

# Digital Phase Tightening for Millimeter-wave Imaging

Khoa M. Nguyen, Anthony Accardi, Helen Kim\*, Gregory W. Wornell, Charles G. Sodini

Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139 USA

\*MIT Lincoln Laboratory, 244 Wood Street, Lexington, Massachusetts 021420 USA

kmnguyen@mit.edu

## Abstract

A new technique called digital phase tightening reduces phase noise from receiver front-end circuits to allow precise phase estimation for digital beamforming in millimeter-wave (MMW) imaging applications. This is achieved by leveraging the large ratio between the MMW carrier frequency and the relatively low frame rates in imaging applications. By mixing down to an intermediate frequency (IF) and then averaging over many samples to estimate the phase, we reduce phase noise and attain phase error of the MMW beamformer in the femtosecond range. A test chip demonstrating the phase tightening concept was designed and fabricated using 0.13  $\mu\text{m}$  CMOS, and we show that an RMS error of 3.5fs is feasible with this technique.

## I. Introduction

Millimeter wavelengths are small enough to offer sufficient spatial resolution for certain imaging applications, such as automotive radar and concealed weapons detection. Advances in silicon processes have developed devices that are capable of operating at these frequencies, which has led to the potential for low cost MMW imaging, provided that the circuit design can meet the challenging performance specifications for these applications.

We are interested in the 77GHz band designated for automotive radar. We envision an imaging system coupled with range finding radar for more intelligent collision avoidance. Fig. 1 shows the MMW imaging system that consists of an array of antenna receivers. Each receiver determines the phase and amplitude of its received signal and sends the information to a central processor which beamforms the aggregate digital data to create an image at a frame rate of 10fps. We focus on the phase because beamforming requires precise phase estimates, which is a challenge for MMW signals with periods on the order of picoseconds. Our system targets 6cm pixel resolution at 30m distance. This corresponds to a beamwidth of  $0.1^\circ$ , which requires one thousand antenna elements with  $\lambda/2$  spacing. The time delays of signals arriving at each antenna with this pixel resolution is on the order of femtoseconds! In addition, noise contributions from the analog circuits further degrade the phase estimates and the image quality. This paper discusses an approach for estimating phase in a MMW imaging system with femtosecond resolution in the presence of front-end circuits that contribute picoseconds of jitter.

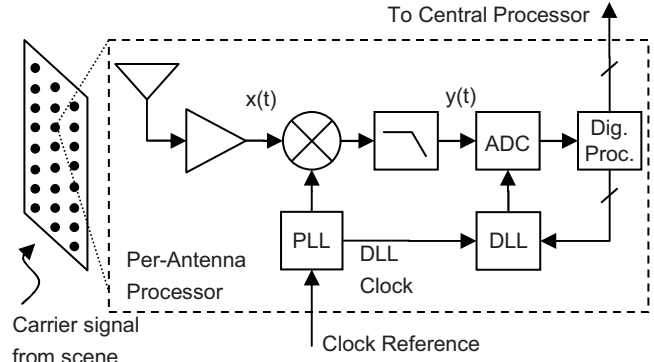


Fig. 1. Block diagram of an antenna imaging array with a single antenna receiver shown.

## II. Digital Phase Tightening

Our method to solve this problem has been coined digital phase tightening. We model the received input signal,  $x(t)$ , as a single tone at the carrier frequency  $\omega_c$  and a time delay  $\delta$  expressed in (1). We want to estimate the signal phase  $\Delta$ , equivalent to  $\omega_c \cdot \delta$ , relative to the reference clock and in the presence of zero-mean phase noise  $\phi(t)$ .

$$x(t) = \exp(j[\omega_c(t + \delta) + \phi(t)]) \quad (1)$$

We first mix the MMW signal with a local oscillator (LO) operating at  $\omega_{LO}$  to an intermediate frequency  $\omega_{IF}$ . The result  $y(t)$  is expressed in (2). The time delay  $\delta$  has been scaled up by  $\omega_c/\omega_{IF}$ , which is more manageable and easier to estimate. However, this benefit comes at the cost of increased noise, which is introduced by the mixer and imperfect LO and is represented as  $\zeta(t)$ .

$$y(t) = \exp\left(j\left[\omega_{IF}\left(t + \frac{\omega_c}{\omega_{IF}}\delta\right) + \phi(t) + \zeta(t)\right]\right) \quad (2)$$

