Journal of VLSI Signal Processing 36, 57–71, 2004

FPGA Implementation of Carrier Synchronization for QAM Receivers

CHRIS DICK
Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124, USA

FRED HARRIS
CUBIC Signal Processing Chair, College of Engineering, San Diego State University, CA 92182, USA

MICHAEL RICE
Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84602, USA

Received February 14, 2002; Revised August 27, 2002; Accepted October 15, 2002

Abstract. Software defined radios (SDR) are highly configurable hardware platforms that provide the technology for realizing the rapidly expanding third (and future) generation digital wireless communication infrastructure. While there are a number of silicon alternatives available for implementing the various functions in a SDR, field programmable gate arrays (FPGAs) are an attractive option for many of these tasks for reasons of performance, power consumption and flexibility. Amongst the more complex tasks performed in a high data rate wireless system is synchronization. This paper examines carrier synchronization in SDRs using FPGA based signal processors. We provide a tutorial style overview of carrier recovery techniques for QPSK and QAM modulation schemes and report on the design and FPGA implementation of a carrier recovery loop for a 16-QAM modern. Two design alternatives are presented to highlight the rich design space accessible using configurable logic. The FPGA device utilization and performance for a carrier recovery circuit using a look-up table approach and CORDIC arithmetic are presented. The simulation and FPGA implementation process using a recent system level design tool called *System Generator*™ *for DSP* described.

Keywords: FPGA, software define radio, QAM, synchronization, wireless communication, carrier recovery, system generator, CORDIC

1. Introduction

The last two-decades has borne witness to a steady trend of migrating many radio functions, traditionally performed by analog processing tasks, to DSP based implementations [1]. In conjunction with this DSP insertion, we have seen the boundary between the analog and digital segments in the signal conditioning chain move inexorably towards the antenna. The implementation of these high-performance digital communication systems has been made possible by advances in semiconductor process technology in the form of application specific standard parts (ASSPs), full custom silicon chips, instruction set based digital signal processors (DSPs) and high performance general-purpose processors (GPP). In the early 1990's field programmable gate arrays (FPGAs) also played a role in digital communication hardware where they were often applied as glue logic to support bus interfacing, complex state machines and memory controller tasks. However, the Silicon landscape is now changing. In recent years FPGA technology has undergone revolutionary changes. Not only have gate densities and clock speeds of recent generation FPGAs experienced dramatic improvements, but integrated functions like dedicated high-speed hardware multipliers

(Virtex-II® [2]) and embedded processors in platform FPGAs (Virtex-II Pro [2]) are now providing the communication system architect with a highly configurable silicon engine that can be used for realizing sophisticated high-MIPs (millions of instructions per second) real-time signal processing functions.

The ever-increasing demand for mobile and portable communication forces two competing requirements on system design: (1) the requirement for high-performance systems employing advanced signal processing techniques to allow operation with very small implementation losses, and (2) the requirement to respond to market and fiscal pressures to easily accommodate evolving and fluid standards. Software defined radios (SDRs) are emerging as a viable solution for meeting the conflicting demands in this arena. SDRs support multimode and multi-band modes of operation and allow service providers an economic means of future-proofing these increasingly complex and costly systems. While first generation soft radios may have used a combination of ASIC, ASSP and GPP technologies, state-of-the-art FPGAs are now moving us closer to the true objectives of the SDR, without the need (and impact on flexibility) for ASICs and/or ASSPs.

Many sophisticated signal processing tasks are performed in a SDR, including advanced compression algorithms, power control, channel estimation, synchronization, equalization, forward error control, adaptive antennas, rake processing in a WCDMA (wideband code division multiple access) system and protocol management. This paper examines maximum likelihood carrier phase synchronization for QAM (quadrature amplitude modulation) based SDR personalities.

First, a tutorial style overview of the problem is presented. Next the FPGA mechanization of a 16-QAM carrier recovery loop is described. Two design alternatives are presented to highlight the rich design space accessible using configurable logic. The FPGA device utilization and performance for a carrier recovery circuit using a look-up table approach and CORDIC arithmetic are presented. The simulation and FPGA implementation process using a recent system level design tool called *System Generator for DSP* is also described.

2. Carrier Recovery in an Digital Receiver

The basic operations required by an all digital QPSK or QAM receiver are illustrated in Fig. 1. QPSK and QAM systems carry information on the amplitudes of quadrature carriers. The quadrature down-conversion

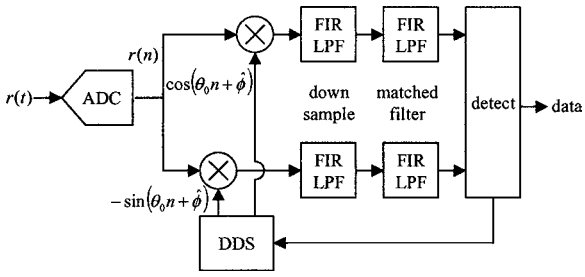

Figure 1. Block diagram of basic QPSK/QAM digital receiver.

and matched filter operations produce estimates of the quadrature amplitudes that are the basis of the data decisions. The role of carrier phase synchronization is to perform the quadrature down-conversion using phase coherent replicas of the quadrature carriers.

There are many options for implementing carrier phase and frequency synchronization in a digital communication system. At the heart of all synchronizers is the phase-locked loop (PLL). An all-digital receiver can be implemented with a digital phase-locked loop (DPLL) as shown in Fig. 2. This DPLL employs a proportional plus integral loop filter formed by a scaled digital integrator and a scaled direct path. The filter coefficients K_p and K_l control the PLL bandwidth and damping factor. In the digital implementation, the VCO takes the form of a direct digital synthesizer (DDS). The phase detector is implemented using the arctangent operation suggested by the ATAN block.

The most complex component in the loop is the phase detector. Since the phase of QPSK or QAM signals is data dependent, the phase detector must strip the modulation from the received signal and produce a signal

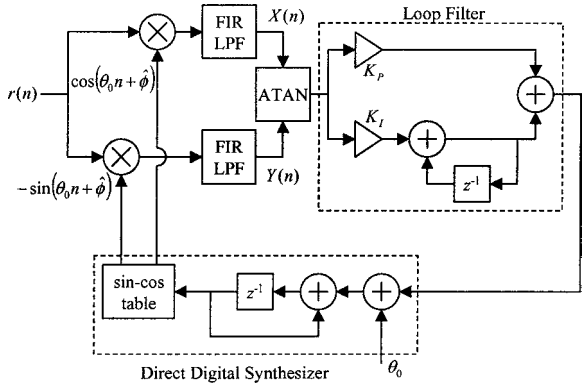

Figure 2. Digital phase locked loop for carrier phase synchronization.

proportional to the phase difference between the local generated quadrature carriers and those of the received signal. The complexity of the phase detector can be reduced by computing a signal proportional to the sine of the phase difference $\Delta\phi = \phi - \hat{\phi}$. Note that $\sin(\Delta\phi)$ is monotonic with $\Delta\phi$ for $-\pi/2 \leq \Delta\phi \leq \pi/2$ and is a good phase estimator in that interval. The periodicity of the sine function produces the $\pi/2$ phase ambiguity associated with the phase detector. For small $\Delta\phi$, $\sin(\Delta\phi) \approx \Delta\phi$ so that the sine function approximates the ideal phase detector for small $\Delta\phi$.

The reduced complexity phase detector for QPSK is illustrated in Fig. 3. The phase error is computed by comparing the phase difference between the received signal $x(n) + jy(n)$ and the closest constellation point $\hat{I}(n) + j\hat{Q}(n)$ as illustrated in Fig. 3. For this constellation, the four constellation points are located in the center of the four quadrants so that the nearest constellation point corresponding to the received signal is easily computed by taking the sign of the in-phase and quadrature amplitudes. The sine of the phase difference may be expressed as

$$\sin(\Delta\phi(n)) = \frac{\hat{I}(n)Y(n) - \hat{Q}(n)X(n)}{\sqrt{\hat{I}^2(n) + \hat{Q}^2(n)}\sqrt{X^2(n) + Y^2(n)}} \quad (1)$$

which shows that a phase error signal proportional to the sine of the phase difference can be generated by forming the difference of the cross product as illustrated.

