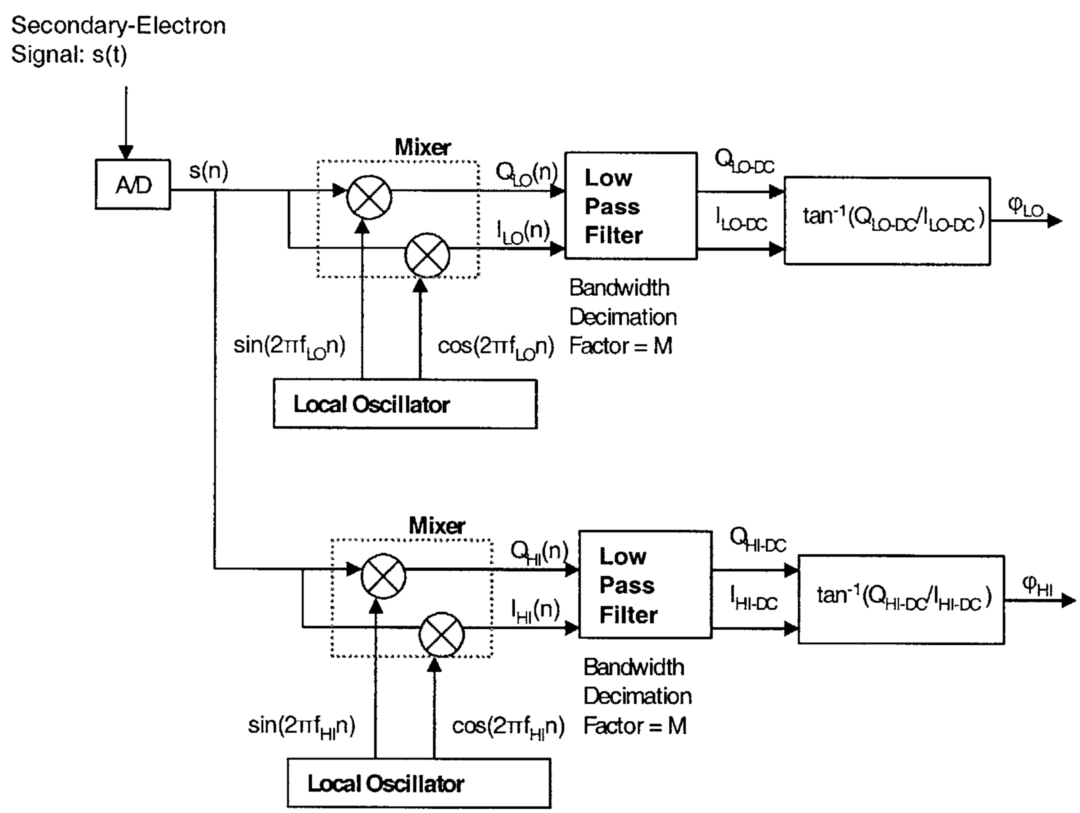


Figure 2-7: The heterodyne phase technique adapted to SPLEBL extracts the phases, ϕ_{LO} and ϕ_{HI}, from the fundamental frequency components at f_{LO} and f_{HI} of the secondary-electron signal.

The DC values of I_{LO} and Q_{LO} are each calculated by averaging the signal over an integral number of periods, where the period N is given by

$$N = \frac{1}{f_{LO}}. \tag{2.37}$$

An inverse tangent function is applied to the DC components, I_{LO-DC} and Q_{LO-DC}, to yield the phase estimate, ϕ_{LO}.

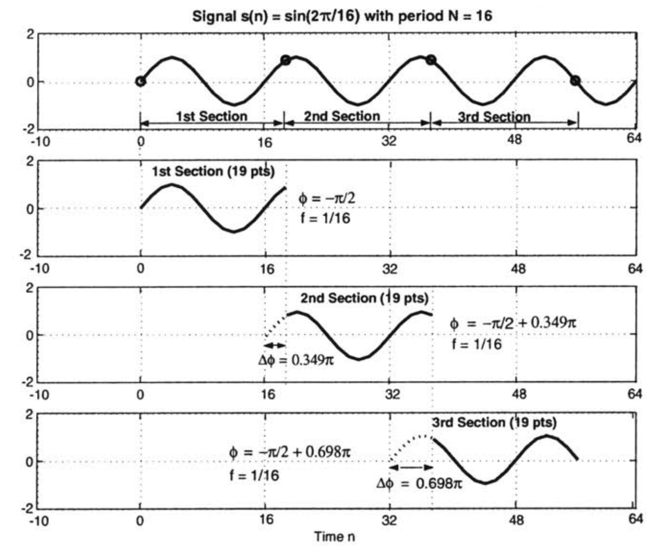
The number of averaged points must be exactly equal to an integer times the period, or else an error will be accumulated into the phase estimate. Figure 2-8, shows the phase-estimate error accumulation that arises when the number of sample points is not equal to an integral period. Though the phase of the signal ($\sin(2\pi n/16)$ with period N = 16) is constant, it appears to be changing by $\Delta\phi$ for each section of 19 samples. When an integral period of sample points are taken however, $\Delta\phi$ is equal to zero as it should be.

2.3.1 Phase Detection Adapted for an Arbitrary Sample Number

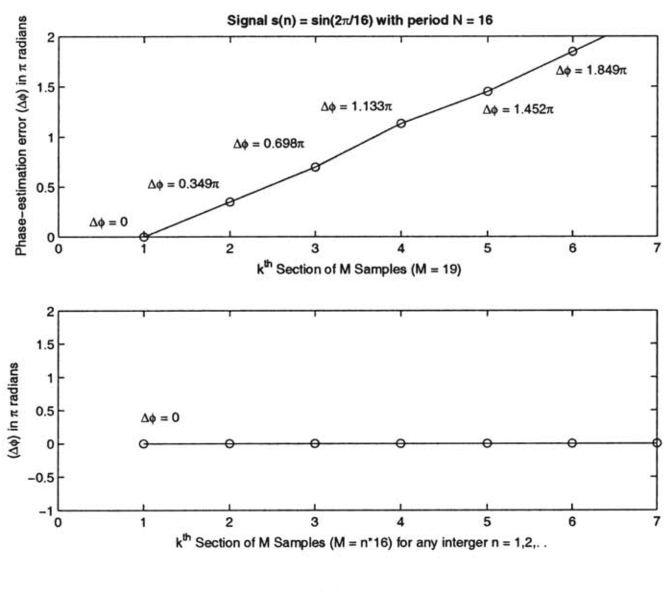
Requiring that the number of samples equal an integral number of periods constrains the choice of rotation angle θ, beam velocity, and scan-field size. Moreover, position-error correction is delayed during the time in which the full number of sample points are collected. These limitations create a need for a phase-detection algorithm that can accomodate an arbitrary sample number.

The first heterodyne phase detection scheme described in this section detects the phase at a predetermined frequency ω_o equal to the fundamental frequency of the signal. In the Fourier domain, the power spectrum of the signal will have a peak at the fundamental frequency provided the phase is constant. When the phase changes in time however, the peak frequency of the signal, ω_{pk}, will vary slightly as the phase, $\phi(t)$, changes at a given time t:

$$\omega_{pk} = \omega_o + \Delta\omega(t), \tag{2.38}$$



(a)



(b)

Figure 2-8: A sinusoidal signal with constant phase $\phi = -\frac{\pi}{2}$ and period N = 16, is (a) divided into section of non-integral-period samples (19 pts each). (b) shows the phase-estimation error accumulation from section to section as a result of sampling a non-integral period of points (M = 19) and that this error disappears when an integral period of samples is taken (M = n·16) where n is an integer.

where $\Delta\omega(t)$ is the frequency offset from ω_o, the expected peak frequency value [11]. A sinusoidal signal with a time-varying phase is given by s(t) = $\cos(\omega_o t + \phi(t))$. The instantaneous phase of s(t) is defined as

$$\theta_i(t) = w_o t + \phi(t) \tag{2.39}$$

and the instantaneous frequency, $\omega_i(t)$, is defined as

$$\begin{aligned} \omega_i(t) &= \frac{d\theta_i}{dt} \\ &= \omega_o + \frac{d\phi}{dt}. \end{aligned} \tag{2.40}$$

Comparing expressions (2.38) and (2.40), $\omega_i(t)$ is equal to $\omega_{pk}(t)$, the peak frequency at time t, and

$$\frac{d\phi}{dt} = \Delta\omega(t) \tag{2.41}$$

$$= \omega_{pk}(t) - \omega_o. \tag{2.42}$$

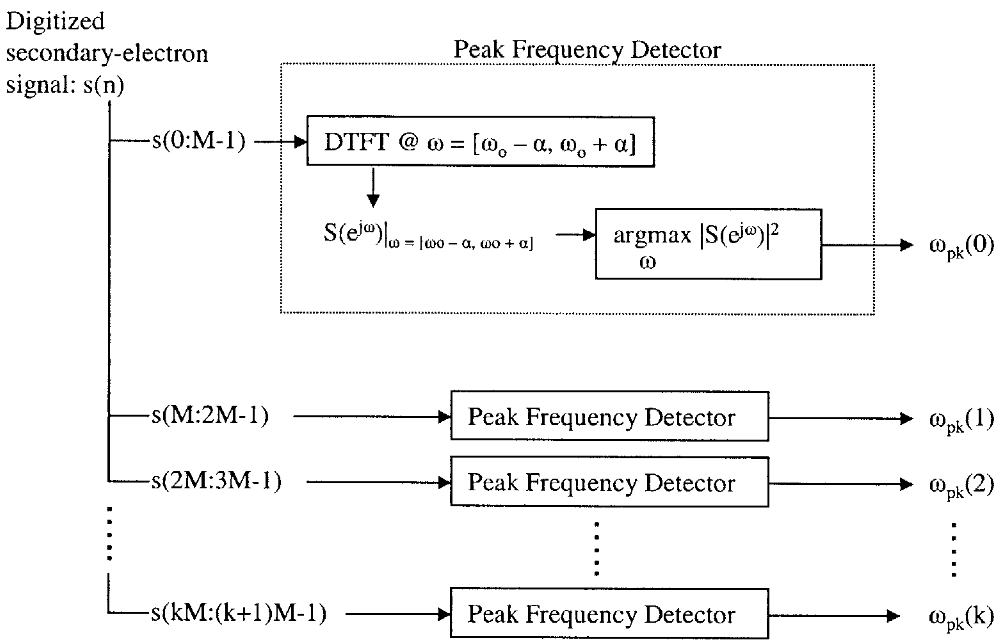
The phase at time t can be estimated from the change in the peak frequency as

$$\phi(t) = \int_0^t (\omega_{pk}(t) - \omega_o)\, dt, \tag{2.43}$$

and in the discrete-time domain this expression becomes

$$\phi(k) = \sum_{n=0}^{k} \omega_{pk}(n) - \omega_0. \tag{2.44}$$

Instead of determining the phase directly with the heterodyne method, the phase can be found indirectly from summing the change in peak frequency over k sections of M samples, as shown in Figure 2-9. Sections of M samples of s(n) are collected and their discrete-time Fourier Transform are taken for a small range of frequencies, $\omega_o - \alpha$ to $\omega_o + \alpha$. The Fourier Transform is then multiplied by its complex conjugate to form a periodogram from which the peak frequency is determined. The difference between the actual peak frequency, ω_{pk}, and



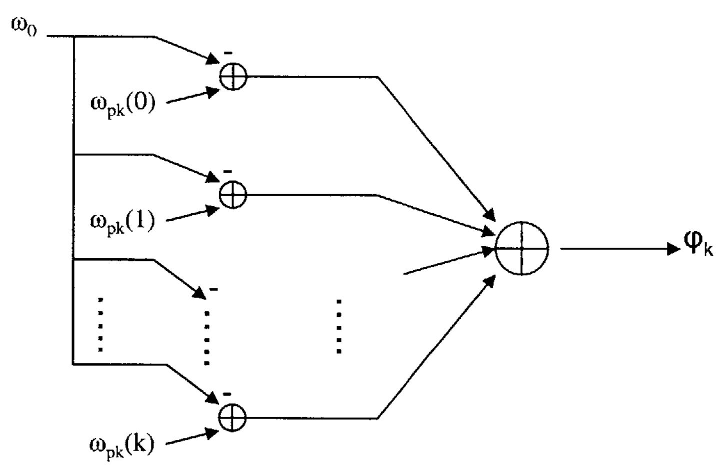


Figure 2-9: The phase of the secondary-electron signal is detected indirectly by monitoring the change in the frequency at which the peak of the signal power spectrum lies.

the fundamental frequency, ω_o, of the consecutive sections are summed to yield the phase. In this scheme there is no constraint on M as illustrated in Figure 2-8(a), whereas for the standard heterodyne phase detection, M was required to be equal to an integral number of periods. Because it does not depend on sample number, indirect phase estimation appears to be more robust than straightforward heterodyne phase detection. However, the ultimate conclusion will be based upon how each phase-estimation algorithm performs with noise.

Chapter 3

Characterization of the Error-Detection Algorithms

In the previous chapter, a position-error detection scheme using heterodyne phase detection was proposed. Two variations of the method were described, one that takes an integral number of samples, and another that uses an arbitrary number of sample points. This chapter evaluates the performance of the error-detection mechanism by analyzing the speed and accuracy of both phase detection methods.

3.1 Noise Effects

The quality of the detected signal largely determines the accuracy and speed at which placement errors are detected and corrected. Undesired disturbances, known collectively as noise, will distort the signal, thus undermining the accuracy of any taken measurement, such as phase. Noise can be categorized as either random or deterministic depending upon the source. Deterministic noise is caused by an identifiable external source, such as a hum in a loudspeaker, and can usually be eliminated by proper methods such as grounding or shielding. Random noise, on the other hand, is generated by almost everything in nature and cannot be described deterministically. Instead, it is represented with a probability distribution. Random noise consists of a large number, N, of random, independent occurrences and therefore, its probability distribution tends to a Gaussian as $N \rightarrow \infty$, in accordance with the central limit theorem. The noise is modeled as a stochastic process consisting of a collection of independent, Gaussian random variables, $\mathbf{w}(t_1), \mathbf{w}(t_2), ..., \mathbf{w}(t_k)$, defined for a set of finite times, $t_1, t_2, ..., t_k$. At any instance in time, the random noise, $\mathbf{w}(t_m) = \mathbf{w}_m$, will have a Gaussian probability distribution,

$$f_{\mathbf{w}_m}(w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{w^2}{2\sigma^2}\right)} \tag{3.1}$$

where σ^2 is variance of $\mathbf{w}_m$. The mean is zero because random noise contains no DC component. The random noise can be characterized as a white Gaussian noise process having a power spectral density (power per unit frequency), $S_{\mathbf{w}}(f) = \frac{\sigma^2}{2}$, over all frequencies [12].

3.1.1 Signal-to-Noise Ratio

The signal-to-noise ratio (SNR) is defined as the ratio of signal power to the overall noise power for a nominal bandwidth W [12]. Thus for any signal with power P in white Gaussian noise, the SNR γ becomes

$$\gamma = \frac{2P}{\sigma^2}. \tag{3.2}$$

For SPLEBL, the samples of the periodic secondary-electron signal, $\mathbf{y}(n)$, can be modeled as a sinusoidal signal in white Gaussian noise, $\mathbf{w}(n)$, as

$$\mathbf{y}(n) = A\cos(\omega_{\mathbf{o}} n + \phi) + \mathbf{w}(n) \tag{3.3}$$

where $\mathbf{y}$, $\omega_{\mathbf{o}}$, ϕ, and $\mathbf{w}$ are all random variables and $A > 0$. The signal power at the frequency of interest, ω_o, is found to be

$$P = \frac{1}{T}\int_t^{t+T} A^2\cos^2(\omega_o t + \phi) \tag{3.4}$$

$$P = \frac{A^2}{2} \tag{3.5}$$

over the frequency band W $= \omega_o \pm \frac{\delta}{2}$. Combining this result and (3.2), the SNR γ of the sinusoidal signal is

$$\gamma = \frac{A^2}{\sigma^2}. \tag{3.6}$$

3.2 Standard Heterodyne-Phase Estimation with Noise

In the case of SPLEBL, the SNR γ of the secondary electron signal ultimately determines the accuracy of the position-error estimate and the speed at which the feedback loop can follow beam-position disturbances. The variance of the position-error estimate quantifies the accuracy of the error-detection algorithm. From the previous section, the beam-position error in the x-direction is given by

$$\Delta x = \frac{\phi}{k_o} \tag{3.7}$$

for the beam scanning in the x-direction with the grid axes aligned to the scan axes, and with k_o as the spatial frequency of the grid (in radians per meter). The detected signal will have the form given in (3.3) with a temporal frequency, ω_o, equal to k_o scaled by a known beam velocity, v_b,

$$\omega_o = k_o v_b. \tag{3.8}$$

Phase estimation via a heterodyne phase detection scheme is equivalent to a periodogram-based phase estimation [13], with an added assumption that ω_o is deterministic. A periodogram estimator yields the following estimates [13] for frequency and phase of a noisy sinusoidal signal, $\mathbf{y}(n)$, with a discrete-time Fourier transform, $Y(e^{j\omega})$:

$$\hat{\omega_o} = \underset{\omega}{argmax} \left|Y(e^{j\omega})\right|^2, \tag{3.9}$$

$$\hat{\phi} = -\tan^{-1}\left(\frac{Im\left[Y(e^{j\hat{\omega}_o})\right]}{Re\left[Y(e^{j\hat{\omega}_o})\right]}\right), \tag{3.10}$$

where $\hat{\omega_o}$ is the estimated peak frequency of the periodogram ($\left|Y(e^{j\omega})\right|^2$), and $\hat{\phi}$ is the phase estimate at $\hat{\omega_o}$ [13]. For the case of SPLEBL, the peak frequency of the secondary-electron signal periodogram will be equal to one of the fundamental frequencies, ω_{LO} or ω_{HI}, letting ω_o denote either of these two fundamental frequencies. As phase changes with time the peak frequency will vary slightly over a small range ($\omega_o \pm \alpha$). Heterodyne phase estimation assumes that the phase remains fairly constant over this small frequency range such that the phase at the actual peak frequency is essentially equal to the phase at ω_o. The heterodyne phase estimate is given by (3.10) except with $\hat{\omega_o}$, an estimated value, replaced by ω_o, a deterministic number.

