

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**DESIGN, IMPLEMENTATION, AND TESTING OF A VLSI
HIGH PERFORMANCE ASIC FOR EXTRACTING THE
PHASE OF A COMPLEX SIGNAL**

by

Ronald Christopher Altmeyer

September 2002

Thesis Advisor:

Douglas J. Fouts

Co-Advisor:

Phillip E. Pace

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2002	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: Design, Implementation, and Testing of a VLSI High Performance ASIC For Extracting the Phase of a Complex Signal			5. FUNDING NUMBERS	
6. AUTHOR(S) R. Chris Altmeyer				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Joint Services Electronic Warfare Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Code 313 Arlington, VA			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) This thesis documents the research, circuit design, and simulation testing of a VLSI ASIC which extracts phase angle information from a complex sampled signal using the arctangent relationship: $\phi = \tan^{-1}(Q/I)$. Specifically, the circuit will convert the In-Phase and Quadrature terms into their corresponding phase angle. The design specifications were to implement the design in CMOS technology with a minimum transistor count and ability to operate at a clock frequency of 700 MHz. Research on the arctangent function was performed to determine mathematical calculation methods and the CORDIC method was chosen to achieve the stated design specifications. MATLAB simulations were used to calculate and verify accuracy and to implement Quine-McClusky logic minimization. T-SPICE netlists were generated and simulations were run to determine transistor and circuit electrical operation and timing. Finally, overall circuit logic functionality of all possible input combinations was completed using a VHDL simulation program.				
14. SUBJECT TERMS Digital Image Synthesizer, DIS, VLSI, ASIC, CMOS, Inverse Tangent, arctangent, arc tangent, atan, atan2, Coordinate Rotation Digital Computer, CORDIC, Amplitude to Phase Conversion, In-phase, Quadrature, Quine-McClusky, VHDL, Tanner, MOSIS.			15. NUMBER OF PAGES 130	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**DESIGN, IMPLEMENTATION, AND TESTING OF AN ASIC VLSI HIGH
PERFORMANCE ARCTANGENT FUNCTION**

Ronald Christopher Altmeyer
Captain, Canadian Army
B.Sc., Royal Roads Military College, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
September 2002**

Author: R. Chris Altmeyer

Approved by: Douglas J. Fouts,
Thesis Advisor

Phillip E. Pace,
Co-Advisor

John P. Powers,
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis documents the research, circuit design, and simulation testing of a VLSI ASIC which extracts phase angle information from a complex sampled signal using the arctangent relationship: $\phi = \tan^{-1}(Q/I)$. Specifically, the circuit will convert the In-Phase and Quadrature terms into their corresponding phase angle. The design specifications were to implement the design in CMOS technology with a minimum transistor count and ability to operate at a clock frequency of 700 MHz. Research on the arctangent function was performed to determine mathematical calculation methods and the CORDIC method was chosen to achieve the stated design specifications. MATLAB simulations were used to calculate and verify accuracy and to implement Quine-McClusky logic minimization. T-SPICE netlists were generated and simulations were run to determine transistor and circuit electrical operation and timing. Finally, overall circuit logic functionality of all possible input combinations was completed using a VHDL simulation program.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND OF THE DIGITAL IMAGE SYNTHESIZER	1
B.	PRINCIPAL CONTRIBUTIONS	5
C.	ORGANIZATION OF THESIS	7
II.	INVESTIGATION OF CONVERSION METHODS.....	9
A.	TASKS	9
1.	Aim	9
2.	Implied Tasks	9
B.	GENERAL.....	9
1.	The Arctangent Function of a Real Number	9
2.	The Arctangent Function of a Complex Number	11
3.	Five-Bit Phase Quantization	12
C.	AMPLITUDE-TO-PHASE CONVERSION METHODS.....	13
1.	Calculus.....	15
2.	Polynomial Approximation	15
3.	Look Up Table.....	16
4.	Sum-of-Products Logic Block	20
5.	CORDIC	21
D.	CORDIC HARDWARE IMPEMENTATION.....	29
III.	SCHEMATIC DESIGN OF THE Q/I PHASE CONVERTER.....	33
A.	HIERARCHICAL SCHEMATIC DESIGN OVERVIEW	33
B.	TRANSISTORS	33
1.	N-FET.....	33
2.	P-FET	34
3.	N/P-FET Current and Voltage – Drain to Source	35
C.	LOGIC GATES.....	37
1.	Inverter	37
2.	Pass Gates	43
3.	Buffer	44
4.	XOR.....	45
5.	NAND2	46
D.	SECONDARY SUB-CIRCUITS	48
1.	Registers.....	48
2.	RCA_PSVCHAIN	50
3.	RCA_13927Buffer.....	51
4.	16-Bit Pipelined Adder/Subtractor	52
E.	NUMBER FORMATS.....	55
F.	CORDIC ROTATION LEVEL.....	57
1.	Overview	57
2.	Rotation Circuit Diagram	57
3.	Constant Phase Loading.....	59
4.	Negative 128 Fix Circuit.....	60

5.	Two's Complement Circuit.....	60
G.	CORDIC GENERAL VECTOR LEVEL.....	61
1.	Overview	61
2.	Schematic	62
3.	Example: CORDIC Vector Level 14.03.....	64
H.	9-BIT TO 5-BIT NUMBER CONVERSION	67
1.	Overview	67
2.	Schematics	72
I.	COMPLETED CIRCUIT	75
1.	Completed Circuit.....	75
2.	Circuit Verification and Parameters.....	77
3.	Verification	79
IV.	CONCLUSIONS AND RECOMMENDATIONS.....	83
A.	GENERAL.....	83
B.	LESSONS LEARNED.....	83
C.	RECOMMENDATIONS FOR FUTURE WORK.....	84
APPENDIX A.	MATLAB CODE	85
A.	OVERVIEW.....	85
B.	QUINE-MCCLUSKY MINIMIZATION.....	85
1.	File – Main.m.....	85
C.	OTHER CODES	86
1.	File – NoiseMargins.m.....	86
2.	File – Arctangent.m	86
3.	File – Polynomial_Approx.m	88
APPENDIX B.	MOSIS TSMC 0.18 MICRON FET PARAMETERS [14].....	91
A.	PROCESS PARAMETERS FILE – TSMC018EPI.MD.....	91
APPENDIX C.	9-TO-5 BIT CONVERSION – MINTERM CALCULATION.....	93
A.	OVERVIEW.....	93
APPENDIX D.	PROCESS TECHNOLOGY	101
A.	OVERVIEW.....	101
B.	MOSIS PROCESSES	101
1.	Overview	101
2.	SCMOS Design Rules	102
3.	Standard SCMOS	102
4.	Well Type.....	102
APPENDIX E.	TANNER TOOLS DESCRIPTION	105
A.	OVERVIEW.....	105
B.	TANNER TOOLS.....	105
1.	Simulation Tools	105
2.	Frontend and Netlist.....	105
3.	Mask-Level Tools.....	105
	LIST OF REFERENCES.....	107

INITIAL DISTRIBUTION LIST	109
--	------------

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	USS Crockett (From [1]).	2
Figure 2.	AN/APS-137 ISAR image of the USS Crockett (From [1]).	2
Figure 3.	Block Diagram of the False Target Radar Image Synthesizer System (After [2]).	3
Figure 4.	Plot of the Arctangent Function.	10
Figure 5.	Complex Plane Showing Example Angle Measurement.	11
Figure 6.	First Quadrant Arctangent of a Complex Number.	12
Figure 7.	8 x 8 Combinational Multiplier (From [6]).	14
Figure 8.	15 th Order Polynomial Approximation to $\text{atan}(x)$.	16
Figure 9.	Arctangent(Q/I) – Floating Point Precision.	17
Figure 10.	Arctangent(Q/I) – Floating Point Precision, Quantized to 32 Values.	17
Figure 11.	Example: $Q = 5$, and $I = -5$, Phase Angle is 135° .	18
Figure 12.	Fourth Quadrant Five-bit Arctangent LUT.	19
Figure 13.	CORDIC Example: $z_0 = -8 + 4j$.	25
Figure 14.	CORDIC Arctangent(Q/I).	28
Figure 15.	CORDIC Arctangent(Q/I) Quantized to 32 Values.	28
Figure 16.	Bit Parallel Iterative CORDIC (From [9]).	29
Figure 17.	Bit Serial Iterative CORDIC (From [9]).	30
Figure 18.	Bit Parallel Unrolled CORDIC [After 9].	31
Figure 19.	N-FET.	34
Figure 20.	P-FET.	34
Figure 21.	N-FET I_{DS} vs. V_{DS} .	35
Figure 22.	P-FET I_{DS} vs. V_{DS} .	36
Figure 23.	Inverter Symbol (top left) and Schematic (right).	37
Figure 24.	β_N/β_P Ratios and Inverter Noise Margins.	39
Figure 25.	Inverter Proper Operation.	41
Figure 26.	Inverter T_F and T_{PDHL} .	41
Figure 27.	Inverter T_R and T_{PDLH} .	42
Figure 28.	Inverter Current Draw.	42
Figure 29.	Pass Gate Symbol (top left) and Schematic (right).	43
Figure 30.	Buffer Symbol (top left) and Schematic (right).	44
Figure 31.	XOR Symbol (top left) and Schematic (right).	46
Figure 32.	NAND2 Symbol (top left) and Schematic (right).	47
Figure 33.	Register Schematic (From [12]).	49
Figure 34.	Register Control Logic (From [12]).	50
Figure 35.	One-bit DMSFF Register Symbol.	50
Figure 36.	16 Clock Period Delay – <i>Phase Signal Valid</i> Circuit.	51
Figure 37.	Buffer Driver Circuit.	52
Figure 38.	16-bit Pipelined Adder/Subtractor.	54
Figure 39.	Rotation Level Symbol (left) and Circuit (Right).	58
Figure 40.	$\pm 90^\circ$ Phase Loading.	59
Figure 41.	Negative 128 Circuit Fix.	60
Figure 42.	Two's Complement Circuit.	61

Figure 43.	CORDIC General Vector Level Schematic.	63
Figure 44.	CORDIC Level 14.03 Symbol (left) and Circuit (right).	64
Figure 45.	Zoomed In Phase Input – Phase Loading of 14.03125°.	65
Figure 46.	Zoomed in Q Hardwire Shifts.	66
Figure 47.	Zoomed in I Hardwire Shifts.	67
Figure 48.	RCA_9to5_Conversion Circuit.	72
Figure 49.	R0 Function Hardware Implementation.	74
Figure 50.	Completed Circuit.	76
Figure 51.	Power Supply Current Draw.	78
Figure 52.	<i>Clock</i> and <i>Load</i> Skew.	78
Figure 53.	VHDL Functional Verification.	80
Figure 54.	T-SPICE Timing Verification.	81

LIST OF TABLES

Table 1.	Output Number in Decimal vs. Phase in Degrees.....	13
Table 2.	Multiplying Complex Numbers.	22
Table 3.	Rotating by $\pm 90^\circ$	22
Table 4.	Add/subtract phases less than 90°	23
Table 5.	CORDIC Vectoring Flow (After [8]).	23
Table 6.	Detailed Calculation Flow of CORDIC Example (After [8]).	27
Table 7.	Detailed Hardware Flow of a six-iteration CORDIC Implementation.	32
Table 8.	N/P-FET Operating Point Data.	35
Table 9.	Inverter Noise Margins.	40
Table 10.	Inverter Electrical Parameters.	40
Table 11.	Pass Gate Electrical Characteristics.....	44
Table 12.	Buffer Noise Margins.	44
Table 13.	Buffer Electrical Characteristics.	45
Table 14.	XOR Truth Table.	45
Table 15.	XOR Electrical Characteristics.	46
Table 16.	NAND2 Truth Table.	47
Table 17.	NAND2 Noise Margins.	48
Table 18.	NAND2 Electrical Characteristics.	48
Table 19.	DMSFF1 Characteristics.....	49
Table 20.	Register Operation.	51
Table 21.	XOR Truth Table.	53
Table 22.	CORDIC Iteration Number Format.	55
Table 23.	16-Bit Number Format.....	56
Table 24.	Q7 Programming of $\pm 90^\circ$	57
Table 25.	Phase Groupings.	68
Table 26.	Minterms Necessary to Generate the 9-to-5 Conversion Circuit.....	68
Table 27.	Quine-McClusky Results of 9-to-5 Bit Logic Minimization.....	70
Table 28.	Circuit Parameters.....	77
Table 29.	9-to-5 Bit Phase Conversion Truth Table.	100

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to extend his utmost thanks to Associate Professor Douglas J. Fouts and Professor Phillip E. Pace. To Professor Fouts, for all his assistance, support, and patience during the completion of this thesis. For willingly offering assistance at all hours, for outstanding digital design advice provided, for acting as a sounding board, and for being the best thesis advisor a Naval Postgraduate School student could have. To Professor Pace, for providing a study carol, outstanding computer equipment, more RAM when requested, and for encouragement on the research of this thesis. Without both of your willing assistance, this thesis would not have been completed within the same time frame, and certainly would have been an inferior design.

Finally, I would also like to express my heartfelt thoughts to my Mother, Father, Sister and Grandparents for their support, understanding, and love throughout my life. Although they have been many miles away, they were always in my heart, where they will stay forever.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis documents the research, circuit design, and testing of a Very Large Scale Integration (VLSI) Application Specific Integrated Circuit (ASIC) which extracts the phase information from a complex signal via the arctangent function. The purpose of this chip design is for inclusion and use in a Digital Image Synthesizer (DIS) electronic warfare chip which generates false target radar images to counter wide band imaging Inverse Synthetic Aperture Radars (ISAR). Specifically, the circuit will convert the In-phase and Quadrature terms, comprised of eight data bits each, into the corresponding phase angle value expressed as a five-bit number. The design specifications are to implement the Amplitude-to-Phase Converter in Standard Complementary Metal Oxide Semiconductor (SCMOS) technology with a minimum transistor count, and ability to operate at a clock frequency of 700 MHz.

The first part of the thesis consists of arctangent function research to determine the different mathematical calculation methods. The most efficient implementation method to achieve the above stated design specifications was determined to be the CORDIC (Coordinate Rotation Digital Computer) algorithm and, thus, circuit design was completed. MATLAB simulations were used to verify calculation errors, determine noise margins, plot phase graphs, and to implement Quine-McClusky logic minimization. T-SPICE netlists were generated from the schematic and simulations were run to determine transistor and circuit electrical operation. Circuit simulations included: power and current draw, speed of operation, noise margins, DC transfer characteristics, and the verification of both timing and logic functionality. Finally, overall circuit logic functionality for all 65,536 input combinations was completed using a VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (VHDL) program.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND OF THE DIGITAL IMAGE SYNTHESIZER

The primary purpose and the *raison-d'être* for the design, development, and implementation of a phase converter is for its intended use in an electronic warfare chip. This chapter covers the Digital Image Synthesizer (DIS) and the mathematics required for its digital computer hardware implementation. As detailed in the paper, *A Single-Chip False Target Radar Image Generator for Countering Wideband Imaging Radars*, the following describes the DIS (After [1]):

Modern shipboard and airborne wideband synthetic aperture radars (SARs) and inverse synthetic aperture radars (ISARs) are capable of generating images of target objects. Figures 1 and 2 (courtesy of the Tactical Electronic Warfare Division of the U.S. Naval Research Laboratory) show a photograph of the USS Crockett and an image of the ship obtained from a U.S. Navy AN/APS-137 ISAR. Such imaging capability is an advantage over previous technology because it improves the ability to identify the specific type of target, distinguish friend from foe, accurately guide weaponry, and defeat electronic protection such as false target decoys. Thus, modern wideband imaging SARs and ISARs create a difficult ship defense problem. For example, if an adversary is using a wideband imaging ISAR, an electronic protection system cannot synthesize a false target by just transmitting a signal that emulates a radar return off a single or a few scattering surfaces. Instead, such a transmitted signal must emulate a coherent sequence of reflections with proper delay, phase, and amplitude that is similar to what would come from the multiple scattering surfaces at multiple ranges (distances from the radar) of an actual ship.

Analog methods for generating false radar targets have included the use of acoustic charge transport (ACT) tapped delay lines and fiber optic tapped delay lines. ACT devices are no longer commercially available and also have limited bandwidth, making them impractical against wideband imaging radars. Optical devices are bulky and costly to manufacture, especially for the longer delay line lengths needed to synthesize a false target image of even a moderately-sized ship. However, the equations and algorithms needed to digitally synthesize a false target radar image have evolved considerably over the last several years. With modern, digital signal processing (DSP) techniques and advanced VLSI fabrication processes, it is now possible to digitally synthesize a realistic false target radar image of even a large war ship such as an aircraft carrier.

Figures 1 and 2 are reproduced below:



Figure 1. USS Crockett (From [1]).

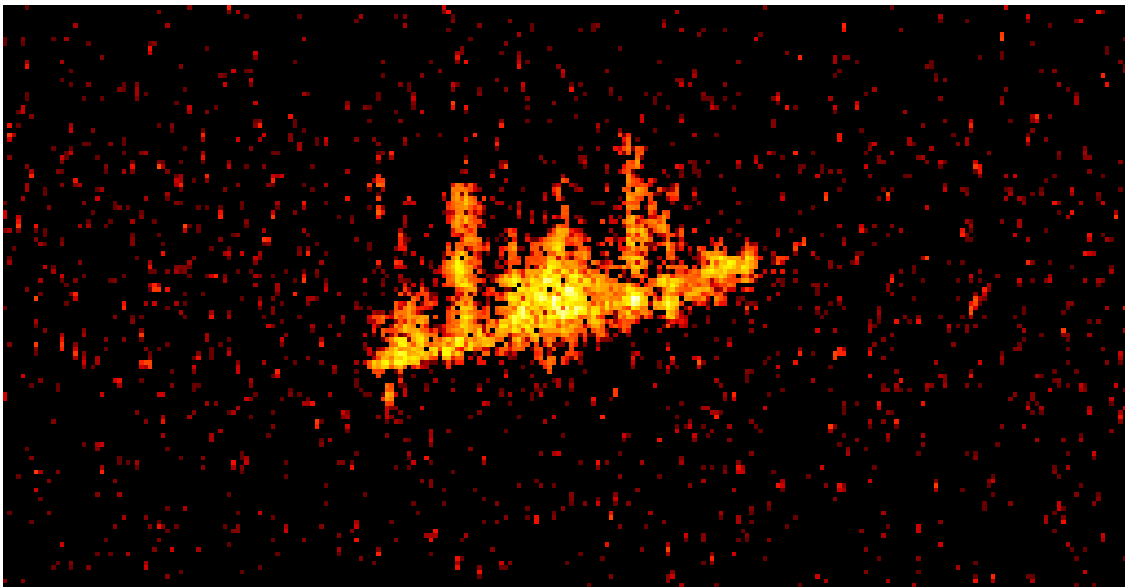


Figure 2. AN/APS-137 ISAR image of the USS Crockett (From [1]).

Figure 3 shows the high-level circuit block diagram that illustrates the virtual architecture of the DIS. The Q and I amplitude to phase conversion portion is highlighted in red and yellow:

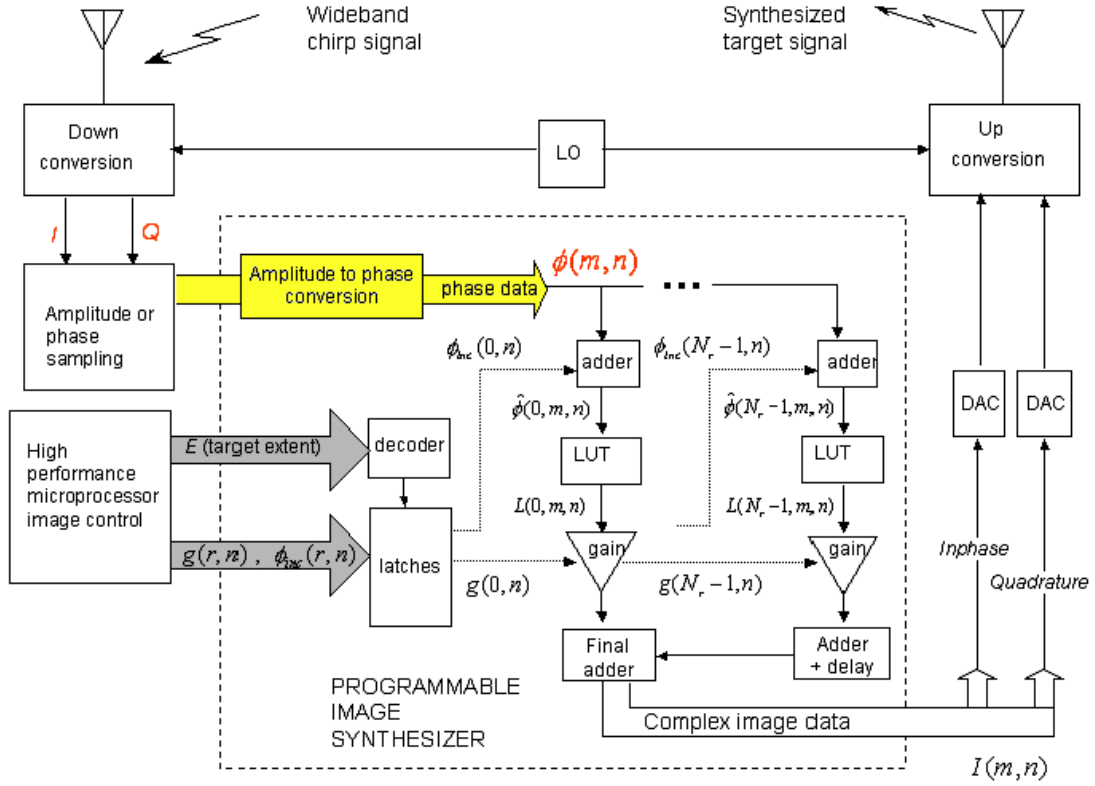


Figure 3. Block Diagram of the False Target Radar Image Synthesizer System (After [2]).

Representing the virtual architecture as a mathematical one for hardware implementation, the intercepted chirp pulse, s , as a function of time, can be represented as a complex signal of the form [2]:

$$s(t) = \text{rect}\left(\frac{t}{\tau}\right) e^{j2\pi(f_d t + \Delta f^2 / 2\tau)} \quad (1.1)$$

where:

- f_d Doppler frequency (between DRFM platform and ISAR),
- τ Pulse width,
- Δ Modulation bandwidth,

and the definition of the ‘rect’ function is [2]:

$$\text{rect}\left(\frac{t}{\tau}\right) = \begin{cases} 1 & \text{for } \left|\frac{t}{\tau}\right| < \frac{1}{2} \\ 0 & \text{for } \left|\frac{t}{\tau}\right| > \frac{1}{2} \end{cases}. \quad (1.2)$$

The intercepted chirp pulse is sampled and quantized via an Analog-to-Digital Converter (ADC) generating the eight-bit In-Phase, I , and Quadrature, Q , data. The DIS chip accepts as inputs this Q and I data and, via the amplitude-to-phase conversion circuit, produces a corresponding phase angle, ϕ . Mathematically using Euler's Theorem, the complex exponential of Equation 1.1 can be represented as a sum of cosine and sin terms:

$$e^{\pm jx} = \cos x \pm j \sin x. \quad (1.3)$$

The ADC converter samples the waveform 90° out of phase to generate the $\cos x$ and $\sin x$ terms for each I and Q data set, respectively, the real and imaginary parts of the complex signal. To determine the corresponding phase value of the Q and I data set, the phase angle can be expressed as [2]:

$$\phi(m, n) = \angle \left\{ \frac{\text{Im}\{s(t)\}}{\text{Re}\{s(t)\}} \right\} \bigg/ \frac{2\pi}{2^{k_p}}, \quad (1.4)$$

where:

- $\angle$, the angle of the parenthesized arguments, then quantized to k_p bits,
- $\text{Im}\{s(t)\}$, imaginary Quadrature Term, Q , and
- $\text{Re}\{s(t)\}$, real In-Phase Term, I .

Stated another way, this represents the inverse (arc) tangent function because this function produces a phase angle based upon its argument:

$$z = \tan^{-1}(Q/I) \quad (1.5)$$

which is then subsequently quantized to $2^{k_p} = 2^5 = 32$ values as the DIS hardware operates on five-bit phase data.

Therefore, in order to generate the phase value $\phi(m, n)$ from Q and I data, it is necessary to implement the arctangent function in hardware. This thesis thereby contributes to the DIS chip and is both an essential and critical module.

B. PRINCIPAL CONTRIBUTIONS

Initially, research was conducted on the arctangent function to determine its characteristics and the different means of calculation. In an effort to minimize transistor count, methods that avoid division and multiplication were further investigated. Calculus, polynomial approximations, and direct logic function implementation were quickly excluded as options because they would require high transistor counts and/or produced excessive errors.

A lot of research was spent on the Look Up Table (LUT), as it is fundamentally a very simple way to implement the amplitude to phase conversion. Indeed, the Q and I values can be used as indexes to a table where a five-bit result is stored in the corresponding addressed location. Unfortunately, the LUT size would be very large (65,536 entries) and thus methods to reduce the table size were considered. Primarily, only one quadrant of arctangent data would be required to be stored in a LUT because the remaining three quadrant results could be determined by the sign of the input Q and I values. Simple addition can be performed to translate the first quadrant phase result to another quadrant. Reducing the number of bits by truncating the eight-bit Q and I data to seven-bits was also researched, and discarded because of high resulting errors.

The CORDIC implementation method was researched to determine the required transistor count and compared to the LUT approach. The CORDIC algorithm can be unrolled, which leads to a very nice pipelined implementation. Since the number of transistors to implement CORDIC is approximately one quarter the amount needed for a one-quadrant LUT, the CORDIC method was chosen and circuit design undertaken.

Using the S-Edit circuit design CAD tool from Tanner Research [3], the Rotation level was first designed and debugged. Mask layout of this circuit was completed in the Tanner Research layout editor L-Edit, and the results verified via test vectors using the

circuit simulator T-SPICE. A general vector level was designed, and individual CORDIC iteration levels were tailored using the general level by changing hardwired constants and shifts. After renditions to tweak the design and debug it, a completed schematic was finished and laid out which produced a nine-bit, z , phase result.

A conversion circuit is required to convert the nine-bit phase result produced by the CORDIC algorithm into a five-bit value for use in the DIS circuit. Different implementation options including ROM, comparators, Sum-of-Products minimal logic, and multiplexers were considered. A ROM (essential to a LUT approach) and multiplexers would have a very large gate count and were, therefore, excluded as options. The comparator implementation method was fundamentally the most straightforward, elegant, and simple to implement, but suffered from having approximately three times more logic gates than a minimal sum-of-products implementation. Quine-McClusky minimization algorithms were run using MATLAB to generate the five logic functions (one function for each bit), which were subsequently designed in S-Edit and laid out in L-Edit.

At this point in the research, a completed schematic and mask layout was finished and test vector cases simulated using T-SPICE and verified. As a final functional test of the circuit, the entire schematic was exported to a VHDL program and tested for all 65,536 possible inputs, and phase result errors were found for some Q and I input vectors. We determined that the number of bits used in the CORDIC hardware for precision was insufficient, and the circuit was fixed to account for them. Unfortunately, the previously finished mask layout would have to be redesigned, as it could not be easily patched to account for the new circuit changes, and as such, this old mask layout is not included as part of this thesis. This updated schematic was re-verified for both functional and timing correctness, producing 100% correct results. Mask layout of the updated schematic is to be completed at a later date.

C. ORGANIZATION OF THESIS

This thesis documents the research, hardware implementation considerations, design, and testing of a VLSI ASCI complex signal amplitude to phase conversion circuit for use in the DIS. It is organized as follows:

Chapter II presents detailed research on the arctangent function and methods that can be used to calculate it. These methods are examined to determine whether they meet the stated goals, and how they compare to each other in terms of accuracy, digital hardware transistor count, and minimum clock frequency of operation.

Chapter III presents the hierarchical progression of the circuit design of the complex signal amplitude-to-phase converter. Important design considerations, techniques, and approaches are presented for the transistor up to the complete design.

Chapter IV summarizes the results of the thesis, key lessons learned, and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. INVESTIGATION OF CONVERSION METHODS

A. TASKS

1. Aim

The goal of this research was to design and implement an amplitude-to-phase conversion circuit for the inputs to the Digital Image Synthesizer (DIS) from a Digital Radio Frequency Memory (DRFM). The basic algorithm is to extract a five-bit phase angle from eight-bit Quadrature and In-Phase inputs by performing the arctangent function.

2. Implied Tasks

- Determine the minimum transistor count hardware solution to implement the arctangent function by investigating different computational methods
- Determine the most efficient method for digitally implementing an eight-bit Q and I amplitude to phase conversion circuit which produces a five-bit number
- Design and test a pipelined optimum circuit able to run at a minimum of 700 MHz, including architectural and logic design, verification, and simulation
- Minimize circuit average power draw.

B. GENERAL

This chapter investigates the different means of implementing an amplitude-to-phase conversion circuit. It first covers information on the arctangent function to investigate its characteristics, as it is important to know the domain, range, and function dependency on the independent variable. Within the guidelines of the implied tasks, those mathematical methods that show potential for digital implementation are further examined.

1. The Arctangent Function of a Real Number

The arctangent function: $y = \tan^{-1} x$ is the inverse of the restricted function [4]:

$$y = \tan x, \quad -\frac{\pi}{2} < x < \frac{\pi}{2}. \quad (2.1)$$

For every real value of x , $y = \tan^{-1} x$ is the angle between $-\pi/2$ and $\pi/2$ whose tangent is x . The domain of the x values is $-\infty$ to $+\infty$. The graph of $\tan^{-1} x$ is symmetric about the origin, and is an odd function of x :

$$\tan^{-1}(-x) = -\tan^{-1} x. \quad (2.2)$$

From the Figure 4 graph of $\tan^{-1} x$, it can be seen that $\tan^{-1} x$ has the same sign as x and that $\tan^{-1} 0 = 0$. The red lines are the asymptotes that the function approaches as x goes to $\pm\infty$.

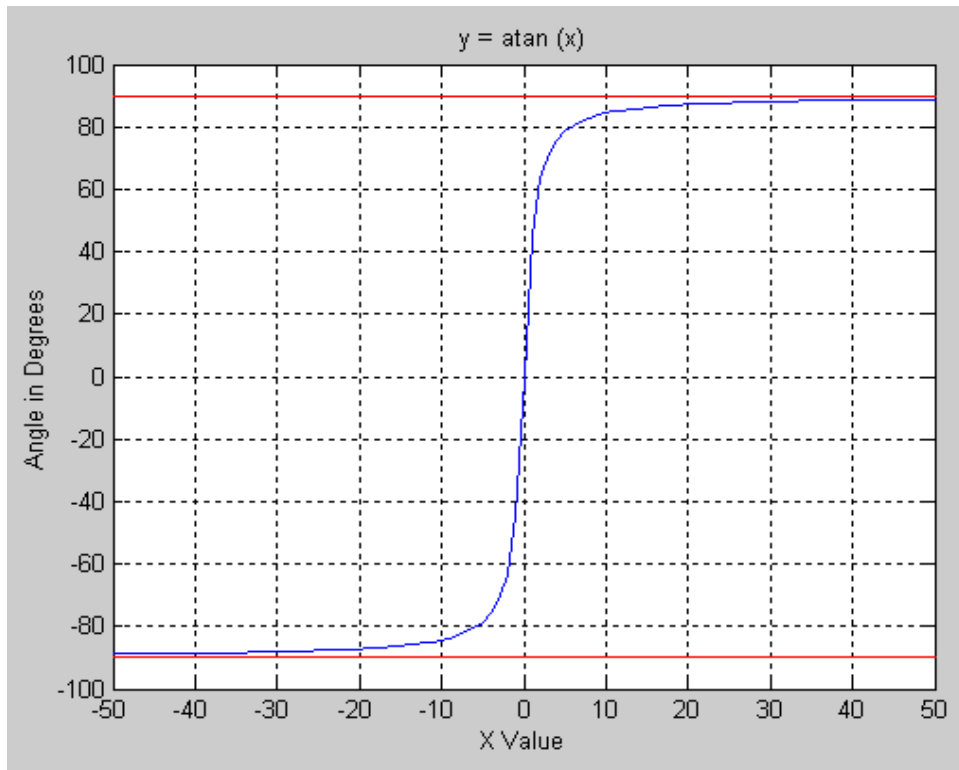


Figure 4. Plot of the Arctangent Function.

By visual inspection, it is easy to verify that most changes of the function occur for values of x between ± 15 . Values of x larger than this change incrementally more slowly as the asymptotic limits are approached. Another observation is that the ranges of angles produced by the arctangent are 0° to 45° for $0 < x < 1$. Thus, to determine values of phase from 45 to 90° for $x > 1$, one can calculate:

$$\pi - \tan^{-1}\left(\frac{1}{x}\right). \quad (2.3)$$

2. The Arctangent Function of a Complex Number

For a complex number $z = x + jy$, given the real and imaginary parts, the arctangent function is (written in different formats) [5]:

$$\arctan 2(x, y) = \arctan(y/x) = \tan^{-1}(y/x) = \arg(x + yi) \quad (2.4)$$

where:

- $y = \text{Im}\{z\}$
- $x = \text{Re}\{z\}$.

