



**DEVELOPMENT OF A
SOFTWARE-DEFINED GNSS SIMULATION
PROTOTYPE FOR ADVANCED SIGNALS
RESEARCH**

THESIS

Michael J. Ledford, Second Lieutenant, USAF
AFIT-ENG-MS-17-M-043

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-17-M-043

**DEVELOPMENT OF A SOFTWARE-DEFINED GNSS SIMULATION
PROTOTYPE FOR ADVANCED SIGNALS RESEARCH**

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Michael J. Ledford, BSEE, BSCPE
Second Lieutenant, USAF

March 2017

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-17-M-043

DEVELOPMENT OF A SOFTWARE-DEFINED GNSS SIMULATION
PROTOTYPE FOR ADVANCED SIGNALS RESEARCH

THESIS

Michael J. Ledford, BSEE, BSCPE
Second Lieutenant, USAF

Committee Membership:

Dr. Richard K. Martin
Chairman

Dr. Sanjeev Gunawardena
Member

Mr. Pranav R. Patel
Member

Abstract

Global Positioning System (GPS) has become the de-facto standard for precision time transfer. By definition, all signal structure details of the open civilian GPS signals are known and published. This makes them vulnerable to a growing array of threats. One possible solution is to incorporate signal authentication features for future GPS. However, there lacks a hardware platform to verify these features. This thesis proposes a Field Programmable Gate Array (FPGA)-based GPS Waveform Prototyping Platform (GWPP) that features a software-defined all-digital signal synthesis architecture in conjunction with a high-speed Digital-to-Analog Converter as a solution.

GWPP is designed to synthesize GPS waveforms based on the emerging and maturing signal generation technology known as Direct Digital Synthesis (DDS). DDS adds dimensions not possible with analog designs in the sense that the fundamental nature of the waveform and its features are generated digitally; hence, providing precise control over the amplitude, frequency, phase, and modulation scheme of the output waveform.

The goal of this research is to design, implement, and integrate the GWPP architecture into the Virtex[®]-6 FPGA framework. The limited resources in the FPGA presents a trade-off between signal quality and FPGA resource utilization. This work not only explores the feasibility of the DDS-based architecture to synthesize GPS waveforms, namely the Link-1 Course Acquisition (L1 C/A) signal, but also assesses the spectral performance with consideration in FPGA resource utilization. Currently, the architecture achieves a spurious free dynamic range (SFDR) of -46 decibels relative to the carrier (dBc) consuming only 10% of FPGA logic slices.

*To my beautiful wife whose unconditional love and support kept me
motivated during this 18 month journey.*

Acknowledgements

First, I must thank God for giving me the wisdom and strength to complete this thesis. He has provided me with the strength to persevere through the challenging times and stay focused.

A special thanks to my wife for all of the motivation and support she showed me during this journey. Without her believing in me and motivating me through this project, this work would not have been completed. Also, for being strong during the times when I had to spend Friday, Saturday, and Sundays at the lab or the weeks leading up to my defense where I was super busy every single day. Thanks to my family and friends for believing in me.

A special thanks to Pranav Patel for his guidance and mentorship in digital design and FPGA development. Also, for the all of the times we spent talking about FPGA, VHDL, and design practices to some of the late nights spent helping me debug code, teaching me VHDL, or answering my emails regarding FPGAs and digital design.

A special thanks to Dr. Sanjeev Gunawardena for his support and guidance during the past 18 months. With his busy schedule, I appreciate the time he did spend teaching me GPS concepts as well as digital design practices. He allowed me to work on this project where I learned a lot about digital design and GPS.

A special thanks to my advisor Dr. Richard Martin for all of the support and patience he showed me during the past 18 months. Also, for the numerous, spontaneous discussions we had reviewing DSP concepts or teaching me new concepts.

Michael J. Ledford

Contents

	Page
Abstract	iv
Acknowledgements	vi
List of Figures	x
List of Tables	xiii
I. Introduction	1
1.1 Research Motivation and Background	1
1.1.1 Introduction to Global Navigation Satellite System	1
1.1.2 Introduction to Global Positioning System	2
1.1.3 Introduction to Advanced Signals Research	4
1.1.4 Introduction to GPS Waveform Prototyping Platform	9
1.2 Research Goal	10
1.3 Thesis Organization	10
II. Background	12
2.1 Overview of Global Positioning System	12
2.1.1 System Architecture	15
2.1.2 Signal Characteristics	20
2.2 Signal Generation Techniques	25
2.2.1 Signal Synthesis Specifications	26
2.2.2 Direct Analog Synthesizers	28
2.2.3 Indirect Analog/Digital Synthesizers	29
2.2.4 Direct Digital Synthesizers	30
2.2.5 Comparative Analysis	32
2.3 Direct Digital Synthesis	34
2.3.1 Direct Digital Synthesis Theory of Operation	35
2.3.2 Direct Digital Synthesis Architecture	35
2.4 Fundamentals of Field Programmable Gate Arrays	46
2.4.1 Introduction to Field Programmable Gate Arrays	47
2.4.2 Architecture of Xilinx Field Programmable Gate Arrays	49
2.4.3 Design Flow	53
2.4.4 Virtex-6 FPGA ML605 Development Board Features	56

	Page
III. Methodology	59
3.1 Design Overview	59
3.2 Hardware Implementation	64
3.2.1 High-Level Overview	64
3.2.2 GPS Waveform Prototyping Platform Clocking Scheme	67
3.2.3 GPS L1 Course Acquisition 1 Sample Generator	75
3.3 Experiment Setup	104
3.3.1 Experiment Setup Overview	104
3.3.2 Verification of the Register Transfer Level Models	104
3.3.3 Test Environment Setup	106
3.3.4 Final Experiment Setup	114
IV. Results and Verification	116
4.1 Test Environment Setup Results	116
4.1.1 Sampling Clock Selection Results	117
4.1.2 Reference Oscillator Selection Results	119
4.2 GPS Waveform Prototyping Platform Performance Metrics	122
4.3 Overview of Experiments	125
4.4 Spur Performance Due To Phase Truncation: 5-bit Amplitude Resolution	127
4.4.1 Wideband Spurious Performance Due To Phase Truncation: 5-bit Amplitude Resolution	129
4.4.2 Narrowband Spurious Performance Due To Phase Truncation: 5-bit Amplitude Resolution	131
4.5 Spur Performance Due To Phase Truncation: 6-bit Amplitude Resolution	132
4.5.1 Wideband Spurious Performance Due To Phase Truncation: 6-bit Amplitude Resolution	133
4.5.2 Narrowband Spurious Performance Due To Phase Truncation: 6-bit Amplitude Resolution	136
4.6 Spur Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution	136
4.6.1 Wideband Spurious Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution	137
4.6.2 Narrowband Spurious Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution	140
4.7 Spur Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution	140

	Page
4.7.1 Wideband Spurious Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution	141
4.7.2 Narrowband Spurious Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution	144
4.8 Spur Performance vs FPGA Resource Utilization	144
4.9 Experimental Spurious Performance vs Theoretical Spurious Performance	146
4.10 Summary of Results	148
V. Conclusion and Future Work	149
5.1 Summary	149
5.2 Recommendations for Future Work	150
Appendices	153
A. FMC230 Digital to Analog Converter Configuration Fundamentals	154
B. Distributed Clock Tree Configuration Fundamentals	159
C. Serializer/Deserializer Configuration Fundamentals	162
D. Universal Asynchronous Receiver Transmitter to L1 Course Acquisition 16 Sample Generator Interface	165
E. Universal Asynchronous Receive/Transmit Receiver	171
F. User Defined Blocks RTL Verification	178
6.1 Verification of Carrier NCO	178
6.2 Verification of Code NCO	178
6.3 Verification of C/A-Code Generator	179
6.4 Verification of Half-Wave Sine LUT	181
G. Narrowband Spurious Performance Results	183
7.1 Narrowband Spurious Performance Due To Finite Amplitude Resolution:5-bit Phase Resolution	183
7.2 Narrowband Spurious Performance Due To Finite Amplitude Resolution:6-bit Phase Resolution	184
Bibliography	186

List of Figures

Figure		Page
1.	GPS Waveform Prototyping Platform High Level Architecture	11
2.	Navigation Data Message Structure	23
3.	Binary Phase Shift Keying Modulation	25
4.	Direct Analog Synthesizer Architecture	29
5.	Phase Lock Loop Synthesizer Architecture	31
6.	Direct Digital Synthesis High Level Architecture	32
7.	Direct Digital Synthesis Theory of Operation	36
8.	Direct Digital Synthesis Phase Wheel	40
9.	Xilinx Configurable Logic Block Architecture	50
10.	Arrangement of Slices within the Configurable Logic Block	51
11.	Typical Field Programmable Gate Array Design Flow	55
12.	Virtex [®] -6 ML605 Board Features	58
13.	GPS Waveform Prototyping Platform Methodology Overview	60
14.	FMC230 Functional Block Diagram	61
15.	GWPP Architecture Configuration Cycle	63
16.	Hardware Implementation High Level Overview Functional Block Diagram	65
17.	GPS Waveform Prototyping Platform Clocking Scheme	68
18.	OSERDESE Bit Arrangement for a DAC Input Bus	74
19.	GPS L1 C/A 1 Sample Generator Block Functional Block Diagram	77
20.	C/A-Code Generator Functional Block Diagram	81

Figure		Page
21.	Code Generator Functional Block Diagram	83
22.	NCO Staircase Output	91
23.	Code Generator 16x Expansion	93
24.	Chip Detection Design and BPSK Offset Calculation	96
25.	BPSK Offset MUX Theory of Operation	97
26.	Carrier NCO Architecture and Signal Flow	99
27.	Carrier NCO 4x Expansion	101
28.	Phase-to-Amplitude Conversion High Level Overview and Signal Flow	102
29.	GPS L1 C/A 1 Sample Generator Register Transfer Level Verification	107
30.	Test Environment Setup Flow	108
31.	Error Sources in DDS Systems	123
32.	6-bit Amplitude Resolution Sine Wave LUT	128
33.	Wideband Spectrum Output Due To Phase Truncation: 5-bit Amplitude Resolution	130
34.	Narrowband Spectrum Output Due To Phase Truncation: 5-bit Amplitude Resolution	132
35.	6-bit Amplitude Resolution Sine Wave LUT	134
36.	Wideband Spectrum Output Due To Phase Truncation: 6-bit Amplitude Resolution	135
37.	Narrowband Spectrum Output Due To Phase Truncation: 6-bit Amplitude Resolution	137
38.	5-bit Phase Resolution Sine Wave LUT	138
39.	Wideband Spectrum Output Due To Finite Amplitude Resolution: 5-bit Phase Truncation	139
40.	5-bit Phase Resolution Sine Wave LUT	142

Figure	Page
41. Wideband Spectrum Output Due To Finite Amplitude Resolution: 6-bit Phase Resolution	143
42. Digital to Analog Converter Functional Block Diagram	155
43. 8:1 Data Stream Using DDR	156
44. Distributed Clock Tree Functional Block Diagram	160
45. SERDES Theory of Operation	163
46. OSERDES Block Diagram	164
47. UART to Sample Generator Interface Functional Block Diagram	167
48. Universal Asynchronous Receiver/Transmitter Interface Functional Block Diagram	172
49. UART 8N1 Frame	174
50. UART Algorithm Finite State Machine	177
51. Verification of Carrier NCO Block	179
52. Verification of Code NCO Block	180
53. Verification of C/A-Code Generator, Code Phase 0	181
54. Verification of C/A-Code Generator, Code Phase 1	181
55. Verification of Half-Wave Sine LUT	182
56. Narrowband Spectrum Output Due To Finite Amplitude Resolution: 5-bit Phase Resolution	184
57. Narrowband Spectrum Output Due To Finite Amplitude Resolution: 6-bit Phase Resolution	185

List of Tables

Table		Page
1.	Space Segment Satellite Development Starting with Block IIR	16
2.	Direct Digital Synthesis Number of Phase Points	39
3.	Code Phase Assignments and Code Sequences for C/A-Code	87
4.	Expanded Code Phase Assignments and Code Sequences for C/A-Code	88
5.	Example GPS C/A-Code Generator Initial States	90
6.	Spectrum Analyzer Calibration Settings	114
7.	Internal VCO Sampling Clock with 3 Different Reference Oscillators	118
8.	10 MHz TRIGR Oscillator vs 10 MHz SyncServer Oscillator	121
9.	Spur Performance Due To Phase Truncation: 5-bit Amplitude Resolution	127
10.	Wideband: 5-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions	129
11.	Wideband: 5-bit Amplitude Resolution: Spurious Free Dynamic Range	130
12.	Narrowband: 5-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions	131
13.	Narrowband: 5-bit Amplitude Resolution: Spurious Free Dynamic Range	131
14.	Spur Performance Due To Phase Truncation: 6-bit Amplitude Resolution	133
15.	Wideband: 6-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions	134

Table	Page
16. Wideband: 6-bit Amplitude Resolution: Spurious Free Dynamic Range	135
17. Narrowband: 6-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions	136
18. Narrowband: 6-bit Amplitude Resolution: Spurious Free Dynamic Range	136
19. Spur Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution	137
20. Wideband: 32 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions	139
21. Wideband: 5-bit Phase Resolution: Spurious Free Dynamic Range	140
22. Spur Performance Due To Finite Amplitude Resolution: 6-bit Phase Truncation	141
23. Wideband: 64 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions	141
24. Wideband: 6-bit Phase Resolution: Spurious Free Dynamic Range	142
25. Power _{CSE} vs FPGA Resource Utilization	145
26. SFDR 100 MHz Span vs FPGA Resource Utilization	146
27. SFDR 20 MHz Span vs FPGA Resource Utilization	146
28. SFDR 100 MHz Span vs Theoretical	147
29. Summary of Results.....	148
30. Narrowband: 32 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions	183
31. Narrowband: 5-bit Phase Resolution: Spurious Free Dynamic Range	183
32. Narrowband: 6-bit Phase Resolution, 64 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions.....	184

Table		Page
33.	Narrowband: 6-bit Phase Resolution: Spurious Free Dynamic Range	185

List of Acronyms

3-D	Three-Dimensional
AC	Alternating Current
AFIT	Air Force Institute of Technology
AR	Amplitude Resolution
ASICs	Application Specific Integrated Circuits
BNTS	BeiDou Navigation Test System
BPSK	Binary Phase-Shift Keying
bps	Bits Per Second
BRAM	Block RAM
C/A	Course Acquisition
CDC	Clock Domain Crossing
CDMA	Code Division Multiple Access
CE	Clock Enable
CLBs	Configurable Logic Blocks
CORDIC	COordinate Rotation DIgital Computer
COTS	Commercial Off-The-Shelf
CSE	Carrier to Spurious Emission
CW	Continuous Wave
dB	Decibels
dBc	Decibels Relative to Carrier
dBm	Decibel Milliwatt
DC	Direct Current
DCI	Data Clock Input
DCT	Distributed Clock Tree

DCO	Data Clock Output
DCR	Duty Cycle Restorer
DAC	Digital-to-Analog Converter
DDR	Double Data Rate
DDS	Direct Digital Synthesis
DoD	Department of Defense
DSP	Digital Signal Processing
DSSS	Direct Sequence Spread Spectrum
DUT	Design Under Test
FFs	Flip Flops
FFT	Fast Fourier Transform
FIFO	First In First Out
FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FTW	Frequency Tuning Word
GEO	Geosynchronous Earth Orbit
GHz	Gigahertz
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input / Output
GPS	Global Positioning System
Gsps	Gigasamples Per Second
GWPP	GPS Waveform Prototyping Platform
HDL	Hardware Description Language
Hz	Hertz

HOW	Handover Word
HPC	High Pin Count
ICs	Integrated Circuits
IF	Intermediate Frequency
IF	Interface
IO	Input/Output
IOBs	Input/Output Blocks
IP	Intellectual Property
kHz	Kilohertz
km	Kilometer
L1	Link 1
L1 C/A	Link 1 Course Acquisition
L2	Link 2
L2C	L2-Civil
L5	Link 5
LEO	Low Earth Orbit
LFSRs	Linear Feedback Shift Registers
LLC	Limited Liability Company
LNAV	Lateral Navigation
LSBs	Least Significant Bits
LUT	Lookup Table
LVDS	Low-Voltage Differential Signaling
LVPECL	Low-Voltage Positive-Emitter Coupled Logic
MEO	Medium Earth Orbit
MHz	Megahertz
Mcps	Mega Chips Per Second

MMCM	Mixed-Mode Clock Manager
ms	Millisecond
MSB	Most Significant Bit
MTBF	Mean Time Before Failure
MUX	Multiplexer
NCO	Numerically Controlled Oscillator
NRE	Non-Recurring Engineering
ns	Nanoseconds
OCXO	Oven-Controlled Crystal Oscillator
PC	Personal Computer
PCB	Printed Circuit Board
PLDs	Programmable Logic Devices
PLLs	Phase Lock Loops
PLO	Phase-Locked Oscillator
PPS	Precise Positioning Service
PR	Phase Resolution
PRN	Pseudo Random Noise
PVT	Position, Velocity, Timing
RAIM	Receiver Autonomous Integrity Monitoring
RAM	Random Access Memory
RF	Radio Frequency
RHCP	Right-Hand Circularly Polarized
RNSS	Regional Navigation Satellite System
ROM	Read Only Memory
RTL	Register-Transfer Level
SERDES	Serializer/Deserializer

SFDR	Spurious Free Dynamic Range
SDR	Software-Defined Receiver
SDR	Single Data Rate
SPS	Standard Positioning Service
SVs	Satellite Vehicles
TCXO	Temperature-Compensated Crystal Oscillator
TLM	Telemetry
TOA	Time of Arrival
TOW	Time-of-Week
UART	Universal Asynchronous Receiver Transmitter
UCF	User Constraint File
UDBs	User-Defined Blocks
USB	Universal Serial Bus
VCO	Voltage Controlled Oscillator
VHDL	VHSIC Hardware Description Language
XOR	Exclusive-Or
XST	Xilinx Synthesis Technology

DEVELOPMENT OF A SOFTWARE-DEFINED GNSS SIMULATION
PROTOTYPE FOR ADVANCED SIGNALS RESEARCH

I. Introduction

THIS chapter presents a brief introduction to Global Navigation Satellite System (GNSS), Global Positioning System (GPS), advanced signals research, and GPS Waveform Prototyping Platform (GWPP). It discusses the potential applications of GWPP, as well as limitations in advanced signals research GWPP attempts to solve. This chapter also defines the research goal and outlines the organization of this document.

1.1 Research Motivation and Background

1.1.1 Introduction to Global Navigation Satellite System.

Navigation satellite systems provide the Position, Velocity, Timing (PVT) information that enables many applications we use in our daily lives. The PVT information is provided by the satellite signals transmitted from Satellite Vehicles (SVs). Once SVs are launched into space, the signals are unchangeable. This applies to all satellite-based navigation systems. There are several satellite-based navigation systems being utilized throughout the world—each providing global or regional coverage. The term GNSS refers to a collection of constellations that provide global coverage, whereas the term Regional Navigation Satellite System (RNSS) refers to the constellations that provide regional coverage. There are two GNSS platforms currently in use and two others in development. The platforms currently in use include Global

Navigation Satellite System (GLONASS), developed by Russia, and GPS, developed by the United States. While, the two platforms in development include GALILEO, created by the European Union, and BeiDou Navigation Test System (BNTS), created by China[1]. The proliferation and evolution of these platforms into enabling technologies has increased research efforts in the field of GNSS. However, in this research effort, we primarily focus on the GNSS platform native to the United States: GPS.

1.1.2 Introduction to Global Positioning System.

GPS is an essential element in our information infrastructure. The free, accessible, and reliable nature of GPS has attributed to its increased integration into many civilian applications and technologies. Transportation, recreation, space, emergency response services, surveying and mapping are just a few of the applications that rely on GPS positioning [1]. GPS has also become the de-facto standard for applications that rely on precision time transfer. The national power grid, cellular phone networks, and global financial systems depend on precise timing for synchronization and operational efficiency. Precision timing is a direct benefit from the architecture of GPS satellites. Each GPS satellite contains several atomic clocks regulated by the vibrations of atomic or molecular systems [1]. The principle of operation of an atomic clock allows GPS satellites to contribute precise time data to GPS signals. GPS receivers can then decode the received signal and effectively synchronize their clocks to atomic clocks. Consequently, users can accurately determine time, up to 100 billionths of a second, without needing to purchase an atomic clock [1]. The ability to approximate time this accurately reinforces why communication systems, electrical power grids, and financial networks all rely on GPS for precise timing. The critical role these components play in affecting every aspect of modern life emphasizes the

importance of GPS technology in civilian applications.

Although there are numerous benefits associated with GPS and its applications, there are also several challenges. For example, at its most basic form, GPS signals are Radio Frequency (RF) signals transmitted from space. After leaving the satellite antenna, the RF signals suffer from free space propagation loss and additional atmospheric losses. By the time it reaches the receiver, the signal's power with respect to one watt is -129 Decibel Milliwatt (dBm), which is approximately 30 Decibels (dB) below the thermal noise floor, -100 dBm. Another challenge of GPS is that, by definition, all signal structure details of the open civilian GPS signals are known and published. Therefore, the published signal structure details as well as the precariously weak nature of GPS signals make civilian GPS signals easy to block and disrupt. As a result, GPS signals are highly vulnerable to a growing array of threats.

Threat sources can be broken down into two types: natural and man-made. Multipath, scintillation, and obscuration are just a few threat sources created naturally. Multipath is caused by signals from one or more satellites reflecting off nearby structures resulting in a disintegrated path to the receiver; thus, generating ranging errors in the receiver processing. Scintillation is caused by the refraction or diffraction of the satellite signal in space due to irregular ionospheric activities [2]. Often, this threat type results in inaccurate timing data. Obscuration occurs when there are not enough satellites visible to provide an accurate position measurement negatively causing the user to experience signal loss. On the other hand, hacking, spoofing, jamming, and RF interference are threat sources that are man-made [2]. Hence, these are the sources that feasibly pose a threat due to the open publications of GPS civilian signal structures. Hacking refers to the manipulation of the software layer in receivers to change its interpretation of satellite signal data whereas, spoofing is the maliciously faking of GPS signals to deceive devices into believing false location and timing information

[2]. Hacking and spoofing impact the user in similar ways—both result in false time and location readings, which severely impacts autonomous devices that rely on GPS for precise timing [2]. Jamming is the process where locally-generated RF signals are used to interfere and drown out satellite signals whereas, RF interference is the process where noise from nearby RF transmitters—inside or outside device—disrupts the satellite signals. Likewise, jamming and RF interference impact the user in similar ways—both resulting in loss of signal or inaccuracies affecting location and timing information.

As GPS signals continue to enormously influence the way we live, mitigating threats affecting GPS-reliant systems become just as important as understanding them. For the people who use these systems on a daily basis, GNSS is simply available; we trust GPS will provide accurate data at the right place at the right time. But for the people whose responsibility lies in ensuring these systems work correctly, this process is not as straightforward. With different threats affecting GPS-reliant systems in various ways, this process now turns into one of critical risk management.

1.1.3 Introduction to Advanced Signals Research.

Advanced signals research focuses its efforts on applying critical risk management to not only resolve current challenges in GPS, but mitigate future ones; however, solutions resolving challenges extend beyond mitigating vulnerabilities: they must be cost-efficient as well. Advanced signals research evaluates both the development of solutions and implementation process.

There are many mitigation solutions currently under research and development. These include, but are not limited to: improved antennas, multi-constellation GNSS receivers, multi-frequency receivers, innovative Digital Signal Processing (DSP) technology in the receiver, and using Receiver Autonomous Integrity Monitoring (RAIM)

techniques in the receiver [2]. Although efficient at countering some of the threats, several require redesigning hardware, in this case, receivers or antennas. While feasible, these techniques are not always viable on a large scale; consider the following: as technology rapidly increases, vulnerabilities increase and adversaries have new ways of attacking GPS-reliant systems. To adapt, manufacturers need to constantly develop new hardware while users purchase it. Software-defined receivers alleviate this issue by allowing the hardware inside receivers to be reconfigured through software. This enables full control in reconfiguring the receiver to adapt to the threat without obsolescence. Although, this implementation only works if the user has adequate processing power in their devices. In the case where users do not, additional costs are incurred to upgrade or purchase new hardware. The costs involved for both users and manufacturers demonstrate why this mitigation technique is feasible, but not viable on a large-scale. Therefore, a robust solution would be one that is resilient, viable and prevents user equipment from rapid obsolescence.

One possible solution is to alter GPS signal characteristics. The benefit of such a solution can be generalized by analyzing the underlying core of all man-made threat sources. At the core of each threat is the exploitation of the architecture of GPS, to include both the signals and SVs. The exploitation of GPS signals is a direct result of the publicly known civilian signal structure details, which are at the disposal of adversaries for misuse. Consequently, adversaries have the recipe to directly spoof, jam, and hack civilian signals to maliciously attack GPS-reliant infrastructures. Alternatively, the exploitation of GPS satellites is a direct result of the SV architecture. GPS SVs are designed to transmit both civilian and military signals on various frequency bands. Traditionally, there were only two frequency bands used by GPS, Link 1 (L1) (1575.42 Megahertz (MHz)) and Link 2 (L2) (1227.60 MHz), respectively; however, the modernization of GPS introduced a third band, Link 5 (L5) (1176.45 MHz). Fur-

thermore, since GPS was first conceived for military use, only a subset of these bands are available for civilian users. Once the SVs are launched into space, the signals are essentially unchangeable. With this in mind, if a specific civilian signal is compromised, GPS for civilian use is essentially futile. As shown, the exploitation of GPS infrastructure, to include both signals and SVs, forces GPS-reliant infrastructures to remain critically vulnerable to man-made threats. Although, if truly analyzing the GPS component that fuels the vulnerabilities, it can be seen that this component is effectively the publicly known details of the GPS civilian signal structure. Hence, this case reinforces the need to alter GPS signal characteristics. Instead, if the GPS civilian signal structure were altered to include carefully designed authentication capabilities then this decreases the capability for adversaries to decrypt GPS signals; thus, mitigating several man-made threats. To make this solution robust and viable, two important factors must remain in consideration. First, the signal authentication technique must be theoretically and operationally verified in a hardware platform to ensure correctness in an environment similar to that of a SV. Second, the technique must ensure backwards compatibility with traditional receivers on the market, also referred to as legacy receivers. With verification in both of these factors, this solution is both viable and robust; viable in the sense that backwards compatibility prevents GPS receiver obsolescence and robust in the sense that signal authentication adds a layer of protection to civilian GPS signals. Although the technique to alter signal characteristics seems straightforward, there are current limitations in advanced signals research which introduce challenges in implementation. The challenges exist in both the design cycle as well as the verification of the signal authentication technique.

First, in the design cycle, the entire design process from conception to implementation is strictly sequential. In other words, it follows a series of tasks from start to finish. Although the design process is eventually completed, if the entire process

is sequential, then the rate of completion is ultimately limited by the slowest task in this process. In addition, for GPS signal design, it is severely crucial that the signal authentications are thoroughly verified. In a sequential process, the signals are initially verified through theoretical and mathematical analysis as well as through software simulations. The point at which the signals are verified for correctness in hardware occurs when the SV is launched into space. However, it is important to note that correct theory of operation in software does not guarantee correct operation in hardware. With this in mind as well as the unchangeable architecture of GPS signals essentially introduces a degree of operational risk. For example, the signal is incompatible with user equipment or the authentication technique is not applied correctly in hardware. At this time, countless resources spent during the design process are wasted. However, parallelizing the design process in the sense that new signal characteristics are verified in hardware as the characteristics are being developed effectively reduces operational risk and provides two benefits. The first benefit is early on testing in hardware, which allows designers to catch any errors in the design before moving further in the design process. The second benefit is that the designer can observe how the new signal characteristics implemented in hardware behave with current user equipment; hence, providing the designer with the opportunity to verify compatibility.

In order to parallelize the design process, a hardware testing platform is required. Typically, when signal designs in hardware are being tested, test equipment such as signal generators and functional generators come to mind. Although they can generate a wide range of frequencies, the circuit that powers these devices are designed for a limited set of functions. Therefore, there lacks a wide range of flexibility. Also, the software installed on these devices only provide the user with a finite set of capabilities. Usually to acquire more capabilities requires the purchase of additional

licenses or hardware; thus, the cost of verification can quickly escalate while providing limited flexibility. Ideally, the best hardware platform for verification is one that is reconfigurable to test different signal designs; cost-efficient to heavily used for research and development; generates RF waveforms since the application is for GPS; and is highly flexible to synthesize waveforms of arbitrary characteristics.

In summary, GPS is a vital component in the civilian infrastructure. The national power grid, cellular phone networks, and global financial systems all rely on GPS for precise timing. By definition, all signal structure details of the open civilian GPS signals are known and published. This makes them vulnerable to a growing array of threats such as hacking, spoofing, and jamming. Advanced signals research focuses its efforts on applying critical risk management to resolve current challenges in GPS. As previously discussed, some solutions are feasible, but not viable on a large-scale. These solutions typically include the need for manufacturers to design new test equipment and users to purchase them. Another possible idea, which if designed with consideration in backwards compatibility, provides both a feasible and viable solution to mitigate man-made threats. This idea is ultimately the alteration of GPS signal characteristics. One possible extension to this idea is to include signal authentication capabilities into future GPS. However, as previously mentioned, there lacks an ideal hardware platform to verify these features. Recall that in this case, ideal refers to a platform that is reconfigurable, flexible, cost-efficient, and can generate RF waveforms.

This research effort proposes a Field Programmable Gate Array (FPGA)-based GPS waveform prototyping platform that features a software-defined all-digital signal synthesis architecture in conjunction with a high-speed Digital-to-Analog Converter (DAC) as a solution. The architecture of this platform not only provides designers with increased flexibility in signal generation compared to signal generators

and function generators, but also serves as the desired hardware platform to parallelize the signal design process. The flexibility enables the ability to synthesize arbitrary waveforms not only limited to GPS. In addition, the architecture of this platform can potentially serve as a testing platform for new signal designs. Thus, allowing the designer to evaluate signal performance in a simulation environment similar to that of a satellite vehicle in the early stages of the signal design cycle. For the remainder of this document, this platform is referred to as GWPP.

1.1.4 Introduction to GPS Waveform Prototyping Platform.

GWPP features a software-defined all-digital signal synthesis architecture in conjunction with a high-speed DAC. It is designed to synthesize GPS waveforms based on the emerging and maturing technology known as Direct Digital Synthesis (DDS). DDS adds dimensions not possible with analog designs in the sense that the fundamental nature of the waveform and its features are generated digitally; hence, providing precise control over the amplitude, frequency, phase, and modulation scheme of the output waveform. Consequently, this architecture is capable of producing RF sinusoidal waveforms with very fine frequency resolution (sub-nanohertz) and fast switching speed without the need for additional circuitry such as Phase Lock Loops (PLLs) and analog mixers. The software-defined component of this architecture allows the tedious and time-consuming digital hardware blocks to be designed once and the focus of all subsequent development and capability enhancement to be predominantly software based. This results in the high-rate computations such as DDS to be performed in the FPGA and the configuration of the waveform characteristics to be performed within a workstation-class computer.

1.2 Research Goal

The goal of this research effort is to design, implement, and integrate the software-defined DDS architecture of GWPP into the Virtex[®]-6 FPGA framework. The limited resources in the FPGA presents a trade-off between signal quality and FPGA resource utilization. The high-level architecture of GWPP is shown in Figure 1.

This research effort focuses primarily on a subcomponent of this architecture; namely, the GPS Link 1 Course Acquisition (L1 C/A) signal synthesis shown in Figure 1. However, for the remainder of this thesis, the term GWPP will refer to the architecture that solely generates the GPS L1 C/A signal. This work not only explored the feasibility of the DDS-based architecture to synthesize GPS waveforms, namely the L1 C/A signal, but also assesses the spectral performance of the platform with consideration in FPGA resource utilization. It is important to note that in this research effort, the Lateral Navigation (LNAV) message is not incorporated into the synthesis of the GPS L1 C/A signal. In addition, the analysis is performed with respect to the generation of a Continuous Wave (CW), namely the GPS L1 carrier since the quality of the carrier ultimately affects the quality of the GPS L1 C/A composite waveform.

1.3 Thesis Organization

The remainder of this thesis is organized as follows. Chapter II presents background information related to the field of GPS, signal generation techniques, and FPGAs. The general concept of GPS such as system architecture and signal characteristics are discussed, along with a brief introduction to GPS receiver architecture. Additionally, a comparative analysis is presented on signal generation techniques followed by the signal generation technique used in this research effort. Finally, an introduction to FPGAs is presented.

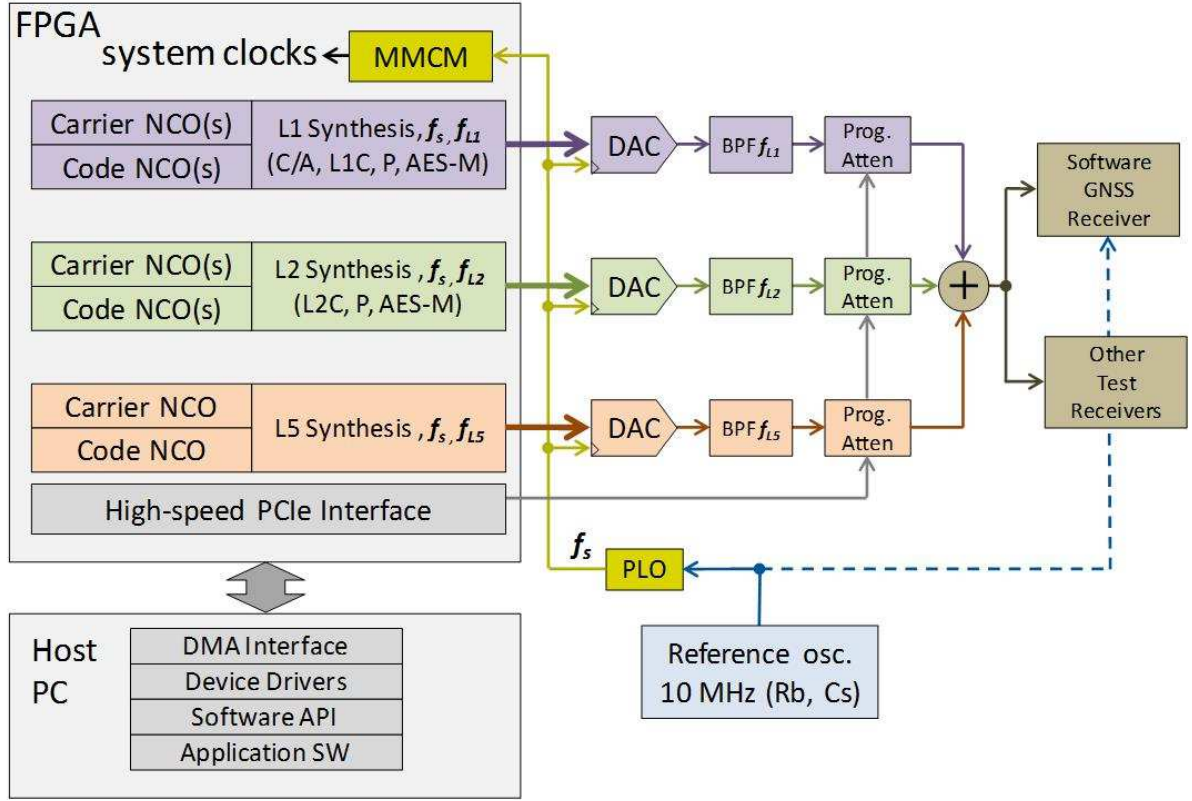


Figure 1. GPS Waveform Prototyping Platform High Level Architecture. This thesis focuses on the implementation of the lavender design blocks; namely, GPS L1 C/A

Chapter III discusses the methodology and tools used to design, build and test the GWPP architecture. First, a high level overview of this design is introduced, which describes the main building blocks of the design implemented in hardware. Afterwards, verification of the hardware design blocks are presented followed by the setting up the experiment to evaluate the platform.

Chapter IV presents the analysis of the platform using the experiment setup discussed in Chapter III. In this chapter, the performance metrics used to analyze the platform are introduced. Afterwards, the results with respect to the performance metrics are presented.

Finally, Chapter V summarizes the design and results. As a final point, potential areas for future research are also suggested.

II. Background

THIS chapter provides a theoretical framework for concepts that lie at the foundation of the proposed research architecture, which comprises GPS, signal generation techniques, and FPGAs. First, Section 2.1 covers an overview of GPS. Specifically, GPS architecture and signal characteristics. Next, signal generation techniques are discussed in Section 2.2. In this section, a comparative analysis is presented with respect to three conventional techniques currently being used for signal synthesis. Afterwards, Section 2.3 presents the theory of operation and the architecture of DDS based systems. Finally, this chapter concludes with an introduction to FPGAs.