The IF signal is sent to a feedback loop that consists of an analog-to-digital converter (ADC), a delay-locked loop (DLL), and some digital logic. By locking to the DLL clock, the DLL generates evenly-spaced rising edges of the clock signal throughout its period. Based on the output of the ADC, the digital logic selects the rising edge from the DLL that triggers the ADC near the IF signal's zero crossings. The digital control signal represents the phase of the input signal relative to the DLL clock. Assuming an IF of about 100MHz, we can oversample from the IF to the frame rate to capture about one million samples, which are averaged in the digital domain. This reduces the zero-mean noises by the square root of the number of samples and essentially tightens our estimate to the true phase of the received signal.

### III. Implementation

A test chip was designed and fabricated using 0.13  $\mu\text{m}$  CMOS devices in IBM's BiCMOS8HP process. The block diagram of the circuits in the implemented system is presented in Fig. 2. These blocks operate after the received signal has been mixed down to the IF. Since we focus on the phase and not the amplitude, the ADC has been replaced with a simple comparator to prove the phase tightening concept. The comparator determines whether the IF input signal is above or below the zero reference level and is sufficient for determining the zero-crossing. The digital logic, which consists of a counter, reads the comparator output and determines whether to increment or decrement the output of the DLL. The counter controls a MUX which selects one of the delay taps from the DLL to be the comparator clock.

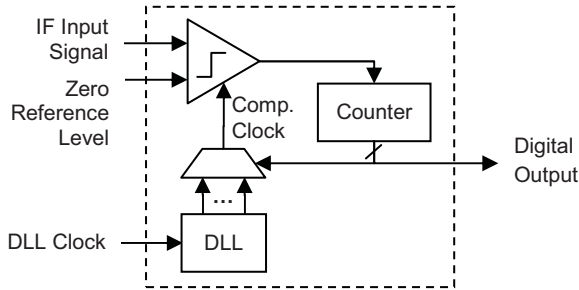


Fig. 2. Block diagram of proof-of-concept phase tightening system.

The DLL is composed of a variable delay line in a feedback loop with a phase detector, charge pump, and loop filter. The output of the loop filter drives the control signal of each delay cell. Each stage in the delay line consists of an inverter pair which takes differential inputs and has cross-coupled inverters to ensure the outputs are complementary. The variable delay was achieved by controlling the resistance to shunt capacitors as described in [1]. Compared to the standard method of current-starving the inverters, this architecture has lower gain, which reduces sensitivity to noise on the control voltage at the cost of lower tuning range. The outputs of each delay cell are buffered and routed to the MUX.

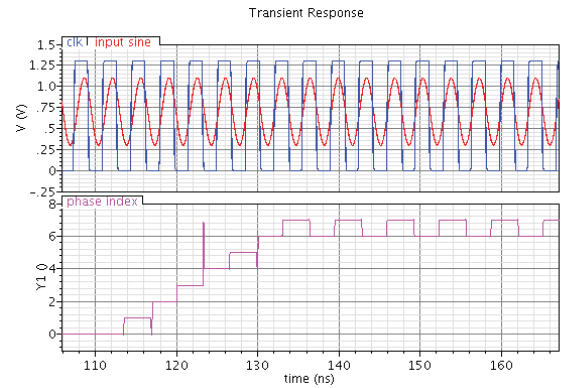
For this implementation, the clock and input signal are the same frequency. The system locks onto the rising zero-crossings of the input signal. However, it is possible to use a higher clock frequency to use both the rising and falling zero-crossings.

### IV. Simulation Results

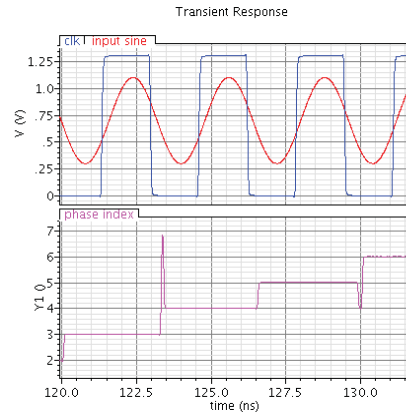
Fig. 3(a) shows the results of a Spectre simulation that demonstrates the system's operation as it locks to the zero-crossings of an input sine wave. The top plot shows the IF input signal relative to the comparator clock, which is the clock edge that has been MUXed from the DLL's delay line. The bottom plot shows the digital code output of the counter from Fig. 2, which is labeled as phase index with each increment representing 1 LSB. Initially the IF signal's rising edge lags the clock's rising edge. The glitches in this plot are the result of transitioning bits and are removed in the chip implementation