The arctangent of the quotient y/x is the angle of the magnitude vector measured counter-clockwise on the unit circle from the positive real x -axis. The *arg* in Equation 2.3 stands for the argument of the complex number and represents the phase value of a complex number. Figure 5 shows the complex plane, the four quadrants, and a sample angle measurement of a complex number vector shown in red.

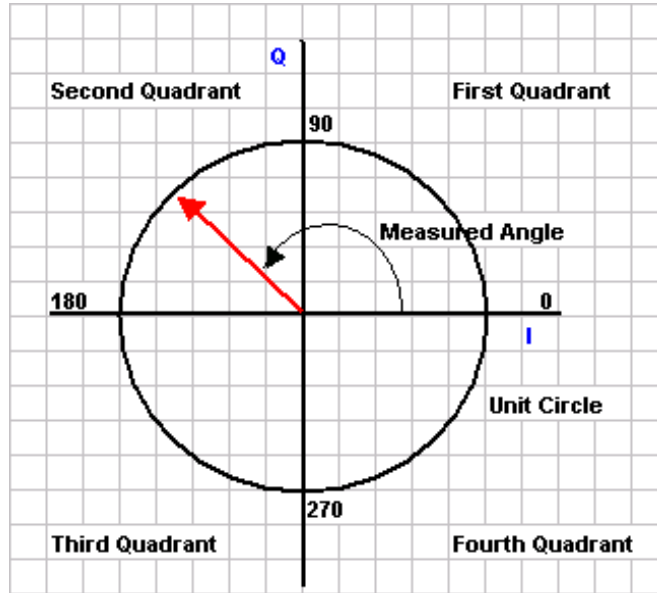


Figure 5. Complex Plane Showing Example Angle Measurement.

Figure 6 graphically displays the first quadrant phase values of $\tan^{-1}(Q/I)$ for quadrant (positive) values of Q and I . The reader can verify the phase ranges from 0° to 90° within the first quadrant, as expected. The different colors show different resulting

phase values for changing Q and I inputs. Constant colors for changing Q and I inputs logically imply that the arctangent function generates the same phase value. Consider two vectors of the form (Q,I) : (5,5) and (10,10). They both possess a 45°-phase value although they have different magnitudes.

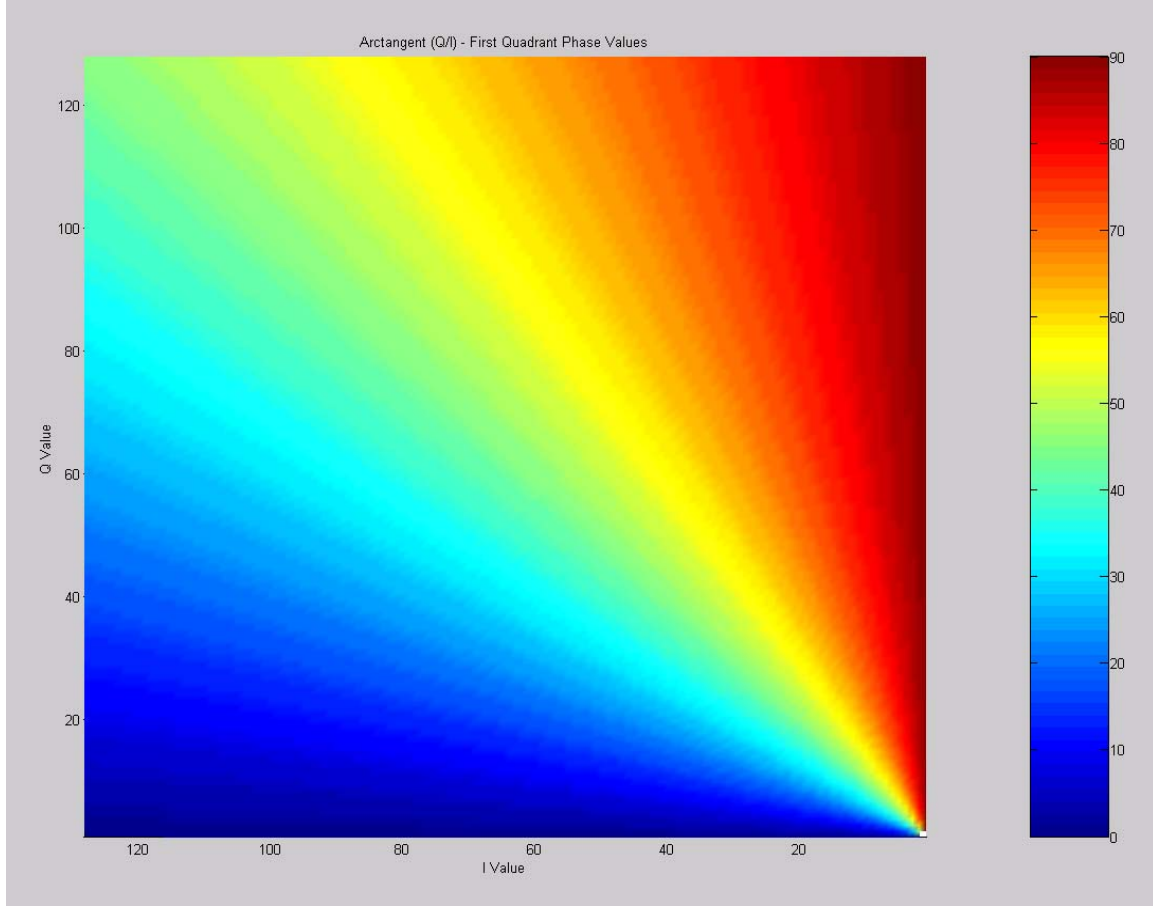


Figure 6. First Quadrant Arctangent of a Complex Number.

3. Five-Bit Phase Quantization

The DIS uses a five-bit phase value that linearly increases from 0 to 31 as one maps out the unit circle in a counter-clockwise direction. As it is not possible to represent every integer phase from 0° to 360° using only 32 values, ranges of phase will be indistinguishable from one another. Equivalently, the resolution of phase angles is:

$$\frac{360^\circ}{5 \text{ bits}} = \frac{360^\circ}{2^5 \text{ values}} = 11.25 \text{ }^\circ/\text{bit}. \quad (2.5)$$

A tabular description showing five-bit decimal values verse corresponding degree values is shown as Table 1:

Output Number in Decimal vs. Phase in Degrees			
Number	Phase	Number	Phase
0	0	16	180
1	11.25	17	191.25
2	22.5	18	202.5
3	33.75	19	213.75
4	45	20	225
5	56.25	21	236.25
6	67.5	22	247.5
7	78.75	23	258.75
8	90	24	270
9	101.25	25	281.25
10	112.5	26	292.5
11	123.75	27	303.75
12	135	28	315
13	146.25	29	326.25
14	157.5	30	337.5
15	168.75	31	348.75

Table 1. Output Number in Decimal vs. Phase in Degrees.

With only five-bit phase resolution, all values of phase between 0° and 11.25° are indistinguishable, and similarly for 11.26° to 22.5° , etc. This is important and will be discussed later in the conversion of a nine-bit phase value to five-bits. A nine-bit number has $2^9 = 512$ different values and is the minimum number of bits required to represent all 360° integer numbers.

C. AMPLITUDE-TO-PHASE CONVERSION METHODS

There are several means to calculate the arctangent function including:

- Calculus,
- Look-up table,
- Sum-of-Products Logic Block,
- Polynomial approximation, and
- Coordinate Rotation Digital Computer (CORDIC).

The issue is to select a method that best meets the stated goals. The simplest hardware method which possesses a minimum gate count is required and, as such, computational

and gate heavy methods which involve multiplication, or worse, division should be avoided. Consider an eight by eight multiplier shown in Figure 7:

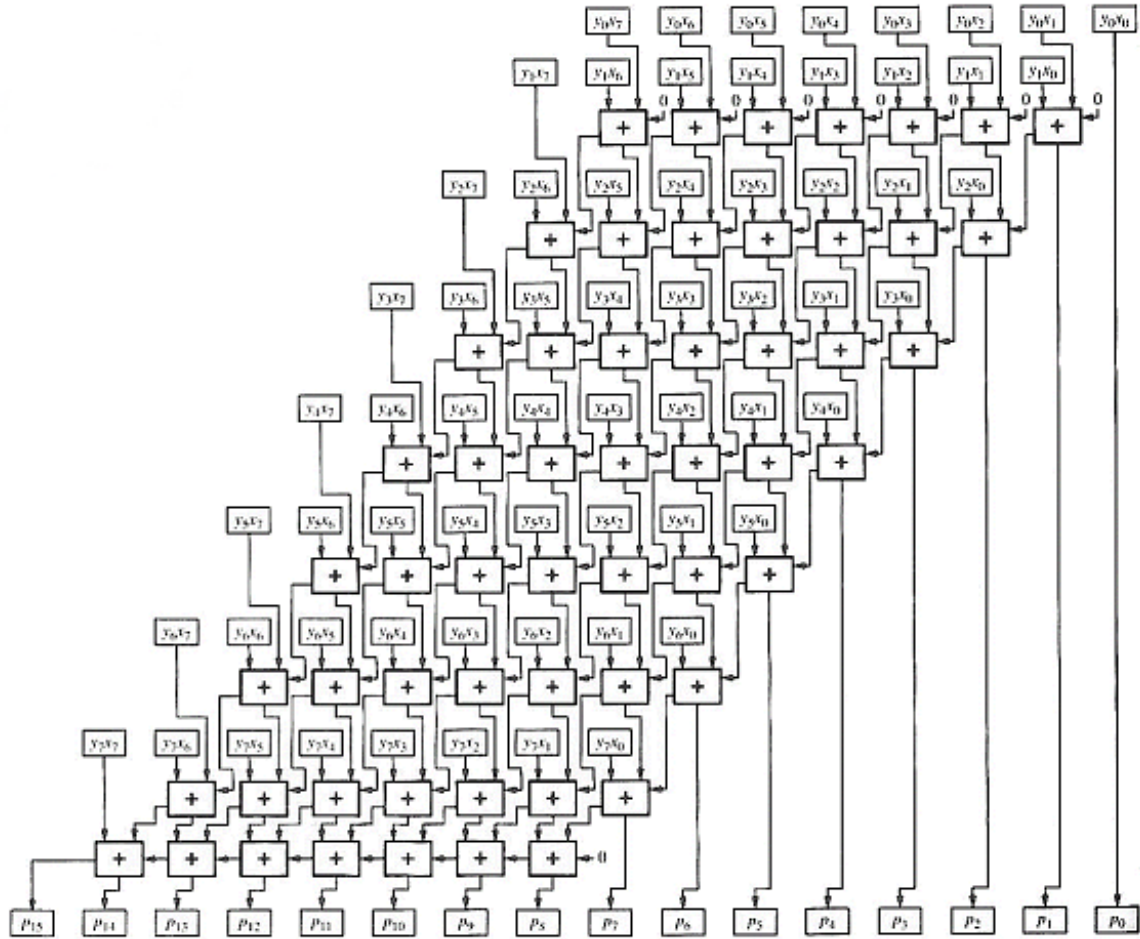


Figure 7. 8 x 8 Combinational Multiplier (From [6]).

Terms $y_i x_j$ are product terms generated by the logical AND function, and graphical boxes with the plus symbol are one-bit Full Adders. Implementation of multiplication has many logic gates and correspondingly long combinatorial delay. Division in hardware is one of most complex and difficult to achieve at high speeds. It is very hardware intensive, slow, and requires high bit precision and/or many iterations to produce accurate results.

Therefore, methods that do not require the computation of Q/I or have multiplication are to be favored.

1. Calculus

Calculus methods involve limits and series. The derivative of the arctangent may be defined as [4]:

$$\frac{d}{dx} \tan^{-1} x = \frac{1}{1+x^2} \quad (2.6)$$

and, therefore by integration, the arctangent function can be calculated [4]:

$$\tan^{-1} x = \int \frac{1}{1+x^2} dx. \quad (2.7)$$

Alternatively, for a complex number, z , it can be equated log-arithmetically [5]:

$$\tan^{-1} z = \frac{i}{2} \cdot \log \left(\frac{i+z}{i-z} \right). \quad (2.8)$$

To implement a calculus method would require the determination of x^2 , a division of Q/I to produce x , a multiplication of x times x , followed by an addition and division operation to produce $1/(1+x^2)$. Subsequently, a large number of summations would be required to closely approximate the integration. As such, further consideration of a calculus hardware implementation was abandoned.

2. Polynomial Approximation

The arctangent function may be represented as a Taylor series [4]:

$$\tan^{-1} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2n+1}}{2 \cdot n + 1} \quad |x| \leq 1 \quad (2.9)$$

and it is obvious by inspection that a large number of multiplications and divisions are required. Using MATLAB, a 15th-order polynomial approximation to the arctangent function was coded. Figure 8 shows a plot of $\text{atan}(x)$ vs. the polynomial approximation and the resulting large error between the functions is readily apparent. To more closely approximate the arctangent function, an even higher order polynomial would be required. As hardware implementation would require many multiplications, divisions, and additions, and because even a 15th-order polynomial approximation generates a large resulting error, further consideration was not given to implementing a polynomial

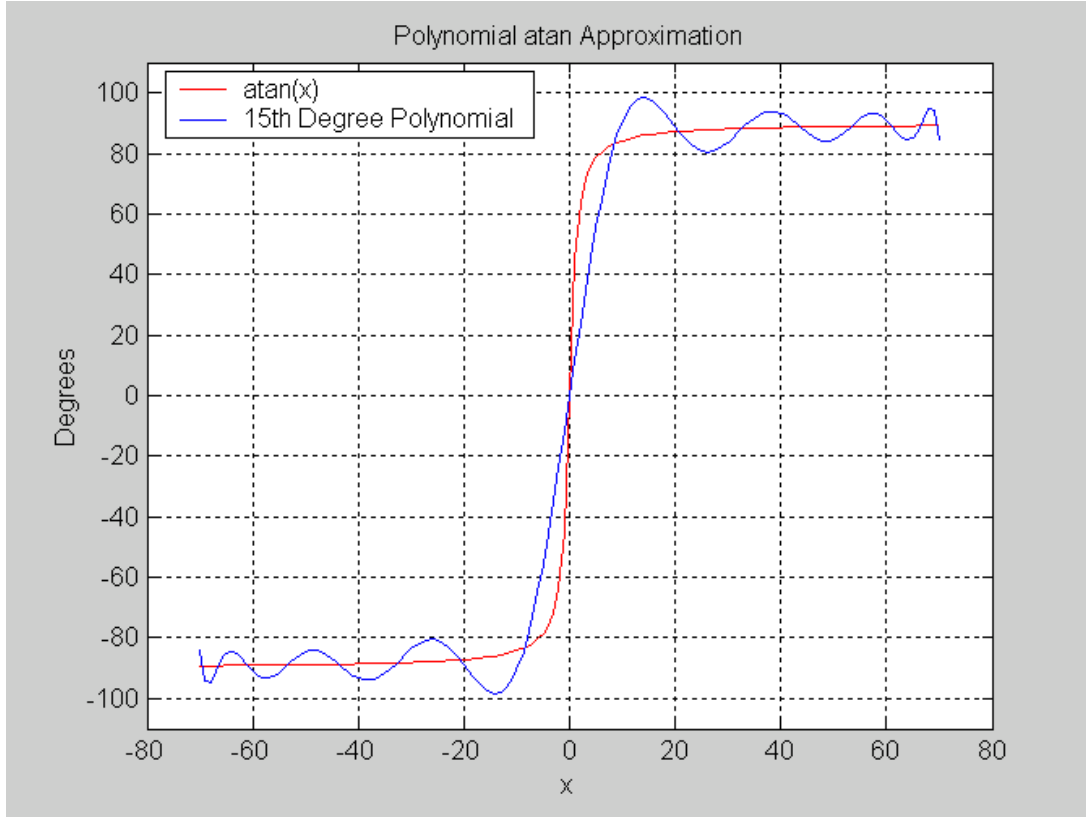


Figure 8. 15th Order Polynomial Approximation to $\text{atan}(x)$.
approximation.

3. Look Up Table

One of the issues behind computationally calculating the arctangent is the necessity to determine the integer division of Q/I . A lookup table (LUT) allows inherent computation of the division because the actual value of $\tan^{-1}(Q/I)$ can be programmed into the table and hence this is a strong favorite for implementation. Of all methods studied, it is the most simple. Indexes into the table are the values of Q and I , and the value at that addressed location is the corresponding result stored as a five-bit number. Figure 9 shows all four quadrants of $\tan^{-1}(Q/I)$ with eight-bit Q and I inputs. There are 65,536 different phase values, as there are 2^8 times $2^8 = 2^{16}$ possible input combinations. The plot was generated in MATLAB using short floating-point precision calculations. The representation of the eight-input Q and I data is a signed two's complement number and, therefore, the range of values is -128 to $+127$. Figure 10 shows the same results with the resulting phase quantized to five-bits. There are now only 32 different phases (colors), showing 11.25° degree "steps".

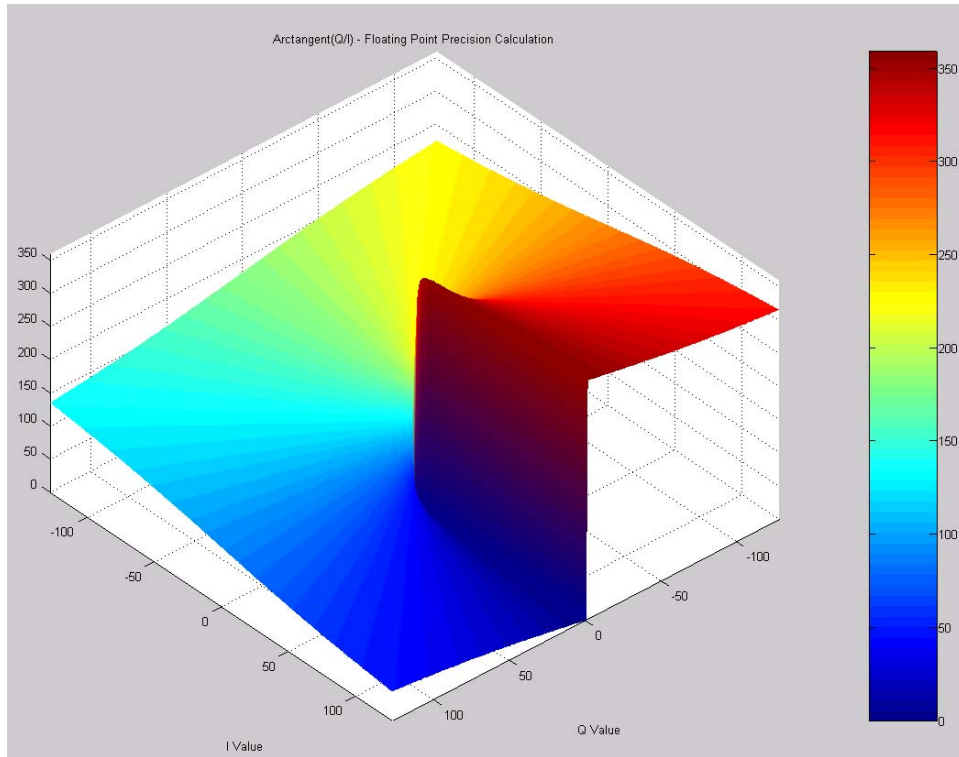


Figure 9. $\text{Arctangent}(Q/I)$ – Floating Point Precision.

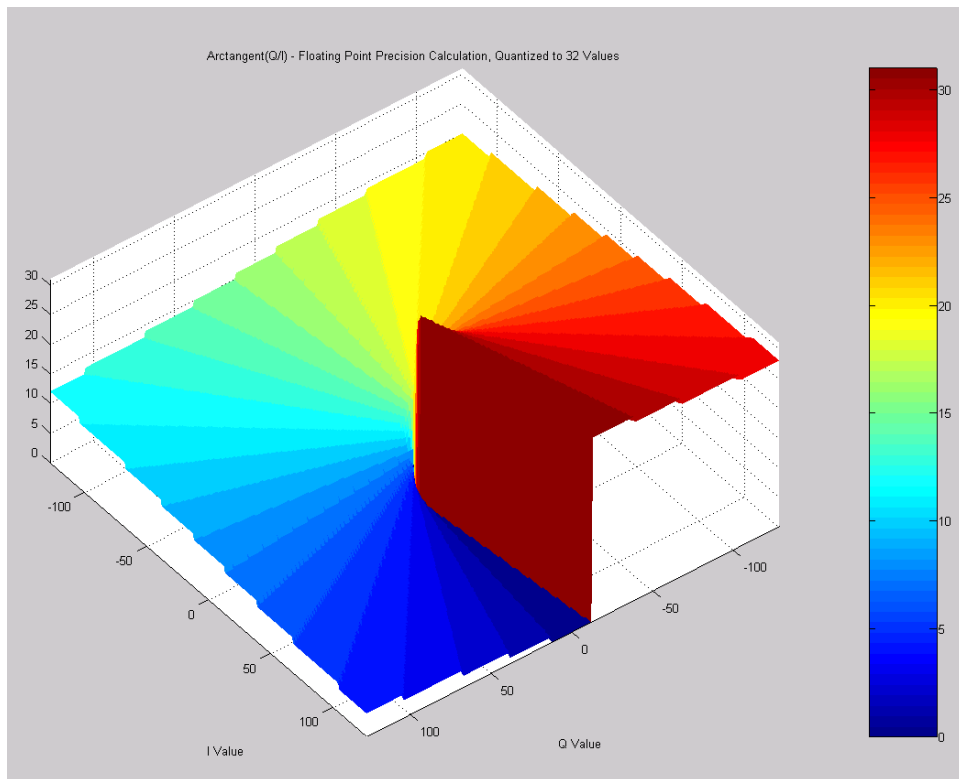


Figure 10. $\text{Arctangent}(Q/I)$ – Floating Point Precision, Quantized to 32 Values.

A 16-bit input LUT would require 65,536 entries with each address containing a five-bit value. This would be a very large transistor array to implement and would require on the order of 65,536 locations * 5 FETs = 328k FETs, not including decoding circuitry. Research was then done to determine if the LUT size could be reduced. As a particular quadrant of any given phase is determined by the sign values of the Q and I inputs, the LUT is inherently redundant. It is only necessary to look up one quadrant of stored arctangent data because the phase values of Q and I inputs to other quadrants can be determined through rotation by the addition of a constant phase. Some calculators operate on this same principle and require the user to verify that the answer is correctly translated. For example, consider data input of (5, -5) shown in Figure 11:

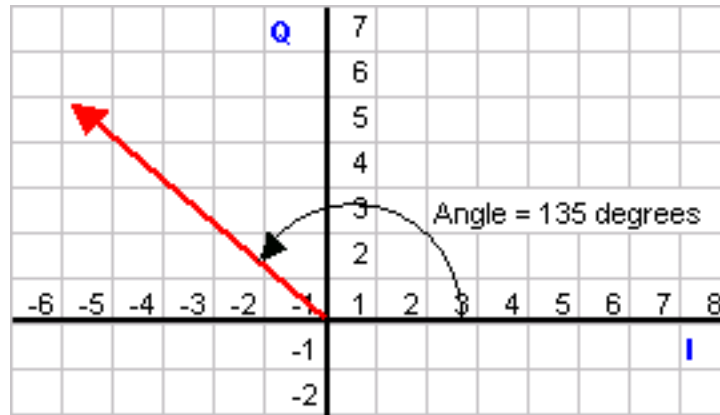


Figure 11. Example: $Q = 5$, and $I = -5$, Phase Angle is 135° .

the Ti89 calculator gives the result:

$$\tan^{-1}\left(\frac{5}{-5}\right) = -\frac{\pi}{4} = -0.785398 \text{ rad} = -45^\circ$$

The addition of 180° would translate the answer to the second quadrant:

$$180^\circ + (-45^\circ) = 135^\circ$$

which is the correct answer. Referring to Figure 5, for positive values of Q and I , the values of phase lie in the range of 0° to 90° , and the procedure to use to translate a one-quadrant LUT would be:

- 1) If Q or I are negative, convert the number to a positive value via the two's complement. Store a sign bit result that this was done; one bit each for Q and I .
- 2) With both Q and I now positive numbers, index into a first quadrant LUT and read the five-bit phase angle.
- 3) Based on the stored sign bits, if:
 - i. $Q > 0$ and $I < 0$, second quadrant, then angle = angle + 90°
 - ii. $Q < 0$ and $I < 0$, third quadrant, then angle = angle + 180°
 - iii. $Q < 0$ and $I > 0$, fourth quadrant, then angle = angle + 270° .

Thus, a one quadrant LUT would only require $16,384 \text{ addresses} * 5 \text{ FETs/address} = 82\text{k}$ FETs, plus some addressing-decoding control logic and a five-bit adder. A transistor count of saving is roughly one-fourth. Figure 12 shows the fourth quadrant portion of a five-bit Q and I LUT as an example, where Excel calculated the values stored. The color fill shows groups of 11.25° quantized five-bit values, and thus groupings of indistinguishable phase values.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-15	0	-4	-8	-11	-15	-18	-22	-25	-28	-31	-34	-36	-39	-41	-43	-45
-14	0	-4	-8	-12	-16	-20	-23	-27	-30	-33	-36	-38	-41	-43	-45	-47
-13	0	-4	-9	-13	-17	-21	-25	-28	-32	-35	-38	-40	-43	-45	-47	-49
-12	0	-5	-9	-14	-18	-23	-27	-30	-34	-37	-40	-43	-45	-47	-49	-51
-11	0	-5	-10	-15	-20	-24	-29	-32	-36	-39	-42	-45	-47	-50	-52	-54
-10	0	-6	-11	-17	-22	-27	-31	-35	-39	-42	-45	-48	-50	-52	-54	-56
-9	0	-6	-13	-18	-24	-29	-34	-38	-42	-45	-48	-51	-53	-55	-57	-59
-8	0	-7	-14	-21	-27	-32	-37	-41	-45	-48	-51	-54	-56	-58	-60	-62
-7	0	-8	-16	-23	-30	-36	-41	-45	-49	-52	-55	-58	-60	-62	-63	-65
-6	0	-9	-18	-27	-34	-40	-45	-49	-53	-56	-59	-61	-63	-65	-67	-68
-5	0	-11	-22	-31	-39	-45	-50	-54	-58	-61	-63	-66	-67	-69	-70	-72
-4	0	-14	-27	-37	-45	-51	-56	-60	-63	-66	-68	-70	-72	-73	-74	-75
-3	0	-18	-34	-45	-53	-59	-63	-67	-69	-72	-73	-75	-76	-77	-78	-79
-2	0	-27	-45	-56	-63	-68	-72	-74	-76	-77	-79	-80	-81	-81	-82	-82
-1	0	-45	-63	-72	-76	-79	-81	-82	-83	-84	-84	-85	-85	-86	-86	-86
0	0	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90

Figure 12. Fourth Quadrant Five-bit Arctangent LUT.

The LUT approach has several different advantages and disadvantages. The computation of $\tan^{-1}(Q/I)$ can be completed in one simple step by the logical use of Q and I as indices into the table. By only using the first quadrant of the LUT, the number of transistors required can be minimized. Sign values of Q and I can be used to add a constant phase to the output which only requires a small five-bit hardware implemented

adder. On the downside, fairly large decoding circuitry is needed, and a one-quadrant LUT is still large in terms of transistor count. Thus, the LUT approach was decided against for implementation of the amplitude to phase conversion circuit.

4. Sum-of-Products Logic Block

The amplitude to phase conversion can (theoretically) be accomplished directly by a logic function that requires a minimum of three logic levels of gates: NOT-AND-OR, or via Demorgan's Theorem, NOT-NAND-NAND. Mano explains in *Digital Design* [7]:

The complexity of the digital logic gates that implement a Boolean function is directly related to the complexity of the algebraic expression which the function is implemented. Although the truth table representation of a function is unique, expressed algebraically, it can appear in many different forms. Boolean functions may be simplified by algebraic means...

Minimization methods include Karnaugh Maps, Quine-McClusky (a tabular algorithm), and heuristic methods such as the Espresso program. The Sum-of-Products (SOP) uses minterms (output of the function is true or a logical one) to form the function expression. The product denotes the AND operation and the sum denotes the OR operation. As an example, a Full Adder function expressed in sum of minterms is:

$$F(x, y, z) = \sum(1, 2, 4, 7) = x' \cdot y' \cdot z + x' \cdot y \cdot z + x \cdot y' \cdot z' + x \cdot y \cdot z$$

There are impracticable issues with implementing large functions as direct logic functions. The amplitude-to-phase conversion circuit has a minimum of 16 inputs, not including a *Clk* input for registers, and therefore, the number of possible output combinations is 2^{16} , an incredibly large number! There are five bits produced by the phase conversion and, thus, five different functions are required to be implemented. Quine-McClusky minimization on a nine-input function took 10 computing days on a 1.4 GHz Pentium IV machine. As the number of possible output functions increases (the number of inputs increases), the time to tabular search and minimize the function grows exponentially. A 16-bit input function would take an exceedingly long time to compute. Second, a very large number of function terms would be generated, well exceeding fan

out and fan in limitations of the logic gates. As a test, Professor Fouts used Espresso to generate the logic equations to produce five-bit output amplitude-to-phase conversion circuit of 16 inputs. The time of computation took three days, which is orders of magnitude faster than a Quine-McClusky computation. However, Espresso does not guarantee a minimal solution. The gate count to implement the functions via Espresso were:

First Plane of Logic Gates:

- 0 inverters
- 1, 2-input NAND
- 5, 3-input NAND
- 1, 4-input NAND
- 1, 5-input NAND
- 26, 6-input NAND
- 100, 7-input NAND
- 232, 8-input NAND
- 432, 9-input NAND
- 627, 10-input NAND
- 857, 11-input NAND
- 755, 12-input NAND
- 557, 13-input NAND
- 182, 14-input NAND
- 16, 15-input NAND.

Second Plane of Logic Gates:

- 109-input NAND to generate output bit 4
- 233-input NAND to generate output bit 3
- 784-input NAND to generate output bit 2
- 1429-input NAND to generate output bit 1
- 2251-input NAND to generate output bit 0.

The total transistor count equaled 92,468 FETs, not including required buffers. Brute force SOP implementation of the amplitude to phase conversion is, therefore, not a viable option.

5. CORDIC

A Coordinate Rotation Digital Computer (CORDIC) algorithm is a class of iterative shift and add algorithms for rotating vectors in a plane until a result converges to any desired precision or error. The error is proportional to the number of iterations

performed, unlike analytic iterative processes. In a simple operation, CORDIC performs a sequence of rotations on two-dimensional vectors using a series of specific incremental rotation angles selected so that each is performed by a shift and add operation [8].

Rotation of unit vectors provides a way to accurately compute trigonometric, logarithmic, exponential, square root, and hyperbolic functions, as well as a mechanism for computing the magnitude and phase angle of an input vector. The rotation of a vector is executed by multiplying it by a series of constant phases, where the multiplication is always a power of two. Thus, by shifting the vector (multiply by one-half or divide by two), no actual multiplication hardware is required. CORDIC generally produces one additional bit of accuracy for each iteration [8].

Mathematically describing the basic principles from the CORDIC FAQ [After 8]:

$$\begin{array}{ll} \text{Given a complex value:} & C = I_c + jQ_c \\ \text{create a rotated value:} & C' = I_{c'} + jQ_{c'} \\ \text{by multiplying by a rotation value:} & R = I_r + jQ_r \end{array}$$

1. When multiplying a pair of complex numbers, their phases add and their magnitudes multiply:

To add R 's from phase C :	$C' = C \cdot R$	$I_{c'} = I_c \cdot I_r - Q_c \cdot Q_r$
		$Q_{c'} = Q_c \cdot I_r + I_c \cdot Q_r$
To subtract R 's phase from C :	$C' = C \cdot R^*$	$I_{c'} = I_c \cdot I_r + Q_c \cdot Q_r$
		$Q_{c'} = Q_c \cdot I_r - I_c \cdot Q_r$

Table 2. Multiplying Complex Numbers.

2. To rotate by $+90^\circ$, multiply by $R = +j$. Similarly, to rotate by -90° , multiply by $R = -j$:

To add 90° :	$I_{c'} = -Q_c$	(negate Q , then swap)
	$Q_{c'} = I_c$	
To subtract 90° :	$I_{c'} = Q_c$	(negate I , then swap)
	$Q_{c'} = -I_c$	

Table 3. Rotating by $\pm 90^\circ$.

3. To rotate by phases of less than 90° , successively multiply by numbers of the form " $R = I \pm jK$ " where K will be decreasing in powers of two, starting with

$2^0 = 1.0$. The symbol " L " designates the power of two itself: 0, -1, -2, etc. Since the phase of a complex number " $I + jQ$ " is $\text{atan}(Q/I)$, the phase of " $1 + jK$ " is $\text{atan}(K)$. Likewise, the phase of " $1 - jK$ " = $\text{atan}(-K) = -\text{atan}(K)$. To add phases " $R = 1 + jK$ " is used; to subtract phases " $R = 1 - jK$ ". Since the real part of this, I_r , is equal to one, the table of equations can be simplified to add and subtract phases for the special case of CORDIC multiplications to:

To add a phase, multiply by $R = 1 + jK$:	$Ic' = Ic - K \cdot Qc = Ic - (2^{-L}) \cdot Qc$ $Qc' = Qc + K \cdot Ic = Qc + (2^{-L}) \cdot Ic$
To subtract a phase, multiply by $R = 1 - jK$:	$Ic' = Ic + K \cdot Qc = Ic + (2^{-L}) \cdot Qc$ $Qc' = Qc - K \cdot Ic = Qc - (2^{-L}) \cdot Ic$

Table 4. Add/subtract phases less than 90° .