2.1 Overview of Global Positioning System

GPS is a constellation of satellites that provide precise, continuous, global, Three-Dimensional (3-D) position and velocity information to users with the appropriate receiver technology. The satellite constellation nominally consists of 24 satellites, however, currently there are 32; each one fully operational. The satellites broadcast navigation data and ranging codes on two different frequencies, in the L-band, using a technique called Code Division Multiple Access (CDMA). CDMA provides efficient use of spectrum and enables full bandwidth for all users all the time. The two frequencies currently in use by the system are L1 (1575.42 MHz) and L2 (1227.60 MHz). With the expansion of GPS under the GPS modernization program, the system will broadcast on an additional frequency, L5 (1176.45 MHz) [1]. The purpose for additional signals is to use with differential processing techniques to rapidly provide users with higher accuracy. In addition, it increases robustness to interference in user equipment. In order to exploit the benefits of CDMA, each satellite must transmit unique ranging codes. Each satellite generates a short code referred to as the Course

Acquisition (C/A) code and a long code referred to as the precision, or P(Y), code. The ranging codes are also known as Pseudo Random Noise (PRN) codes. Typically there is confusion between the SV number and the PRN number. SV number refers to the space vehicle number, whereas the PRN number refers to its unique pseudorandom noise code. Often times, these numbers are equivalent; however, this is not always the case.

The L1 and L2 carriers are modulated by one or more bit trains, each of which is a composite constructed by the modulo-2 addition of a PRN ranging code and the navigation data message. The navigation data message, uploaded daily from a world-wide ground control/monitoring network, provides the means for the user equipment to determine the location of the satellite at time of transmission, whereas the ranging code enables the user equipment to determine the propagation time of the signal. The propagation time is used in a concept called one-way Time of Arrival (TOA) ranging, which is employed by the user equipment to determine the satellite-to-user range. In order to use TOA ranging to calculate 3-D user position, two requirements must be fulfilled. First, the user receiver must be equipped with a clock. Recall, that GPS satellites have on-board atomic clocks with high accurate atomic frequency standards in sync with a GPS time base. To extend this clocking architecture on all receivers would be highly expensive. Instead, a crystal clock is typically placed in receiver architectures to minimize the cost, complexity, and size of the receiver. Second, 3-D positioning requires TOA ranging measurements to be made on at least four satellites to solve for the four unknowns: latitude, longitude, height, and receiver clock offset from GPS time. The receiver clock offset is an unknown due to the clocks on both the receivers and satellites operating independently, which introduces synchronization biases. The bias in the receiver clock at the instant of the measurements affects the observed transit times for all satellites equally [3]. The corresponding measurements

are off by a common amount. For this reason, these measurements are referred to as *pseudoranges*. The method of using several TOA ranging measurements to solve for these unknowns is known as trilateration. Furthermore, if any of these offsets are accurately known, less than four satellites are required. Therefore, GPS is essentially a passive ranging system. That is, satellites broadcast signals in the L-band that are composed of unique ranging codes and a navigation message. Users constantly “listen” to them in order to determine their position. Both ranging and positioning are based on two concepts: TOA and trilateration. As a passive system, GPS can service an unlimited number of users, whereas an active system requires the interaction with each user. The disadvantage of an active system is the limitation in the number of concurrent users. Therefore, to provide service to an unlimited amount of users, GPS is designed as a passive system. To exploit this advantage, user receivers must also operate passively, which means the receivers are only designed to receive.

In addition to being a passive system, GPS is also a dual-use system. This simply means that it provides separate services for military and civil users. These are referred to as Precise Positioning Service (PPS) and Standard Positioning Service (SPS). PPS is intended for United States authorized military, whereas SPS is designated for the civil community [1]. Both PPS and SPS are made possible due to the GPS signal structure. The responsibilities involved in supporting GPS and its services are compartmentalized into three segments: Space, Control, and User. Each of these support GPS in unique ways and are vital to the health and status of all satellites.

This section is structured to first familiarize the reader with the fundamentals of the GPS system architecture (Section 2.1.1). That is, the role each segment (Space, Control and User) plays to preserve the health and status of the GPS constellation. Once this groundwork has been established, a description of the GPS signal characteristics is presented in Section 2.1.2. That is, the signal frequencies and modulation

techniques involved in creating the GPS signal. However, it is important to note that this discussion solely focuses on GPS L1 C/A signal since its synthesis using this platform is a goal of this research effort.

2.1.1 System Architecture.

GPS is comprised of three unique segments: Space, Control, and User. This section covers the fundamentals of each segment beginning with the Space Segment. Next, the Control Segment will be presented followed by the User Segment. This section develops the principles for understanding the GPS system architecture.

The Space Segment comprises the satellites in the GPS constellation. As previously mentioned, the constellation nominally consists of 24 active satellites—21 operational and 3 active spares. However, recently, the constellation has expanded to 32 fully operational satellites. The constellation is made up of six orbital planes with 55° inclination relative to the equatorial plane. There are four or more satellites per plane, distributed unevenly, with each having a period of approximately 11 hours and 58 minutes [3]. The rationale behind the uneven distribution is to reduce the effect of single satellite failures. Since the spin period of Earth in space is about 23 hours and 56 minutes, each satellite repeats the same ground track after two revolutions [3].

GPS satellites orbit the Earth in a Medium Earth Orbit (MEO). MEO is a class of orbits with altitudes above Low Earth Orbit (LEO) (2000 Kilometer (km)) and below Geosynchronous Earth Orbit (GEO) (36000 km) [3]. MEO is chosen for GPS due to the availability and coverage from satellites in this altitude. Typically, satellites in this altitude range are visible for several hours in each pass. Consequently, 24-36 satellites are needed to provide global coverage [3]. However, there is one disadvantage of MEO. Due to the altitude range, the costs of launch and maintenance increase. Other orbit altitudes such as LEO and GEO are available, but considerations in

costs and geometric properties compared to MEO make these altitudes undesirable and MEO preferable. This concept is generalized in the following discussion. First, consider the choice to launch satellites in LEO. The launch costs are less expensive due to low altitude, however, the disadvantage is that the satellites are only visible for 10-20 minutes at a time. Hence, GPS receivers are constantly reacquiring satellites which results in the need for 100-200 satellites to provide global coverage [3]. Next, consider the case for GEO. A smaller satellite constellation is required to provide global coverage; however, GEO offers poor coverage at higher latitudes; additionally, resulting in much higher launch and maintenance costs [3]. With consideration in both cases, it is evident to why MEO is preferred for GPS.

The Space Segment also entails the history and future of constellation expansion, or satellite block development. Table 1 provides an overview of the block development from Block II (Initial Production Satellites) through the current development of Block III (Next Generation Satellites). In between these blocks are Block IIA (Upgraded Production Satellites), Block IIR (Replenishment Satellites), Block IIR-M (Modernized Replenishment Satellites), and Block IIF (Follow-On Sustainment Satellites).

Table 1. Space Segment Satellite Development Starting with Block IIR

	Block IIR	Block IIR-M	Block IIF	Block III(A,B,C)
Number SVs	13	8	12	32
First Launch	1997	2005	2010	2016+
Design Life (Yrs)	7.5	7.5	12	15
In Use (Jan. 2015)	12	7	11	0
L1 Signals	C/A,P(Y)	C/A,P(Y),M	C/A,P(Y),M	C/A,P(Y),M,L1C
L2 Signals	P(Y)	P(Y),L2C,M	P(Y),L2C,M	P(Y),L2C,M
L5 Signals	-	-	L5	L5

It is important to note that the satellites in Blocks (II, IIA, and IIR) are referred to as legacy GPS satellites and the signals L1 C/A, L1 P(Y), and L2 P(Y) are legacy signals. The salient features of each block in Table 1 are summarized in [3] and [1].

However, for the sake of this discussion, the developments made in Block IIR-M and Block III are presented

- *Block IIR-M- Modernized Replenishment Satellites:* Addition of three new signals: two new military codes on L1 and L2, M-code, and a new civilian code on L2, L2-Civil (L2C). L2C is an improved signal sequence over L1 C/A, allowing for both ionospheric corrections to be done by civilian users and backward-compatibility with existing signal structures (L1 C/A and P(Y), and L2 P(Y)). M-code on L1 and L2 provides authorized users with additional signal security.
- *Block III- Next Generation Satellites:* Conceived in 2000 to reassess the entire GPS architecture to provide sufficient architectural flexibility to satisfy evolving requirements through 2030. It is expected to provide higher signal power to meet military anti-jam requirements as well as both greater timing accuracy and sub-meter position accuracy.

The basic idea in illustrating the subsequent block developments is to demonstrate the constant motivation to expand GPS. Specifically, exploiting modernized technology to reshape the space and control segments to become more architecturally flexible. With modernized hardware, it is becoming more achievable to enhance GPS accuracy, integrity, availability and continuity all to support military missions and civilian applications. As the block developments enhanced each iteration, the number of satellites launched and their respective design life increased. With this in mind, it is evident how important design considerations and verification are to the success of a satellite block development. Also, it provides additional insight to the fact that the user equipment market is primarily dominated by legacy receivers. That is, receivers that acquire and track L1 C/A, L1 P(Y), and L2 P(Y) signals. Therefore, in order to prevent the massive number of receivers from being obsolete, new signals with signal authentication capabilities must be backwards compatible with the legacy receivers

on the market. Hence, the application of this platform enables the ability to test backwards compatibility.

The Control Segment essentially manages satellite operations. It ensures that satellites maintain their proper orbital positions and are healthy. The Control Segment consists of a global network of ground stations that track GPS satellites. The Master Control Station, located at Schriever Air Force Base, is at the heart of the Control Segment since it operates the system and provides command and control functions. The Control Segment also consists of additional elements such as monitor stations and ground antennas, which are unmanned and controlled remotely from the Master Control Station [4]. Some functions of the Master Control Segment include, but are not limited to: monitoring satellite orbits, monitoring and maintaining satellite health, maintaining GPS time, updating satellite navigation messages, performing analysis on satellite transmission, and sending commands and data to the constellation [3],[4]. However, its salient task is to generate and distribute the navigation data message. The navigation message is based on data collected and processed from the monitor stations. This data aids in the estimation and prediction of satellite orbits and clock biases. The navigation message is important to the receiver since it comprises data on the satellite health status, ephemeris, and clock bias parameters. In addition, it provides an almanac specifying reduced-precision ephemeris data on each satellite in the constellation. It effectively provides clock corrections and precise orbital measurements that are decoded by the receiver to determine the propagation time of the signal. This is further discussed in Section 2.1.2. An excellent overview of the entire process regarding the generation and distribution of the navigation message is provided in [1].

The User Segment comprises the development and application of civil and military user equipment. The user equipment is designed to process the the L-band

signals transmitted from the satellites to calculate PVT. This equipment is generally found in the marine, land, sea, and space arenas of operation. Technology trends in process technology and large-scale manufacturing have led to a proliferation of both inexpensive receiver equipment to include both physical components (stand-alone or embedded) and software-defined receivers. Typical receivers include components such as an antenna, pre-filter/pre-amplifier, receiver, processor, input/output device. The antenna, which is Right-Hand Circularly Polarized (RHCP), is the component that receives the satellite signals. Design factors such as transfer response, polarization, and coverage must be considered to effectively receive the signals at L1 and L2. A low noise amplifier is included to maintain a low-noise figure within the receiver; the combination of an antenna and low-noise amplifier is referred to as an active antenna [1]. The antenna and receiver front end must have sufficient bandwidth in order to pass the signals of interest which are L1, L2 and L5. To receive the full P(Y) code, a bandwidth of 20.46 MHz is required, whereas to track the L1 C/A signal, a bandwidth of 2.046 MHz is required [3]. It should also be noted that this bandwidth is directly proportional to the accuracy required for a particular receiver. That is, a receiver that is able to process a larger bandwidth of the receiver satellite signal will have better accuracy in terms of performance [1]. Recall that GPS is a dual-use system in that it provides separate services for military and civil users, referred to as PPS and SPS, respectively. Correspondingly, SPS can potentially have different receivers than PPS users to satisfy their operations at a lower costs. Hence, this introduces one important principle of receiver design. The ability to track multiple signals from different GNSS platforms as well as multiple satellites increases the cost of the receiver. For that reason, GPS receiver selection is dependent upon intended application.

2.1.2 Signal Characteristics.

In this section, the properties of the legacy GPS satellite signals to include frequency assignment, modulation schemes, navigation data message, and the generation of PRN codes are discussed. The scope of this thesis is limited to the GPS-SPS broadcast. In building GWPP, it is vital to understand the signal characteristics on both the transmitting and receiving side of the system in order to build an accurate prototype. In addition, providing this background of signal characteristics facilitates understanding the design of this platform in hardware. This section is organized as follows. First, the signal structure for GPS-SPS is discussed. Following this discussion is a breakdown of its respective components: frequency assignment, modulation format, navigation data message, and PRN code. Afterwards, this section concludes with a summary of the GPS L1 C/A signal structure.

The structure of the legacy GPS satellite signals is described exhaustively in Interface Specifications. The Interface Specification for GPS-SPS signals is made public [IS-GPS-200H] [5]. Each GPS signal consists of three components: RF carrier, Ranging Code, and Navigation Data. Since this thesis is limited to the GPS-SPS broadcast, this signal structure is only considered. The signal structure for the GPS-SPS transmitted signal, $S(t)$, can be modeled as:

$$S(t) = A(t) G(t) D(t) \sin(2\pi(f_0 + \Delta f_{sv}) + \phi_0) \quad (2.1.1)$$

Where:

$A(t)$: time dependent amplitude of transmitted signal (-130 dBm for C/A)

$G(t)$: C/A-code for the satellite (1.023 Mega Chips Per Second (Mcps))

$D(t)$: navigation data message (50 Bits Per Second (bps))

f_0 : L1 carrier frequency (154*10.23MHz), or 1575.42 MHz

Δf_{sv} : satellite frequency offset

ϕ_0 : carrier phase offset

Normally, a legacy satellite will broadcast on two frequencies, $L1$ and $L2$, respectively. Two signals are transmitted on $L1$, one for civil users (C/A), and the other for Department of Defense (DoD)-authorized users (Precision Encrypted, or P(Y)), whereas one signal is transmitted on $L2$, which is designated for DoD-authorized users. The signal components associated with DoD-authorized users is neglected in this thesis since the focus is on SPS-users. Therefore, this discussion will only be limited to the $L1$ signal. Each SV transmits at the same $L1$ center frequency, 1575.42 MHz. That is, 154 times the GPS fundamental frequency of 10.23 MHz. The capability of each satellite to broadcast on the same frequency is an inherent benefit of exploiting CDMA technology. CDMA provides efficient use of the spectrum and enables all users to fully utilize the entire GPS bandwidth. While communication systems use CDMA mainly for its inherent security and time synchronization capabilities, GPS utilizes it for ranging.

However, in order to exploit the advantages of CDMA, each SV must transmit its own unique ranging code. Each service, SPS and PPS, have their own associated ranging code. Both codes are essentially binary codes called PRN sequences or PRN codes. The SPS codes are referred to as C/A-codes and PPS codes are referred to as precision (encrypted) codes, or P(Y)-codes. Since this research is limited to GPS-SPS, the PPS codes are no longer considered. Each satellite transmits its own C/A-code. Each C/A-code is a pseudorandom sequence of 1,023 bits, called *chips* in the context of GPS. The period of the C/A-code is 1 Millisecond (ms). The C/A-code chip has specific properties, which are critical in developing the C/A- code

generator in hardware. First, the rate of the C/A-code chips, called the *chipping rate*, is 1.023 MHz [or Mcps]. Thus, the period of each chip is approximately 977.5 Nanoseconds (ns). The purpose of the C/A-code is two-fold [1]. First, it allows multiple satellites to broadcast on the same frequency. Second, it is used by the receiver to calculate precise ranging from the user to each satellite. The C/A-code belongs to the family of Gold codes. Gold codes are generated by the combination of delayed maximum-length sequences and provide families of codes with favorable cross- and auto-correlation properties [1]. The auto- and cross-correlation properties of a PRN sequence are exploited in GPS receivers during acquisition and tracking. These concepts are critical to building the C/A-code generator and will be further reviewed in the hardware implementation of the C/A-code generator. For further details on the generation of the C/A-code refer to reference [5].

Modulo-2 added on the C/A-code is the 50 bps binary-coded navigation data message. The navigation message comprises data on the satellite health status, ephemeris (satellite position and velocity), and clock bias parameters associated with the on-board satellite atomic clock [1]. In addition, it includes an almanac to provide the receivers with reduced-precision ephemeris data on all satellites in the GPS constellation. Each bit in the navigation message has a duration of 20 ms. The essential clock parameters and ephemeris data are repeated each thirty seconds. The data bits are generated in synchronous to the C/A-code. The navigation message structure is illustrated in Figure 2. As shown in this illustration, the data message structure is transmitted in five 300-bit subframes, 1 through 5, which equates to a 1,500-bit long frame. Each subframe is composed of ten 30-bit words, where the last 6 bits in each word of the navigation message are used for parity checking to provide the GPS receiver with a method of detecting bit errors upon transmission. The full data message comprises 25 full frames since subframes 4 and 5 are subcommutated 25 times. Hence,

1	TLM	HOW	SV Health/Accuracy and Clock Corrections
2	TLM	HOW	Ephemeris Data (Orbital Information)
3	TLM	HOW	Ephemeris Data (Orbital Information)
4	TLM	HOW	Almanac, Ionospheric Model, UTC Info
5	TLM	HOW	Almanac

Figure 2. 50 bps Navigation Data Message Structure

the total time for transmission is roughly 12.5 minutes. However, since the first three subframes contain the necessary clock and ephemeris data, decoding one full frame, which is approximately 30 seconds, is sufficient for a position calculation. Also shown in this illustration are the first two words of each subframe, Telemetry (TLM) data and Handover Word (HOW), respectively. The TLM contains a known 8-bit sequence called the preamble that is used to synchronize the frame data. The HOW provides the precise GPS Time-of-Week (TOW) that will be valid at the starting edge of the next subframe. Therefore, TOW updates are available every 6 seconds.

The composite binary signal resulting from the modulo-2 addition of the navigation message and the PRN ranging code is impressed upon the carrier through modulation. The specific form of modulation used is called Binary Phase-Shift Keying (BPSK). BPSK is a digital modulation scheme that conveys data by changing the phase of a carrier wave. It essentially uses two phases which are separated by 180°.

The effect of the modulation on the carrier is best shown in Figure 3. Essentially, the phase of the carrier is shifted by 180° whenever the chip transitions from 0 to 1 or 1 to 0. The purpose of modulating the carrier with a binary code has several advantages:

1. This modulation spreads the signal energy, initially concentrated at a single frequency, over a wide frequency band. This is referred to as Direct Sequence Spread Spectrum (DSSS). For C/A-code, the spread occurs over 2 MHz, centered at the L1 carrier frequency. Measuring the null-to-null bandwidth of the main lobe in the spectrum is one way to visually verify if the C/A-code is generated correctly.
2. Chip transitions can be used for timing/ranging calculations.
3. Enables the ability to employ CDMA and have all SVs transmit on the same frequency.
4. The signal energy can be de-spread in a receiver, if the code is known. De-spreading the C/A-code is used to decode the data bits.

This section has described the signal characteristics specific to GPS-SPS. Essentially, GPS-SPS signals are constructed by BPSK modulating a L1 RF carrier with a binary code. The RF carrier frequency is 1575.42 MHz. The binary code is a composite of the modulo-2 addition of the 50 bps navigation message and the PRN ranging code. The PRN code is 1,023 chips long with a chipping rate of 1.023 Mcps. The signal characteristic properties are crucial to understanding the hardware implementation of this research effort. However, this only constitutes the theoretical principles of the signal properties.

GWPP will essentially become an architecture to generate multiple GPS waveforms. In fact, this platform is designed to replicate the satellite transmission envi-

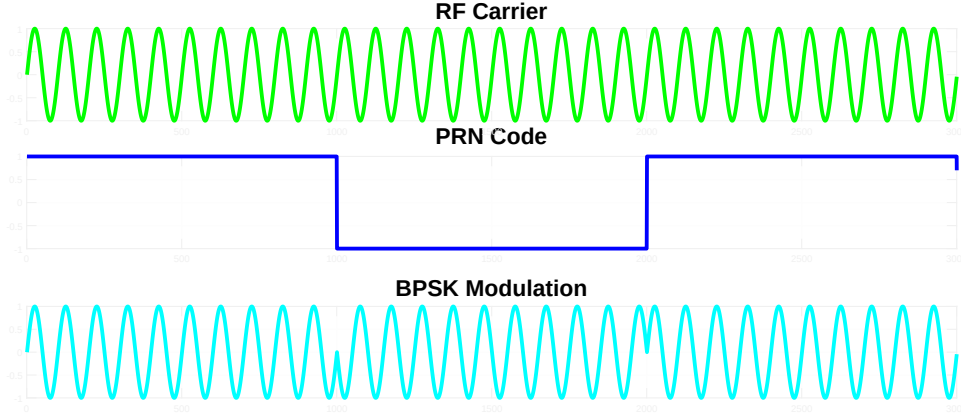


Figure 3. Binary Phase Shift Keying Modulation

ronment in a SV on a hardware platform to be used for advanced signals research. GPS satellites transmit multiple signals at once, both for SPS and PPS users. However, for the scope of this research effort, the current architecture is limited to first show software-defined GPS signal synthesis capability through the synthesis of one signal: GPS L1 C/A signal, which its signal characteristics were recently discussed in this section. However, in order to build a platform to generate GPS signals, a signal synthesis technique must be incorporated into this design.

2.2 Signal Generation Techniques

A signal generation technique refers to the process involved in which specific characteristics such as phase, amplitude, and frequency of a waveform is generated. Although in signal synthesis discussions, the main consideration is in the technique used to generate the frequency; other characteristics such as amplitude and phase are typically a byproduct of external elements. Since this research focuses on GPS-SPS signals, which are essentially composed of BPSK modulated sine waves, this section deals with emerging modern digital techniques utilized in the generation and modulation of sine waves. Equally as important to signal generation is accuracy since the application of this platform is to be used for advanced signals research to perform both

development and verification of advanced GPS signals. Therefore, the generation of accurate waveforms plays a crucial role in the design of this architecture. In order to decide which signal generation technique is advantageous for this architecture, it is important to understand the conventional signal generation techniques currently being used for signal synthesis. Currently, there are several conventional techniques being used for signal synthesis: indirect analog, indirect digital, direct analog, and direct digital. This section begins with the high-level overview of each technique followed by a comparative analysis. This section strictly focuses on comparing the techniques to exemplify the reasoning for choosing a direct digital-based system for the signal generation component of this architecture. The actual theory and operation of direct digital synthesis systems is presented in Section 2.3. However, before beginning this discussion, it is worthwhile to first discuss the specifications used to compare each technique.

2.2.1 Signal Synthesis Specifications.

Like any design, the components inside the design infrastructure must meet specifications suited for the design application. As a GPS waveform prototyping platform, this architecture must be flexible to accommodate for both current and modernization architectures of GPS. Although the scope of this research is limited to the synthesis of the GPS L1 C/A signal, it is important to design a flexible architecture to facilitate future expansion. In the case for this project, future expansion includes the capability of generating multiple GPS signals. Therefore, this requires a signal synthesis technique with specific specifications to not only meet the requirements for this thesis, but also facilitate expansion. The list of common specifications important to this architecture are described below

Frequency Range specifies the dynamic range of the synthesizer to include the low-

est and highest frequencies obtainable. The technique must be able to synthesize RF carriers.

Frequency Resolution specifies the minimum step size of the frequency increment.

It is essential to have fine frequency resolution to accurately generate the GPS center frequencies.

Cost of Implementation specifies the resources involved to implement the signal synthesis technique.

Switching Speed specifies the speed at which the signal synthesizer changes from one frequency to another. Although this characteristic is not vital to the generation of the GPS L1 C/A signal, it is essential for other GPS signals where the rate of the binary code changes to apply another modulation scheme.

Phase Continuity specifies the transient state of the phase. Since the phase of sinusoidal signals follow a linear function, an architecture that provides smooth phase increments is preferred.

Harmonics specifies the level of harmonics as a result of signal synthesis. A harmonic is refers to the spectral content of a frequency that is an integer multiple of the fundamental frequency. This specification is expressed in Decibels Relative to Carrier (dBc) and will be used to analyze the performance of this architecture (Chapter IV).

Spurious Emission specifies the level of any discrete, periodic output frequency spectral lines that are not related to the fundamental carrier. This is also expressed in dBc and will be a consideration factor in the performance analysis of this architecture (Chapter IV).

Phase Noise specifies the level of small random fluctuations or uncertainty in the phase of a signal. It is an attribute of all real world signals and corrupts the signal in various ways such as loss of signal energy. These effects ultimately present challenges during the acquisition and tracking of GPS signals in user equipment. Therefore, this specification is crucial for this application. However, in order to measure phase noise, the phase noise of the test equipment used to verify the performance must be at least 10 dB better than the phase noise of the device under test. For reasons discussed in Chapter III, this is not a metric used to evaluate this platform.

Presented above are the specifications used to determine which signal synthesis technique befits the GWPP architecture. An overview of each technique is presented first beginning with direct analog. Next, indirect analog/digital techniques will be discussed followed by direct digital. Afterwards, a comparative analysis of the three conventional techniques is presented.

2.2.2 Direct Analog Synthesizers.

In direct analog frequency synthesis, a group of reference frequencies are derived from the main reference. Essentially, as depicted in Figure 4, direct analog synthesis relies on several stages of mixing and filtering. Different references and filters are utilized in each stage to create higher or lower output frequencies. Afterwards, the resulting signal after the last filter is sent to a divider. The divider has both an advantage and disadvantage. The advantage is that using a divider reduces costs due to fewer references needing to be generated. However, the disadvantage is that this configuration introduces phase ambiguity [6].

In direct analog synthesis, to increase frequency resolution requires additional circuitry, which increases implementation costs. However, it is important to mention

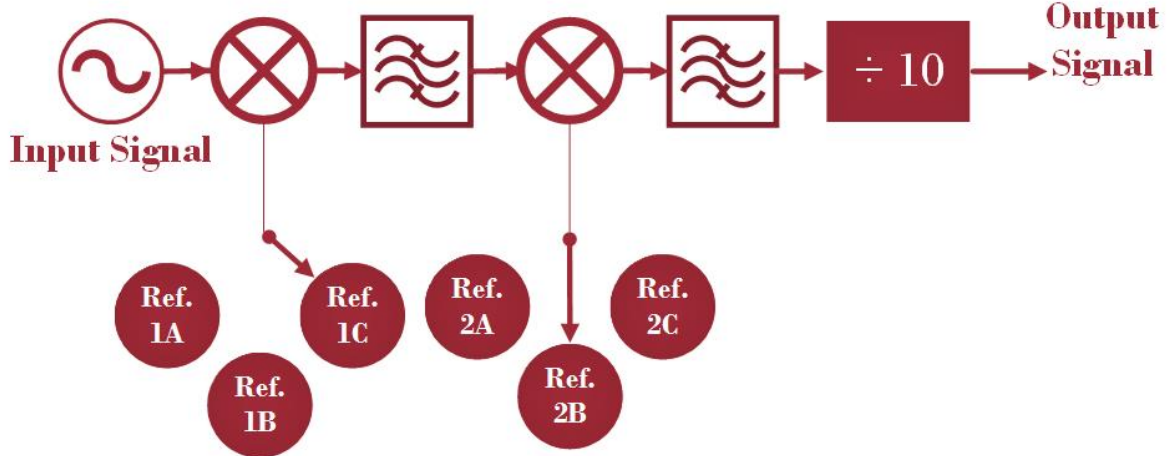


Figure 4. Direct Analog Synthesizer Architecture

that in this configuration, the quality of the output signal correlates directly with the quality of the input source. Hence, the reason for calling this a direct method. As a result of the digital process, phase noise performance, particularly close to the carrier, is excellent. In addition, this configuration yields very fast switching speeds. Typically, references are used to switch among frequencies. Unfortunately, if the application requires several different frequencies, the number of references as well as filters increases, which makes this implementation extremely expensive.

2.2.3 Indirect Analog/Digital Synthesizers.

Indirect frequency synthesis is based around phase lock loop technology. The final output signal is generated by an oscillator that is controlled by other signals. For this reason, the output signal is generated indirectly. Indirect analog and indirect digital both rely on PLL technology. The only difference is in what additional components (digital or analog) are used with the phase lock loop to generate the output frequency. Since both rely on PLL technology, this discussion is limited to the basic theory of operation regarding the PLL synthesizer. A PLL synthesizer is a feedback circuit that locks its output frequency to a reference. PLLs consist of a Voltage Controlled

Oscillator (VCO), phase detector, charge pump, loop filter, and various dividers. The loop filter can be configured as either a first, second, or third order filter. However, for the sake of brevity, the architecture of a conventional single-loop PLL synthesizer is shown in Figure 5. PLLs work by comparing the output of the VCO with a reference. As the output drifts, errors are detected, which results in corrections in the form of voltages. The voltages are fed back to the system from the loop filter and applied to the VCO. The errors and corrections occur in the phase detector, which adds phase noise close to the fundamental carrier. Due to the entire process of generating the output frequency, it is evident that this configuration does not yield fast switching speeds.

2.2.4 Direct Digital Synthesizers.

Parallel to the advance of traditional PLLs, there emerged another synthesis technique called DDS, which is mainly digital in nature. This term is also synonymous with Numerically Controlled Oscillator (NCO). In this technique, the generation of waveforms are performed digitally, meaning all of the signal parameters such as phase, frequency, and amplitude are generated digitally. Hence, this technique is deterministically controlled and fundamentally different than PLL and direct analog synthesizers. Although both indirect analog/digital and digital analog techniques partially rely on digital devices such as dividers and phase detectors to generate frequencies, the overall concept of operation is based on analog principles. That is, an oscillator, which is essentially a feedback-tuned amplifier, is configured to operate under certain conditions that once manipulated generates a range of frequencies. Thus, these methods introduce a level of instability. Alternatively in DDS, the signal is generated and manipulated digitally, where the signal is represented by digital numbers and converted to an analog waveform via a DAC. The architecture of a DDS system

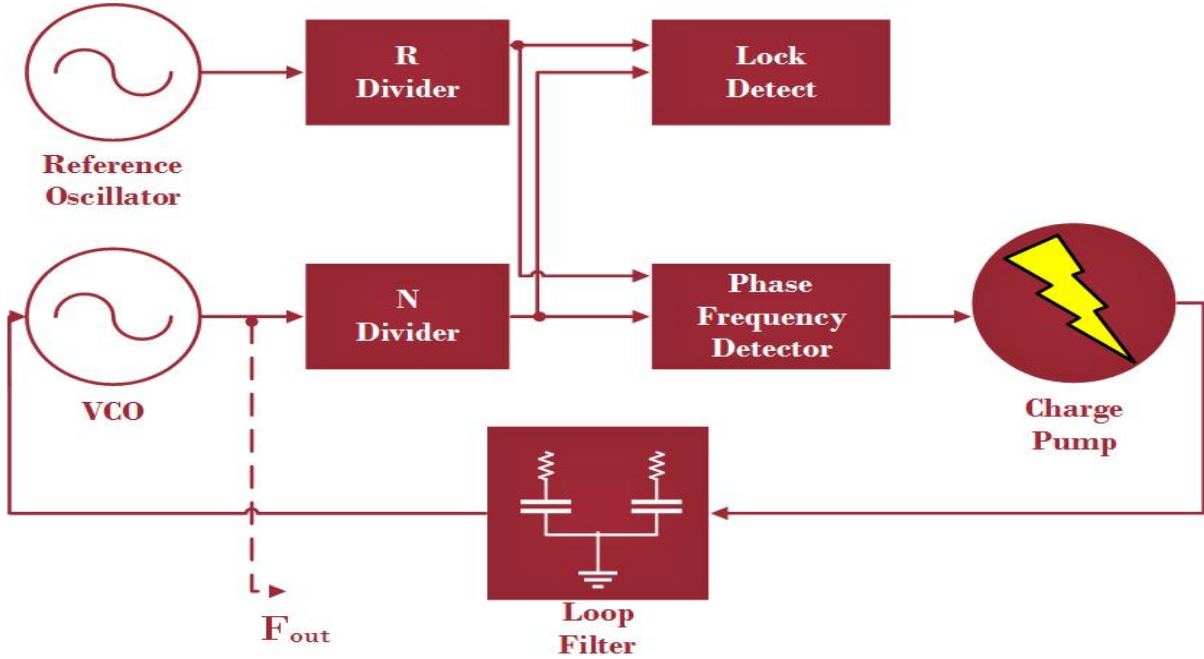


Figure 5. Phase Lock Loop Synthesizer Architecture

is shown in Figure 6.

From Figure 6, it can be seen that DDS utilizes the combination of logic, memory, and a data conversion device to generate its output waveform. The logic and memory circuitry is used to digitally construct the desired output signal, whereas the data conversion device converts it from a digital form to analog. Typically, a reconstruction anti-aliasing filter is employed after the DAC to smooth the output from the DAC and remove any out of band signals such as aliases and harmonics coming from the DAC. Nonetheless, the construction of the signal in this configuration is almost entirely digital. The logic and memory circuitry consists of a phase accumulator and a phase to amplitude converter. The phase accumulator determines the resolution and is controlled by an input word. It essentially generates phase samples and the phase-to-amplitude converter translates them into digital amplitude representations. There are several different methods for performing the phase to amplitude conversion. Some methods include Coordinate Rotation Digital Computer

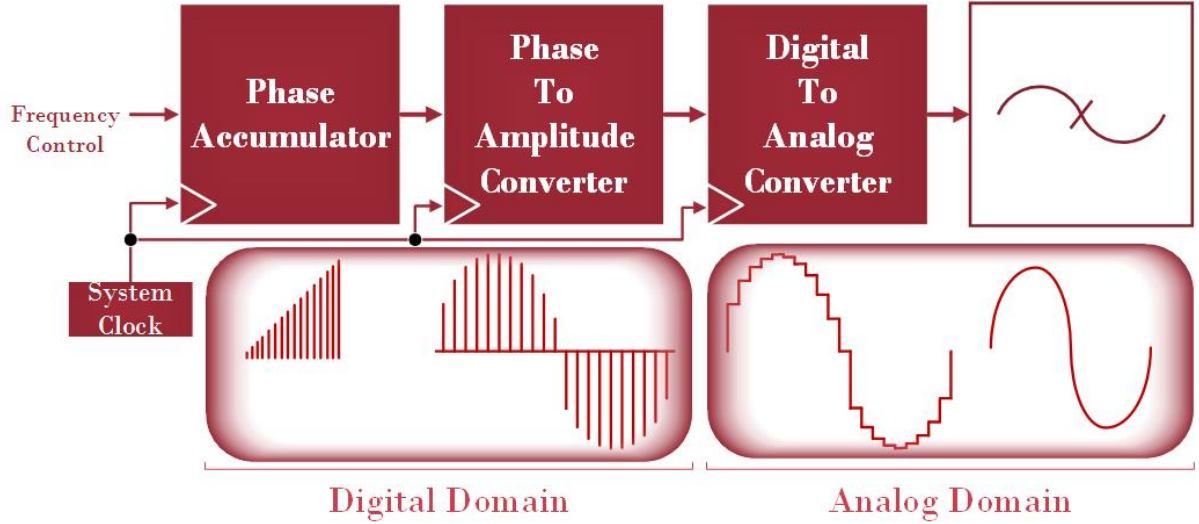


Figure 6. Direct Digital Synthesis High Level Architecture

(CORDIC) Algorithm, Sunderland Algorithm, Direct Taylor Approximation, Read Only Memory (ROM) Algorithm, to name a few [6]. However, the most intuitive is the ROM algorithm in which it is essentially a Lookup Table (LUT) consisting of amplitude values corresponding to the desired waveform. Although different LUTs can result in the generation of different waveforms such as triangle, square, sine, the most popular is the sine wave configuration. Since DDS is a primarily digital synthesizer, it offers excellent switching speed, frequency resolution, and phase noise performance. However, there are various limitations to this approach. First, DDS covers an operating range limited by the sampling theory; practically, the output is limited to approximately 45% of the maximum clock rate at which the logic operates. The second limitation is spectral purity, which is a function of the quantization effects from the phase accumulator and ROM as well as the properties of the DAC.