Let $\sigma^2_{\hat{\phi}}$ represent the variance of this phase estimate. The variance of the position-error estimate, $\sigma^2_{\widehat{\Delta x}}$, is given by

$$\sigma^2_{\widehat{\Delta x}} = \frac{\sigma^2_{\hat{\phi}}}{k_o^2}, \tag{3.11}$$

using (3.7) with the property that for any constant A and random variables, **X** and **Y**, where **Y** = A**X**, $\sigma^2_Y = A^2\sigma^2_X$.

The variance of an estimator indicates how far its estimates vary from their true values. The more accurate an estimator is, the smaller its variance will be. The smallest acheivable variance for the heterodyne phase estimator is bounded by a lower limit, termed the Cramer-Rao bound, given as

$$\sigma^2_{\hat{\phi}} \geq \sim O\left(\frac{1}{\gamma N}\right), \tag{3.12}$$

where γ is the SNR from (3.6) and N is the data length [13]. The size of N required for the approximation in (3.12) to be valid depends upon the true value of ω_o. As derived in [14], the closer that ω_o is between zero and π (recall that the frequency of any digital signal lies between zero and π), the larger N must be. In other words ω_o must adhere to the following,

$$\frac{\pi}{N} \ll \omega_o \ll \pi\left(1 - \frac{1}{N}\right). \tag{3.13}$$

The variance of the phase estimate of a sinusoidal signal can be no smaller than the Cramer-Rao bound, expressed in order notation[1] on the right side of (3.12). Further-

[1]For functions f(N) and g(N) the order notation f(N) $\sim O(g(N))$ indicates that f(N) grows no faster than g(N), i.e, $\lim_{N\to\infty} \frac{f(N)}{g(N)} < \infty$.

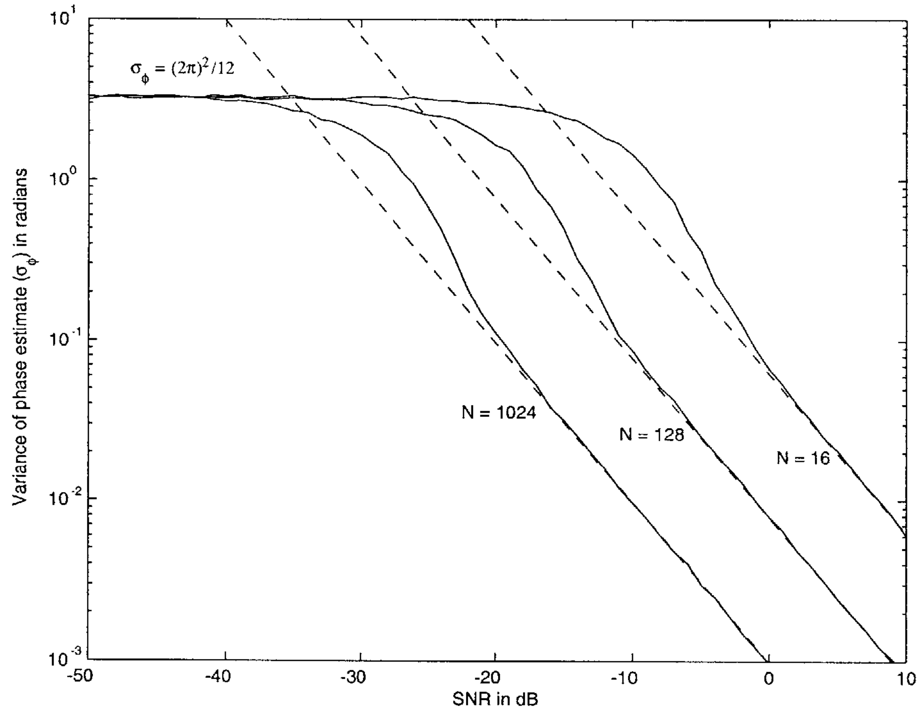


Figure 3-1: Variance of the heterodyne phase estimator of a sinusoid with known frequency and unknown phase in white Gaussian noise as a function of SNR γ for data lengths N = 16, 128, 1024. The dashed lines are the corresponding Cramer-Rao bounds.

more, the closer the variance is to the Cramer-Rao bound, the better the performance of the estimator. Estimators whose variance equals the Cramer-Rao bound are "efficient estimators". For certain ranges of N and γ, the heterodyne phase estimator behaves as an efficient estimator, as shown in Figure 3-1. For instance, when N = 16, the SNR γ must be greater than $\sim$ 4 dB for the relationship in (3.12) to hold. As the data length increases the threshold SNR γ required for efficient estimation decreases. Below the threshold SNR γ the estimator accuracy degrades substantially. At SNR γ far below the threshold the phase-estimate, variances converge to $(2\pi)^2/12$, which is the same variance that would result if ϕ were chosen at random from a range of phase values uniformly distributed between -π and π.

Variable	Description	Value
Λ_G	Grid Period	250 nm
k_o	Grid Spatial Frequency $\left(\frac{2\pi}{\Lambda_G}\right)$	$2\pi \cdot 8$ rad/μm
R	ADC Sample Rate	10 MHz
γ	SNR	-24 dB to -12 dB

Table 3.1: Typical values for Λ_G, k_o, R, and γ.

3.2.1 Bandwidth Considerations

The data length N not only affects the accuracy of the position-error estimates but also the bandwidth of the entire feedback loop. The bandwidth determines the speed at which the feedback loop can cancel out the deviation of the beam's position. If R represents the rate at which the N data samples are collected, according to the Nyquist criterion, the bandwidth, W, is given by

$$W = \frac{R}{2N}. \tag{3.14}$$

Assuming N and SNR γ are sufficient for efficient estimation, the relations in (3.11), (3.12), and (3.14) are combined to give an expression that relates the accuracy of the position-error estimate to the feedback loop bandwidth:

$$\sigma^2_{\widehat{\Delta x}} = \sim O\left(\frac{2}{\gamma R k_o^2}\right) W, \tag{3.15}$$

where $2/\gamma R k_o^2$ is set by physical constraints and therefore allows little variation. Table 3.1 shows typical values for k_o, R, and SNR γ. Note that the range for secondary-electron signal SNR γ is derived in [15] for a 10 kV beam striking a substrate covered with PMMA resist, and an aluminum grid.

A tradeoff between accuracy and speed is evident from (3.15). On one hand we want the variance of the position-error estimate to be as small as possible. On the other, we desire a large bandwidth W. Since both values are proportional to each other however, a compromise must be made given the constraints of the physical parameters, k_o, R, and SNR γ. Figure 3-2 shows the standard deviation of the position-error estimate in white Gaussian

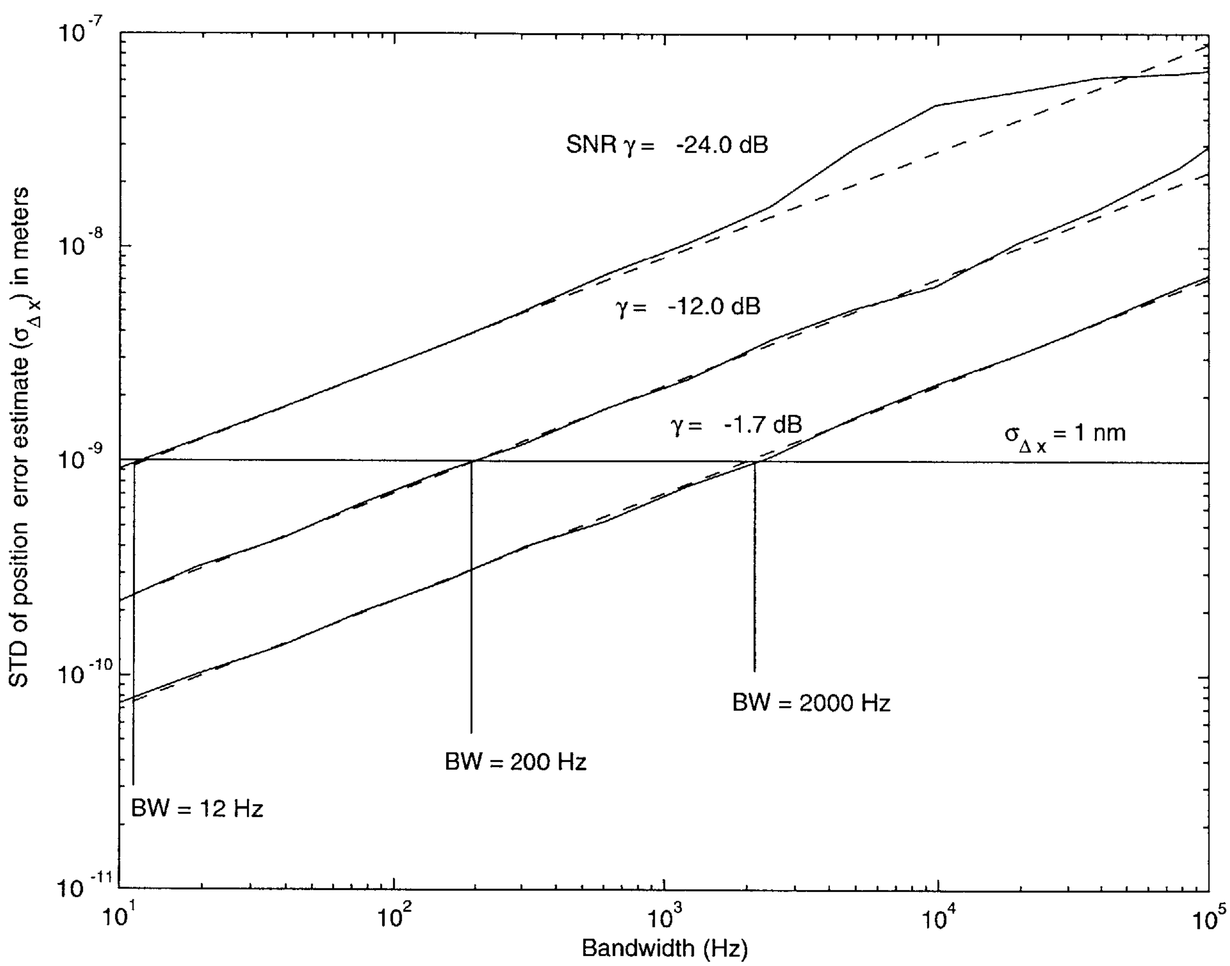


Figure 3-2: The accuracy of the error-detection algorithm as function of bandwidth for SNR γ = -24.0 dB, -12.0 dB, and -1.7 dB.

noise as a function of bandwidth for various SNRs γ. To acheive pattern-placement accuracy at the nanometer level, the beam-position error estimate must also have an accuracy on the order of a nanometer,

$$\sigma^2_{\widehat{\Delta x}} \leq 1nm.$$

When SNR γ ranges between -24 dB and -12 dB, 1nm pattern-placement accuracy is possible only at bandwidths in the range 12 Hz to 200 Hz. Is this bandwidth acceptable? As a rule of thumb, the bandwidth of the feedback loop should be a decade faster than the frequency of the disturbance to accurately cancel the effects of that disturbance [16]. According to [2], mechanical vibrations, electrical charging and stray electromagnetic fields are some of the fastest time-varying disturbances, with frequencies on the order of 100 Hz. These types of short-time perturbations will adversely affect pattern-placement from feature to feature.

Measurements were taken on the vibrational disturbances affecting the SPLEBL setup in the Nanostructures Laboratory at MIT [17]. Figure 3-3 shows the noise power spectrum for the total vibration measured at the top of the Raith EBL chamber. Though the measurements are specific to area and environment, they roughly show which vibrational frequencies have the most impact. The vibrations at 30 Hz, having a magnitude of -93 dB $\cdot$ a.u, correspond to a displacement on the order of 5 nm. The vibrational frequencies that correspond to displacements less than 1 nm will roughly have a magnitude no greater than -113 dB $\cdot$ a.u, which is 20 dB below the 30 Hz magnitude. The vibrations at frequencies below 200 Hz will contribute the most to the beam displacement error; therefore, the bandwidth of the feedback loop should be at least 2 kHz in order for SPLEBL to effectively cancel out noise contributions of up to 200 Hz. According to Figure 3-4, a SNR of at least -1.7 dB would be needed to cancel out disturbances of up to 200 Hz with an error less than or equal to 1 nm.

3.3 Indirect Phase Estimation with Noise

A variation on the heterodyne-phase detection method was presented in Section 2.3.1 in which the phase is determined from the change in peak frequency, rather than calculated

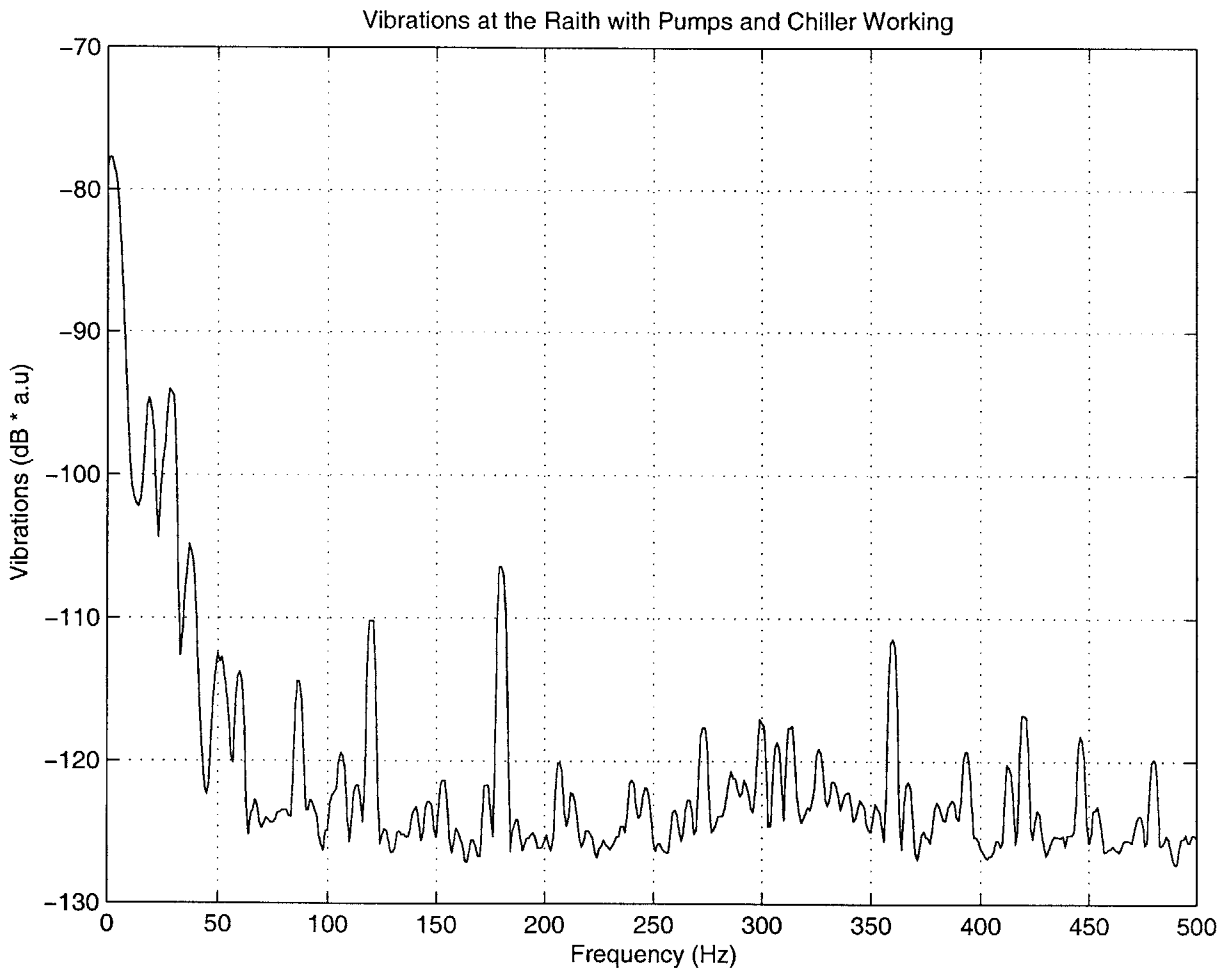


Figure 3-3: Vibrational disturbances at the Raith EBL system (courtesy of Tymon Barwicz).