by latching the outputs. When the comparator is triggered by the clock edge, its output increments the counter and selects the next delayed clock edge to trigger the next cycle. While the IF signal is lagging, the counter continues to increment and select more delayed versions of the DLL clock. Fig. 3(b) zooms in on this behavior. We observe the comparator clock's rising edge is increasingly delayed relative to the IF signal's rising zero crossing as the phase index increments. Eventually, the IF signal will lead the comparator clock. When this occurs the comparator decrements the counter and selects the previous delayed clock edge, causing the IF signal to lag the clock again.

This process will repeat, and the digital output will alternate between these two values as it selects between the two DLL clock edges that trigger the comparator around the IF signal's zero crossings. We see that when the phase tightening system is locked, the comparator clock and the IF signal are in phase. We average these digital phase index values to obtain the phase estimate.



(a)



(b)

Fig. 3. Simulation of the comparator clock locking to the IF signal's zero crossings with the counter's digital code output. (a) overall locking behavior. (b) zoomed-in plot demonstrating the incrementally delayed comparator clock.

### V. Measurement Results

The measurement setup is shown in Fig. 4. Two synchronized signal generators were used to create the DLL clock and IF signal. These signals had the same frequency but could be programmed with different phases. The phase tightening system was characterized by sweeping the phase of one signal relative

to the other via a computer and General Purpose Interface Bus (GPIB). The phase difference was ramped in increments of  $0.5^\circ$ . Since these were analog inputs and variations were observed in the phase increments, an oscilloscope was used to measure and record the actual phase difference. For each phase input increment, a logic analyzer was used to sample and store the digital outputs. This data was then post-processed in Matlab, and a digital phase estimate was calculated for each phase input. A pattern generator was used to control all digital inputs into the test chip.

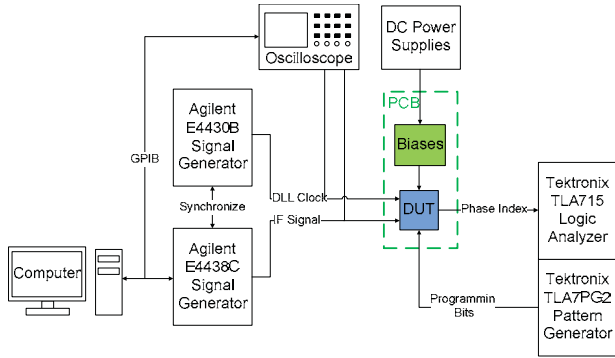


Fig. 4. Test measurement setup.

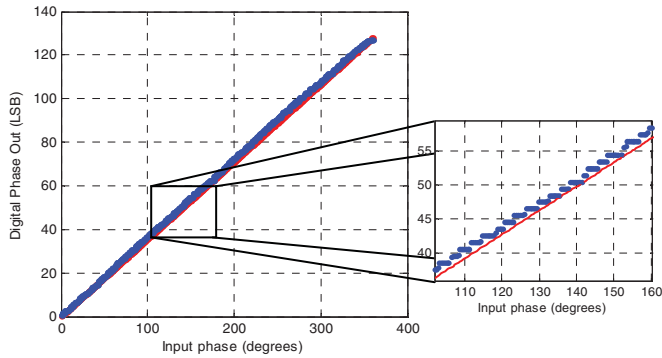


Fig. 5. Measured digital phase output versus input phase.

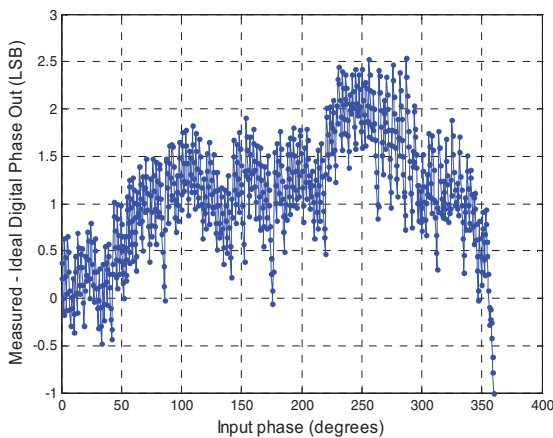


Fig. 6. Measured error between phase output and ideal transfer curve.