2.2.5 Comparative Analysis.

Due to a continuous evolution in technology, all three techniques have been available to designers for decades. However, it is the DDS-based systems that are evolving

most rapidly today. With the generation of accurate waveforms as a crucial component in this architecture, it is imperative to choose a technique that best meets the specifications important to this research. Three main conventional signal generation techniques were presented. In each respective presentation, the terms frequency resolution, switching speed, phase noise, spectral purity were used as evaluation metrics. One last considering factor is flexibility and cost in each implementation. Recall that in GPS signal generation, two concepts are required: signal generation and modulation. Therefore, the technique must also offer a cost-effective method for applying modulation. In the case for the scope of this research, the technique must offer a cost-effective solution for phase modulation since this is the modulation technique applied to the GPS L1 C/A signal. In indirect analog/digital synthesizers, phase modulation is performed by adding a baseband voltage along with the output of the loop filter into the VCO tuning terminal. To perform this simple modulation requires additional analog circuitry. With phase modulation in direct analog synthesis, it also requires additional analog circuitry. The addition of analog circuitry increases design costs and is impractical for this research effort. In DDS, since this technique offers total control of the signal parameters, the ability to add modulation is easier compared to the other techniques. To add phase modulation in DDS essentially requires an adder to be included between the phase accumulator and the ROM. The input to the adder effectively combines with the output of the phase accumulator to shift the phase. With these considerations in mind, DDS is the most architecturally suitable to implement for this architecture. In addition, it provides persuasive advantages that are difficult or expensive to achieve with alternative signal synthesis techniques. These advantages include fine step sizes (fine frequency resolution), excellent phase noise, phase continuity, to name a few. Not to mention that operating as a fully digital synthesizer offers extraordinary flexibility as a modulator with consideration

in size and implementation costs. For this reason, DDS is the chosen signal generation technique for this platform. This section focused on the high level overview of DDS. However, it is vital to the understanding of the hardware implementation of this platform to understand the theory of operation. For this reason, the next section provides insight to DDS theory of operation.

2.3 Direct Digital Synthesis

DDS, as a new signal synthesizer technique, has been an interesting field of study since it was first developed in 1971 by Tierney, Rader, and Gold [7]. It essentially produces an analog waveform—typically a sine wave—by generating a time-varying signal in digital form and then converting it to its analog form using a DAC. Because operations within a DDS system are primarily digital, it can offer precise control of signal characteristics such as amplitude, phase, and frequency as well as modulation. Hence, the output waveform of this technique is inherently deterministic. For this architecture to be successful, the ability to accurately produce and control waveforms is a key requirement. With the comparative analysis presented on the three conventional techniques of signal synthesis, it is evident to why DDS is chosen for this architecture. In order to understand the DDS implementation in this platform it is essential to understand the basics of DDS, which includes the theory of operation, architecture, and typical performance metrics. The rest of this section focuses on providing background information on these topics. First, the DDS theory of operation is presented. Followed by the theory of operation is the DDS architecture. Finally, this section concludes with the performance metrics typically associated with DDS.

2.3.1 Direct Digital Synthesis Theory of Operation.

DDS, at its basic nature, is derived from DSP principles in the sense that it is based on the sampling theorem. The theorem states that any signal confined in a band-limited spectrum can be reconstructed from discrete samples in time, provided that the signal is sampled at a rate $F_s > 2F_{max}$, where F_{max} is the maximum frequency in the signal to be sampled to prevent aliasing. This sampling rate is sufficient to avoid *aliasing*, or false replicas. In DDS, this process is essentially reversed. Instead of sampling waveforms, the samples are being generated. In essence, DDS is generating samples based on the sampling rate, which is determined by the input clock provided to the DDS system. This can be generalized through the explanation of the DDS architecture.

2.3.2 Direct Digital Synthesis Architecture.

A DDS system is a digital circuit that produces an output signal in which its frequency is set by means of a multi-bit digital control word. The architecture of a DDS system is illustrated in Figure 7. Based on Figure 7, the architecture comprises three main building blocks: accumulator, phase-to-amplitude converter, and a DAC. The combination of the accumulator and phase-to-amplitude converter is commonly referred to as an NCO. Figure 7 also demonstrates the signal flow in a DDS system, which clearly illustrates that the phase is generated in the accumulator, mapped to an amplitude value via the phase-to-amplitude converter, and then converted to its analog counterpart via the DAC. It is important to understand two high-level concepts regarding the architecture of a DDS system. First, DDS systems typically require at least two inputs: A system clock signal with F_s denoting its sampling frequency, and a digital numeric value, which sets the frequency of the output signal. The terms system clock and sampling clock will be used interchangeably. For the

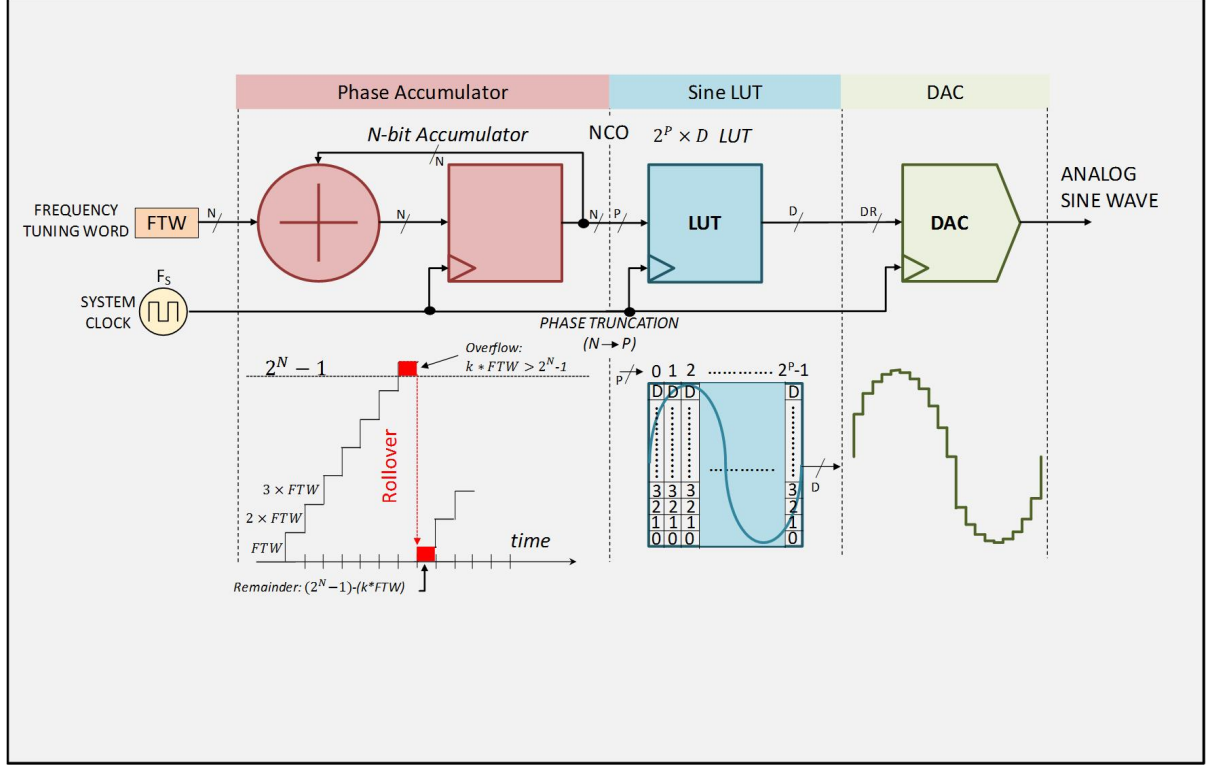


Figure 7. Direct Digital Synthesis Theory of Operation

current and future discussions of DDS, the digital numeric value will be denoted as the Frequency Tuning Word (FTW). Thus, this name is intuitive in the sense that it controls and tunes the frequency of the output waveform in a DDS system. It is also important to keep in mind that DDS is limited by the system clock. Therefore, the output frequency of the DDS system, as defined by FTW, spans a range from Direct Current (DC) to half of the system clock. The second important concept is that the overall shape of the output waveform is determined by the amplitude values stored in the phase-to-amplitude converter. However, since this architecture is designed for GPS applications, the contents stored in the phase-to-amplitude converter represent amplitude values corresponding to one cycle of a sine wave. The design of each component in the DDS system is essential to both the accuracy and spectral purity of the output waveform. In this thesis, spectral purity refers to the generation of spurs and harmonics. Thus, it is important to proceed with further discussions covering

the design implementation of each component. However, before this discussion begins with the accumulator, it is worthwhile to review the overall concept of DDS one last time. First, DDS is a digital signal synthesis technique meaning its waveform characteristics such as phase, amplitude, and frequency are controlled and generated digitally. Hence, it is fundamentally different from the theory of operation with respect to a PLL or direct analog synthesizer. The combination of the accumulator and phase-to-amplitude converter constitutes a NCO. The NCO produces a time series of digital words at a rate corresponding to the frequency of the system clock. The time series of digital words represent a sine wave sampled at a rate equal to the system clock frequency and having a frequency dictated by the FTW. Afterwards, the DAC converts this output to an analog waveform representing a real-world signal. As mentioned previously, it is important to understand the role of each component in the generation of the output signal. Therefore, the following discussion will cover the theory of operation of each component; first, starting with the phase accumulator. Afterwards, the phase-to-amplitude converter will be discussed, followed by the discussion of the DAC.

The ***phase accumulator*** is the heart of a DDS system. As the name implies, it essentially serves as the phase generator and determines the resolution as well as the frequency of the DDS system. In addition, it is one factor in the generation of spurious signals in the output spectrum, which will be a performance metric used to evaluate this platform. An accumulator is essentially a bank of Flip Flops (FFs) with a digital adder connected in a feedback configuration. All of the FFs share the same clock input, namely the system clock. The number of FFs in the accumulator constitutes the width of its input and output buses. An accumulator with N -FFs occupies an N -bit bus, which requires the FTW to be N bits. An N -bit FTW is capable of representing 2^N unique digital codes corresponding to numeric values ranging from

zero to 2^N minus one. The theory of operation of the accumulator is also illustrated in Figure 7.

At the rising edge of the system clock, the FTW value is added to the previous accumulated result. Subsequent rising edges of the system clock cause the accumulation process to repeat. The accumulator output increments in a staircase function that translates to a linear ramp whose rate of change is defined by FTW. Eventually, however, the accumulator output will grow until it reaches or exceeds the upper limit of the accumulator, 2^N . At that point, it overflows and the overflow amount is a remainder. The remainder essentially represents the starting phase for the next period. Therefore, the accumulator counts from zero to $2^N - 1$ since any accumulation above this number results in an overflow. The theory of operation behind the accumulator is genuinely used to represent the generation of linear phases corresponding to a sine wave; consider the following: if 0 equates with zero phase and 2^N with 2π , then this device generates phase from 0 to 2π . The point at which the accumulator overflows represents the generation of one full period of a sine wave. Hence, the rate at which the accumulator overflows determines the frequency of the output sinusoidal waveform. Therefore, the concept of an overflowing accumulator is critical to comprehending how the accumulator establishes the output frequency of the DDS system.

In essence, DDS relies on the behavior of an overflowing accumulator to establish the output frequency. This process depends on the frequency of the system clock and the value of the FTW. Before discussing how the FTW is calculated, it is worthwhile to mention that the behavior of the accumulator exhibits two beneficial features. First, the sequence of operation with respect to the accumulator results in the generation of a repeating sequence of digital values uniformly separated in time by the period of the system clock. This reinforces the concept that DDS-based systems

can generate a constant frequency. Secondly, the accumulation of an unchanged FTW provides numeric control between adjacent digital values; ultimately, providing the generation of smooth phase increments from cycle to cycle as well as control of the overflow period of the accumulator.

It has been established that the FTW controls the step size between consecutive accumulator output values. The point at which the accumulator overflows is dependent upon the capacity of the accumulator, which is determined by the bus width of the accumulator. For an N -bit accumulator, the capacity is equivalent to 2^N . Thus, any accumulation output greater than 2^N results in an overflow. As mentioned previously, the concept of an overflowing accumulator is essential to understanding how the accumulator establishes the output frequency. It is important to note that the capacity, system clock, and FTW are all essential characteristics in DDS systems. The importance of each term is generalized in the following discussion. Recall that the phase accumulator essentially accumulates phase values in the range from 0 to 2π effectively producing a sine wave oscillation. To understand this basic function, visualize the generation of the sine wave as a vector rotating around a digital phase wheel. The digital phase wheel representation is shown in Figure 8. The number of phase points in the wheel depends on the accumulator size. This is generalized in Table 2. As the vector rotates around the digital phase wheel, visualize that the

Table 2. Direct Digital Synthesis Number of Phase Points

N	Number of Phase Points
4	16
8	256
16	65535
32	4294967296
64	18446744073709551616

contents of the phase accumulator represent the points on the cycle of the sine wave and thus a corresponding output sine wave is being constructed. In DDS, the output

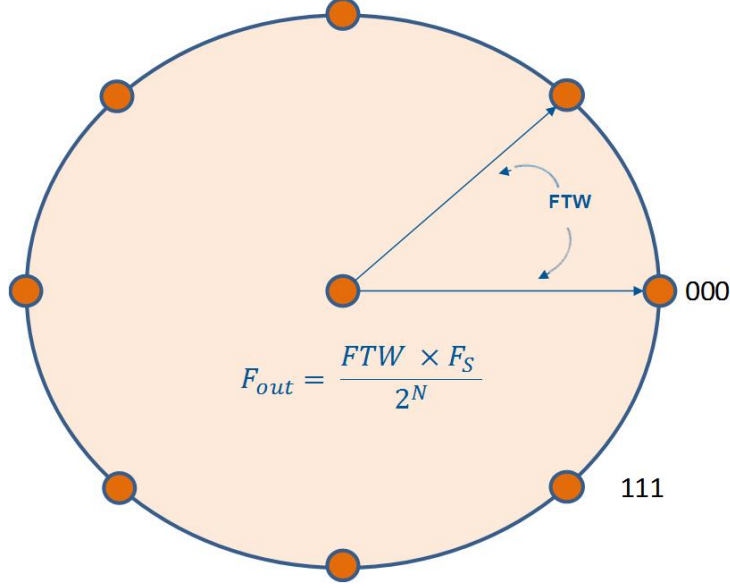


Figure 8. Direct Digital Synthesis Phase Wheel. One complete revolution represents one period of a sine wave.

frequency calculated

$$F_{out} = \frac{F_s \cdot FTW}{2^N} \quad (2.3.1)$$

where F_{out} represents the DDS output frequency; F_s represents the system clock; N is the accumulator resolution; and FTW is the frequency tuning word. Recall that 2^N is the capacity of the accumulator. Furthermore, FTW is always less than capacity, therefore the quantity FTW divided by 2^N is always less than one. Hence, this implies that F_{out} is always less than F_s . Equation (2.3.1) illustrates how to calculate the DDS output frequency, F_{out} , given N , FTW , and F_s . However, Equation (2.3.1) can be rearranged to

$$FTW = \text{round} \left[2^N \left(\frac{F_{out}}{F_s} \right) \right] \quad (2.3.2)$$

Equation (2.3.2) is referred to as the alternate DDS equation and allows the calculation of FTW given F_{out} , F_s and N . The *round* function is needed due to the fact that FTW is an integer by definition. The presence of the round function is significant in the sense that it implies that a DDS-based system is not always capable of

generating the desired F_{out} frequency. This is the case when the quantity calculated using Equation (2.3.2) is not an integer. The alternate DDS equation is essential in this platform due to the fact that the output frequencies which correspond to the chipping rate and carrier frequencies are known. For this reason, the alternate DDS equation is primarily used in this research to calculate the FTW value. It is important to note that since DDS is based on the sampling theorem, FTW cannot exceed the Nyquist frequency. Another important DDS equation is the frequency resolution of the accumulator. The frequency resolution is calculated by

$$Frequency\ Resolution = \frac{F_s}{2^N} \quad (2.3.3)$$

The importance of frequency resolution as it relates to phase truncation is generalized in the following example. Consider a 7-bit accumulator. With this accumulator, the phase resolution is $1/128^{\text{th}}$ of a full circle, or 2.81° ($360^\circ/2^7$). If only the most 4 bits of the accumulator are used to convey the phase information, then the resolution becomes $1/16^{\text{th}}$ of a full circle, or 22.50° ($360^\circ/2^4$). Now, assume a FTW equal to 6 is used. On the first phase step, a discrepancy arises between the phase of the 7-bit accumulator and the phase as determined by the 4-bit resolution. This discrepancy results in a phase error of approximately 16.86° ($6 \times 2.81^\circ$). The pattern repeats as the vector rotates around the phase wheel. Therefore, it can be seen that phase truncation is an important aspect of DDS systems. The evaluation of phase errors due to phase truncation with respect to DDS systems are presented in Chapter IV. It is important to note that based on Equation (2.3.3), faster sampling rates along with larger accumulators will yield excellent frequency resolution. Excellent frequency resolution decreases the phase errors associated with the relationship between the number of phase points and the FTW. This concept is important if the application requires a resolution less than a certain number of

Hertz (Hz).

In summary, the accumulator is the heart of a DDS system. FTW, sampling clock, and accumulator size are all very important considerations. The frequency resolution is determined by the sampling clock and the accumulator size. The frequency of the system clock and the value of the FTW are utilized by the accumulator to establish the DDS output frequency. The output frequency is essentially based on the behavior of an overflowing accumulator. To understand this basic function, it is helpful to visualize the construction of the sine wave as a vector rotating around a digital phase wheel where each designated point on the phase wheel corresponds to the equivalent point on a cycle of a sine wave. One complete revolution of the vector around the digital phase wheel, at a constant rate, results in the generation of one cycle of the output sine wave. The accumulator resolution determines the number of phase points contained in the phase wheel. The output of the accumulator increments in a staircase manner, which translates to a linear ramp whose rate of change is defined by FTW. The accumulation of a constant FTW results in the generation of an output waveform with constant frequency. Typically, this output waveform is a sine wave. However, in order to use the output to generate a sine wave, a phase-to-amplitude converter must be used.

The *phase-to-amplitude* converter is the building block that enables a DDS system to generate an output waveform. The algorithm or configuration of the phase-to-amplitude converter determines the shape of the output waveform, whether it is square, sinusoidal, triangle, etcetera. However, since this platform is specific to GPS waveforms, this thesis only focuses on configuring the phase-to-amplitude converter to generate an output sine wave. As previously mentioned, DDS, since its conception in 1971 by Tierney, Rader, and Gold [7], has gained great interest in literature as a signal synthesis technique. As a result, many authors have experimented with differ-

ent phase-to-amplitude converters. Originally, Tierney, Rader, and Gold designed the architecture of a DDS system with the phase-to-amplitude converter implemented as a sine LUT in ROM. The name is given for two reasons. First, a LUT is an array that replaces runtime computations with precalculated and statically placed values. ROM refers to static memory that performs read-only operations. Hence, the term sine LUT ROM essentially means the construction of a read-only LUT populated with precalculated sine amplitude values. This approach is by far the most popular in literature [8] and [9], to name a few sources. However, other implementations of the phase-to-amplitude converter are COordinate Rotation DIgital Computer (CORDIC), Sunderland, Hutchison, and Direct Taylor Approximation algorithms, [6]. Each of these methods have their specific advantages and disadvantages. However, the GWPP architecture focuses on implementing the phase-to-amplitude converter using the sine LUT method. Therefore, the other methods are no longer considered and the discussion proceeds with the phase-to-amplitude converter implemented as a sine LUT ROM.

The sine LUT is connected between the phase accumulator and the DAC. That is, the output bus from the phase accumulator is connected to the input of the sine LUT while the output of the sine LUT is connected to the input of the DAC. It essentially maps the phase, ϕ , to its sine amplitude value, $\sin \phi$. In an ideal DDS system, the width of the input bus to the sine LUT, P , is the same as the width of the output bus from the accumulator, N . That is N equals P . However, this is impractical if N is large. Consider the case for $N = 32$. This would amount to over 4 billion entries inside the LUT. In addition, if each entry is stored with 8-bit accuracy, then 4-gigabytes of LUT memory would be required. Hence, it would be impractical to implement such a design. For this reason, a truncated version of the instantaneous output value of the accumulator is fed to the input of the sine LUT.

As presented in the discussion of the accumulator, phase truncation is an important aspect of DDS architectures. It ultimately generates its own set of spurious emissions due to quantization errors produced from the truncation. In the truncation process, it is generally the case in DDS to truncate the least significant bits of the phase accumulator and retain the P most significant bits to index the LUT. Intuitively, the truncation of the least significant bits corresponds to the truncation of the fractional bits. Although it significantly reduces the size of the LUT in ROM, it introduces a degree of quantization/truncation error. Hence, the output of the ROM can never represent a perfect sine wave since an error is already present at the input of the LUT.

The input and output bus widths are generally of differing widths. Typically, the width of the output bus corresponds to the number of bits associated with the resolution of the DAC. However, this also results in a considerable amount of resources required for the ROM. For this reason, the number of bits used to represent an amplitude value stored in the LUT are typically less than the resolution of the DAC. Additionally, techniques such as exploiting the symmetrical properties of sine waves are used to compress the ROM requirements. The DAC theory of operation is discussed next, however, before this discussion it is important to review some key concepts of the sine LUT. First, the overall concept of the sine LUT is to convert the phase information into an amplitude value. The index to the sine LUT is the truncated version of the instantaneous output of the phase accumulator. Phase resolution refers to the number of bits used to index the LUT. It ultimately determines the number of phase points contained in one cycle of the output sine wave. Second, the input and output bus widths are generally of different widths. The output corresponds to a predetermined amplitude value at that phase point. Amplitude resolution refers to the number of bits used to represent an amplitude value. It typically coincides with

the DAC resolution, however, this is often not the case due to finite memory resources. Third, to represent an ideal sine wave with the least spurious output, the input bus width to the sine LUT is equal to the accumulator size and the output bus width is equal to the DAC resolution. However, this constitutes extremely large ROMs with high power consumption, which is impractical for most design implementations. As a result, the input and output bus widths are truncated. The truncated inputs and outputs create truncation errors, which translate to spurious emissions in the spectral output. The spurious emissions degrade the quality of the output signals in a DDS system. The evaluation of spurious emissions due to finite amplitude resolution and phase truncation are two performance metrics presented in Chapter IV to analyze the GWPP platform.

The **DAC** converts the digital amplitude word fed from the sine LUT to its analog form. The resolution of the DAC is specified by the number of its input bits. For instance, the resolution of a DAC with 14 input bits is referred to as having a DAC with “14-bit resolution”. Since there are additional quantization effects with the DAC, it also produces spurious content in the spectrum of the output signal. The DAC has a huge impact on the range of frequencies able to be generated in a DDS system. Recall that the accumulator, phase-to-amplitude converter, and DAC are all clocked at the same clock rate; namely, the system clock. In the accumulator, this clock is referred to as the sampling clock and with consideration in the sampling theorem, it limits the highest frequency content achievable. In the DAC, the system clock is the sampling clock. Therefore, the choice of the DAC plays a vital role on the maximum attainable frequency for the DDS system design.

The choice of the DAC and accumulator size in the GWPP architecture are what mainly separates this DDS-based design from other designs presented in literature. While the majority of sources incorporate a DAC with maximum frequency of 1

Gigahertz (GHz), this architecture incorporates a DAC with a max frequency of 2.85 GHz. The choice of DAC used in this architecture enables the ability to generate RF carriers in the L-band without analog upconversion or PLL techniques. Whereas, the sampling theory limits the DAC using a 1 GHz sampling clock from generating any frequency content above 500 MHz. Therefore, additional circuitry is required to synthesize GPS carriers. However, in order to exploit the advantages of the DAC chosen for this research, the logic design for the DDS architecture must be designed accordingly, which is explained in Chapter III.

2.4 Fundamentals of Field Programmable Gate Arrays

DDS is not a computationally aggressive implementation in the sense that the only design blocks in this system are the phase accumulator and phase-to-amplitude converter. The DAC is typically an external device manufactured by a company. Therefore, in implementation terms, this device is not considered when choosing a platform to host the DDS system. The other DDS components such as the phase accumulator and phase-to-amplitude converter are simple to design. The phase accumulator is implemented using an N -bit adder and register combination, while the phase-to-amplitude converter is implemented using ROM. However, as mentioned previously, amplitude and phase resolution affect the spectral purity of the output signal. Since the generation of accurate waveforms are crucial to this thesis, it is desirable to find a platform that can support the DDS computations required to generate experimental GPS waveforms with a considerable amplitude and phase resolution. Although the platform is only currently required to support the synthesis of one GPS signal, it will ultimately need to support the synthesis of all GPS signals. Therefore, a cost-effective, re-programmable platform is ideal. In this section, FPGAs are presented as platforms ideally suited for implementing the software-defined GWPP

architecture. This section provides an introduction to FPGAs and describes the Xilinx Virtex-6 series device as a representative architecture. In addition, it highlights the typical FPGA design flow used to implement this architecture.

2.4.1 Introduction to Field Programmable Gate Arrays.

FPGAs are the newest addition to the family of Programmable Logic Devices (PLDs). PLDs are Integrated Circuits (ICs) with an extensive number of logic gates and flip flops that can be configured with basic software to perform both application-specific or unspecific logic functions for both simple and complex circuits. PLDs are popular among hardware designers since they offer higher reliability and design flexibility at a reduced level of complexity. In the family of PLDs, the most widely utilized for hardware designs are Application Specific Integrated Circuits (ASICs) and FPGAs. ASICs are designed specifically for one purpose; hence, the name application-specific. After fabrication, the logic functions embedded into the ASICs are unchangeable. Thus, to change the logic requires the design and fabrication of a new ASIC. Unlike ASICs, FPGAs comprise grids of logic gates that are *re-programmable* to enable *designers* to program customized digital logic; hence, the name field programmable. The process of changing the logic stored on the FPGA is simply a matter of reprogramming the FPGA with a new bitstream. The bitstream, after compilation, contains the configuration data to be loaded into the FPGA to perform a specific task. When choosing between ASICs and FPGAs, designers typically consider several factors such as Non-Recurring Engineering (NRE) costs, performance, and volume, to name a few. NRE costs are used to monetarily quantify the design process from specification to design to development. Previously, ASICs were preferred due to their high volume and performance compared to FPGAs. However, today, FPGAs, in terms of density and speed, essentially push the same performance

barriers as ASICs. With this in consideration, designers turn to NRE costs as the final deciding factor. NRE costs are higher for ASICs due to their architecture. In ASIC design flow, the designer must perform additional steps such as complex and time-consuming floorplanning, place and route, and timing analysis to ensure sufficient verification before fabrication; thereby, increasing the time and complexity of the ASIC design cycle. Whereas, in FPGA design flow, these same steps are eliminated since the targeted FPGA device is already verified in FPGA design tools. As a result, the design logic is synthesized accordingly, which simplifies the FPGA design cycle. In summary, FPGAs approach the performance of ASICs while offering lower NRE costs and infinite re-programmability. For this reason, it is the ideal platform for implementing current and future design requirements for the GWPP architecture. In addition, FPGAs offer simple interfacing to external devices such as DACs and other test equipment. These are the reasons why FPGAs were chosen as the focus of this research effort.

During this research thrust, an FPGA-based Commercial Off-The-Shelf (COTS) design platform known as the Xilinx Virtex[®]-6 ML605 board was used to implement digital designs [10]. The ML605 features the Virtex[®]-6 XCVLX240-T-1FFG1156 FPGA. Virtex-6 FPGA features three sub-families: LXT, SXT, and HXT [11]. The ML605 is a member of the LXT family, which are designed with high-performance logic and advanced serial connectivity [11]. With the utilization of any FPGA, it is important to understand what resources are available in order to ensure that the design fits on the board. Although there are newer, faster, and larger Xilinx FPGAs currently on the market, this FPGA was chosen as an initial implementation candidate to demonstrate proof of concept. Therefore, this research effort not only exemplifies proof of concept for this current FPGA, but also suggests that the design can also be implemented on newer FPGAs for increased performance. The following section

presents a brief introduction to the architecture of generic Xilinx FPGAs as well as the Virtex[®]-6.

2.4.2 Architecture of Xilinx Field Programmable Gate Arrays.

FPGAs contain an array of programmable logic blocks called Configurable Logic Blocks (CLBs). The architecture of CLBs is shown in Figure 9. CLBs are the primary logic resources for implementing complex combinatorial circuits or simple logic gates like AND, XOR, etcetera. In addition, CLBs can also be reconfigured to implement sequential circuits. Each CLB element connects to a switch matrix which connects the inputs and output of the CLB to the general routing matrix or to each other. The switch matrix is essentially responsible for connecting the CLB to the rest of the FPGA. A CLB unit comprises a pair of slices as shown in Figure 10. As a sub-unit within the CLB, slices contain resources for implementing sequential and combinatorial logic circuits. Every slice contains four LUTs, eight storage elements (FFs), carry-logic, and multi-function multiplexers. These elements provide logic, arithmetic, and ROM function capabilities. Some Xilinx FPGAs further categorize these slices into three types: SLICEX, SLICEM, and SLICEL. However, Virtex-6 CLBs only contain SLICEL and SLICEM [13]. SLICELs are designed to implement combinatorial functions, hence the suffix *L* can be thought of as *logic*. SLICEMs, which are 25%-50% of all slices, support two additional functions such as storing data using distributed Random Access Memory (RAM) or shifting data using one 32-bit register or two 16-bit registers; hence, the suffix *M* can be thought of as *memory*. The Virtex[®]-6 FPGA used in this research has total of 37,680 slices. Out of these slices, 23,080 are SLICELs and 14,600 SLICEMs. These quantities were taken from the Virtex[®]-6 CLB User Guide, [13], which presents an excellent overview of Virtex[®]-6 CLBs. In the Virtex[®]-6, the LUTs can be reconfigured as either one 6-input LUT or

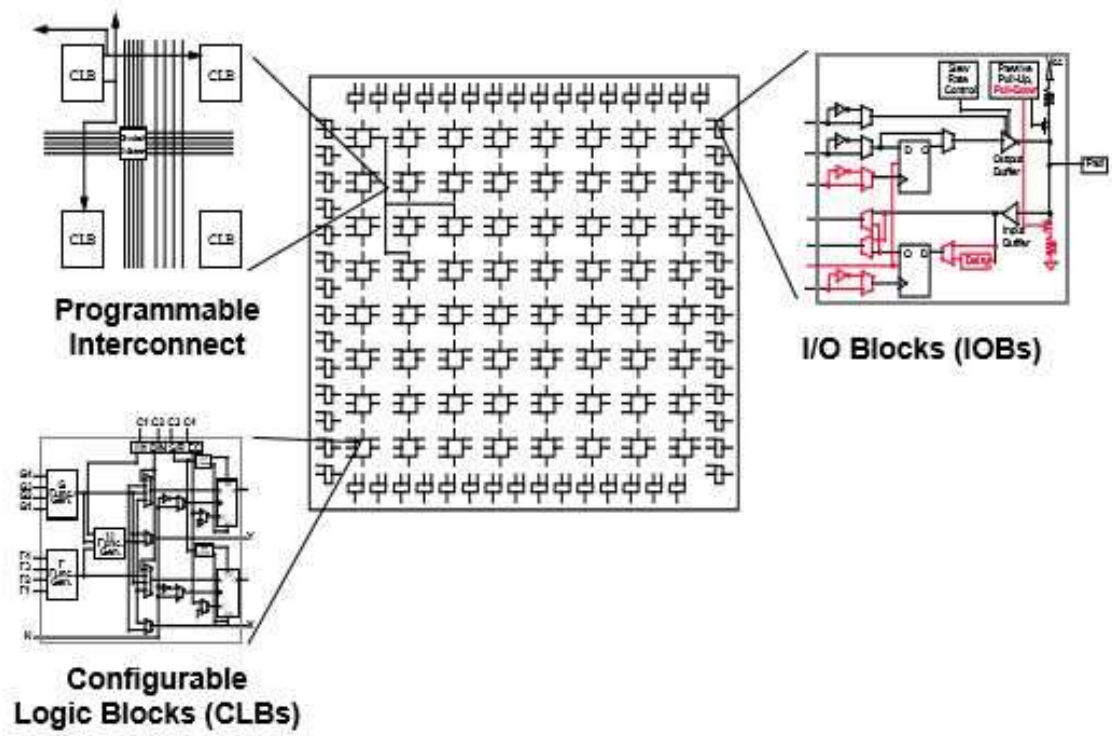


Figure 9. Xilinx Configurable Logic Block Architecture [12]

as two 5-input LUTs. The Virtex[®]-6 ML605 has a total of 150,720 6-input LUTs. The importance of available CLB resources come into play when designing the sine LUT. Since the Sine LUT is implemented in distributed ROM, the size of the LUT will greatly depend on the number of slices available. As shown previously, the Virtex[®]-6 FPGA used for this research has a finite number of slices available to the designer. With that in mind, available FPGA resources is a crucial consideration in the design and implementation of this research. The limited resources on the FPGA presents a trade-off between signal generation quality and FPGA resource utilization.

Along the periphery of the device are Input/Output Blocks (IOBs) which control the input and output to the FPGA. IOBs essentially provide the FPGA with connectivity to external devices, whereas CLBs implement the logic functions of the digital design. Inside FPGAs are additional resources such as Block RAM (BRAM)

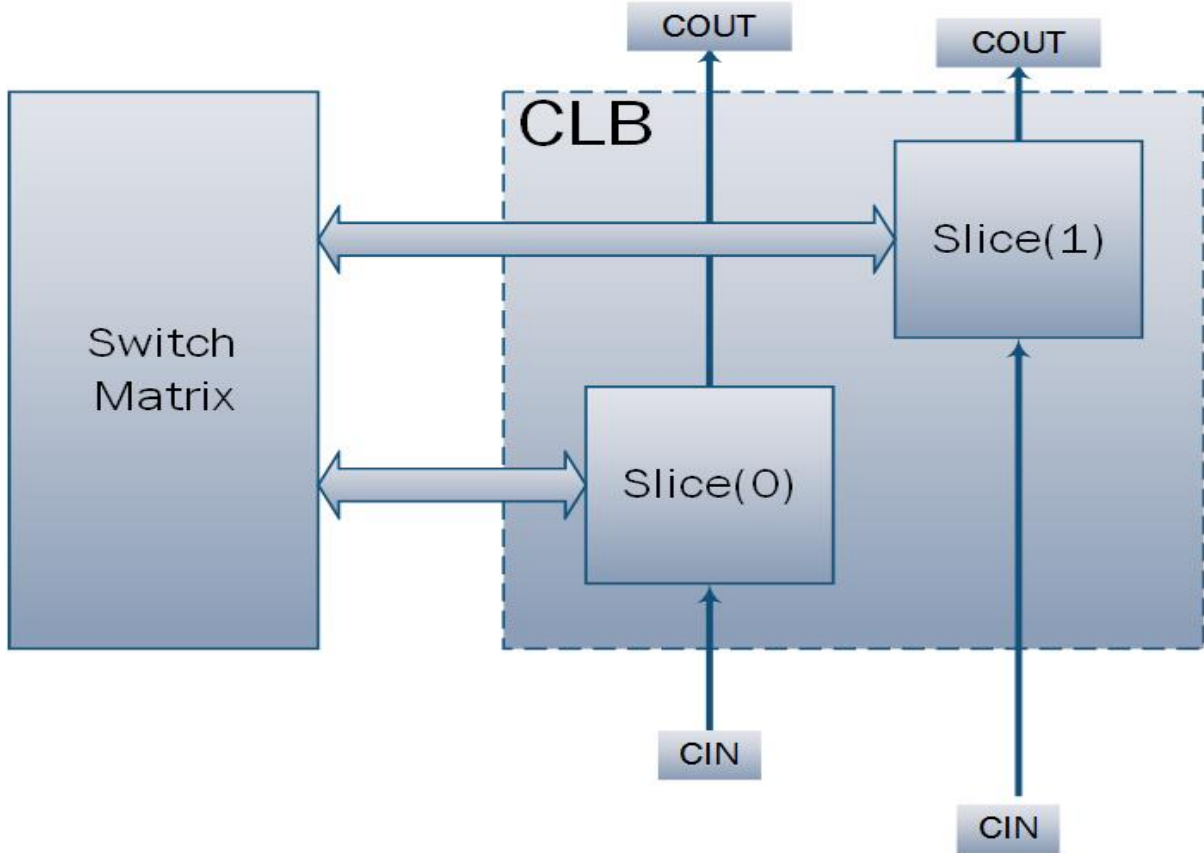


Figure 10. Arrangement of Slices within the CLB

and DSP48 slices, which provide additional functionality for digital design implementations. However, these resources are utilized for specific tasks and are not essential to understanding the implementation of this research in hardware.