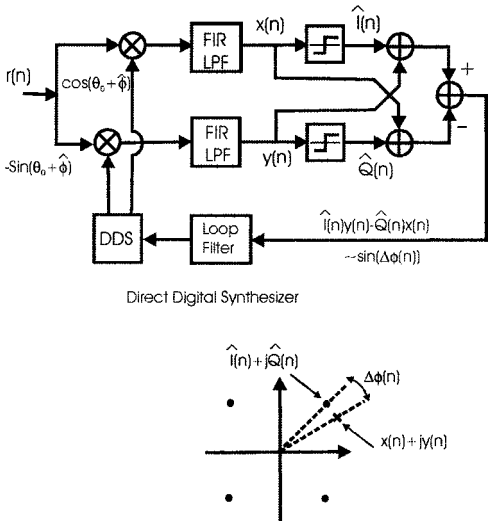

Figure 3. QPSK carrier phase PLL using the cross product phase detector and the constellation. The difference of the cross product is proportional to the sine of the phase error.

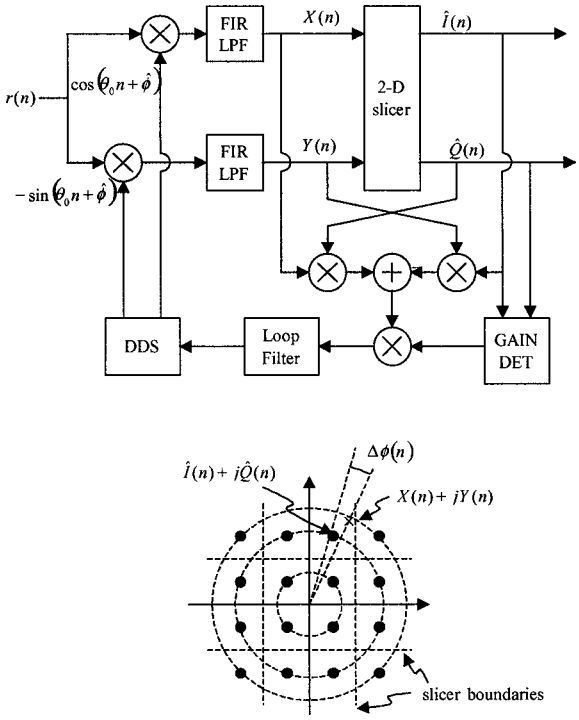

Figure 4. 16-QAM carrier phase PLL with SNR dependent gain and constellation with decision boundaries.

The phase detector for a received signal with a denser modulation constellation, such as 16-QAM, must conduct additional comparisons (other than the sign) to bracket the location of the received two-tuples. This function is incorporated into the phase detector using a 2-dimensional slicer as illustrated in Fig. 4. The phase difference is still given by Eq. (1) so that the cross product difference is still computed. Note that detected points in the 16-QAM constellation reside on one of three contours indicating distance from the origin. The larger radii contours represent signals with higher signal to noise ratio. Phase measurements for data points with larger radii contain less noise and the PLL should take advantage of this side information and rely more heavily on measurements with high SNR. The block labeled "DET GAIN" in the phase detector forms the SNR dependent gain by assigning different scalar gains as a function of the radius of the detected two-tuple.

3. FPGA Implementation

The highly flexible nature of FPGAs means that the implementation space is rich with architectural

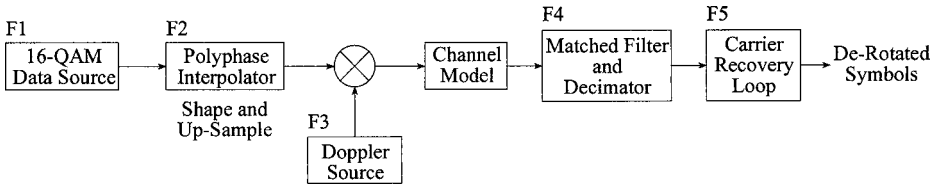

Figure 5. Simulation model used to develop and verify the carrier recovery loop.

alternatives for constructing virtually any signal processing function. A particular solution would be selected based on an optimization process that combines system level performance requirements along with FPGA resource considerations. For example, consider a design that requires a particular arithmetic function, and that the function can be computed using either a look-up table (LUT) based procedure or an alternative approach that is more logic intensive. LUT-based computations align well with the block memory resources in Xilinx Virtex-II [2] FPGAs. However, if the block memories are scheduled to be used by other functions in the system that are implemented in the same physical device, it is preferable to use the logic intensive solution and realize the calculation in the FPGA logic fabric. This type of tradeoff will be demonstrated below in the 16-QAM carrier recovery circuit.

3.1. 16-QAM Carrier Recovery Circuit

For QPSK modulation the simplified phase detector shown in Fig. 3 provides the most compact FPGA implementation. However, for higher-order modulation formats the phase detector is more complex, and, as highlighted earlier, a common choice is to measure the angle between the baseband signal complex envelope and the nearest constellation point using an *atan2*(I,Q) computation. In the context of the FPGA implementation several questions arise: what is the most FPGA-efficient technique for computing the arc-tangent? and, what accuracy is required?

There are many techniques for computing trigonometric functions. The most suitable candidate will depend on the device resource at the focus of the optimization criterion. For example, a memory intensive approach is not going to be a good choice if we are interested in minimizing block memory [2] utilization. Conversely, a computation rich technique would not be suitable if it is desirable to conserve logic fabric resources (logic slices [2]).

Two approaches to the problem will be considered: the first is a lookup table based approach while the second method employs CORDIC [3, 4] arithmetic.

3.2. Memory-Based Phase Detector

Figure 5 shows the system model that was used to develop the FPGA implementation of the carrier recovery loop. A 16-QAM message source $F1$ was shaped (using a root raised cosine filter) and unsampled to 4 samples per symbol using the polyphase interpolator $F2$. The Doppler source $F3$ is used to introduce a carrier offset to the channel data. The signal is then processed by the receiver matched filter and the sample rate reduced to the symbol rate T using a root raised cosine polyphase decimator $F4$. The baud-rate data is then directed to the carrier recovery loop (CRL) $F5$. The signal generation and receiver matched filter were constructed using operators from the Mathworks Simulink® [5] DSP blockset. These functions utilize floating-point arithmetic. The CRL was realized using the Xilinx *System Generator for DSP* [6] environment. System Generator is a visual dataflow design environment that allows the system developer to work at a suitable level of abstraction and use the same computation graph not only for simulation and verification, but also for FPGA hardware implementation. System Generator blocks are bit- and cycle-true behavioral models of FPGA intellectual property components, or library elements. The library based approach results in design cycle compression in addition to generation area efficient high-performance circuits. For the simulation plots shown below the Doppler frequency was 0.001 Hz, the CRL loop bandwidth was $2\pi/1000$ with a damping factor $\xi = \sqrt{2}/2$. Figure 6(a) and (b) show the CRL architecture and loop filter flowgraph respectively.

Figures 7 to 9 show the transmitter modulation transition diagram, constellation, and eye diagrams respectively.