Table 5 details the phases and magnitudes of each of these multiplier values, listing values of L , starting with 0, and shows the corresponding values of K , phase, magnitude, and CORDIC Gain. Each rotation has a magnitude greater than 1.0 for using rotations of the form " $1 + jK$ ", which is usually undesirable, but unimportant in the calculation of the phase of a vector. The CORDIC Gain column in the table is a cumulative magnitude calculated by multiplying the current magnitude by the previous magnitude. It converges to about 1.647; however, the actual CORDIC Gain depends on how many iterations are done.

L	$K = 2^{-L}$	$R = 1 + jK$	Phase of R in degrees = $\text{atan}(K)$	Magnitude of R	CORDIC Gain
0	1.0	$1 + j1.0$	45.00000	1.41421356	1.414213562
1	0.5	$1 + j0.5$	26.56505	1.11803399	1.581138830
2	0.25	$1 + j0.25$	14.03624	1.03077641	1.629800601
3	0.125	$1 + j0.125$	7.12502	1.00778222	1.642484066
4	0.0625	$1 + j0.0625$	3.57633	1.00195122	1.645688916
5	0.03125	$1 + j0.031250$	1.78991	1.00048816	1.646492279
6	0.015625	$1 + j0.015625$	0.89517	1.00012206	1.646693254
7	0.007813	$1 + j0.007813$	0.44761	1.00003052	1.646743507
...	...	...	...	...	...

Table 5. CORDIC Vectoring Flow (After [8]).

There are two operations to the CORDIC algorithm for trigonometric calculations:

1. Rotation – the vector is rotated by a specified angle; and
2. Vectoring – the vector is rotated to the x -axis while recording the angle required to make that rotation.

In order to calculate phase, the Rotation step is completed using the angle $\pm 90^\circ$. The objective is to rotate the vector to the right half of the complex plane so that the vector can subsequently be vectored to the positive x -axis. The sign of the Q data determines whether an addition or subtraction takes place. If the phase is positive, rotate by -90° , and if the phase is negative, rotate by $+90^\circ$. After the initial Rotation, CORDIC Vectoring as per Table 5 is executed, and in each addition/subtraction step, the actual number of degrees rotated is accumulated. After the requisite number of rotations to calculate a result with a desired maximum error, the phase of the complex number is the negative of the rotation required to bring it to a phase of zero. Consider a CORDIC Implementation example: given a complex number $z = a + bj$, for example $z_0 = -8 + 4j$, determine the phase ϕ .

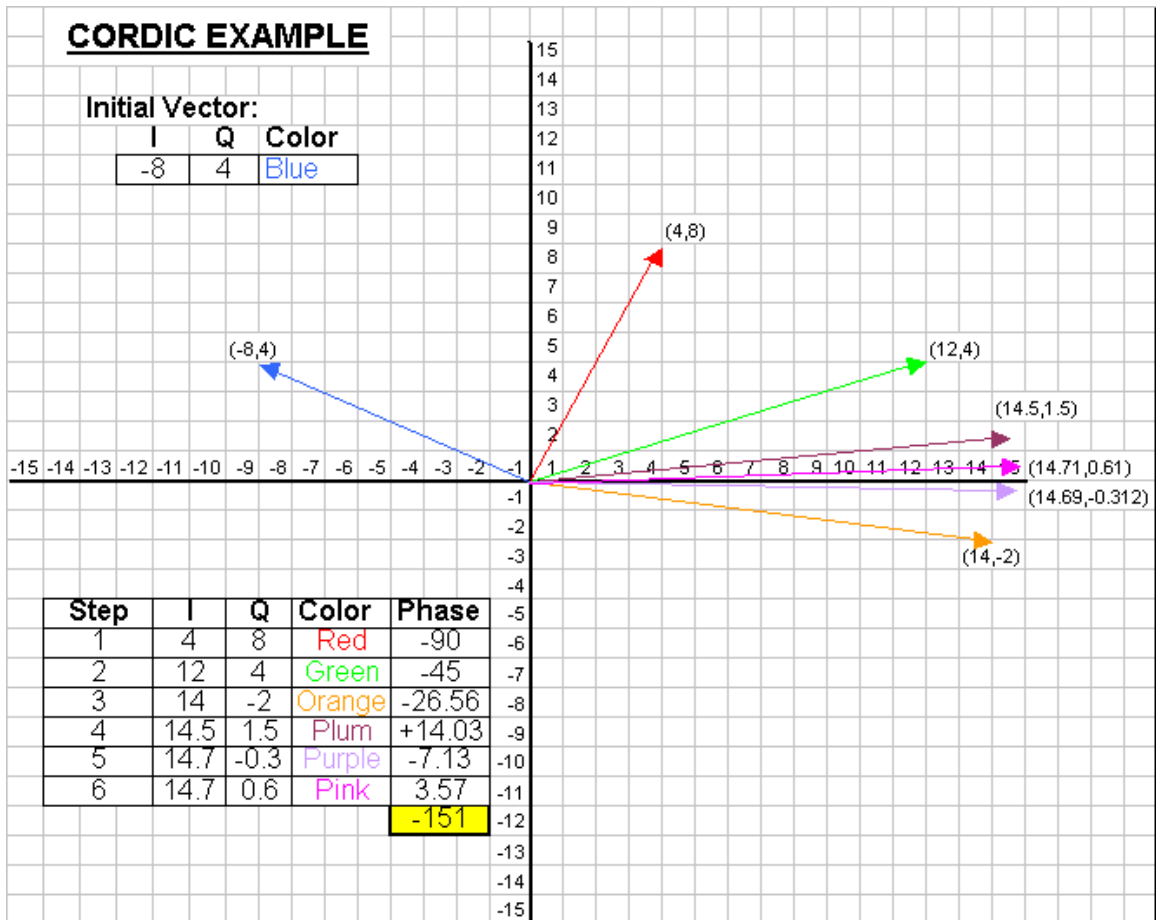


Figure 13. CORDIC Example: $z_0 = -8 + 4j$.

Beginning with the blue vector input from Figure 13, the following details the CORDIC steps:

1. The sign of the Q data is positive ($Q = 4$), so the Rotation step is -90° . The new vector in red is produced as per Table 3: negate I and swap. Store rotation of -90° . The new vector is $(4, 8)$.
2. Vector the new complex number, iteration $L = 0$. The sign of Q is positive ($Q = 8$) so subtract 45° to produce the green vector as per Table 4. Accumulate -45° , thus the phase equals: $-90^\circ + -45^\circ = -135^\circ$. New vector is $(12, 4)$.
3. Vector the new complex number, iteration $L = 1$. The sign of Q is positive ($Q = 4$) so subtract 26.56505° to produce the orange vector. New accumulated phase is -161.56505° . New vector is $(14, -2)$.

4. Vector the new complex number, iteration $L = 2$. The sign of Q is negative ($Q = -2$) so add 14.03624° to produce the plum vector. New accumulated phase is -147.52881° etc.

In tabular form, Table 6 continues the CORDIC to $L = 17$ iterations. The resulting angle of -153.435° must then be negated, giving the proper phase angle of 153.435° for an input complex number of $(-8,4)$.

There are three key salient points to note from Table 6. First, an eight-bit two's complement number has the range of values from -128 to $+127$. As the constant value of phase added or subtracted during each iteration has a decimal portion, the number of bits on Q and I must increase to hold these calculated values. For example, the I data value during each iteration continues to positively grow in magnitude, and depending on input values of Q and I , may exceed $+127$. Thus, more than eight integer bits are required to represent growing I values. Second, as the Q data value is rotated to zero, it becomes a fractional number and requires high decimal precision, vice integer precision. Otherwise the sign of the Q data may not be precise enough to properly determine the next iteration decision for addition or subtraction. Third, after the initial Rotation, the values of I lie on the right half of the complex plane and are always positive values, while the values of Q still change positive and negative depending on the value of phase added. Hence, I data bits larger than the MSB ($I/5$) are always a logical zero after the Rotation step.

I	Q	A/S	I Q degs	Cum. Angle L	k	Phase(k)	Mag(k)	Cum. Mag(k)
-8.000000	4.000000		153.435	0.000				
4.000000	8.000000	-1	63.435	-90.000		90.000000		
12.000000	4.000000	-1	18.435	-135.000	0	1.000000	1.414214	1.414213562
14.000000	-2.000000	-1	-8.130	-161.565	1	0.500000	26.56505	1.581138830
14.500000	1.500000	1	5.906	-147.529	2	0.250000	14.03624	1.629800601
14.687500	-0.312500	-1	-1.219	-154.654	3	0.125000	7.12502	1.642484066
14.707031	0.605469	1	2.357	-151.077	4	0.062500	3.57633	1.645688916
14.725952	0.145874	-1	0.568	-152.867	5	0.031250	1.78991	1.646492279
14.728231	-0.084219	-1	-0.328	-153.763	6	0.015625	0.89517	1.646693254
14.728889	0.030845	1	0.120	-153.315	7	0.007813	0.44761	1.646743507
14.729010	-0.026689	-1	-0.104	-153.539	8	0.003906	0.22381	1.646756070
14.729062	0.002078	1	0.008	-153.427	9	0.001953	0.11191	1.646759211
14.729064	-0.012306	-1	-0.048	-153.483	10	0.000977	0.05595	1.646759996
14.729070	-0.005114	1	-0.020	-153.455	11	0.000488	0.02798	1.646760193
14.729071	-0.001518	1	-0.006	-153.441	12	0.000244	0.01399	1.646760242
14.729071	0.000280	1	0.001	-153.434	13	0.000122	0.00699	1.646760254
14.729071	-0.000619	-1	-0.002	-153.437	14	0.000061	0.00350	1.646760257
14.729072	-0.000169	1	-0.001	-153.436	15	0.000031	0.00175	1.646760258
14.729072	0.000055	1	0.000	-153.435	16	0.000015	0.00087	1.646760258
14.729072	-0.000057	-1	0.000	-153.435	17	0.000008	0.00044	1.646760258
Magnitude:								
8.94427191				Phase		Phase Sum:		Normalizer:
				153.435		189.88253		0.607252935

Table 6. Detailed Calculation Flow of CORDIC Example (After [8]).

Figure 14 shows MATLAB calculations of the CORDIC implementation of the amplitude to phase conversion, to nine iteration accuracy with eight-bit Q and I inputs, while Figure 15 shows the same results quantized to 32 bits.

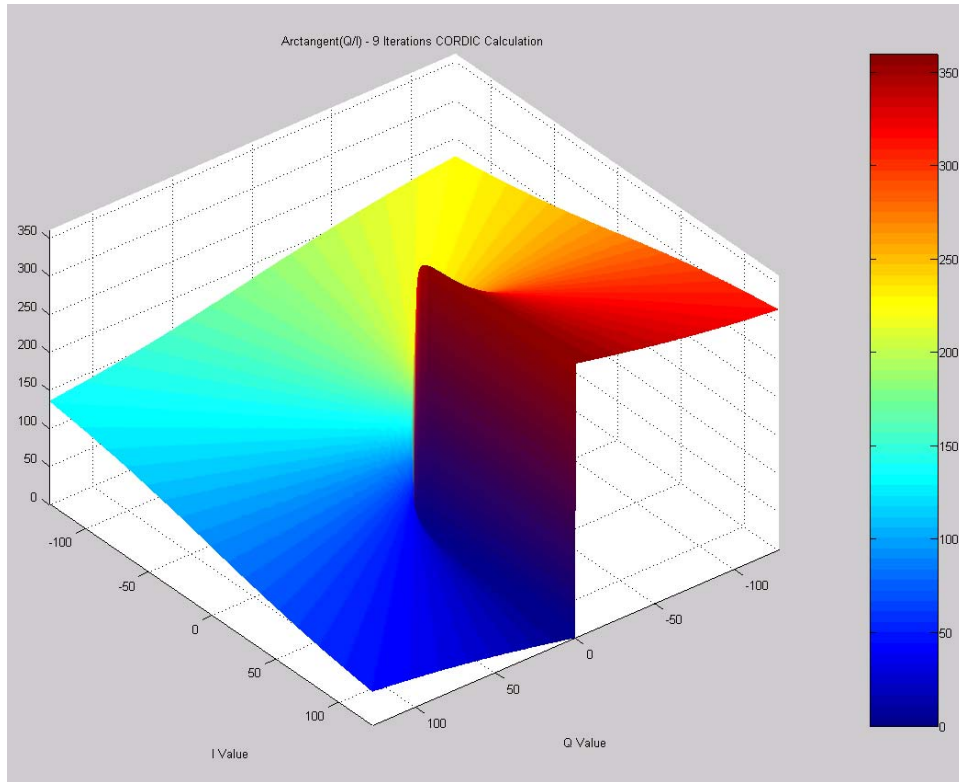


Figure 14. CORDIC Arctangent(Q/I).

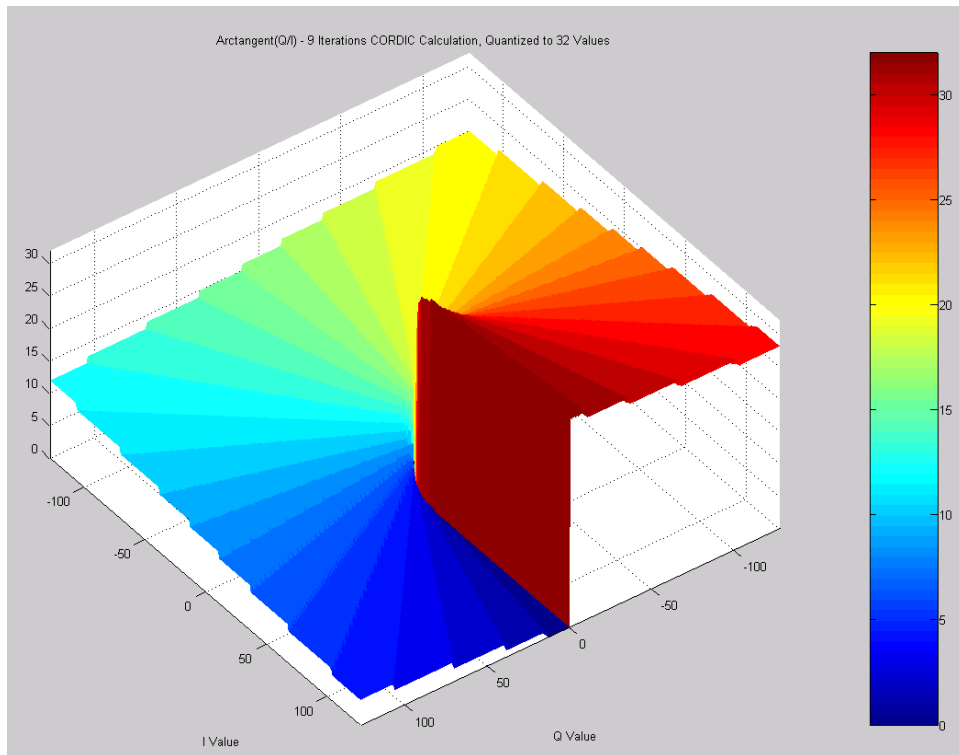


Figure 15. CORDIC Arctangent(Q/I) Quantized to 32 Values.

The CORDIC method offers a hardware-simple, pipeline-capable, low-transistor count hardware implementation. It can achieve any desired accuracy and avoids multiplication by using shifts by powers of two. For this reason, the CORDIC method was chosen to implement the amplitude to phase conversion.

D. CORDIC HARDWARE IMPEMENTATION

There are three primary methods for the hardware implementation of the CORDIC algorithm. These are [After 9]:

- Bit Parallel Iterative CORDIC. Each branch consists of an adder-subtractor combination, a shift unit and a register to buffer the output. A finite-state machine is needed to control the multiplexers, the shift distance and the addressing of the constant values.

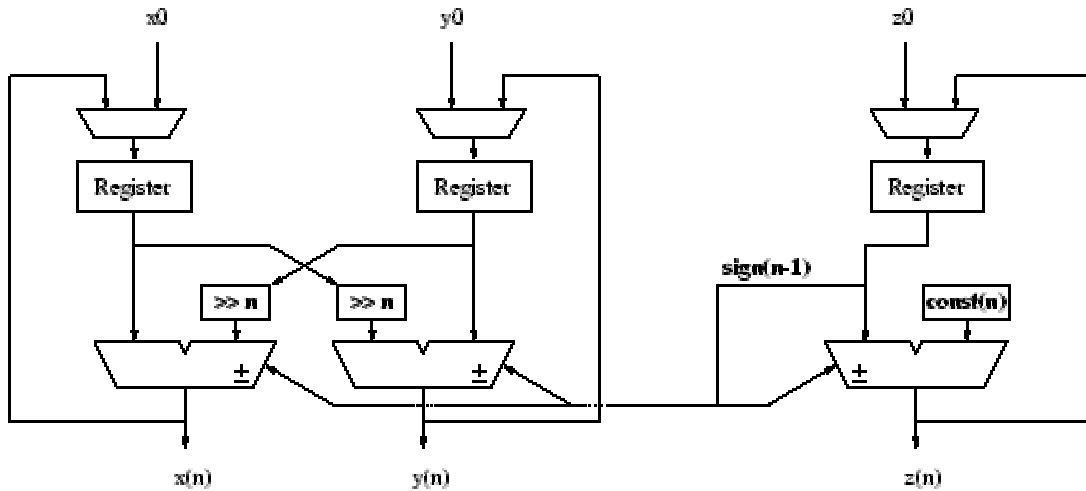


Figure 16. Bit Parallel Iterative CORDIC (From [9]).

For each input vector, it takes n clocks to achieve n iterations, assuming that no additional pipelining is required. Thus this is not conducive to a high speed implementation when, on each clock, new data is presented as inputs.

- Bit Serial Iterative CORDIC. Bit-serial means only one bit is processed at a time and hence the cross connections become one bit-wide data paths. The throughput becomes a function of [9]:

$$\frac{clock_rate}{number_of_iterations * word_width}$$

which is not a fast implementation method.

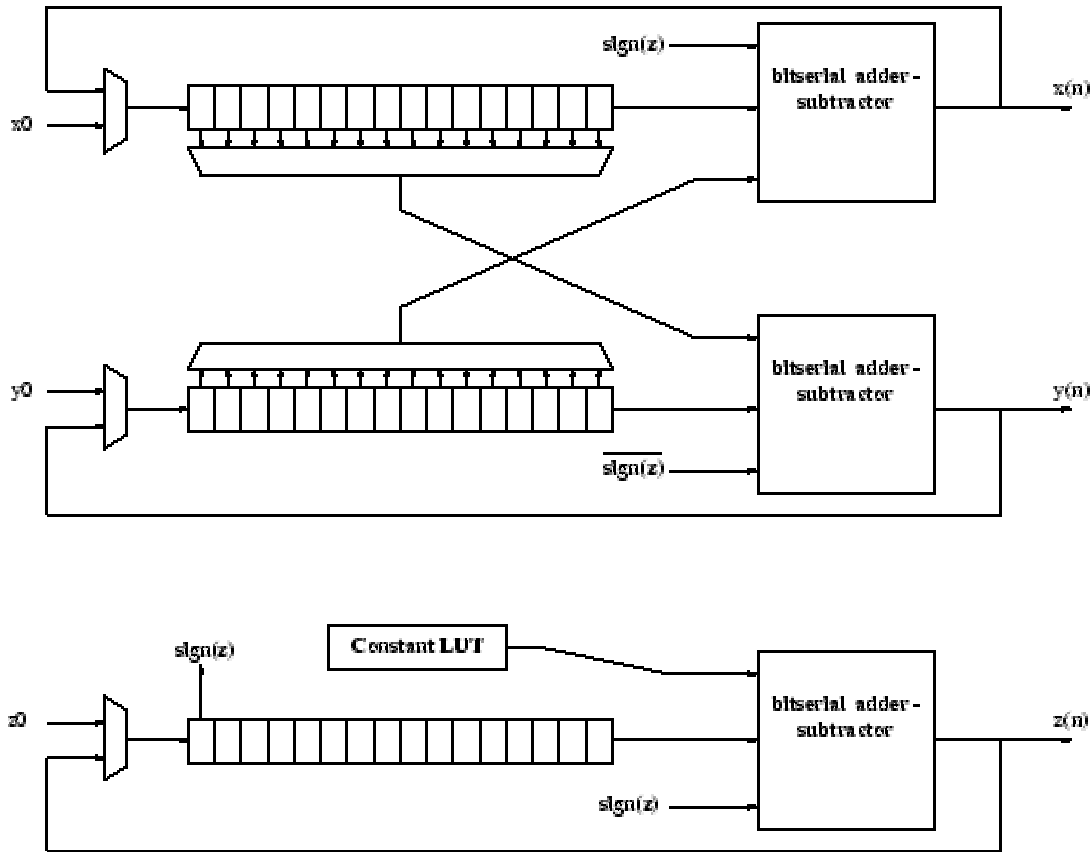


Figure 17. Bit Serial Iterative CORDIC (From [9]).

- **Bit Parallel Unrolled CORDIC.** Instead of buffering the output of one iteration and using the same resources again, one can simply cascade the iterative CORDIC, which means rebuilding the basic CORDIC structure for each iteration. Consequently, the output of one stage is the input of the next one and in the face of separate stages two simplifications become possible. First, the shift operations for each step can be performed by wiring the connections between stages appropriately. Second, there is no need for changing constant phase values and they can therefore be hardwired. The purely unrolled design only consists of combinatorial components and computes one value per clock cycle. Input values find their path through the architecture on their own and do not need to be controlled [9].

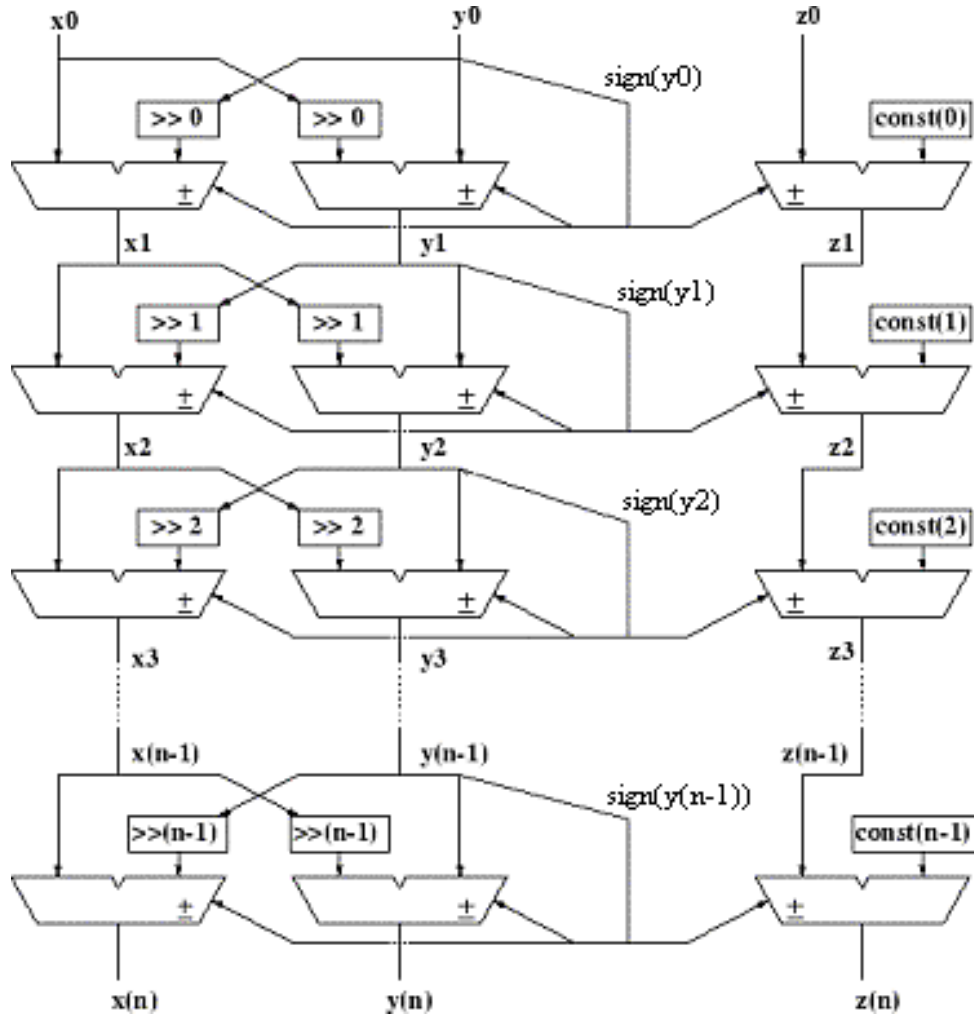


Figure 18. Bit Parallel Unrolled CORDIC [After 9].

Of the three methods, the Bit Parallel Unrolled implementation, shown in Figure 18, provides the highest throughput speed and simplicity of design, albeit at the expense of replicated hardware iteration stages. Therefore, this CORDIC hardware method was implemented. A decision made outside the research of this thesis was the accuracy of the amplitude-to-phase conversion required. MATLAB simulations were run by a fellow student, Fernando LeDantec, to determine the acceptable error. It was decided that six iterations were sufficient for providing a result to the DIS architecture. Table 7 shows the flow of the previous example using a six-iteration CORDIC hardware, including the binary values at each stage. The highlighted number is the CORDIC produced result in degrees, and the red numbers show the input values and corresponding binary five-bit output.

Iteration Level	I Decimal	Integer I Binary	Q Decimal	Integer Q Binary	Add/Subtract	I Q degs	Cum. Angle	Integer Cum. Angle Binary	L	k	Phase(k)
Input Data	-8.000000	1111111000	4.000000	100		153.435	0.000	0			
Rotation	4.000000	100	8.000000	1000	-1	63.435	-90.000	1110100110			90.00000
45	12.000000	1100	4.000000	100	-1	18.435	-135.000	1101111001	0	1.000000	45.00000
26	14.000000	1110	-2.000000	1111111110	-1	-8.130	-161.565	1101011111	1	0.500000	26.56505
14	14.500000	1110	1.500000	1	1	5.906	-147.529	1101101101	2	0.250000	14.03624
7	14.687500	1110	-0.312500	0	-1	-1.219	-154.654	1101100110	3	0.125000	7.12502
3	14.707031	1110	0.605469	0	1	2.357	-151.077	1101101001	4	0.062500	3.57633
1	14.725952	1110	0.145874	0	-1	0.568	-152.867	1101101000	5	0.031250	1.78991
CORDIC Phase Actual Phase = $\pi/4$, Truncate If Negative, 5 Bit Output in Phase + 360 5 Bit Output in Binary											
-152.867	152		152	13	01101						

Table 7. Detailed Hardware Flow of a six-iteration CORDIC Implementation.

III. SCHEMATIC DESIGN OF THE Q/I PHASE CONVERTER

A. HIERARCHICAL SCHEMATIC DESIGN OVERVIEW

The circuit was designed using Tanner Research software, detailed in Appendix D, for MOSIS fabrication using the TSMC CL018 process. This CMOS process has six metal interconnect layers and one polysilicon layer. The process is for 1.8-volt applications and has a thick oxide layer for making 3.3-volt transistors. The 0.18-micron sized CMOS logic process uses epitaxial wafers and possesses the characteristics of fabricating: silicide blocks, thick gate oxide, electrostatic discharge (3.3 V), NT_N, deep n_well, ThickTopMetal (inductor), and MiM [10]. This chapter details: low level transistor and example gate electrical characteristics; secondary sub-circuits that are important in the amplitude to phase conversion circuit design; number system format; CORDIC Rotation and Vector level implementation; 9-to-5-bit conversion; and circuit verification. Appendix B details the MOSIS TSMC 0.18-micron Field Effect Transistor Parameters.

B. TRANSISTORS

1. N-FET

The N-channel Field Effect Transistor (N-FET) is modeled in S-Edit as a symbol and has no schematic representation. As it is a lowest level circuit building block, it is declared via a property definition for T-SPICE netlist extraction. The T-SPICE definition of the N-FET:

```
M#  %[1]  %[2]  %[2]  %{B}  CMOSN  W=5*lambda  L=2*lambda
AS=5.5*lambda*5*lambda  AD=5.5*lambda*5*lambda  PS=5*lambda  +
5.5*lambda + 5*lambda + 5.5*lambda PD=5*lambda + 5*lambda + 5.5*lambda
+ 5.5*lambda
```

Salient points are the transistor size of width of five lambda and length of two lambda, where lambda is 0.09×10^{-6} meters. These sizes can be changed to make different sized FETs for different transistor characteristics, most importantly, gain and current

sourcing/sinking capabilities. A FET of width 0.45-microns and length of 0.18-microns is the minimum sized possible for the CL018 fabrication process. The other parameters defined for the FET are areas for capacitance calculations. Figure 19 shows the S-Edit symbol, detailing the Gate (G), Source (S), Drain (D), and substrate connection (B). It should be noted that all ports are unidirectional, i.e., source to drain current flow, because this is important for VHDL extraction and simulation.

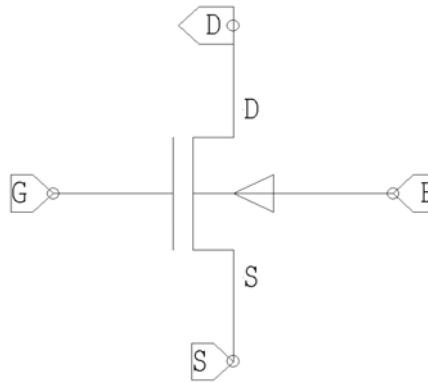


Figure 19. N-FET.

2. P-FET

Similarly, the P-channel FET is a property and is defined:

```
M#  %[2]  %G  %S  %B  CMOSF  W=5*lambda  L=2*lambda
AS=5*lambda*5.5*lambda  AD=5*lambda*5.5*lambda  PS=5*lambda  +
5*lambda + 5.5*lambda + 5.5*lambda PD=5*lambda + 5*lambda + 5.5*lambda
+ 5.5*lambda
```

The P-FET symbol:

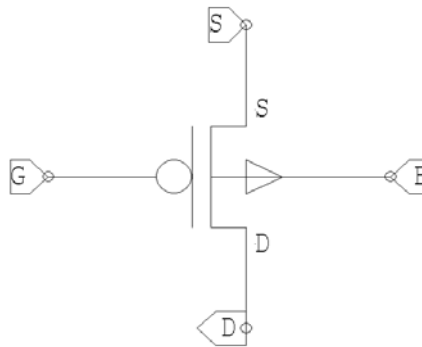


Figure 20. P-FET.

3. N/P-FET Current and Voltage – Drain to Source

Figure 21 shows the I_{DS} vs. V_{DS} characteristic curves for the N-FET and, similarly, for the P-FET in Figure 22.

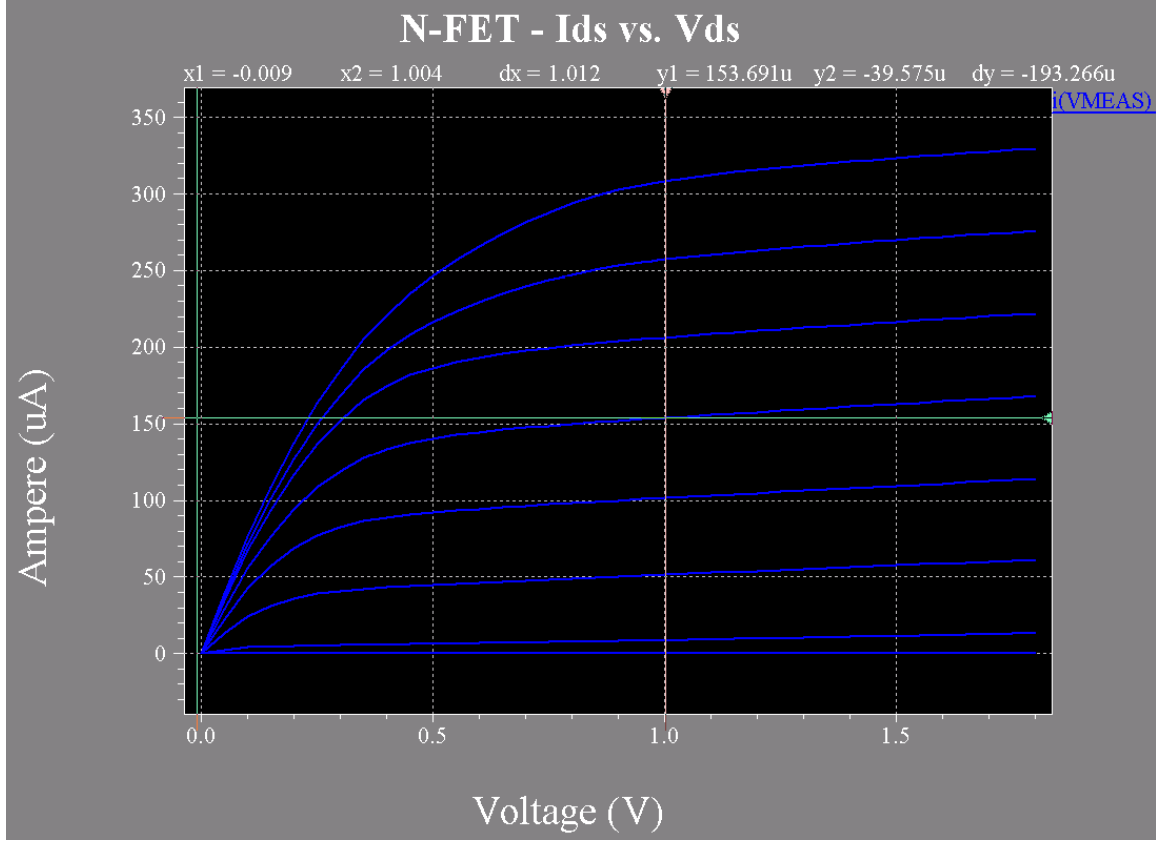


Figure 21. N-FET I_{DS} vs. V_{DS} .