In digital design using FPGAs, the design is implemented by mapping the design into the framework of the FPGA infrastructure. To ensure that the design fits in the available resources requires thorough understanding of the FPGA architecture. This process is performed by FPGA design tools, which the tools available depend on the manufacturer of the FPGA as well as the type of FPGA. For this research, Xilinx ISE[®] design suite is used with Xilinx Synthesis Technology (XST). Regardless of what tools are used, every tool generally translates the design through a series of stages. In Xilinx ISE[®] design suite, these stages are synthesize, translate, map, place

and route, and generate programming file. A brief summary of each stage is presented below

Synthesize: XST essentially performs both syntax checking and synthesis of the design. In this step, compilation is performed after syntax checking to translate and optimize the Hardware Description Language (HDL) code into a set of components that the synthesis tool can recognize. Logic is typically designed at a level of abstraction referred to as Register-Transfer Level (RTL). HDL is the language used to model the RTL. As the final step in this stage, the tool maps the components from after compilation into the primitive components of the targeted device. In this case, it would be mapped into the Virtex-6 primitive components.

Translate: In this stage, the input design is converted into a netlist that describes the logic in the design as well as any location and timing constraints. Timing constraints allow the designer to describe the timing performance requirements of the design. By providing a concise set of constraints not only increases the performance of the ISE software, but also allows the software to create a design that meets requirements specified by the design.

Map: In this stage, the design is mapped into CLBs and IOBs. Essentially, Map allocates CLB and IOB resources for all basic logic elements in the design. Additionally, it processes all location and timing constraints, performs target device optimizations. This results in the generation of another netlist, however, this netlist describes the mapping to the resources. Finally, a design rule check is performed on the netlist.

Place and Route: In this stage, Place and Route performs a Timing-Driven algorithm to analyze the timing constraints specified in the User Constraint

File (UCF). The UCF, as the name implies, is created by the user and specifies the constraints on the logical design, which ultimately affect how the logical design element is implemented in the target device [14]. In this research, a UCF was used to define timing constraints as well as pin assignments. In this case, the timing constraints specified in the UCF are processed with the input netlist.

Generate Programming File: In this stage, the configuration data is created. The configuration data is stored in a bitstream, which is created to be downloaded to a target device. After the bitstream is downloaded to the device, the FPGA is ready to perform the logic functions synthesized by XST.

Although the majority of the challenges in FPGA design are handled by the synthesizer, there are also additional challenges presented to digital designers due to the generic nature of FPGA architecture. As a result, different mapping translations can occur that do not always yield optimal implementation performance. Consider the following case where the design logic is not efficiently structured. Consequently, a situation presents itself where the designer thinks the logic performs a specific function and the synthesizer translates it to a different function. Therefore, the design in the FPGA does not perform its intended function. This is a challenging task in FPGA design in the sense that the designer must efficiently design the logic to allow the synthesizer to correctly map the logic to the correct resources.

2.4.3 Design Flow.

With any digital design process, there is a set of procedures the designers follow to yield optimum results. This process is commonly referred to as the design flow. It is important to note that different PLDs such as ASICs and FPGAs have their own unique design flow. Since this research focuses on a FPGA implementation, a FPGA-

based design flow is utilized. The typical design flow for FPGA-based digital design is illustrated in Figure 11. The input to the design process is a design specification outlining the design requirements. The requirements describe what design elements to include in the design as well as the overall structure of the design. In addition, constraints such as costs in resources and power consumption are generally included. As with this research, some design specifications are divided into individual phases to be designed, verified, and implemented sequentially. The next step of the design flow is to develop a model that simulates the behavior of the design. This is typically performed at a high-level of abstraction to verify the algorithmic approach used for the design. At the end of this stage, the designer has confidence that the behavior of the design satisfies the design requirements. Typically, this process is performed using both floating- and fixed-point operations. It is crucial to model the behavior first using floating point representation to verify the algorithm works. Floating point representation models 64-bit accuracy, which is not the case for FPGAs that rely on fixed-point operations. Therefore, it is important to verify the algorithm using fixed-point representation to design, implemented in fixed-point representation used in FPGAs, accurately satisfies the design requirements. At the end of this step, the designer is confident that the design algorithmically works on the FPGA. Once the behavioral model is fully validated to meet the constraints outlined in the design specification, the designer implements the behavioral model in RTL. RTL is a level of abstraction in digital design that describes the system in terms of its dataflow and timing between functional units such as registers, arithmetic-logic units, etcetera. It is essentially used in HDLs, such as Verilog and VHSIC Hardware Description Language (VHDL), to create high-level representations of a logic circuit which separate datapath and control. In this implementation, RTL is modeled using VHDL. It is a common design practice to break down larger design components into individual

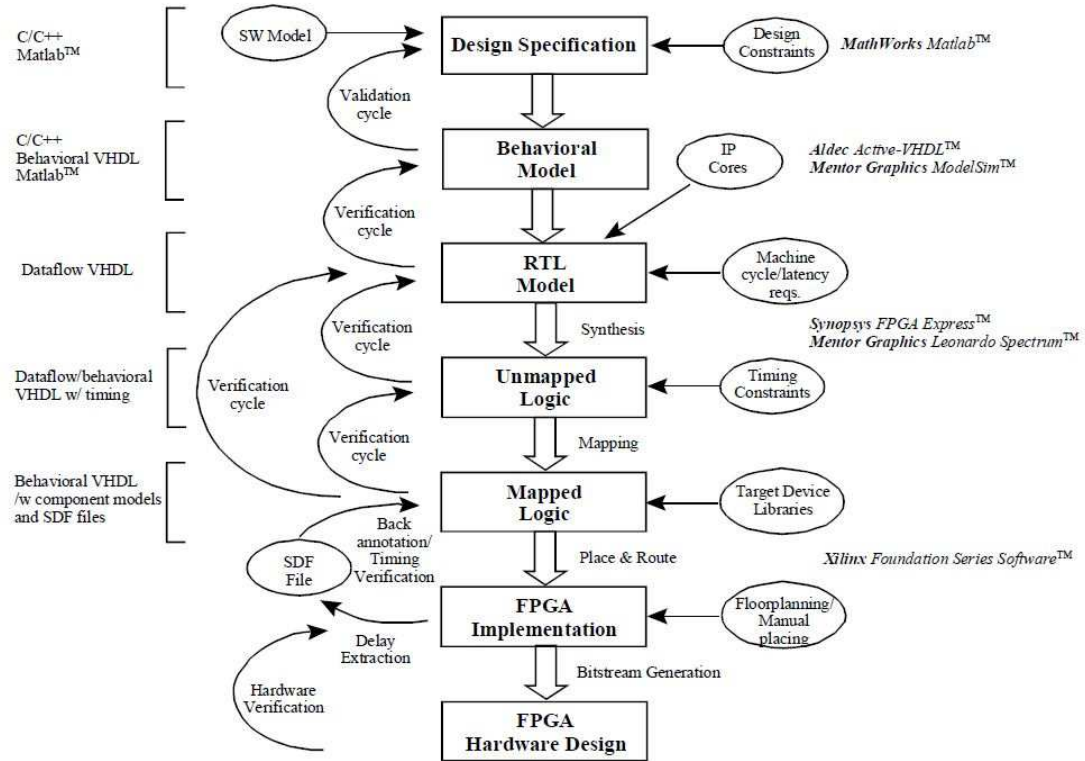


Figure 11. Typical FPGA Design Flow. Permission from [15]

subcomponents to first be designed and verified individually then combined to form the larger design component. This approach represents a bottom-up design technique where the designer creates basic functional units with very simple tasks. Once the functional units are fully verified, a more complex model is constructed using the combination of the simpler, basic functional units. In this process, the designer can also include pre-designed Intellectual Property (IP) cores that can be integrated into the design. IP cores simplify the design process by offering already verified and optimized functional blocks. However, the disadvantage is that these blocks are cores, which provide the designer with limited flexibility. After RTL implementation, the RTL models are verified with respect to the behavior model. This method is performed by creating and implementing testbenches. A testbench is designed to stress the Design Under Test (DUT), which is the RTL design component, with various inputs to test

functionality. This process allows the designer to perform functional verification of the design.

Following the verification process, the RTL model is synthesized into hardware elements such as memory elements and logic gates. This process is performed by the synthesis tool. It is important to note that at this stage the design elements are not necessarily mapped to actual hardware in the target device. As mentioned previously, the synthesis tool does not always map the logic to yield correct functionality. As a result, it is imperative to also verify the gate-level design from the synthesizer against the RTL model to reveal any errors during synthesis. After this process, the synthesizer tool handles the remaining stages of the design flow by means of its synthesis stages (map, place and route, and generate programming file).

In this research thrust, MATLAB[®] was the software chosen to perform behavioral validation of the design. However, it was only used to verify the high-level behavior of a DDS system. In addition, it is heavily used in this research to exploit the software-defined architecture of this research. As mentioned previously, the RTL models were implemented in VHDL and functionally verified using two different tools: ModelSim[®] provided by Mentor Graphics and Xilinx ChipScope[™]. More information on design and verification of the RTL models are presented in Chapter III. That said, the design logic was synthesized, translated, and mapped using Xilinx ISE[®] design suite. The targeted device was the Virtex[®]-6 XCVLX240-T-1FFG1156 FPGA. This FPGA is embedded into the ML605 evaluation board. The board features for the ML605, which is the platform used in this research, is presented next.

2.4.4 Virtex-6 FPGA ML605 Development Board Features.

As mentioned previously, this research work uses the Xilinx based Virtex[®]-6 FPGA ML605 development board for implementing digital designs. The board fea-

tures are illustrated in Figure 12. In Figure 12, it is worthwhile to introduce some of the features utilized on this board to support communication between the FPGA and external elements. Recall that this thesis proposes a FPGA-based GPS waveform prototyping platform. The design of this platform at a high level of abstraction features a software-defined all-digital signal synthesis architecture in conjunction with a high-speed DAC. This work explores using this architecture to synthesize the GPS L1 C/A signal. The actual hardware implementation of this research architecture is presented next in Chapter III; however, it is worthwhile to describe the ML605 features that provide the capability to implement this architecture. The first feature in this discussion is the Universal Serial Bus (USB) to Universal Asynchronous Receiver Transmitter (UART) component. Software-defined capability, in this research, is defined as the capability to reconfigure the digital computation blocks used to implement the functionality of this research an infinite amount of times through software. The idea is to design the digital computation blocks to be simple in function, yet have a multitude of configuration capabilities that are all accessible through software. In order to exploit this capability, a communication channel between the host Personal Computer (PC) and FPGA is required. In this thesis, the communication channel is implemented using UART protocol. Therefore, the USB to UART port provides this capability. The next feature in this discussion is the High Pin Count (HPC) FPGA Mezzanine Card (FMC) connector. As mentioned previously, this design utilizes DDS for signal generation. The final component in a DDS system is the DAC, which converts the digital waveform to its analog representation. The performance of the DAC, in terms of the range of frequencies able to be generated, depends on the sampling rate of the DAC. Other FPGA-based DDS implementations described in literature use standard DACs with sampling rates no greater than 1 GHz. As a result, the max frequency reproducible is 500 MHz. In order to synthesize GPS center

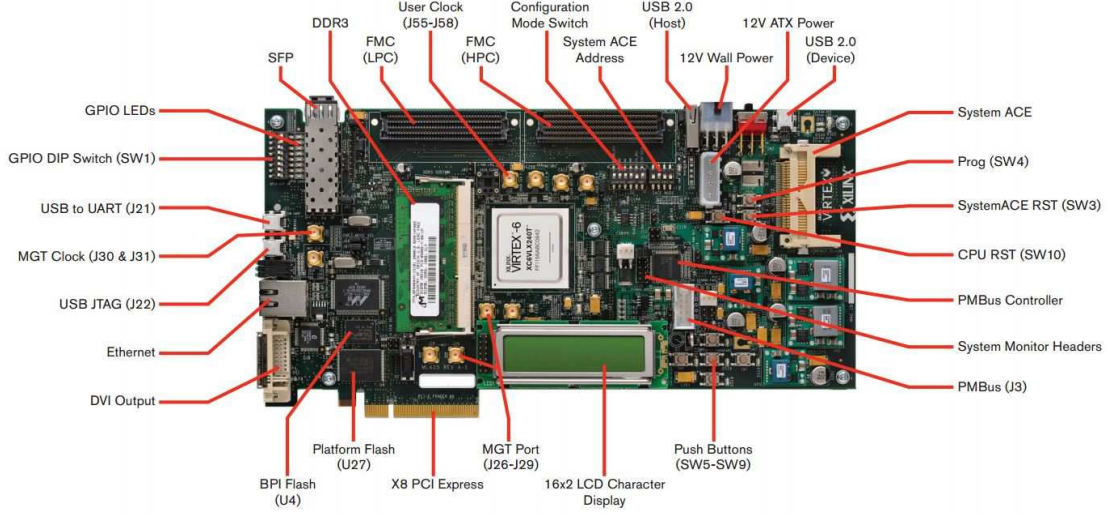


Figure 12. Virtex®-6 ML605 Board Features [16]

frequencies with this DAC would require additional resources such as PLLs or mixers. This research, on the other hand, utilizes a high-speed DAC with a sampling rate up to 5.7 Gigasamples Per Second (Gsp/s). Therefore, the DDS architecture built in this research is able to synthesize GPS center frequencies without the need of additional resources such as PLLs and mixers. The DDS architecture in conjunction with the high-speed DAC is what really separates this DDS-based research design from other DDS implementations presented in literature. For this reason, the GWPP architecture is able to generate RF carriers without additional circuitry such as analog mixers or PLL synthesizers. The feature on the FPGA that enables the connection between the FPGA and the high-speed DAC is the HPC FMC connector.

III. Methodology

THIS chapter covers the methodology used in this research effort to design and verify the GWPP architecture. The methodology is broken down into two main components: design implementation and experiment setup, which is illustrated in Figure 13. The software-defined DDS architecture built and tested in this design can best be explained and understood in progressive steps. The clearest explanation is to first start with a design overview (Section 3.1). In this section, a block diagram showing the design overview is presented. Accommodating the block diagram (also in Figure 13) is an overview describing the role of each block in generating an experimental GPS waveform. Following the design overview is the actual implementation of the architecture. The implementation is best understood by further dividing it into separate hardware and software implementations. The hardware implementation, which is covered in Section 3.2, describes the hardware component blocks that build the DDS architecture. Inside the discussion of these components is explicit detail on how the component supports the software-defined paradigm. Finally, this chapter concludes with the methodology involved in the experiment setup (Section 3.3). This section provides insight to what experiments were executed to calibrate lab equipment to perform verification.

3.1 Design Overview

The heart of this design features an all-direct digital signal synthesis architecture. Figure 13 graphically details the design overview and illustrates the design flow for this architecture. First, medium to low rate operations are performed within a workstation-class computer. The host PC and FPGA are bridged via UART interface, which is also shown in Figure 13. High rate computations such as DDS

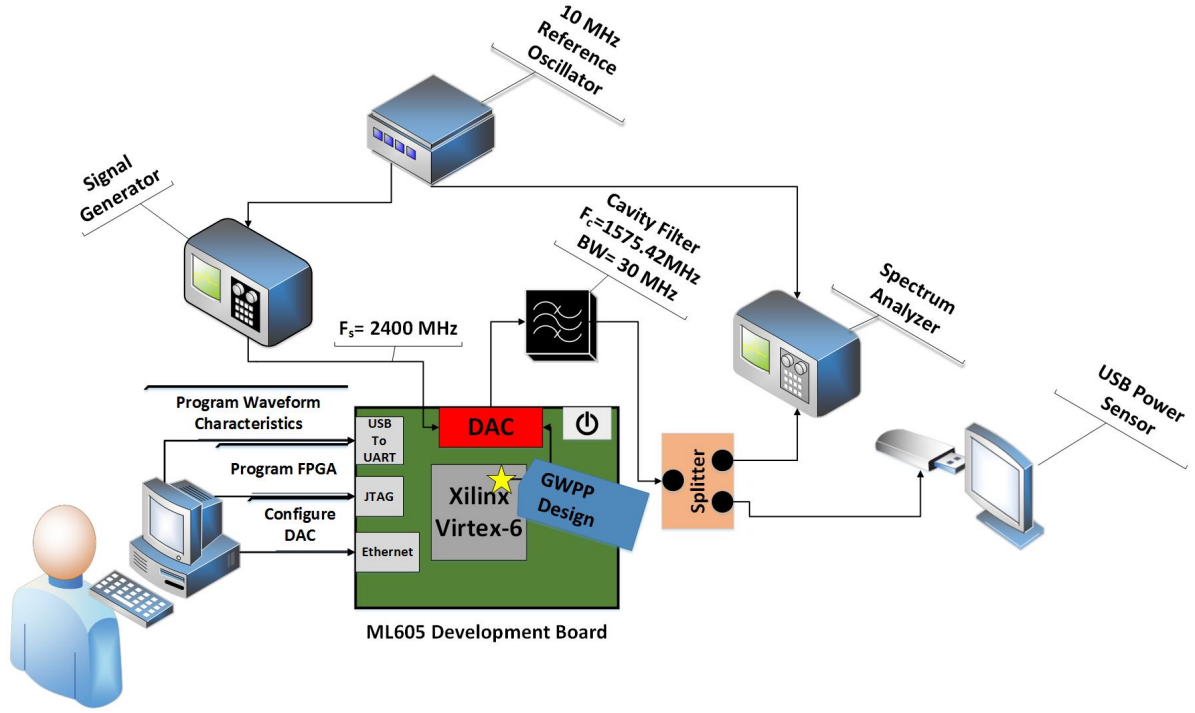


Figure 13. GWPP Design Methodology Overview

are performed within a FPGA development board. The FPGA development board used for this research is the Xilinx Virtex[®]-6 ML605. Updates from the host PC to the FPGA will occur at a fixed rate of 115200 baud using MATLAB[®] software. These updates consists of parameters pertaining to setting the carrier frequency and phase, code frequency and phase, and PRN sequence of the output waveform. The parameters, in turn, define the experimental GPS waveform characteristics. The experimental GPS waveform samples are generated using logic programmed in the FPGA and transmitted through the Serializer/Deserializer (SERDES) interface on the FPGA to the Low-Voltage Differential Signaling (LVDS) inputs of the high-speed DAC. The DAC, AD9129, is located on the FMC230 daughterboard, which also has a high pin count (HPC) FPGA Mezzanine Card (FMC) connector. The HPC FMC connector of the FMC230 daughterboard connects to the HPC FMC connector on the FPGA. The functional block diagram of the FMC230 daughterboard is shown in

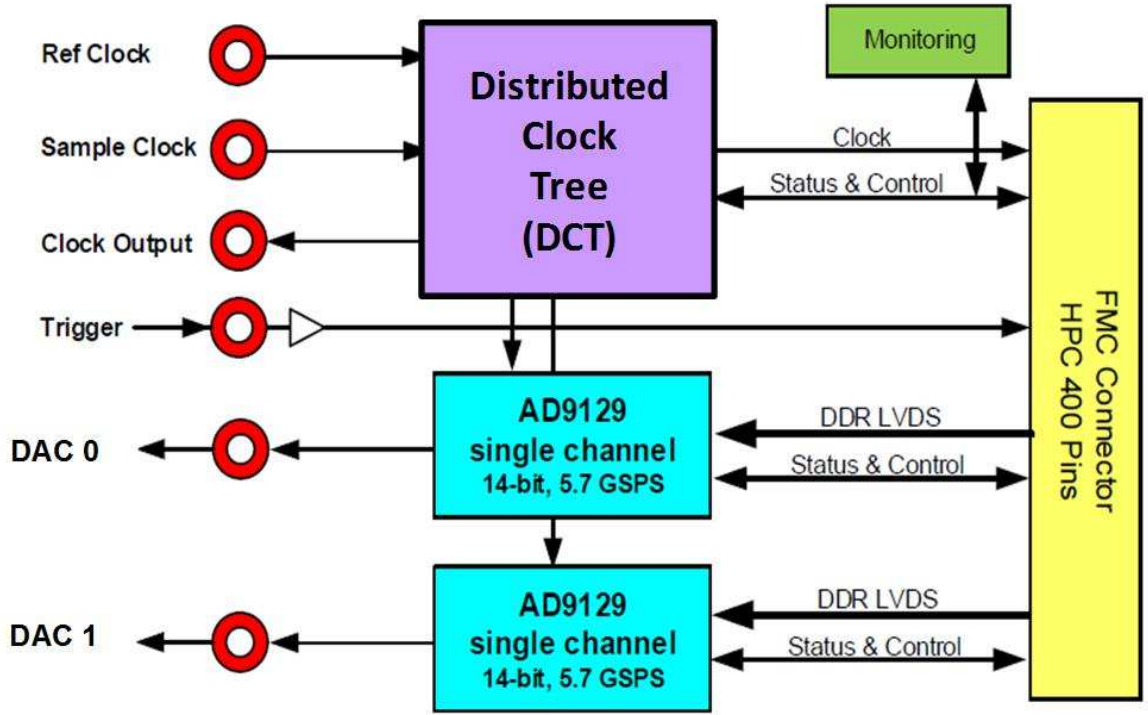


Figure 14. FMC230 Functional Block Diagram. HPC (yellow) connects the DAC with FPGA [17].

Figure 14. This connection allows communication between the FPGA and DAC. In addition, the FMC230 daughterboard has a clock tree chip referred to as Distributed Clock Tree (DCT). The DCT supplies both the DAC with a sampling clock and the FPGA with a synchronous clock, respectively. The DCT has various sources to serve as the sampling clock for the DAC such as an internal VCO and external sample clock input. For this research, the internal VCO is bypassed and the external clock input is routed through the sample clock input port of the DCT and used as the sampling clock to the DAC. The external sample clock input is a 2400 MHz sinusoidal waveform provided by the RF output of the Keysight N5182B MXG Vector Signal Generator. The signal generator is externally phase locked to a 10 MHz, low phase noise reference oscillator provided by a 4-Channel GNSS RF front-end (developed by Ohio University). Therefore, the sampling clock source for the DAC is essentially

the 2400 MHz RF output provided by the signal generator and routed through the DCT. Furthermore, the sampling rate of this clock along with special configurations in the DAC, which are explained later in this section, allows for the synthesis of experimental GPS waveforms in the L-band without additional analog circuitry such as PLLs and mixers. The FMC230 daughterboard is developed by 4DSP Limited Liability Company (LLC). The DAC on the FMC230 daughterboard has two 14-bit D/A channels that operate at 5.7 Gigasamples per second (Gsps) (2.85 Gsps direct RF synthesis). However, through experimentation, the sampling rate for this DAC does not exceed 2.85 GHz. At the output of the DAC, the synthesized digital waveforms are precisely converted to analog. The analog waveform is sent through a bandpass cavity (reconstruction) filter, which has a center frequency of 1575.42 MHz and a 30 MHz wide bandwidth. The purpose of the bandpass filter is to smooth the output of the DAC and eliminate any out of band signals coming from the DAC as a result of sampling and quantization. However, in the experiment setup to analyze the platform, the cavity filter is removed to prevent the filter characteristics from suppressing the theoretically-driven performance metrics created for this platform. Finally, the output of the filter is connected to test equipment. The test equipment used in this research effort consisted of a USB Power Sensor, Oscilloscope, and Spectrum Analyzer, which allowed the opportunity to view the average power of the waveform and the waveform representation in both the time domain and frequency domain, respectively. Although for analysis, only the Spectrum Analyzer and USB power sensor were utilized.

In addition, the design follows the software-defined paradigm where high-rate digital computation blocks are designed to be simple in function, yet have a multitude of configuration capabilities that are all accessible through software. As a result, the hardware design remains static while providing infinite programmability features

via software. The programmability feature include the ability to alter the waveform characteristics such as carrier/code frequency/phase as well as PRN sequence dynamically. This is crucial considering the correlation between design complexity and compile time. As complexity increases, the Xilinx FPGA design tools (ISE, in this thesis) requires more time to place and route the design. Therefore, if the user wanted to change the GPS waveform characteristics, without software-defined capability enabled, it would require the user to change the design and download the FPGA with the new bitstream. If multiple changes need to be made in one day, a lot of time will be wasted running the entire design process for an FPGA. Therefore, with software-defined capability, the GWPP architecture allows the designer to only program the FPGA and configure the DAC once, while providing infinite programming of the output waveform characteristics. This concept is reinforced in Figure 15.

In conclusion, the software-defined DDS architecture is composed of both hardware and software implementations. The hardware implementation, which is discussed next, describes the components that build the DDS architecture, whereas the software implementation encompasses the software-defined capability utilized to configure the hardware. However, it is important to note that the hardware design

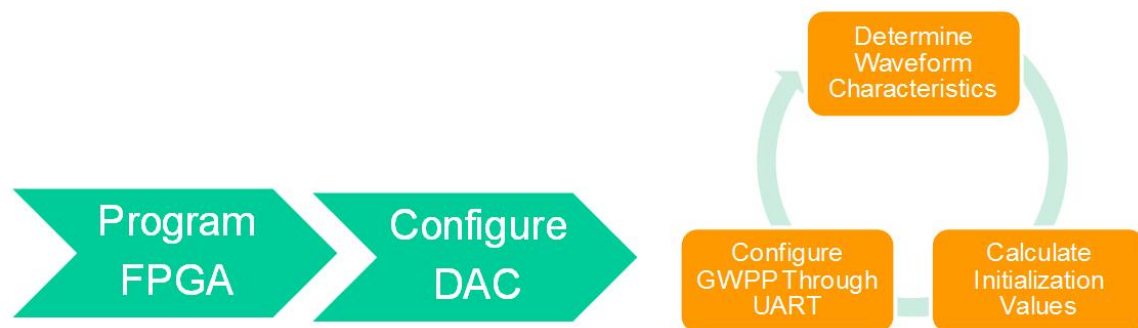


Figure 15. GWPP Architecture Configuration Cycle

components that ultimately build the DDS architecture must be designed to exploit the software-defined nature of this architecture.

3.2 Hardware Implementation

3.2.1 High-Level Overview.

Figures 13 and 15 illustrated the overall design flow for the GWPP architecture. The main abstraction shown in Figures 13 and 15 is that the GWPP architecture generates the samples dictated by the waveform characteristics and sends them to the DAC. However, not shown in Figures 13 and 15 are the internal details in which the samples are generated and sent to the DAC. Therefore, this section focuses on a high level overview illustrating these internal details. The discussion begins with a high level overview shown in Figure 16.

Figure 16 depicts the high level datapath for the GWPP architecture. The main abstraction in Figure 16 is that the GPS L1 C/A samples are generated in the *L1 C/A 16 Sample Generator* design block. Since the GWPP platform employs a high-speed DAC (2400 MHz sampling rate) to generate a “real-world” GPS L1 C/A signal, the DAC requires the FPGA application to transfer data at a specific rate and format, which will be discussed later in this section. However, for the purpose of this discussion, it is sufficient to only know that the SERDES component handles the data transfer from the FPGA to the DAC. It is also important to note that in this current discussion, *data* refers to the GPS L1 C/A samples. Therefore, at the highest level of abstraction, the SERDES component sends the samples to the DAC at the required data rate and format. In order to achieve the data rate requirements imposed by the DAC, the GWPP architecture in the FPGA must generate 16 samples every clock cycle; hence, the reason why the design element is called *L1 C/A 16 Sample Generator*. Before proceeding to the details of how the samples are generated, there are four

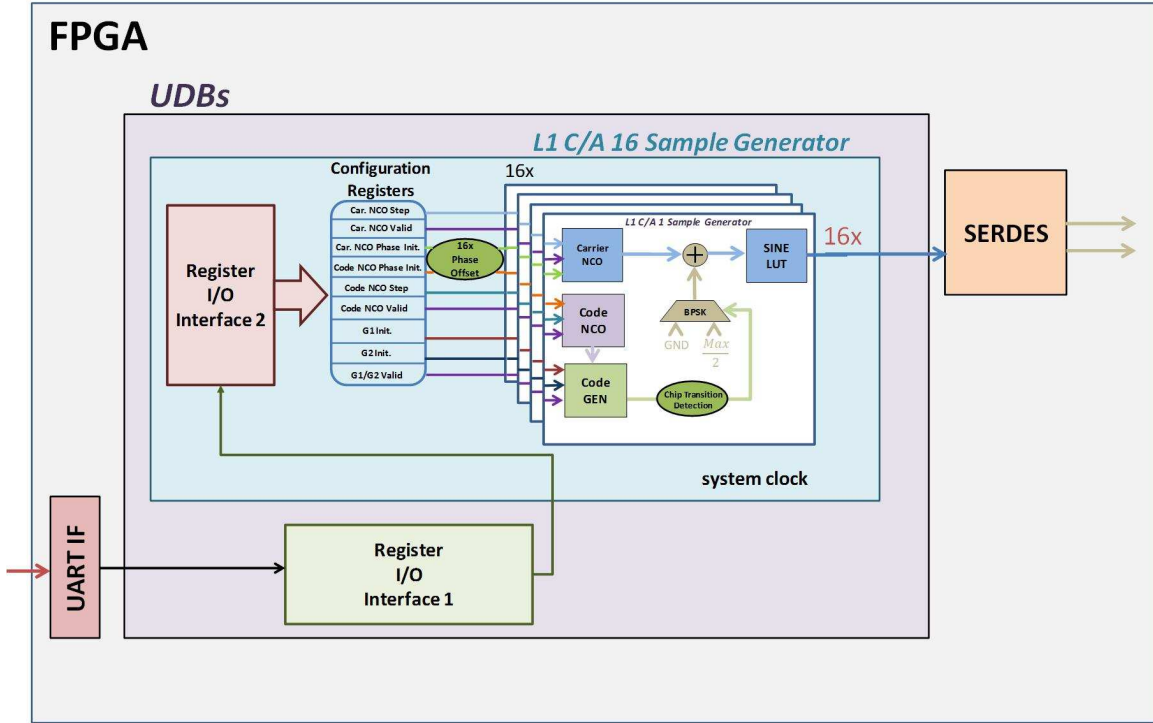


Figure 16. Hardware Implementation High Level Overview Functional Block Diagram

main concepts worth discussing.

The first of these is that all operations inside the L1 C/A 16 Sample Generator are clocked by the same clock source, namely the system clock. More details on the generation of the system clock are provided later in this section. The second factor is that characteristics of the output samples generated by the L1 C/A 16 Sample Generator depend on the contents stored in the configuration registers. In other words, the configuration registers hold the contents fed to each L1 C/A 1 Sample Generator design block which ultimately configures the carrier frequency/phase, code frequency/phase and PRN sequence of the output GPS L1 C/A signal. The third factor is that the combination of the Configuration Registers, Register I/O Interface 2, Register I/O Interface 1 logic allows for the ability to change the contents stored in the configuration registers thereby establishing the software-defined capability provided by the GWPP architecture. Essentially, the waveform configuration parameters are

sent from the host PC to the FPGA via the UART Interface (IF) in an address/data format and the combination of the logic described above routes the parameters to their respective configuration register based on the address; hence, the address/data format enables the ability to change the contents in the configuration registers, which correspond to the carrier frequency/phase, code frequency/phase, and PRN sequence, dynamically without the need to download a new bitstream to the FPGA.

Before presenting the last main concept regarding Figure 16, it is worth briefly describing the implementation of Register I/O Interfaces 1 and 2. First, Register I/O Interface 2 is essentially an address decoder used to route the incoming waveform configuration parameters to their respective configuration register. Second, Register I/O Interface 1 handles the handshaking between the UART IF and Register I/O Interface 2; it essentially receives the UART bytes and reassembles them into the original address/data format to be correctly routed by the Register I/O Interface 2 logic. More details on the implementation of Register I/O Interface 1 is presented in Appendix D and Appendix E. Finally, the last main concept is that the design elements such as the L1 C/A 16 Sample Generator (and its subcomponents) as well as the Register I/O Interface 1 are considered User-Defined Blocks (UDBs) since these design elements are not provided by any board support package and are therefore developed in RTL to build the software-defined DDS architecture of the GWPP platform. With consideration in the UDBs, it is worth noting that the L1 C/A 16 Sample Generator design element is the heart of the GWPP architecture. Furthermore, the L1 C/A 1 Sample Generator design element is the heart of the L1 C/A 16 Sample Generator design element. For this reason, the main objective in the hardware implementation portion of this chapter strictly focuses on the implementation of the L1 C/A 1 Sample Generator design element. However, before proceeding on to the details of how this design element is implemented, it is first important to understand the clocking scheme with

regards to the GWPP architecture. This concept is crucial to understanding the data rate and format requirements imposed by the DAC and nonetheless provides clarity in the future discussions regarding the design decisions made in the implementation of the L1 C/A 1 Sample Generator design element. Additionally, understanding the clocking scheme provides insight to how the GWPP architecture communicates with the DAC to generate the GPS L1 C/A signal by means of direct RF synthesis.

3.2.2 GPS Waveform Prototyping Platform Clocking Scheme.

As mentioned previously, the concepts regarding the GWPP clocking scheme is crucial to understanding the data rate and format requirements imposed by the DAC. In other words, it provides insight to how many samples and at what rate the GWPP architecture must transfer the samples to the DAC to maintain optimum Alternating Current (AC) performance at the output of the DAC. The GWPP clocking scheme is represented in the functional block diagram in Figure 17. The main abstraction in Figure 17 is that the Phase-Locked Oscillator (PLO) serves as the master clock to the subsequent circuitry. The PLO used in this research is the Keysight N5182B MXG Vector Signal Generator externally phase locked to a low phase noise 10 MHz reference oscillator provided by a 4-Channel GNSS TRIGR front-end. The experiments performed to determine the signal generator as the sampling clock as well as the TRIGR front-end as the reference oscillator are further discussed later in this chapter. The output clock (2400 MHz RF output) provided by the PLO is sent to the DCT where it is then routed to the DAC core (as its sampling clock) as well as to a clock distribution block in the DAC (not shown in Figure 17). The clock distribution block divides the sampling clock by four and sends it to the Mixed-Mode Clock Manager (MMCM) primitive on the FPGA. Essentially, the divided clock is an external clock output from the DAC to allow for synchronization between the DAC and FPGA.

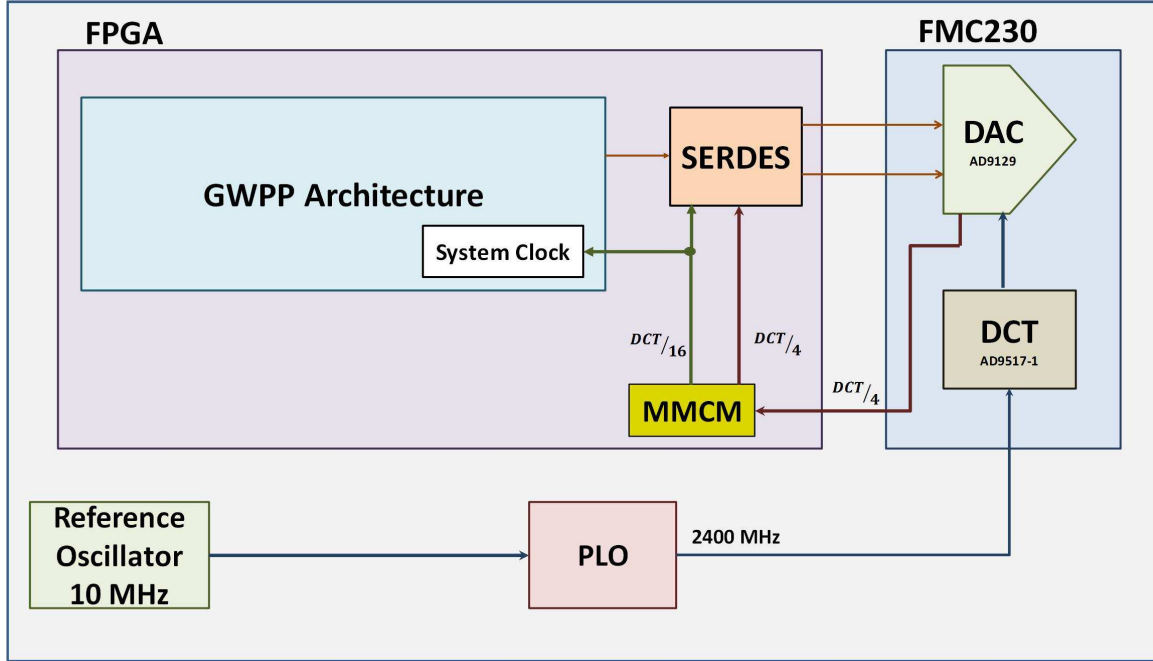


Figure 17. GWPP Clocking Scheme. Essentially, the clocks sent to the architecture and Digital to Analog Converter are derived from very clean Phase-Locked Oscillators.

Furthermore, the MMCM primitive utilizes this clock to generate all critical internal and external clocks for the FPGA. This includes the system clock in the GWPP architecture in addition to the two different clocks required by the SERDES primitive. The clock source quality, as defined by its phase noise characteristics and jitter, affects the clocking of the GWPP architecture as well its spectral performance. If the GWPP system clock or DAC sampling clock suffers from jitter and phase noise, then both components will be clocked at non-uniform intervals. The result is a spreading of the spectral line (spurs) at the output frequency of the waveform synthesized by the DAC. The degree of spreading is proportional to the amount of jitter and phase noise present; hence, the clock source quality of the sampling/system clock is another important consideration in maintaining optimum AC performance. In addition, the spectral performance of the GWPP platform is an important factor when considering its integration with GPS receivers that have sensitive tracking loops. Nonetheless, since the GWPP architecture operates with a clock derived from a PLO, the quality

of the sampling clock as well as the system clock mirrors its performance. Therefore, utilizing a PLO with low phase noise and jitter guarantees a sampling clock and system clock with the same performance. This reduces the spurious emissions created by the platform due to the sampling/system clock performance.