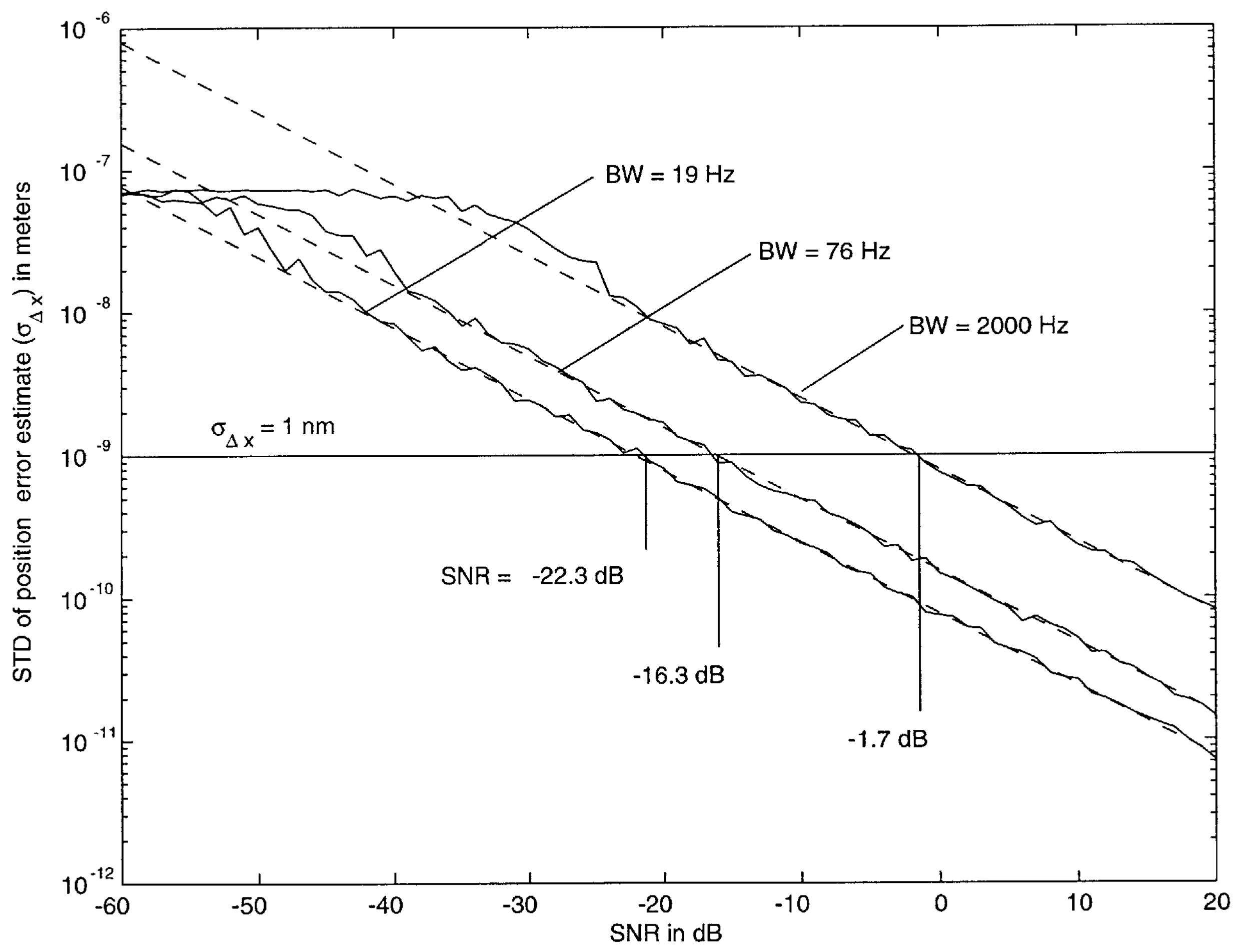


Figure 3-4: Standard deviation of the position-error estimate as a function of SNR γ for various bandwidths. The dashed lines are the corresponding Cramer-Rao bounds.

directly from a section of samples. This scheme has the advantage of being able to accomodate an arbitrary number of samples per section, whereas the standard heterodyne scheme requires that the number of samples per section be equal to an integral period. Using (3.3) to model the fundamental-frequency component of the secondary-electron signal, the peak frequency of the signal is estimated using the periodogram estimator in (3.9), as

$$\hat{\omega}_{pk} = \underset{\omega}{argmax} \left|Y(e^{j\omega})\right|^2, \tag{3.16}$$

by taking the Fourier transform over a small range of frequencies $[\omega_o - \alpha, \omega_o + \alpha]$. Using the relationship between phase and frequency from Section 2.3.1, the phase estimate for the kth section of sample points of the signal is

$$\hat{\phi}(k) = \sum_{n=1}^{k} \hat{\omega}_{pk}(n) - \omega_o, \tag{3.17}$$

where $\hat{\omega}_{pk}$(n) is the peak frequency estimate of the nth section of sample points. When applied to SPLEBL, this phase-estimation technique has a major disadvantage that the change of peak frequency of the signal is so small that very long Fourier transforms are required to detect the change. For any given section of samples, the Fourier transform of that section must be heavily zero-padded to provide the kind of resolution needed. Although a fast Fourier Transform (FFT) could be used to compute the Fourier transforms more efficiently, the calculation would still be very computationally intensive when compared to the standard heterodyne method. Moreover, a higher SNR γ is required for indirect phase estimation to yield the same accuracy as the heterodyne method. Indirect phase estimation will therefore be set aside.

Chapter 4

Translating the Error-Detection Algorithm into Hardware

An error-detection algorithm using heterodyne phase detection has been developed and characterized in the previous chapters. For the algorithm to be of any real use it must be realizable with hardware. The hardware design process, illustrated in Figure 4-1, proceeds in a top-to-bottom fashion in which a simple high-level design representation is broken down into successively more detailed levels. The spatial-phase-locking subsystem is first divided into two major functions, phase detection and position-error calculation. Figure 4-2 shows this first level of abstraction. The phase-detection block is subsequently subdivided into its basic components: a local oscillator, a mixer, and a low-pass filter, which are then modeled with primitive hardware functional blocks (i.e adders, multipliers, lookup tables, etc) provided in the libraries of MathWorks Simulink^{TM} and Xilinx System Generator^{TM} modelling tools. These hardware blocks (called blocksets) provide abstractions of mathematical, logic, memory and DSP functions that can be connected together in the Simulink^{TM} block editor to create a functional model of the algorithm. The System Generator^{TM} blocks are "bit-accurate" meaning that the values they produce in Simulink^{TM} match the corresponding values that would be produced with physical hardware. Moreover the timing of the blocks are identical to their physical hardware counterparts [18]. The System Generator^{TM} blocksets

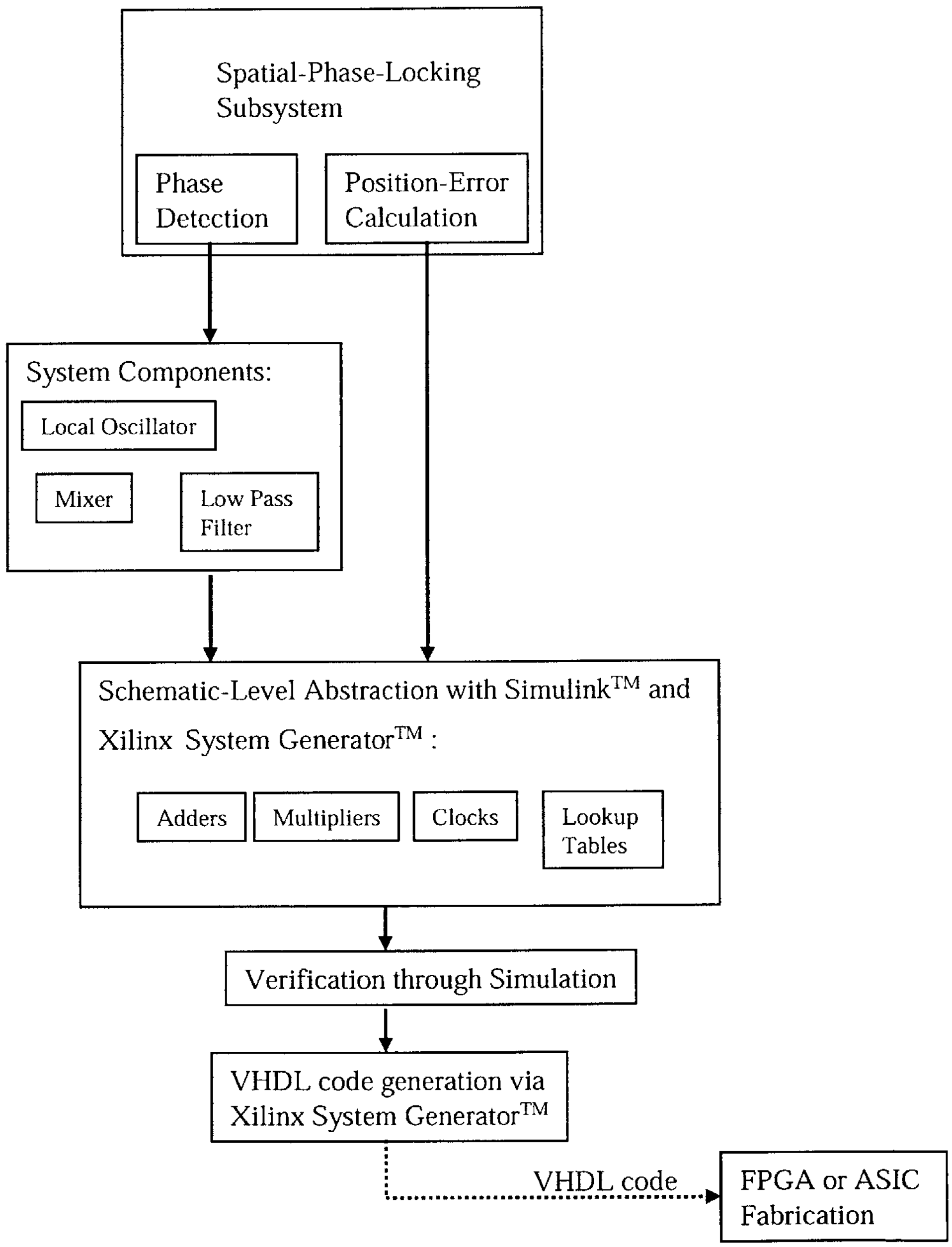


Figure 4-1: Hardware design flow.

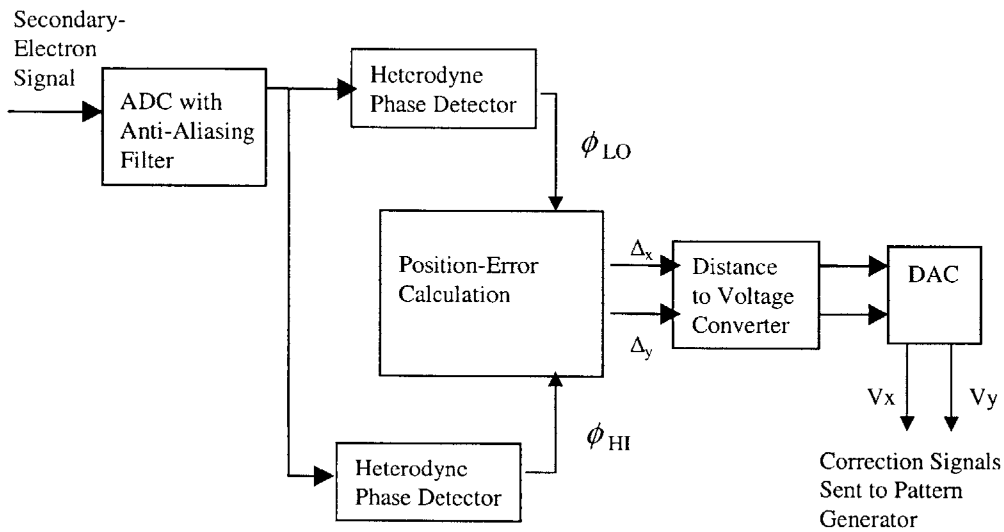


Figure 4-2: Functional block diagram of spatial-phase-locking subsystem.

are organized into libraries according to the functions they provide. Some blocks represent primitive hardware, such as registers and simple logic, while others implement higher level functions, such as DSP algorithms. Each block can be configured to optimize speed or silicon area by specifying the degree to which the architecture is pipelined. Each functional block of the hardware design is modeled with Simulink^{TM} and System Generator^{TM} blocksets, simulated individually, and then combined to form a complete design. The final design is verified through simulation and translated into a hardware description language (HDL) using the System Generator HDL code generation block. This HDL representation could later be used to fabricate an application-specific integrated circuit (ASIC) or execute a field-programmable gate array (FPGA).

4.1 Phase Detector

Phase detection is the critical function of the error detection algorithm. In the previous two chapters a heterodyne phase detection scheme for SPLEBL was described and characterized. Recall that a heterodyne phase detector is comprised by a local oscillator, a mixer, and a low-pass filter as shown in Figure 4-3. Each of these components will be designed with basic hardware elements stored in the Simulink^{TM} and System Generator^{TM} libraries.

4.1.1 Local Oscillator Module

The local oscillator generates digital sine and cosine outputs to modulate the f_o-component of the input signal to baseband. The frequency, f_o, in the case of SPLEBL will be either the low or the high fundamental frequencies, f_{LO} or f_{HI}, of the secondary-electron signal input. The sinusoidal waveforms generated by the local oscillator will have a frequency of either f_{LO} or f_{HI}, depending on which of the fundamental-frequency components is processed. For simplicity, let f_o designate one of these.

The local oscillator is implemented in hardware by a block of memory and an accumulator. A configurable local oscillator module, available as a Xilinx System Generator^{TM} blockset, can be directly integrated into the design so there is no need to construct it from scratch. In the circuit schematic, the local oscillator is represented by the symbol in Figure 4-4(a) and comprised internally with basic hardware building blocks that are shown in Figure 4-4(b) [19].

The frequency of the local oscillator's sinusoidal outputs can be adjusted at any time through a programmable interface. Uniformly spaced samples of one waveform cycle are stored in read-only memory (ROM), also called a "lookup table". When an address value is input to the lookup table, the value stored at that address is expressed. By sampling the table repeatedly at uniform address increments, the local oscillator generates a sinusoidal waveform. The samples in the lookup table represent a single cycle of length N = 2^B, where B is the number of bits per sample. As illustrated in Figure 4-5, each of the N memory addresses (A_0, A_1, ... A_k,... A_N) holds a sample value equal to the sine or the cosine of an

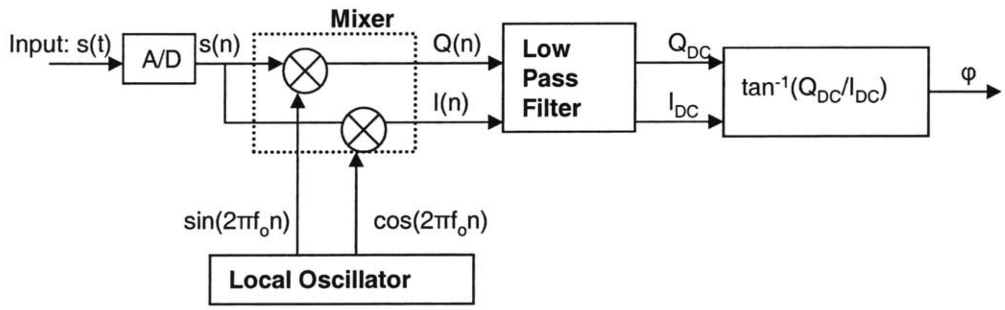
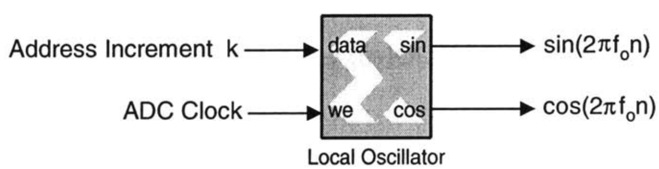
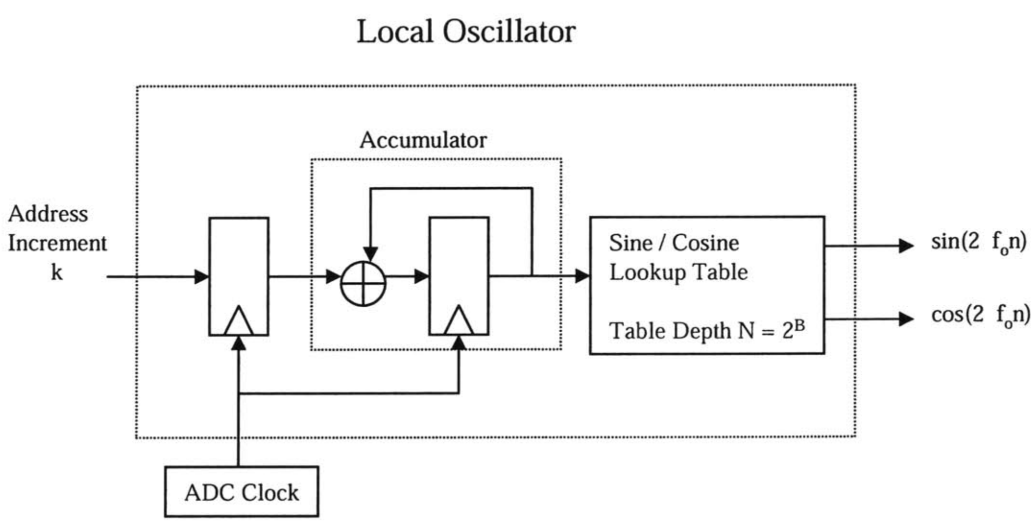


Figure 4-3: Functional block diagram of the heterodyne phase-detection algorithm.



(a)



(b)

Figure 4-4: The local oscillator is shown in (a) along with its internal hardware in (b).