The MOSIS Parametric test results provide a threshold voltage $V_{TN} = V_{TP} = 0.359V$. The applicable operating characteristics at the hatched data points on the above and below figures is summarized as Table 8:

PFET			NFET		
V_{DS}	V_{GS}	$I_{DS} \mu A$	V_{DS}	V_{GS}	$I_{DS} \mu A$
-1.000	-1.2	-51.793	1.004	1.2	153.691

Table 8. N/P-FET Operating Point Data.

The current from Drain to Source of the FET can be first order equated as [11]:

$$I_{DS} = \frac{\beta(V_{GS} - V_T)^2}{2} \quad (3.1)$$

where:

- β is the transistor gain factor and is used in the calculation of transconductance,
- V_{GS} is the voltage from Gate to Source, and
- V_T is the threshold voltage.

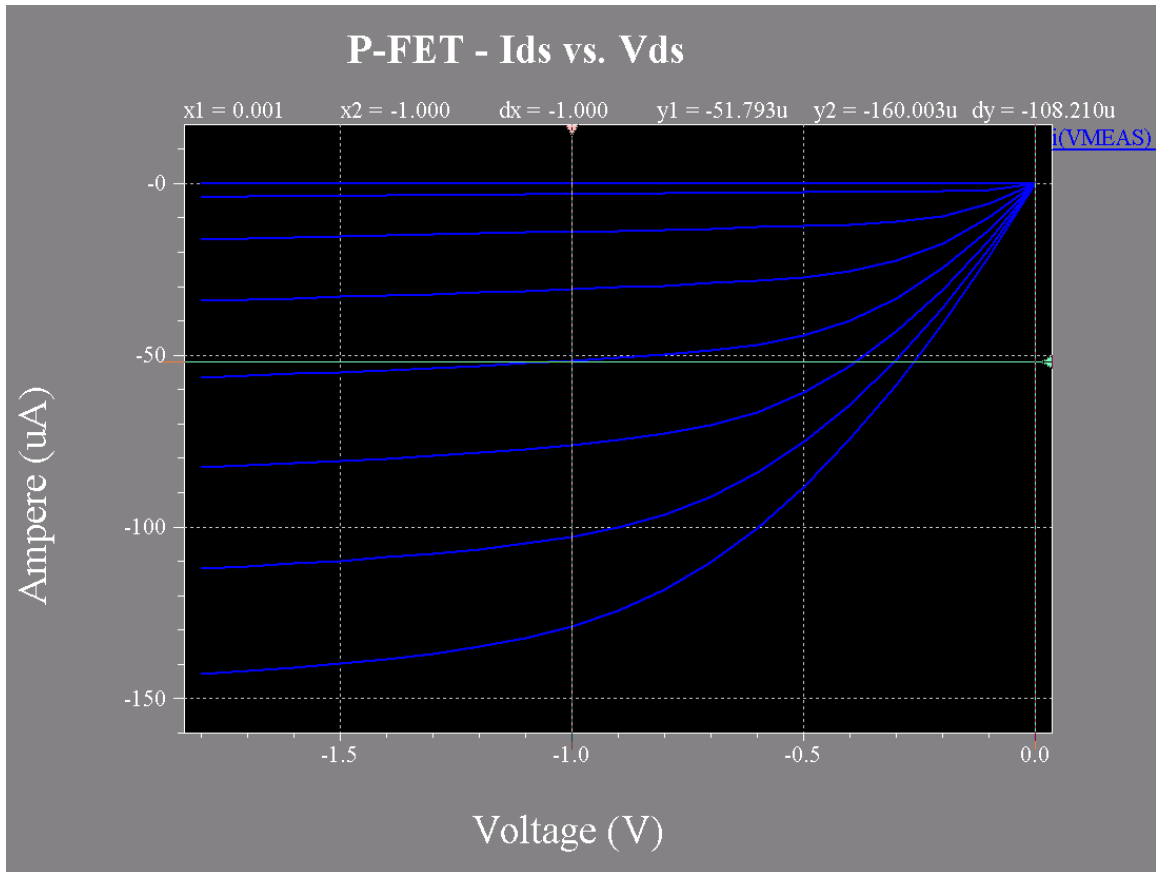


Figure 22. P-FET I_{DS} vs. V_{DS} .

Calculations of the gains is therefore:

$$\beta_N = \frac{2I_{DS}}{(V_{GS} - V_T)^2} = \frac{2 \cdot 153.691 \times 10^{-3}}{(1.2 - 0.359)^2} = 4.350 \times 10^{-4} \text{ A/V}^2$$

$$\beta_P = \frac{2I_{DS}}{(V_{GS} - V_T)^2} = \frac{2 \cdot 51.793 \times 10^{-6}}{(1.2 - 0.359)^2} = 1.465 \times 10^{-4} \text{ A/V}^2.$$

C. LOGIC GATES

This section provides an overview of the most basic CMOS gates, their noise margins, transfer functions, and operating characteristics. It is not all-inclusive, but is provided to show some of the electrical properties and characteristics that compose the amplitude-to-phase converter design.

1. Inverter

An N-FET and P-FET combined in series form an inverter when their gates and drains are tied together, and is one of the most basic CMOS logic cells. Figure 23 shows the schematic (bottom right) and symbol (top left) for the inverter:

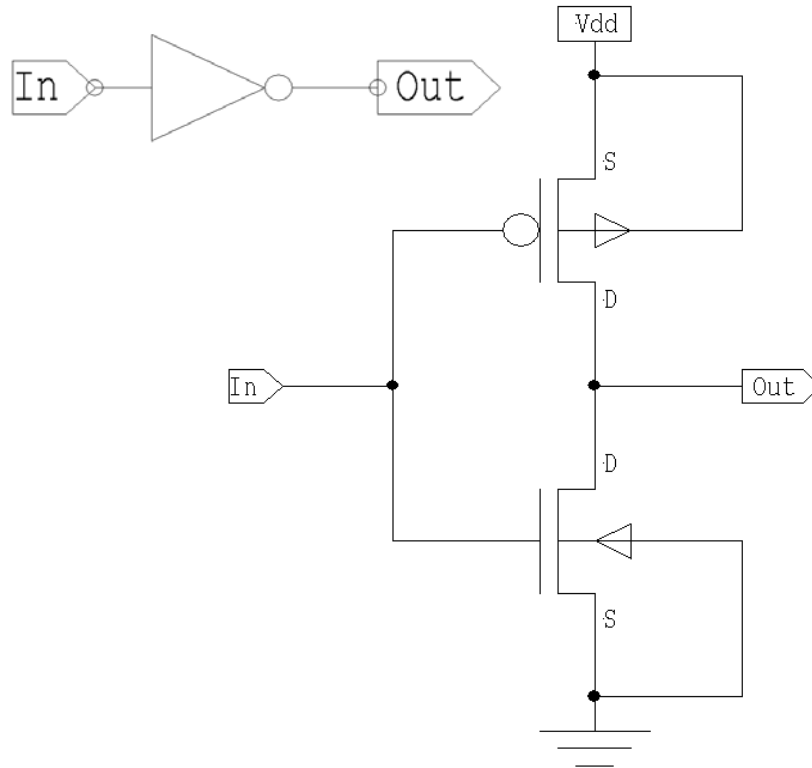


Figure 23. Inverter Symbol (top left) and Schematic (right).

Logically, an inverter provides an output 180° phase shift of the input signal and thereby provides the logic transfer function NOT, or complement. In CMOS inverters, the gate threshold voltage is dependant on the β_N/β_P ratio. A ratio of one allows a capacitive load

to charge and discharge in equal times by providing equal current-source and –sink capabilities as it relates to the mobility of holes being less than the mobility of electrons. It also affects the allowable noise voltage on the input of a gate so that the output will not be affected [After 11]. The ratio may be calculated to be:

$$\frac{\beta_N}{\beta_P} = 2.969.$$

Another important design factor is that the gain is directly proportional to the transistor width and length. For N-FETs [11]:

$$\beta_N = \frac{\mu_N \epsilon}{t_{ox}} \left(\frac{W_N}{L_N} \right) \quad (3.2)$$

where:

- μ is the effective surface mobility of the carriers in the channel,
- ϵ is the permittivity of the gate insulator,
- t_{ox} is the thickness of the gate insulator,
- W_N is the width of the transistor channel, and
- L_N is the length of the channel.

Therefore, a three times sized P-FET would (approximately) provide a β_N/β_P ratio of one, but would also require three times more layout area. A decision was made to implement all lower-level cells using minimum sized P-FETs to conserve layout, rather than to equate noise margins. Figure 24 shows three different β_N/β_P ratios for differently sized P-FETs where the NFET size is five lambda. Notice that the three-to-one sized P-FET to N-FET (in yellow, β_N/β_P ratio of one) transitions through the center voltage of 0.9V. The voltage out vs. voltage in plot characteristics is used to calculate the gates noise margins.

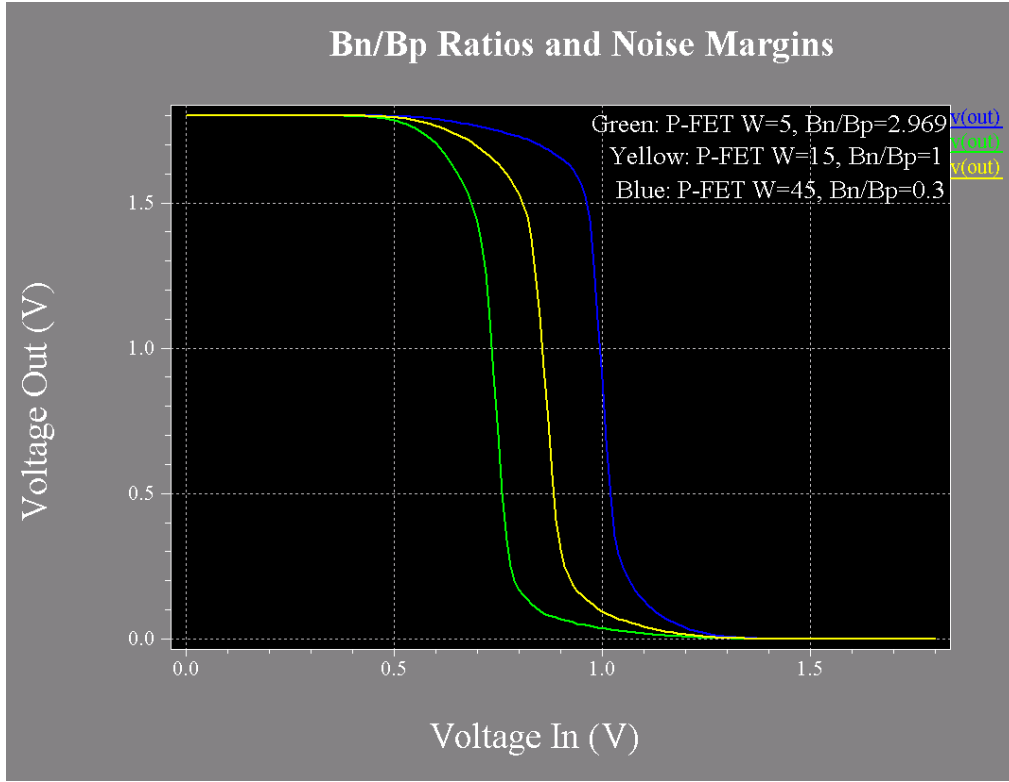


Figure 24. β_N/β_P Ratios and Inverter Noise Margins.

Noise margins low and high are determined via Equations 3.3 and 3.4 respectively [11]:

$$NM_L = |V_{IL_{MAX}} - V_{OL_{MAX}}| \quad (3.3)$$

$$NM_H = |V_{OH_{Min}} - V_{IH_{Min}}| \quad (3.4)$$

where:

- NM_L – noise margin low,
- NM_H – noise margin high,
- $V_{IL_{MAX}}$ – voltage input low maximum,
- $V_{OL_{MAX}}$ – voltage output low maximum,
- $V_{OH_{MIN}}$ – voltage output high minimum, and
- $V_{IH_{MIN}}$ – voltage input high minimum.

The data from Figure 24 was imported into MATLAB and the program code listed at Appendix A was used to calculate the noise margins. Results are listed in Table 9.

Voltage	Value	Units	Noise Margins	Value	Units
V_{OHMIN}	1.73	V	NM_H	0.86	V
V_{IHMIN}	0.87	V			
V_{ILMAX}	0.572	V	NM_L	0.49	V
V_{OLMAX}	0.082	V			

Table 9. Inverter Noise Margins.

It should be noted that the noise margins are uneven because the β_N/β_P ratio equals 2.969, the green plot from Figure 24. Values determined and displayed in Table 10 were determined using T-SPICE netlists where the circuit had an input shaping circuit of four inverters and the inverter test circuit possessing a load of seven other inverters.

Name	Value	Units
T_R	0.290	ns
T_F	0.129	ns
T_{PDLH}	0.172	ns
T_{PDHL}	0.063	ns
I_{PEAK}	-133.573	μA
Voltage	1.8	V
P_{PEAK}	240.43	μW

Table 10. Inverter Electrical Parameters.

The figures supporting the above table values are listed as Figures 25 through 28.

Applicable terms are [11]:

- T_R – rise time, the time for a waveform to rise from 10% to 90% of its steady-state value,
- T_F – fall time, the time for a waveform to fall from 90% to 10% of its steady-state value,
- T_{PDLH} – time of propagation delay low to high, the time difference between input transition (50%) and the 50% output level,
- T_{PDHL} – time of propagation delay high to low, the time difference between input transition (50%) and the 50% output level,
- I_{PEAK} – peak current draw, and
- P_{PEAK} – peak power draw.

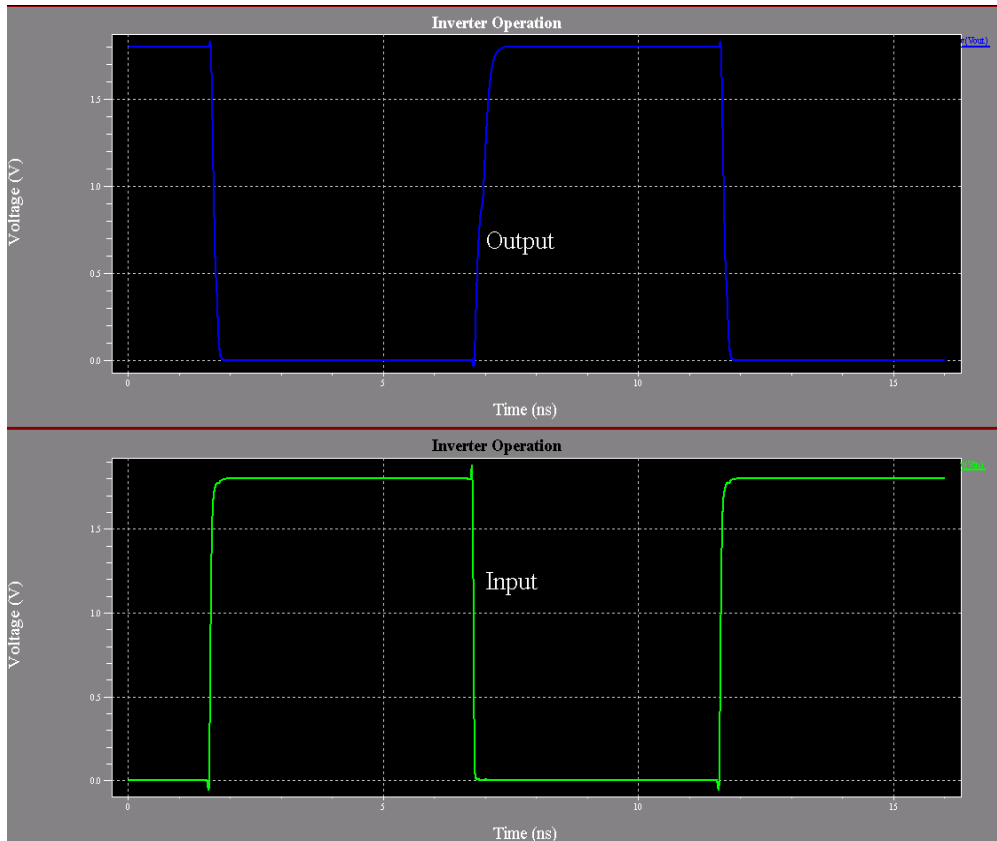


Figure 25. Inverter Proper Operation.

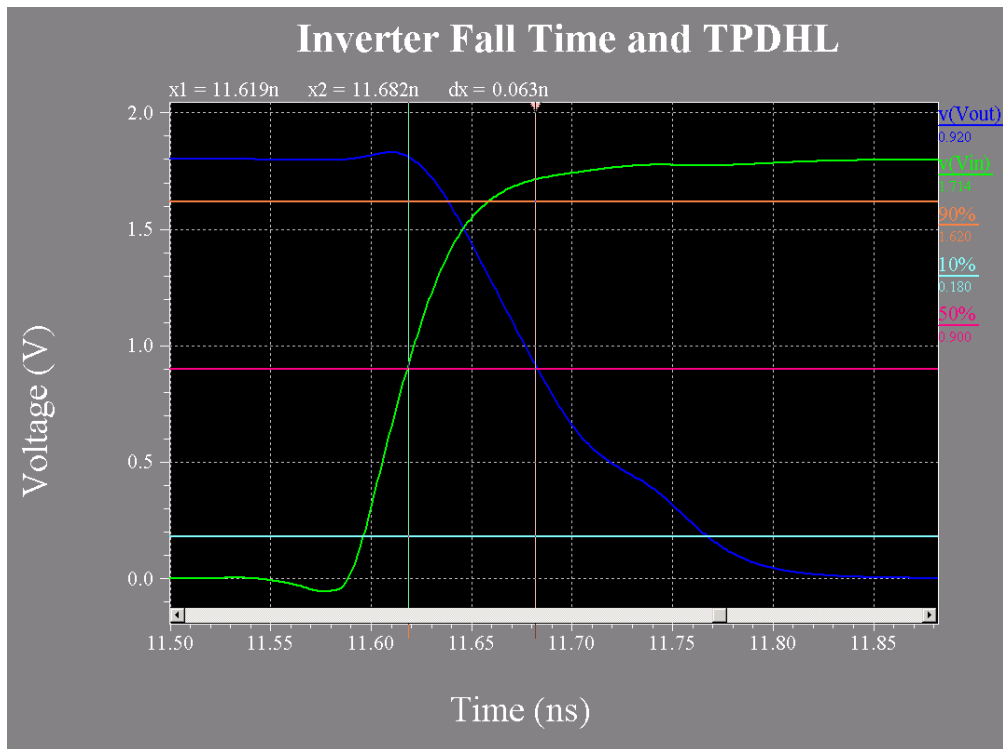


Figure 26. Inverter T_F and T_{PDHL} .

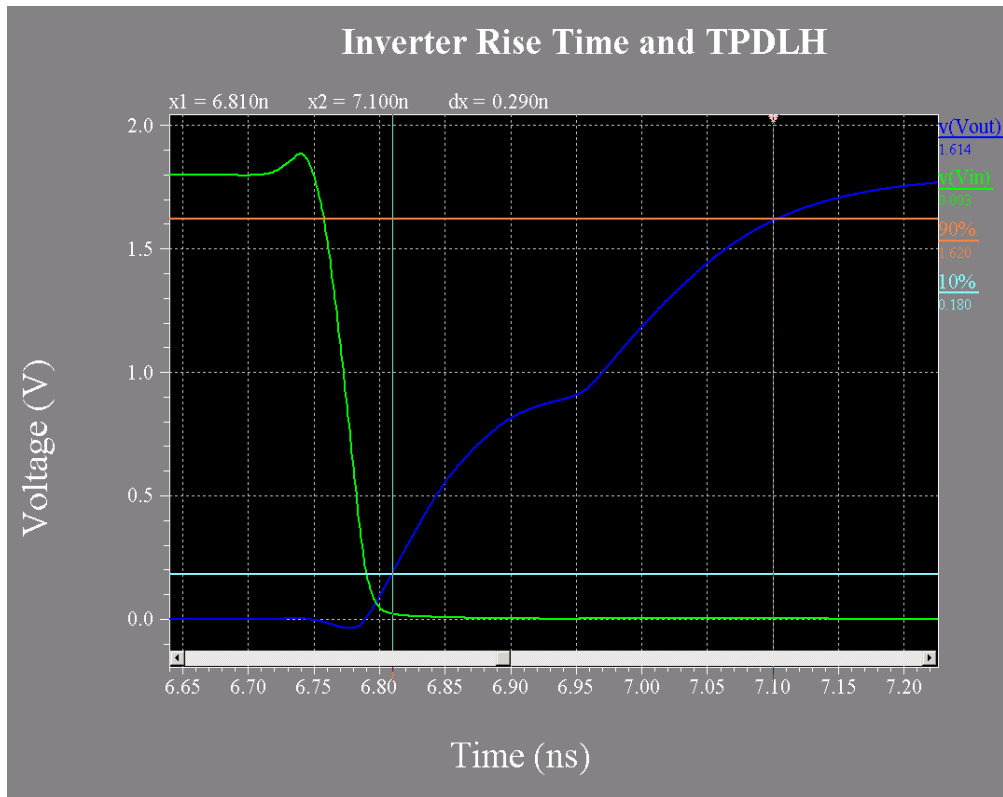


Figure 27. Inverter T_R and T_{PDLH} .

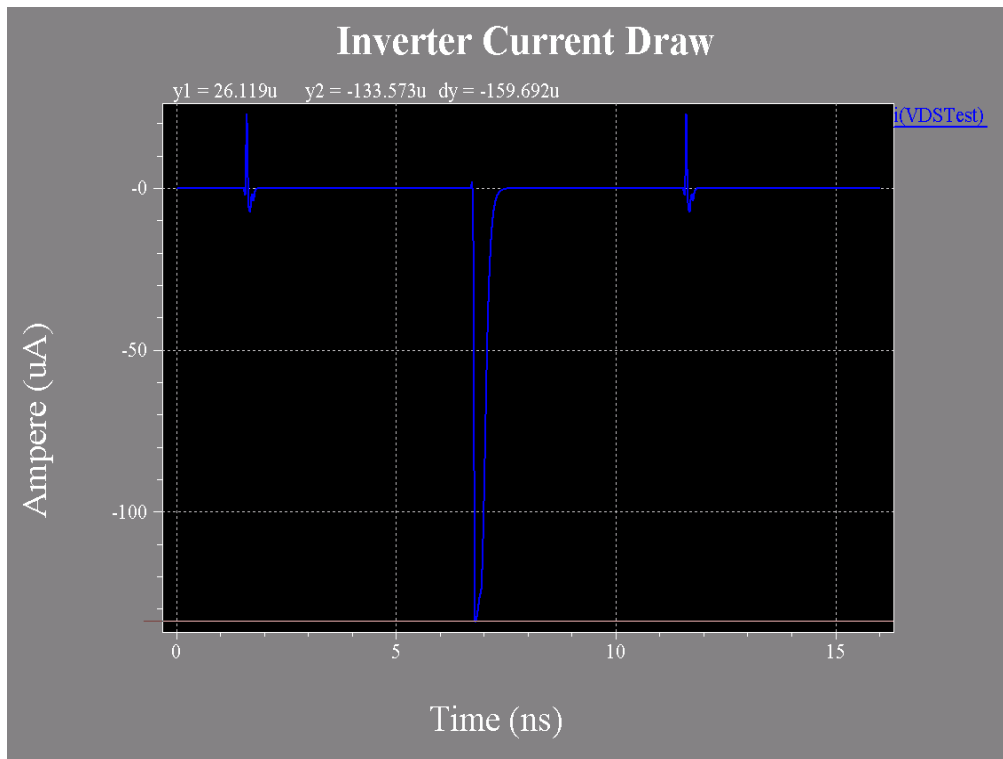


Figure 28. Inverter Current Draw.

The peak power drawn by an inverter can be calculated using the equation:

$$\text{Power} = \text{Voltage} * \text{Current} \quad (3.5)$$

which yields 240.34 μW .

2. Pass Gates

Combining an N-FET and P-FET in parallel forms the pass gate, also known as a complementary switch or transmission gate. This circuit acceptably passes both a logic '0' and logic '1' depending upon the *CON* and $\sim\text{CON}$ signals. The pass gate is another basic CMOS logic circuit that can be used to build higher order circuits such as the Exclusive-OR and Exclusive-NOR gates, registers, etc. Figure 29 shows the pass gate symbol and circuit, while Table 11 details T-SPICE circuit parameters using a three-inverter output load. The pass gate is a bi-directional structure, but has been updated in the circuit schematic to have uni-directional logic transmission for VHDL definition extraction.

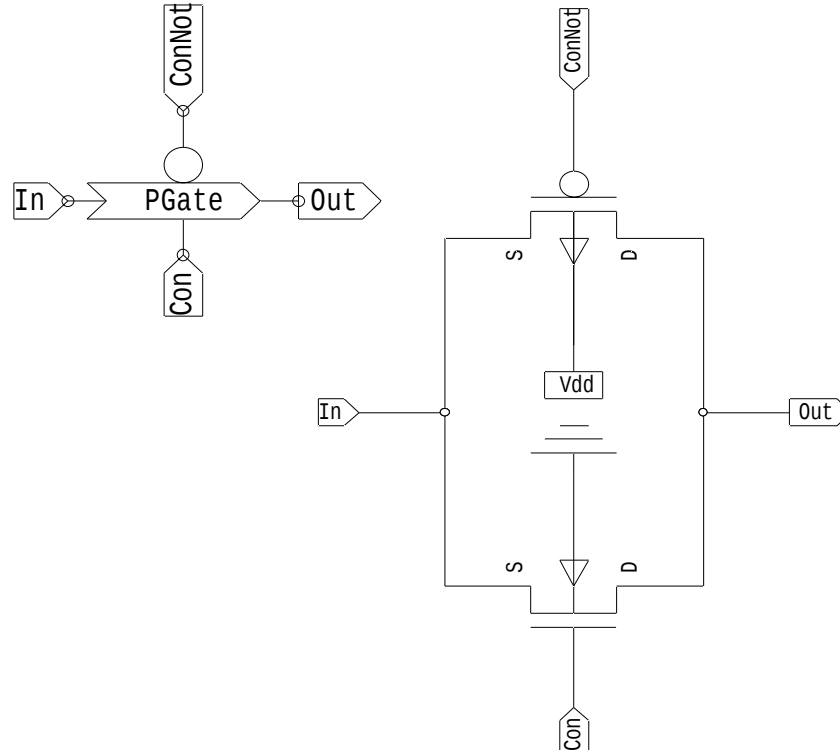


Figure 29. Pass Gate Symbol (top left) and Schematic (right).

Name	Value	Units
T_R	0.278	ns
T_F	0.106	ns
T_{PDLH}	0.051	ns
T_{PDHL}	0.044	ns
I_{PEAK}	-36.41	μA
$Voltage$	1.8	V
P_{PEAK}	65.5	μW

Table 11. Pass Gate Electrical Characteristics.

3. Buffer

A buffer is a non-inverting gate formed by combining two inverters in series such that it restores output voltage levels to their peak levels. It does not perform any logic transfer function itself, as the output is the equal to the input. The symbol and circuit are shown as Figure 30:

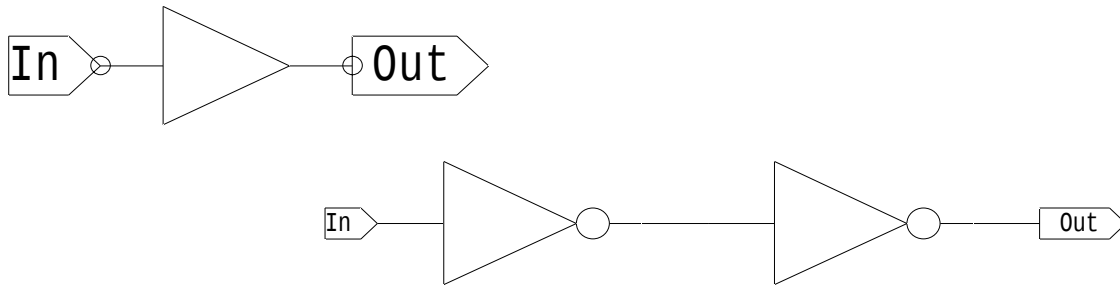


Figure 30. Buffer Symbol (top left) and Schematic (right).

Tables 12 and 13 detail buffer noise margins and electrical parameters:

			Noise		
Voltage	Value	Units	Margins	Value	Units
V_{OHMIN}	1.798	V	NM_H	1.075	V
V_{IHMIN}	0.723	V			
V_{ILMAX}	0.765	V	NM_L	0.759	V
V_{OLMAX}	0.006	V			

Table 12. Buffer Noise Margins.

Name	Value	Units
T_R	0.183	ns
T_F	0.084	ns
T_{PDLH}	0.118	ns
T_{PDHL}	0.087	ns
I_{PEAK}	-133.17	μA
$Voltage$	1.8	V
P_{PEAK}	240	μW

Table 13. Buffer Electrical Characteristics.

4. XOR

The exclusive-OR (XOR) gate has the logical transfer function output of x or y but not both, where x and y are gate inputs:

Input X	Input Y	Output Z
0	0	0
0	1	1
1	0	1
1	1	0

Table 14. XOR Truth Table.

The XOR gate is built using pass gates, and therefore, has a non-restoring output z value. The symbol and circuit are shown as Figure 31:

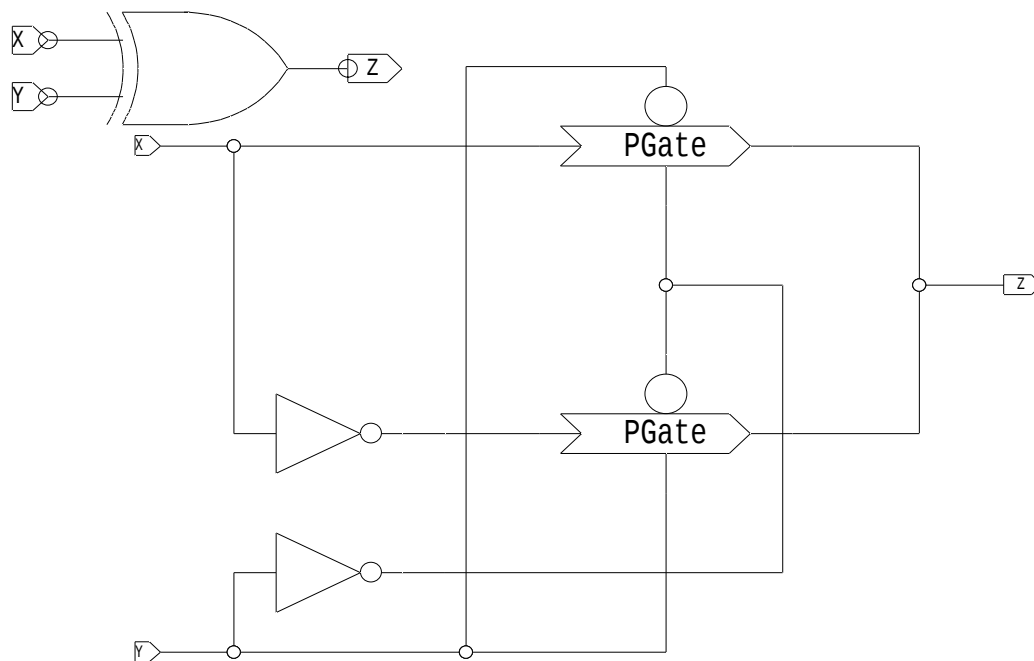


Figure 31. XOR Symbol (top left) and Schematic (right).

Table 15 details the XOR electrical operating characteristics:

Name	Value	Units
T_R	0.232	ns
T_F	0.081	ns
T_{PDLH}	0.028	ns
T_{PDHL}	0.061	ns
I_{PEAK}	-212.4	μA
$Voltage$	1.8	V
P_{PEAK}	382.2	μW

Table 15. XOR Electrical Characteristics.