It has been established that the clocking scheme is vital to the quality of the signal generation process. In addition, the quality is dictated by the phase noise and jitter performance of the PLO. Next, it is worth establishing one of the major concepts that is vital to understanding many of the design decisions regarding the GWPP architecture. This concept essentially explains why the GWPP architecture must generate 16 samples every clock cycle to achieve the data rate requirements imposed by the DAC. In addition, it explains how the system clock used in the GWPP architecture is derived. The clearest explanation sequence is to first start with the discussion of the DCT followed by the discussion of the DAC used in this research. Afterwards, this section will conclude with the discussion of the MMCM and SERDES primitives.

3.2.2.1 Distributed Clock Tree Overview.

The DCT is essentially a multi-output clock distribution component. It generates the sampling clock for both DACs in addition to other clocks used for test and monitoring purposes. More information regarding theory of operation for this DCT component can be found in reference [18]. In addition, the configuration of the DCT is explained in further detail in Appendix B. However, for the sake of this discussion, only consider one key concept regarding the DCT.

First, the DCT is one of the most vital components in the GWPP architecture. It essentially generates the sampling clock for the DAC core to perform the digital to analog conversion. Therefore, the quality of the output signals generated by the

GWPP architecture mirrors the quality of the sampling clock provided by the DCT. The DCT has two options for choosing which sampling clock to provide to the DAC. The first option is the on-chip internal VCO in configuration with a PLL. The second option is an external sample clock input. To utilize the second option, the PLL is bypassed and the external sample clock input is fed to the DAC. Since the quality of the sampling clock provided by the DCT is an important consideration in both maintaining AC performance in the DAC as well as optimal performance in the GWPP architecture, it is imperative to choose the best sampling clock option. In this context, *best* refers to the cleanest sampling clock with the lowest phase noise performance. Furthermore, the phase noise performance is dependent upon the process in which the sampling clock is generated as well as the structure of the sampling clock (i.e., oscillator type). If the sampling clock is surrounded by various noisy components, this digital noise ultimately contributes to the noise of the sampling clock source. In addition, if the oscillator is old or endures shock or mechanical vibrations then this causes the oscillator to drift. In turn, the oscillator drift is detrimental to the performance of the sampling clock. Hence, for the GWPP architecture, it is imperative to choose the cleanest sampling clock. As mentioned previously, the sampling clock chosen for this architecture is the 2400 MHz RF output provided by the Keysight N5182B MXG Vector Signal Generator. The experiments performed to reinforce this choice are presented later in this chapter.

3.2.2.2 Digital to Analog Converter Overview.

A DAC is a device that converts digital codes into an analog signal. The digital code consists of digital samples where the digital samples are finite-precision numbers (usually a fixed-point binary number), whereas the analog is a continually varying physical signal. The DAC used in this research is the AD9129 enclosed on the FMC230

Printed Circuit Board (PCB) (refer to Figure 14). It is important to note that the FMC230 contains two DACs, but for this research effort, only *DAC 0* was used to demonstrate proof of concept. There are many categories of DACs. However, this research focuses on a 2x interpolating high-speed DAC configurable in two different modes namely, normal or mix-mode. Mix-mode operation is used to reconstruct RF carriers in the second Nyquist zone. More information regarding theory of operation for this DAC can be found in reference [19]. In addition, the configuration of the DAC used in this research is explained in further detail in Appendix A. However, for the sake of this discussion, only consider several concepts regarding the DAC used in this research.

First, the DAC accepts 14-bit data on two LVDS inputs in a Double Data Rate (DDR) format. DDR simply means that data will be transferred on both the falling and rising edges of the clock. LVDS simply means that the port transmits or receives information as the difference between the voltages on a pair of wires. DDR in conjunction with LVDS creates a high data rate, simplified interface between the FPGA and DAC. From the perspective of the DAC, it expects the FPGA to send both de-interleaved data to the LVDS ports in a DDR format. Therefore, this reinforces the fact that the samples being generated in the GWPP architecture must occupy 14 bits. In addition, the DDR format imposes data rate requirements the GWPP architecture must comply with in order to exploit the high-speed sampling rate of the DAC. If the data rate is not met, the waveform samples generated by the GWPP architecture will not be sampled correctly. As a result, the analog waveform after the digital to analog conversion will be distorted.

Second, the sampling clock fed to the DAC is provided by the DCT. It was established that the clock source quality, as defined by its phase noise characteristics, jitter, and drive capability, is an important consideration in maintaining optimum

DAC performance. However, the frequency of the sampling clock is an important consideration in performing direct RF synthesis of analog waveforms. It is worth noting that for this discussion the frequency of the sampling clock will be referred to as the update rate of the DAC. As mentioned previously, the update rate for the DAC used in this research is 2400 MHz. Recall that the sampling theorem indicates that a continuous signal can be properly sampled only if it does not contain frequency components above one-half of the sampling rate. This theorem can be applied to the DAC in the sense that the DAC can only synthesize frequency components less than one-half of the DAC update rate. Therefore, with the current update rate of F_s equals 2400 MHz, the DAC can directly synthesize frequency components up to $\frac{F_s}{2}$, or 1200 MHz, without aliasing. This process refers to synthesizing frequency components in the first Nyquist zone, DC to $\frac{F_s}{2}$. The GPS L1 C/A center frequency of 1575.52 MHz is outside the first Nyquist zone and therefore resides in the second Nyquist zone, $\frac{F_s}{2}$ to F_s . To get around this problem, the DAC used in this research offers an additional operating mode called Mix-ModeTM that enables the ability to reconstruct carriers in the second Nyquist zone. This is accomplished by generating a carrier with an intermediary frequency in the first Nyquist in which its corresponding frequency is equivalent to the GPS L1 C/A center frequency. Therefore, using Mix-Mode operation allows for the ability to use the GPS L1 C/A signal, which resides in the second Nyquist zone for the DAC update rate used in this research, as the primary output signal of the GWPP platform.

3.2.2.3 Serializer/Deserializer Overview.

SERDES is a pair of functional blocks commonly used in high speed communication to compensate for limited Input/Output (IO). These functional blocks are found in the IOB area of the FPGA fabric. Each IOB contains an input deserial-

izer (ISERDES) and output serializer (OSERDES) block. ISERDES allows serial-to-parallel conversion, whereas OSERDES allows parallel-to-serial conversion. Recall that the DAC requires parallel data. More specifically, the DAC LVDS data ports require interleaved data from the connecting interface on the FPGA. This requires that the FPGA interface provides two data buses of DAC resolution. Since a 14-bit DAC is used in this architecture, it effectively requires the FPGA to provide two 14-bit differential buses. Each bit of the bus is represented by one OSERDES component. Without a SERDES primitive, the FPGA would need to dedicate 28 pins to transmit data to the DAC. In addition, these pins must support high data rate transmission due to the sampling rate of the DAC. Considering FPGAs have limited IO pins capable of such data transmission rates, this would essentially increase the design costs. However, the SERDES primitive simplifies the design of a deserializer circuit, while allowing higher data transfer speeds. Therefore, the OSERDES primitive is utilized to transmit data from the FPGA interface to the DAC while minimizing the number of IO pins and interconnects. The OSERDES bit arrange for a DAC input is presented at a high level of abstraction in Figure 18. Essentially, the SERDES primitive supplies serial streams of parallel data in DDR format to the LVDS ports on the DAC interface. Recall that the DAC has two LVDS input ports. Furthermore, the parallel data sent to the DAC are the 14-bit samples generated by the GWPP architecture. The rate at which data is sent depends on the two clock inputs to the SERDES. Typically, there is a clock to drive the serial side of the SERDES component, namely CLK, and another clock to drive the parallel side, namely CLKDIV. In addition, the format in which data is transmitted depends on the configuration of the SERDES. Typically, there are two formats: Single Data Rate (SDR) and DDR. In this research, the SERDES primitive is configured for 8-bit DDR transmission. More information regarding the configuration of the SERDES primitive is provided

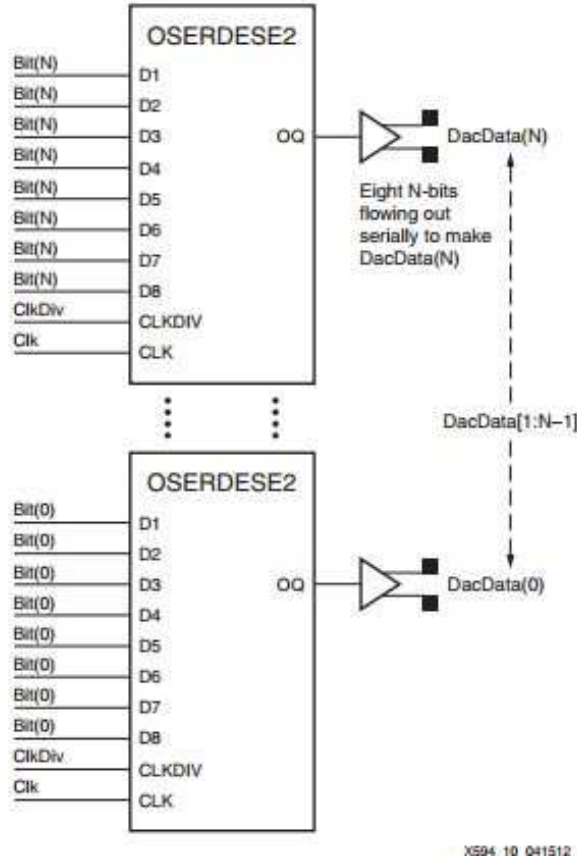


Figure 18. OSERDESE Bit Arrangement for a DAC Input Bus[20]

in Appendix C. However, for this present discussion, it is sufficient to only know that the SERDES transmits the samples in DDR format to both 14-bit input channels of the DAC. Since it is configured for 8-bit DDR, the 14-bit DAC essentially requires the FPGA to transmit $14 \times 8\text{-bit} = 112$ bits every clock cycle. Since the DAC used in this research operates with two channels, this amounts to 224 bits every clock cycle. Therefore, to exploit the high-speed sampling rate of the DAC to perform direct RF synthesis, the FPGA must transmit 224 bits worth of samples every clock cycle to the DAC. With a 14-bit DAC, this is accomplished in the GWPP architecture by generating 16 samples every clock cycle and sending them to the SERDES.

3.2.2.4 Mixed-Mode Clock Manager Overview.

The MMCM primitive is a clocking circuit that is customizable to the user's clocking requirements. It is essentially utilized to generate multiple clocks with defined phase and frequency relationships to a given input clock [21]. It has been established that the input clock to the MMCM primitive is the DCT clock divided by four. This clock is divided by four due to both channels of the DAC configured for DDR reception. Nonetheless, the input clock provided to the MMCM primitive mirrors the performance of the PLO, with additional noise contributed by the divider, and is used to generate all critical internal and external clocks for the FPGA. This includes the system clock that drives the GWPP architecture as well as the input clocks required by the SERDES to achieve the data rate requirements imposed by the DAC. Recall that in the discussion of the SERDES it was established that the GWPP architecture must generate 16 samples every clock cycle. With a 2400 MHz DAC update rate, this corresponds to each sample generated at $\frac{2400}{16}$, or 150 MHz. Since the GWPP architecture generates the samples, 150 MHz is effectively used as the *system clock*. This system clock is fed to each instantiation of the GPS L1 C/A 1 Sample Generator design element, which its implementation is discussed in the next section. In addition, this clock as well as the original DCT clock divided by four are sent to the SERDES, which handles the transfer of samples from the FPGA to the DAC.

3.2.3 GPS L1 Course Acquisition 1 Sample Generator.

It was established in the hardware implementation high level overview that the GWPP architecture generates 16 samples every clock cycle to achieve the data rate requirements imposed by the DAC. In turn, this allows the combination of the GWPP architecture and the DAC (in Mix-Mode operation) to perform direct RF synthesis to the GPS L-band without the need for analog mixers and PLLs. Essentially, the 16

samples are generated in the L1 C/A 16 Sample Generator design element. All operations inside this design element are clocked by the system clock. The RTL design of the GWPP architecture is based on the principles of modular design. Modularity in design is a digital design approach that subdivides the architecture into smaller parts called modules. These modules can be independently constructed and utilized in different system. Additionally, modularity in design simplifies the design complexity and facilitates debugging procedures. Based on the modular design practice, the GWPP architecture is essentially built using one subcomponent, namely the *GPS L1 C/A 1 Sample Generator* design element; hence, it is the most fundamental component in the hardware implementation of this thesis. For this reason, this section focuses on the implementation of the GPS L1 C/A 1 Sample Generator design element. Furthermore, the modular design practice used in this research allows for the ability to replicate this design element 16 times to build the GPS L1 C/A 16 Sample Generator. The implementation of the GPS L1 C/A 1 Sample Generator design element can best be explained and understood in progressive steps. The clearest explanation sequence is to start with the high level overview of this design element. This discussion will primarily focus on the theory of operation regarding this design element. Following the high level overview is the implementation of each subcomponent that builds the GPS L1 C/A 1 Sample Generator design element.

The theory of operation regarding the L1 C/A 1 Sample Generator design element is best understood through the analysis of Figure 19. The main abstraction in Figure 19 is that the objective of the subcomponents inside the C/A 1 Sample Generator design element is to generate one GPS L1 C/A sample. Since the GWPP architecture is based on DDS, this essentially means that the L1 C/A sample is first constructed as a L1 C/A phase quantity which is then converted to its sine amplitude representation via the Sine LUT. The phase quantity, at its most simplest form, is one phase sample

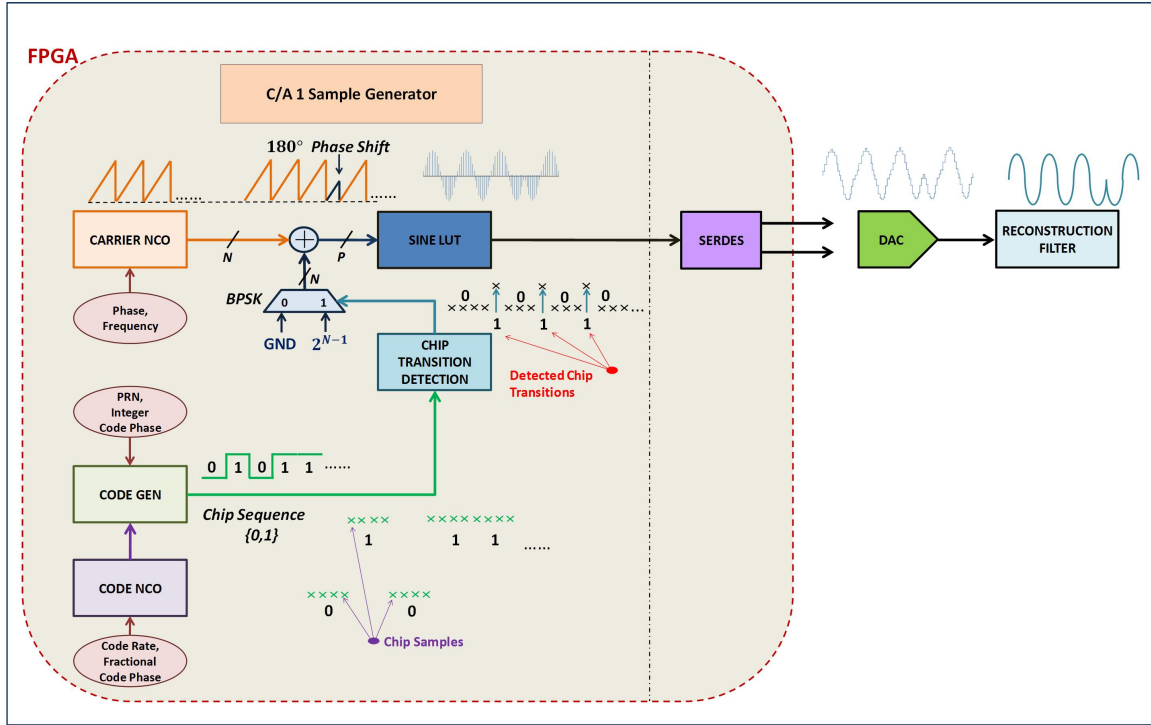


Figure 19. GPS L1 C/A 1 Sample Generator Block Functional Block Diagram

of the GPS L1 carrier BPSK modulated with one phase sample of the C/A-code. The Carrier NCO, which is derived from a phase accumulator, generates the GPS L1 carrier sample. The initial carrier phase and carrier frequency is programmed through software. It is important to note that the burgundy ovals with arrows pointing to its respective design element is a notation used in Figure 19 to denote that this design element is programmed through software. Furthermore, since the Carrier NCO is derived from a phase accumulator, its output is represented as a ramp function in the analog domain. One period of this ramp function corresponds to one period of the GPS L1 carrier. This process constitutes the generation of the GPS L1 carrier phase sample.

The process involved in generating the C/A-code sample is based on both the Code Gen and Code NCO design elements. Like the Carrier NCO, both the Code Gen and Code NCO design elements are programmed through software. The Code

Gen design element sets the integer code phase of the C/A-code and is unique to each PRN, whereas the Code NCO design element sets the C/A-code chipping rate as well as the fractional code phase. The C/A-code sample is an output of the Code Gen design element. The chip sequence consists of zeros and ones, which is also shown in Figure 19. Since the Code NCO design element is derived from DDS principles, the overflow rate of its phase accumulator controls the chipping rate of the chip sequence. Furthermore, the chip sequence samples are sent to the Chip Transition Detection logic. Recall from Section 2.1 that BPSK modulation is a technique that involves either leaving the phase unaltered or shifting the phase by 180° . However, the 180° phase shift only occurs when the chip sequence transitions from 0 to 1 or 1 to 0. Therefore, the Chip Transition Detection design element is needed. It essentially translates the chip sequence into pulses of binary zeros and ones where a pulse equivalent to one corresponds to a chip transition. Afterwards, the pulse essentially acts as a select line to a Multiplexer (MUX) holding phase offsets predetermined to shift the phase by 0° or 180° depending on the binary value of the pulse. Moreover, the N -bit selected offset as well as the N -bit output of the Carrier NCO are both fed to an adder. Since the output of the Carrier NCO is an instantaneous phase quantity, adding the selected offset represents shifting the phase. Therefore, the adder essentially performs the phase modulation and the inputs to the adder, in this case, are configured to perform BPSK as shown in Figure 19. The output of the adder represents a phase modulated quantity and it is truncated to index the Sine LUT. This truncation is referred to as phase truncation and it effectively reduces the amount of memory resources required to convert the phase value to its sine amplitude representation. However, it is important to note that the output of the Sine LUT is a time series of digital words that represent a sinusoidal waveform. Furthermore, the DAC converts the time series of digital words to an analog waveform, yielding a real-world

GPS L1 C/A signal at the DAC output. As mentioned previously, the samples must be sent to the SERDES primitive to be transferred to the DAC in the correct format at the correct data rate.

In summary, the C/A 1 Sample Generator design element is the most fundamental component in the GWPP architecture. Essentially, it is instantiated 16 times to satisfy the 16 samples per clock cycle requirement imposed by the DAC. The high level overview detailed the theory of operation regarding this design element as well as the signal flow. Next, the actual RTL implementation of each subcomponent of the C/A 1 Sample Generator is presented beginning with the generation of the C/A-code. Afterwards, the implementation of the Carrier NCO followed by the Sine LUT are presented.

C/A-Code Generation. The design of the C/A-code generator in the GWPP architecture is based on the properties and characteristics of the C/A-code, which are well defined in reference [5]. However, before designing the C/A-code generator, there are several characteristics to take into consideration; namely, *chip period*, *chipping rate* and *sequence length*, to ensure accurate generation of the C/A-code in hardware. The acknowledgment of these characteristics provides insight to the design decisions made to build the C/A-code generator in the GWPP architecture. First, the chip period refers to the minimum time interval between the transitions in the PRN waveform, which for GPS C/A-code, the chip period is 977.5 ns. Next, the chipping rate is simply the reciprocal of the chip period, which is 1.023 Mcps, or 1.023 MHz, for the GPS C/A-code. Last but not least, the sequence length refers to the number of chips used to generate the PRN waveform over one period, which for C/A-code is 1,023 chips. Another key characteristic used frequently in the discussion of the C/A-code generator is *code phase*, the independent time parameter for the PRN ranging code expressed in the units of chips. The definition of code phase is generalized in the

following example. In GPS terminology, code phase refers to the position of a point in time (chip) on the PRN waveform. It is described as a quantity consisting of both an integer part and a fractional part. For example, a PRN waveform with code phase of 100.5, really means that at this moment in time, the PRN sequence is halfway between chip 100 and chip 101. In this example, the integer code phase is 100, whereas .5 is the fractional code phase. Understanding code phase as well as the previously mentioned characteristics are valuable to understanding how the C/A-code generator is designed in this architecture. Essentially, the C/A-code generator is designed to generate the C/A-code based on the C/A-code properties and characteristics defined in reference [5].

The high level overview of the C/A-code generator is illustrated in Figure 20. At a high level of abstraction, this design architecture shows that the generation of the C/A-code for this platform is derived from two unique design elements: *Code NCO* and *Code Gen*. The Code Gen design element sets the integer code phase, whereas the Code NCO sets the fractional code phase. The ability to set the code phase to different quantities illustrates the capability this architecture has to simulate different code delays in the C/A-code with different PRN sequences. Hence, adding to the flexibility this architecture has in generating GPS waveforms. In addition, recall that one of the goals of GWPP is to provide a reconfigurable architecture to verify arbitrary signals of interest. Whether this includes verifying new signal designs or generating new ones for advanced signals research, one purpose of this platform is to provide signal prototyping capability in realized hardware. To fully create a flexible, realistic platform, the design must incorporate key factors that would be present in a real life SV signal transmission environment such as code phase delay effects. Code phase delay effects are not only vital to building a realistic C/A-code generator, but also provides advanced opportunities to incorporate signal authentication techniques

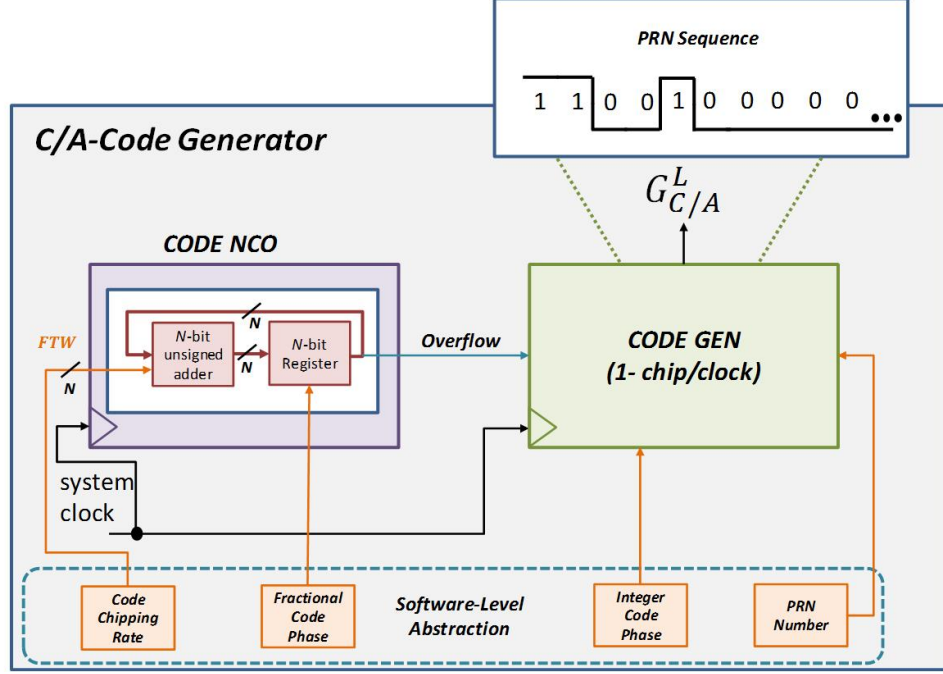


Figure 20. C/A-Code Generator Functional Block Diagram

in the C/A-code. It has been established that the Code NCO and Code Gen design elements are vital to generating a unique PRN sequence with zero or arbitrary code phase delay. It is also important to note that the chipping rate of the PRN sequence is controlled by the output of the Code NCO, namely the rate at which the N -bit register overflows. Before presenting further details on the implementation of the Code NCO and Code Generator design elements, it is worth reinforcing the concept of software-defined capability in the C/A-Code Generator. Essentially, the waveform characteristics such as code frequency/phase and PRN sequence are programmed through software. This effectively allows the capability to any one of these characteristics dynamically without having to redesign the C/A-Code Generator.

It has been established that the C/A-Code generator is derived from two design elements—Code NCO and Code Gen. The design of each of these elements is important to generating the correct C/A-code at the correct chipping rate for an arbitrary PRN. Next, the focus will be on the design and implementation of each of these

elements. First, the discussion will start with the implementation of the Code Gen design element. Afterwards, the discussion will transition to the implementation of the Code NCO.

Code Generator. This design element is one of the principle components that builds the C/A-Code Generator on this platform. It sets the integer code phase for a given PRN and thereby allows the C/A-Code Generator to begin a PRN sequence at any given integer code phase described in the following set $\{x \in \mathbb{N} \mid 0 \leq x \leq 1022\}$, where x refers to the chip number. The functional block diagram for the Code Generator design element is illustrated in Figure 21. There are several important concepts that can be extracted from Figure 21. First, the Code Gen design element implementation in hardware is derived from two 10-bit maximum length Linear Feedback Shift Registers (LFSRs), namely $G1$ and $G2$. A LFSR is a digital circuit that comprises n bits of memory and some feedback logic, all clocked at the same rate. The feedback logic is what creates the pseudo-random characteristic of the output sequence. The purpose of using LFSRs is summarized below:

1. C/A-codes are Gold codes with a sequence length of 1,023 chips and a period of 1 ms. Gold codes, which consists of two m -sequences with equal length combined through modulo-2 arithmetic, are chosen for their good auto- and cross- correlation properties.
2. The theory of operation regarding LFSRs follow the same properties used in the generation of Gold codes. For example, an n -bit LFSR can also generate an output sequence of length $N = 2^n - 1$ before repeating. A LFSR that produces a sequence of this length is referred to as maximum-length.
3. Since the C/A-code is a member of the Gold code family, the architecture of the Code Generator is built using two n -bit LFSRs, with feedback logic, and

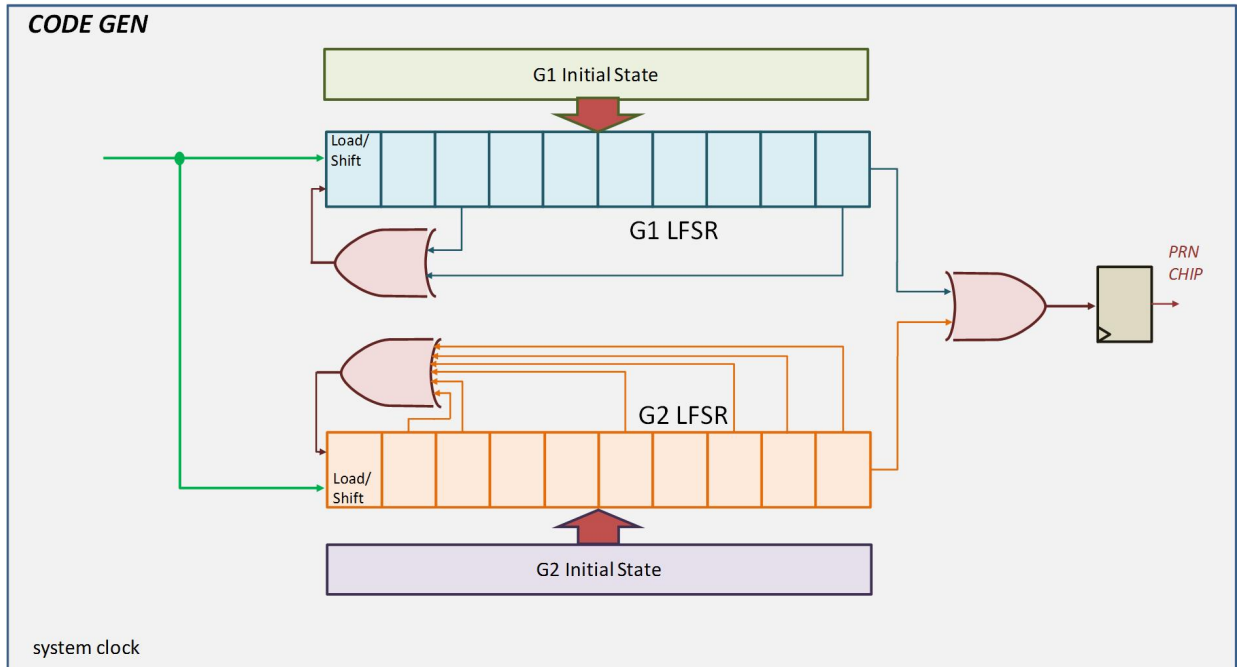


Figure 21. Code Generator based on two 10-bit maximum length LFSRs

a Exclusive-Or (XOR) circuit. Since the C/A-code is 1,023 chips long, this requires each LFSR size to be $\log_2(1024)$, or 10 bits.

It is important to note that using 10-bit LFSRs results in the ability to generate 1,023 different sequences; however, since one of these sequences is a sequence of all zeros and a sequence of zeros does not produce a psuedo-random sequence, this effectively reduces the number of usable sequences from 1,023 to 1,022. In addition, since PRN sequences are required to have great auto- and cross- correlation properties, this effectively reduces the number of usable sequences for GPS applications from 1,022 to 210. This is further discussed later in this section.

The next important concept that can be extracted from Figure 21 is that the Code Generator design element operates in one of two states: load or shift. Load is essentially a state in which the G1 and G2 LFSRs are programmed with a new initial state, respectively. The initial states are calculated and programmed through

software; hence, extending the software-defined capability to this design element. The ability to initialize each LFSR to an initial state in hardware is based on the set/reset property of FFs. Furthermore, the initial state for each LFSR determines the output PRN sequence and a further description of how this is accomplished is presented later in this section. On the other hand, shift, as the name implies, is a state in which the G1 and G2 LFSRs essentially follow the theory of operation regarding LFSRs. In other words, an output chip is produced, the shift operation occurs followed by the linear feedback operation. The taps chosen for linear feedback with respect to both G1 and G2 LFSRs are based on the polynomial equations for G1 and G2 described in [5]. For the G1 shift register, the design specification for the C/A-code calls for the feedback taps to be connected to stages 3 and 10. These register states are combined with each other by an XOR circuit and fed back to stage 1. The polynomial that describes this shift register architecture is: $G1 = 1 + X^3 + X^{10}$. For the G2 shift register, the design specification for the C/A-code calls for the feedback taps to be connected to stages 2, 3, 6, 8, 9 and 10. These register states are combined with each other by an XOR circuit and fed back to stage 1. The polynomial that describes this shift register architecture is: $G2 = 1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10}$. This entire process (output, shift, apply linear feedback), for the sake of this discussion, will be referred to as one cycle of the shift state. The rate at which the shift stage completes one cycle is clocked by the system clock, however, it is controlled by the Clock Enable (CE) input (not shown in Figure 21). The CE input is provided by the Code NCO and is used in this design element to output the PRN sequence at the nominal C/A-code chipping rate, 1.023 Mcps, using the 150 MHz system clock. Essentially, the CE pin divides the system clock down to a slower clock rate; consider the following: a FF is clocked at 150 MHz; although, the frequency at which the clock enable pin is asserted is 1.023 MHz. This ultimately results in the data transferring from the

input of the FF to the output at a rate of 1.023 MHz. Initially, without using the CE pin, a 1.023 MHz input clock would be required to generate the PRN sequence at the nominal chipping rate. This introduces Clock Domain Crossing (CDC) challenges since all of the other design elements inside the GWPP architecture is clock by the system clock (150 MHz). Interfacing two logic blocks working at two different clocks is challenging in the sense that the design becomes asynchronous at the boundary of each interface. This often results in setup/hold violations, metastability, an unreliable data transfer. In addition, performing clock domain synchronization requires additional logic resulting in the use of extra hardware resources. Therefore, utilizing the CE pin for this design element not only allows for the ability to generate the PRN sequence at the nominal chipping rate, but also eliminates possible CDC challenges.

It has been established that the Code Gen design element, every clock cycle, operates in one of two states—load or shift. In the load state, G1 and G2 LFSRs are initialized to an initial state which thereby sets the PRN sequence. In the shift state, depending on the assertion of the CE input, outputs the PRN chip, performs the shift operation, and applies the linear feedback operation. The output rate of the chip, which is referred to as the chipping rate, is determined by the CE input provided by the Code NCO. For this thesis, the CE input is configured for active-high logic, therefore, the rate at which the CE input is active-high sets the chipping rate of the output PRN sequence. Before proceeding to the details of how the CE input is provided by the Code NCO, it is worth discussing how the Code Gen architecture designed in this research effort differs from traditional implementations described in literature, [1] and [3].

The code generator in this thesis is designed to generate a PRN chip based on the theory of initializing the LFSRs to pre-calculated initial settings. Then, the PRN chip is generated through the XOR logic operation between the direct outputs from

the G1 and G2 LFSRs, respectively. In other words, the unique C/A-code for each SV is the result of $G1_{10}$ (last stage in G1 LFSR) XOR'ed with $G2_{10}$ (last stage in G2 LFSR), illustrated in the equation $Gi(t) = G1(t) \oplus G2(t)$, where i corresponds to a satellite and t represents the 1.023 MHz clock. This equation shows that C/A-code is different for each PRN as long as the initial contents in the G1 and G2 LFSRs are unique to that PRN. However, this equation differs from the traditional C/A-code equation presented in [1]. Before the advantage of the GWPP code generator design architecture can be understood, the limitations of the previous design architecture used in [1] and [3] must be understood. The traditional equation for the C/A-code follows the equation $Gi(t) = G1(t) \oplus G2(t + diTg)$, where $diTg$ represents the amount of delay in chips to apply the G2 shift register. As a result, it is already shown that the G2 LFSR implemented in this design element differs from traditional implementations presented in [1] and [3]. Although, it is worth discussing as it provides insight to the reasoning behind the design of the code generator in this platform. In previous implementations of the C/A-code generator, the unique C/A-code was the result of the G1 direct output sequence XOR'ed with a delayed version of the G2 direct output sequence. This concept is based on a specific property of the maximum-length PRN code sequence that states: adding a phase-shifted version of itself generates the same PRN sequence, but at a different phase [1]. To accommodate for this property in design, the delay effect in the G2 PRN code is created through a process called phase selection. In phase selection, two specific taps in the G2 LFSR are selected and the XOR of these taps is the output of the G2 LFSR, which is then fed as one input to the output XOR circuit (last XOR operation that generates the PRN chip). The other input to the last XOR circuit is the direct output of the G1 LFSR, which this operation results in the generation of the PRN chip. Therefore, the two tap phase selection logic shifts the code phase in G2 with respect to the code phase in G1

without the need for an additional shift register to compute this delay. Furthermore, the phase selection taps are unique to generating one specific PRN sequence. Hence, two pairs of tap selections cannot be equivalent or else replica PRN sequences will be generated. This concept is illustrated on a small scale in Table 3, which is a subset of [1, Table 4.3].

Table 3. Code Phase Assignments and Code Sequences for C/A-Code

<i>SV PRN Number</i>	<i>C/A-Code Tap Selection</i>	<i>C/A-Code Delay (Chips)</i>	<i>First 10 C/A Chips (Octal)</i>
1	$2 \oplus 6$	5	1440
2	$3 \oplus 7$	6	1620
3	$4 \oplus 8$	7	1710
4	$5 \oplus 9$	8	1744
5	$1 \oplus 9$	17	1133

Table 3 reinforces the concept that the phase selection taps are uniquely paired to generate one unique PRN. In order for phase selection to work, both G1 and G2 LFSRs must be initialized to all ones. For example, the first 10 C/A-chips for PRN 1, is a result of $G1_{10} \oplus (G2_2 \oplus G2_6)$. Although this technique enables the ability to generate multiple PRNs, it also introduces long term limitations. With this implementation, based on combinations and permutations, the G2 generator can only create 37 possible unique outputs, or PRN sequences, shown in further detail in [1, Table 4.3]. In other words, the two-tap technique is only capable of generating a limited set of valid C/A-codes. Since GPS utilizes CDMA principles, each satellite employs their own unique PRN sequence. With only 37 possible PRN sequences able to be generated, this limits the GPS constellation to only 37 satellites. At a time when 32 satellites were sufficient to operate with, the phase selection method presented no long-term limitations. However, only able to generate 37 possible PRNs limits the future for GPS expansion and modernization. Interface specification, [5], presents a new concept for generating the C/A-code based on an initial G2 setting.