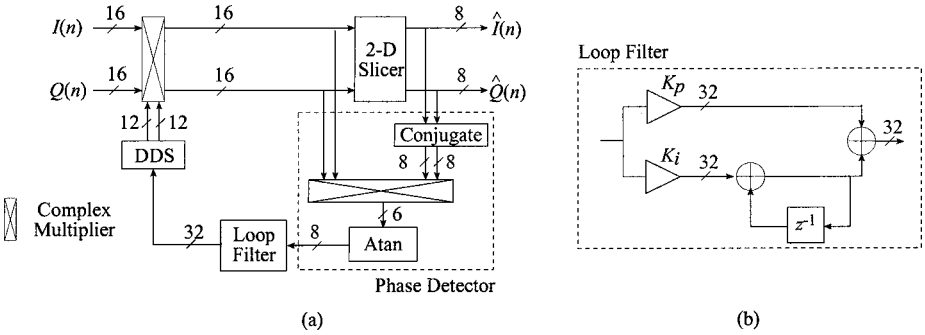

Figure 6. 16-QAM carrier recovery loop. (a) Top-level architecture. (b) Loop filter signal flowgraph.

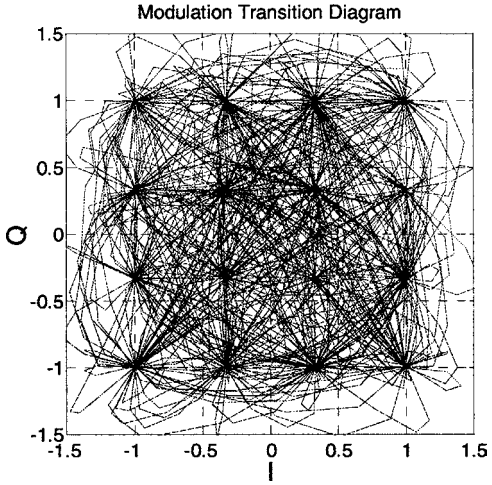

Figure 7. Transmitter modulation transition diagram.

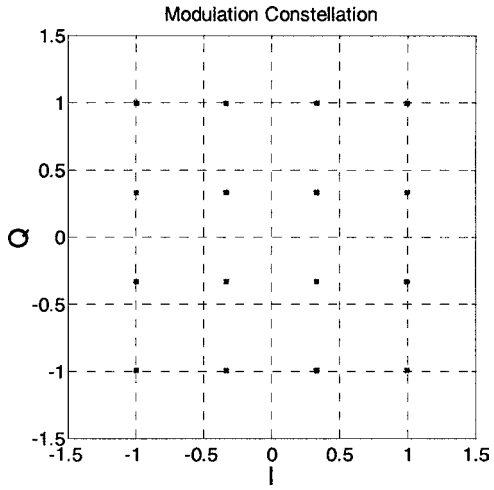

Figure 8. Transmitter modulation constellation.

The Doppler shift introduced during transmission causes the constellation to rotate as shown in Fig. 10. In an actual system this frequency offset could be the result of relative motion between the transmitter and receiver, as would be the case with a mobile terminal traveling away or towards a receiver. It could also be attributed to small frequency variations of the various synthesizers in both the transmitter and receiver, which are functions of time and temperature. The eye diagram at the receiver is shown in Fig. 11. Observe that the receiver cannot make valid decisions until the signal is de-rotated.

The System Generator implementation of the CRL is shown in Fig. 12. This description produces a highly optimized FPGA realization since each token in the model maps to an FPGA library component that has been carefully constructed for the FPGA target device. The visual representation of the system not only serves

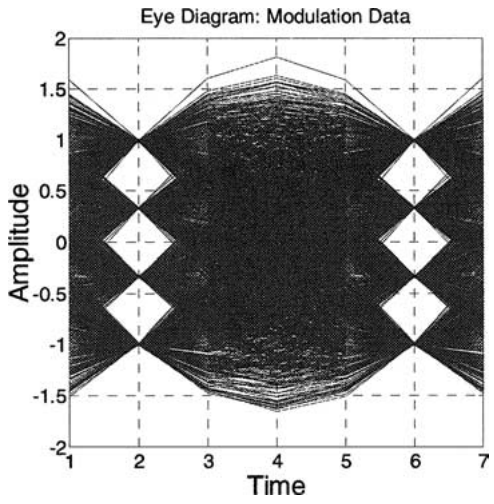

Figure 9. Eye diagram at transmitter.

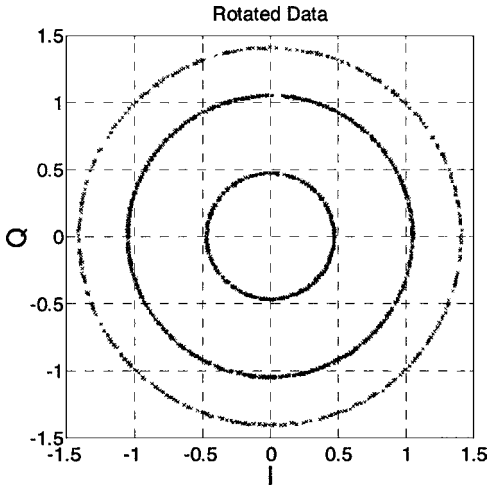

Figure 10. Rotating constellation presented to the carrier recovery loop.

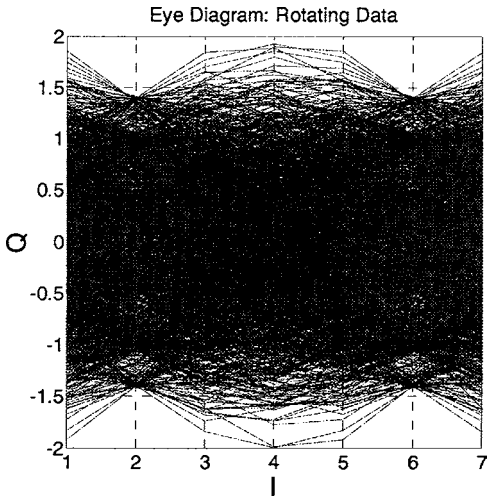

Figure 11. Receiver eye-diagram before carrier acquisition.

as the design specification, but as the behavioral simulation model and the source definition for the hardware. This design flow also facilitates the rapid investigation of various quantized versions of the system. The precision definition fields of the elements in the dataflow graph reference variables in the Matalb workspace [7]. This data is initialized with a Matlab m-file that is executed before the System Generator simulation is started. This script is easily modified to produce a new hardware profile and simulation results.

Several functional units are required to implement the carrier recovery loop. The complex heterodyne $G1$ required to down-convert the input signal is a complex multiplier. The direct digital synthesizer (DDS) $G2$ requires an integrator and a sine/cosine look-up table. The second-order loop filter $G3$ is realized using two multipliers, an integrator and an adder. The phase detector $G4$ comprises a symbol de-mapper, conjugator and a complex multiplier to form the angle between the signal envelope and the symbol decision. The conjugator can of course be incorporated into the complex multiplier with a small modification to the multiplier circuit. The FPGA target technology for the design was Virtex-II [2]. The Virtex-II embedded multipliers were used in the mixer, phase detector and the loop filter. To minimize the number of multipliers required the mixer was constructed using the 3 multiply-5-add algorithm [8].

The phase detector is shown in Fig. 13 and consists of a symbol de-mapper, complex multiplier and a look-up table for computing the arctangent of the received constellation's phase angle.