5. NAND2

The two-input NAND gate symbol and circuit are shown as Figure 32:

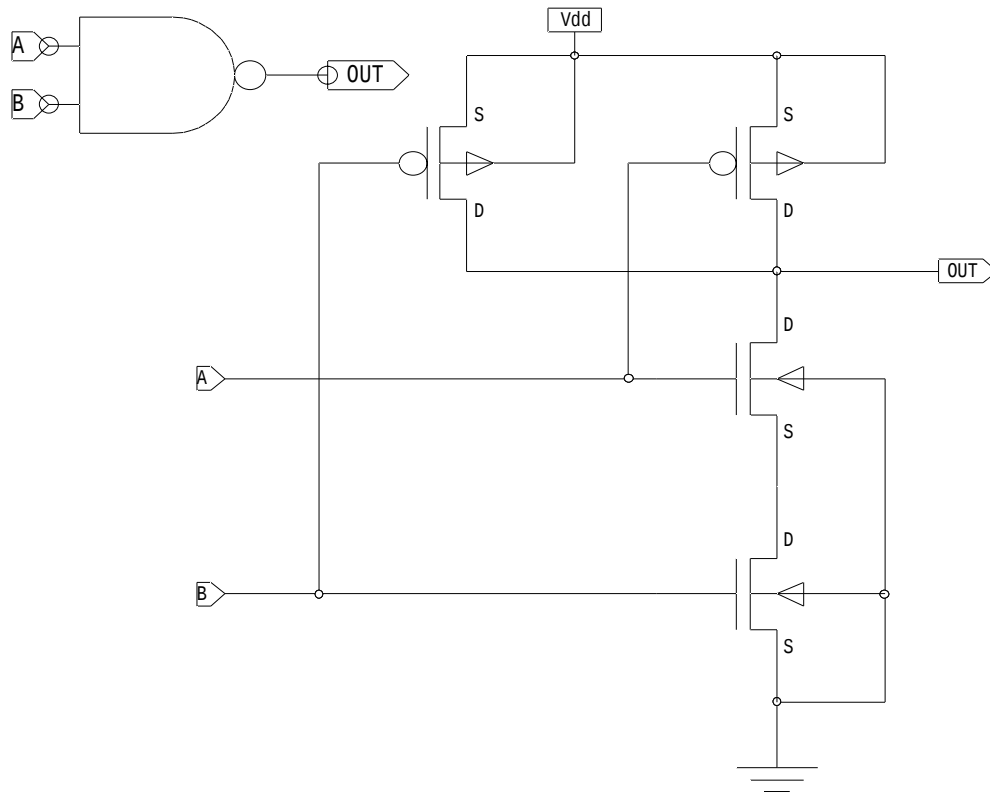


Figure 32. NAND2 Symbol (top left) and Schematic (right).

The NAND is the complement output of the AND function, and is directly realizable (built) in CMOS hardware by using two N-FETs in series and two P-FETs in parallel. The NAND2 truth table:

Input X	Input Y	Output Z
0	0	1
0	1	1
1	0	1
1	1	0

Table 16. NAND2 Truth Table.

Higher-level input NAND gates are built by consecutively adding N-FETs in series to the ground pull-down chain and P-FETs in parallel to the Vdd pull-up. NAND2 noise margins and operating characteristics are shown in Tables 17 and 18:

Voltage	Value	Units	Noise Margins	Value	Units
V_{OHMIN}	1.70	V	NM_H	0.684	V
V_{IHMIN}	1.016	V			
V_{ILMAX}	0.773	V	NM_L	0.63	V
V_{OLMAX}	0.143	V			

Table 17. NAND2 Noise Margins.

Name	Value	Units
T_R	0.062	ns
T_F	0.052	ns
T_{PDLH}	0.038	ns
T_{PDHL}	0.087	ns
I_{PEAK}	-225.4	μA
Voltage	1.8	V
P_{PEAK}	406	μW

Table 18. NAND2 Electrical Characteristics.

D. SECONDARY SUB-CIRCUITS

Previous work by students and a professor laid some of the foundation for S-Edit logic circuits. Those of primary importance and used in this circuit design of the amplitude-to-phase converter are registers [12] and the 16-bit Carry Look Ahead Adder (CLAH).

1. Registers

A one-bit register is built using pass gates to form a D-Master/Slave Flip-Flop (DMSFF) with corresponding control logic circuitry. Register loading can be controlled using the *Load (LD)* signal. When the *LD* signal is a logic '1', the register loads the *D* input value. When *LD* is a logic '0', the *Q* output of the register is fed back into the *D* input, thereby, holding the last state value. The rate at which the register samples and produces outputs is controlled via the *Clock (Clk)* signal. On each rising edge of the *Clk*,

the one-bit input D Master-Slave register samples the input data value and holds it via feedback. Figure 33 shows the circuit design of a one-bit DMSFF register with corresponding control signal logic in Figure 34. Table 19 details the operating characteristics.

Name	Value	Units
T_R	0.255	ns
T_F	0.158	ns
T_{PDLH}	0.343	ns
T_{PDHL}	0.345	ns
I_{PEAK}	-195.08	μA
<i>Voltage</i>	1.8	V
P_{PEAK}	351.1	μW

Table 19. DMSFF1 Characteristics.

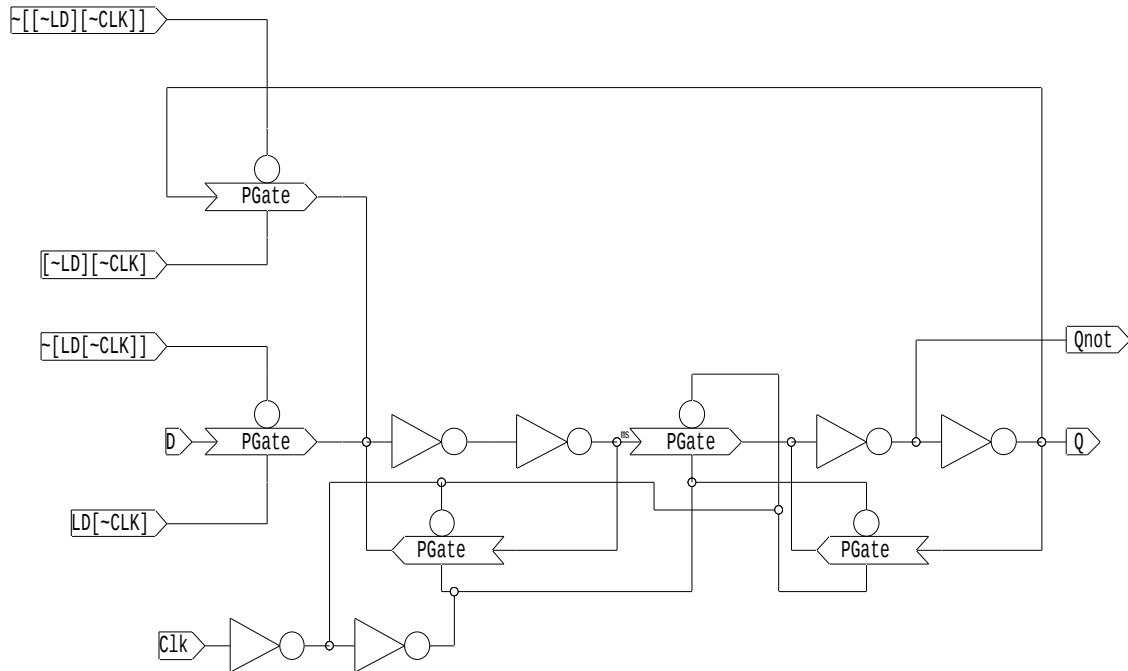


Figure 33. Register Schematic (From [12]).

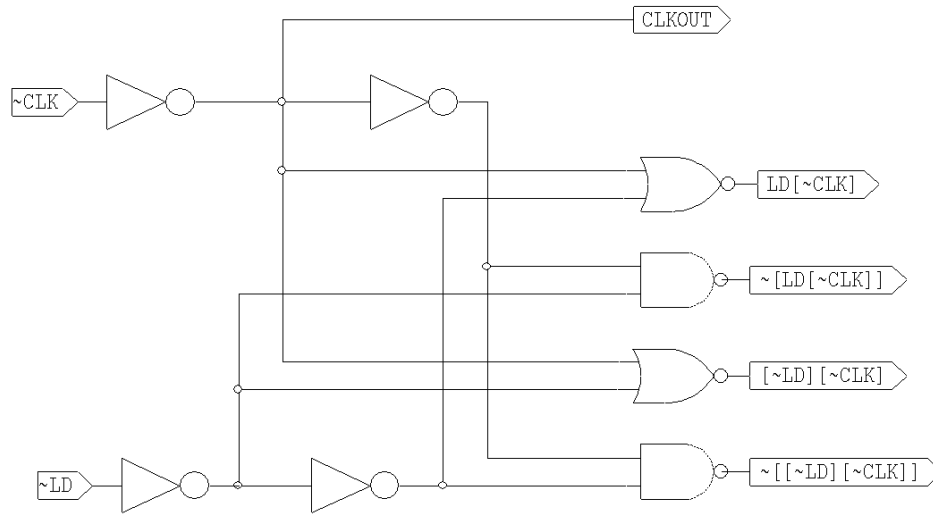


Figure 35. One-bit DMSFF Register Symbol.

2. RCA_PSVCHAIN

As previously stated, the method chosen to implement the phase conversion is the bit parallel unrolled CORDIC. Each level of iteration is a stand alone functional block, and is pipeline registered to ensure clock speed operation of 700 MHz. The *Phase Signal Valid (PSV)* is a control signal that specifies when the output is a valid result. As detailed later on, there are sixteen pipeline registers in the amplitude to phase conversion circuit and, therefore, there is pipeline latency of sixteen clock periods from input data until the correct output is produced. On the first clock signal to the circuit, values of Q and I are loaded into the first input stage register. On the *next clock signal*, the output register of that stage produces output values to the next stage register. Therefore, this “pipeline startup” and sixteen clock periods of delay to see the correct output must be accounted for. The amplitude-to-phase conversion circuit, therefore, has a register chain of sixteen one-bit registers to follow the Q and I data through the CORDIC iterations and declare

when the output is valid. Table 20 shows the operation of the register and the corresponding latency. The ‘X’ is “don’t care”.

Inputs			
A	Clk	LD	Register Outputs
X	X	0	No Change – Last Register Output
A		1	A loaded on rising edge, A seen on output after propagation delay

Table 20. Register Operation.

The circuit is show in Figure 36, and is comprised of two eight-bit registers connected in series.

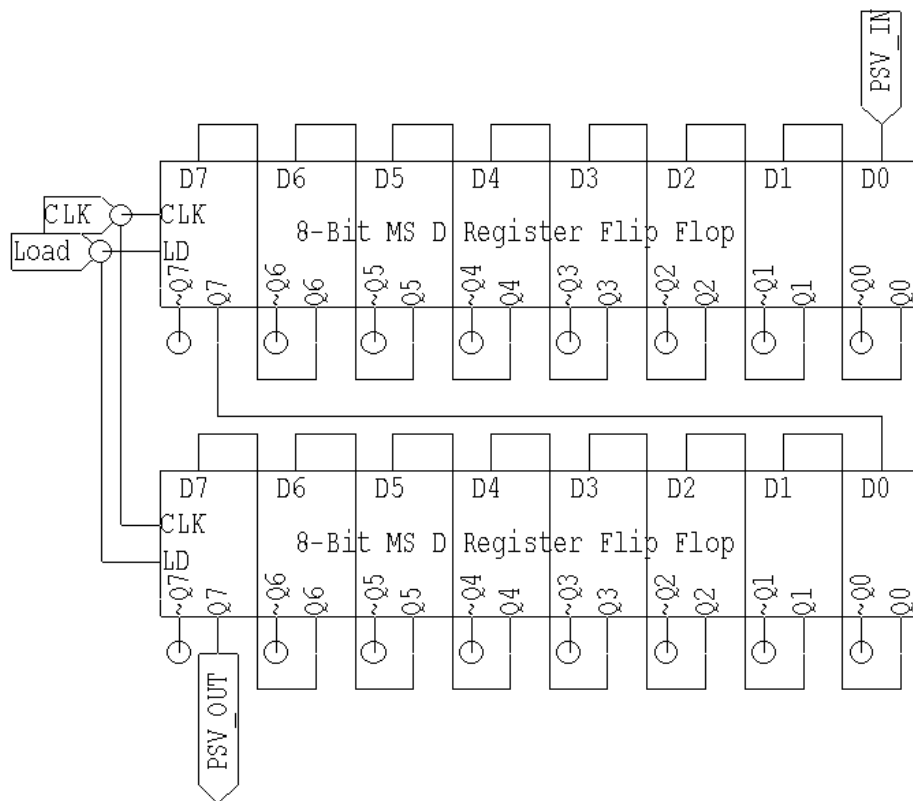


Figure 36. 16 Clock Period Delay – *Phase Signal Valid* Circuit.

3. RCA_13927Buffer

The global control signals, *Clk* and *LD*, control 119 registers and have a large fan out of gates (capacitive load to charge). There are two primary means to distribute these signals to all registers while minimizing skew, maintaining the rectangular waveform

shape and ensuring fast rise and fall times. The signals can be buffered through a fan out tree or can be buffered by one large driver circuit. Note that the nine-bit phase values in the 9-to-5 Bit Conversion circuit also utilizes this sub-circuit as well. To minimize the number of transistors required and the amount of layout wiring that would be required, the large driver circuit option was implemented. An optimum stage-sizing ratio ' a_{opt} ' was determined iteratively to be the value 3.4 from [11]:

$$a_{opt} = e^{\frac{K+a_{opt}}{a_{opt}}} \quad (3.5)$$

and

$$k = \frac{C_{drain}}{C_{gate}} = 0.776 \quad (3.6)$$

where:

- C_{drain} is the drain capacitance, and
- C_{gate} is the gate capacitance.

Inverter size scaling should therefore increase (approximate 3.4 to an integer value) by a factor of three for optimum gate driving properties. T-SPICE testing shows that a 27x inverter could adequately drive a capacitive load of 100+ inverters within a 700 MHz clock period. Figure 37 shows the non-inverting (buffer) driver circuit:

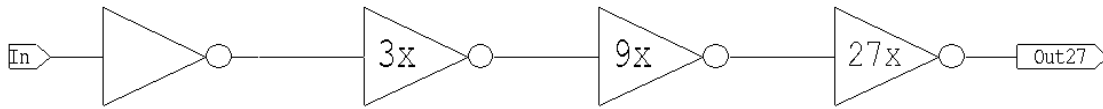


Figure 37. Buffer Driver Circuit.

4. 16-Bit Pipelined Adder/Subtractor

Dr. Fouts designed the 16-bit CLAH that is used as the fundamental building block in the design of a 16-bit Adder/Subtractor (A/S). The adder was tested alone and found to operate at a maximum clock speed of 780 MHz. However, with added gate delay, because of additional gates on the front end to change the design to an A/S, timing problems were found in the circuit operating at 700 MHz. As such, it could not be

guaranteed to produce proper timing results and a pipeline register stage was inserted to meet the required clock period. The A/S works on the principle of always doing addition, but via the two's complement, adds the negative of the B input number to execute subtraction:

- Addition: $A + B = \text{Sum}$
- Subtraction: $A + (-B) = \text{Difference}$.

The circuit is shown in Figure 38. XOR gates are used to form the one's complement of the B input. Subtraction is enabled via a control line M when $M = '1'$. Table 21 shows the operation of the one's complement. In rows two and four, whenever the M bit is a one, the output Z is the complement of the input X . In rows one and three, the output Z follows the input when $M = '0'$. Thus, the control signal M controls the one's complement on the B input, and by adding a one to the CLA_H via the Carry In signal, C_i , by also routing the M signal there, the two's complement is formed, and subtraction occurs.

Input X	Input M	Output Z
0	0	0
0	1	1
1	0	1
1	1	0

Table 21. XOR Truth Table.

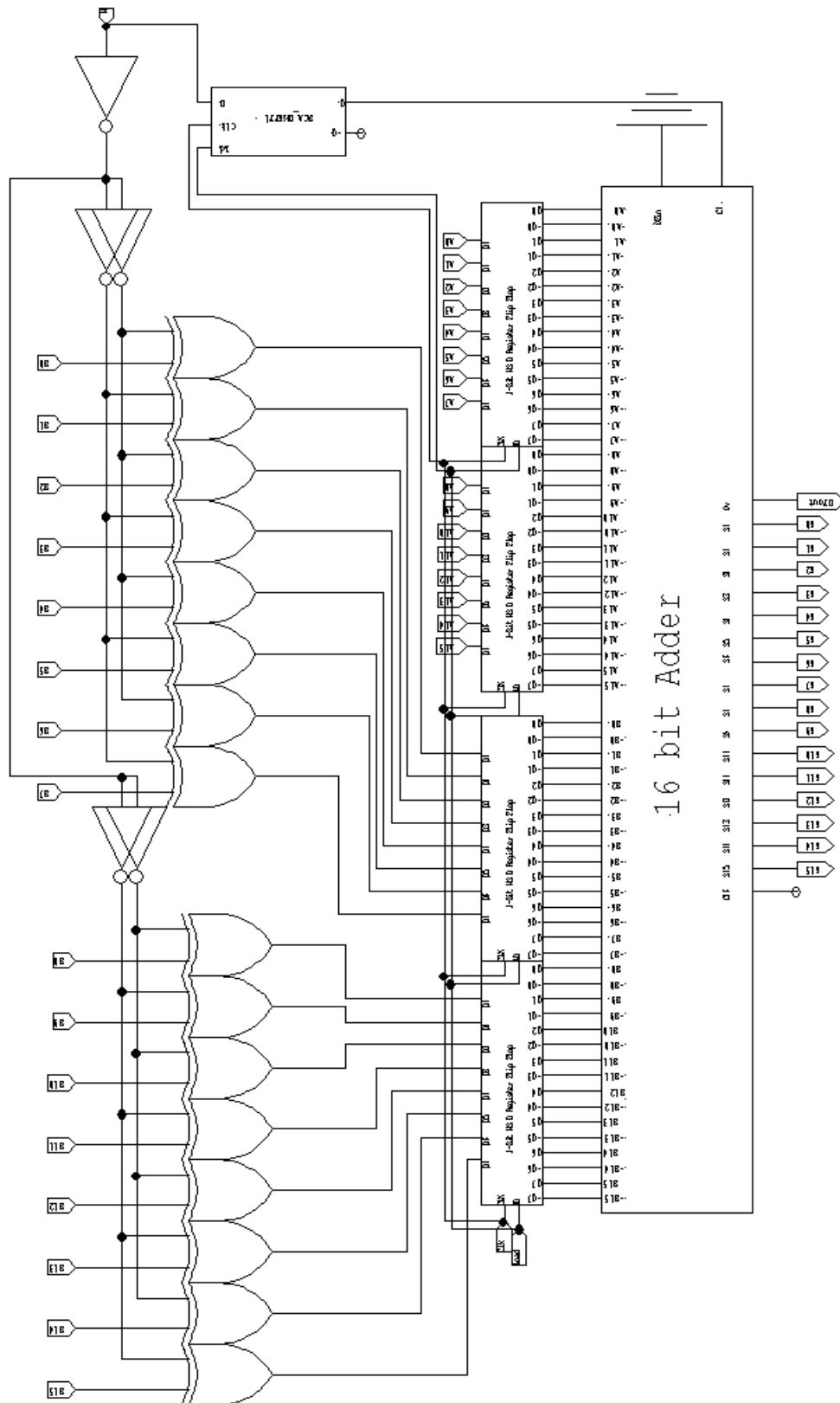


Figure 38. 16-bit Pipelined Adder/Subtractor.

E. NUMBER FORMATS

The eight-bit two's complement Z (accumulated phase value), Q and I numbers are physically converted to a 16-bit two's complement fixed-radix positional number system when passed as inputs to the RCA CORDIC Level 45. The 16-bit number is of the form of a 9+7 fixed-point decimal number, which has a two's complement integer radix (base) of nine bits and an implicit digit set of seven bits (see Table 22). It is of sufficient precision to account for increasing I and decreasing Q values during CORDIC iterations. The I values as discussed previously, however, can require more than nine integer bits after the initial Rotation stage.

Bit Position Number:	MSB									Decimal Point		LSB						
	15	14	13	12	11	10	9	8	7	.		6	5	4	3	2	1	0
	9-bit Integer											7-bit Decimal						

Table 22. CORDIC Iteration Number Format.

The different phase values of R expressed in degrees are listed in Table 23 using this number format. The table also presents the negligible error by representing the constant R phase values using this number format of only 16-bits.

During Q and I shifts shown previously in Table 4, the numbers should be, in general, hardwired-shifted sign extended by replicating the MSB. Indeed, this is the means by which the Q values are shifted. However, because I is always a positive number after Rotation, the fictional 10^{th} integer bit of I (and higher) is always a logical zero. Thus, some MSB I shifts are grounded as appropriate.

Level	Phase in Deg	16 Bit Phase in Deg	Error	Error %	MSB Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB Bit 0
	90	90.00000000	0.00000000	0.00	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0
0	45	45.00000000	0.00000000	0.00	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0
1	26.56505	26.56250000	0.00255000	0.25	0	0	0	0	1	1	0	1	0	1	0	0	1	0	0	0
2	14.03624	14.03125000	0.00499000	0.50	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0
3	7.12502	7.12500000	0.00002000	0.00	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
4	3.57633	3.57031250	0.00601750	0.60	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1
5	1.78991	1.78906250	0.00084750	0.08	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1
6	0.89517	0.89062500	0.00454500	0.45	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0
7	0.44761	0.44531250	0.00229750	0.23	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1
8	0.223811	0.21875000	0.00506100	0.51	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0

Table 23. 16-Bit Number Format.

F. CORDIC ROTATION LEVEL

1. Overview

The CORDIC rotation level performs the initial Rotation step by adding a constant phase value of $\pm 90^\circ$. It takes as inputs, two eight-bit Q and I values on each rising edge clock pulse and performs the rotation by negating one of the values and swapping, dependant upon the sign of Q in accordance with Table 3. A stand-alone functional block, it possesses input and output stage pipeline registers. The internal state machine control is the $Q7$ signal, the most significant sign bit of the Q data. When $Q7 = '1'$, the number is negative, and $+90^\circ$ is added to the Z phase output register. This is accomplished by directly using the value of $Q7$ to program the register with either $+90^\circ$ or -90° in binary. Table 24 shows both phase constant numbers and the corresponding $Q7$ logic value, the appropriate binary value to load into each bit position displayed as Z logic. For example, $Z0$, the least significant bit loaded into the $D0$ of the register, is always zero, and thus, is a grounded input. $Z1$ is always one and is, therefore, always pulled high to V_{dd} . The remaining $Z<7..2>$ bits are buffered and inverted, as applicable, $Q7$ values.

Bit	MSB 7	6	5	4	3	2	1	LSB 0
+90	0	1	0	1	1	0	1	0
-90	1	0	1	0	0	1	1	0
Z Logic	$Q7'$	$Q7$	$Q7'$	$Q7$	$Q7$	$Q7'$	1	0

Table 24. $Q7$ Programming of $\pm 90^\circ$.

The negation of one input number is accomplished by an algorithmic two's complement implementation. Swapping is a "logical switch" of the output data. Q input data becomes the I output data and vice versa. Finally, the circuit has a correction block to catch an input value of -128 , which cannot be two's complemented to $+128$ because this positive number cannot be represented with only eight bits.

2. Rotation Circuit Diagram

The complete circuit is shown as Figure 39, with alphabet numbered sub-circuits detailed in Sections 3 through 5.

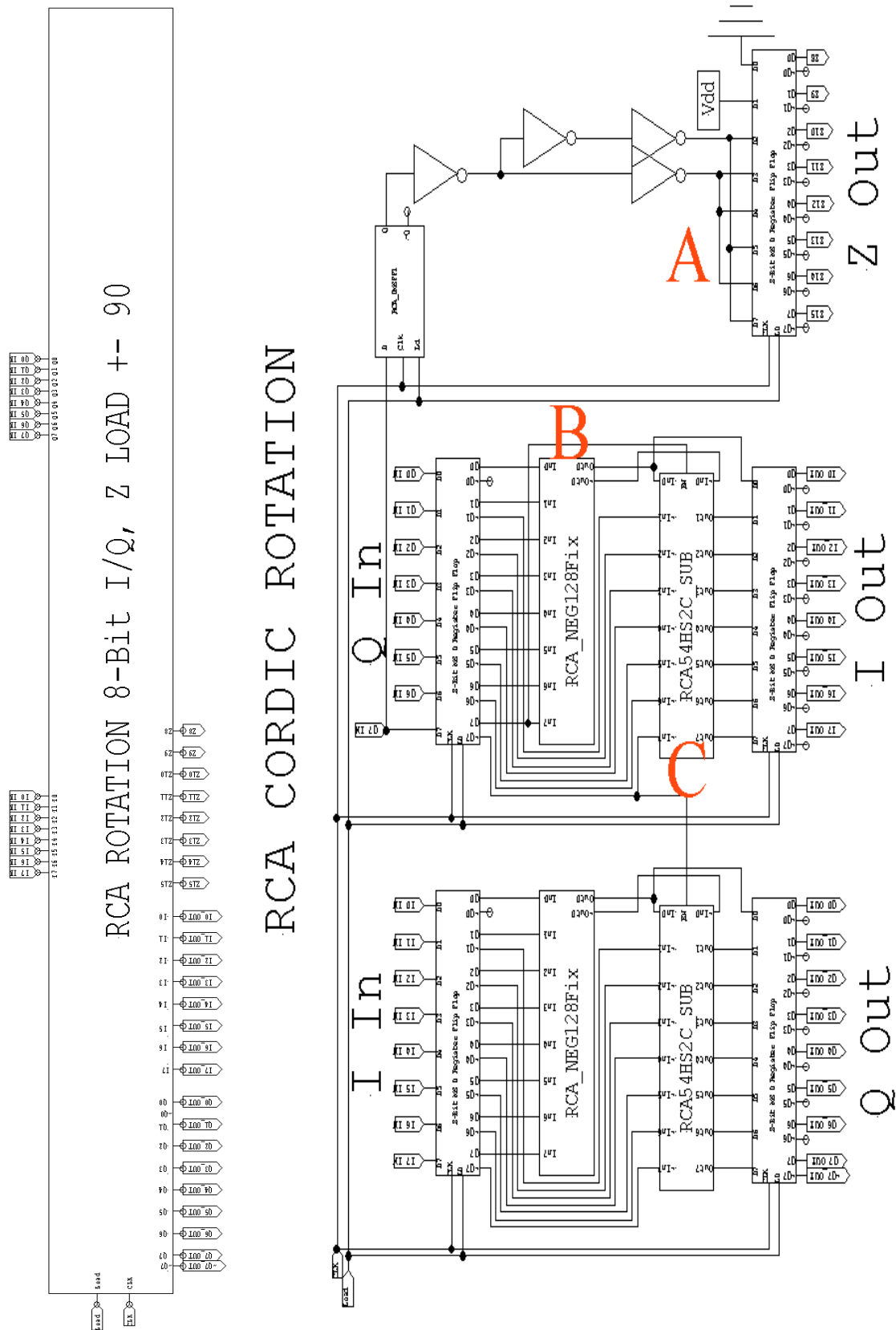


Figure 39. Rotation Level Symbol (left) and Circuit (Right).

3. Constant Phase Loading

Figure 40 shows the zoomed location ‘A’ of Figure 39. The two vertical paths present a two-inverter (buffered and non-inverting) and three-inverter (buffered and complementing) chains, respectively, to provide the $Q7$ and $\sim Q7$ signals to the output pipeline register as explained in the overview section above. The one-bit DMSFF ensures $Q7$ control signal data synchronization with the flow of data through the $Q Out$ and $I Out$ circuit sections.

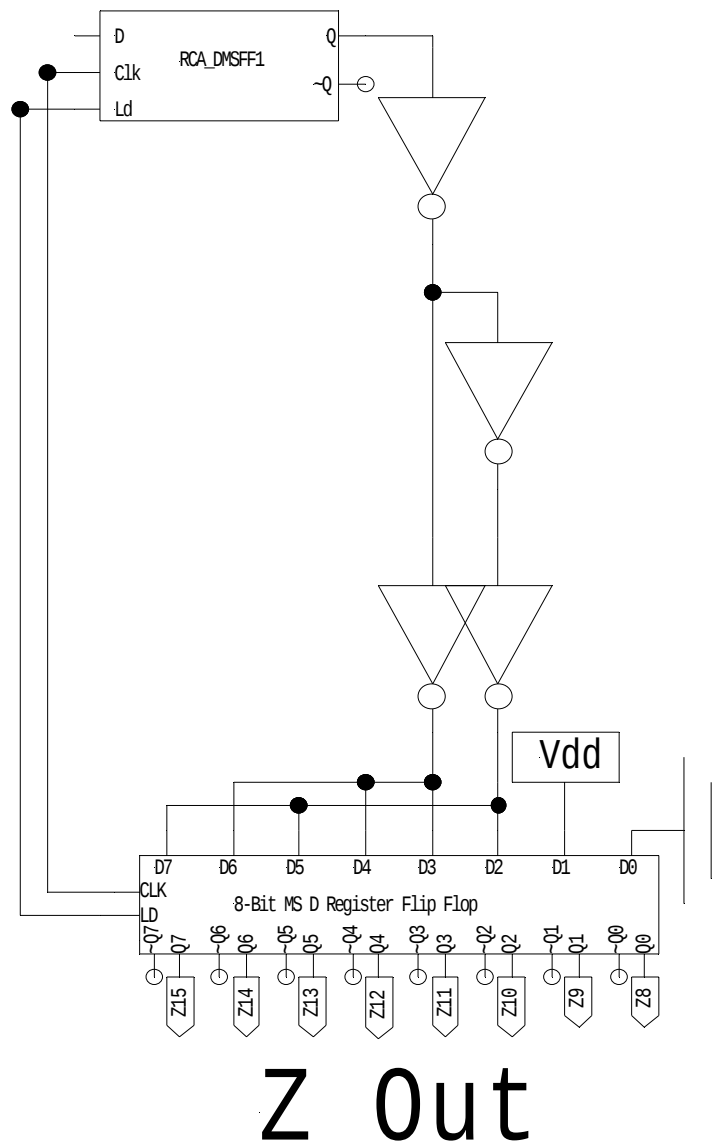


Figure 40. $\pm 90^\circ$ Phase Loading.

4. Negative 128 Fix Circuit

Shown as block ‘B’ in Figure 39, this sub-circuit detects when an input number is -128 , or $1000\ 0000$ in binary, via the NOR and AND gates, and changes it to -127 prior to the two’s complement circuit. If the output of the AND gate is true, the XOR flips the least significant bit, thereby, changing the number to -127 , or $1000\ 0001$ in binary, and thus, allowing the number to be negated to become $+127$, a value which is represented as an eight-bit number. Increasing the number by the “addition” of one to produce -127 has no effect on the phase results produced by the CORDIC algorithm. An alternative method to correct this -128 overflow error, is to increase the Q and I data to nine-bits in this stage, which would result in an increase to nine-bit output registers and additional logic in the two’s complement circuit, vice a four gates solution shown below as Figure 41.

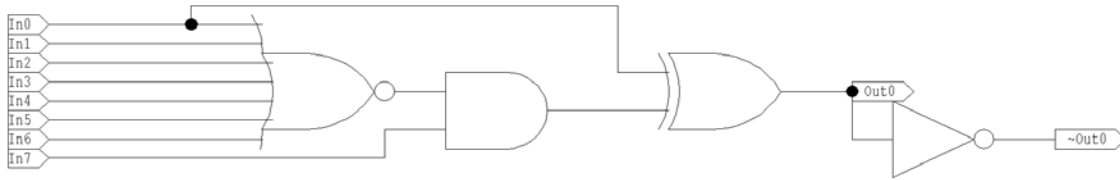


Figure 41. Negative 128 Circuit Fix.

5. Two’s Complement Circuit

Shown as block ‘C’ in Figure 39, this circuit negates an input number. The negation of the number is not implemented via the two’s complement using eight-bit addition, as in:

$$\begin{array}{rcl}
 \text{Number:} & 0011\ 0011 & = 51 \\
 \text{Ones Complement:} & 1100\ 1100 & = -52 \\
 \text{Add One:} & \underline{\quad + 1 \quad} & \\
 \text{Two’s Complement:} & 1100\ 1101 & = -51
 \end{array}$$

Rather, to save on transistor count, the circuit uses an algorithmic approach, described by the following pseudo-code:

```

BEGIN
    Set Flag = 0;
    FOR bits 0 to n:
        IF (Flag == 1)
            THEN Flip bit;
        ELSEIF (bit == 1)
            THEN Set Flag = 1;
        END FOR Loop
    END

```

This algorithm is implemented using only 21 logic gates, which is much smaller than the number of gates required for an eight-bit adder circuit. The hardware implementation uses XOR gates to execute bit flips, controlled by the ripple Flag that can be thought of as an enable signal. The ripple of the Flag signal has considerable delay from the least significant bit (LSB) until it arrives at the most significant bit (MSB), and therefore, has the inclusion of a “carry-look ahead” block. This block, seen as the center top three gates and highlighted in yellow in Figure 42, quickly propagates the Flag signal to the MSB, thereby reducing combinatorial logic delay and achieving high clock speed operation.

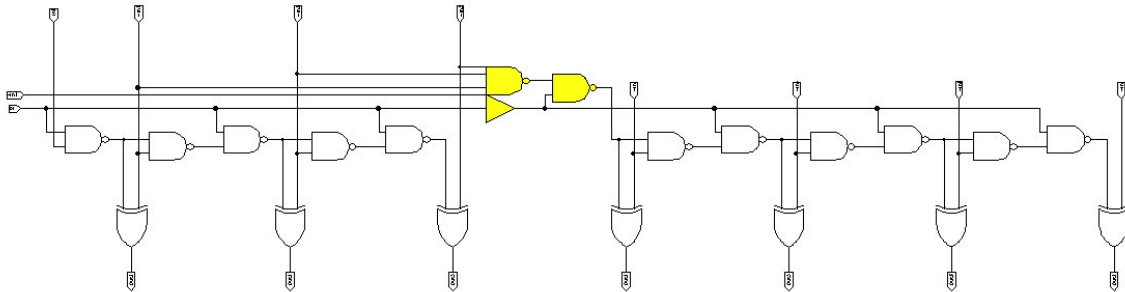


Figure 42. Two's Complement Circuit.