With this method, code phase assignments have expanded from 37 possible PRNs to 210. This concept takes advantage of the maximum-length LFSR property discussed earlier where loading the LFSRs with unique initial settings results in the generation of different sequences. This concept is illustrated in Table 4, which shows how several C/A-codes for different PRNs such as {38,63,95,125,210} are generated using different initial G2 settings. This concept applies to the set of PRNs in the range (1:210), which is discussed further in [5, Table 3-IB and Table 6-I].

Table 4. Expanded Code Phase Assignments and Code Sequences for C/A-Code

<i>SV PRN Number</i>	<i>G2 Delay (Chips)</i>	<i>Initial G2 Setting (Octal)*</i>	<i>First 10 C/A Chips (Octal)</i>
38	67	0017	1760
63	1018	1745	0032
95	446	1234	0543
125	235	1076	0701
210	310	0731	1046

The main takeaway from Table 4 is that loading the G2 LFSR with an initial setting enables the expansion of PRNs from 37 to 210. Although this results in the generation of a 210 unique PRN sequences, the integer code phase with respect to each sequence, based on Table 4, is set to zero. In turn, it inhibits the ability to apply a code delay to the C/A-code. This capability is very important when implementing a realistic C/A-code generator. In addition, applying code delays to the C/A-code is an important factor in advanced signals research. Advanced signals research focuses on new signal design concepts to mitigate the ability to spoof or disrupt GPS signals. One method that is becoming interestingly popular is the alteration of signal characteristics. If this platform can only generate the PRN sequence at an integer code phase set to zero, it inhibits the flexibility in designing new methods for altering the C/A-code. Therefore, the capability to change the integer code phase is an important consideration in this design. In order to set any integer code phase for a given a PRN,

it requires both G1 and G2 shift registers to be initialized with specific initial settings that are derived from special properties of the maximum-length PRN code sequence in which adding a phase-shifted version of a PRN code sequence produces the same sequence but at a different phase. In other words, adding phase delays to the initial settings previously presented in Table 4 allows for the ability to generate any one of the 210 PRN codes at any integer code phase in the set $\{0,1022\}$. This concept is reinforced in Table 5. To begin a PRN code sequence at a specific integer code phase, the initial settings for G1 must be phase shifted, whereas the initial settings for G2 must be both phase shifted and specific to that PRN, based on the G2 initial setting. Therefore, the initialization settings for G1 and G2 need to be carefully constructed to generate the desired PRN sequence with the desired code phase.

Since the intent of this platform is to be used with advanced signals research, the ability to generate any of the 210 PRN sequences at any integer code phase in the set $\{0,1022\}$ introduces additional flexibility in the C/A-code generation logic designed in the GWPP architecture. This constitutes the design of the Code Gen design element, which sets the integer code phase for a specific PRN. In result, the output of this design element is a PRN sequence consisting of pseudo-random 1's and 0's. The rate of the PRN sequence, as mentioned previously, is determined by the Code NCO. The design and implementation of the Code NCO is presented next.

Code NCO. This design element is the second principle component that builds the C/A-code generator on this platform. It generates the chipping rate provided to the code generator CE pin. In addition, it sets the fractional code phase for a given PRN; effectively allowing the C/A-code generator to begin a PRN sequence at any given fractional code phase described in the following set $\{y \in \mathbb{Q} \mid .00 \leq y \leq .99\}$, where y represents a fraction of a chip. The combination of the Code NCO and Code generator enables the C/A-Code Generator in theory to begin a PRN sequence at

Table 5. Example GPS C/A-Code Generator Initial States

<i>Register</i>	<i>PRN</i>	<i>Code Phase</i>	<i>Initial Setting</i>
C/A-code G1	1	0	111111111
C/A-code G2	1	0	1111101100
C/A-code G1	1	1	011111111
C/A-code G2	1	1	1111110110
C/A-code G1	1	2	001111111
C/A-code G2	1	2	1111111011
C/A-code G1	31	100	0110001001
C/A-code G2	31	100	1000101010
C/A-code G1	210	0	111111111
C/A-code G2	210	0	1001101110

any code phase in the set $\{c \in \mathbb{R} \mid 0.00 \leq c \leq 1022.99\}$, where c represents the code phase and $\mathbb{R}$ indicates it is a member of the real numbers domain. However, in this current release, the acceptable range for fractional code phase values is a subset of this range for several reasons discussed later in this section.

The architecture of the Code NCO is derived from DDS principles. Therefore, the heart of the Code NCO is a phase accumulator. Recall from Section 2.3 that the phase accumulator increments in a staircase manner, which in the analog domain represents a ramp function. The staircase output of the phase accumulator is shown in Figure 22. Essentially, Figure 22 shows that the FTW value is accumulated every clock cycle. The output of the phase accumulator increases linearly in a staircase manner. However, its analog waveform equivalent is inherently a sawtooth, or ramp function. One overflow period of the accumulator equates to one period of the sawtooth or ramp waveform. Intuitively, the staircase represents the fractional code phase. Eventually, however, the accumulator output grows to the point where the next rising edge of the system clock and subsequent accumulation of the FTW causes the accumulator to reach or exceed the upper limit, $2^N - 1$. That is, the accumulator will overflow and the overflow amount is a remainder. The overflow intuitively represents the transition to the next chip of the PRN sequence. It is also important to note that when the

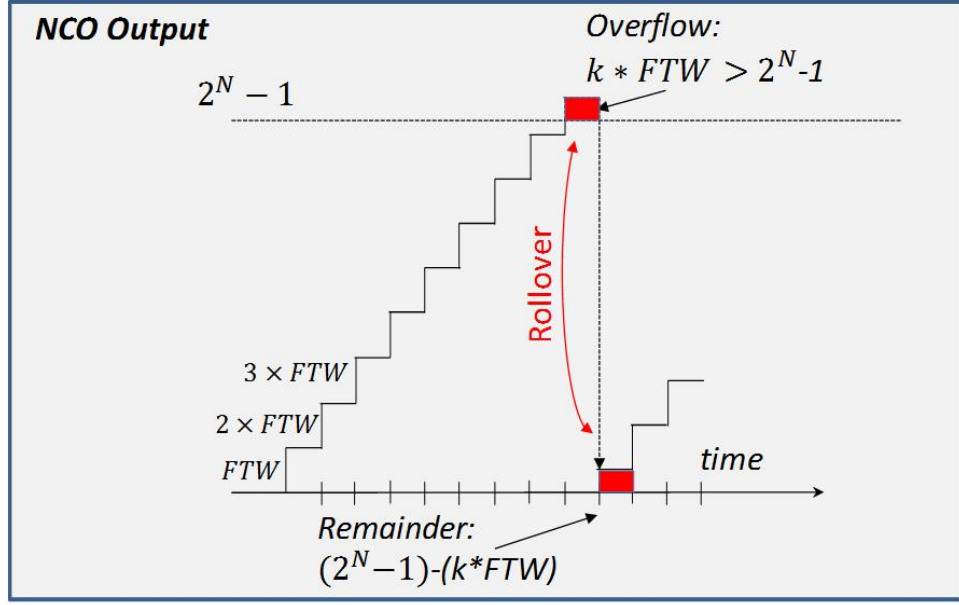


Figure 22. NCO Staircase Output

accumulator overflows it rolls over to begin the start at a new fractional code phase. The start value is the remainder, which may or may not be zero depending on the accumulator size and FTW . The main abstraction is that the accumulation will reach or exceed its maximum value and overflow. The rate at which the overflow occurs dictates the output frequency of the Code NCO. The increment at which the staircase shape grows is dictated by the FTW value. Hence, it is crucial to choose the FTW value that forces the accumulator to overflow at a rate equal to the GPS C/A-code nominal chipping rate. It is also important to note that based on Figure 22, the accumulation can begin at any “step”. This essentially represents the ability to start at an arbitrary fractional code phase. For example, starting at the step that is halfway between 0 and $2^N - 1$ intuitively represents starting the Code NCO at a fractional code phase offset of .5, or half a chip. Since the accumulation begins at half a chip, the accumulator will overflow twice as fast compared to starting at zero fractional code phase. Furthermore, to start at any arbitrary phase requires the user to program the phase accumulator register in the Code NCO design element.

Equation (3.2.1), which is shown below, is used in software to calculate the initial phase value to program the Code NCO to begin accumulation at an initial fractional code phase.

$$\text{Code NCO Initial Phase} = \text{round}(2^N * \text{FractionalCodePhase}) \quad (3.2.1)$$

where *FractionalCodePhase* is any number, x , ideally in the range $.00 \leq x \leq .99$. However, for reasons discussed later in this section, for the current GWPP architecture, the fractional code phase can only be set to any number in the range $.00 \leq x \leq .50$. Therefore, two important concepts can be seen. First, the concept of an overflowing accumulator is essential to understanding how the chipping rate is established. Second, the Code NCO, based on Equation (3.2.1), can be programmed to start at any fractional code phase offset $.00 \leq x \leq .50$. Ideally, the Code NCO design element should be able to support the ability to start at any arbitrary fractional code phase offset in the range $.00 \leq x \leq .99$. However, synchronization challenges occur due to the fact that 16 Code NCOs are essentially instantiated every clock cycle. As mentioned previously, the reason for 16 instantiations is because 16 samples must be generated every clock cycle to achieve the high speed data rate requirements imposed by the DAC used in this research.

The challenge of synchronization is generalized in Figure 23. Essentially, each Code NCO is connected to its own Code Generator. The idea is for each Code Generator to generate a sample of the C/A-code. As the accumulation continues every clock cycle in the Code NCO, subsequent NCOs will overflow causing the transition to new PRN chips. However, to ensure both smooth phase increments between clock cycles and the generation of a unique sample per NCO, two important synchronization properties must be followed. The first property consists of scaling the FTW value by 16 to ensure smooth phase increments from cycle to cycle. Smooth phase

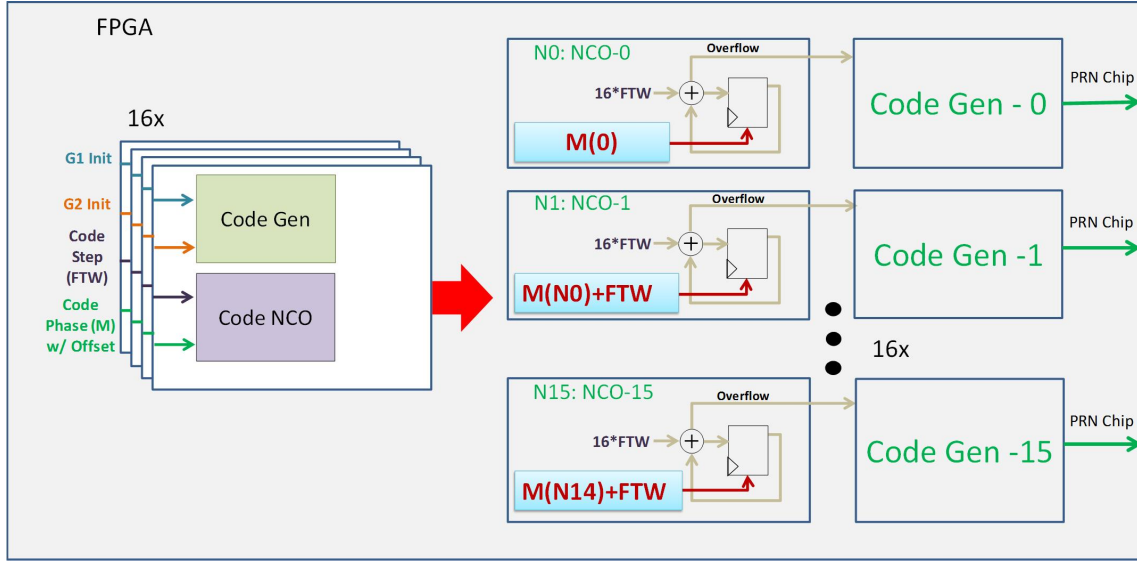


Figure 23. Code Generator 16x Expansion. Main objective is to generate 16 samples at once to meet the data requirements of the DAC and its high speed sampling rate.

increments are desired when considering the test equipment that will eventually be used with GWPP such as GPS receivers. GPS receivers use several tracking loops to acquire/track GPS signals. If the output signal from the GWPP architecture has numerous short- or long-term phase jumps, this will essentially break the tracking loops in the GPS receivers. Therefore, it is imperative to ensure smooth phase increments from cycle to cycle. The reason to multiply by 16 is due to the fact that 16 samples are being generated every clock cycle. If only 8 samples were being generated, the scaling will effectively change to multiplying the FTW value by 8. The multiplication is performed in hardware using the left shift operation. The logical left shift has the following property, $CodeStep * 16 = CodeStep * 2^4 = CodeStep \ll 4$. Essentially, shifting to the left by 4 replicates the multiply by 16 mathematical operation. This is efficient to perform in hardware because it is implemented in the switch matrix instead of using multipliers in the DSP48 slices of the FPGA, which are limited resources. The ability to perform this operation in the switch matrix allows for the ability to save DSP48 slices for other applications.

The second property, which introduces the challenge of currently not being able to start at any arbitrary fractional code phase offset in the range $.00 \leq x \leq .99$, is that each Code NCO register must be initialized at an offset. This offset is dependent upon the previous NCO initialization and FTW value. It is important to note that NCO-0 is the NCO that would be initialized to start at a desired fractional code phase offset. The remaining NCOs will be initialized in a recursive format based on the original, unscaled FTW value and the previous NCO's phase initialization value. The challenge lies in initializing the Code NCO phase register to a fractional code phase offset that creates an overflow during the initialization of subsequent Code NCOs. The principle of transitioning to the next PRN chip relies on the overflow property of the Code NCO. If initializing the Code NCOs with the correct fractional code phase offset results in an overflow then its respective Code Generator will transition to the next PRN chip. The challenge is that the previous NCOs that did not overflow are stuck generating samples for the previous PRN chip. Therefore, this issue results in the misalignment of PRN chips during the sample generation process causing an incorrect PRN sequence. To get around this problem, the overflow condition for a desired fractional code phase is tested first in software. Based on the fractional code phase and the FTW value, if an overflow occurs during initialization, the software notifies the user that this code phase offset is not supported in the current GWPP architecture. However, resolving this issue in hardware will be a topic of discussion for future work in Chapter V.

This concludes the discussion of how the C/A-code is generated in the C/A 1 Sample Generator design element. Recall from the high level overview that output PRN sequence is sent to the Chip Transition Detection design element, which ultimately determines when to shift the carrier phase by 180° or leave it unaltered. The discussion of the design and implementation of the Chip Transition Detection logic

is presented next.

Chip Transition Detection and BPSK Modulation. These elements are designed to generate the BPSK offset provided to the Carrier NCO. Recall that phase-shift keying is a digital modulation scheme that conveys data by modulating the phase of the carrier wave. Furthermore, in GPS transmission, the code is combined with the binary navigation data using modulo-2 addition. The composite binary signal is then impressed upon the carrier using BPSK modulation. However, in the first release of this architecture, the navigation message is not implemented. Therefore, only the code is impressed upon the carrier using BPSK modulation. In BPSK modulation, the phase of the carrier signal is shifted by 180° at chip transitions from 0 to 1, or from 1 to 0. Therefore, this presents a need for chip transition detection logic as a method to determine the phase offset to apply to the carrier. The design of this logic is shown in Figure 24. The main abstraction in Figure 24 is that the chip transition detection logic translates the output chip sequence provided by the C/A-Code Generator into pulses of zeros and ones where a pulse equivalent to one corresponds to a chip transition. Afterwards, the pulse essentially acts as a select line to a MUX holding phase offsets predetermined to shift the phase by 0° or 180° depending on the binary value of the pulse. Therefore, in the GWPP architecture, BPSK modulation is performed by adding an offset to the carrier phase accumulation process. In turn, the phase is shifted by either 0° or 180° , as it should in BPSK modulation, however, without the need of costly multiplication in hardware. The BPSK MUX holds the two offsets, which for this case, are the offsets calculated to shift the phase by 180° or leave it unaltered. The design decision to use this combination of circuitry is to incorporate an efficient, flexible method of performing modulation. The MUX can easily store additional offsets to further support additional modulation schemes. Since this architecture is currently designed for GPS L1 C/A synthesis, the

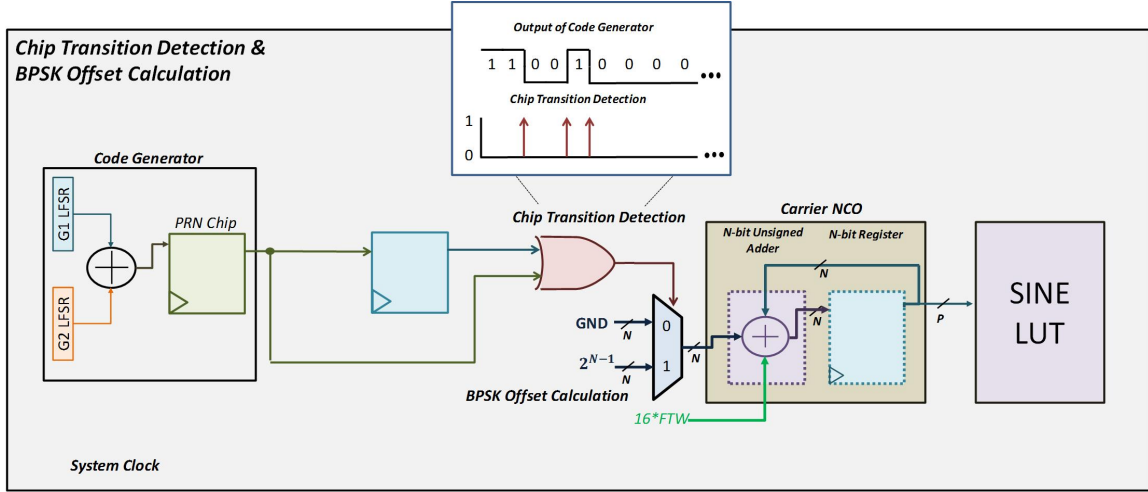


Figure 24. Chip Detection Design and BPSK Offset Calculation

MUX is only limited to support BPSK modulation offsets. Furthermore, the offsets are provided to the Carrier NCO design element every clock cycle to either shift the phase or leave it unaltered.

The theory of using 2^{N-1} as a BPSK offset to the carrier compared to performing multiplication is illustrated through theory and the analysis of Figure 25. First, the theory is presented. Essentially, the theory of using this offset is based on principles of binary representation. The principle is that the first Most Significant Bit (MSB) has a weight equivalent to 180° . Furthermore, the second MSB has a weight equivalent to 90° and so forth. Therefore, the quantity 2^{N-1} effectively represents an offset that will shift the phase 180° . Next, the illustration in Figure 25 will introduce another perspective to reinforce how adding 2^{N-1} to the Carrier NCO shifts the phase by 180° . First, recall the process of accumulation in DDS systems. The phase accumulator increases linearly in a staircase manner. In the analog domain, the staircase represents a ramp or sawtooth function. Figure 25 represents one period of the ramp waveform. Thus, after the accumulator overflows, this intuitively represents the completion of one full cycle. This full cycle can also be translated to one full cycle of a sine wave. Furthermore, this intuitively means that the first half of the ramp waveform period

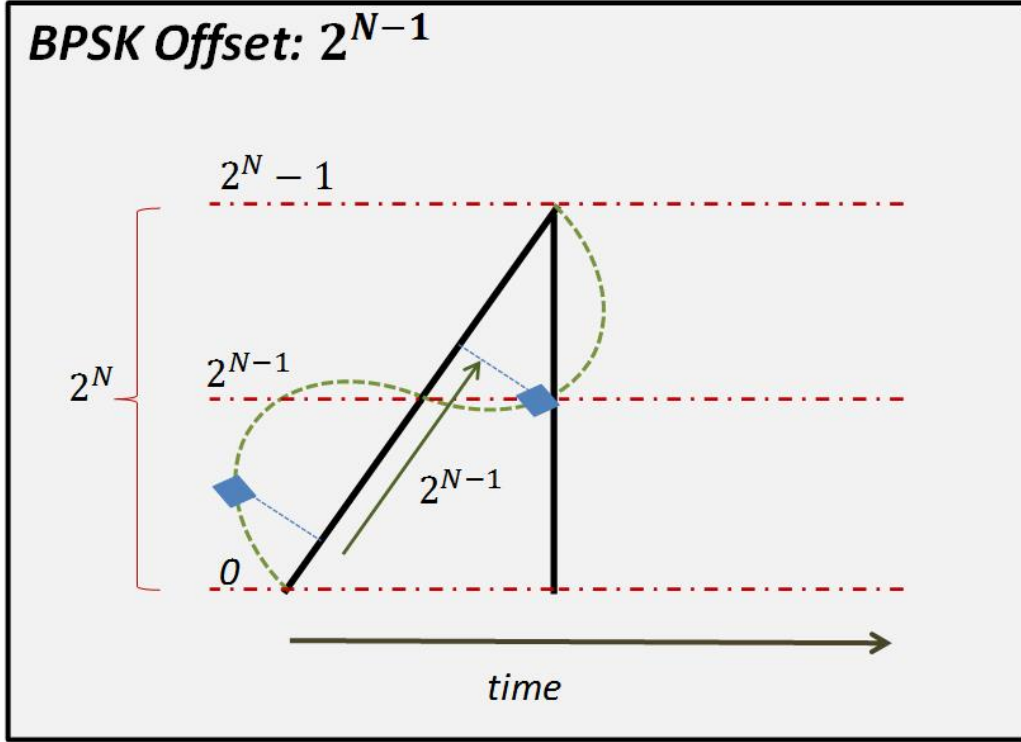


Figure 25. BPSK Offset MUX Theory of Operation. The MSB of a N -bit value, 2^{N-1} , represents 180° .

translates to the first half of a sine wave period. Additionally, the second half of the ramp waveform period translates to the second half of the sine wave period. The ramp function is essentially the analog representation of a N -bit accumulator. Once the accumulator reaches or exceeds its maximum capacity, one period of the ramp function is completed. With an N -bit accumulator, its dynamic range is 2^N , which puts half its dynamic range at 2^{N-1} . Extending this argument, any phase point between zero and half of the period of the ramp function can be shifted to the second half of the ramp function by adding half of its dynamic range. If the first half of a sine wave corresponds to the first half of the ramp function and the second half of the sine wave corresponds to the second half of the ramp function, then using half of its dynamic range to shift the phase value from one half of the waveform to the other is essentially the same as performing BPSK modulation. This concept is also generalized

in Figure 25; consider the case for the phase point marked by the blue diamond on the sine wave. The phase point is initially denoted in the first quadrant of the sine wave where the phase magnitude is increasing. After an offset of 2^{N-1} was added, the phase point shifted to the third quadrant where the phase magnitude is increasing once again. Essentially, the majority of samples in the second quadrant, where the phase magnitude decreases, are bypassed. This effectively reinforces how adding an offset 2^{N-1} performs BPSK modulation. This technique also provides several benefits for signal prototyping using the GWPP architecture. First, using the MUX allows for a cost-efficient and flexible way to apply additional modulation schemes in hardware. Additionally, it enhances the ability to generate advanced signals where precise BPSK modulation can be applied for signal authentication capability.

It has been established that the Chip Transition Detection design element translates the PRN sequence into a sequence of pulses consisting of 1's and 0's. The 1's essentially correspond to the detection of chip transitions and effectively causes an offset of 2^{N-1} to be provided to the Carrier NCO design element. This offset is used to shift the carrier phase by 180° . This concludes how the chip sequence is translated to phase offsets to be applied to the instantaneous carrier phase. Recall that the instantaneous carrier phase is produced in the Carrier NCO design element. The discussion of the design and implementation of the Carrier NCO is presented next.

Carrier NCO. This design element generates the samples to construct the GPS L1 carrier wave. Recall that the carrier frequency for GPS L1 C/A is 1575.42 MHz. The architecture and signal flow of the Carrier NCO design element is illustrated in Figure 26. Essentially, the main abstraction in Figure 26 is that the Carrier NCO design element is based on DDS principles. Therefore, a phase accumulator, which is derived from a N -bit unsigned adder and N -bit register, is at the heart of this architecture. Since the output of the phase accumulator is a phase quantity, the

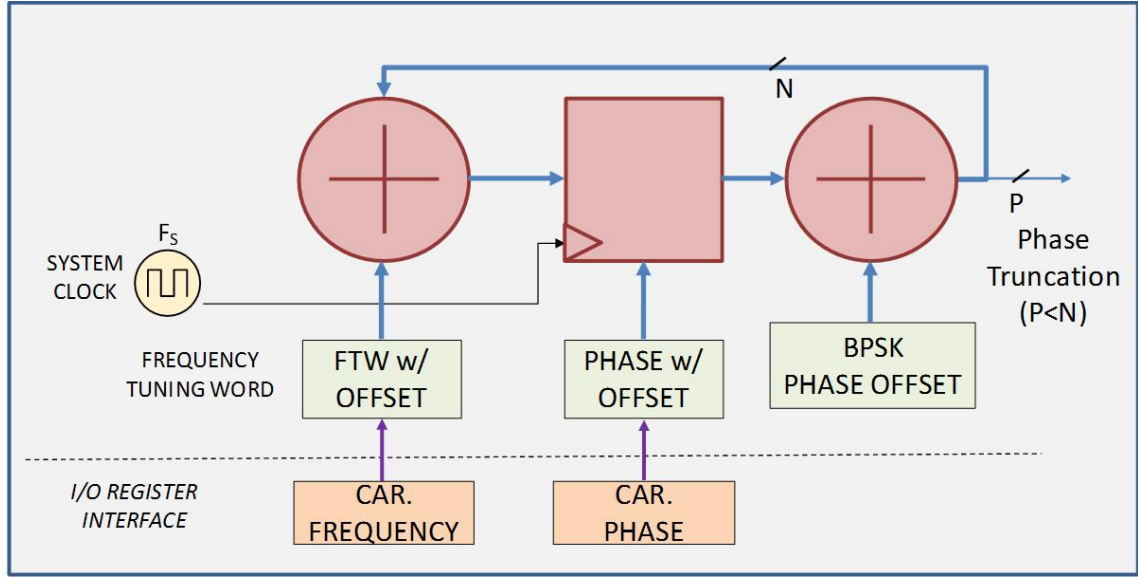


Figure 26. Carrier NCO Architecture Signal Flow

adder placed after the register essentially shifts the phase by a specific offset. As mentioned in the discussion of the Chip Transition Detection design element, the BPSK phase offset is either 0 or 2^{N-1} . These offsets leave the carrier unaltered or shift the carrier by 180° , respectively. There are two main concepts that can be extracted from Figure 26. First, this design element also incorporates support for software-defined reconfigurability. Essentially, the carrier frequency and carrier phase are programmed through software and can be dynamically changed without changing the Carrier NCO architecture. The second concept is that the programmed quantities corresponding to the carrier frequency and phase are applied an offset to ensure both smooth phase increments from cycle to cycle and the generation of unique L1 C/A samples. The application of the offsets follows the same process described in the discussion of the C/A-Code Generator. However, this design element does not face the synchronization challenge faced by the C/A-Code Generator. This is because the output of the Carrier NCO addresses the Sine LUT and does not control the generation of a pseudo-random sequence like the C/A-Code Generator. Last, the third concept is that N -bit instantaneous phase modulated output is phase truncated. The

reason for performing phase truncation was presented in Section 2.3, however, it will be reinforced in the discussion of the Sine LUT design element. Before proceeding on to the details regarding how the Sine LUT design element was implemented, it is worth presenting one last final concept regarding the Carrier NCO design element, which is shown in Figure 27. Essentially, Figure 27 shows how the carrier phase accumulation is performed using four instantiations of the Carrier NCO design element. Each Carrier NCO is responsible for building one “step” in the staircase output of the Carrier NCO. Since there are four instantiations, four accumulations occur every clock cycle. Eventually one Carrier NCO will overflow first causing the subsequent Carrier NCOs to overflow. Recall that once the accumulator overflows, the accumulation begins at the remainder. The concept shown in Figure 27 can be expanded to 16 instantiations to represent the GWPP architecture. However, four instantiations are sufficient to demonstrate proof of concept.

It has been established that the Carrier NCO design element generates the BPSK modulated GPS L1 C/A phase samples. The next step in the C/A 1 Sample Generator design element is to convert the L1 C/A phase quantity to its sine amplitude representation to be sent to the DAC. To perform this step, the phase sample is sent to the Sine LUT design element to perform the phase to amplitude conversion. The implementation of the Sine LUT design element is presented next.

Sine LUT. This design element essentially performs the phase-to-amplitude conversion. This component is the building block that enables a DDS system to produce a sinusoidal output. Furthermore, it is essentially a look-up table of stored sine wave values. Figure 28 illustrates both the signal flow and high level overview of the phase-to-amplitude conversion. The main abstraction in Figure 28 is that the Carrier NCO design element generates the phase samples of the GPS L1 C/A signal and the Sine LUT design element translates this phase value to its sine amplitude represen-

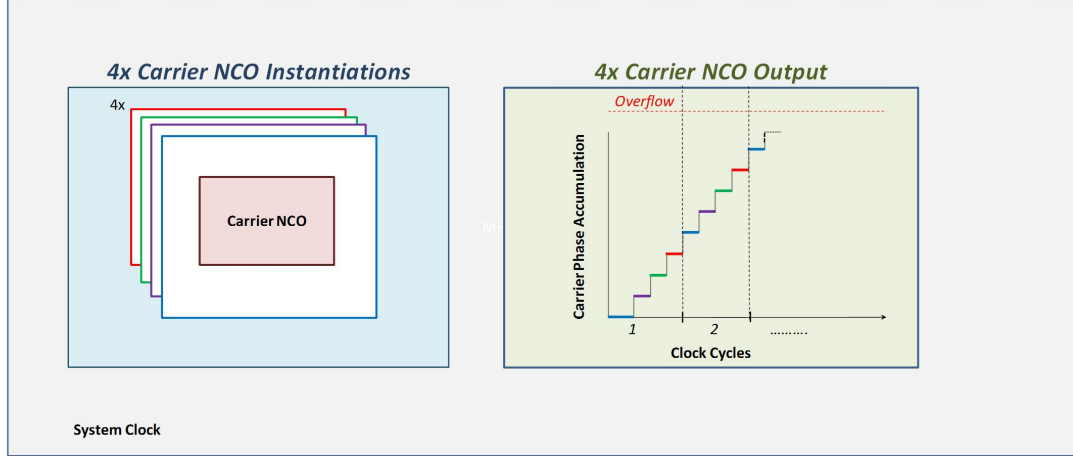


Figure 27. Carrier NCO 4x Expansion

tation to be sent to the DAC. The phase-to-amplitude conversion is performed in two arrangements of logic: half wave Sine LUT and symmetry logic. The half wave Sine LUT is designed to exploit the symmetrical properties of sinusoids; specifically, $\sin(\alpha) = -\sin(\pi + \alpha)$. It is synthesized as distributed ROM in the FPGA. The resources in the FPGA to implement to half wave LUT are namely the SLICEMs in the CLBs. SLICEMs occupy 25% to 50% of all slices in the FPGA. Therefore, these are limited resources. The idea of using a half wave Sine LUT is to free up SLICEMs to be used for other applications. The symmetry logic is designed to generate the second half of the sine wave. The symmetry logic is essentially negating the sign of the output of the LUT depending on the value of phase sample's MSB to generate the second half of the sine wave. This logic effectively allows for the ability to store half the sine wave in the LUT synthesized in distributed ROM and use symmetry logic to generate the other half of the sine wave. Another important concept shown in Figure 28 is that the D -bit output of the Sine LUT is increased by 1 bit. This is another design decision employed to optimize the memory resources used for the Sine LUT implementation.

One important design consideration when implementing the Sine LUT design

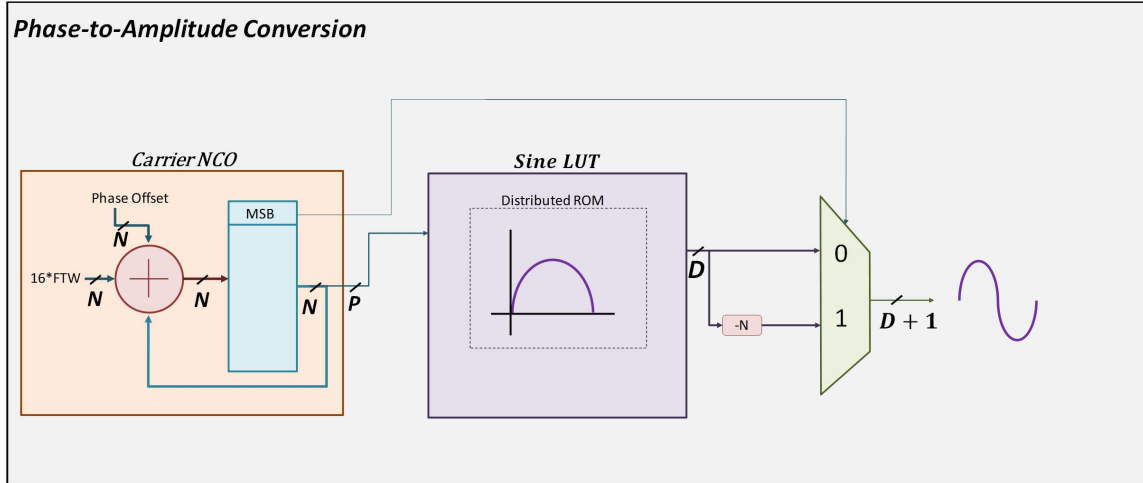


Figure 28. Phase-to-Amplitude Conversion High Level Overview and Signal Flow

element is the size of the LUT. In an ideal DDS, the width of the input bus to the Sine LUT is the same as the width of phase accumulator output. This is impractical, however, is N is large. For example, an accumulator with a width of 32 bits would contain over 4-billion entries. If each entry is stored in 8-bit precision, this amounts to almost 4 gigabytes of required memory. This is clearly not practical for a hardware implementation. To get around this problem, the Carrier NCO relies on truncating the N -bit accumulator output at the input to the Sine LUT design element. Recall from Section 2.3 that this is typically accomplished by retaining the P MSBs of the N -bit accumulator output as the input to the Sine LUT. This is known as phase truncation and significantly reduces the size of the LUT in distributed ROM where N is large. Furthermore, the number of bits retained to perform the phase-to-amplitude conversion determines the number of samples used to represent a sine wave. The number of samples used to generate the sine wave is referred to as the phase resolution. The number of bits used to represent an amplitude value of the sine wave is referred to as the amplitude resolution. The naming convention is based on the principle that the stored samples correspond to amplitude values of a sine wave while the number of samples corresponds to the number of phase points. It is

important to note that since the Sine LUT has finite phase and amplitude, its output values are not an exact reproduction of an ideal sinusoid. In fact, there is loss of precision in terms of reproducing an ideal sinusoid. The loss of precision has a direct impact on the spectral purity of the Sine LUT. Therefore, the objective is to use a LUT with considerable phase and amplitude resolution in order to minimize the spurious emissions due to quantization. As previously mentioned, phase resolution translates to the number of samples used to represent a sine wave. If the phase resolution is small, this translates to generating a sine wave with very few samples. On the spectrum, this would introduce more quantization error. Furthermore, the amplitude resolution translates to the number of bits used to represent a sample of the sine wave. If the amplitude resolution is small, the output values are not an exact reproduction of an ideal sinusoid. However, increasing both the phase and amplitude resolution increases the amount of resources required to implement the Sine LUT. Since the GWPP architecture generates 16 samples every clock cycle, this translates to the instantiation of 16 LUTs. Hence, the effects increasing the resolution has on the FPGA resources as well the spectral purity will be examined in Chapter V.

This concludes the discussion of the C/A 1 Sample Generator design element. The theory, design, and implementation of each subcomponent used to build this design element was presented. Furthermore, it was introduced and reinforced throughout this section that this design element is instantiated 16 times to build the C/A 16 Sample Generator design element used to generate 16 samples every clock cycle. The requirement to generate 16 samples every clock cycle was presented in the high level overview of the hardware implementation. Additionally, throughout this section it was shown how each subcomponent in the C/A 1 Sample Generator design element established the software-defined capability built into the GWPP architecture. Now that the hardware implementation has been presented, the discussion will transition

to the methodology used to setup the GWPP architecture for evaluation.