The symbol de-mapper maps the correctly timed input sample to the nearest (in the Euclidian sense) constellation point. This circuit is shown in Fig. 14 and consists of some simple relational operators, multiplexing

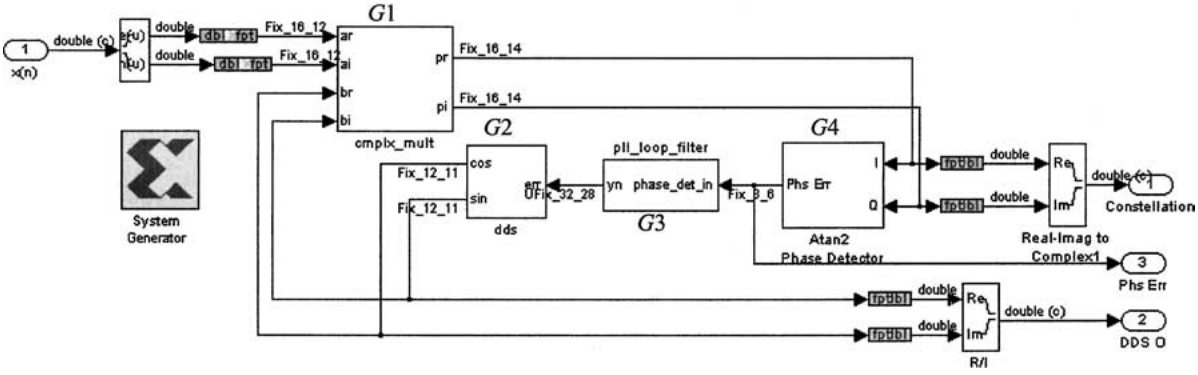

Figure 12. System Generator model of the 16-QAM CRL.

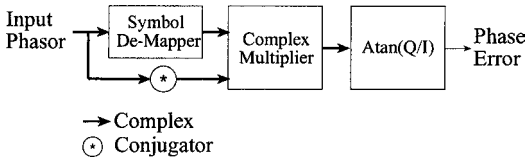

Figure 13. Phase detector consisting of symbol de-mapper, conjugator, complex multiplier and arc-tangent look-up table.

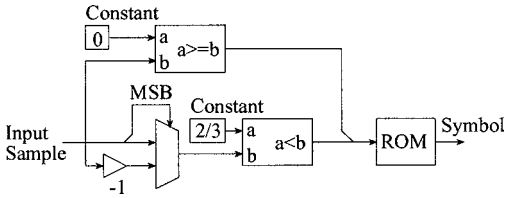

Figure 14. 16-QAM symbol de-mapper.

and a small 4-entry look-up table that stores the constellation coordinates, in this case the vector $[1/31\ -1/3\ -1]$. The compact table is best realized in distributed memory [2]. There are of course 2 de-mapping circuits, one for each of the in-phase and quadrature signals.

The first implementation of the arctangent processor is based on a look-up table approach. The look-up table contains samples of the function *atan2*(I,Q). The table consists of multiple 100's of samples and is therefore best realized using the FPGA on-chip block memory (SelectRAM [2]). To conserve these resources it is desirable to employ as compact a look-up table as possible. The table depth and width will ideally both be small. The averaging process of the proportional-integral (PI) loop filter means that the arctan processor can in fact be heavily quantized. The real and imaginary components of the phase angle were each quantized to 6-bits of precision, while the samples of the *atan2*(I,Q) function where represented using 8 bits. The resulting 4096-deep-by-8-bit memory consumed 2 Virtex-II block memories.

The carrier recovery loop filter is shown in Fig. 6(b) and consists of two multipliers, one for each of the proportional K_p and integral K_i coefficients, an integrator and adder that consume a total of 32 slices. While area efficient constant coefficient multipliers could be used for these products, using the embedded multipliers offers several advantages. For example, the phase-error signal could be monitored by a lock detector, and the loop bandwidth and damping factors be dynamically adjusted to open or close the pull-in range of the DPLL. This servo control mechanism would widen the loop filter bandwidth when the carrier is lost and reduce the bandwidth when the variance of the error signal drops below a certain threshold. In providing this dynamic we are balancing the conflicting requirements for a wide-band loop to support a large pull-in range, yet on the other hand preferring a narrowband loop to minimize the amount of noise introduced into the CRL, and hence potentially improve the system bit error rate. This adjustment process is relatively low-bandwidth and does not necessarily have to be implemented in the logic fabric, and is probably more reasonably handled using an instruction set processor (ISP). One interesting design possibility is enabled with the Virtex-II Pro FPGA family. These devices augment the Virtex-II architecture with one or more Power-PC 405 (PPC405) processors. An intelligent CRL function could be realized by performing a hardware-software partitioning of the problem and realizing the CRL itself in the logic fabric and implementing the lock detector, coefficient update process (of K_i and K_p) and CRL adjustment using embedded software running on the PPC405.

The final component in the circuit is the direct digital synthesizer (DDS) shown in Fig. 15.

A simple phase truncation architecture [9] using a 32-bit phase accumulator, 1024-deep sine/cosine look-up table and 12-bit precision samples was used in the design.

The final de-rotated constellation is shown in Fig. 16. For comparison, the de-rotated constellation for a double-precision floating-point implementation of the loop is shown in Fig. 17. The fixed-point data clearly shows the loop has acquired and locked, and that the correct decisions can be made from the CRL output samples. The constellation for the fixed-point arithmetic implementation exhibits artifacts that are characteristic of phase noise.

The phase noise can be reduced by increasing the precision of the *atan2* computation. The most dramatic improvement is achieved by allocating more precision to the constellation angle coordinates. This of course requires a larger look-up table. If one additional bit is used for each of the I and Q samples the table size increased

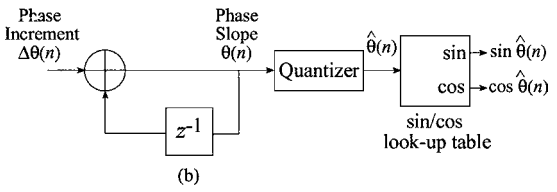

Figure 15. Direct digital synthesizer in CRL.

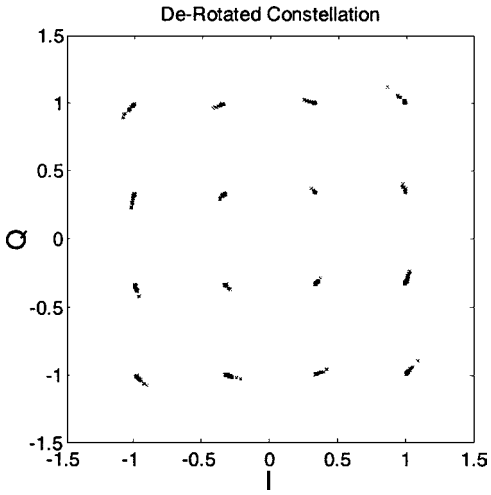

Figure 16. De-rotated constellation—fixed-point arithmetic model with 4096 × 8-bit precision *atan2* look-up table.

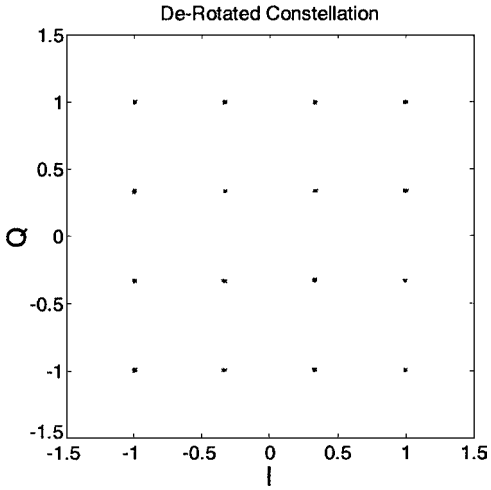

Figure 17. De-rotated constellation—double-precision floating-point arithmetic simulation.

four-fold to 16,384 entries. The recovered constellation using this enhancement is shown in Fig. 18.

By comparing Figs. 16 and 18 we can clearly see the reduction in phase noise reflected in the tighter clustering of the received symbols in the constellation. This improved performance of course increases the block memory utilization from two memory primitives to eight. The FPGA resource utlilization for each component in the CRL is provided in Table 1.