We tested the phase tightening system with clock and IF signals at 175MHz. The DLL produces 128 rising clock edges with 45ps delay steps. The blue data points in Fig. 5 show the measured digital output plotted against the input phase. The red

curve shows the ideal transfer curve for  $360^\circ$  phase sweep with 128 delay steps. The quantization steps of the phase to digital conversion can be observed. Taking the difference between these signals gives the error plot shown in Fig. 6. This plot shows the accumulation of nonlinearities in the delay line and mismatches in the MUX. However, these nonlinearities are consistent through multiple measurement runs, so they are calibrated out in the digital domain. The resulting error plot is shown in Fig. 7. The RMS error of this plot is 0.31LSB, corresponding to 13.8ps at the IF or 31.3fs when referenced to a 77GHz carrier.

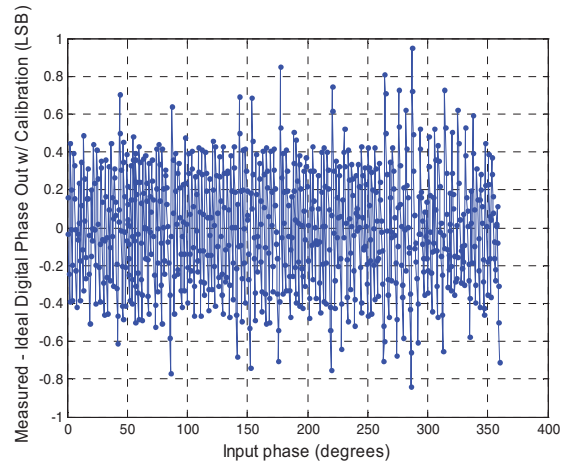


Fig. 7. Measured error plot after calibration.

## VI. Digital Phase Tightening in MMW Imaging

In our MMW imaging system, we require phase estimates on the order of femtoseconds at the carrier frequency. The effect of quantization is too large for direct use in a MMW imaging system. However, our target frame rate is 10fps, which is low compared to the IF. This allows ample time to capture and average a large number of samples to produce a better phase estimate. For each frame, we assume a generous allocation of 90ms for beamforming and image processing in the digital domain, which gives 10ms of time remaining for data sampling. Our current test setup has an IF of 175MHz which gives approximately one million samples to average for each frame.

Although we want to minimize the noise, having some noise is beneficial since we are averaging. When the phase noise is small compared to the delay step, the system will always alternate between two values and the phase estimate is limited by the quantization of the delay line. However, in the presence of noise, the system will alternate beyond the two values which contribute more information during the averaging process and will result in estimates with lower RMS error [2].

Noise was introduced into the system using the phase modulation feature of the Agilent E4438C signal generator to modulate the generator's internal noise source into the IF signal. The phase noise level could be controlled and measured on a spectrum analyzer, and the corresponding timing jitter could be measured using an oscilloscope. We measure the RMS error for different levels of averaging and jitter and observe the improvement in phase estimation.

Fig. 8 shows measured RMS error versus the number of averaging samples,  $N$ , for different levels of timing jitter. Due to limitations in the logic analyzer, the maximum number of stored data for averaging is currently limited to 225ksamples. We observe that the RMS error level is directly related to the amount of jitter introduced, which we expect because more noise should result in larger error given the same amount of averaging. We also observe that the noise stays constant and then rolls off for increasing  $N$ . Once it rolls off, the error is reduced by the expected  $1/\sqrt{N}$ .

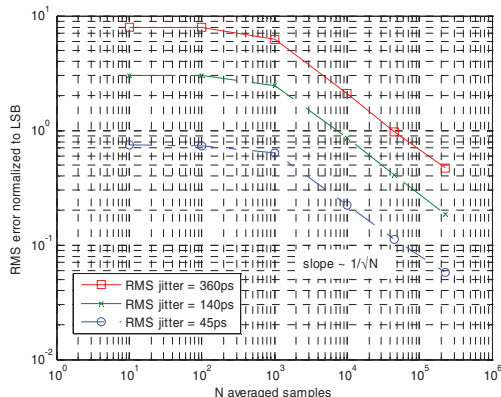


Fig. 8. Measured RMS error plotted against  $N$  averaged samples for various levels of timing jitter.