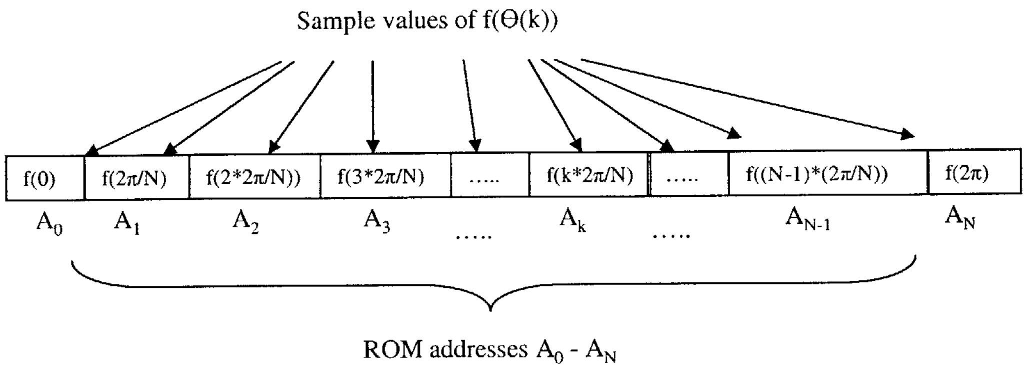


Figure 4-5: Sample values of a sinusoidal function, f($\Theta(k)$), where $\Theta(k)$ = k$\frac{2\pi}{N}$ and k = 0,1,..N are stored in memory addresses A_0 through A_N.

integral multiple of the argument, $\Theta(k)$, given by

$$\Theta(k) = k\frac{2\pi}{N}. \tag{4.1}$$

After every clock cycle, the ROM receives the address of a stored sample. The ROM then outputs the sample value held at that address of the lookup table. For a given clock frequency, f_{clk}, and cycle length, N, the output frequency, f_{out}, of the generated waveform is determined by the address increment, k, which corresponds to acccssing a sample at every kth address, i.e (A_0, A_k, A_{2k},...etc.), where k is an integer between 1 and N/2. The effect that the choice of k has on f_{out} is illustrated in Figure 4-6. Suppose the lookup table consists of $2^5 = 32$ (5-bit) samples. If the address increment k is set to one, the synthesizer calculates the sine function for the phase angles: 0, $2\pi/32$, $2(2\pi/32)$, $3(2\pi/32)$,..., N$(2\pi/32)$, creating a waveform that contains all the samples stored in the lookup table. If the output samples are generated each clock period, T_{clk}, the period of the output waveform is the number of samples taken per cycle multiplied by the clock period (T_{out} = (N/k)$\cdot T_{clk}$). The frequency of the output waveform is then found to be

$$f_{out} = \frac{f_{clk} \cdot k}{N}. \tag{4.2}$$

For example, if all of the samples are accessed the original waveform is created. However, if

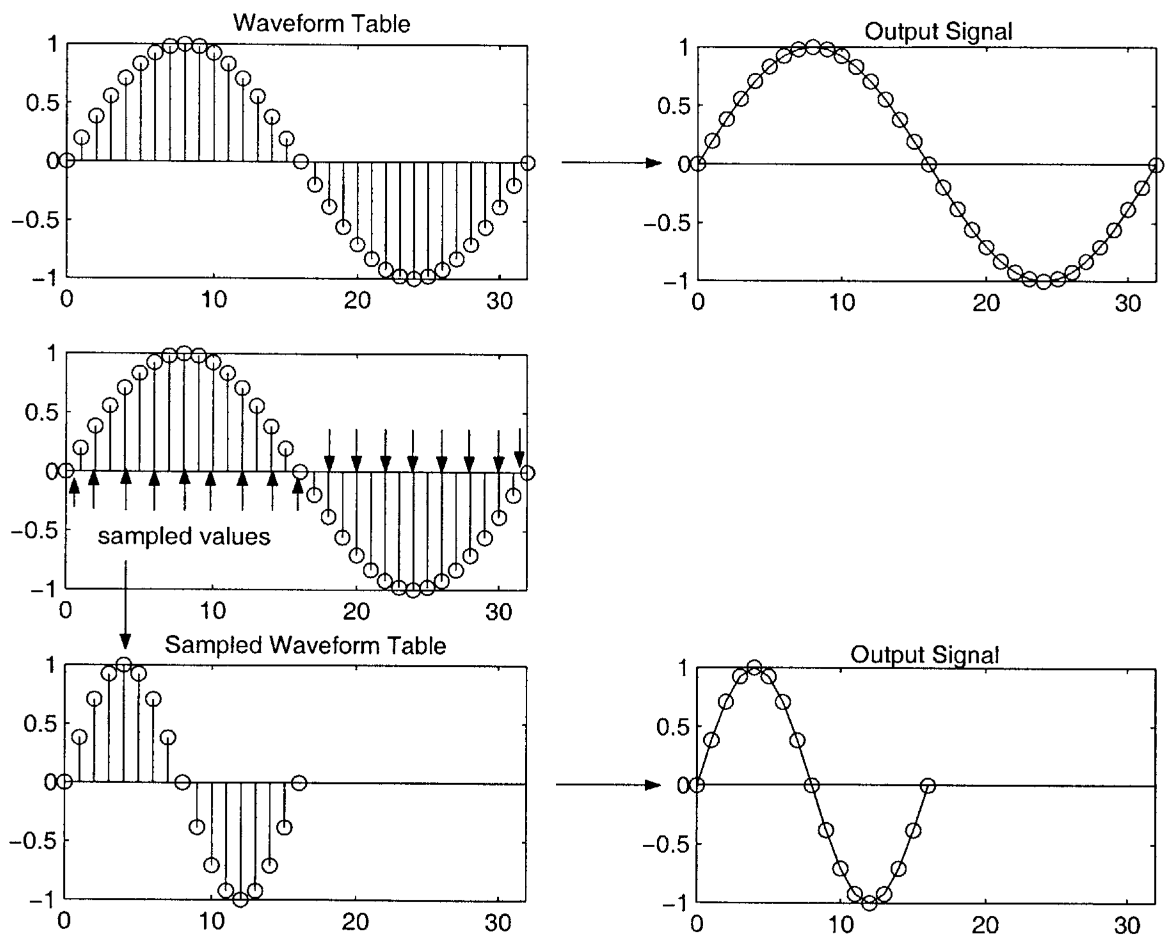


Figure 4-6: As the sampling increment k is increased, the frequency of the output wave increases. When k is doubled for instance, the output frequency is doubled.

only every other sample is accessed, then a sinusoid with twice the frequency of the original waveform is produced.

Given N and f_{clk}, the address increment, k, is chosen according to (4.2) to produce a waveform with a desired frequency, f_{out}. After every clock cycle, the accumulator block of the DCO adds this value of k to previously summed k values and outputs the sum to the lookup table. The accumulator counts from 0 to N by k steps, repeating the count from zero when the sum reaches or exceeds N. The accumulator output serves as the address input into the ROM, instructing it which values to select and express. Each address represents the 0 to 2π index. Skipping addresses in the sequence is equivalent to counting bigger increments of k from zero up to 2π, which results in a higher frequency waveform. The frequency of the waveform can be adjusted in step sizes of Δf, where

$$\Delta f = \frac{f_{clk}}{N}. \tag{4.3}$$

The frequency of the generated waveform is simply a multiple of Δf. The local oscillator generates the synthesized waveform samples at a fixed clock rate, irrespective of the desired waveform frequency. Using the maximum table depth of N $= 2^{16}$ provided by Xilinx System Generator^{TM} local oscillator module and setting f_{clk} to the A/D converter clock rate of 10 MHz, the smallest frequency step is 153 Hz. Recall that the temporal fundamental frequencies of the secondary electron signal are given by

$$f_{LO} = \frac{v_b}{\Lambda_G} \sin\theta, \tag{4.4}$$

$$f_{HI} = \frac{v_b}{\Lambda_G} \cos\theta, \tag{4.5}$$

where v_b is the beam velocity, Λ_G is the spatial grid period, and θ is the rotation angle of the grid. Λ_G and θ will remain fixed; however, v_b could vary depending on the writing requirements of the pattern. The frequencies of the generated sine and cosine waveforms must match either f_{LO} or f_{HI} as closely as possible.

4.1.2 Mixer

The mixer in Figure 4-7, composed of two Xilinx System Generator^{TM} multiplier modules, computes the product of the data on its two input ports, producing the result on its output port. The block supports a size-performance tradeoff in its implementation [20]. It can be designed either as a parallel multiplier that operates on the full width data (faster and larger), or as a sequential multiplier that computes the result from smaller partial products (slower and smaller). Note that this choice affects the hardware implementation only. The simulation behavior of the block is not affected. For the SPLEBL application the physical area of the multiplier on the chip is of no importance; therefore, the parallel multiplier option is chosen to maximize computational speed.

The process of binary multiplication is illustrated with an elementary example in Figure 4-7. In this case the multiplicand is 1010 in binary, which corresponds to the decimal number 10. Likewise, the multiplier, 1110 in binary, is equivalent to decimal number 14. Each bit of the multiplier is multiplied with the bits of the multiplicand. The resulting partial products are aligned according to the position of the bit within the multiplier and summed together to yield final product. If the multiplier bit is a 1, the partial product is simply a shifted copy of the multiplicand, and if the multiplier bit is 0, the partial product is 0. Note that the number of digits in the product is considerably larger than the number in either the multiplicand or the multiplier. Ignoring any sign bits, the length of multiplication of an n-bit multiplicand and an m-bit multiplier results in an (n + m)-bit product. That is, n + m bits are required to represent all possible products.

The Xilinx System Generator^{TM} generates a combinational-logic based multiplier with a pipelined architecture to optimize speed by executing multiple functions simultaneously. Depending on the input bit width and arithmetic type (signed 2's complement or unsigned) the software will automatically choose an appropriate algorithm to maximize the multiplier's performance [20]. In the case of SPLEBL, the inputs are 16-bit signed twos complement numbers. Therefore, the multipliers will generate 32-bit signed twos complement products.

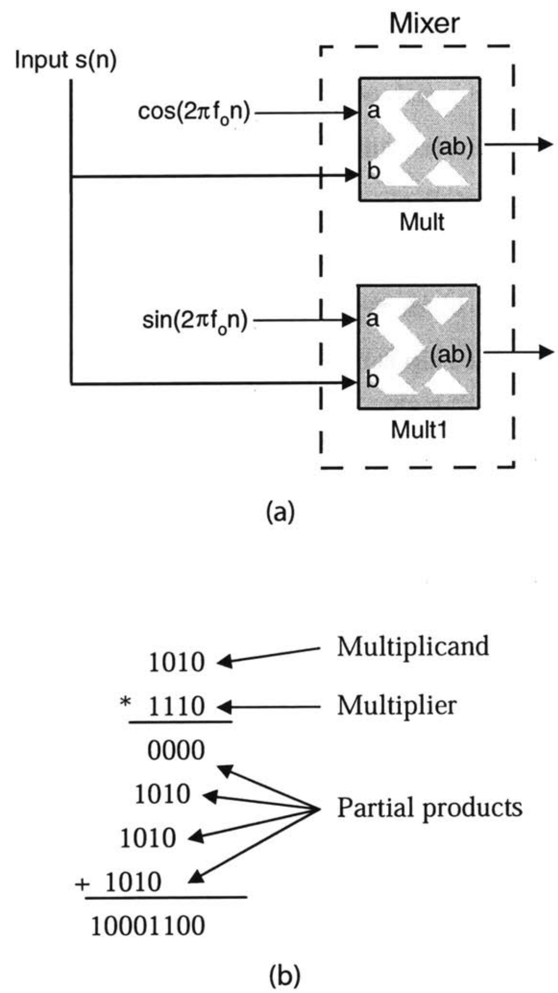


Figure 4-7: A mixer implemented (a) with Xilinx System Generator^{TM} blocks that performs (b) binary multiplication.

4.1.3 Low-Pass Filter

The DC terms of the in-phase and quadrature mixer outputs are extracted with a low-pass filter that averages the samples over an integral number of periods. Recall that the in-phase and quadrature DC components are respectively proportional to the cosine and sine of the input phase. The hardware implementation of this filter is illustrated in Figure 4-8. The first stage of the low-pass filter is an accumulator, composed of an adder and a D flip-flop register, followed by a multiplier that scales the total by the number of summed points. The output must have the same width, arithmetic type and binary point position as the input. On each clock cycle, the input is added to a sum of previous inputs and stored in the register. The register then outputs the running sum; however, the signal is down-sampled such that only the sample corresponding to the average of an exact integral period number of points is kept, and the other output samples are discarded. The register is then reset to zero and the average over the next block of points is taken. After the first down-sampling procedure the data rate of the samples is reduced by a factor equal to the number of averaged samples.

The low-pass filter design is tested by passing a zero-phase test signal, $\cos(\frac{2\pi}{64}\mathrm{n})$, through the phase detector circuit. The test setup is shown in Figure 4-9. The output of the two low-pass filters is monitored in Figure 4-10 along with the outputs of the local oscillator and the mixer. The first two plots show the outputs of the local oscillator, a sine and a cosine, with the same period as the test signal. The first output of the mixer, also referred to as the in-phase component I(n), is shown in the third plot. Because the input is purely sinusoidal, I(n) is equal to $\frac{1}{2}\left[1+\cos(\frac{2\pi}{32}n)\right]$. Likewise, the second output of the mixer, Q(n), shown in the fifth, is $\frac{1}{2}\left[\sin(\frac{2\pi}{32}n)\right]$. The fourth plot shows the output of the low-pass filter that extracts the DC component, I_{DC}, from I(n) by averaging the signal over an integral-period number of samples. In this case, the number of summed samples is 128, or twice the period of the test signal, though any integer multiple could have been chosen. The output I_{DC} is 0.5, as expected for the test signal. The sixth plot depicting Q_{DC}, the result of passing Q(n) through the other low-pass filter, is zero as expected.

The in-phase and quadrature DC components are divided using the CORDIC (Coordinate

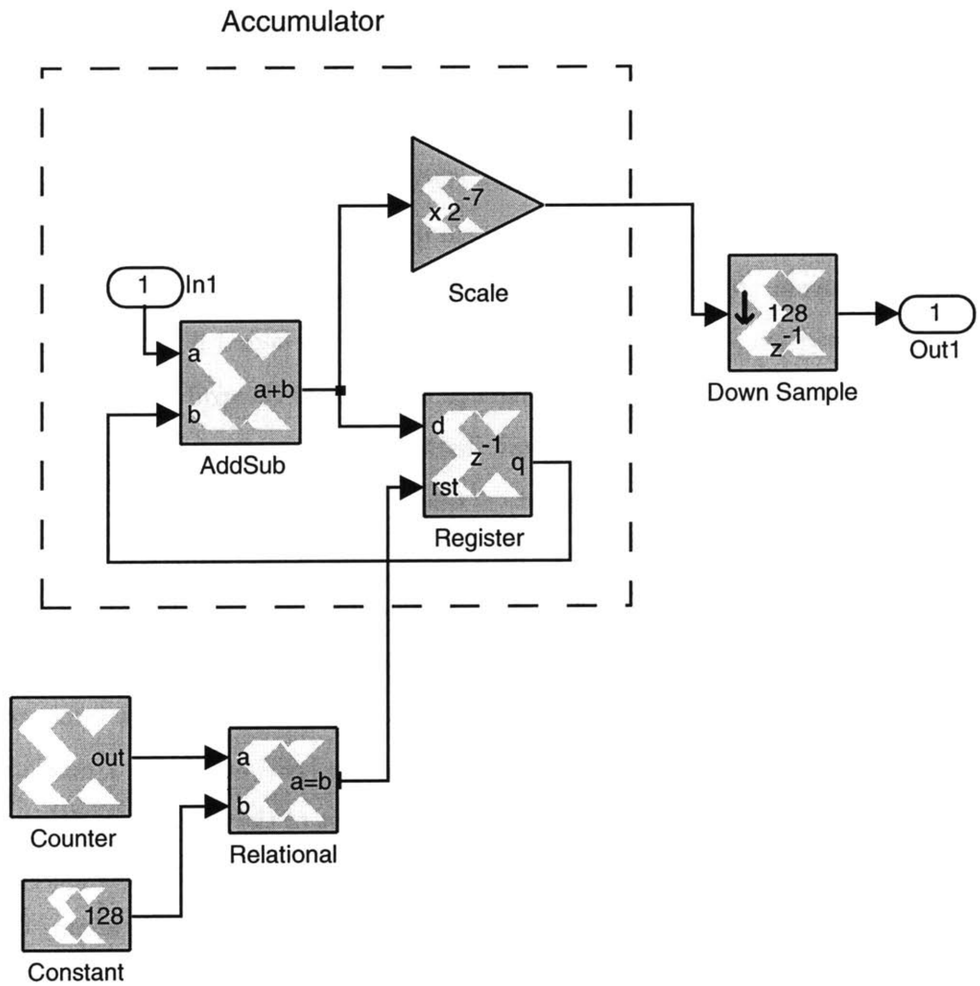


Figure 4-8: Low-pass filter hardware implementation.

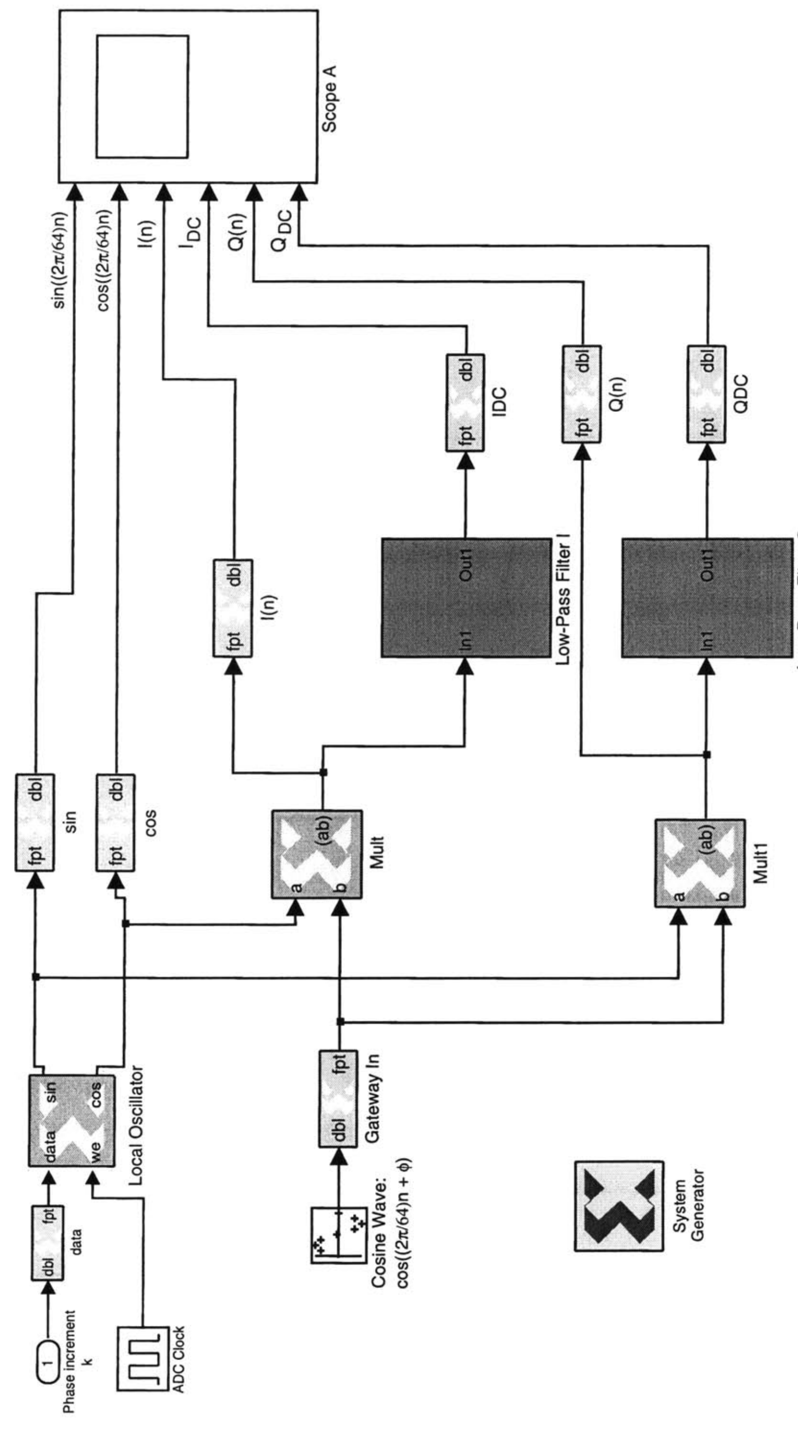


Figure 4-9: Schematic and test setup of the local oscillator, mixer, and low-pass filters.