Another system level figure-of-merit is the loop acquisition time. This is the number of symbols required to acquire the carrier and de-rotate the constellation. Figure 19 shows the phase error time-series from

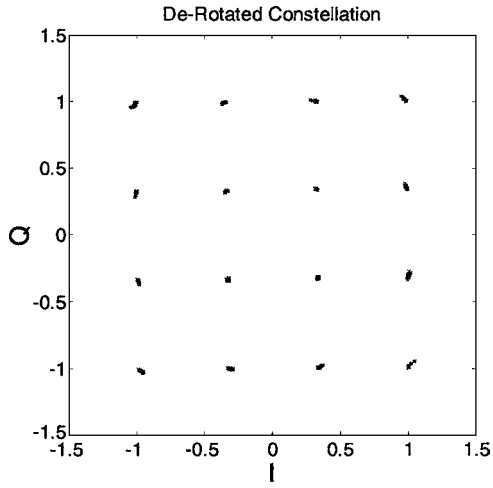

Figure 18. De-rotated constellation—fixed-point arithmetic model with 16384 × 8-bit precision *atan2* look-up table.

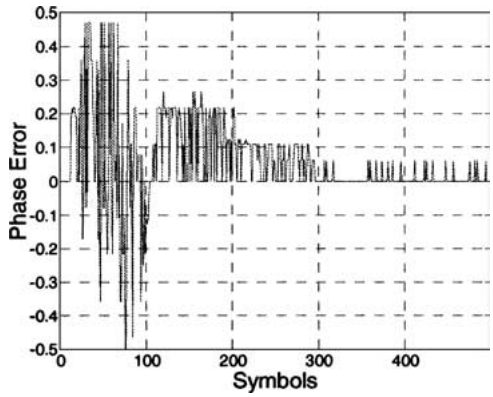

Figure 19. Phase error, look-up table (4,096-by-8b LUT) based phase detector, 0.001 HZ Doppler, 4 samples/symbol. The carrier is acquired in approximately 120 symbols.

the output of the phase detector for the fixed-point arithmetic CRL using a 4,096-by-8b precision look-up table. The carrier is acquired in approximately 120 symbols.

For comparison, the acquisition time for a floating-point arithmetic, or idealized system, is presented in Fig. 20. The acquisition time is approximately 20 symbols.

Quantizing the design both increases the RMS error-vector-magnitude and increases the acquisition time. Allocating more bits of precision to represent the real and imaginary components of the constellation phase angle will reduce the acquisition time. This of course means increasing the depth of the arctangent look-up table. Figures 21 to 23 illustrate the effect of providing

Table 1. FPGA resource utilization for the CRL. Memory-based phase detector.

Function	Slice count	Block RAMs	Embedded multipliers
Heterodyne	111	–	3
DDS	5	1	–
Loop filter	32	–	2
Phase detector	106†	2	3
Symbol De-map (I&Q)	57	–	–
Total	254	3	8

†Includes slice count for complex symbol de-mapper.

one additional bit to the phase angle quantizer. Holding the Doppler frequency constant at 0.001 Hz, Fig. 21 shows the time series for the Doppler in-phase and quadrature signals along with the in-phase and quadrature components of the carrier loop DDS. The time axis in this plot is supplied in the context of the high (4 samples/symbol) sample rate at the receiver matched filter. By inspection, we observe that carrier lock is achieved in approximately 120 symbols. The phase angle quantizer uses 6-bits of precision for each of the I and Q components, for a total of 12-bits. The arctan calculation is therefore based on a 4,096-by-8-bit look-up table.

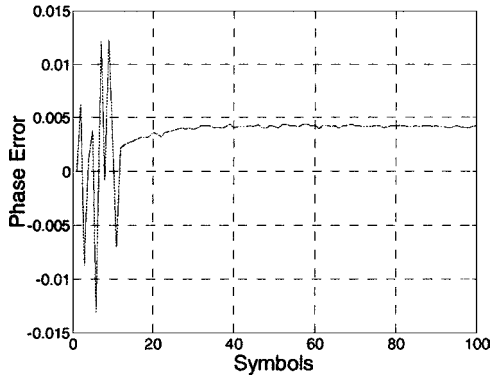

Figure 20. Phase error, full precision floating-point arithmetic model, 0.001 HZ Doppler, 4 samples/symbol. The carrier is acquired in approximately 20 symbols.

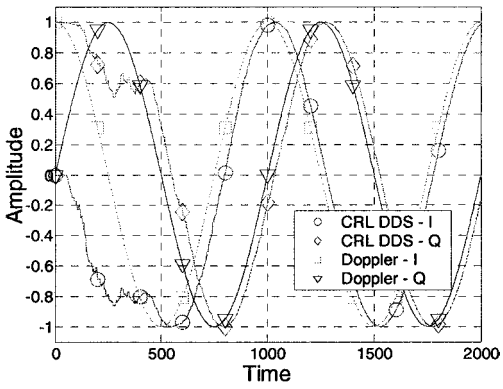

Figure 21. Doppler and CRL DDS time-series. The phase angle error is quantized to a total of 12-bits and the phase detector employs a 4,096-by-8 look-up table.

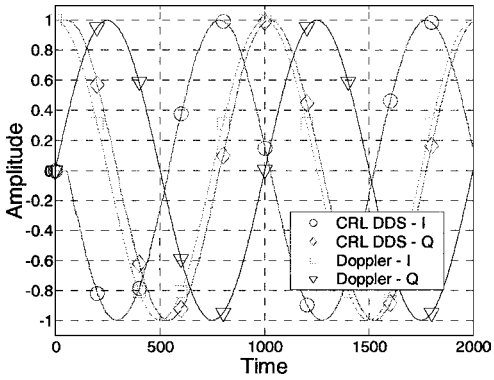

Figure 22. Doppler and CRL DDS time-series. The phase angle error is quantized to a total of 14-bits and the phase detector employs a 16,384-by-8 look-up table.

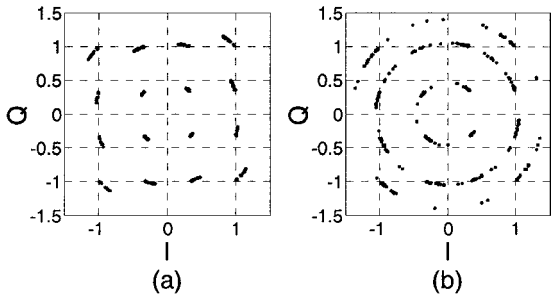

Figure 23. Carrier loop constellations (symbols 50 through 400) for (a) 16,384-deep arctan look-up table and (b) 4,096 deep arctan look-up table.

Figure 22 shows the result when the phase angle quantizer precision is increased to 7-bits for each of the I and Q components. The total number of bits is now 14, and the look-up table profile is 16,384-by-8 bits.

By applying a finer quantization to the phase angle error the acquisition time has been reduced from 120 symbols to approximately 50. Figure 23 shows the de-rotated constellation plots for the two scenarios. Symbols 50 through 400 are plotted. Figure 23(a) corresponds to the 16,384 deep look-up table while Fig. 23(b) is for a table depth of 4,096 samples. There

is still some residual phase offset in Fig. 23(a), however it is significantly reduced in comparison to Fig. 23(b) that uses the smaller table.

The reduction in acquisition time is of course at the expense of the number of FPGA on-chip memory, and in this case requires an increase from 2 to 8 block SelectRAMs.

Holding the look-up table depth constant at 4,096 samples and increasing the precision of the table samples provided no decrease in acquisition time.

Ideally the CRL would have a single delay element in the feedback path, and this would typically reside in the DDS phase accumulator. The block memories in Virtex-II FPGAs are synchronous and so have an inherent 1 cycle access latency. Since these components are used in the phase detector and the DDS two additional delays are introduced in the circuit. This would seem to increase the total loop delay to 3 sample periods. A minor re-structuring of DDS circuit, shown in Fig. 24, to relocate the DDS accumulator register reduces the net delay to two samples.