3.3 Experiment Setup

3.3.1 Experiment Setup Overview.

This section presents the methodology involved in verifying the RTL components that build the hardware infrastructure of this research along with the experiments used to set up the testing environment. The verification of the RTL components is presented first in Section 3.3.2. Afterwards, the discussion proceeds to present the methodology involved in creating the testing environment. To accurately analyze the platform, it is crucial to create a stable, reproducible testing environment that will not skew data collection. Finally, this section concludes with the description of the final test environment used to analyze the performance metrics of the GWPP architecture as described in Chapter IV.

3.3.2 Verification of the Register Transfer Level Models.

The UDBs implemented in RTL are verified in simulation software to test functionality in both simulation and on the FPGA. There are two different software tools used in this research for functional verification: ModelSim[®] and Xilinx ChipScope[™] Pro. ModelSim[®] is a functional simulation tool that specifies input/output behavior without specifying its timing. It is used to verify functional correctness of a design entity. This technique provides a method for ensuring correctness of logic before synthesizing the design on the FPGA. As mentioned previously, Xilinx ChipScope[™] Pro is another software tool used for verification. It enables the ability to debug the FPGA after a design is programmed. It comprises a ChipScope[™] Pro Core using the CORE Generator, and ChipScope[™] Pro Analyzer. The core is inserted in the design to provide communication between the ChipScope[™] Pro Analyzer software and the

captured cores. It provides visibility into the current contents stored in the signals of interest. Like ModelSim® it is another method to verify functionality, but allows the designer to tap into the current signals inside the FPGA associated with their design. It is beneficial to visually analyze the current dataflow in the design and ensure that functional units are receiving the correct data. This provides additional insight when considering typical real-world cases where the FPGA is often loaded with erroneous values that may be stored in signals of interest. These erroneous values could possibly affect the dataflow. In the verification of the hardware design, both tools were used interchangeably. The typical verification flow consisted of first verifying functional correctness in ModelSim®. If the functionality worked in ModelSim®, but failed once synthesized onto the FPGA, ChipScope was used to pinpoint the location of the error. Often times, the use of both software tools resulted in the ability to dismiss functionality in debugging and pinpoint the error as timing related.

The explanation sequence for this subsection continues with the simulation results obtained from verifying the RTL components of this architecture. Although each UDB was verified, the only verification shown in this section is that of the BPSK modulation, which is shown through the verification of the *GPS L1 C/A 1 Sample Generator* block. In turn, this verifies the functional correctness of the Carrier NCO, Code NCO, Code Generator, Chip Transition Detection, and Sine LUT. In addition, since the *GPS L1 C/A 16 Sample Generator* comprises 16 instantiations of the *GPS L1 C/A 1 Sample Generator*, it also verifies correctness for the 16 sample generator as well. The verification results are shown through waveforms captured in ModelSim® waveform viewer. It is important to note that some of the parameters used to verify the functional correctness of the BPSK modulation are only used to show proof of concept. That is, these parameters are not necessarily the ones used to generate the GPS L1 C/A signal. The reason for this method is generalized in the following

discussion. The GWPP architecture is designed to generate samples at a high data rate, however, the ability to show proof of concept through the waveform viewer at this data rate is nonintuitive. The samples are generated 16 times every clock cycle at a clock rate of 2.4 GHz. Trying to show this concept through the waveform viewer will only create further confusion. For that reason, different parameters are used to demonstrate both functionality and proof of concept.

The ***GPS L1 C/A 1 Sample Generator*** proof of concept is verified by showing the ability to generate a BPSK modulated signal. This is shown in Figure 29 specifically in the red rectangles. Essentially, the red rectangles identify the portion in the carrier generation that undergo a 180° phase shift as dictated by BPSK modulation. In Figure 29, the RTL is designed to BPSK modulate a RF carrier with the PRN 1 C/A-code. Figure 29 shows three overflow assertions (blue circles) in the waveform window. Each overflow represents the transition to the next chip. Since the code phase is set to 0.0, the first three chips after three subsequent overflows are '110', which correspond to the first three C/A chips of PRN 1 (purple circles). Additionally, Figure 25 shows both the half wave implementation as well as the BPSK modulated signal. This is shown by the two phase shifts in the waveform window.

The verification of the BPSK modulation is sufficient for showing correct functionality of the hardware implementation. Verification of the other RTL components can be found in Appendix F of this thesis. Next, the methodology involved in setting up the test environment is presented.

3.3.3 Test Environment Setup.

This section describes the experiments performed to create an accurate, reproducible testing environment for the GWPP architecture. Because this platform is utilized for GPS research and eventually advanced signals research, the quality of

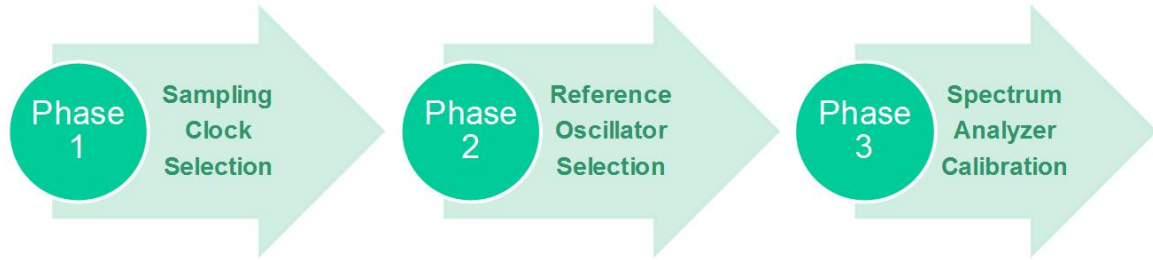


Figure 30. Test Environment Setup Flow. Essentially, three phases were performed before analyzing the performance metrics of the GWPP architecture.

3.3.3.1 Phase 1: Sampling Clock Selection.

In any signal generation technique, the quality of the signal depends on the quality of the sampling clock, which is described in terms of phase noise and jitter. Phase noise is the frequency domain representation of short- or long-term fluctuations in the phase of a waveform. In the time domain, this representation is referred to as jitter. Phase noise surrounds the carrier and has detrimental effects in DSP applications. In DDS systems phase noise and jitter cause the DAC to be clocked at non-uniform intervals. This results in the spreading of line spectra (spurs) at the DDS output frequency. During this research, there were several sampling clock sources available for to choose from. However, with consideration in the detrimental effects imposed by phase noise and jitter, it is imperative to select the sampling clock source with the best phase noise performance. It is also important to note that phase noise performance is characterized in two ways: close-in phase noise, which is close to the fundamental signal, and far-out phase noise, which is opposite of that. Close-in and far-out phase noise performance is a function of the short and long term stability of the oscillator as well as the drift of the oscillator. Some oscillators have very low close-in phase noise, but terrible far-out phase noise since various effects such as aging,

gravity, temperature, and shock/vibrations cause the oscillator to drift over the long term. On the other hand, other oscillators are very stable over the long term, but terrible close-in phase noise. In this research, the first release of the architecture is primarily focused on demonstrating proof of concept. Therefore, the platform will only be generating signals for a short time duration to collect data. For this reason, the primary focus was on finding a sampling clock with better close-in phase noise performance. Eventually, once the platform is fully developed, it will be utilized for extended periods of time and in that case it would be ideal to have a sampling clock with excellent long term stability, which results in better far-out phase noise performance.

With the combination of lab equipment at Air Force Institute of Technology (AFIT) such as the signal generator and the hardware on the FMC230, such as the DCT, there are several sampling clock sources available to choose from. The deciding factor in this matter is the phase noise performance associated with each source. This experiment consisted of performing a comparative analysis to highlight the advantages and disadvantages of each source. Before further details regarding results are presented, it is worth providing a brief introduction to each source. The first source in the comparison is the DCT enclosed in the FMC230 PCB. Recall that the FMC230 is the device that also contains the high-speed DAC utilized in this architecture. In normal configuration, the DCT provides the sampling clock to the DAC. The sampling clock source of the DCT is a PLL synthesizer, which is driven by an internal VCO. The internal VCO is phase locked to an on-board 30.72 MHz Temperature-Compensated Crystal Oscillator (TCXO). The second clock source in comparison is the RF output generated using the Keysight N5182B Vector Signal Generator. The basic configuration is to route the RF output of the signal generator to the DCT and then disable the PLL. This configuration enables the ability

to use the signal generator as an external clock input to the DCT. Consequently, the external clock input serves as the sampling clock instead of the internal VCO. The external clock input is routed through the DCT and then supplied to the DAC. Then, it effectively serves as the sampling clock for the DAC. Although this signal generator has a frequency dynamic range from 9 KiloHertz (kHz) to 6 GHz, only a subset of this dynamic range, namely 0 to 2400 MHz, is allowed at the external clock input of the DCT. The disadvantage is that the internal VCO can generate a sampling clock up to 2650 MHz, whereas the external clock input is limited to 2400 MHz. However, 2400 MHz is sufficient for this architecture. Furthermore, in order to do a comparative analysis of the two different methods, experiments quantifying the performance of each sampling clock must be performed. The experiments performed are discussed in Chapter IV.

3.3.3.2 Phase 2: 10 MHz External Reference Oscillator Selection.

The objective of this phase was to improve the phase noise performance of the sampling clock by phase locking it to an external reference oscillator with better phase noise performance. As mentioned previously, the primary focus is on achieving the best close-in phase noise for the current release of the GWPP architecture. This same concept applies to the selection of the reference oscillator to use for this platform. During the research effort, two different 10 MHz external reference oscillators were available: TRIGR front-end and MicroSemi SyncServer. The design decision to use a 10 MHz reference oscillator came from the technical proposal for this research. Before moving onto the discussion of phase 3, it is worth reinforcing the importance of the reference oscillator. The reference oscillator (clock) is arguably the most important component of a GPS receiver and transmitter. In a GPS receiver, this is true because the reference oscillator has the most direct impact on range measurements

since all other frequency synthesizers and the sampling clock are phase-locked to it. Therefore, the frequency synthesizers and sampling clock mirrors the performance of the reference oscillator. Likewise in a GPS transmitter, the sampling clock and frequency synthesizers are also phase-locked to it. Therefore, the quality of the transmitted signals mirrors the performance of the reference oscillator. Furthermore, the performance of the transmitted signals affects the acquisition/tracking algorithms of the GPS receiver. Hence, the reference oscillator is one of the most important components of the GWPP architecture. To select the best reference oscillator for this research effort, a comparative analysis between the TRIGR front-end and SyncServer was performed. The results are presented in Chapter IV.

3.3.3.3 Phase 3: Spectrum Analyzer Calibration.

The last phase in this test environment setup methodology is to calibrate the spectrum analyzer since the performance metrics for this platform consists of analyzing the spectral performance. The spectrum analyzer has many different configurations to accommodate for a multitude of different waveforms. In addition, it includes built-in functions to alter the characteristics of the signal shown on the spectrum analyzer. Therefore, it is important to calibrate the spectrum analyzer to ensure accurate data collection.

The calibration of the spectrum analyzer requires the setting of certain parameters to be accurate. These parameters are known as the resolution bandwidth and video bandwidth. These parameters change according to the span. Resolution bandwidth is referred to as the eyes of the spectrum since it is used to resolve signals close together. Video bandwidth is used to “smooth” the display by removing noise from the envelope. Both settings directly impact the time it takes the spectrum analyzer to make a measurement. The spectrum analyzer can work in two different modes. These

modes are referred to as swept-tuned and Fast Fourier Transform (FFT). Swept-tuned mode relies on the principle of superheterodyning to down-convert a portion of the input signal to the center frequency of a narrow band-pass filter. This filter is known as the resolution bandwidth filter. As the name implies, this mode essentially sweeps the receiver's center frequency through a dynamic range of frequencies. Since this method performs a sweep it is significantly slower than the FFT method. The sweep time is a function of the span and the resolution bandwidth. In FFT mode, the analyzer computes a time-sequence of spectral density estimates of the signal. It basically digitizes the signal using digital sampling and then performs the mathematics required to convert it to the frequency domain. It is effectively faster than the swept-tuned method. It is important to note that the resolution bandwidth and the span play a key part in what mode the analyzer will operate in. This could ultimately result in longer or shorter measurement times.

The goal of the calibration is to find the correct resolution bandwidth in the spectrum analyzer for a given span to accurately capture the trace points in the spectrum analyzer. The trace points refer to the amplitude value stored in each frequency bin in the spectrum. The frequency bin is calculated by dividing the span of the spectrum by the number of trace points. In this research effort, the spectrum analyzer was configured to display the maximum amount of trace points, which is 100,001 points. There are several key terms that affect how the spectrum analyzer samples the RF input. The first term, which was already mentioned, is the resolution bandwidth. The resolution bandwidth is essentially a bandpass filter that is used as a window for detecting signals. It is imperative that the right resolution bandwidth is chosen to ensure that the spectrum analyzer does not oversample or undersample the signals detected in each frequency bin. The term resolution bandwidth, whether it is expressed as a single sideband or double sideband quantity, is not clearly defined in the

literature. Therefore, the calibration is required to determine the correct definition as well as the value of the resolution bandwidth for a specific span of interest.

The calibration was performed using a USB power sensor. The USB power sensor displays the average power of the spectrum. A cavity filter with a center frequency equal to 1575.42 MHz and a 30 MHz wide bandwidth is used to limit the power going to the USB power sensor. Essentially the power read by USB power sensor is the power centered around the carrier in a 30 MHz bandwidth. The idea is to choose a span and a resolution bandwidth configuration with 100,000 trace points. To verify if the resolution bandwidth for the specific span is correct, the 100,000 trace points are exported to MATLAB[®], converted to a linear scale, and then integrated over the range. If the integrated power matches the power sensor, this verifies that the resolution bandwidth did not oversample or undersample the frequency bins at each trace point. In this research effort, two spans are used in the analysis of the GWPP architecture: wideband (100 MHz) and narrowband (200 kHz). 100 MHz was chosen since it simplifies the bucket widths for calculations and provides a wide span to observe the spectral purity in the GPS 20 MHz bandwidth and beyond. GPS receivers process data in 1 ms blocks. The highest frequency content able to be processed in this block is 500 Hz. Therefore, it is important to analyze the platform in a 500 Hz span centered around the carrier. However, with the configuration set to 100,000 points, this would mean that each frequency bin is five-thousandths of a Hertz. The lowest resolution bandwidth allowed in the spectrum analyzer is 1 Hz, therefore the spectrum analyzer would be oversampling each bin. As a result, the trace points would be represented with incorrect amplitudes. For this reason, 200 kHz is used as the span. The calibration results from this experiment are presented next.

3.3.4 Final Experiment Setup.

The spectrum analyzer calibration settings for the test environment setup are presented in Table 6. These settings are used to calibrate the spectrum analyzer in order to accurately capture the data from the GWPP architecture. The data coming from the GWPP architecture are CW samples generated using DDS. The CW generated from the GWPP architecture is the GPS L1 carrier. Although the goal of this research is to synthesize the GPS L1 C/A carrier, the generation of a CW will be used as the test signal to analyze the performance metrics. The objective of the analysis is to assess the signal quality of the GPS L1 carrier generated using the GWPP architecture. The quality of the generated L1 carrier directly impacts the quality of the BPSK modulated GPS L1 C/A signal. For this reason, it is important to assess the quality of the GPS L1 carrier. In this research effort, quality of the GPS L1 carrier is assessed in the spectral domain. The primary focus is on the spectral purity in a wideband and narrowband centered around the GPS L1 carrier. Once again, the calibration settings to accurately capture the spectral output of the GWPP architecture in each span are shown in Table 6. After the calibration settings are complete for a specific span, the data is ready to be saved to a trace file. The trace file is populated with 100,000 trace points. Each trace point has a frequency and its associated amplitude. This file is read and processed in MATLAB[®]. Several

Table 6. Spectrum Analyzer Calibration Settings

	100 MHz Span	200 kHz Span
Mode	Swept	FFT
Resolution Bandwidth	680 Hz	1 Hz
Video Bandwidth	680 Hz	1 Hz
Trace Detector	Average	Average
Average Type	Power	Power
Average Count	2	100
Envelope Type	Average	Average
Trace Points	100,001	100,001

scripts in MATLAB[®] evaluate the trace data with respect to the performance metrics outlined for this research. The performance metrics are outlined in Chapter IV.

IV. Results and Verification

THIS chapter presents the performance metrics used to analyze the architecture. In addition, it covers a high level overview of the experiments performed to quantify the performance metrics. Afterwards, the results as well as discussions are presented for each experiment. The results presented in this chapter are obtained from testing the platform using the experiment setup described in Chapter III. Before the results used to assess the performance metrics of the GWPP architecture are presented, it is first important to present the results obtained in setting up the test environment.

4.1 Test Environment Setup Results

The test environment setup covered the following two tasks: determining the cleanest sampling clock for the platform and calibrating the lab equipment to accurately collect data to quantify the performance metrics. The spectrum analyzer calibration settings were explained in the end of Chapter III. Therefore, this section presents the results found in determining the cleanest sampling clock to use for the GWPP platform. Recall that the cleanest sampling clock experiment consisted of two phases. Phase 1 experimented with the two different sampling clock options available for this research effort: internal VCO on-board the FMC230 and the external sampling clock input provided by the signal generator. Phase 2 experimented with the two different 10 MHz reference oscillators available for this research effort: TRIGR front-end and SyncServer. The results obtained during the experimentation of phase 1 are presented first followed by the results from phase 2.

4.1.1 Sampling Clock Selection Results.

For the sampling clock selection process three different experiments were performed with the internal VCO and PLL configuration. Recall that in a PLL synthesizer, the VCO is connected in a feedback configuration with a loop filter and a phase detector. The internal VCO on the FMC230 has a frequency dynamic range from 2300 MHz to 2650 MHz. Hence, it can be tuned to generate any frequency in this range. This experiment consisted of leaving the VCO unaltered and changing only the reference inputs to the PLL. For this setup, three different reference inputs were tested. The first was the on-chip 30.72 TCXO reference oscillator. The second and third inputs were 30.72 MHz and 10 MHz external reference inputs provided from the signal generator. In each experiment, the GWPP architecture was used to synthesize a CW with a carrier frequency of 1575.42 MHz. A bandpass filter centered around 1575.42 MHz with a 30 MHz bandwidth was applied to the CW. The sampling clock used was the tuned VCO with different external reference inputs to the PLL. Two performance metrics were used to compare the three different inputs: close-in phase noise to the carrier and Spurious Free Dynamic Range (SFDR). Both metrics were analyzed using the spectrum analyzer. For close-in phase noise, the spectrum analyzer was configured as follows: resolution and video bandwidth set to 1 kHz and a span of 32 MHz. The study of phase noise was performed using markers on the spectrum analyzer. Essentially, the power of the carrier was compared to the power in a 1 Hz bandwidth at different frequency offsets from the carrier. Since both the spectrum analyzer and the signal generator have similar phase noise performance (-127 dBc/Hz at 20 kHz offset), it was difficult to differentiate whether the phase noise performance was a result of the signal generator or the spectrum analyzer. Therefore, another metric called SFDR was used to evaluate the performance of each sampling clock. SFDR is calculated as the power of the highest spur in the span relative to the

power of the carrier. The spectrum analyzer for this metric was configured as follows: resolution and video bandwidths set to 300 kHz and a span of 32 MHz. The results are summarized in Table 7.

Table 7. Internal VCO Sampling Clock with 3 Different Reference Oscillators

Reference Oscillator	Spurious Free Dynamic Range (dBc)
30.72 MHz on-chip	43.9
30.72 MHz external	44.1
10 MHz external	30.0

The second experiment consisted of the same setup used to analyze the internal VCO. Instead of the VCO, the RF output of the signal generator was used as the sampling clock source. The RF output of the signal generator feeds into the external clock input port of the FMC230 PCB. In order to use this source as the sampling clock, the firmware code was altered to disable the PLL and enable the external reference clock input. It is important to note that only of these sources can be active at once. To set up this experiment, the signal generator was configured to generate a 2400 MHz sinusoidal output with an amplitude of 0 dBm. These values are based on the recommended values for the external clock input detailed in the FMC230 User Guide [17]. Again, phase noise observations were made using markers on the spectrum analyzer. The SFDR calculated for this configuration was 60 dBc.

In conclusion, it is clear that using the external clock input provided by the signal generator as the sampling clock yields better spectral performance compared to the internal VCO. In other words, using the signal generator as the sampling clock source results in greater attenuation of spurs and better phase noise performance. The reason for this increased performance is due to the fact that the signal generator generates a very clean and stable clock due to its internal crystal oscillator. In addition, the internal crystal oscillator enclosed in the signal generator has better phase noise performance compared to the on-board TCXO. The performance of the internal

VCO is affected by both the PLL configuration and the on-chip 30.72 MHz TCXO. The placement of the on-chip 30.72 MHz TCXO in the PCB forces the TCXO to be surrounded by various digital components that also produce noise. As a result, this digital noise is added to the phase noise of the TCXO, which detrimentally lowers the spectral performance of the VCO. In addition, replacing the on-chip TCXO reference oscillator with an external oscillator did not produce any noticeable advantages. This phenomenon is attributed to the PLL configuration. To use the internal VCO, the PLL is required, whereas to use the external clock input, the PLL can be bypassed. PLLs, due to the nature of the circuit, generate a considerable amount of phase noise compared to a signal generator (which still has phase noise). The reason for this is due to the PLL circuits comprising several noise sources, which each create their own fair share of phase noise. A PLL usually consists of an oscillator, phase detector, charge pump, and loop filter. The crystal oscillator phase noise affects performance very close to the carrier while the phase detector noise affects the loop bandwidth. Thereafter, the VCO noise becomes the dominant noise factor. In addition, the input frequency to the phase detector plays a role in the generation of noise. Therefore, by bypassing the PLL using the external clock input, the components that additionally contribute to phase noise are bypassed. From this analysis, it is clear that the external reference clock input yields better spectral performance and is therefore used as the sampling clock for this platform.

4.1.2 10 MHz Reference Oscillator Selection Results.

This section describes the results obtained in improving the phase noise performance of the sampling clock generated by the signal generator by means of phase locking it to an external reference oscillator. The signal generator used in this research effort also employs DDS; therefore, its phase noise performance can still be

improved by locking the signal generator to an external reference oscillator. Locking the signal generator to an external reference oscillator creates a PLO. PLOs provide several benefits to the architecture. In GPS receivers, the reference oscillator is arguably the most important component of the receiver because it has the most direct impact on ranging measurements. All of the frequency synthesizers and sampling clocks mirror its performance since they are phase-locked to it. The phase noise of the oscillator directly impacts the phase noise of the frequency synthesizer. Therefore, it is imperative to phase lock the signal generator to an external oscillator with excellent phase noise performance and stability. The technical proposal for this research required the frequency synthesis to be performed using PLOs with low close-in phase noise and low spurious content. Furthermore, the technical proposal called for a 10 MHz reference oscillator. For this research, there were two 10 MHz reference oscillators available to choose from at AFIT. One was a 10 MHz Oven-Controlled Crystal Oscillator (OCXO) enclosed in the 4-Channel GNSS RF TRIGR front-end developed by Ohio University. The other was a 10 MHz rubidium oscillator enclosed in the SyncServer S560 provided by Microsemi. Thus, this experiment consisted of performing a comparative analysis with respect to the spectral performance of each device.

As mentioned previously, the quality of an oscillator is defined by its phase noise and jitter characteristics. There are several different methods explored in research to measure phase noise and jitter. However, in order to perform the measurement, the phase noise performance of the tool that will be used to measure the phase noise of a device must be at least 10 dB greater. This is generalized in the following example. If two devices have the same phase noise performance, it is impossible to distinguish whether the phase noise measurement is a result of the testing tool or the device under test. In addition, if the device under test has better phase noise

performance than the tool that is used for verification, the phase noise measurement will be that of the tool and not the device under test. Therefore, measurement of the phase noise of the different oscillators requires a spectrum analyzer with an oscillator that has 10 dB greater phase noise performance than the oscillator under test. For the scope of this research, the oscillators under test are equipped with one of the top reference oscillators. In the market for oscillators, there is a steep difference in cost between the best oscillator and the second best oscillator. However, this is not a noticeable improvement in performance. For this reason, phase noise is not necessarily measured. Instead, it is satisfiable to claim that the architecture uses a reference oscillator with close to the best phase noise performance on the market. Even though it is impossible to compare the phase noise of each oscillator to determine which one is better, another technique was used. This technique consisted of connecting the 10 MHz reference oscillator to the spectrum analyzer and observing the noise in a 1 Hz bandwidth at different offsets from the carrier. This will partially show which device has better close-in performance. The results are shown in Table 8. Carrier power is specified in dBm and Noise Power in dBc/Hz.

Table 8. 10 MHz TRIGR Oscillator vs 10 MHz SyncServer Oscillator

Reference Oscillator	Span	Offset	Carrier Power	Noise Power
SyncServer	15 kHz	1 kHz	-11.8	-71.2
	15 kHz	5 kHz	-11.8	-110.3
	30 kHz	10 kHz	-14.8	-110.5
	70 kHz	20 kHz	-18.7	-108.1
	150 kHz	50 kHz	-21.8	-107.7
TRIGR	15 kHz	1 kHz	-11.8	-108.7
	15 kHz	5 kHz	-11.8	-111.2
	30 kHz	10 kHz	-14.9	-110.7
	70 kHz	20 kHz	-18.5	-108.0
	150 kHz	50 kHz	-21.9	-107.1

These results show that the TRIGR has better close-in phase noise performance compared to the rubidium oscillator. However, for additional verification, the results

obtained through experimentation were compared with the technical manual for the SyncServer [22]. Reference [22] shows that the close in phase noise at 1 Hz for the SyncServer is approximately -72 dBc/Hz and at 10 Hz it is -90 dBc/Hz. Therefore, this verifies that the SyncServer has worst close-in phase noise compared to the TRIGR. For GWPP, the primary focus using a 10 MHz reference oscillator with the lowest close-in phase noise possible. Therefore, the TRIGR is chosen as the 10 MHz reference oscillator.

4.2 GWPP Performance Metrics

From the analysis of the DDS architecture, there are several sources of error. The error sources are shown in Figure 31. The first source of error is due to the phase truncation of the accumulator output, e_{PR} . The second source of error is due to the finite resolution of the amplitude values stored in the LUT, e_{AR} . Finally, the third source of error is due to the nonlinearities in the DAC, e_{DAC} . The first two errors are a result of the LUT implementation and are the focus of this analysis. Therefore, only changing the LUT configuration while leaving the DAC constant allows the ability to observe the spectral performance due to the LUT. These two errors sources are described in further detail next.

Phase truncation is a technique used in DDS to conserve the amount of resources required to implement the sine LUT. This performance metric consists of observing the relationship between the spurious effects and phase truncation. Typically in DDS, it is ideal to have a large accumulator ($N \geq 32$) to exploit the fine frequency resolution advantages of DDS. Ideally, the N -bit phase output from the accumulator would index the LUT; however, with very large accumulators, this would require enormous amounts of memory and is therefore impractical. For this reason, the output of the phase accumulator is truncated to a finite number of phase bits. Since the real value

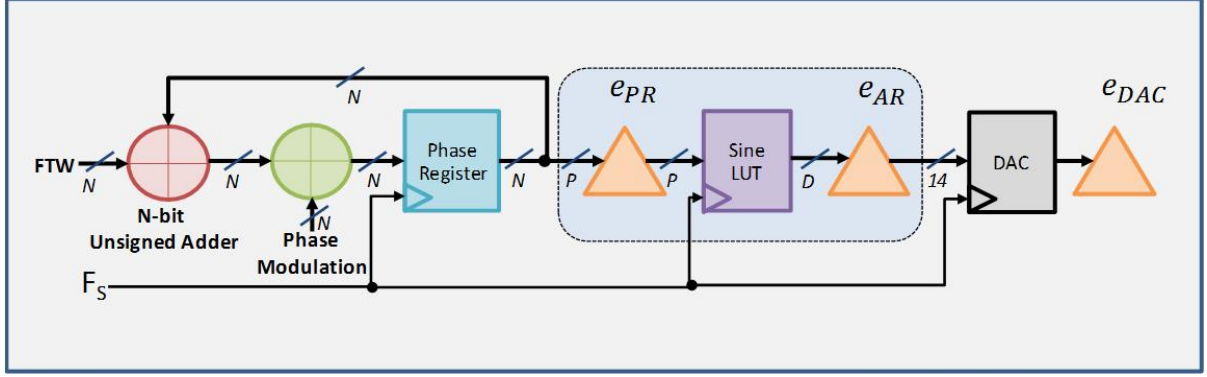


Figure 31. Error Sources in DDR Systems

of the phase is represented by a finite word, there is always a phase quantization error that shows up as spurious emissions in the output spectrum. Recall that in DDS, a finite number of MSBs are retained to index the LUT while the remaining Least Significant Bits (LSBs) are truncated. The term Phase Resolution (PR) will be used to quantify the number of MSBs used in the experiment. In addition, the number of phase points the PR constitutes is 2^{PR} . For example, if PR equals 4, then this would result in 16 phase points.

Finite amplitude resolution is another technique used in DDS to conserve the amount of resources required to implement the sine LUT. It essentially dictates the number of bits used to represent the amplitude of one phase point. This performance metric consists of observing the relationship between the spurious effects and finite amplitude resolution. Typically in DDS, it is ideal to have the amplitude resolution match the DAC resolution. The DAC resolution also plays a role in the quantization errors seen in the spectrum. The DAC resolution determines the amount of quantization levels used to map the digital waveform to analog. Therefore, it is ideal to have a DAC with a high resolution to mitigate quantization errors produced by the DAC. This however can put a strain on the LUT implementation. In this thesis, a DAC with 14-bit resolution is used. Therefore, depending on the phase resolution, the implementation could result in a considerable amount of resources. The term

Amplitude Resolution (AR) will be used to quantify the number of bits used to represent one phase point in the LUT. For example, if AR equals 5, then the samples stored in the LUT have 5-bit accuracy. Therefore, using signed data format would produce maximum and minimum amplitude values at 8 and -8, respectively.

The last performance metric used to analyze the GWPP architecture is referred to as FPGA resource utilization. This metric quantifies the amount of hardware required to realize the desired functionality. Recall that in this architecture, the phase-to-amplitude conversion is designed to exploit the half-wave symmetry property of sinusoids. In addition, the design of this logic constitutes a LUT in distributed ROM and additional logic gates. The benefit of the specific logic designed in this architecture is shown in the following example. Consider the case for $PR = 5$ and $AR = 6$. In a normal architecture without exploiting the symmetrical properties of sine waves, one LUT would constitute $2^5 \times 6$, or 192 bits of memory. Since this architecture requires 16 LUTs to exploit the DAC data rate, now this implementation consumes 3,072 bits of memory. Since the LUT is implemented in distributed ROM, the FPGA synthesis tools can only use the slices that can be configured as distributed ROM, which are SLICEMs. Recall that 25%-50% of all CLBs have SLICEMs. Therefore, SLICEMs are practically scarce compared to the rest of the elements inside the CLBs. However, in this architecture, the half wave symmetrical property of sine waves is utilized to optimize resource utilization. Consider the same case presented before. The half wave symmetry and additional logic is designed to store half the number of phase points in distributed ROM with the amplitude resolution decreased by one. Hence, the 3,072 bits of memory dedicated to distributed ROM now reduces to 1,280 bits. The 1,280 bits constitute half the sine wave with one less bit as the amplitude resolution. The original amplitude resolution as well as the second half of the phase points are generated using other elements inside the CLBs that are more abundant com-

pared to SLICEMs. This implementation reduces the number of bits implemented in SLICEMs by at least half. The number of bits chosen for the PR and AR impact both the spectral purity of the GWPP architecture as well as the resource utilization in the FPGA. Therefore, the limited resources in the FPGA presents a trade-off between spectral signal quality and FPGA resource utilization.

4.3 Overview of Experiments

The performance metrics used to analyze the GWPP architecture include spurious performance due to phase truncation, spurious performance due to finite amplitude resolution, and spurious performance with respect to FPGA resource utilization. The goal of the first metric is to observe the spurious performance of different phase truncation values for the FTW value used in this platform. Recall that since this architecture is configured to generate the GPS L1 C/A signal, the FTW represents the N -bit digital word that generates the GPS L1 center frequency given an N -bit accumulator and a 2400 MHz sampling clock. Likewise, the goal of the second metric is to observe the spurious performance of different amplitude resolutions. Finally, the goal of the last metric is to observe the number of slices and CLBs consumed by different LUT configurations.

One of the metrics used to evaluate the spur performance consists of comparing the total power contained in the carrier to the power lost to spurs and harmonics. This metric is called Carrier to Spurious Emission (CSE) and provides a clear way to observe the spectral difference in each LUT configuration. It is calculated in the linear scale by

$$Power_{CSE} = 10 \log_{10} \left(\frac{Power_{Carrier}}{Power_{Lost}} \right) \quad (4.3.1)$$

where $Power_{CSE}$ refers to the ratio of power contained in the carrier compared to the power contained in the spurious emissions; $Power_{Carrier}$ refers to the power contained

in the fundamental carrier and $Power_{Lost}$ is the power contributed from the spurs and harmonics. It is calculated in the linear scale by

$$Power_{Lost} = Power_{Spurs} + Power_{Harmonics} \quad (4.3.2)$$

where $Power_{Spurs}$ is the calculated power in all of the spurs in the span; and $Power_{Harmonics}$ is the calculated power in all of the harmonics.

In the logarithmic scale, Equation (4.3.1) is defined as

$$Power_{CSE} = Power_{Carrier} - Power_{Lost} \quad (4.3.3)$$

The next metric used to analyze each LUT configuration is SFDR. The definition of this metric varies depending on the source. In this thesis, SFDR characterizes the ratio between the fundamental signal and the highest spurious content in the spectrum. There are two methods to state the SFDR figure. In one method the harmonic distortion is ignored and SFDR is only calculated for all spurious parts. The other method to calculate SFDR includes the harmonics. In this thesis, both types of distortions are considered when calculating SFDR and can be calculated by

$$SFDR = 10 \log_{10} \left(\frac{Power_S}{Power_H} \right) \quad (4.3.4)$$

where $Power_S$ is the power of the fundamental signal in a linear scale such as milliwatts; and $Power_H$ is the power of the highest spur or harmonic in a linear scale. However, if working in the logarithmic scale, then SFDR is calculated by

$$SFDR = Power_S - Power_H \quad (4.3.5)$$

where $Power_S$ is the power of the fundamental signal in dBm and $Power_H$ is the

power of the highest spur or harmonic in dBm.

4.4 Spur Performance Due To Phase Truncation: 5-bit Amplitude Resolution

This experiment is one of two experiments performed to analyze the spurious performance due to phase truncation. To measure the effect of phase truncation, the amplitude resolution is kept constant at 5 bits. A 5-bit amplitude resolution means the amplitude of the corresponding phase points are integers between -8 and 8. Table 9 illustrates the two test cases analyzed in this experiment. The goal is to compare

Table 9. Spur Performance Due To Phase Truncation: 5-bit Amplitude Resolution

Test Case	AR (bits)	PR (bits)	Phase Points
1	5	5	32
2	5	6	64

the spurious performance using two different size LUTs: one with 32 phase points and the other with 64 phase points. It is important to note that the amplitude resolution for this experiment is kept constant. In addition, the term 32 phase point LUT, until noted otherwise, describes a DDS system that uses 32 phase points to represent one period of the output sine wave. The hardware logic used to implement a LUT with 5- or 6-bit phase resolution and 5-bit amplitude resolution can be disregarded for now. Therefore, this experiment primarily focuses on the spectral performance using either 32 phase points or 64 phase points.