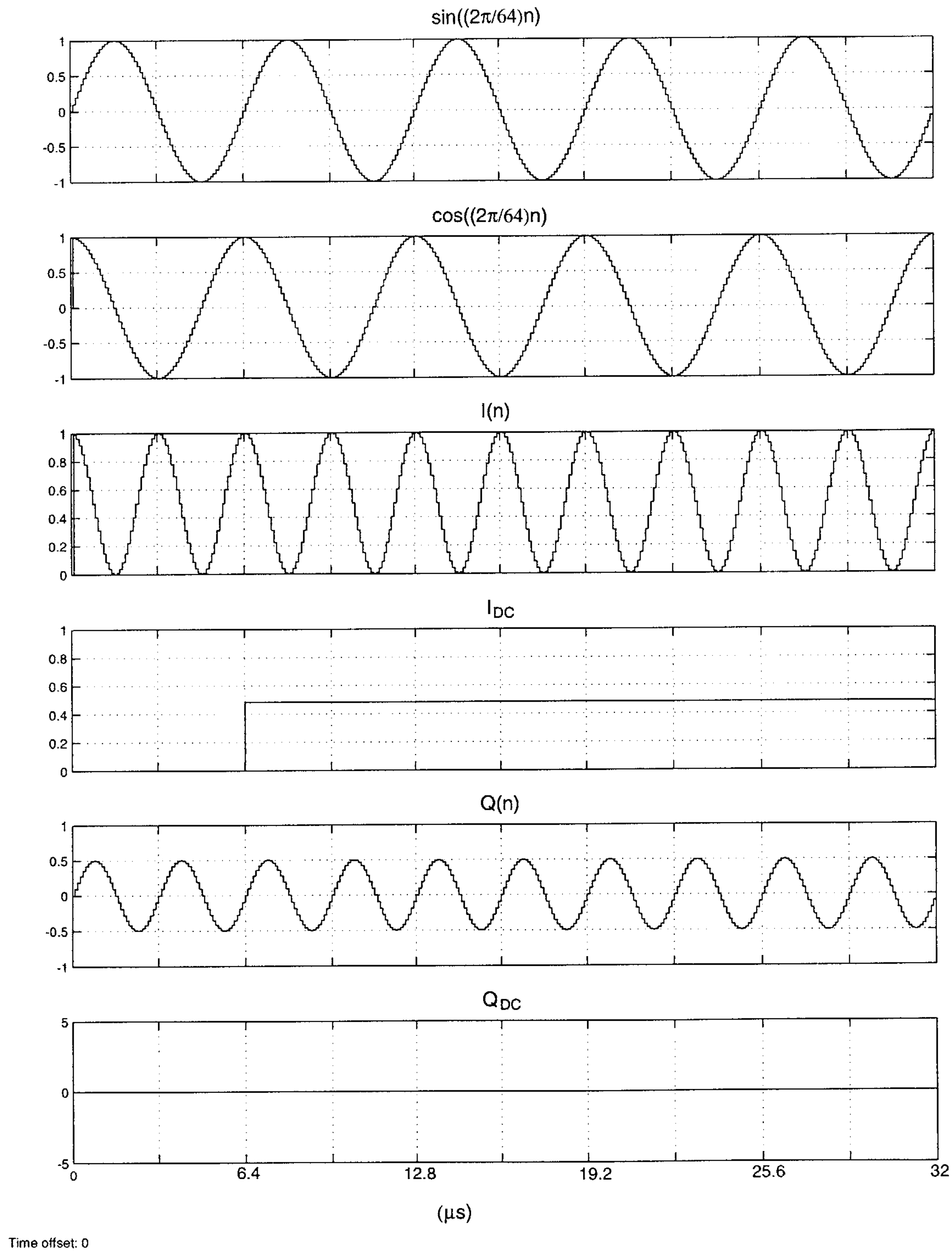


Figure 4-10: Simulation of the local oscillator, mixer, and low-pass filters.

Rotation Digital Computer) algorithm for binary division [21]. The CORDIC algorithm is a hardware-efficient, iterative process that performs a division to any desired precision using only addition and shift operations. The Xilinx System Generator^{TM} library contains a CORDIC divider module that can be configured to maximize performance and precision. Two architectural configurations are available for the CORDIC core. The first is a fully parallel configuration that optimizes throughput at the expense of silicon area. The second is a serial implementation which is slower but saves silicon area. For SPLEBL, silicon area is not a great concern; however, computational efficiency is important. Consequently, a parallel hardware configuration is chosen for the CORDIC divider. A parallel CORDIC core with an N-bit output width produces a new output after every N clock cycles. For SPLEBL N = 32; therefore, 32 clock periods will pass between the time the inputs are sampled and the time that the corresponding samples appear on the output.

4.1.4 Arctangent function

Recall that the phase of the input is computed by taking the arctangent of the quadrature DC-component divided by the in-phase DC-component.

$$\phi = \arctan\left(Q_{DC}/I_{DC}\right).$$

To speed up the computation process, a lookup table is used to find the arctangent of a value instead of calculating the arctangent values directly. A lookup table retrieves the arctangent value from a memory address instead of calculating it using a mathematical formula, such as a Taylor series. The advantage of this method is a significant gain in speed because fewer clock cycles are required to retrieve a value from memory as opposed to calculating it directly.

The arctangent function is implemented with a lookup table containing a finite number of values of arctan(x), called breakpoints. The size of the table in Figure 4-11 is determined by the number of breakpoints and the number of bits used to represent the breakpoint val-

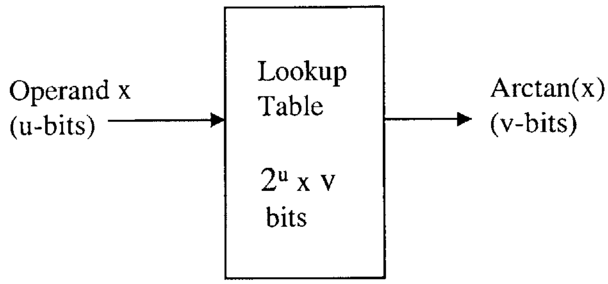


Figure 4-11: A lookup table with dimensions 2^u by v.

ues. If u is the bit-width of operand, x, 2^u breakpoints would be needed to correspond to all possible operand values, resulting in a table size of 2^u by v bits. Increasing the number of breakpoints causes the table size to grow considerably. A large table requires a sizeable amount of memory, which is often impractical and expensive.

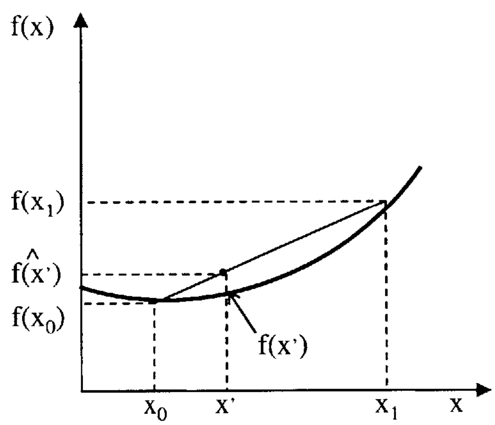


Figure 4-12: Linear Interpolation to approximate f(x').

Instead of directly mapping the operand, x, of a function f(x) to the values stored in the table, a more practical method is to approximate f(x) at a given value of x by linearly interpolating between two adjacent breakpoints closest to that value. To find f(x = x') in

Figure 4-12, for example, the table is read at the breakpoints (x_0 and x_1) that enclose the interval containing x' such that $x_0 < x' < x_1$. The value, f(x'), is approximated as $\widehat{f(x')}$ by finding the line that connects the endpoints, (x_0, $f(x_0)$) and (x_1, $f(x_1)$). Substituting the slope m of the line into its equation, $f(x')$ - $f(x_0) = m(x' - x_0)$, yields the approximation

$$\widehat{f(x')} = f(x_0) + \left(\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right)(x' - x_0), \tag{4.6}$$

also known as Newton's Method.

As shown in Figure 4-13, a lookup table from the Simulink^{TM} blockset, using linear interpolation, applies an arctangent function to the output of the CORDIC divider to estimate the phase of the input sinusoidal signal. The arctangent lookup table holds 256 values, equally spaced between $-\pi/2$ and $\pi/2$ and having a bit-width of 16. Therefore the table requires a ROM of at least 4096 bits. The arctangent function is tested by adding a known phase to the input signal and comparing that phase to the estimated phase that the arctangent function outputs to the display. The result of the simulation is shown in Figure 4-14. The difference between the estimated phase, $\hat{\phi}$, and the known phase, ϕ, is plotted as a function of the known phase. Excluding the phase values of $\pm\pi/2$, the absolute estimated phase error ranges between zero and 0.0235 radians. When ϕ approaches $\pm\pi/2$ the error increases dramatically such that the phase-subsystem can no longer accurately detect the phase of the input signal. The output of the Simulink^{TM} arctangent lookup table module is limited to values between $-\pi/2$ and $\pi/2$ [22].

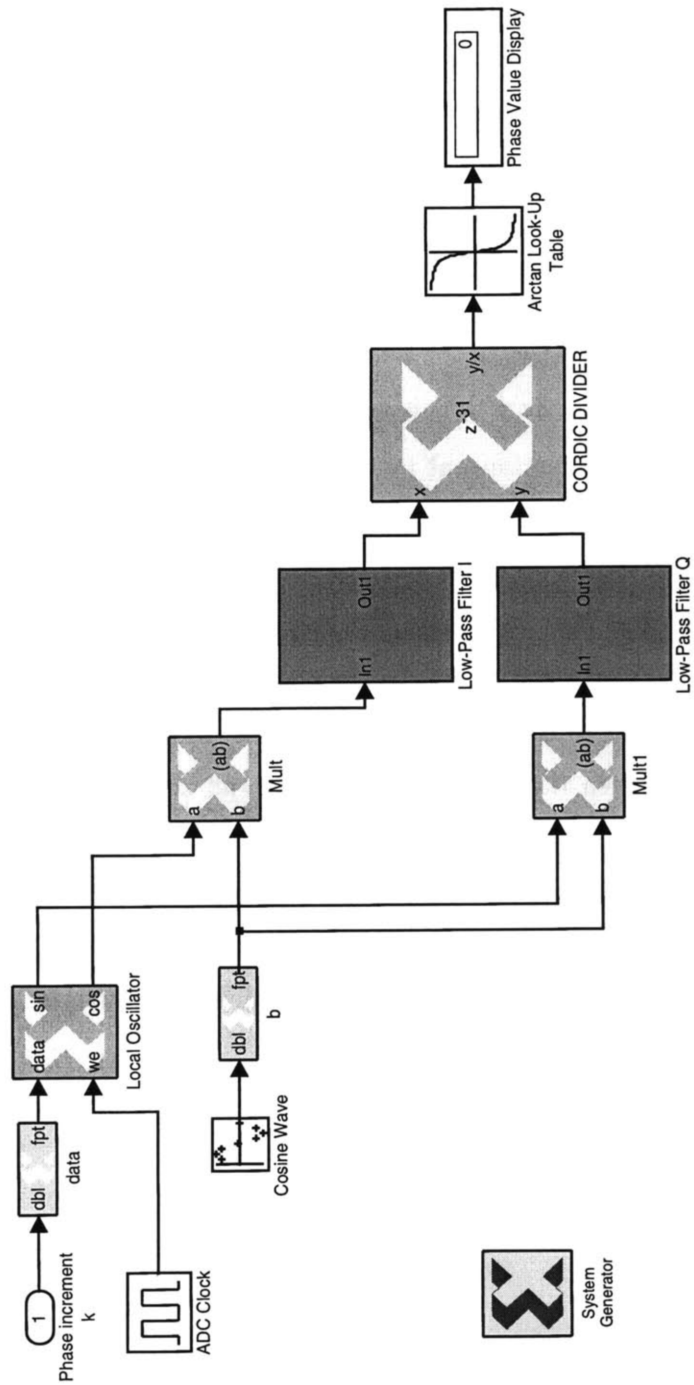


Figure 4-13: Test setup of the phase-detection subsystem.

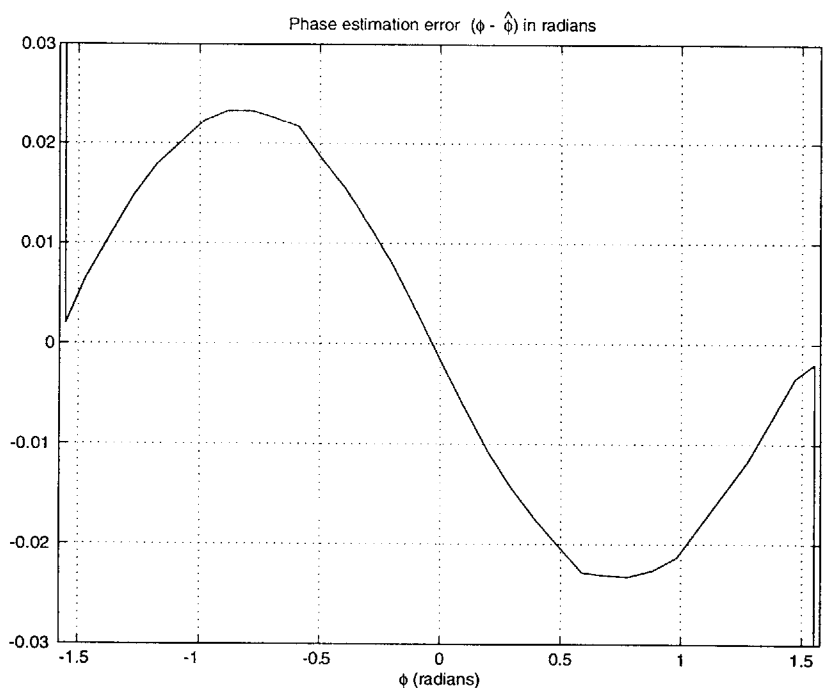


Figure 4-14: Simulation results of the phase-detection subsystem depicts the phase-estimation error as a function of phase.

4.2 Position-Error Calculation

The position-error of the beam is computed from the phase estimates, ϕ_{LO} and ϕ_{HI}, computed by the phase detector. Recall that in Chapter 2 the relation between the x- and y-position errors and the phase estimates was determined to be

$$\Delta x = \frac{k_{LO}\phi_{LO} + k_{HI}\phi_{HI}}{k_{LO}^2 + k_{HI}^2},$$
$$\Delta y = \frac{k_{LO}\phi_{HI} - k_{HI}\phi_{LO} - y_p(k_{LO}^2 + k_{HI}^2)}{k_{LO}^2 + k_{HI}^2}.$$

Figure 4-15 illustrates how these equations are described with simple hardware elements: adders, subtractors, multipliers and dividers. The multiply and divide operations are performed with the same Xilinx System Generator^{TM} multiplier and CORDIC divider blocksets used for the phase detector. The inputs to the position-error detector are the phases (ϕ_{LO} and ϕ_{HI}) received from the phase detector and the known values: $\sin\theta$, $\cos\theta$, k_o, and y_p, where y_p is the distance stepped in the y-direction after a scan is completed in the x-direction. To smoothly integrate the position-error calculation subsystem with the rest of the design, its contents are encapsulted into a "mask", shown in Figure 4-16 as a generic block, in which only the inputs and outputs can be seen. Figure 4-17 depicts the test setup for the subsystem. The grid rotation angle, θ, is set to 20 degrees, the spatial frequency of the grid, k_o, is fixed to $2\pi/250$nm, and y_p is set to 5 nm. The output position errors in the x- and y-directions (Δx and Δy) are monitored with a scope while the phases ϕ_{LO} and ϕ_{HI} are ramped between $-\pi/2$ to $\pi/2$ as expected. The phase inputs are varied monotonically from $-\pi/2$ to $\pi/2$, and the resulting output x- and y-position errors are compared to the theoretical values that result from applying the phase input into the equations for Δx and Δy. The results of the simulation are shown in Figure 4-18. The error between the theoretical and simulated position-error values are greatest when both ϕ_{LO} and ϕ_{HI} are approaching $-\pi/2$ and $\pi/2$. In between this range however, the difference between the theoretical and simulated values is on the order of 10^{-14} meters, though an order of 10^{-10} meters would be adequate.

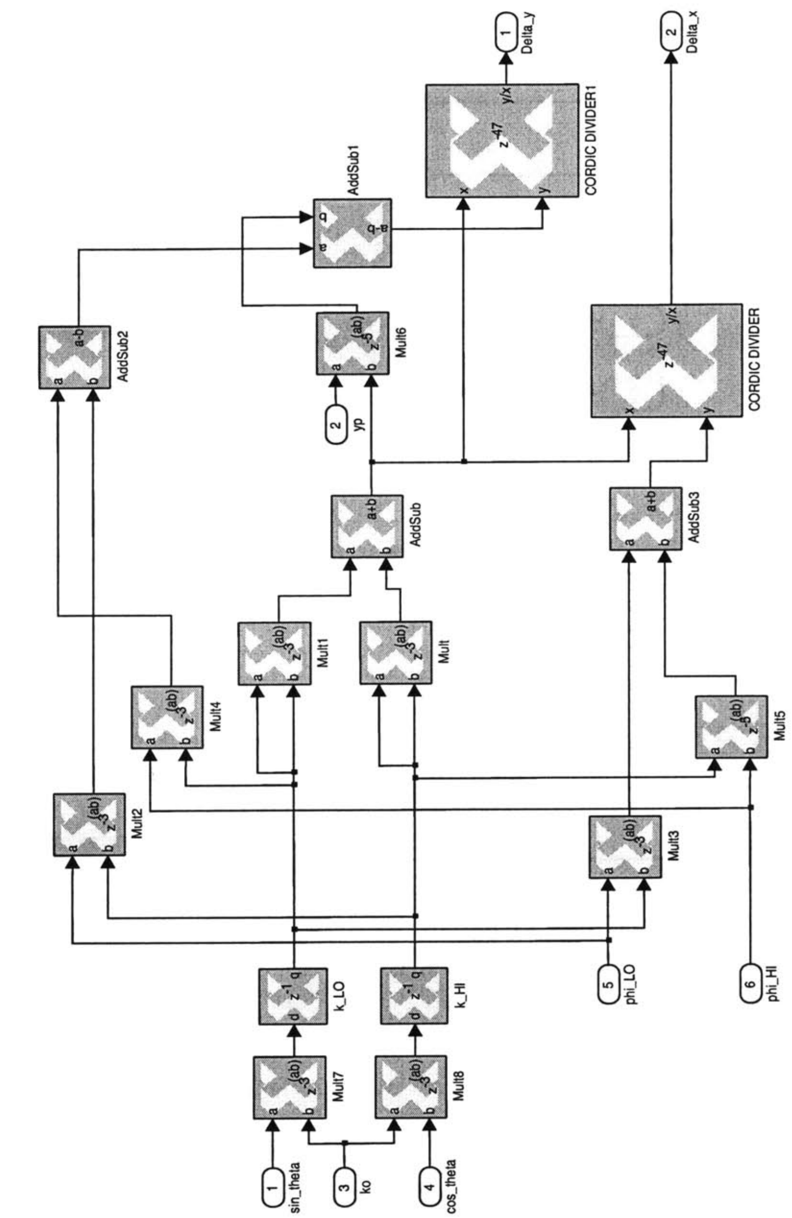


Figure 4-15: Position-error detection circuit schematic.