The net delay is still one more than the ideal case. However, since the loop bandwidth is typically a small fraction of the symbol rate, the effect is minor, and at a system level is reflected by a small reduction in phase margin.

The delays introduced by the memories actually serve to increase the symbol rate of the design by minimizing the critical path in the circuit. The design was implemented using version 4.1.03 of the Xilinx tool suite. The maximum number of logic levels was determined to be 42, and with the place-and-route (PAR) effort level set to 4 (par −ol 4), the maximum clock frequency for the design is 22.8 MHz in a −6 speed grade device. For 16-QAM, with 4-bits carried by each symbol, the loop can support a data throughput of 91.2 Mbps. Since the critical path is so long for this design, moving to a slower (and more economic) −4 speed grade FPGA does not actually impact performance by much. The maximum clock frequency reduces slightly to 22.4 MHz which corresponds to 89.7 Mbps.

3.3. CORDIC Based Phase Detector

The previous section described a look-up table based phase detector. However, there are many options for computing arctangents. One alternative that is well suited to FPGA implementation is *coordinate rotation digital computer* (CORDIC) arithmetic [3, 4]. The CORDIC algorithm is an iterative procedure that can be used to compute a diverse range of mathematical functions. CORDIC arctangents are computed using the *vectoring* mode [3, 4] equations defined by Eq. (2).

$$\begin{aligned} i &= 0 \\ z_i &= 0 \\ a_i &= -1 \quad \text{if } z_i < 0 \text{ otherwise } a_i = +1 \\ x_{i+1} &= x_i + a_i y_i 2^{-i} \\ y_{i+1} &= y_i - a_i x_i 2^{-i} \\ z_{i+1} &= z_i + a_i \tan^{-1}(2^{-i}) \\ i &= i + 1 \end{aligned} \tag{2}$$

The iteration count i is initialized to 0 along with the angle register z. Registers x and y are respectively initialized with the in-phase and quadrature components of the constellation angle that is generated by the complex multiplier in the phase detector.

Each iteration contributes one additional bit of precision to the final result. The conditional test in the algorithm serves to minimize the value of y at each time-step. When the required number of iterations have been completed the angle register z contains an approximation to $atan2(x, y)$.

The CORDIC algorithm is highly suitable for FPGA mechanization because it is dominated by additions and subtractions. These functions are very efficiently implemented by FPGA technology and require only $N/2$ logic slices for an N-bit adder/subtractor.

The CORDIC algorithm does not converge for input angles $|\theta| > \pi/2$. In order to support the full range of input angles the computation is decomposed into three

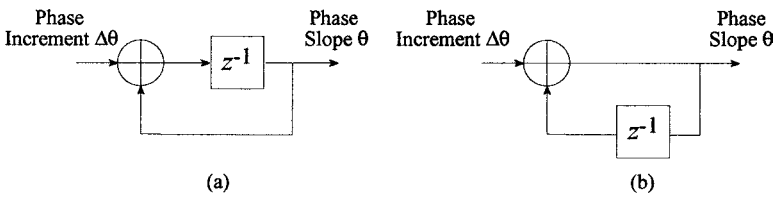

Figure 24. A minor graph modification to the DDS reduces the CRL loop delay to two sample periods.

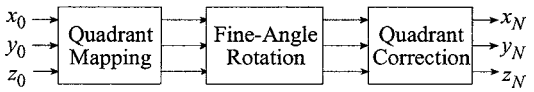

Figure 25. CORDIC processor used for computing artan2(I, Q) in the phase detector.

stages. First, a course angle rotation is performed to map the input argument into quadrant 1, next N micro rotations (using the CORDIC algorithm) are performed, and finally a quadrant correction is applied to account for the coarse angle rotation. These steps are clearly seen from left-to-right in Fig. 25. The quadrant mapping is straightforward and consists of a comparator, negator and a multiplexor. There are several implementation options for realizing the CORDIC processing elements. A sequential approach in which all of the iterations are folded onto a single computation unit is one possibility, while a fully parallel, or unfolded, design is another option. The CORDIC processing elements require 3 adder/subtractors and a barrel shifter. The arctan(2^{-i}) is computed off-line and for the sequential algorithm the N terms are stored in a small look-up table. For the parallel architecture these values simply reduce to a constant connected to one operand input of an adder/subtractor. Adders are implemented extremely efficiently in FPGAs. However, the barrel shifter can consume a reasonable amount of logic. For an unfolded design the barrel shifter is not required and the shift operations simply reduce to wiring between successive CORDIC elements. The parallel architecture was employed for the CRL. Figure 26 shows the CORDIC micro-rotation processing array. A detailed view of a single processing element (PE) is shown in Fig. 27.

Simulations were performed to determine the number of fine-angle iterations required as well as the precision of the CORDIC elements. The performance of the algorithm is quantified by computing the RMS error vector magnitude (EVM) of the de-rotated constellation. Figures 28 to 33 provide the results for several

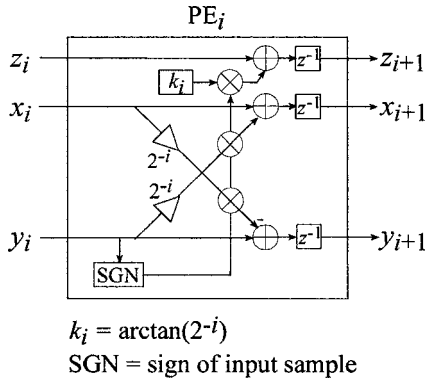

Figure 27. CORDIC micro-rotation processing element for performing vector rotation i.

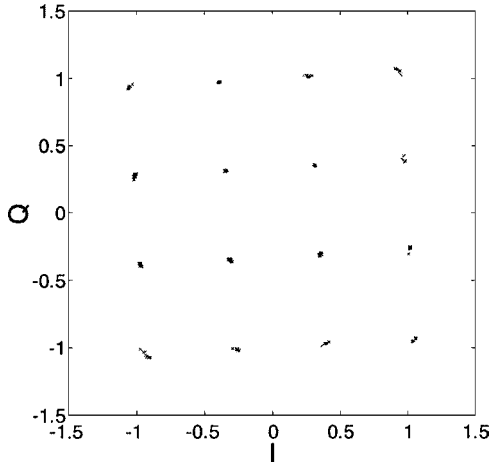

Figure 28. De-rotated constellation. The CRL uses a 4-iteration 16.12 precision CORDIC processor in the phase detector.

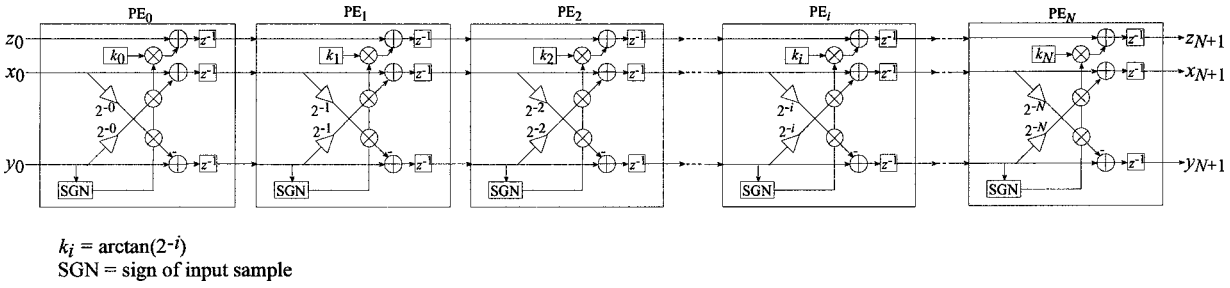

Figure 26. Unfolded and pipelined CORDIC processor for computing the fine angle rotations.