The constant portion of the plots is due to the characteristic of the added phase noise. The amplitude of the modulated phase noise can be adjusted by the signal generator, but the single-sided phase noise bandwidth remains constant at 100kHz. The phase noise bandwidth determines the rate at which the jitter is changing. Since the bandwidth is approximately 1000 times slower than the operating frequency of 175MHz, we must capture at least that many samples before the benefits of averaging can be observed. Otherwise, we can imagine that the noise is changing so slowly compared to the sampling rate that the phase tightening output only alternates between two values. A larger phase noise bandwidth would mean that the phase is varying more rapidly and the system would require fewer samples for the averaging to take effect, so the roll off would occur at a lower  $N$ .

From previous work in [3], the dominant source of phase noise in the imaging receiver is the MMW LO generated by the PLL. The timing jitter from the PLL was about 825fs. Phase noise from the PLL is additive as seen from (2), which means that the PLL timing jitter is scaled up by the ratio of  $\omega_{LO}/\omega_{IF}$ . Scaling the jitter from this PLL to the IF would make it comparable to the red curve of 360ps RMS jitter in Fig. 8. However, the bandwidth of the noise in [3] is 10MHz which is two orders of magnitude higher than the bandwidth of the noise source used in these measurements. Given this noise, we expect the RMS error to roll off two orders of magnitude sooner at approximately  $N = 10$ , and the RMS error to improve by  $\sqrt{100}$ . Averaging 225ksamples for jitter of 360ps gives an RMS error of 0.46LSB. A  $\sqrt{100}$  improvement gives 0.046LSB which corresponds to 2.0ps of error at the IF or 4.5fs of error at the MMW carrier.

We expect to average up to one million samples which reduces the error further. Fig. 9 shows a Monte Carlo simulation of the RMS error given the same level of jitter but changing at a more rapid rate to match the PLL's 10MHz phase noise bandwidth. The RMS error after averaging one million samples is 0.034LSB, which is 1.5ps of error at the IF or 3.5fs of error at the carrier. Therefore the specification for femtosecond estimates required for the digital beamforming is feasible with the use of phase tightening.

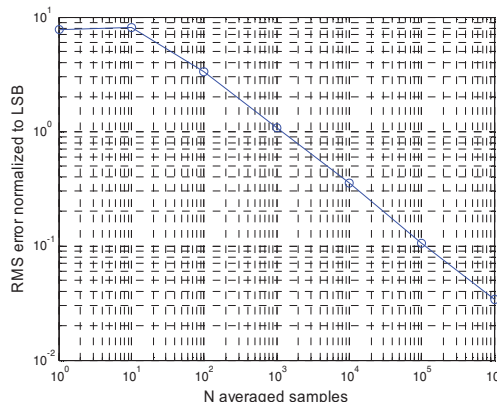


Fig. 9. Monte Carlo simulation result of expected RMS error given PLL phase noise with 10MHz bandwidth.

## VII. Conclusion

Digital phase tightening reduces jitter from MMW front-end circuits through averaging and achieves error on the order of femtoseconds. A test chip was implemented, and it demonstrated how the technique estimates phase with effectively 3.5fs of RMS error at the carrier frequency for the expected amount of phase noise and averaging in the system. A target pixel resolution of 6cm at 30m distance or  $0.1^\circ$  beamwidth for a 77GHz automotive imaging application is possible using digital phase tightening.

## Acknowledgements

The authors acknowledge the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation subsidiary. The authors also wish to acknowledge the Massachusetts Institute of Technology Center for Integrated Circuits and Systems for funding.

## References

- [1] M. Johnson and E. Hudson, "A variable delay line PLL for CPU-coprocessor synchronization," *IEEE Journal of Solid-State Circuits*, vol. 23, No. 5, pp. 1218-1223, Oct. 2008.
- [2] R. Skartlien and L. Oyeaug, "Quantization error and resolution in ensemble averaged data with noise," *IEEE Trans. Instr. and Meas.*, vol. 54, pp. 1303-1312, June 2005.
- [3] K. Nguyen, H. Kim, and C.G. Sodini, "A 76GHz PLL for mm-wave imaging applications," *IEEE Microwave Symposium Digest*, May 2010.