G. CORDIC GENERAL VECTOR LEVEL

1. Overview

The CORDIC General Vector Level is one stage of the bit-parallel unrolled CORDIC hardware that is replicated in the circuit the requisite number of times necessary for the iteration accuracy desired. It allows ease of circuit building in both S-

Edit and layout, because it needs to be designed only once and can then be used many times.

2. Schematic

Figure 43 shows the CORDIC General Vector Level circuit. It differs from one stage of Figure 18 in the addition of output pipeline registers. There are three 16-bit A/S's: two to execute Table 4 mathematics on Q and I , and one to accumulate the phase Z . The MSB (sign) bit of the input Q data and its complement are routed as the ' M ' control signal to the A/S's. Constant phase values from Table 24 for the given iteration are hardwired at location 'A'. Whenever Q is positive, subtraction occurs and conversely addition when Q is a negative number. I data is added to shifted values of Q at 'B', and Q data is added to shifted values of I at 'C'. Multiple levels of the CORDIC General Vector Level are cascaded vertically to implement multiple Vectoring iterations.

RCA Cordic Level General

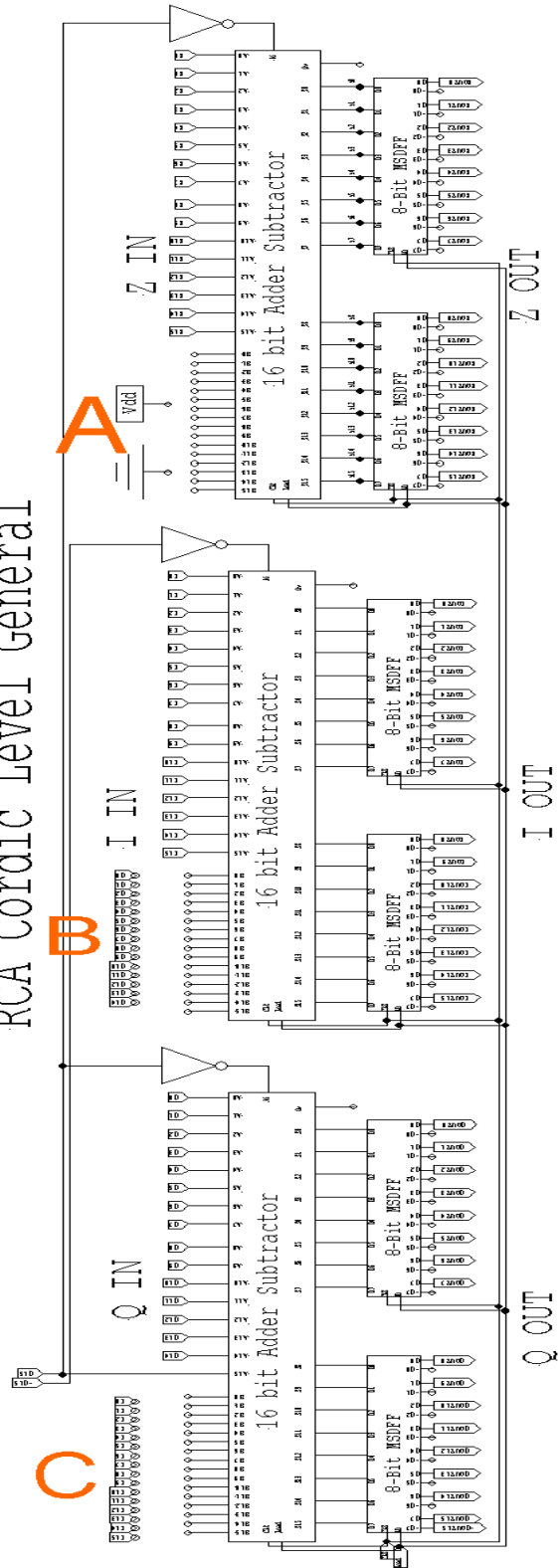


Figure 43. CORDIC General Vector Level Schematic.

3. Example: CORDIC Vector Level 14.03

A programmed CORDIC Vector level implementing $L=2$ is shown as Figure 44:

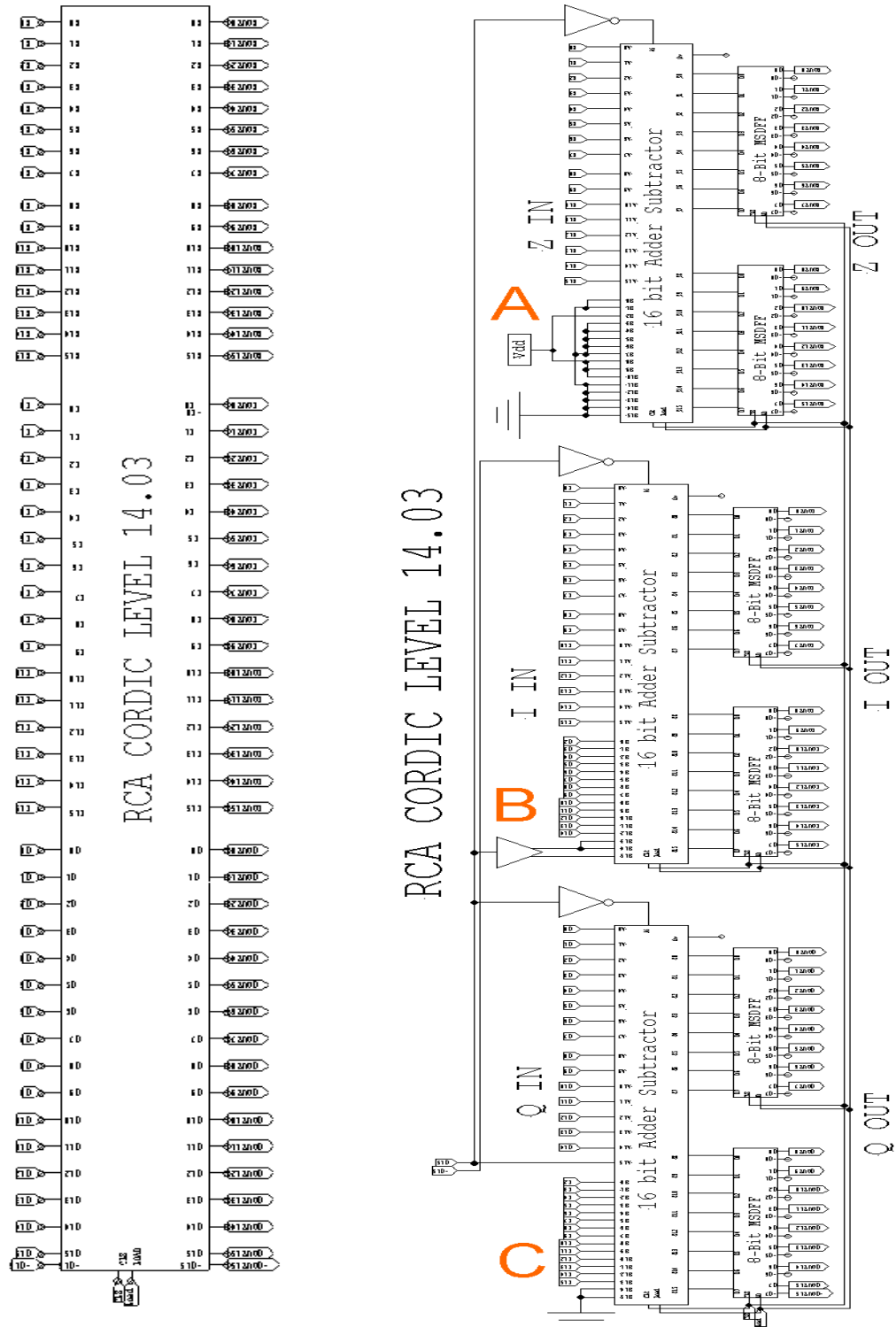


Figure 44. CORDIC Level 14.03 Symbol (left) and Circuit (right).

The phase value of R in degrees for this stage is 14.03624° and the number of hardwire shifts is two, corresponding to a K value of $2^L = 2^2 = 4$. In accordance with Table 24, the phase value R programmed is 14.03125° . This is represented as:

- Logic Value: LLLL LHHH L . LLL LHLL
- In Binary: 0000 0111 0 . 000 0100

where ‘L’ stands for “low” and is a logic ‘0’ (zero volts), and ‘H’ stands for “high” and is a logic ‘1’ (1.8 volts). The location ‘A’ from Figure 44 is zoomed and shown as Figure 45 and details this constant phase value programming. The reader can verify the pull-ups and pull-downs to Vdd and Ground.

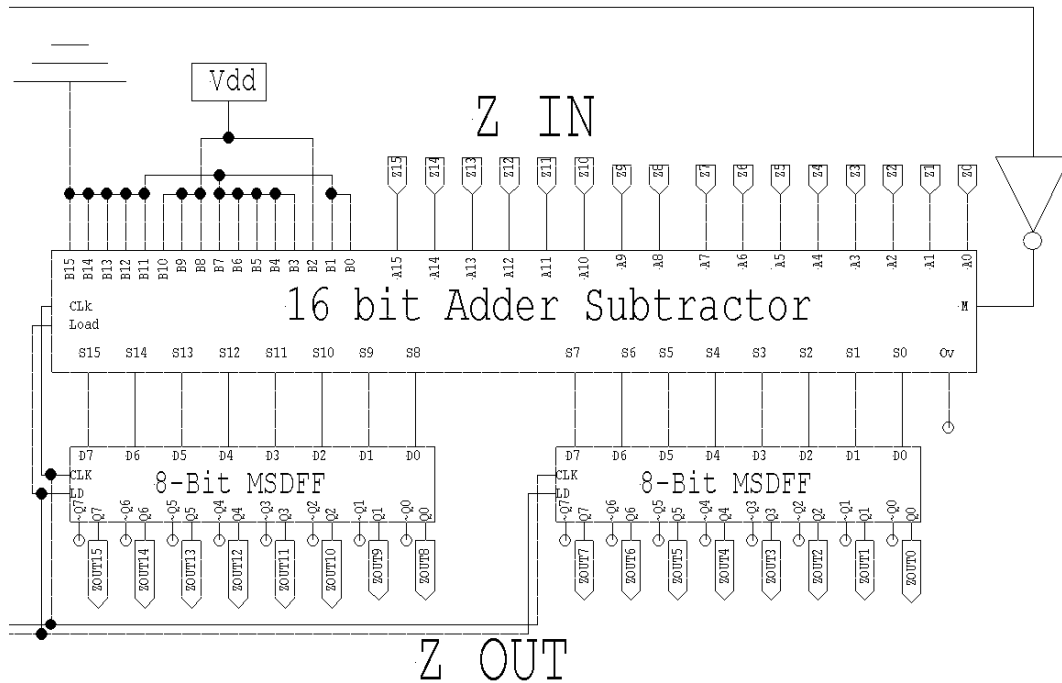


Figure 45. Zoomed In Phase Input – Phase Loading of 14.03125° .

Data is shifted right in order to divide by powers of two. As Q data may be a positive or negative value, and the shift must be an arithmetic right shift with sign extension. As there are two shifts for this iteration stage, the values of $Q1$ and $Q0$ are dropped and the MSB $Q15$ is replicated twice to implement sign extension. Figure 46 is a zoomed picture of Figure 44 location ‘B’. The $Q15$ line from the previous iteration

level is buffered prior to the input of the 16-bit A/S because this is also the ‘ M ’ control line for the other A/S’s, and hence, has a large fan out (capacitive loading).

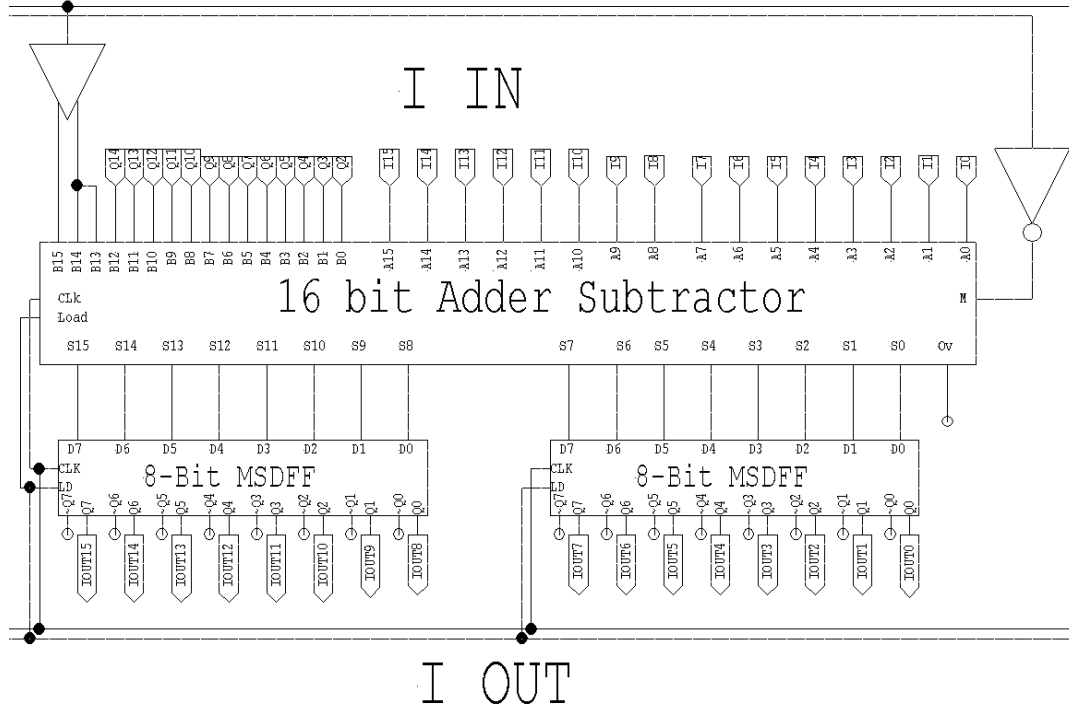


Figure 46. Zoomed in Q Hardwire Shifts.

Figure 47 is a zoomed view of location ‘C’ from Figure 44 showing the hardwire-shifted I data. Again, there are two shifts for this iteration stage. Hence, the values of $I1$ and $I0$ are dropped and ground is shifted in because higher order I data is always positive. The MSB shifted in after integer bit nine is always a logical zero.

Degree Range	5-Bit Phase Value Decimal	Degree Range	5-Bit Phase Value Decimal	Degree Range	5-Bit Phase Value Decimal	Degree Range	5-Bit Phase Value Decimal
0-11	0	91-101	8	181-191	16	271-281	24
12-23	1	102-112	9	192-203	17	282-93	25
24-34	2	113-124	10	204-214	18	294-304	26
35-45	3	125-135	11	215-225	19	305-315	27
46-56	4	136-146	12	226-236	20	316-326	28
57-67	5	147-158	13	237-248	21	327-338	29
68-78	6	159-169	14	249-259	22	339-349	30
79-90	7	170-180	15	260-270	23	350-359	31

Table 25. Phase Groupings.

Using the nine integer Z phase bits produced by the last CORDIC iteration stage, a 9 to 5 Bit Phase Conversion truth table was generated and is detailed in Appendix C. The table was then examined to determine the SOP minterms where each of the output five bits, F4 through F0, were a logic value of ‘1’. Table 26 shows this collection of minterms necessary to generate each F-bit result.

F4	F3	F2	F1	F0
1-10	411-421	456-466	478-488	489-500
11-21	400-410	445-455	467-477	467-477
22-33	388-399	434-444	434-444	445-455
34-44	377-387	422-432	422-432	422-433
45-55	366-376	366-376	388-399	400-410
56-66	354-365	354-365	377-387	377-387
67-78	343-353	343-353	343-353	354-365
79-89	332-342	332-342	332-342	332-342
90-100	1-10	1-10	1-10	1-10
101-111	11-21	11-21	11-21	22-33
112-123	22-33	22-33	45-55	45-55
124-134	34-44	34-44	56-66	67-78
135-145	45-55	90-100	90-100	90-100
146-156	56-66	101-111	101-111	112-123
157-168	67-78	112-123	135-145	135-145
169-179	79-89	124-134	146-156	157-168

Table 26. Minterms Necessary to Generate the 9-to-5 Conversion Circuit.

Using the collection of minterms shown above, a MATLAB coded Quine-McClusky algorithm was used to determine the five SOP logical functions. The resulting logic terms to implement the SOP functions are shown in Table 27 as R4 through R0. Function R0 required the logical OR'ing of 53 different AND products and is a very large array, whereas R4 is very easy to generate, requiring only 11 product terms.

Number of Terms	R0	R1	R2	R3	R4
1	x0x0001xx	xx0001xxx	x0001xxxx	x001xxxxx	01xxxxxxx
2	x001011x1	xx1011x1x	0x11xxxxx	011xxxxxx	0xxxxxxx1
3	x0010111x	xx10111xx	011xxxxxx	10xxxxxxx	100xxxxxx
4	0x111x0xx	0x110xxxx	10x0xxxxx	1x00xxxxx	0xxxxxx1x
5	01x001xxx	011xxxxxx	1x01xx11x	1x0x000xx	0x1xxxxxx
6	01x100xxx	100xxxxxx	1x01x1xxx	1x0x00x0x	0xxxxx1xx
7	01x111xxx	10x0xxxxx	1x011xxxx	x00xxxxx1	x0100x0xx
8	011xxxxxx	10xx11xx1	10xxx0xxx	x010x0xxx	0xxx1xxx
9	100xxxxxx	10xx11x1x	x1100xxxx	x0x001xxx	x01000xxx
10	10x00xxxx	10xx111xx	x0x10x0xx	x0x01x00x	0xxx1xxxx
11	x111100xx	1x000xxxx	10xxx000	0x011x1xx	0xx1xxxxx
12	0x11x0x00	1x0x01xxx	x000x1xxx	x00x1xxxx	
13	1xx0x0011	x11100xxx	x0x10xx00	0x0111xxx	
14	10xx001xx	x1110x000	x00x001xx	x00xxx1xx	
15	x110110xx	xx000x111	01x11x1xx	x00xxxx1x	
16	x10101000	01x010xxx	0x1x111xx		
17	0xx000111	x0011xxxx	01x111xxx		
18	0xx1101xx	01xx110xx	01x000xx0		
19	x00011xxx	01xx1x100	0x0000x01		
20	11x1011xx	1x0110xxx	0x000001x		
21	x11000xxx	xx0111x00	10xx0xxxx		
22	11x0101xx	01x11x1xx	0x1x11x1x		
23	x01x00011	x00x10x0x	x110x000x		
24	10xx0x01x	x1101x1xx	1x10x00x0		
25	x00000xx1	1x101xx11			
26	x0000x010	x000x001x			
27	0xx00100x	x0000x1xx			
28	0x10x1x10	x0x00000x			
29	0x10x101x	x01x0000x			
30	0x10x110x	0x1x000x0			
31	0xx01111x	1x01x10xx			
32	01xx111x1	1x0x0x11x			
33	00x1x000x	1x1011xxx			
34	0x11x00xx	x00x011x1			
35	1x00x00xx	x001x111x			
36	1xx1010x1				
37	10xx01x0x				
38	1xx10101x				
39	10x0x0x0x				
40	10x0x0xx0				
41	10x111xx1				
42	10x111x1x				
43	10x1x110x				
44	00x110xxx				
45	1x001x00x				
46	1x001x0x0				
47	1x0x1000x				
48	x1101x10x				
49	x0001x11x				
50	1x01x111x				
51	1x01x11x1				
52	1x010x11x				
53	x1001000x				

Bit Ordering (x = don't care)

MSB 8 7 6 5 4 3 2 1 0 LSB

example: x 0 0 1 x 1 1 1 x

$$F = \overline{Z7} \cdot \overline{Z6} \cdot Z5 \cdot Z3 \cdot Z2 \cdot Z1$$

Table 27. Quine-McClusky Results of 9-to-5 Bit Logic Minimization.

Using only the nine integer bits from the last CORDIC Vector level produces a truncation error. Consider a 16-bit Z phase data produced at the last iteration of the CORDIC of 1 0110 0111.0010 000. Truncating this number by only using the most significant nine bits, yields 1 0110 0111. This new number in two's complement is:

$$\text{Number:} \quad 1\ 0110\ 0111 \quad = ?$$

$$\text{Ones Complement:} \quad 0\ 1001\ 1000 \quad = 152$$

$$\text{Add One to the LSB:} \quad \underline{\hspace{2cm} + 1}$$

$$\text{Two's Complement:} \quad 0\ 1001\ 1001 \quad = 153$$

Compare the non-truncated number using all 16-bits of precision:

$$\text{Number:} \quad 1\ 0110\ 0111\ .\ 0010\ 000 \quad = ?$$

$$\text{Ones Complement:} \quad 0\ 1001\ 1000\ .\ 1101\ 111 \quad = 152.8671875$$

$$\text{Add One to the LSB:} \quad \underline{\hspace{2cm} + 1}$$

$$\text{Two's Complement:} \quad 0\ 1001\ 1000\ .\ 1110\ 000 \quad = 152.875$$

Thus, the hardware is actually passing the number -153 into the 9 to 5 Conversion, when it should be $0\ 1001\ 1000 = -152$. This is because the decimal bits are ignored in order to use the minimum number of terms in the Quine-McClusky algorithm (nine vs. sixteen bits). Ignoring the decimal bits, in essence, rounds towards the next more negative number. To correct this truncation error, the hardware adds one to the Z data if the number is negative and has any decimal bit. Thus, $-153 + 1 = -152$, is the correct data to use. If the number does not have any fractional bits, then, no truncation error occurs and no extra addition of one is necessary:

$$\text{Number:} \quad 1\ 0110\ 0111\ .\ 0000\ 00 \quad = ?$$

$$\text{Ones Complement:} \quad 0\ 1001\ 1000\ .\ 1111\ 11 \quad = 152$$

$$\text{Add One to the LSB:} \quad \underline{\hspace{2cm} + 1}$$

$$\text{Two's Complement} \quad 0\ 1001\ 1001\ .\ 0000\ 00 \quad = 153$$

i.e., the hardware should in this case produce -153 .

2. Schematics

The completed 9-to-5 Bit Conversion circuit is shown in Figure 48 and as a sample hardware implementation, the logic function implementation of R0 from Table 27 is shown in Figure 49.

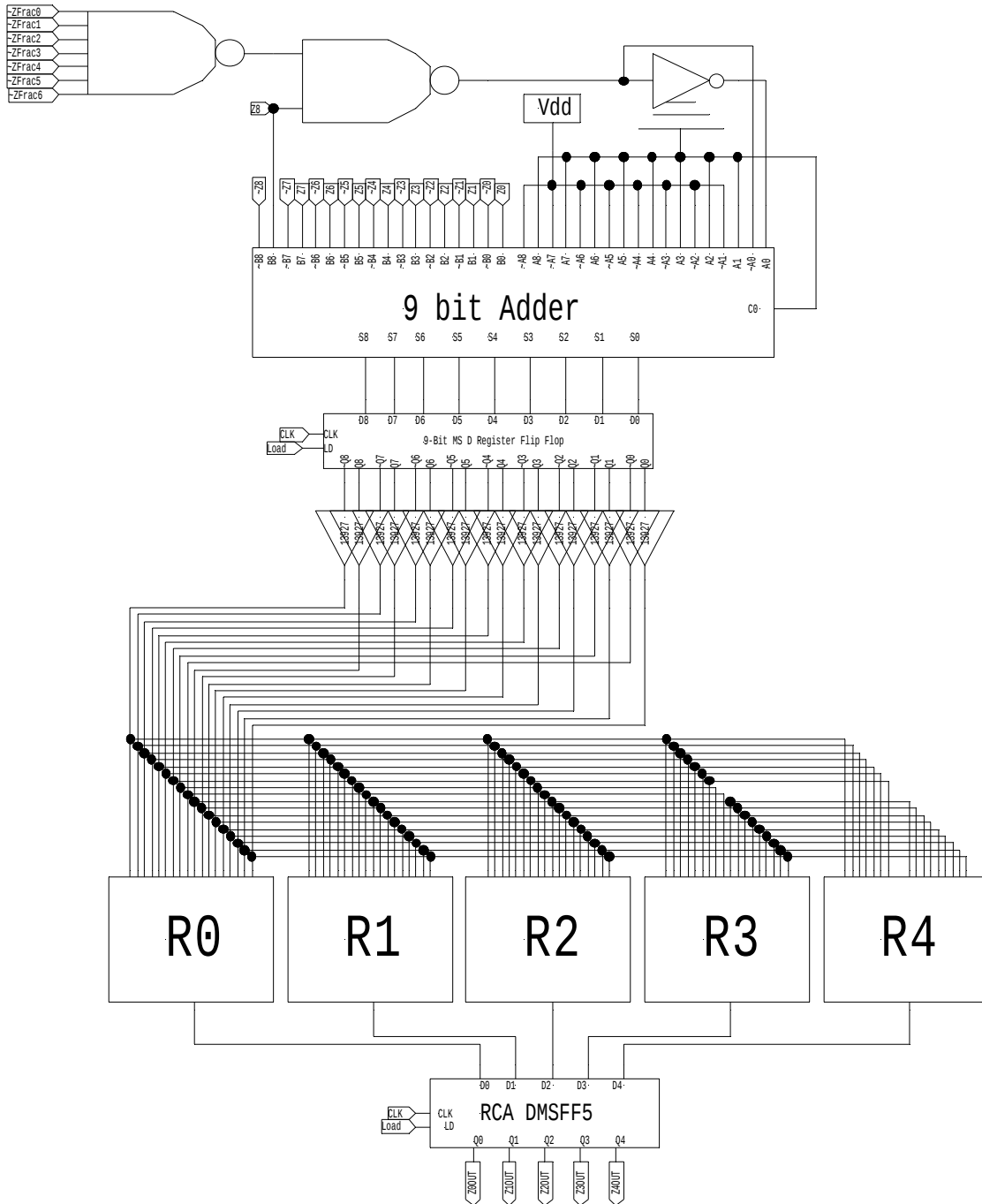


Figure 48. RCA_9to5_Conversion Circuit.

The top portion of the Figure 49 schematic detects a negative number with at least one fractional bit being a logic '1' and correspondingly adds one. The output of the nine-bit adder must drive 138 logic gates to implement the five logic functions of the phase conversion. Thus, the adder outputs after a pipeline register are buffered through RCA_13927Buffer circuits. Finally, the five conversion bits are pipelined before their processing in the DIS.

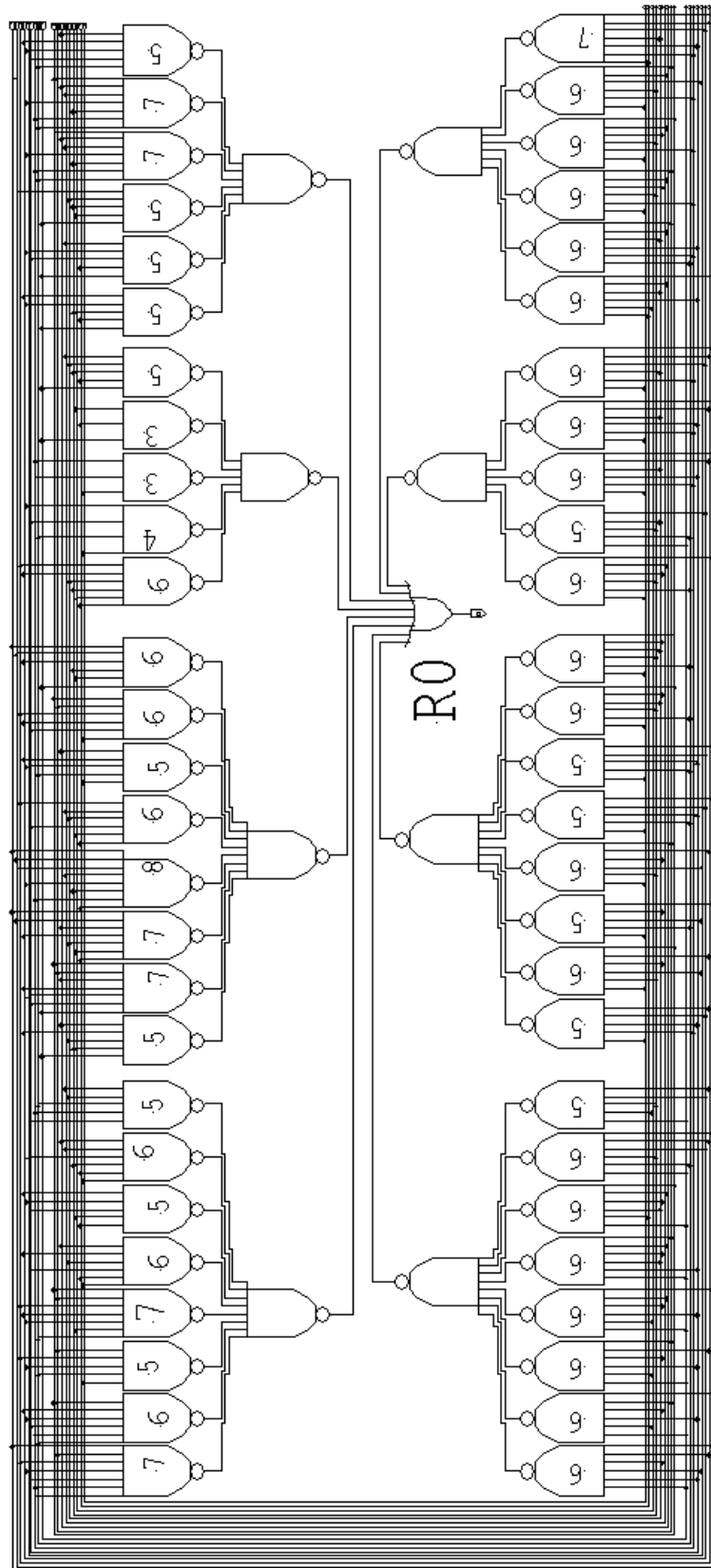


Figure 49. R0 Function Hardware Implementation.

I. COMPLETED CIRCUIT

1. Completed Circuit

Figure 50 shows the completed circuit, which takes as inputs: eight-bit Q and I data, a *Load* signal, a *Clock* signal, and *Phase Signal Valid In*. It produces a five-bit phase result, Z , and a *Phase Signal Valid Out*.

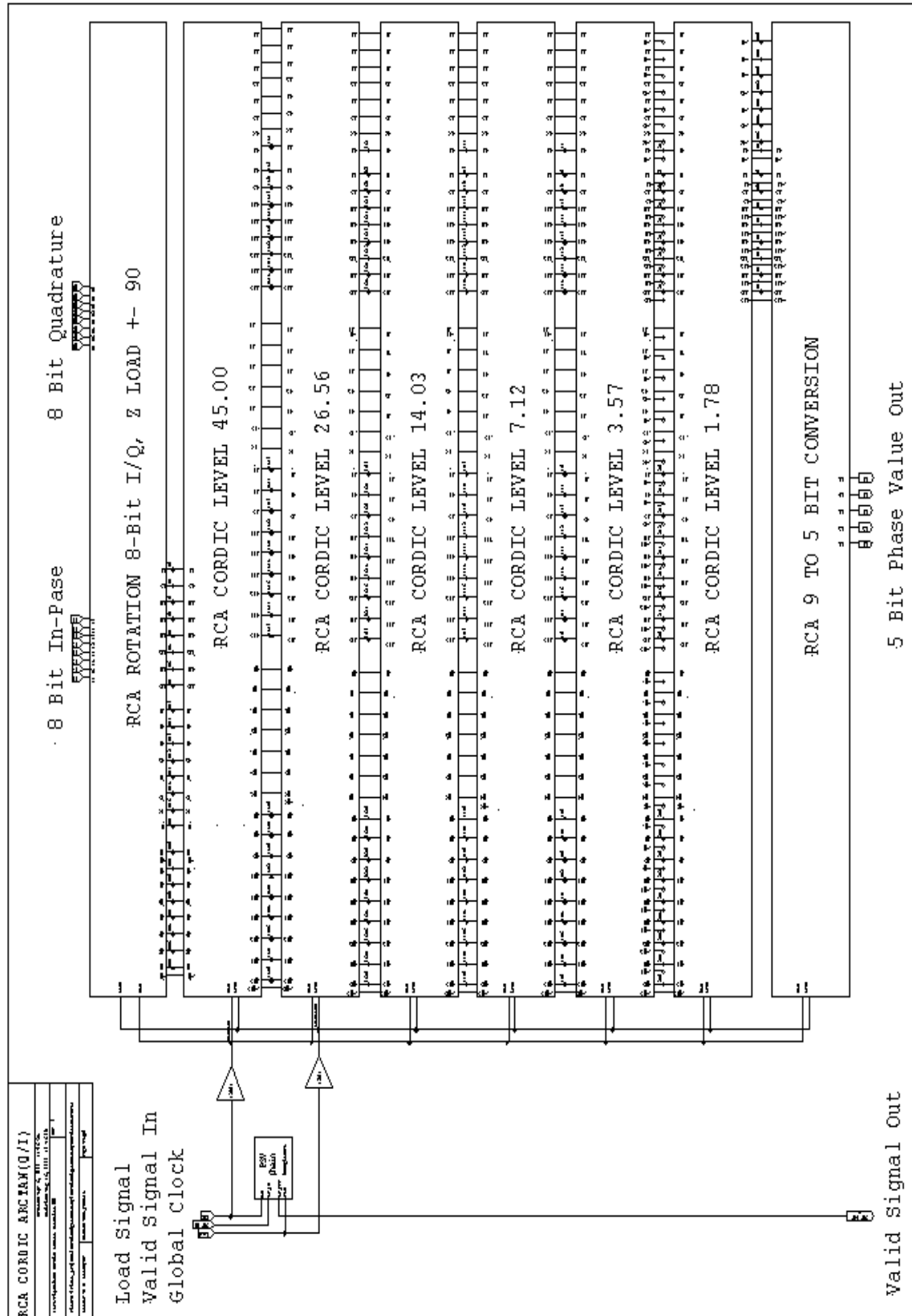


Figure 50. Completed Circuit.