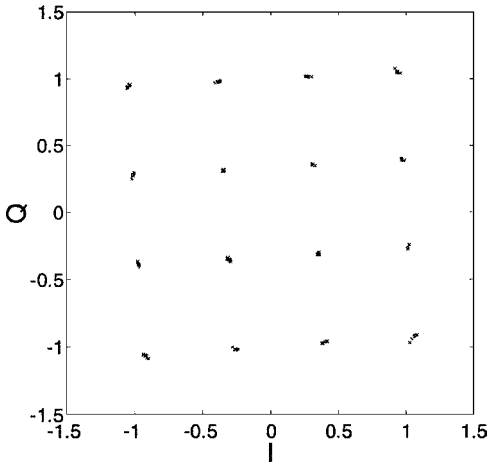

Figure 29. De-rotated constellation. The CRL uses a 4-iteration 14.10 precision CORDIC processor in the phase detector.

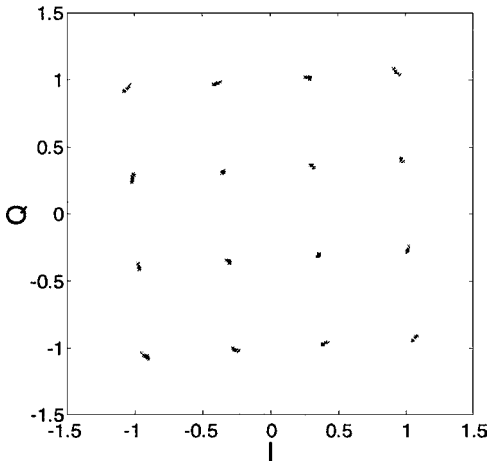

Figure 30. De-rotated constellation. The CRL uses a 4-iteration 12.8 precision CORDIC processor in the phase detector.

simulation runs. In the figure captions the notation $m.n$ designates the CORDIC processing elements as m-bit precision units with (m-n) integer bits and n fractional bits. Figures 28 to 30 all employ 4 iterations with a total processing precision of 16, 14 and 12 bits respectively. The effect of reducing the number of bits is clearly observed by the increased spreading of the symbol clusters as the number of bits is reduced. Also note the residual phase tilt that remains after the carrier has been acquired.

Figures 31 to 33 are the recovered constellation plots for 5 iterations using 16, 14 and 12-bit arithmetic respectively. The 16-bit processor (Fig. 31) produces very tightly clustered symbols and has an EVM

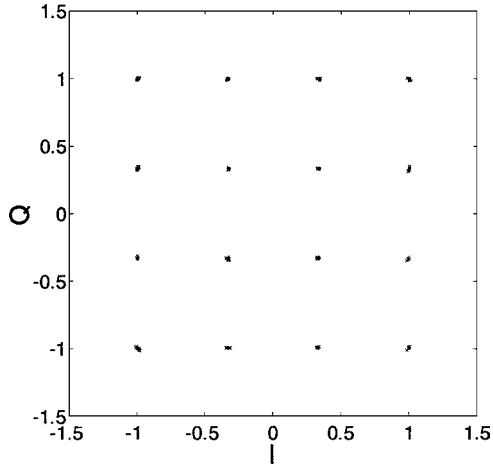

Figure 31. De-rotated constellation. The CRL uses a 5-iteration 16.12 precision CORDIC processor in the phase detector.

comparable to the floating-point simulation shown in Fig. 17. As the precision of the CORDIC algorithm is reduced the symbols disperse around the true symbol coordinates.

The CRL was implemented using a phase detector constructed with a 5 iteration 16-bit precision CORDIC processor. The FPGA resource utilization is shown in Table 2.

It is instructive to examine the acquisition time of the CORDIC-based carrier loop shown in Fig. 34. The figure shows that the carrier is acquired in approximately 20 symbols. This is approximately the same duration as for the idealized (floating-point arithmetic) system demonstrated in Fig. 20 and 6 times faster than the memory based phase detector using a 4,096 deep look-up table.

Table 2. FPGA resource utilization for the CRL. CORDIC-based phase detector.

Function	Slice count	Block RAMs	Embedded multipliers
Heterodyne	111	–	3
DDS	5	1	–
Loop filter	32	–	2
Phase detector	270	–	3
Total	413†	1	8

†The small slice count discrepancy is due to logic optimizations that occur when the individual CRL components are integrated into the complete system.

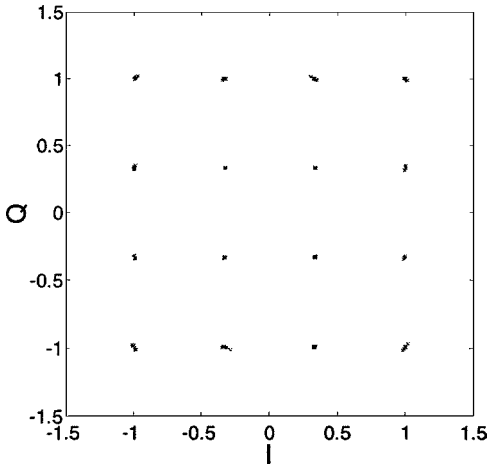

Figure 32. De-rotated constellation. The CRL uses a 5-iteration 14.10 precision CORDIC processor in the phase detector.

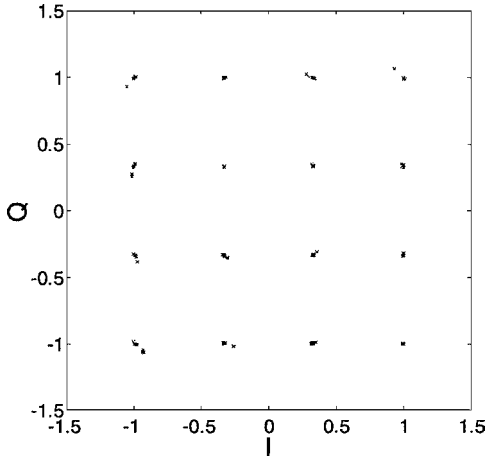

Figure 33. De-rotated constellation. The CRL uses a 5-iteration 12.8 precision CORDIC processor in the phase detector.

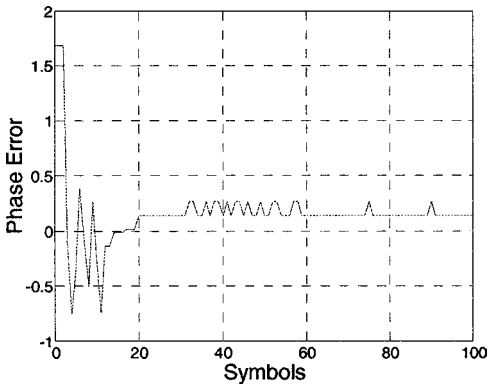

Figure 34. Phase error, CORDIC based phase detector, 0.001 HZ Doppler, 4 samples/symbol. The carrier is acquired in approximately 20 symbols.

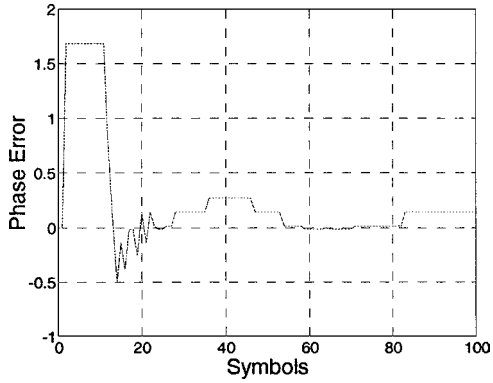

Figure 35. Phase error, CORDIC based phase detector, 0.001 HZ Doppler, 4 samples/symbol. Four levels of pipelining are used in the CORDIC processor in order to reduce the critical path.