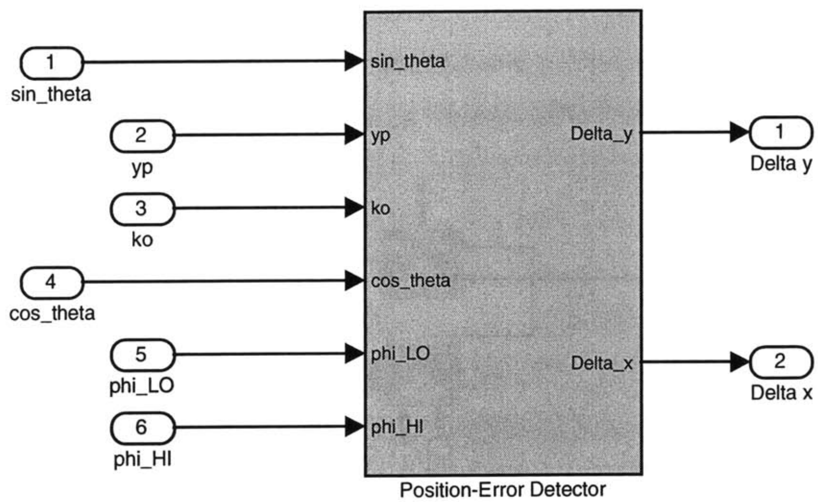


Figure 4-16: Position-error detection subystem mask.

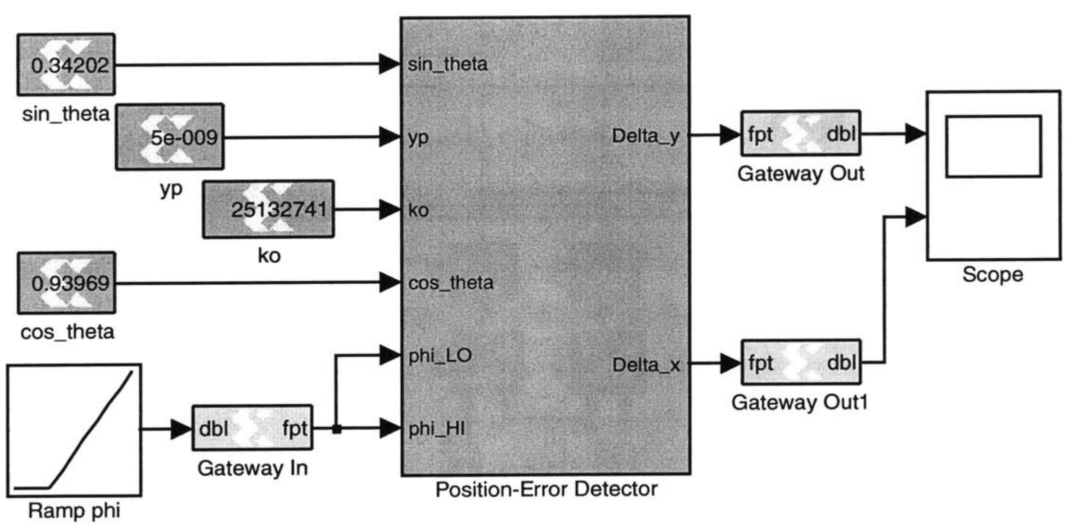


Figure 4-17: Test setup for the position-error calculation subsystem.

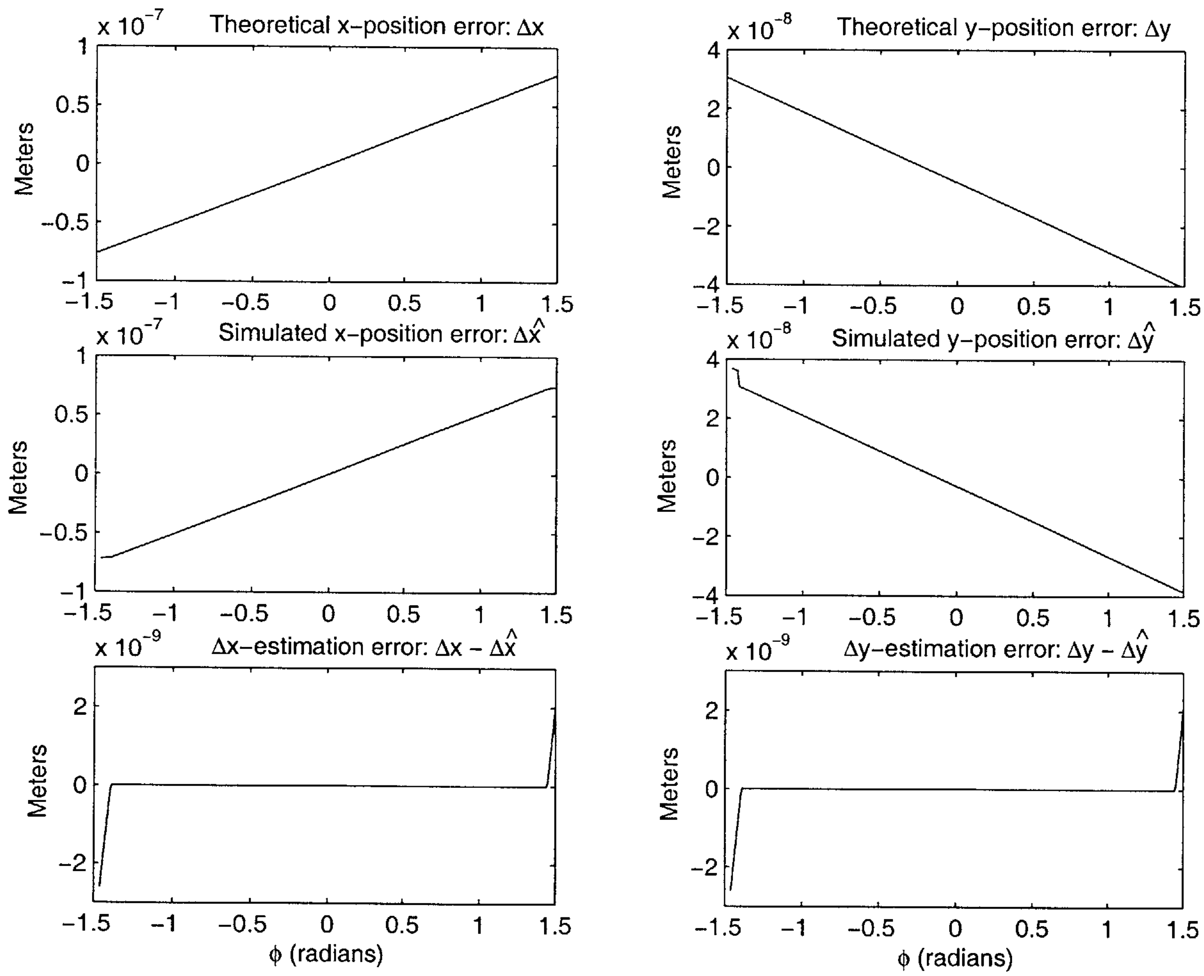


Figure 4-18: Simulation results of the position-error detector. The top two plots show the theoretical position-error values in the x- and y-directions. The middle plots display the position-error values obtained from the circuit simulation, and the bottom plots depict the absolute difference between the theorectical and simulated values.

Input	Description	Value
k_o	Grid Spatial Frequency	$2\pi \cdot 8$ rad/μm
θ	Grid Rotation Angle	20 degrees
f_{clk}	ADC clock frequency	10 MHz
f_{LO}	Low fundamental frequency	34.2 kHz
f_{HI}	High fundamental frequency	93.9 kHz
y_p	Distance stepped in y-direction	5 nm
v_b	Beam velocity	25.0 mm/s
k_{inc} (LO)	Address increment (LO)	224
k_{inc} (HI)	Address increment (HI)	614

Table 4.1: Input values used for testing the spatial-phase-locking subsystem.

4.2.1 Overall System Verification

The phase detection and position-error calculation subsystems are combined together to form the spatial phase-locking (SPL) subsystem that is simulated with and without noise. Figure 4-19 depicts the schematic of the overall SPL subsystem and its test setup. The input test signal, representing the secondary-electron signal, is loaded into the phase detector subsystems from a Matlab file named input.mat. The input signal is a sum of two sinusoids of frequencies, f_{LO} and f_{HI}. Recall that these frequencies are calculated from input parameters, k_o and θ as

$$f_{LO} = 2\pi k_o v_b \cdot \sin\theta, \tag{4.7}$$

$$f_{HI} = 2\pi k_o v_b \cdot \cos\theta. \tag{4.8}$$

The values of the input parameters and frequencies are given in Table 4.1. A known phase ϕ is added to each of the sinusoidal components. For simplicity, an equal phase is added to each component such that $\phi_{LO} = \phi_{HI} = \phi$. The address increment input (k_{inc}) to the phase detector is set so that the frequency of the local oscillator outputs match f_{LO} or f_{HI}. Recall that the address increment is given as

$$k_{inc} = \frac{f_{out} \cdot N}{f_{clk}}, \tag{4.9}$$

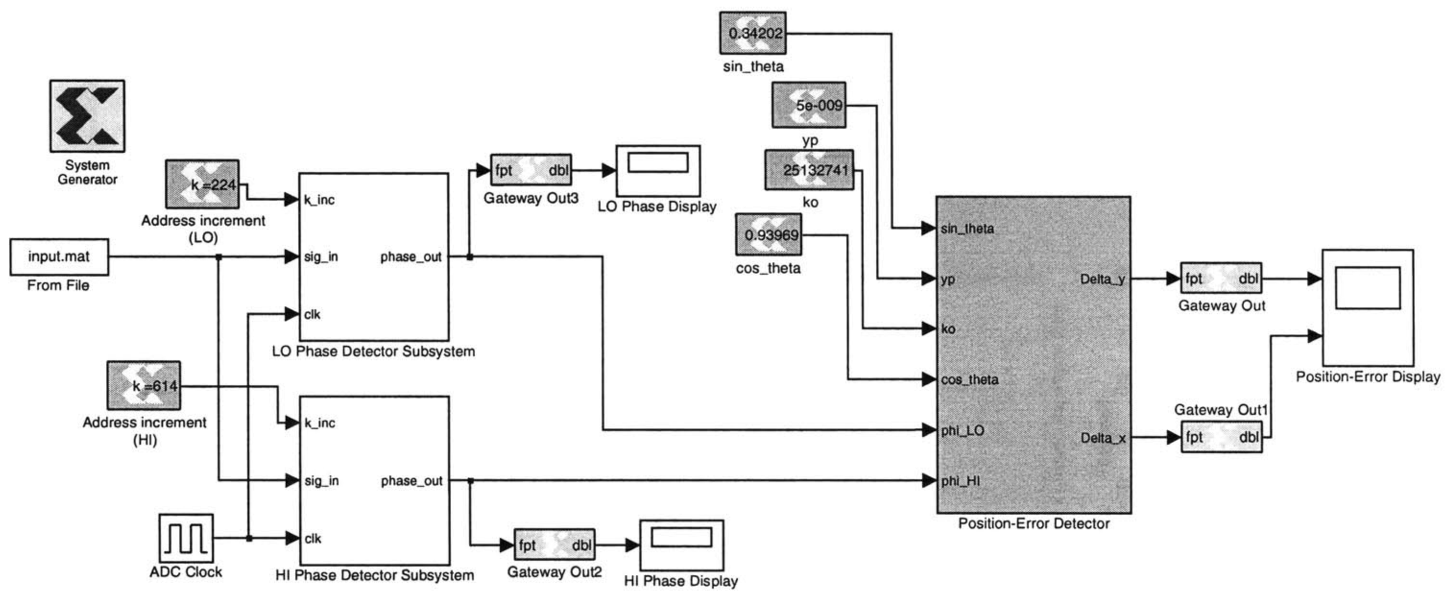


Figure 4-19: Schematic of the overall spatial-phase locking subsystem.

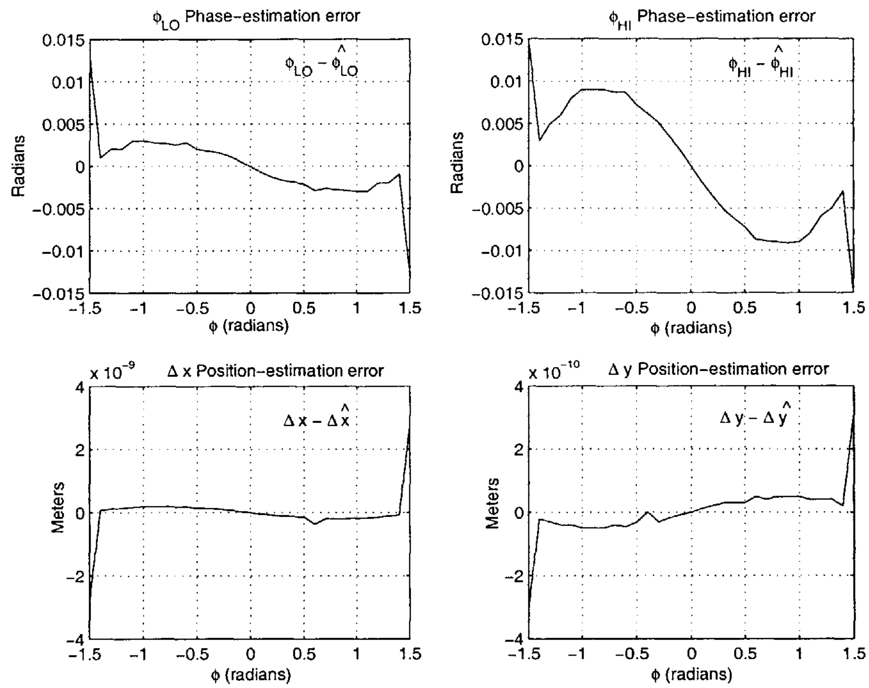


Figure 4-20: Verification of the spatial-phase locking subsystem through simulation.

where f_{out} is either f_{LO} or f_{HI}, f_{clk} is the ADC clock frequency, and N is the depth of the waveform table. The maximum table depth that Xilinx supports, N = 2^{16}, is used in the design.

The functionality of the circuit without noise is verified by comparing the output x- and y-position error estimates ($\widehat{\Delta x}$ and $\widehat{\Delta y}$) with the expected x- and y-position error values (Δx and Δy) for a given phase ϕ. The results of the spatial-phase-locking subsystem simulation without the presence of noise are shown in Figure 4-20. The top two subplots show the difference between the actual phase (either ϕ_{LO} or ϕ_{HI}) and the estimated value ($\hat{\phi}_{LO}$ or $\hat{\phi}_{HI}$) produced by the circuit. The bottom plots show the difference between the expected position-error values (Δx and Δy) and the position-error values ($\widehat{\Delta x}$ and $\widehat{\Delta y}$) obtained through simulation. The phase estimation and position-error estimation errors are greatest when ϕ is equal to $\pm\ \pi/4$. Table 4.2 shows the estimation errors at these two phase values. The values reflect inherent error of the overall circuit, having a maximum value of $\pm$ 0.2 nm

Estimated Value	Estimation Error with $\phi = \pi/4$	Estimation Error with ϕ = -$\pi/4$
ϕ_{LO}	0.0027 rad	-0.0028 rad
ϕ_{HI}	0.0090 rad	-0.0090 rad
Δx	-0.19 nm	0.37 nm
Δy	-0.05 nm	0.05 nm

Table 4.2: Results of the SPL subsystem simulation.

in the x-direction and ± 0.05 nm in the y-direction.

The performance of the circuit with noise is measured by varying the signal-to-noise ratio (SNR) of the input signal and measuring the output x- and y-position error estimates that result when the phase of the input is fixed at $\pi/4$. The standard deviation of the position-error estimates are plotted as a function of SNR in Figure 4-21 along with their corresponding Cramer-Rao bounds. Due to the run-time restrictions of the student version of the software, the minimum bandwidth for which the SPL subsystem can be set to operate at is 16.7 kHz. At this bandwidth, a SNR ratio of 11 dB or greater is required for the standard deviation of both x- and y-position-error estimates to be less than one nanometer. The standard deviations approach their Cramer-Rao bounds indicating that the position-errors are estimated efficiently. The full version of the Xilinx System Generator^{TM} software (which is quite expensive) is needed to run simulations at lower bandwidths. However, at lower bandwidths the circuit would behave similarly if not better because accuracy increases with lower bandwidth.

4.2.2 HDL code generation

The Xilinx System Generator tool easily converts the schematic design into a low-level hardware design language (HDL). The HDL files can then be processed by a series of compilation tools that translate the HDL description into an equivalent low-level, executable program called a "bitstream". A FPGA is programmed (or reprogrammed) by downloading the bitsream into the static on-chip RAM cells. The RAM cells, sprinkled throughout the chip, determine the functionality of the logic blocks and define the connectivity of the signal paths.