2. Circuit Verification and Parameters

Table 28 details T-SPICE determined electrical circuit parameters:

Symbol	Parameter	Value	Units
V_{dd}	Power Supply Voltage	1.8	V
τ	Clock Period	1.4285714	ns
f	Clock frequency	700	MHz
P_{Ave}	Average Power	0.0567	W
P_{Inst}	Maximum Instantaneous Power	0.3141	W
L_P	Minimum Sized PFET Length	180	nm
W_P	Minimum Sized PFET Width	450	nm
L_N	Minimum Sized NFET Length	180	nm
W_N	Minimum Sized NFET Width	450	nm
β_N	N-FET Transistor Gain Factor	435.0	$\mu\text{A}/\text{V}^2$
β_p	P_FET Transistor Gain Factor	146.5	$\mu\text{A}/\text{V}^2$
s	Clock Skew	0.41	ns
β_N / β_p	N-to-P Gain Factor Ratio	2.969	
M_CMOSP	P-Fet count	24236	
M_CMOSN	N-Fet count	24236	
$Nodes$	Total Nodes	20442	
M_Elm	Number of elements	48472	

Table 28. Circuit Parameters.

Figure 51 shows the current draw on the power supply for a string of different test vectors. The average and peak current were used in the calculation of the above power parameters. The *Clock* and *Load* signal before and after the RCA_13927Buffer shows sample skew measurement, s , in Figure 52.

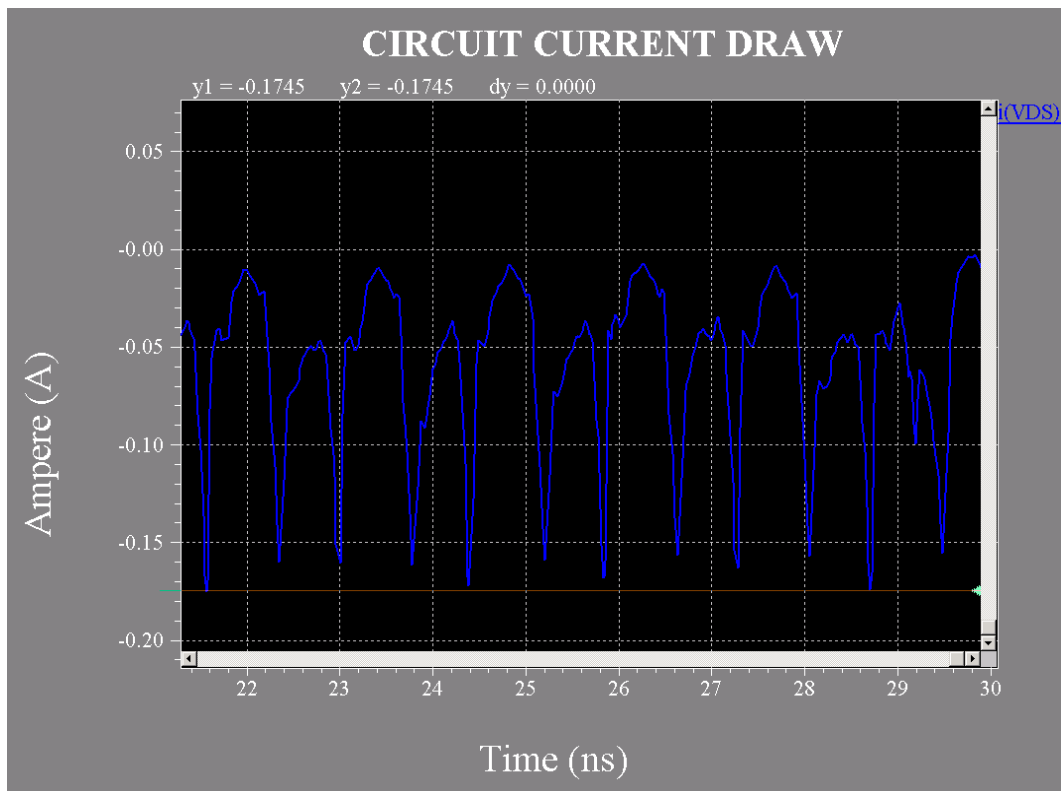


Figure 51. Power Supply Current Draw.

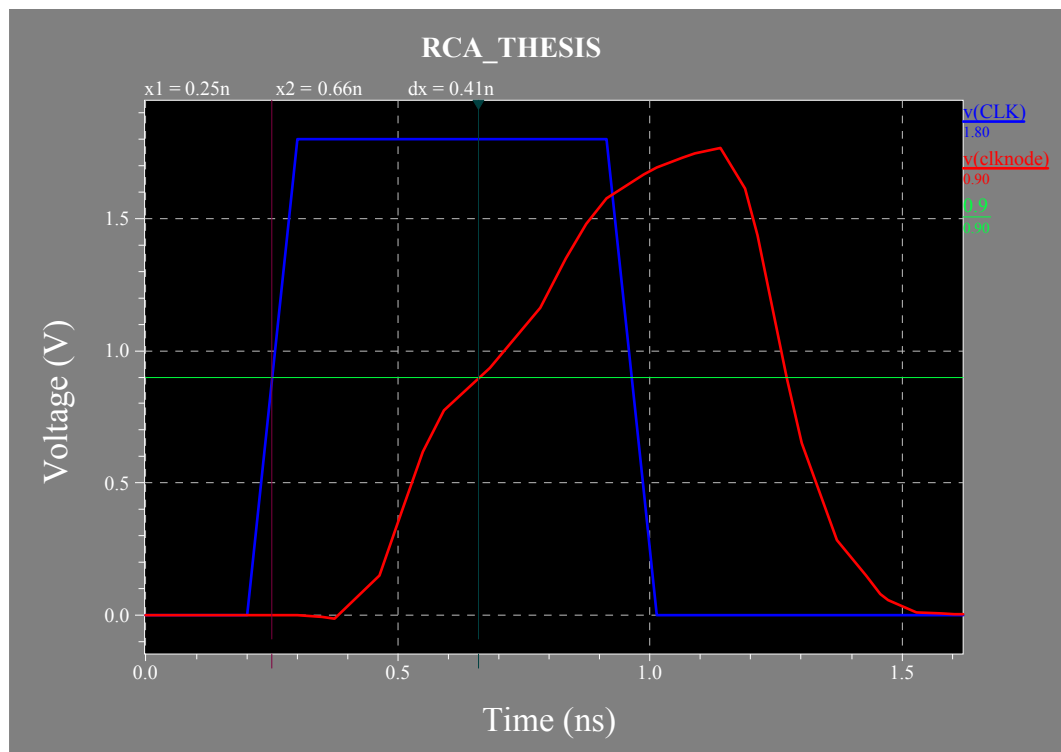


Figure 52. Clock and Load Skew.

3. Verification

Verification of the circuit was done with two different methods:

- Functional (logical) Testing and
- Timing Testing of selected test vectors.

The functional testing was done by extracting the complete S-Edit circuit as a VHDL file. The Aldec VHDL program was used to test all 65,536 possible input combinations of Q and I . A 0.5 ns clock was used in the Aldec test to speed up simulation completion time as only logical testing was important using this software. Figure 53 shows the previous example use of $Q = 4$, $I = -8$. The circuit produces the correct output value of 13, 16-clock pulses later.

T-SPICE was used for all important circuit electrical and timing parameters, but only for small sets of test vectors, because of the excessively long simulation run times. Figure 54 shows T-SPICE timing simulation using the same Q and I inputs and the corresponding correct output results, plotted with annotations using the W-Edit program. The Valid Signal Out goes high 16 clock periods after both inputs change from zero, indicating that the output phase number (in this case 13) is valid, and is correct. T-SPICE timing simulations show that the circuit runs properly at a 700 MHz clock frequency, producing results after a latency of 16-pipeline clock periods.

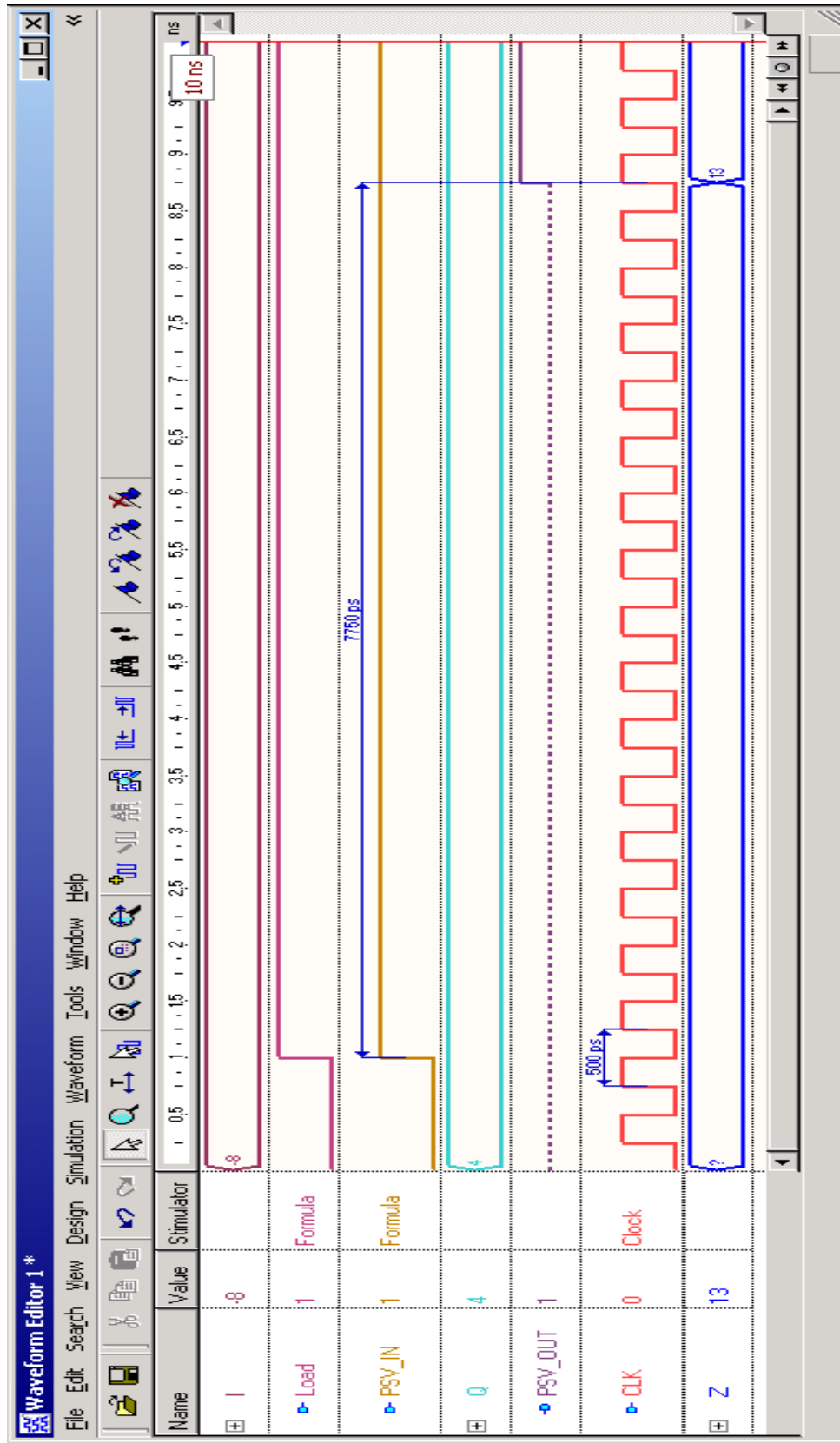


Figure 53. VHDL Functional Verification.

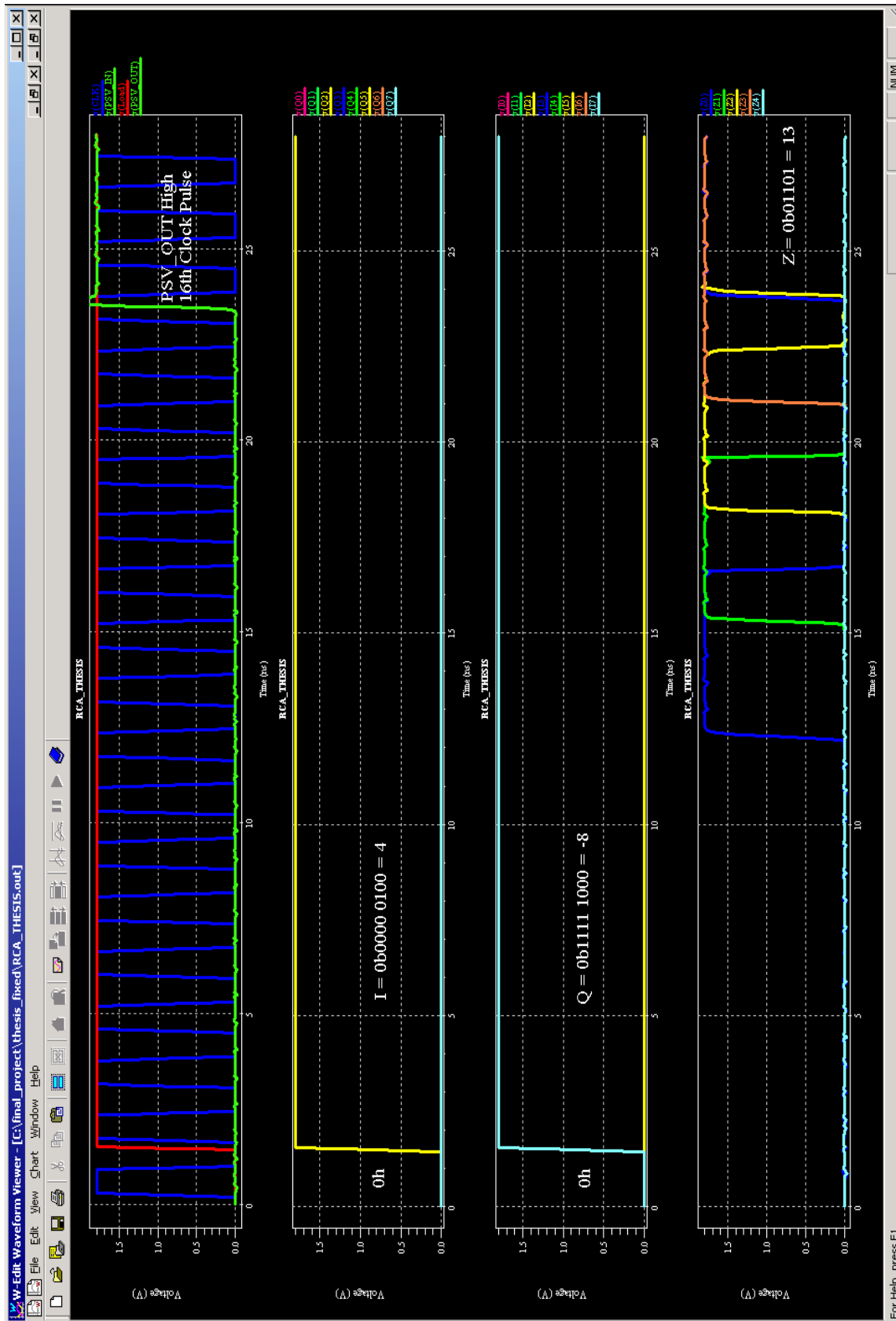


Figure 54. T-Spice Timing Verification.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS AND RECOMMENDATIONS

A. GENERAL

This thesis demonstrated the hierarchical research, design, VLSI circuit implementation, and testing of a high performance ASIC that extracts the phase of a complex signal. The circuit meets all the design goals:

- A minimum transistor count implementation of only 48,472 FETs,
- Ability to operate at 700 MHz clock speed,
- Low average power use of 56.7mW, and
- Accurate production of a five-bit phase value for use in the DIS by implementing the arctangent function using only six CORDIC vector iterations.

The amplitude-to-phase converter is a robust design that produces a five-bit phase result of eight-bit input Q and I data on each rising clock edge after a 16-clock pulse latency delay. Once the pipeline is loaded, phase results are produced on each clock edge. The circuit can easily and quickly be scaled for more CORDIC accuracy by the inclusion of additional General Vector Levels and updating the Z constant phase value loaded and the hardwire shift programming.

Although primarily designed for ASIC implementation, the circuit can also be readily implemented as a Field Programmable Gate Array (FPGA).

B. LESSONS LEARNED

A complicated circuit should be first designed and tested using a high-level software package such as VHDL prior to the S-Edit design. VHDL allows very fast functional testing of all possible circuit inputs and, therefore, facilitates circuit debugging. Test vectors cannot be used to guarantee proper circuit operation under all logical conditions. T-SPICE simulations, though, are an absolute necessity for timing and electrical validation. Simple modular piece-wise designs lead to error-free circuit operation and ease of future upgrading.

C. RECOMMENDATIONS FOR FUTURE WORK

Recommendations for future circuit optimization and work include:

- Removing the negative 128 circuit fix in the RCA_CORDIC_ROTATION level by increasing the size of the two's complement circuit by one-bit
- Investigating the addition and subtraction of a constant Z phase of only five-bit numbers. Rather than carrying a 16-bit two's complement fixed-point radix number to accumulate phase values, preliminary work suggests that a five-bit phase value may be sufficient. This would allow the reduction of the 16-bit A/S at each vector stage to be decreased to a five-bit A/S, and the complete elimination of the 9 to 5 Bit Conversion sub-circuit
- Layout of the working circuit must be finished in L-Edit to facilitate circuit fabrication.

APPENDIX A. MATLAB CODE

A. OVERVIEW

This appendix details the author's MATLAB code. The Quine-McClusky code used for the minimization in B. 1. below is detailed in reference [13].

B. QUINE-MCCLUSKY MINIMIZATION

1. File – Main.m

This program calls the Quine-McClusky minimization routine to compute the minimal Sum-of-Products logic function for each of the five conversion output phase bits. The range of minterms from Appendix X are listed as “x#” correspond to the appropriate phases of each bin. The minterms which are never produced are “don't care” terms and are listed as “dc”. Next, the appropriate five functions, f0 through f4, are listed as a matrix with their appropriate range of bin minterms. The routine is run five times, once for each function bit.

```
%-----  
%create matrix of minterms  
  
x0=[0 511 510 509 508 507 506 505 504 503 502 501];  
x1=489:500; x2=478:488; x3=467:477; x4=456:466; x5=445:455; x6=434:444;  
x7=422:433; x8=411:421; x9=400:410; x10=388:399; x11=377:387; x12=366:376;  
x13=354:365; x14=343:353; x15=332:342; x16=321:331; x17=309:320; x18=298:308;  
x19=287:297; x20=276:286; x21=264:275; x22=253:263; x23=242:252; x24=231:241;  
x25=219:230; x26=208:218; x27=197:207; x28=186:196; x29=174:185; x30=163:173;  
x31=153:162;  
  
%create don't care matrix, values of minterms that will never get  
dc=1:152;  
  
f0=[x1 x3 x5 x7 x9 x11 x13 x15 x17 x19 x21 x23 x25 x27 x29 x31 dc];  
f1=[x31 x30 x27 x26 x23 x22 x19 x18 x15 x14 x11 x10 x7 x6 x3 x2 dc];  
f2=[x31 x30 x29 x28 x23 x22 x21 x20 x15 x14 x13 x12 x7 x6 x5 x4 dc];  
f3=[x31 x30 x29 x28 x27 x26 x25 x24 x15 x14 x13 x12 x11 x10 x9 x8 dc];  
f4=[x31 x30 x29 x28 x27 x26 x25 x24 x23 x22 x21 x20 x19 x18 x17 x16 dc];  
  
m0=sort(f4);
```

```
%call quine-mcclusky minimization
r0=quine(m0,5)
```

C. OTHER CODES

1. File – NoiseMargins.m

The following program calculates the slope of the voltage curve, and locates the two places where the derivative is minus one. The location of these two points contains the voltage data value at that index of the MATLAB data, and is used in the noise margin calculations.

```
load INVERTER.M;
Vin=INVERTER(:,1);
Vout=INVERTER(:,2);
Vin=Vin';
Vout=Vout';
Y=diff(Vout);
X=diff(Vout);
Z=Y./X;
Z=Z';
```

2. File – Arctangent.m

The following program is used to generate the $\tan^{-1}(Q/I)$ by either the MATLAB predefined “atan2” function, or by the CORDIC method. The user may specify either method by setting the perfect variable, and if the CORDIC method is used, the number of iterations to perform.

```
function z=arctangent(I,Q,iterations,perfect)

% returns the phase either using cordic or angle(z) depending on perfect flag
% perfect = 0 use cordic approximation
% perfect = ~0 use built in MATLAB atan2 function
% iterations - affects accuracy of the cordic algorithm, max 8!
% sample result and use:
% >> arctangent(1,1,8,0)
%
%ans =
%
% 4

IP = I;           %temp variables used for angle calculation of ArcTan below
QP = Q;
```

```

mp_cordic_table=[1,.5,.25,.125,.0625,.03125,.015625,.007813,3.90625e-3,1.953125e-3];
mp_cordic_table_phase=[.7853981634,.4636030826,.2449597967,.1243547092,.062381
8854,.0312250311,.0156125156,.0077721693,.0038860847];

```

```

if perfect == 0
%rotate by an initial +/- 90 degrees
if (I < 0)
    tmp_I = I;
    if (Q > 0)
        I = Q;                    % subtract 90 degrees
        Q = -tmp_I;
        acc_phase_rads = -pi/2;
    else
        I = -Q;                    % add 90 degrees
        Q = tmp_I;
        acc_phase_rads = pi/2;
    end
else
    acc_phase_rads = 0.0;
end

```

```

% rotate using "1 + jK" factors
for (L = 0:iterations)
K = mp_cordic_table(L+1);
phase_rads = mp_cordic_table_phase(L+1);
tmp_I = I;
if (Q >= 0.0)                    % phase is positive: do negative rotation
    I = I + Q * K;
    Q = Q - tmp_I * K;
    acc_phase_rads = acc_phase_rads - phase_rads;
else                            % phase is negative: do positive rotation
    Q = Q + tmp_I * K;
    acc_phase_rads = acc_phase_rads + phase_rads;
end
L=L+1;
end

```

```

p_phase_degs = -1*acc_phase_rads*180/pi;    %angle is the negative, convert to
degree
phase_quantized = round((p_phase_degs)/11.25);    %quantize to 5 bits
if phase_quantized < 0                    %if negative angle, make offset binary
    phase_quantized = phase_quantized + 32;
end

```

```

else

```

```

%***** "Perfect" Calculation *****
z=IP+i*QP;
phase_quantized = (round(atan2(Q,I)*180/(pi*11.25)));
if phase_quantized <0 %if negative angle, make offset binary
    phase_quantized = phase_quantized + 32;
end
end
z=phase_quantized;
end

```

3. File – Polynomial_Approx.m

The following program was used to plot Figure 8, and via the MATLAB “polyfit” function, determines a polynomial fit to the arctangent function.

```

clear; clc;

I=[-15:-1 0:15];
Q=[-15:-1 0:15];
h=[];

for i = 1:31
    for j=1:31
        if Q(i) == 0
            h(i,j) = inf;
        end

        if (Q(i) == 0)&(I(j)==0)
            h(i,j) = 0;
        end

        if Q(i) ~= 0
            h(i,j)=I(j)/Q(i);
        end
    end
end
%
% figure(1)
% y=floor(atan(h)*180/pi);

% figure(2)
x=-70:70;
a=(atan(x)*180/pi);
plot(x,a)
grid on
axis ([-15,15,-90,90])

```

```
x=(-70:1:70)';  
y=atan(x)*180/pi;  
z=polyfit(x,y,15)  
f=polyval(z,x);  
plot(x,y,'red',x,f,'blue')  
axis ([-80,80,-110,110])  
legend('atan(x)', '15th Degree Polynomial')  
title('Polynomial atan Approximation')  
ylabel('Degrees'),xlabel('x'),grid on
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MOSIS TSMC 0.18 MICRON FET PARAMETERS [14]

A. PROCESS PARAMETERS FILE – TSMC018EPI.MD

```
* MOSIS PARAMETRIC TEST RESULTS
* RUN: T15J (LO_EPI)
* TECHNOLOGY: SCN018
microns
* T15J SPICE BSIM3 VERSION 3.1 PARAMETERS
* SPICE 3f5 Level 8, Star-HSPICE Level 49, UTMOST Level 8
* DATE: Jul 16/01
* LOT: T15J
* Temperature_parameters=Default
```

```
VENDOR: TSMC
FEATURE SIZE: 0.18
```

```
.MODEL CMOSN NMOS (
+VERSION = 3.1          TNOM    = 27          TOX      = 4.2E-9
+XJ        = 1E-7       NCH     = 2.3549E17   VTH0     = 0.3593426
+K1        = 0.584235   K2      = 1.808939E-3  K3       = 1E-3
+K3B       = 15.9142604 W0      = 6.767602E-6  NLX      = 1.645593E-7
+DVT0W     = 0          DVT1W   = 0          DVT2W    = 0
+DVT0      = 1.3712712  DVT1   = 0.4653446  DVT2     = -0.0430942
+U0        = 319.668247 UA      = -2.46952E-10 UB      = 6.893182E-
19
+UC        = -4.23662E-11 VS.AT   = 9.798045E4   A0       = 1.4231374
+AGS       = 0.1896218  B0     = -1.429899E-8  B1       = -1E-7
+KETA      = 0.0270338  A1     = 5.615435E-4  A2       = 0.8500947
+RDSW     = 133.2722527 PRWG    = 0.5        PRWB     = -0.2
+WR        = 1          WINT     = 0          LINT     = 9.682918E-9
+XL        = -2E-8      XW      = -1E-8      DWG      = -7.78854E-9
+DWB       = -1.003184E-8 VOFF   = -0.0652789  NFACTOR  = 2.5
+CIT       = 0          CDSC    = 2.4E-4      CDSCD    = 0
+CDSCB     = 0          ETA0    = 0.1006785  ETAB     = -0.0446167
+DSUB      = 0.8210518  PCLM   = 0.7765536  PDIBLC1  = 0.1854406
+PDIBLC2   = 9.865273E-3 PDIBLCB = -0.0540508  DROUT    = 0.8266372
+PSCBE1    = 7.672864E10 PSCBE2  = 2.036021E-8 PVAG     = 0
+DELTA     = 0.01       RSH     = 6.8        MOBMOD   = 1
+PRT       = 0          UTE     = -1.5      KT1      = -0.11
+KT1L      = 0          KT2     = 0.022     UA1      = 4.31E-9
+UB1       = -7.61E-18  UC1     = -5.6E-11  AT       = 3.3E4
+WL        = 0          WLN     = 1        WW       = 0
+WWN       = 1          WWL     = 0        LL      = 0
+LLN       = 1          LW      = 0        LWN     = 1
+LWL       = 0          CAPMOD  = 2        XPART    = 0.5
+CGD0      = 7.23E-10   CGS0    = 7.23E-10  CGBO     = 1E-12
+CJ        = 9.89627E-4 PB      = 0.73534   MJ       = 0.3594267
+CJSW      = 2.46165E-10 PBSW    = 0.7840557  MJSW     = 0.1075765
+CJSWG     = 3.3E-10   PBSWG   = 0.7840557  MJSWG    = 0.1075765
+CF        = 0          PVTH0   = -3.498648E-5 PRDSW    = -2.9489679
+PK2       = -1.251474E-3 WKETA   = 1.928603E-3 LKETA    = -8.378587E-
3
+PU0       = 31.1137209 PUA     = 1.155019E-10 PUB      = 0
+PVS.AT    = 1.542088E3 PETA0   = -1.003159E-4 PKETA    = 5.130701E-
3 )
```

```

.MODEL CMOS PMOS (
+VERSION = 3.1
+XJ      = 1E-7
+K1      = 0.5684869
+K3B     = 10.6033883
+DVT0W   = 0
+DVT0    = 0.5244177
+U0      = 124.8628741
+UC      = -1E-10
+AGS     = 0.3427925
+KETA    = 0.0212022
+RDSW    = 304.979313
+WR      = 1
+XL      = -2E-8
8
+DWB     = 5.971841E-9
+CIT     = 0
+CDSCB   = 0
+DSUB    = 1.2865683
+PDIBLC2 = 0.0508323
+PSCBE1  = 1.733444E9
+DELTA   = 0.01
+PRT     = 0
+KT1L    = 0
+UB1     = -7.61E-18
+WL      = 0
+WWN     = 1
+LLN     = 1
+LWL     = 0
+CGD0    = 6.92E-10
+CJ      = 1.204978E-3
+CJSW    = 2.088728E-10
+CJSWG   = 4.22E-10
+CF      = 0
+PK2     = 2.629498E-3
3
+PU0     = -2.2589171
22
+PVS.AT  = -50
3      )

TNOM     = 27
NCH      = 4.1589E17
K2       = 0.0351909
W0       = 1E-6
DVT1W    = 0
DVT1     = 0.2901433
UA       = 1.792035E-9
VS.AT    = 1.551654E5
B0       = 1.666904E-6
A1       = 0.028014
PRWG     = 0.5
WINT     = 0
XW       = -1E-8

VOFF     = -0.100662
CDSC     = 2.4E-4
ETA0     = 0.2098261
PCLM     = 2.544679
PDIBLCB  = -9.99311E-4
PSCBE2   = 5.00159E-10
RSH      = 7.6
UTE      = -1.5
KT2      = 0.022
UC1      = -5.6E-11
WLN      = 1
WWL      = 0
LW       = 0
CAPMOD   = 2
CGS0     = 6.92E-10
PB       = 0.8428469
PBSW     = 0.5832884
PBSWG    = 0.5832884
PVTH0    = 2.844904E-3
WKETA    = 2.438155E-3

PUA      = -7.99545E-11
PETA0    = 1E-4

LEVEL    = 49
TOX      = 4.2E-9
VTH0     = -0.4139661
K3       = 0
NLX      = 9.038631E-8
DVT2W    = 0
DVT2     = 0.1
UB       = 1E-21
A0       = 1.5201757
B1       = 5E-6
A2       = 1
PRWB     = -0.5
LINT     = 2.053267E-8
DWG      = -3.938518E-8

NFACTOR  = 1.9470845
CDSCD    = 0
ETAB     = -0.2406335
PDIBLC1  = 6.316635E-3
DROUT    = 0
PVAG     = 15
MOBMOD   = 1
KT1      = -0.11
UA1      = 4.31E-9
AT       = 3.3E4
WW       = 0
LL       = 0
LWN      = 1
XPART    = 0.5
CGB0     = 1E-12
MJ       = 0.4043249
MJSW     = 0.3016152
MJSWG    = 0.3016152
PRDSW    = 6.5073202
LKETA    = -4.928775E-3