If the total number of delays in the carrier loop is to be minimized so as not to degrade the system phase margin, the CORDIC engine must be combinatorial. This can produce a large critical path. After running the design through the FPGA implementation tools the longest path was found to consist of 91 logic levels and support a maximum clock frequency of 19.5 MHz in a -6 (fastest) speed-grade device. Inserting additional delay in the loop through pipelining will shorten the critical path at the expense of phase margin. For example, 4 additional pipelining stages reduces the critical path from 91 to 39, and increases the clock frequency to 25 MHz (place-and-route effort level $=5$). In the slowest speed-grade device (-4) the maximum clock frequency is 18.387 MHz.

As shown in Fig. 35, the more heavily pipelined CORDIC processor will lead to an increase in acquisition time.

4. Conclusion

The continuing evolution of communication standards and competitive pressure in the market-place dictate that communication system architects must start the engineering design and development cycle while standards are still in a fluid state. Third and future generation communication infrastructure must support multiple modulation formats and air interface standards. FPGAs provide the flexibility to achieve this goal, while simultaneously providing high levels of performance. The SDR implementation of traditionally analog and digital hardware functions opens-up new levels of service quality, channel access flexibility and cost efficiency.

We have reviewed several aspects of carrier recovery for communication systems using QPSK and QAM modulation. This topic is of interest in the context of soft radios because bandwidth efficient modulation schemes, like M-ary QAM, are important radio personalities for configurable communication platforms. The keystone of the FPGA is flexibility. This attribute enables a large number of implementation alternatives for a given datapath. We explored two realizations of a carrier recovery loop for 16-QAM. The examples illustrated how various tradeoffs can be made to, for example, conserve on-chip memory resources or logic fabric. The look-up table based carrier recovery circuit occupied 62% of the logic area of the CORDIC based implementation but used 3 times the number of block memories. The architecture can also impact system level performance. For example, feedback networks, like carrier recovery loops, must take computation latency into account. FPGA block SelectRAM is synchronous memory and will always introduce a one cycle delay. Employing these components in a DPLL will impact (decrease) the system phase margin. A CORDIC based phase detector can be constructed that is completely combinatorial, however the critical path in the CORDIC CRL is longer than that of the look-up table based CRL, and so the maximum symbol rate that can be supported will be lower.

As signal processing systems increase in complexity, software and intellectual property development become harder than the implementation of the target hardware. The carrier recovery loops described were implemented using the System Generator [6] system level design tool. This approach provided a rapid development cycle, while providing a useful environment to explore and quantify the performance of different quantized versions of the system. The benefit of this approach from a hardware development perspective, is that the simulation model is also the FPGA source specification.

References

1. H. Meyr, M. Moeneclaey, and S.A. Fechtal, *Digital Communication Receivers*. New York: John Wiley & Sons Inc., 1998.
2. Xilinx Inc., *Virtex-II Handbook*, http://www.xilinx.com/products/virtex/handbook/index.htm Xilinx Inc., *Virtex-II Pro™ Platform FPGA Handbook* http://www.xilinx.com/publications/products/v2pro/handbook/index.htm
3. J.E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Electronic Computers*, vol. 8, no. 3, 1959, pp. 330–334.
4. Y. H. Hu, "CORDIC-Based VLSI Architectures for Digital Signal Processing," *IEEE Signal Processing Magazine*, July 1992, pp. 16–35.
5. The Mathworks Inc., Simulink, *Dynamic System Simulation for Matlab, Using Simulink*. Natick, Massachusetts, USA. 1999.
6. http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=system_generator
7. The Mathworks Inc., Matlab, *Getting Started with Matlab*. Natick, Massachusetts, USA, 1999.
8. R.R. Blahut, *Fast Algorithms for Signal Processing*. MA: Addison-Wesley, 1985.
9. C.H. Dick and F.J. Harris, "Direct Digital Synthesis—Some Options for FPGA Implementation," *SPIE International Symposium on Voice Video and Data Communication: Reconfigurable Technology: FPGAs for Computing and Applications Stream*. Boston, MA, USA, Sept. 20–21 1999, pp. 2–10.
10. F.J. Harris and C.H. Dick, "On Structure and Implementation of Algorithms for Carrier and Symbol Synchronization in Software Defined Radios," EUSIPCO-2000, "Efficient Algorithms for Hardware Implementation of DSP Systems," Tempere, Finland, 5–8 Sept. 2000.

Chris Dick is the Director of Signal Processing Systems Engineering and the DSP Chief Architect at Xilinx. Chris has worked with signal processing technology for two decades and his work has spanned, the commercial, military and academic sectors.

Chris' work and research interests are in the areas of fast algorithms for signal processing, digital communication, software defined radios, adaptive signal processing, synchronization, hardware architectures for real-time signal processing, parallel computing, interconnection networks for parallel processors, and the use of Field Programmable Arrays (FPGAs) for custom computing machines and signal processing. Chris has published many papers in the fields of signal processing, parallel computing, Inverse Synthetic Array Radar (ISAR) imaging, and the use of FPGAs for building computing platforms for signal processing applications. Chris has over 70 journal and conference publications and has been an invited speaker at many international DSP and communications symposiums.

In addition to his role at Xilinx Chris teaches FPGA Signal processing classes for the University of California Berkeley Extension Program and Santa Clara University.

Chris is active in the Software Defined Radio area and is on the Software Defined Radio Forum's Board of Directors.

He holds a bachelor's and PhD degrees in the areas of computer science and electronic engineering.
chris.dick@xilinx.com

fredric j harris holds the CUBIC Signal Processing Chair of the Communication Systems and Signal Processing Institute at San Diego State University where since 1967 he has taught courses in Digital Signal Processing and Communication Systems. He has extensive practical experience in communication systems, high performance modems, sonar and advanced radar systems and high performance laboratory instrumentation. He holds a number of patents on digital receiver and DSP technology and lectures throughout the world on DSP applications. He consults for organizations requiring high performance, cost effective DSP solutions.

He is the author of the book Multirate Signal Processing for Communication Systems: Current Practice and Next Generation Techniques and has contributed to a number of other books on DSP. In 1990 and 1991 he was the Technical and then the General Chair of the Asilomar Conference on Signals, Systems, and Computers and was Technical Chair of the 2003 Software Defined Radio Conference. He became a Fellow of the IEEE in 2003, cited for contributions of DSP to communications systems. His education includes a Bachelors Degree in EE from the Polytechnic Institute of Brooklyn (1961), a Masters Degree in EE from San Diego State University (1967) and Ph.D. work at the University of California, San Diego (1968–1973).

He is the traditional absent-minded professor and drives secretaries and editors to distraction by requesting lower case letters when spelling his name. He roams the world collecting old toys and slide-rules and riding old railways.

Michael Rice (M'82 SM'98) received a BSEE from Louisiana Tech University in 1987 and his PhD from Georgia Tech in 1991. Dr. Rice was with Digital Transmission Systems, Inc. in Atlanta and joined the faculty at Brigham Young University in 1991 where he is currently the Jim Abrams and Anita Schiller Professor in the Department of Electrical & Computer Engineering. Professor Rice was a NASA/ASEE Summer Faculty Fellow at the Jet Propulsion Laboratory during 1994 and 1995 where he worked on land mobile satellite systems. During the 1999-2000 academic year, Professor Rice was a visiting scholar at the Communication Systems and Signal Processing Institute at San Diego State University.

Professor Rice's research interests are in the area of digital communication theory and error control coding with a special interest in applications to telemetering and software radio design. He has been a consultant to both government and industry on telemetry related issues. He is a member of the IEEE Communications Society. He was Chair of the Utah Section of IEEE from 1997 to 1999 and is currently chair of the Signal Processing & Communications Society Chapter of the Utah Section.
mdr@ee.byu.edu