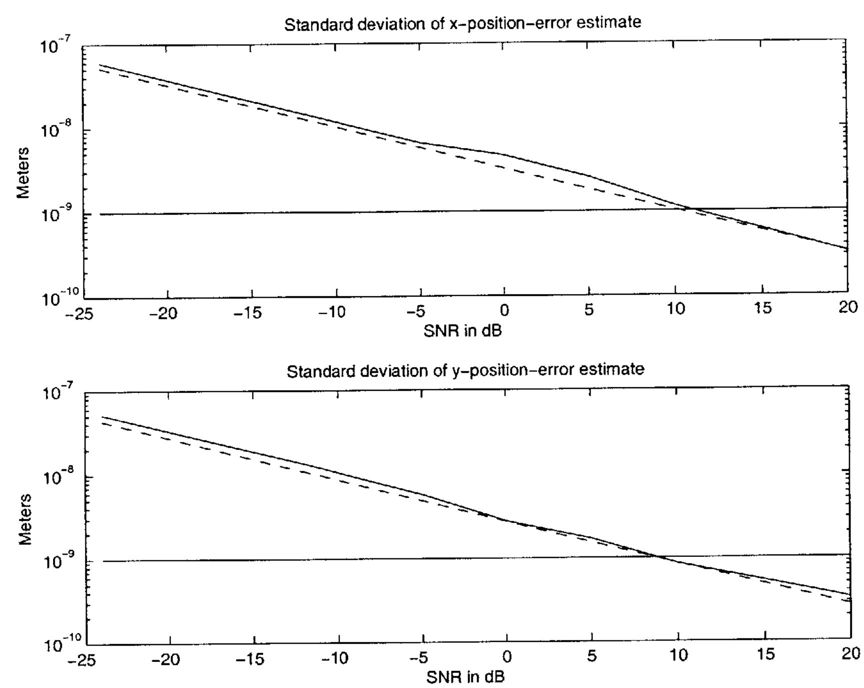


Figure 4-21: Standard deviation of the x- and y-position-error estimates as a function of SNR at a bandwidth of 16.7 kHz. The dashed lines are the correspoinding Cramer-Rao bounds.

Chapter 5

Conclusion

The objective of this thesis has been to design a spatial-phase locking algorithm that is suitable for hardware implementation and to describe how that algorithm could be realized as a stand-alone chip, in the form of either a FPGA or an ASIC. A heterodyne phase-detection algorithm was chosen because it requires only basic mathematical operations. Furthermore, the algorithm was shown to perform well in a noisy environment and to maintain nanometer level accuracy at high bandwidths.

The design procedure of the spatial-phase-locking subsystem proceeded in a top-to-bottom fashion, beginning with the algorithm development. A highly accurate, yet computationally efficient phase-detection algorithm was needed, and two possible phase-detection algorithms were examined. The first was a heterodyne-based method, and the second algorithm estimated phase indirectly by monitoring the shift in the peak frequency, resulting from a changing phase. The heterodyne method required that the number of collected samples equal a full integral number of periods, while the indirect method had no such limitation and could accommodate an arbitrary sample number.

The speed and accuracy of the algorithms were characterized by analyzing their performances with noise. The variance of the heterodyne-phase estimator was determined as a function of SNR for various bandwidths. As bandwidth increased, a higher SNR was required for the heterodyne algorithm to maintain a given level of accuracy. From data showing the vibrational disturbances as a function of frequency [17], a bandwidth of at least 2 kHz was deemed necessary to effectively cancel out the strongest vibrations. The algorithm employing indirect phase-estimation was found to be considerably more computationally intensive than the heterodyne method. Indirect phase-estimation requires very high-resolution Fourier transforms to detect the small shift in the peak frequency that results from a changing phase. To provide the necessary resolution, the Fourier transforms would need to be heavily padded with zeros, making them very long and unsuitable for efficient computation.

The algorithm has been modeled as a digital circuit with the Simulink^{TM} and Xilinx System Generator^{TM} blocksets, which are identical representations of their physical hardware counterparts. The design was tested using a simulated secondary-electron signal with a known varying phase and shown to exhibit a position-error estimate with a variance that comes very close to the Cramer-Rao bound, signifying efficient estimation.

The design is a general representation of the error-detection algorithm, and further work is necessary to integrate the subsystem with a specific SEBL tool. Though the basic functionality of the design would remain the same, the hardware, software, and timing details of the interfaces would need to be customized for each SEBL machine. This thesis provides a basic blueprint of the subsystem itself and serves as a tutorial to the SPLEBL spatial-phase locking subsystem to facilitate the integration of SPLEBL into the next generation of electron-beam lithography tools.

The transition of SPLEBL from research into industry must be completed shortly for the semiconductor industry to keep pace with Moore's Law. Resolution and pattern-placement accuracy go hand in hand. As feature size is reduced, pattern-placement accuracy must also be improved to ensure the fidelity of the written design. SPLEBL initiates a new paradigm for electron-beam lithography, and is currently the only method that can achieve nanometer accuracy. SPLEBL also opens a new market for low-cost, high performance EBL tools by eliminating the need for expensive shielding and isolation equipment. Universities, small companies, and other members of the research community who use EBL but cannot afford the multi-million dollar price tag of a modern EBL tool, will benefit from an affordable, SPLEBL-based option. The work presented in this thesis is a single contribution of a greater endeavor to commercialize SPLEBL, and I feel privileged to have been part of this effort and vision.

Appendix A

Matlab Scripts

A.1 Chapter 3 Figures

A.1.1 Figure 3-1: Variance Versus SNR

```
%Variance of phase versus SNR at various data length N

clear
figure;
%close all

N = [16, 128, 2^10];
%Nfft = 2^14;
A = 1;
w = pi/2;
Lambda = 250*10^-9;
conv = Lambda^2/((2*pi)^2);
%%%%%%
wm = pi/1000
wc = pi/2
%%%%%%%
for m = 1:3

    Nfft = N(m);
    n = 1:N(m);
    sm = 100*(pi/5)*cos((wm)*n);
    sig = A*cos(wc*n + sm);
    range = -50:10;
```

```
    %Calculate maximum frequency of orig signal
    Yo = (1/N(m))*fft(sig-mean(sig),Nfft);
    [max_Yo, w_maxo] = max(abs(Yo).^2);
    Tho = angle(Yo(w_maxo));

    iter = 1;
    for raw_snr = range
      %fprintf('For snr %d\n', raw_snr);
      snr = 10^(raw_snr/10);
      count = 1;
      for i = 1:200


        nois_amp = sqrt((A^2)/(2*snr));
        nois = nois_amp * randn(size(sig));
        nsig = sig + nois;

        Y = (1/N(m))*fft(nsig-mean(nsig),Nfft);
        [max_Y, w_max] = max(abs(Y).^2);

        w_hat(count) = (w_max-1)/(Nfft/2);
        A_hat(count) = sqrt((4/N(m))*max_Y)
        Th_hat(count) = angle(Y(w_maxo));

        count = count + 1;
      end
      ww(iter) = w_hat(3);
      tt(iter) = Th_hat(3);
      ttm(iter) = mean(Th_hat);
      w_var(iter) = var(w_hat);
      CR_w(iter) = 12/(snr*N(m)*(N(m)^2-1));
      CR_w(iter) = 0.3/(snr*N(m)*(N(m)^2-1));
      CR_A(iter) = 1/(N(m)*snr);
      CR_th(iter) = 1/(N(m)*snr);

      A_var(iter) = var(A_hat/A);
      Th_var(iter) = var(Th_hat);
      iter = iter + 1;
    end
    semilogy(range,Th_var);hold on;semilogy(range,CR_th,'--');
    xlabel('SNR in dB');
    ylabel('Variance of phase estimate in radians');

end

hold off;
```

A.1.2 Figure 3-2: STD of the Position-Error Versus Bandwidth

```
%STD of the position-error estimate versus BW at various SNR.
clear
figure;

raw_snr = [-24,-12,-2];
R = 10^7;
A = 1;
w = pi/2;
Lambda = 250*10^-9;
conv = Lambda^2/((2*pi)^2);
for m = 1:3
   iter = 1;
   rl = 1;
   rh = 19;
   for range = rl:rh
      N = 2^range
      Nfft = N;
      n = 1:N;
      sig = A*cos((w)*n);
      snr = 10^(raw_snr(m)/10);

      Yo = (1/N)*fft(sig-mean(sig),Nfft);
      [max_Yo, w_maxo] = max(abs(Yo).^2);
      Tho = angle(Yo(w_maxo));
      count = 1;
      for i = 1:500

         nois_amp = sqrt((A^2)/(2*snr));
         nois = nois_amp * randn(size(sig));
         nsig = sig + nois;

      Y = (1/N)*fft(nsig-mean(nsig),Nfft);
         [max_Y, w_max] = max(abs(Y).^2);

         w_hat(count) = (w_max-1)/(Nfft/2);
         A_hat(count) = sqrt((4/N)*max_Y);
         Th_hat(count) = angle(Y(w_maxo));

         count = count + 1;
      end
      ww(iter) = w_hat(3);
      tt(iter) = Th_hat(3);
```

```
        ttm(iter) = mean(Th_hat);
        w_var(iter) = var(w_hat);
        CR_w(iter) = 12/(snr*N*(N^2-1));
        CR_w(iter) = 0.3/(snr*N*(N^2-1));
        CR_A(iter) = 1/(N*snr);
        CR_th(iter) = 1/(N*snr);

        A_var(iter) = var(A_hat/A);
        Th_var(iter) = var(Th_hat);
        iter = iter + 1;
    end
    M = 2.^(rl:rh);
    x = R./(2*M);
    loglog(x,(Th_var*conv).^.5);
    hold on;
    loglog(x,(CR_th*conv).^.5,'--');
    hold on;
    loglog(x,(10^-9)*ones(1,length(CR_th)),'r-');
    xlabel('Bandwidth (Hz)');
    ylabel('STD of position-error estimate in nm');
    %axis([2^rl 2^rh 10^-4 10^2]);
end
hold off;
```

A.1.3 Figure 3-4: STD of the Position-Error Versus SNR

```
%STD of position-error estimate versus SNR at various Bandwidths
clear
figure;
N = [2500, 2^16, 2^18];
A = 1;
w = pi/2;
Lambda = 250*10^-9;
conv = Lambda^2/((2*pi)^2);
for m = 1:3
   m
   Nfft = N(m);
   n = 1:N(m);
   sig = A*cos((w)*n);
   range = -60:20;

   %Calculate maximum frequency of orig signal
   Yo = (1/N(m))*fft(sig-mean(sig),Nfft);
   [max_Yo, w_maxo] = max(abs(Yo).^2);
   Tho = angle(Yo(w_maxo));

   iter = 1;
   for raw_snr = range
      snr = 10^(raw_snr/10);
      count = 1;
      for i = 1:25

         nois_amp = sqrt((A^2)/(2*snr));
         nois = nois_amp * randn(size(sig));
         nsig = sig + nois;

          Y = (1/N(m))*fft(nsig-mean(nsig),Nfft);
         [max_Y, w_max] = max(abs(Y).^2);

         w_hat(count) = (w_max-1)/(Nfft/2);
         A_hat(count) = sqrt((4/N(m))*max_Y);
         Th_hat(count) = angle(Y(w_maxo));

         count = count + 1;
      end
      CR_th(iter) = 1/(N(m)*snr);
```

```
    Th_var(iter) = var(Th_hat);
    iter = iter + 1;
  end
  semilogy(range,(conv*Th_var).^.5);
  hold on;semilogy(range,(conv*CR_th).^.5,'--');
  hold on;
  semilogy(range,(1*10^-9)*ones(1,length(CR_th)),'r-');
  xlabel('SNR in dB');
  ylabel('STD of position-error estimate in meters');

end

hold off;
%_______________________________________
%Ballpark SNR according to Feng:
g = 0.1;
D = 1.58e-2;
dx = 1e-8;
frac = 1.6e-2;
el = 1.6e-19;
SNRbpk = 10*log10(g*D*(dx^2)*frac/(4*el))
```

Appendix B

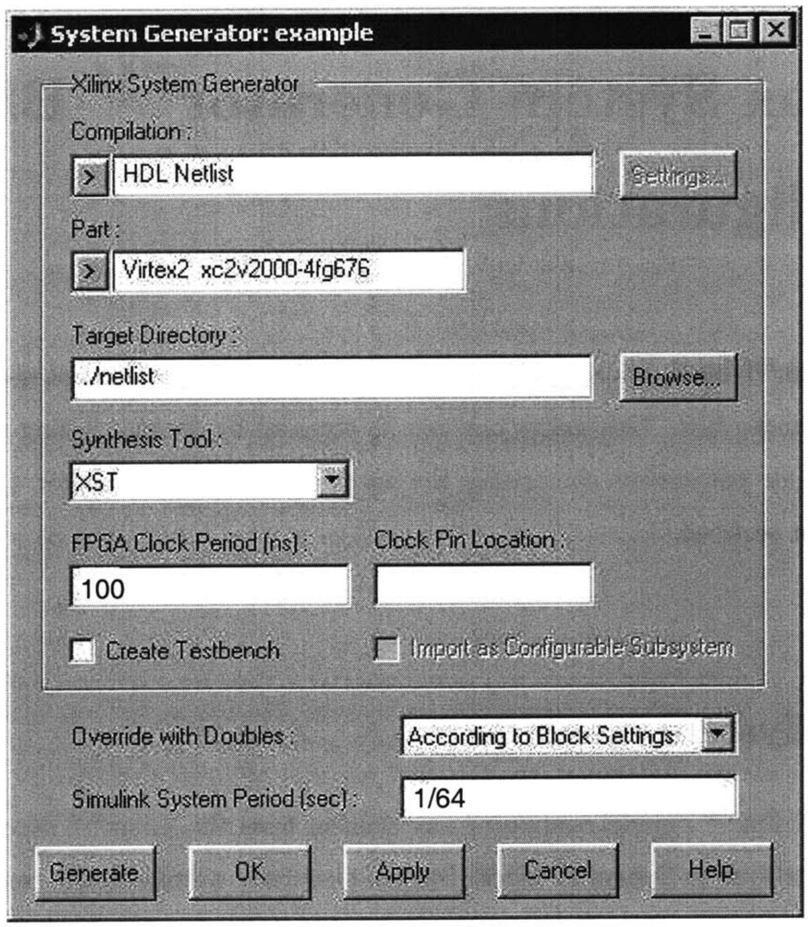
Xilinx System Generator^{TM} Blockset Configurations

Each Xilinx^{TM} block has several controls and configurable parameters, seen in its block parameters dialog box. This dialog box can be accessed by double-clicking on the block. The block-specific parameters comprising the hardware models in Chapter 4 are shown in the subsequent sections.

B.1 System Generator

Every Simulink^{TM}. model containing any element from the Xilinx^{TM} Blockset must contain at least one System Generator block. System Generator automatically compiles designs into low level representations. Designs are compiled and simulated using the *System Generator* block. Pressing the **Generate** button instructs System Generator to compile a portion of the design into equivalent low level results. The portion that is compiled is the subtree whose root is the subsystem containing the block. The compilation type (under **Compilation**) specifies the type of result that should be produced. The possible types are **HDL Netlist** and various varieties of hardware co-simulation.

The *HDL Netlist* is the type used most often. In this case, the result is a collection of VHDL and EDIF files, and a few auxiliary files that simplify downstream processing. The collection is ready to be processed by a synthesis tool (e.g., XST), and then fed to the Xilinx physical design tools (i.e., ngdbuild, map, par, and bitgen) to produce a configuration bitstream for a Xilinx FPGA.



Parameters specific to the System Generator block are:

• Compilation: Specifies the type of compilation result that should be produced when the code generator is invoked.

- Part: Defines the FPGA part to be used.

- Target Directory: Defines where System Generator should write compilation results.

- Synthesis Tool: Specifies the tool to be used to synthesize the design.

- FPGA Clock Period: Defines the period in nanoseconds of the hardware clock. The value need not be an integer.

- Clock Pin Location: Defines the pin location for the hardware clock.

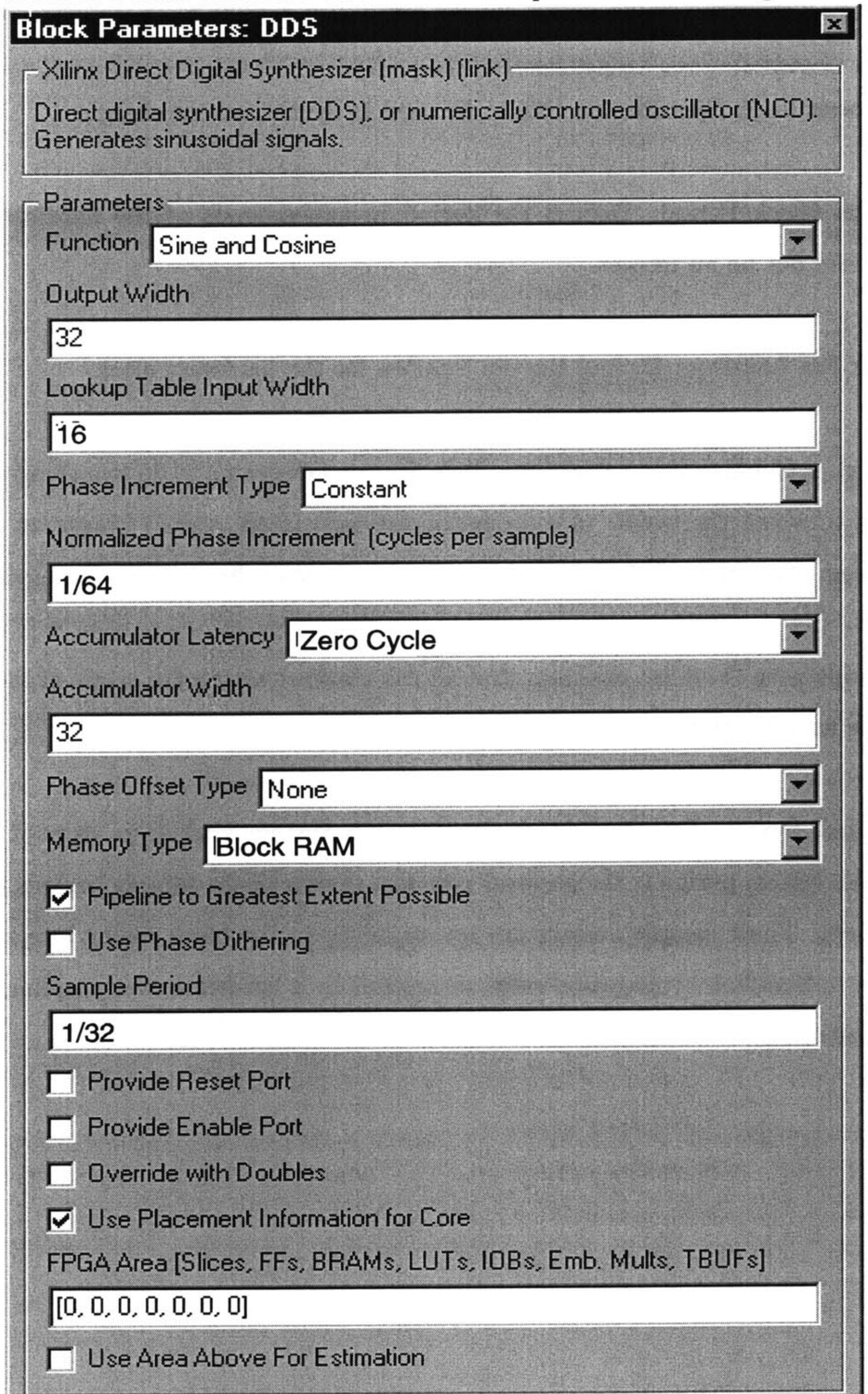
- Import as Configurable Subsystem: Tells System Generator to do two things: 1) Construct a block to which the results of compilation are associated, and 2) Construct a configurable subsystem consisting of block and the original subsystem from which the block was derived.