PUB      = 2.472552E-3
PKETA    = 2.018007E-3

```

APPENDIX C. 9-TO-5 BIT CONVERSION – MINTERM CALCULATION

A. OVERVIEW

Using Excel, all Z phases were listed in Table 29, along with their corresponding negative value, the Actual Phase, as a reference (recall the CORDIC give the negative of the actual phase). The Z Decimal was then converted to binary which represents its minterm. For example:

$-4^\circ = 1\ 1111\ 1100$ in two's complement = 508 as an unsigned binary number.

The data Z8 through Z0 show this minterm representation in binary. Phases, and thus minterms, were then grouped IAW Table 26. and the five bit corresponding number, F4 through F0 was entered according to the Actual Phase value. For the case of the Z Decimal value of -4° , actual phase of 4° this would correspond to a five-bit phase number of 0 as it falls in the range of 0° to 11° . Thus, the table automatically converts the Z Decimal phase to its complement number via the method it is laid out and encoded. By comparing when a particular F bit was one, the sum of products minterms were then determined and entered into Table 27. As well, the “don’t care” terms were determined by examining which minterms were never used because they represent a phase value larger than 360 degrees, which by definition, is impossible. The “don’t cares” were thus determined to be the minterm values between 1 and 152.

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	-1	511	1	1	1	1	1	1	1	1	1					
2	-2	510	1	1	1	1	1	1	1	1	0					
3	-3	509	1	1	1	1	1	1	1	0	1					
4	-4	508	1	1	1	1	1	1	1	0	0					
5	-5	507	1	1	1	1	1	1	0	1	1					
6	-6	506	1	1	1	1	1	1	0	1	0		0			
7	-7	505	1	1	1	1	1	1	0	0	1					
8	-8	504	1	1	1	1	1	1	0	0	0					
9	-9	503	1	1	1	1	1	0	1	1	1					
10	-10	502	1	1	1	1	1	0	1	1	0					
11	-11	501	1	1	1	1	1	0	1	0	1					
12	-12	500	1	1	1	1	1	0	1	0	0	0	0	0	0	1
13	-13	499	1	1	1	1	1	0	0	1	1					
14	-14	498	1	1	1	1	1	0	0	1	0					
15	-15	497	1	1	1	1	1	0	0	0	1					
16	-16	496	1	1	1	1	1	0	0	0	0					
17	-17	495	1	1	1	1	0	1	1	1	1					
18	-18	494	1	1	1	1	0	1	1	1	0		1			
19	-19	493	1	1	1	1	0	1	1	0	1					
20	-20	492	1	1	1	1	0	1	1	0	0					
21	-21	491	1	1	1	1	0	1	0	1	1					
22	-22	490	1	1	1	1	0	1	0	1	0					
23	-23	489	1	1	1	1	0	1	0	0	1					
24	-24	488	1	1	1	1	0	1	0	0	0	0	0	0	1	0
25	-25	487	1	1	1	1	0	0	1	1	1					
26	-26	486	1	1	1	1	0	0	1	1	0					
27	-27	485	1	1	1	1	0	0	1	0	1					
28	-28	484	1	1	1	1	0	0	1	0	0					
29	-29	483	1	1	1	1	0	0	0	1	1		2			
30	-30	482	1	1	1	1	0	0	0	1	0					
31	-31	481	1	1	1	1	0	0	0	0	1					
32	-32	480	1	1	1	1	0	0	0	0	0					
33	-33	479	1	1	1	0	1	1	1	1	1					
34	-34	478	1	1	1	0	1	1	1	1	0					
35	-35	477	1	1	1	0	1	1	1	0	1	0	0	0	1	1
36	-36	476	1	1	1	0	1	1	1	0	0					
37	-37	475	1	1	1	0	1	1	0	1	1					
38	-38	474	1	1	1	0	1	1	0	1	0					
39	-39	473	1	1	1	0	1	1	0	0	1					
40	-40	472	1	1	1	0	1	1	0	0	0					
41	-41	471	1	1	1	0	1	0	1	1	1		3			
42	-42	470	1	1	1	0	1	0	1	1	0					
43	-43	469	1	1	1	0	1	0	1	0	1					
44	-44	468	1	1	1	0	1	0	1	0	0					
45	-45	467	1	1	1	0	1	0	0	1	1					

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
46	-46	466	1	1	1	0	1	0	0	1	0	0	0	1	0	0
47	-47	465	1	1	1	0	1	0	0	0	1					
48	-48	464	1	1	1	0	1	0	0	0	0					
49	-49	463	1	1	1	0	0	1	1	1	1					
50	-50	462	1	1	1	0	0	1	1	1	0					
51	-51	461	1	1	1	0	0	1	1	0	1					
52	-52	460	1	1	1	0	0	1	1	0	0		4			
53	-53	459	1	1	1	0	0	1	0	1	1					
54	-54	458	1	1	1	0	0	1	0	1	0					
55	-55	457	1	1	1	0	0	1	0	0	1					
56	-56	456	1	1	1	0	0	1	0	0	0					
57	-57	455	1	1	1	0	0	0	1	1	1	0	0	1	0	1
58	-58	454	1	1	1	0	0	0	1	1	0					
59	-59	453	1	1	1	0	0	0	1	0	1					
60	-60	452	1	1	1	0	0	0	1	0	0					
61	-61	451	1	1	1	0	0	0	0	1	1					
62	-62	450	1	1	1	0	0	0	0	1	0		5			
63	-63	449	1	1	1	0	0	0	0	0	1					
64	-64	448	1	1	1	0	0	0	0	0	0					
65	-65	447	1	1	0	1	1	1	1	1	1					
66	-66	446	1	1	0	1	1	1	1	1	0					
67	-67	445	1	1	0	1	1	1	1	0	1					
68	-68	444	1	1	0	1	1	1	1	0	0					
69	-69	443	1	1	0	1	1	1	0	1	1	0	0	1	1	0
70	-70	442	1	1	0	1	1	1	0	1	0					
71	-71	441	1	1	0	1	1	1	0	0	1					
72	-72	440	1	1	0	1	1	1	0	0	0					
73	-73	439	1	1	0	1	1	0	1	1	1					
74	-74	438	1	1	0	1	1	0	1	1	0					
75	-75	437	1	1	0	1	1	0	1	0	1		6			
76	-76	436	1	1	0	1	1	0	1	0	0					
77	-77	435	1	1	0	1	1	0	0	1	1					
78	-78	434	1	1	0	1	1	0	0	1	0					
79	-79	433	1	1	0	1	1	0	0	0	1					
80	-80	432	1	1	0	1	1	0	0	0	0	0	0	1	1	1
81	-81	431	1	1	0	1	0	1	1	1	1					
82	-82	430	1	1	0	1	0	1	1	1	0					
83	-83	429	1	1	0	1	0	1	1	0	1					
84	-84	428	1	1	0	1	0	1	1	0	0					
85	-85	427	1	1	0	1	0	1	0	1	1		7			
86	-86	426	1	1	0	1	0	1	0	1	0					
87	-87	425	1	1	0	1	0	1	0	0	1					
88	-88	424	1	1	0	1	0	1	0	0	0					
89	-89	423	1	1	0	1	0	0	1	1	1					
90	-90	422	1	1	0	1	0	0	1	1	0					
91	-91	421	1	1	0	1	0	0	1	0	1	0	1	0	0	0
92	-92	420	1	1	0	1	0	0	1	0	0					
93	-93	419	1	1	0	1	0	0	0	1	1					
94	-94	418	1	1	0	1	0	0	0	1	0					
95	-95	417	1	1	0	1	0	0	0	0	1					
96	-96	416	1	1	0	1	0	0	0	0	0					
97	-97	415	1	1	0	0	1	1	1	1	1		8			
98	-98	414	1	1	0	0	1	1	1	1	0					
99	-99	413	1	1	0	0	1	1	1	0	1					
100	-100	412	1	1	0	0	1	1	1	0	0					
101	-101	411	1	1	0	0	1	1	0	1	1					

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
102	-102	410	1	1	0	0	1	1	0	1	0	0	1	0	0	1
103	-103	409	1	1	0	0	1	1	0	0	1					
104	-104	408	1	1	0	0	1	1	0	0	0					
105	-105	407	1	1	0	0	1	0	1	1	1					
106	-106	406	1	1	0	0	1	0	1	1	0					
107	-107	405	1	1	0	0	1	0	1	0	1					
108	-108	404	1	1	0	0	1	0	1	0	0		9			
109	-109	403	1	1	0	0	1	0	0	1	1					
110	-110	402	1	1	0	0	1	0	0	1	0					
111	-111	401	1	1	0	0	1	0	0	0	1					
112	-112	400	1	1	0	0	1	0	0	0	0					
113	-113	399	1	1	0	0	0	1	1	1	1	0	1	0	1	0
114	-114	398	1	1	0	0	0	1	1	1	0					
115	-115	397	1	1	0	0	0	1	1	0	1					
116	-116	396	1	1	0	0	0	1	1	0	0					
117	-117	395	1	1	0	0	0	1	0	1	1					
118	-118	394	1	1	0	0	0	1	0	1	0		10			
119	-119	393	1	1	0	0	0	1	0	0	1					
120	-120	392	1	1	0	0	0	1	0	0	0					
121	-121	391	1	1	0	0	0	0	1	1	1					
122	-122	390	1	1	0	0	0	0	1	1	0					
123	-123	389	1	1	0	0	0	0	1	0	1					
124	-124	388	1	1	0	0	0	0	1	0	0					
125	-125	387	1	1	0	0	0	0	0	1	1	0	1	0	1	1
126	-126	386	1	1	0	0	0	0	0	1	0					
127	-127	385	1	1	0	0	0	0	0	0	1					
128	-128	384	1	1	0	0	0	0	0	0	0					
129	-129	383	1	0	1	1	1	1	1	1	1					
130	-130	382	1	0	1	1	1	1	1	1	0					
131	-131	381	1	0	1	1	1	1	1	0	1					
132	-132	380	1	0	1	1	1	1	1	0	0		11			
133	-133	379	1	0	1	1	1	1	0	1	1					
134	-134	378	1	0	1	1	1	1	0	1	0					
135	-135	377	1	0	1	1	1	1	0	0	1					
136	-136	376	1	0	1	1	1	1	0	0	0	0	1	1	0	0
137	-137	375	1	0	1	1	1	0	1	1	1					
138	-138	374	1	0	1	1	1	0	1	1	0					
139	-139	373	1	0	1	1	1	0	1	0	1					
140	-140	372	1	0	1	1	1	0	1	0	0		12			
141	-141	371	1	0	1	1	1	0	0	1	1					
142	-142	370	1	0	1	1	1	0	0	1	0					
143	-143	369	1	0	1	1	1	0	0	0	1					
144	-144	368	1	0	1	1	1	0	0	0	0					
145	-145	367	1	0	1	1	0	1	1	1	1					
146	-146	366	1	0	1	1	0	1	1	1	0					
147	-147	365	1	0	1	1	0	1	1	0	1	0	1	1	0	1
148	-148	364	1	0	1	1	0	1	1	0	0					
149	-149	363	1	0	1	1	0	1	0	1	1					
150	-150	362	1	0	1	1	0	1	0	1	0					
151	-151	361	1	0	1	1	0	1	0	0	1					
152	-152	360	1	0	1	1	0	1	0	0	0					
153	-153	359	1	0	1	1	0	0	1	1	1		13			
154	-154	358	1	0	1	1	0	0	1	1	0					
155	-155	357	1	0	1	1	0	0	1	0	1					
156	-156	356	1	0	1	1	0	0	1	0	0					
157	-157	355	1	0	1	1	0	0	0	1	1					
158	-158	354	1	0	1	1	0	0	0	1	0					

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
159	-159	353	1	0	1	1	0	0	0	0	1	0	1	1	1	0
160	-160	352	1	0	1	1	0	0	0	0	0					
161	-161	351	1	0	1	0	1	1	1	1	1					
162	-162	350	1	0	1	0	1	1	1	1	0		14			
163	-163	349	1	0	1	0	1	1	1	0	1					
164	-164	348	1	0	1	0	1	1	1	0	0		343-353	343-353	343-353	
165	-165	347	1	0	1	0	1	1	0	1	1					
166	-166	346	1	0	1	0	1	1	0	1	0					
167	-167	345	1	0	1	0	1	1	0	0	1					
168	-168	344	1	0	1	0	1	1	0	0	0					
169	-169	343	1	0	1	0	1	0	1	1	1					
170	-170	342	1	0	1	0	1	0	1	1	0	0	1	1	1	1
171	-171	341	1	0	1	0	1	0	1	0	1					
172	-172	340	1	0	1	0	1	0	1	0	0		15			
173	-173	339	1	0	1	0	1	0	0	1	1					
174	-174	338	1	0	1	0	1	0	0	1	0					
175	-175	337	1	0	1	0	1	0	0	0	1					
176	-176	336	1	0	1	0	1	0	0	0	0					
177	-177	335	1	0	1	0	0	1	1	1	1					
178	-178	334	1	0	1	0	0	1	1	1	0					
179	-179	333	1	0	1	0	0	1	1	0	1					
180	-180	332	1	0	1	0	0	1	1	0	0					
359	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1
358	2	2	0	0	0	0	0	0	0	1	0					
357	3	3	0	0	0	0	0	0	0	1	1		31			
356	4	4	0	0	0	0	0	0	1	0	0					
355	5	5	0	0	0	0	0	0	1	0	1					
354	6	6	0	0	0	0	0	0	1	1	0					
353	7	7	0	0	0	0	0	0	1	1	1					
352	8	8	0	0	0	0	0	1	0	0	0					
351	9	9	0	0	0	0	0	1	0	0	1					
350	10	10	0	0	0	0	0	1	0	1	0					
349	11	11	0	0	0	0	0	1	0	1	1	1	1	1	1	0
348	12	12	0	0	0	0	0	1	1	0	0					
347	13	13	0	0	0	0	0	1	1	0	1		30			
346	14	14	0	0	0	0	0	1	1	1	0					
345	15	15	0	0	0	0	0	1	1	1	1					
344	16	16	0	0	0	0	1	0	0	0	0					
343	17	17	0	0	0	0	1	0	0	0	1					
342	18	18	0	0	0	0	1	0	0	1	0					
341	19	19	0	0	0	0	1	0	0	1	1					
340	20	20	0	0	0	0	1	0	1	0	0					
339	21	21	0	0	0	0	1	0	1	0	1					
338	22	22	0	0	0	0	1	0	1	1	0	1	1	1	0	1
337	23	23	0	0	0	0	1	0	1	1	1					
336	24	24	0	0	0	0	1	1	0	0	0					
335	25	25	0	0	0	0	1	1	0	0	1		29			
334	26	26	0	0	0	0	1	1	0	1	0					
333	27	27	0	0	0	0	1	1	0	1	1					
332	28	28	0	0	0	0	1	1	1	0	0					
331	29	29	0	0	0	0	1	1	1	0	1					
330	30	30	0	0	0	0	1	1	1	1	0					
329	31	31	0	0	0	0	1	1	1	1	1					
328	32	32	0	0	0	1	0	0	0	0	0					
327	33	33	0	0	0	1	0	0	0	0	1					

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
326	34	34	0	0	0	1	0	0	0	1	0	1	1	1	0	0
325	35	35	0	0	0	1	0	0	0	1	1					
324	36	36	0	0	0	1	0	0	1	0	0					
323	37	37	0	0	0	1	0	0	1	0	1		28			
322	38	38	0	0	0	1	0	0	1	1	0					
321	39	39	0	0	0	1	0	0	1	1	1					
320	40	40	0	0	0	1	0	1	0	0	0					
319	41	41	0	0	0	1	0	1	0	0	1					
318	42	42	0	0	0	1	0	1	0	1	0					
317	43	43	0	0	0	1	0	1	0	1	1					
316	44	44	0	0	0	1	0	1	1	0	0					
315	45	45	0	0	0	1	0	1	1	0	1	1	1	0	1	1
314	46	46	0	0	0	1	0	1	1	1	0					
313	47	47	0	0	0	1	0	1	1	1	1					
312	48	48	0	0	0	1	1	0	0	0	0		27			
311	49	49	0	0	0	1	1	0	0	0	1					
310	50	50	0	0	0	1	1	0	0	1	0					
309	51	51	0	0	0	1	1	0	0	1	1					
308	52	52	0	0	0	1	1	0	1	0	0					
307	53	53	0	0	0	1	1	0	1	0	1					
306	54	54	0	0	0	1	1	0	1	1	0					
305	55	55	0	0	0	1	1	0	1	1	1					
304	56	56	0	0	0	1	1	1	0	0	0	1	1	0	1	0
303	57	57	0	0	0	1	1	1	0	0	1					
302	58	58	0	0	0	1	1	1	0	1	0					
301	59	59	0	0	0	1	1	1	0	1	1		26			
300	60	60	0	0	0	1	1	1	1	0	0					
299	61	61	0	0	0	1	1	1	1	0	1					
298	62	62	0	0	0	1	1	1	1	1	0					
297	63	63	0	0	0	1	1	1	1	1	1					
296	64	64	0	0	1	0	0	0	0	0	0					
295	65	65	0	0	1	0	0	0	0	0	1					
294	66	66	0	0	1	0	0	0	0	1	0					
293	67	67	0	0	1	0	0	0	0	1	1	1	1	0	0	1
292	68	68	0	0	1	0	0	0	1	0	0					
291	69	69	0	0	1	0	0	0	1	0	1					
290	70	70	0	0	1	0	0	0	1	1	0					
289	71	71	0	0	1	0	0	0	1	1	1		25			
288	72	72	0	0	1	0	0	1	0	0	0					
287	73	73	0	0	1	0	0	1	0	0	1					
286	74	74	0	0	1	0	0	1	0	1	0					
285	75	75	0	0	1	0	0	1	0	1	1					
284	76	76	0	0	1	0	0	1	1	0	0					
283	77	77	0	0	1	0	0	1	1	0	1					
282	78	78	0	0	1	0	0	1	1	1	0					
281	79	79	0	0	1	0	0	1	1	1	1	1	1	0	0	0
280	80	80	0	0	1	0	1	0	0	0	0					
279	81	81	0	0	1	0	1	0	0	0	1					
278	82	82	0	0	1	0	1	0	0	1	0		24			
277	83	83	0	0	1	0	1	0	0	1	1					
276	84	84	0	0	1	0	1	0	1	0	0					
275	85	85	0	0	1	0	1	0	1	0	1					
274	86	86	0	0	1	0	1	0	1	1	0					
273	87	87	0	0	1	0	1	0	1	1	1					
272	88	88	0	0	1	0	1	1	0	0	0					
271	89	89	0	0	1	0	1	1	0	0	1					

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
270	90	90	0	0	1	0	1	1	0	1	0	1	0	1	1	1
269	91	91	0	0	1	0	1	1	0	1	1					
268	92	92	0	0	1	0	1	1	1	0	0					
267	93	93	0	0	1	0	1	1	1	0	1					
266	94	94	0	0	1	0	1	1	1	1	0					
265	95	95	0	0	1	0	1	1	1	1	1		23			
264	96	96	0	0	1	1	0	0	0	0	0					
263	97	97	0	0	1	1	0	0	0	0	1					
262	98	98	0	0	1	1	0	0	0	1	0					
261	99	99	0	0	1	1	0	0	0	1	1					
260	100	100	0	0	1	1	0	0	1	0	0					
259	101	101	0	0	1	1	0	0	1	0	1	1	0	1	1	0
258	102	102	0	0	1	1	0	0	1	1	0					
257	103	103	0	0	1	1	0	0	1	1	1					
256	104	104	0	0	1	1	0	1	0	0	0					
255	105	105	0	0	1	1	0	1	0	0	1					
254	106	106	0	0	1	1	0	1	0	1	0					
253	107	107	0	0	1	1	0	1	0	1	1		22			
252	108	108	0	0	1	1	0	1	1	0	0					
251	109	109	0	0	1	1	0	1	1	0	1					
250	110	110	0	0	1	1	0	1	1	1	0					
249	111	111	0	0	1	1	0	1	1	1	1					
248	112	112	0	0	1	1	1	0	0	0	0	1	0	1	0	1
247	113	113	0	0	1	1	1	0	0	0	1					
246	114	114	0	0	1	1	1	0	0	1	0					
245	115	115	0	0	1	1	1	0	0	1	1					
244	116	116	0	0	1	1	1	0	1	0	0					
243	117	117	0	0	1	1	1	0	1	0	1					
242	118	118	0	0	1	1	1	0	1	1	0		21			
241	119	119	0	0	1	1	1	0	1	1	1					
240	120	120	0	0	1	1	1	1	0	0	0					
239	121	121	0	0	1	1	1	1	0	0	1					
238	122	122	0	0	1	1	1	1	0	1	0					
237	123	123	0	0	1	1	1	1	0	1	1					
236	124	124	0	0	1	1	1	1	1	0	0	1	0	1	0	0
235	125	125	0	0	1	1	1	1	1	0	1					
234	126	126	0	0	1	1	1	1	1	1	0					
233	127	127	0	0	1	1	1	1	1	1	1					
232	128	128	0	1	0	0	0	0	0	0	0		20			
231	129	129	0	1	0	0	0	0	0	0	1					
230	130	130	0	1	0	0	0	0	0	1	0					
229	131	131	0	1	0	0	0	0	0	1	1					
228	132	132	0	1	0	0	0	0	1	0	0					
227	133	133	0	1	0	0	0	0	1	0	1					
226	134	134	0	1	0	0	0	0	1	1	0					
225	135	135	0	1	0	0	0	0	1	1	1	1	0	0	1	1
224	136	136	0	1	0	0	0	1	0	0	0					
223	137	137	0	1	0	0	0	1	0	0	1					
222	138	138	0	1	0	0	0	1	0	1	0					
221	139	139	0	1	0	0	0	1	0	1	1		19			
220	140	140	0	1	0	0	0	1	1	0	0					
219	141	141	0	1	0	0	0	1	1	0	1					
218	142	142	0	1	0	0	0	1	1	1	0					
217	143	143	0	1	0	0	0	1	1	1	1					
216	144	144	0	1	0	0	1	0	0	0	0					
215	145	145	0	1	0	0	1	0	0	0	1					

Actual Phase	Z Decimal	Unsigned Binary	Z8	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	F4	F3	F2	F1	F0
214	146	146	0	1	0	0	1	0	0	1	0	1	0	0	1	0
213	147	147	0	1	0	0	1	0	0	1	1					
212	148	148	0	1	0	0	1	0	1	0	0					
211	149	149	0	1	0	0	1	0	1	0	1					
210	150	150	0	1	0	0	1	0	1	1	0		18			
209	151	151	0	1	0	0	1	0	1	1	1					
208	152	152	0	1	0	0	1	1	0	0	0					
207	153	153	0	1	0	0	1	1	0	0	1					
206	154	154	0	1	0	0	1	1	0	1	0					
205	155	155	0	1	0	0	1	1	0	1	1					
204	156	156	0	1	0	0	1	1	1	0	0					
203	157	157	0	1	0	0	1	1	1	0	1	1	0	0	0	1
202	158	158	0	1	0	0	1	1	1	1	0					
201	159	159	0	1	0	0	1	1	1	1	1					
200	160	160	0	1	0	1	0	0	0	0	0					
199	161	161	0	1	0	1	0	0	0	0	1					
198	162	162	0	1	0	1	0	0	0	1	0		17			
197	163	163	0	1	0	1	0	0	0	1	1					
196	164	164	0	1	0	1	0	0	1	0	0					
195	165	165	0	1	0	1	0	0	1	0	1					
194	166	166	0	1	0	1	0	0	1	1	0					
193	167	167	0	1	0	1	0	0	1	1	1					
192	168	168	0	1	0	1	0	1	0	0	0					
191	169	169	0	1	0	1	0	1	0	0	1	1	0	0	0	0
190	170	170	0	1	0	1	0	1	0	1	0					
189	171	171	0	1	0	1	0	1	0	1	1					
188	172	172	0	1	0	1	0	1	1	0	0					
187	173	173	0	1	0	1	0	1	1	0	1					
186	174	174	0	1	0	1	0	1	1	1	0		16			
185	175	175	0	1	0	1	0	1	1	1	1					
184	176	176	0	1	0	1	1	0	0	0	0					
183	177	177	0	1	0	1	1	0	0	0	1					
182	178	178	0	1	0	1	1	0	0	1	0					
181	179	179	0	1	0	1	1	0	0	1	1					

Table 29. 9-to-5 Bit Phase Conversion Truth Table.

APPENDIX D. PROCESS TECHNOLOGY

A. OVERVIEW

The following is a reproduction from the MOSIS website and provides a general description of the fabrication processes and rules [15].

B. MOSIS PROCESSES

1. Overview

This CMOS process has 6 metal layers and 1 poly layer. The process is for 1.8 volt applications. A thick oxide layer can be used for 3.3 volt transistors. MOSIS multiproject runs support designs for the 0.18 micron CMOS logic process (CL018) using epitaxial wafers, and mixed signal/RF process (CM018) using non-epitaxial wafers.

Silicide block, thick gate oxide (3.3 V), ESD 3.3 V, NT_N, deep n_well, ThickTopMetal (inductor), and MiM options are available on multiproject runs. The Thick_Top_Metal option must be explicitly specified with each design submission that requires it. MiM (Cap_Top_Metal, also known as Metal 5 Prime, to Metal 5) provides a capacitance of 1 fF/ μm^2 . Designs for this process require Metal 6 in the pad stack.

MOSIS Scalable CMOS (SCMOS) is a set of logical layers together with their design rules, which provide a nearly process- and metric-independent interface to all CMOS fabrication processes available through MOSIS. The designer works in the abstract SCMOS layers and metric unit ("lambda"). He then specifies which process and feature size he wants the design to be fabricated in. MOSIS maps the SCMOS design onto that process, generating the true logical layers and absolute dimensions required by the process vendor. The designer can often submit exactly the same design, but to a different fabrication process or feature size. MOSIS alone handles the new mapping.

By contrast, using a specific vendor's layers and design rules ("vendor rules") will yield a design which is less likely to be directly portable to any other process or feature size. Vendor rules usually need more logical layers than the SCMOS rules, even though both fabricate onto exactly the same process. More layers means more design rules, a higher learning curve for that one process, more interactions to worry about, more complex design support required, and longer layout development times. Porting the design to a new process will be burdensome.

SCMOS designers access process-specific features by using MOSIS-provided abstract layers which implement those features. For example, a designer wishing to use second-poly would use the MOSIS-provided second-poly abstract layer, but must then submit to a process providing for two polysilicon layers. In the same way, designers may access multiple metals, or different types of analog structures such as capacitors and resistors, without having to learn any new set of design rules for the more standard layers such as metal-1.

Vendor rules may be more appropriate when seeking maximal use of silicon area, more direct control over analog circuit parameters, or for very large production runs, where the added investment in development time and loss of design portability is clearly justified. However the advantages of using SCMOS rules may far outweigh such concerns, and should be considered.

2. SCMOS Design Rules

In the SCMOS rules, circuit geometries are specified in the Mead and Conway's lambda based methodology. The unit of measurement, lambda, can easily be scaled to different fabrication processes as semiconductor technology advances.

Each design has a technology-code associated with the layout file. Each technology-code may have one or more associated options added for the purpose of specifying either (a) special features for the target process or (b) the presence of novel devices in the design. At the time of this revision, MOSIS is offering CMOS processes with feature sizes from 1.5 micron to 0.18 micron.

3. Standard CMOS

The standard CMOS technology accessed by MOSIS is a single polysilicon, double metal, bulk CMOS process with enhancement-mode *n*-MOSFET and *p*-MOSFET devices.

4. Well Type

The Scalable CMOS (SC) rules support both *n*-well and *p*-well processes. MOSIS recognizes three base technology codes that let the designer specify the well type of the process selected. SCN specifies an *n*-well process, SCP specifies a *p*-well process, and SCE indicates that the designer is willing to utilize a process of either *n*-well or *p*-well.

An SCE design must provide both a drawn *n*-well and a drawn *p*-well; MOSIS will use the well that corresponds to the selected process and ignore the other well. As a convenience, SCN and SCP designs may also include the other

well (p -well in an SCN design or n -well in an SCP design), but it will always be ignored.

MOSIS currently offers only n -well processes or foundry-designated twin-well processes that from the design and process flow standpoints are equivalent to n -well processes. These twin-well processes may have options (deep n -well) that provide independently isolated p -wells. For all of these processes at this time use the technology code SCN. SCP is currently not supported, and SCE is treated exactly as SCN.

SCN6M_DEEP: Scalable CMOS N-well, 6 metal, 1 poly, thick oxide option, and supports silicide block. MiM (Cap_Top_Metal, also known as Metal 5 Prime, to Metal 5) capacitors are available. Uses revised layout rules for better fit to sub-micron processes.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. TANNER TOOLS DESCRIPTION

A. OVERVIEW

The following is a reproduction from the Tanner website and provides a general description of the suite of Tanner Tools Pro [3].

B. TANNER TOOLS

1. Simulation Tools

Analog Circuit Simulator

T-Spice Pro™ offers fast and accurate simulation for analog and mixed analog/digital circuits. Full chip designs where more than 300,000 elements can be simulated. T-Spice includes standard SPICE models like the latest BSIM3 models, and the advanced Maher/Mead model which scales to submicron lengths and is continuous from subthreshold to above-threshold operation.

Waveform Viewer

W-Edit streamlines and customizes graphical data representation using data files without modification from T-Spice and GateSim simulation runs.

2. Frontend and Netlist

Layout vs. Schematic

LVS. accurately and efficiently compares two SPICE netlists. Element and node mismatches are quickly traced back to their origins and unresolvable nodes and devices are pinpointed. When trail matching is turned on, LVS. attempts to resolve ambiguous elements and nodes by assigning matches between a pair of elements or nodes. LVS. can use topological information, parametric values, or geometric values to compare netlists with a specified tolerance. The ability to specify pre- and post-iteration matching or parameter matching speeds up the comparison process. Other time saving features include the ability to queue and run verification in batch mode.

3. Mask-Level Tools

Layout Editor

L-Edit™ is a full-featured, high-performance, interactive graphical mask layout editor. L-Edit generates layouts quickly and easily, supports fully hierarchical designs, and allows an unlimited number of layers, cells, and levels of hierarchy. It includes all major drawing primitives and supports 90-degree, 45-degree, and all-angle drawing modes. L-Edit offers advanced editing features such as edit-in-place, slice/merge, group/ungroup, window stretch editing, and reads

and writes GDS II and CIF file formats. L-Edit also includes a unique cross-section viewer that allows you to simulate and preview grow/deposit, implant/diffuse, and etch steps.

Design Rule Checking

L-Edit/DRC™ is a user-configurable design rule checker that can verify a full chip or just a specific region. Errors can be collated in a text file or reported on screen using error objects or error labels representing a description of the violated rule. Design rule setup uses lambda units that allow for easy rescaling for new technologies. The domain decomposition algorithm enables rapid checking of large designs. Easy portability across platforms allows you to move large DRC runs to higher performance or multitasking hardware.

Device Extraction

L-Edit/Extract™ creates SPICE-compatible circuit netlists from L-Edit layouts. It can recognize active and passive devices, subcircuits, and most common device parameters, including resistance, capacitance, device length, width, and extension rules. Full chip and region-only DRC is supported. DRC offers Error Browser and Object Browser functions for quickly and easily cycling through rule-checking errors, and supports 45-degree and 90-degree geometry.

LIST OF REFERENCES

1. Fouts, D. J., Pace, P.E., Karow, C., Ekestorm, S., “*A Single-Chip False Target Radar Image Generator for Countering Wideband Imaging Radars*”, IEEE Journal on Solid State Circuits, Vol. 37, No. 6, June 2002.
2. Pace, P. E., Fouts, D. J., Karow, C., Ekestorm, S., “*An All-Digital Image Synthesizer for Countering High-Resolution Imaging Radars*”, Naval Postgraduate School Technical Report, NPS-EC-00-005, February 24, 2000.
3. *Tanner EDA Products*,
[<http://www.tanner.com/eda/products/tannertoolspro/default.htm>]
4. Finney, R. L., and Thomas, G. B., *Calculus*, 2nd Ed., Addison-Wesley Publishing Company, Inc. 1994.
5. Brown, J. W., and Churchill, R. V., *Complex Variables and Applications*, 6th Ed., McGraw-Hill, Inc. 1996.
6. Wakerly, F. John, *Digital Design: Principles and Practice*, 3rd Ed., updated, Prentice-Hall, Inc., 2001.
7. Mano, M. Morris, *Digital Design*, 2nd Ed., Prentice-Hall, Inc., 1991.
8. Griffin, Grant R., *CORDIC FAQ*,
[<http://www.dspguru.com/info/faqs/cordic2.htm>]
9. Lindlbauer, Norbert, *Implementation of Various CORDIC Architectures*, 2000-01-19, [<http://cnmat.cnmat.berkeley.edu/~norbert/cordic/node5.html>]
10. *MOSIS Process Information for TSMC*,
[<http://www.mosis.com/Technical/Processes/proc-tsmc-cmos018.html>]
11. Weste, Neil H. E., and Eshraghian, Kamran, *Principles of CMOS VLSI Design: A Systems Perspective*, 2nd Ed., AT&T, 1993.
12. Guillaume, C. H., *Circuit Design and Simulation for a Digital Image Synthesizer Range Bin Modulator*, Monterey, CA, Masters Thesis, Naval Postgraduate School, March 2002.
13. Amundson, C. A., *Design, Implementation, and Testing of a High Performance Summation Adder for Radar Image Synthesis*, Monterey, CA, Masters Thesis, Naval Postgraduate School, September 2001.
14. *MOSIS Parametric Test Results*,
[http://www.mosis.org/cgi-bin/cgiwrap/umosis/swp/params/tsmc-018/t15j_lo_epi-params.txt]

15. *MOSIS Scalable CMOS (SCMOS) Design Rules*, Revision 8.0, Updated March 13, 2002, [<http://www.mosis.org/Technical/Designrules/scmos/scmos-main.html>]

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Chairman
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California
4. Dr. Douglas Fouts, Code EC
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California
5. Dr. Phillip Pace, Code EC
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California
6. Dr. Herschel H. Loomis
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, California
7. Dr. David Laister
National Defence Head Quarters
Ottawa, Ontario
Canada
8. Mr. Ubee Fehr
National Defence Head Quarters
Ottawa, Ontario
Canada
9. Dr. John A. Montgomery
Naval Research Laboratory
Washington, D.C.
10. Mr. Alfred A. Di Mattesa
Naval Research Laboratory

Washington, D.C.

11. Mr. Gregory P. Hrin
Naval Research Laboratory
Washington, D.C.
12. Mr. Daniel W. Bay
Naval Research Laboratory
Washington, D.C.
13. Dr. Frank Klemm
Naval Research Laboratory
Washington, D.C.
14. Mr. Brian W. Edwards
Naval Research Laboratory
Washington, D.C.
15. Mr. George D. Farmer
Naval Research Laboratory
Washington, D.C.
16. Dr. Preston W. Grounds
Naval Research Laboratory
Washington, D.C.
17. Dr. Peter Craig
Office of Naval Research
Arlington, Virginia
18. Dr. Joseph Lawrence
Office of Naval Research
Arlington, Virginia
19. Mr. James Talley
Office of Naval Research
Arlington, Virginia