- Override with Doubles: Specifies that all calculations within the scope of the block should be done using double precision arithmetic

- Simulink System Period: Defines the Simulink System Period, in units of seconds. The Simulink system period is the greatest common divisor of the sample periods that appear in the model. These sample periods are set explicitly in the block dialog boxes, inherited according to Simulink propagation rules, or implied by a hardware oversampling rate in blocks with this option.

B.2 Local Oscillator

The DDS block is a local oscillator that uses a lookup table scheme to generate sinusoids.



- Function: specifies the function that the block will calculate.

- Output Width: number of bits in the output signal; value must be between 4 and 32 inclusive.

- Lookup Table Input Width: specifies the number of address bits into the sine/cosine lookup table; value must be at least 3. The width must be less than or equal to min(a,b) where a is the accumulator width, and b is 16 (if block RAM is used) or 10 (if distributed RAM is used).

- Normalized Phase Increment Type: specified to be either constant or register. Choice of register activates optional ports on the block.

- Phase Increment: specifies value of phase increment constant, a multiple of 2π. The number of bits is determined in one of two ways. If the increment type is Register, the number of bits is set to the width of the data port. If the increment type is Constant, the number of bits is inferred from the phase increment value.

- Accumulator Latency: specifies the latency in the phase accumulator to be zero or one.

- Accumulator Width: specifies the phase accumulator width; value must be between 3 and 32 inclusive.

- Phase Offset Type: specifies phase offset to be Constant, Register, or None. Choice of register activates optional ports on the block.

- Normalized Phase Offset: specifies value of phase offset constant, as a multiple of 2π. The number of bits is determined in one of two ways. If the offset type is Register, the number of bits is set to the width of the data port. If the offset type is Constant, the number of bits is inferred from the phase offset value.

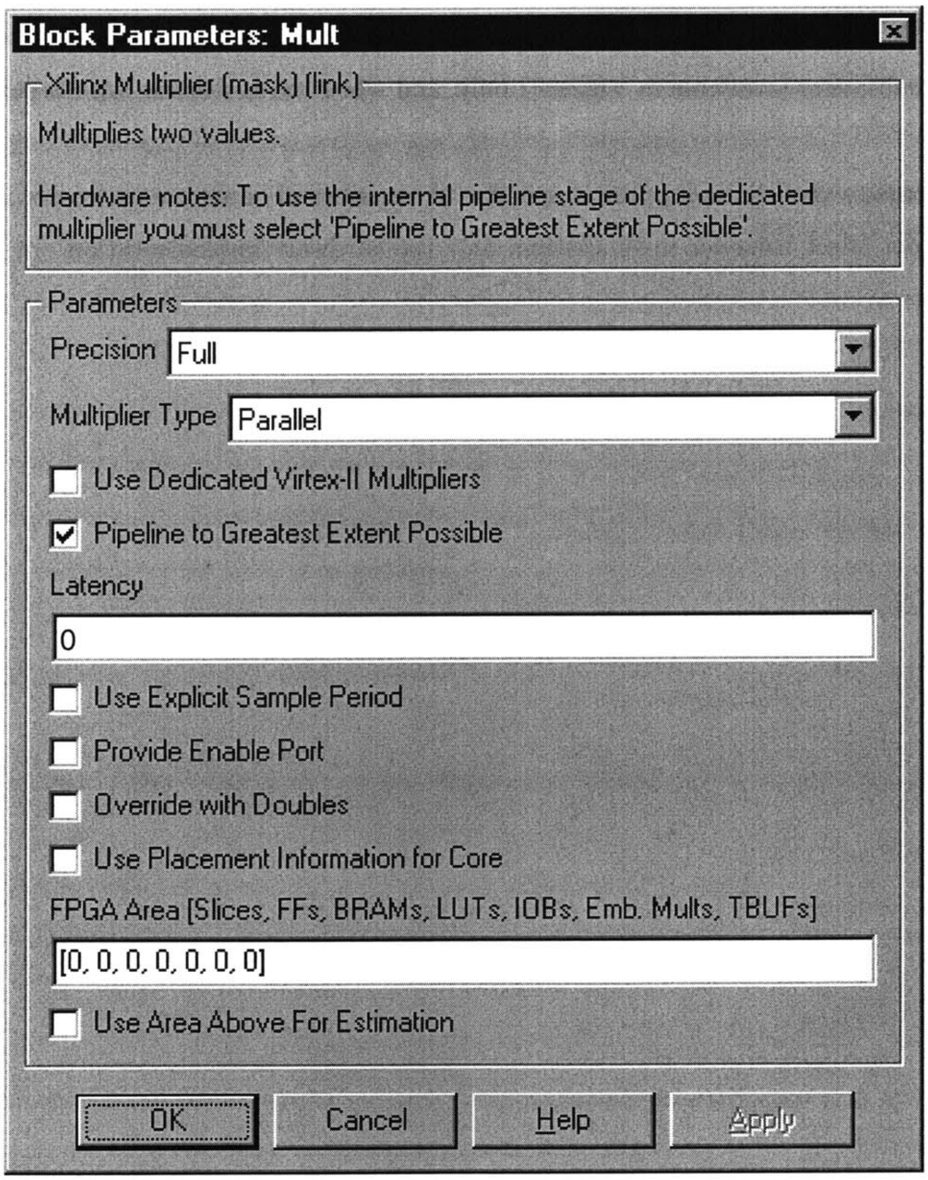
• Memory Type: directs the block to be implemented either with distributed or block RAM.

• Pipeline to the Greatest Extent Possible: when checked, the implementation is fully pipelined.

• Use Phase Dithering: when checked, a dither sequence is added to the result of the phase accumulator.

B.3 Multiplier

The Xilinx^{TM} Mult block implements a multiplier.



Parameters specific to the Mult block are:

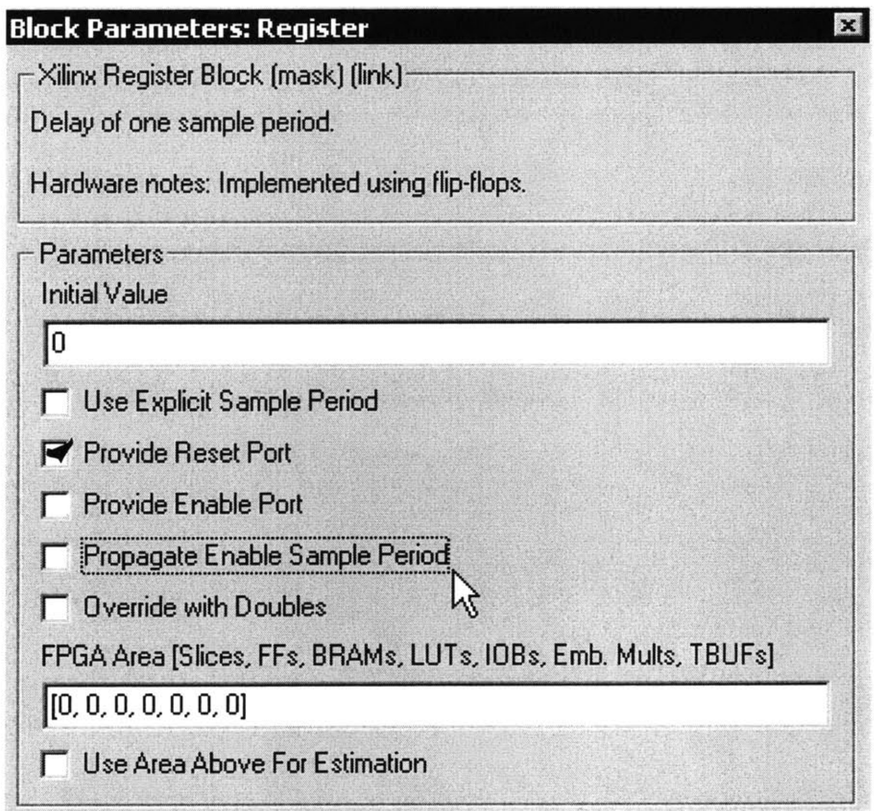
• Multiplier Type: directs the implementation to be either parallel or sequential.

• Pipeline to Greatest Extent Possible: directs the core to be pipelined to the fullest extent possible. Use Dedicated Virtex-II Multipliers: when checked, directs the core to use embedded multipliers (available in Virtex-II only, and when the multiplier type is parallel).

• Hardware Over-Sampling Rate: specifies the number of hardware cycles per input sample; does not affect behavior in simulation, only the hardware implementation.

B.4 Register

The Xilinx Register block models a D flip flop-based register, having latency of one sample period. The block has one input port for the data and an optional input reset port. The initial output value is specified by the user in the block parameters dialog box. Data presented at the input will appear at the output after one sample period. Upon reset, the register assumes the initial value specified in the parameters dialog box.



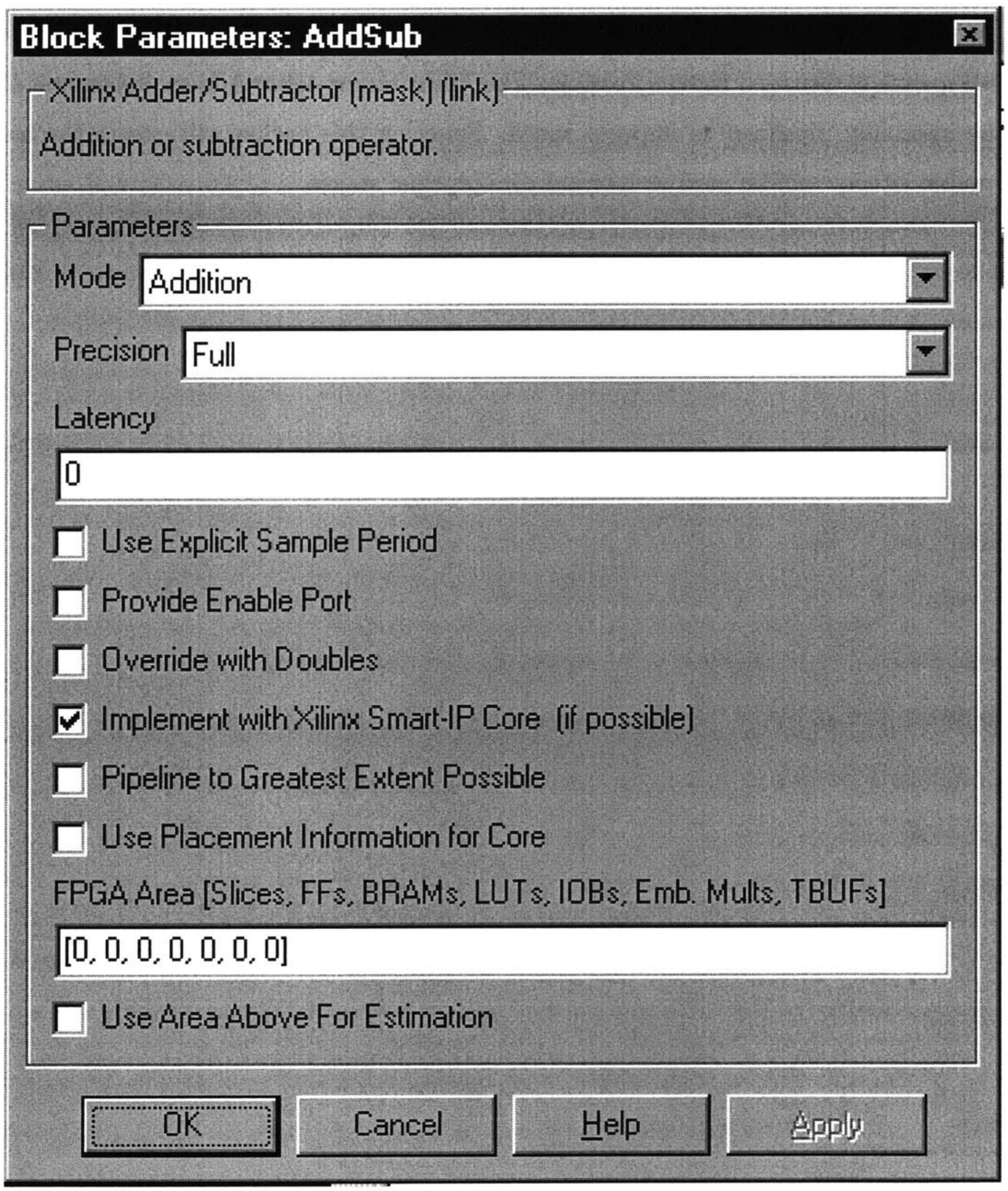
Parameters specific to the block are:

- Initial Value: specifies the initial value in the register.

- Propagate Enable Sample Period: This option is available only when the enable port is selected. When this option is specified, data at the input is sampled to the output at the same rate as the enable signal. The enable signal has to run at a multiple of the block's sample rate. Reset port should be driven at the same rate as the enable signal, when the output is driven at the same rate as the enable port.

B.5 Adder/Subtractor

The AddSub block implements an adder/subtractor.



Parameters specific to the AddSub block are:

• Mode: specifies the block operation to be Addition, Subtraction, or Addition/ Subtraction. When Addition/Subtraction is selected, the block operation is determined by the sub input port, which must be driven by a 1-bit unsigned signal. When the sub input is 1, the block performs subtraction. Otherwise, it performs addition.
• Pipeline to Greatest Extent Possible: The Xilinx Smart-IPtm Adder/ Subtracter Core can be internally pipelined to improve speed. Selecting this option will ensure the maximum usable latency will be used as internal core pipeline stages.

Bibliography

[1] "International Technology Roadmap for Semiconductors,"2003. http://public.irts.net.

[2] J.T. Hastings, *Nanometer-Precision Electron-Beam Lithography with Applications in Integrated Optics.* PhD thesis, Massachusetts Institute of Technology, 2003.

[3] P. Rai-Choudhury, editor, *SPIE Handbook of Microlithography, Micromachining, and Microfabrication: Volume I* Table 2.3 Comparison of Gaussian spot, raster scan mask making systems. http://www.nnf.cornell.edu/spiebook/spie5.htm

[4] "MEBES 5500," 2003. http://www.etec.com/products/assets/mask products/mebes 5500 printable datasheet.pdf.

[5] "Leica VB6 Series," 2004. http://www.leica-microsystems.com/website/lms.nsf.

[6] M.J. Pawitter, *Critical Dimension and Image Placement Issues for Step and Flash Imprint Lithography Templats.* Molecular Imprints, Inc. Technical Article, 2004. http://www.molecularimprints.com/Technology/tech articles/Moto Template BACUS-Photomask 2002.pdf.

[7] H.I. Smith, *Submicron- and Nanometer-Structures Technology.* Lecture Notes for Course 6.781, Massachusetts Institute of Technology, 2002. Sudbury, MA: NanoStructures Press, 1994.

[8] J. Ferrera, *Nanometer-Scale Placement in Electron-Beam Lithography.* PhD thesis, Massachusetts Institute of Technology, 2000.

[9] J.T. Hastings, Feng Zhang, M.A. Finlayson, J.G. Goodberlet, and H.I. Smith, "Two-Dimensional Spatial-Phase-Locked Electron-Beam Lithography via Sparse Sampling," *J. Vac. Sci. Technol. B*, vol. 18, Nov/Dec. 2002, pp. 3268-3271.

[10] R.H. Hosking, *Pentex Digital Receiver Handbook.* Pentek Inc., 2001.

[11] R.E. Best, *Phase-Locked Loops: Design, Simulation, and Applications.* New York: McGraw-Hill, 1999.

[12] R. Gallager, *6.450 Principles of Digital Communications.* Cambridge, MA: Massachusetts Institute of Technology, 2002.

[13] A.S. Willsky, G.W. Wornell, and J.H. Shapiro, *Stochastic Processes Detection and Estimation*, Section 3.4. Cambridge, MA: Massachusetts Institute of Technology, 2002.

[14] A.S. Willsky, G.W. Wornell, and J.H. Shapiro, *Stochastic Processes Detection and Estimation*, Section 3.B. Cambridge, MA: Massachusetts Institute of Technology, 2002.

[15] F. Zhang. PhD thesis, Massachusetts Institute of Technology, 2004. (In preparation).

[16] M. Schattenburg, 2003. NSL Group Meeting, September 5, 2003.

[17] T. Barwicz, 2004. Private Communication.

[18] *Xilinx System Generator^{TM} User Guide: System Generator Blocksets*, 2003. http://support.xilinx.com/products/software/sysgen/app docs/user guide Chapter 3 Section 3 Subsection 1.htm.

[19] *Xilinx System Generator^{TM} User Guide: Local Oscillator*, 2003. http://support.xilinx.com/products/software/sysgen/app docs/userguide Chapter10 Section 3 Subsection 16.htm.

[20] *Xilinx System Generator^{TM} User Guide: Multiplier*, 2003. http://support.xilinx.com/products/software/sysgen/app docs/user guide Chapter 10 Section 3 Subsection 36.htm.

[21] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," *Proceedings of the Sixth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, 1998.

[22] *Simulink^{TM} Reference: Lookup Table*, 2003. http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/slref2.html.