The sine wave representation of test cases 1 and 2 are shown in Figure 32. Three separate traces are shown for each LUT. The blue trace (smooth sine wave) represents an exact sinusoid with 64-bit precision. Hence, the sine wave display is smooth compared to the quantized, stair-step output of the sine LUT, which is represented by the magenta trace. The red trace is the difference between the blue and magenta

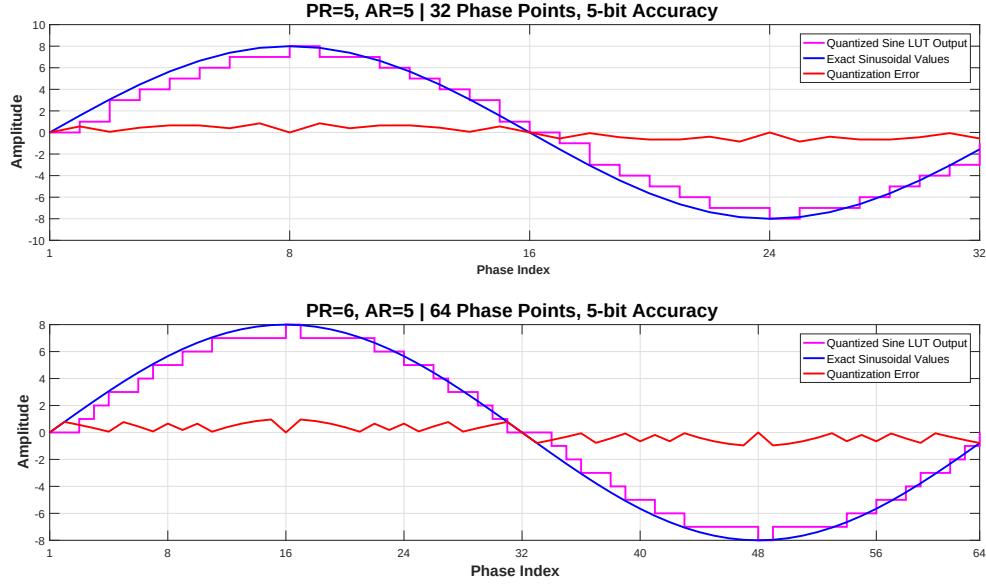


Figure 32. Sine Wave Quantization Effects Due To 5-bit and 6-bit Phase Truncation

traces. Hence, the red trace is the quantization error of the LUT. The red trace also provides insight to the quantization error present at each phase index. However, it is also important to note that the 5-bit amplitude values are left shifted and resized to 14 bits to match the resolution of the DAC. The left shift operator amplifies the amplitude value of the sample. The spectral performance resulting from the two different LUTs is analyzed in both wideband and narrowband spans. Recall that the wideband constitutes a span of 100 MHz centered around the GPS L1 carrier. The narrowband constitutes a span of 200 kHz centered around the GPS L1 C/A carrier.

The analysis for each LUT configuration consists of analyzing the performance metrics in both the wideband and narrowband spans. In each span Equations (4.3.1) and (4.3.5) are used to characterize the performance of the LUT under test. In each explanation, the wideband spurious performance results are presented first followed by the narrowband results.

4.4.1 Wideband Spurious Performance Due To Phase Truncation: 5-bit Amplitude Resolution.

The spectrum outputs of the two different LUTs in the wideband span are shown in Figure 33. In addition to displaying the spectrum outputs of both LUTs, Figure 33 also displays a horizontal dashed line across the spectrum referred to as the spurious emission threshold. In the wideband experiments, a threshold is used to characterize whether the trace point is a spur/harmonic or part of the noise floor. It is essentially calculated by finding the mean of the spectrum without the power contained in the carrier and adding two standard deviations. Figure 33 clearly shows that the LUT with 64 phase points yields better spectral performance. Table 10 reinforces this claim with the calculation of the performance metrics. The first performance metric presented is the comparison of the fundamental carrier power to the power lost to spurious emissions. In Figure 33, each grid block represents 10 MHz. With this in mind, it can be seen that the LUT with 32 phase points has its highest harmonic approximately 13 MHz away from the lower sideband of the fundamental carrier. The harmonic at this location is roughly -46 dBm. On the other hand, this same harmonic displayed in the spectrum of the 62 phase point LUT is severely attenuated. Hence, it shows how different phase resolution values impact the spectral performance. In

Table 10. Wideband: 5-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions

Phase Points	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
32	-15.45	-41.20	25.74
64	-15.39	-48.08	32.89

this phase truncation experiment, it can be seen from Table 10 that the LUT with 64 phase points effectively loses less power to harmonics and spurs compared to using 32 phase points. This is essentially due to more phase points resulting in less quantization. Table 11 presents the results from analyzing the SFDR of each

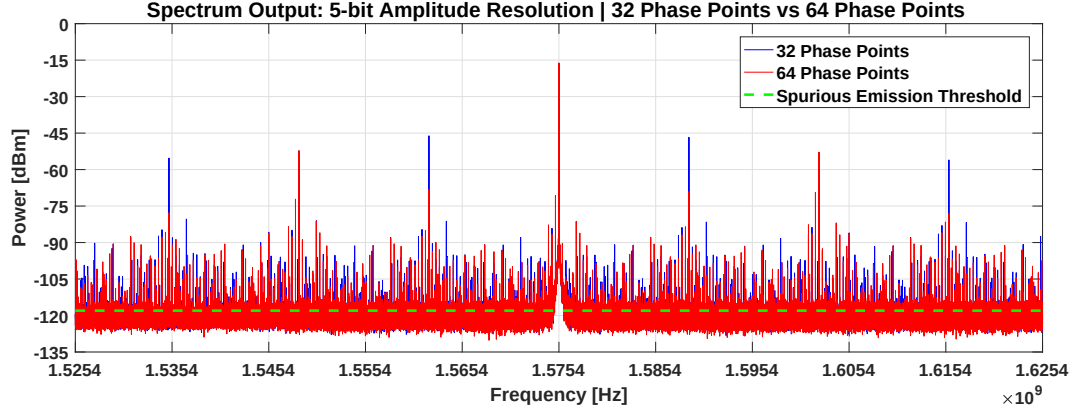


Figure 33. Wideband Spectrum Output Due To Phase Truncation: 5-bit Amplitude Resolution.

LUT configuration. The 20 MHz span is used since the bandwidth of the GPS L1 band is approximately 20 MHz. Therefore, it is important to analyze the spurious performance in this span. From the analysis of Table 11 it can be seen that the

Table 11. Wideband: 5-bit Amplitude Resolution: Spurious Free Dynamic Range

Span [MHz]	Metric	32 Phase Points	64 Phase Points
20	Carrier Power [dBm]	-15.45	-15.39
	Max Spur [dBm]	-81.27	-70.62
	SFDR [dBc]	-65.82	-55.22
100	Carrier Power [dBm]	-15.45	-15.39
	Max Spur [dBm]	-46.12	-52.26
	SFDR [dBc]	-30.66	-36.86

LUT with 32 phase points yields a better SFDR figure compared to the LUT with 64 phase points. However, the opposite is true with respect to the 100 MHz span. This behavior is also presented in references [6] and [23]. References [6] and [23] explain how the location of the spurs/harmonics depend on the grand repetition rate of the DDS system. Hence, different phase resolutions can result in similar spurs, but at different locations. Therefore, in this implementation, the LUT with 32 phase points yields a better close-in SFDR compared to the 64 phase point LUT, whereas the opposite is true for far-out SFDR. Hence, it can also be seen that both LUTs have their highest spurious content outside the 20 MHz span centered around the GPS L1

carrier.

4.4.2 Narrowband Spurious Performance Due To Phase Truncation: 5-bit Amplitude Resolution.

This experiment focuses on observing the spectral performance of both the 32 phase point and 64 phase point LUTs in a narrow span. Recall that narrowband refers to a span of 200 kHz. Therefore, it is essentially measuring the spurious performance close to the carrier. The spectrum outputs of the two different LUTs in a 200 kHz span are shown in Figure 34.

Practically, the same performance metrics analyzed in the wideband experiments are analyzed in the narrowband experiments. The only difference is that the SFDR is only calculated in a 200 kHz span centered around the fundamental carrier. The results are presented in Tables 12 and 13.

Table 12. Narrowband: 5-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions

Phase Points	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
32	-16.7392	-81.05	64.31
64	-16.7041	-81.15	64.45

Table 13. Narrowband: 5-bit Amplitude Resolution: Spurious Free Dynamic Range

Span [kHz]	Metric	32 Phase Points	64 Phase Points
200	Carrier Power [dBm]	-16.85	-16.81
	Max Spur [dBm]	-98.25	-98.98
	SFDR [dBc]	-81.40	-82.17

The results from Figure 34 and Tables 12 and 13 show that for the FTW chosen for this architecture, and a 5-bit amplitude resolution, the number of phase points does not change the spectral performance of the GWPP architecture. The power lost to harmonics and spurs as well as the SFDR for both LUTs are essentially the same. However, this theory could very well be the opposite for different FTWs and amplitude

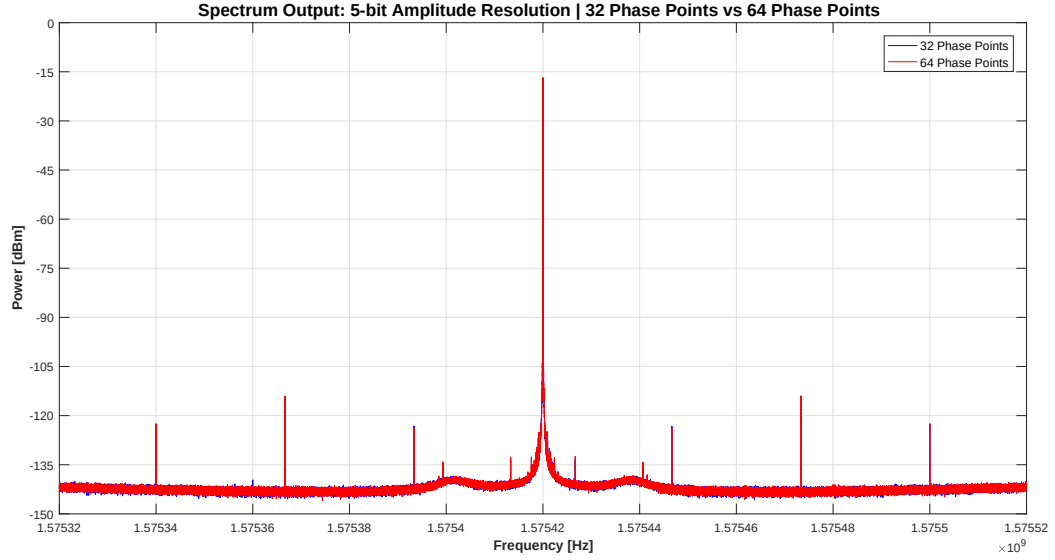


Figure 34. Narrowband Spectrum Output Due To Phase Truncation: 5-bit Amplitude Resolution.

resolution values. Furthermore, it can be seen that for either LUT configuration, the GWPP platforms achieves a SFDR figure of approximately -81 dBc. The spectral performance close to the carrier is an important performance metric for the acquisition and tracking of the carrier with a GPS receiver.

4.5 Spur Performance Due To Phase Truncation: 6-bit Amplitude Resolution

This is another experiment performed to analyze the spurious performance due to phase truncation. It is essentially an extension to the experiments performed in Section 4.4. However, the spurious performance due to phase truncation is analyzed using a 6-bit amplitude resolution. In addition, three LUT configurations will be analyzed instead of two. The first consists of 32 phase points, the second consists of 64 phase points, and the third consists of 512 phase points. Table 14 provides an overview of the three test cases analyzed in this experiment.

Table 14. Spur Performance Due To Phase Truncation: 6-bit Amplitude Resolution

Test Case	AR (bits)	PR (bits)	Phase Points
1	6	5	32
2	6	6	64
3	6	9	512

The sine wave representation of test cases 1, 2, and 3 are shown in Figure 35. Three separate traces are shown for each LUT. The blue trace (smooth sine wave) represents an exact sinusoid with 64-bit precision. Hence, the sine wave display is smooth compared to the quantized, stair-step output of the sine LUT, which is represented by the magenta trace. The red trace is the difference between the blue and magenta traces. Hence, the red trace is the quantization error of the LUT. The red trace also provides insight to the quantization error present at each phase index.

4.5.1 Wideband Spurious Performance Due To Phase Truncation: 6-bit Amplitude Resolution.

The spectrum outputs for the three different LUTs are shown in Figure 36. Figure 36 clearly shows that the LUT with 512 phase points yields the best spectral performance. The majority of the spurious emissions in the 512 phase point LUT spectrum is highly attenuated compared to the 32 and 64 phase point LUTs. Table 10 verifies this claim with the calculation of the performance metrics. Essentially, the majority of the spurs and harmonics present in the other two LUTs are severely attenuated in the 512 phase point LUT configuration. This theory is further verified through the analysis of the performance metrics calculated in Tables 15 and 16. Table 15 compares the power contained in the carrier to the power contained in the combination of all of the spurs and harmonics. Again a threshold is used to characterize which phase points are considered harmonics and spurs. Comparing the three different LUTs based on the results in Table 15 verifies that the LUT consisting of

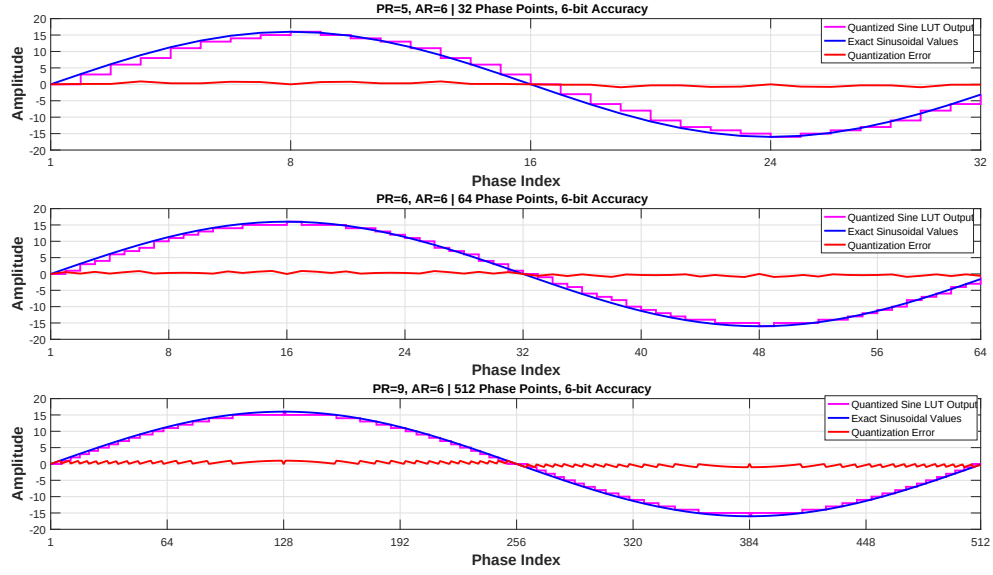


Figure 35. Sine Wave Quantization Effects Due To 5-bit, 6-bit, and 9-bit Phase Resolution

Table 15. Wideband: 6-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions

Phase Points	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
32	-15.03	-40.83	25.79
64	-15.00	-48.24	33.24
512	-15.14	-55.23	40.08

512 phase points yields the best spectral performance. In other words, less power is contained in the spurs and harmonics compared to the other implementations. This is attributed to the increased number of phase points. Also, this is attributed to having a phase resolution larger than or equal to $AR + 2$ or $AR + 3$ as noted in [23] and [6]. This configuration allows for the generation of every possible output without skipping values, which eliminates deliberate and arbitrary quantization errors.

Finally, Table 16 analyzes the SFDR as a result of each LUT implementation. In both spans, the 512 phase point LUT outperformed the 32 and 64 phase point LUTs in the calculation of SFDR. Previously, the 32 phase point LUT dominated in the close-in calculation of the SFDR figure with respect to a large LUT size. However, in this

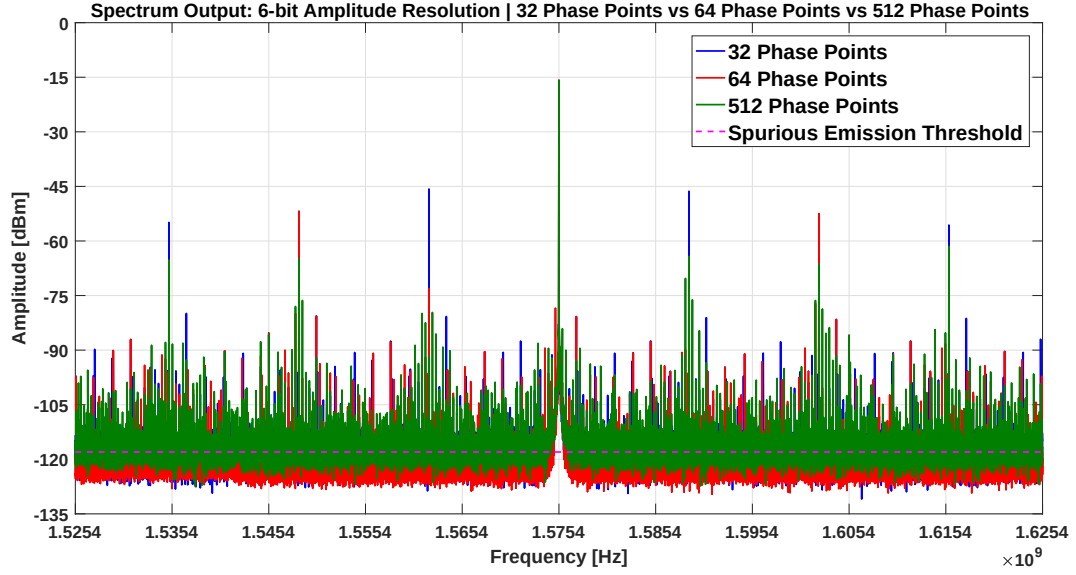


Figure 36. Wideband Spectrum Output Due To Phase Truncation: 6-bit Amplitude Resolution.

Table 16. Wideband: 6-bit Amplitude Resolution: Spurious Free Dynamic Range

Span [MHz]	Metric	32 Phase Pts.	64 Phase Pts.	512 Phase Pts.
20	Carrier Power [dBm]	-15.03	-15.00	-15.14
	Max Spur [dBm]	-80.77	-78.44	-84.11
	SFDR [dBc]	-65.73	-63.44	-68.97
100	Carrier Power [dBm]	-15.03	-15.00	-15.14
	Max Spur [dBm]	-45.71	-51.87	-61.47
	SFDR [dBc]	-30.67	-36.87	-46.32

case, the 512 phase point LUT outperformed the 32 phase-point LUT. Once again, for the chosen LUT configurations, the highest spurious content for each configuration is located outside the 20 MHz span centered around the GPS L1 carrier. In addition, the 32 phase point LUT outperformed the 64 phase point LUT in the 20 MHz span calculation of the SFDR figure. However, the 64 phase point LUT outperformed the 32 phase point LUT in the 100 MHz span.

4.5.2 Narrowband Spurious Performance Due To Phase Truncation: 6-bit Amplitude Resolution.

The narrowband spectrum outputs of the three different LUTs are shown in Figure 37. The results presented in Figure 37 and Tables 17 and 18 show that the 32, 64, and 512 phase point LUTs did not impact the spectral purity close to the carrier. The spurious performance of each LUT in the narrowband span are practically identical. Therefore, it can be said that for a LUT with 6-bit amplitude resolution and the FTW value chosen for this architecture, the spurious performance close to the carrier is not greatly impacted by phase truncation.

Table 17. Narrowband: 6-bit Amplitude Resolution: Fundamental Carrier Power vs Power Lost To Spurious Emissions

Phase Points	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
32	-16.35	-80.74	64.38
64	-16.30	-80.63	64.32
512	-16.39	-80.72	64.32

Table 18. Narrowband: 6-bit Amplitude Resolution: Spurious Free Dynamic Range

Span [kHz]	Metric	32 Phase Pts.	64 Phase Pts.	512 Phase Pts.
200	Carrier Power [dBm]	-16.46	-16.42	-16.50
	Max Spur [dBm]	-98.29	-97.46	-98.02
	SFDR [dBc]	-81.82	-81.04	-81.51

4.6 Spur Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution

This experiment is one of two experiments performed to analyze the spurious performance due to finite amplitude resolution in the LUT. Recall that finite amplitude resolution refers to the number of bits used to represent the amplitude of one phase point. To measure the effect of finite amplitude resolution, the phase resolution is

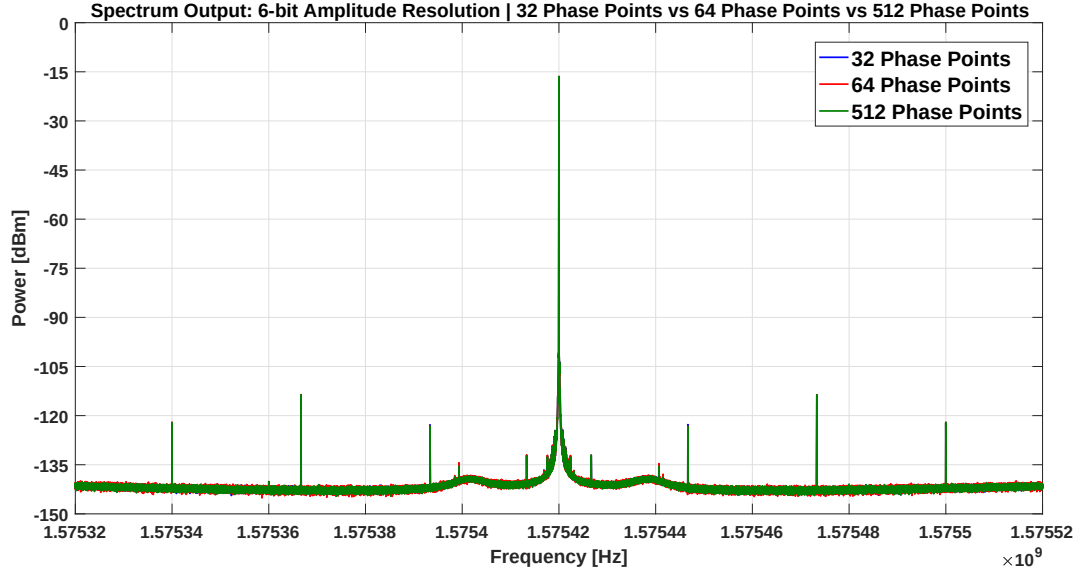


Figure 37. Narrowband Spectrum Output Due To Phase Truncation: 6-bit Amplitude Resolution.

kept constant at 5 bits in this experiment. A 5-bit phase resolution constitutes a LUT with 32 phase points. The objective is to observe the spurious performance due to two different amplitude resolution values: 5 bits and 6 bits. Table 19 provides an overview of the two test cases analyzed in this experiment. Figure 38 illustrates the sine wave representations of the LUT configurations in test cases 1 and 2.

Table 19. Spur Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution

Test Case	PR (bits)	Phase Points	AR (bits)	Amplitude Range
1	5	32	5	(-8,8)
2	5	32	6	(-16,16)

4.6.1 Wideband Spurious Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution.

The spectrum outputs of the two different LUTs are shown in Figure 39. From the analysis of Figure 39, it can be seen that the spectrum for the 5-bit and 6-bit amplitude

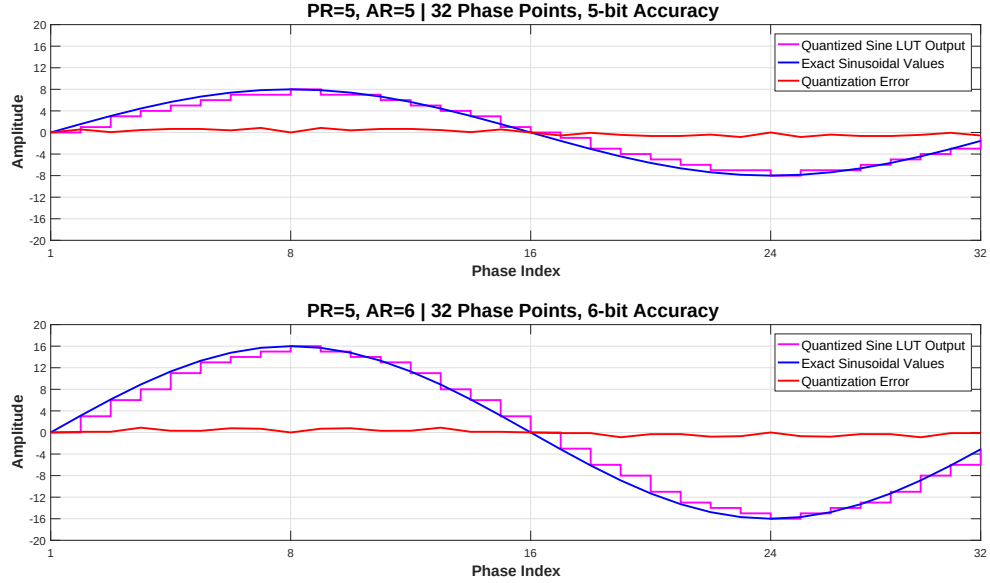


Figure 38. Sine Wave Quantization Effects Due To 5-bit and 6-bit Amplitude Resolution

resolution implementations are similar, unlike the spectrum for the phase resolution analysis. One notable difference between the 5-bit and 6-bit amplitude resolution implementations is the location of the spurs. The harmonics for both implementations appear at the same offset frequencies with the same amplitude. However, the location of the spurs vary in each implementation. In the 5-bit amplitude resolution implementation, the spectrum shows several spurs above -90 dBm approximately 1 MHz away from the lower sideband of the carrier. In the 6-bit amplitude resolution, at this location, these spurs are below -95 dBm. This is just one example that illustrates the difference in spur location in the spectrum of the two different implementations. To better analyze the spurious performance due to finite amplitude resolution in the LUT, the same metrics calculated for the phase truncation experiments are calculated for this experiment.

The results are shown in Tables 20 and 21. Table 20 compares the power contained in the carrier to the power contained in all of the spurs and harmonics in the 100 MHz span. Again, a threshold was used to determine which trace points are

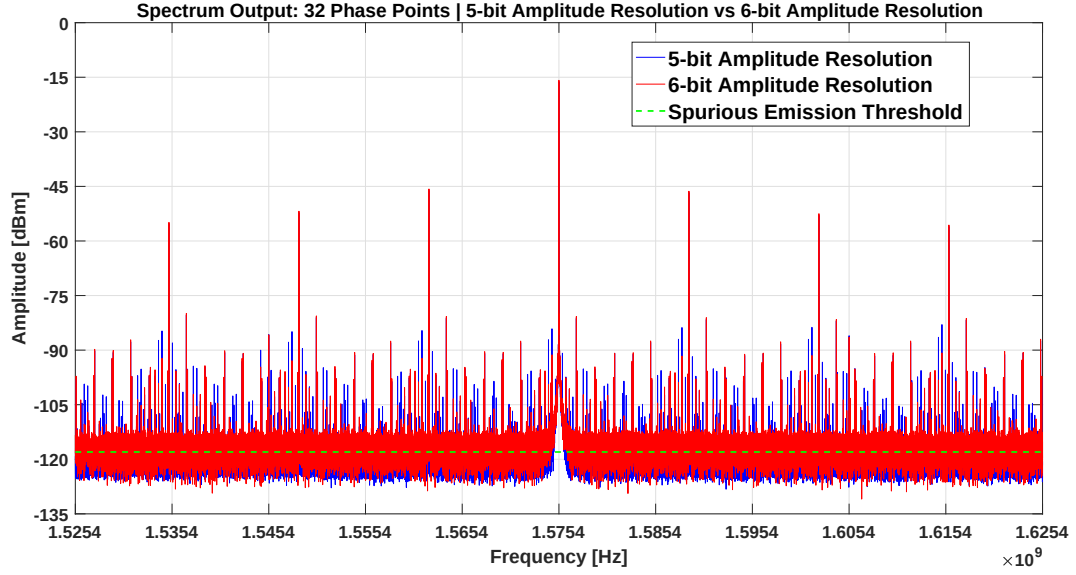


Figure 39. Wideband Spectrum Output Due To Finite Amplitude Resolution: 5-bit Phase Truncation.

Table 20. Wideband: 32 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions

AR	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
5 bits	-15.45	-41.23	25.77
6 bits	-15.03	-40.83	25.79

spurs/harmonics. It can be seen that both implementations yield similar results for all three metrics in Table 20. Therefore, this experiment shows that in the wideband analysis of this spectrum, the phase resolution has greater impact on spectral performance than amplitude resolution. This theory is further verified by analyzing the results shown in Table 20, which analyzes the SFDR performance with respect to each LUT implementation. Again, it can be seen that both LUT implementations produce extremely similar results. Hence, further proving that amplitude resolution plays a smaller role in spurious performance compared to phase resolution.

Table 21. Wideband: 5-bit Phase Resolution: Spurious Free Dynamic Range

Span [MHz]	Metric	5-bit AR	6-bit AR
20	Carrier Power [dBm]	-15.45	-15.03
	Max Spur [dBm]	-81.27	-80.77
	SFDR [dBc]	-65.82	-65.73
100	Carrier Power [dBm]	-15.45	-15.03
	Max Spur [dBm]	-46.12	-45.71
	SFDR [dBc]	-30.66	-30.67

4.6.2 Narrowband Spurious Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution.

The narrowband spurious performance results did not produce any noticeable differences compared to the previous narrowband experiments. Hence, it reinforces the concept that for the current LUT configuration and the FTW value chosen for this architecture, the spurious performance of the GWPP architecture close to the carrier is not directly impacted by finite amplitude resolution. Further details and results from this experiment are described in Appendix G Section 7.1.

4.7 Spur Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution

This is another experiment performed to analyze the spurious performance due to finite amplitude resolution. It is essentially an extension to the experiments performed in Section 4.6. However, the spurious performance due to finite amplitude resolution is analyzed using a 6-bit phase resolution, whereas Section 4.6 experimented with a 5-bit phase resolution. Therefore, this experiment analyzes the effects of using 5- and 6-bit amplitude resolution with a LUT consisting of 64 phase points. Last section experimented with the same effects using a LUT with 32 phase points, which showed minimal spurious performance improvement resulting from changing the amplitude resolution. The experiment in this section will explore the spurious performance with

64 phase points and different amplitude resolution implementations. Table 22 illustrates the two test cases analyzed in this experiment. The sine wave representation for each is shown in Figure 40.

Table 22. Spur Performance Due To Finite Amplitude Resolution: 6-bit Phase Truncation

Test Case	PR (bits)	Phase Points	AR (bits)	Amplitude Range
1	6	64	5	(-8,8)
2	6	64	6	(-16,16)

4.7.1 Wideband Spurious Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution.

The spectrum outputs of the two different LUTs are shown in Figure 41. From initial observation of Figure 41, it can be seen that more spurs and harmonics are generated as a result of the 5-bit amplitude resolution implementation. To fully analyze the spurious performance using a 6-bit phase resolution with two different amplitude resolution implementations, same performance metrics were used to analyze the previous LUT implementations. The results are presented in Tables 23 and 24. First, the results in Table 23 are discussed followed by the results in Table 24.

Table 23. Wideband: 64 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions

AR	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
5 bits	-15.39	-48.44	33.04
6 bits	-15.00	-48.24	33.24

In Table 23, the power contained in the carrier is compared to the power contained in all of the spurs and harmonics combined in the spectrum. The span of this spectrum is 100 MHz. The results obtained from this experiment precisely match the results obtained from Section 4.6. The only difference is that the performance metrics in this section are calculated using a LUT with 64 phase points instead of 32. However,

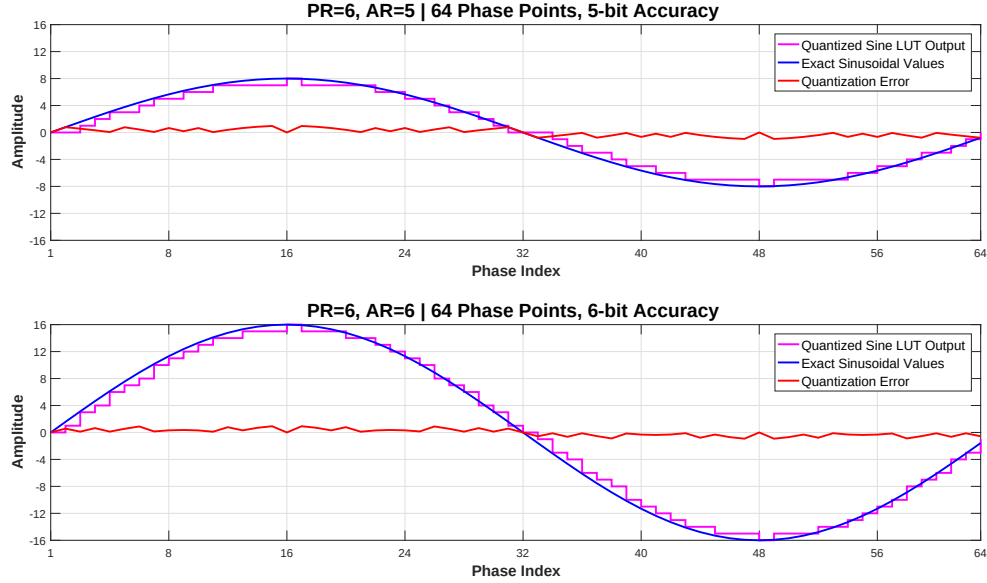


Figure 40. Sine Wave Quantization Effects Due To 5-bit and 6-bit Amplitude Resolution

both sets of results indicate that minimal changes in the amplitude resolution do not directly impact the spurious performance. This is shown by observing the metrics shown in Table 23; each metric calculation for both LUTs are precisely equivalent.

Table 24. Wideband: 6-bit Phase Resolution: Spurious Free Dynamic Range

Span [MHz]	Metric	5-bit AR	6-bit AR
20	Carrier Power [dBm]	-15.39	-15.00
	Max Spur [dBm]	-70.62	-78.44
	SFDR [dBc]	-55.22	-63.44
100	Carrier Power [dBm]	-15.39	-15.00
	Max Spur [dBm]	-52.26	-51.87
	SFDR [dBc]	-36.86	-36.87

Table 24 compares the ratio of the fundamental signal power to the power of the highest spur/harmonic in a specific span. For this performance metric, the two specific spans correspond to 20 MHz and 100 MHz. From the analysis of Table 24, it can be seen that both the 5-bit and 6-bit amplitude resolution LUT have equivalent SFDR calculations in the 100 MHz span; however, in the 20 MHz span, the 6-bit amplitude resolution configuration yields a better SFDR calculation. Therefore, for

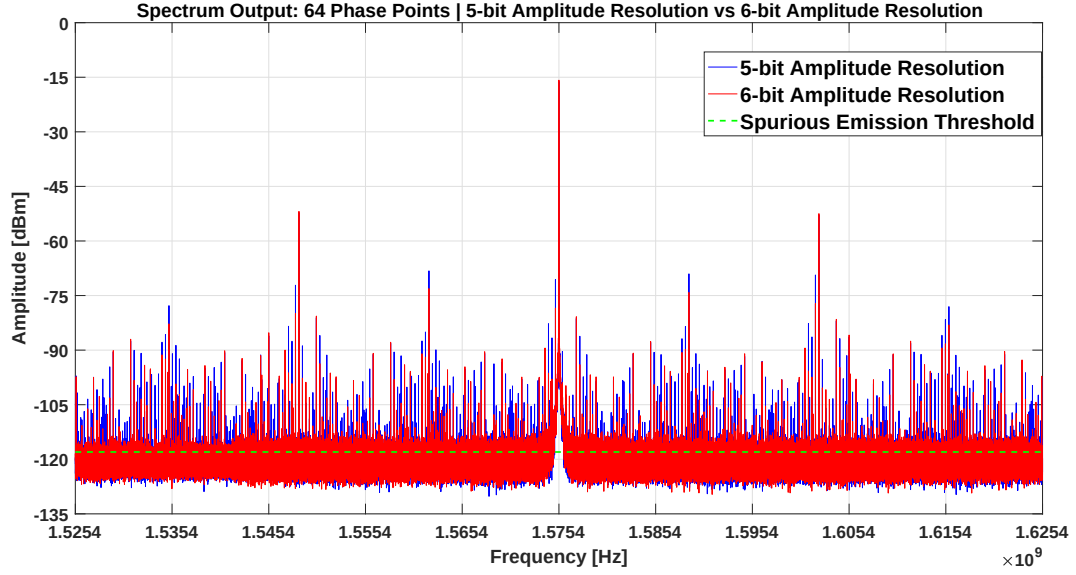


Figure 41. Wideband Spectrum Output Due To Finite Amplitude Resolution: 6-bit Phase Resolution.

a 64 phase point LUT, the 6-bit amplitude resolution implementation yields an 8 dB SFDR improvement in a 20 MHz span compared to the 5-bit amplitude resolution. This is attributed to the fact that the highest amplitude value obtainable in the 5-bit amplitude resolution implementation is 8. Therefore, towards the peak of the oscillation, multiple sine LUT indexes hold the same amplitude value. Therefore, more interpolation occurs resulting in a slight increase in quantization. It is also important to note that comparing the results obtained from testing the two different LUTs with 32 phase points introduces interesting findings. First, in terms of the ratio of the power contained in the carrier compared to the power contained in the spurs/harmonics, the 64 phase point LUT outperforms the 32 phase point LUT by 8 dB. This is attributed to the fact that there is less quantization error due to the increased number of points. Therefore, less power is contained in the spurs/harmonics for a LUT consisting of 64 phase points. Next, in terms of SFDR in the 100 MHz span, the 64 phase point LUT outperformed the 32 phase point LUT by 6 dB. Again, this is attributed to the 64 phase point LUT having greater attenuation of spurs/harmonics

in a wideband span. Finally, in terms of SFDR in the 20 MHz span, the 32 phase point LUT outperforms the 64 phase point LUT by 6 dB. Since the SFDR figures in the 64 phase point LUT were different, the average of the two were taken. This value is compared with the average of the two 20 MHz SFDR figures from the 32 phase point LUT to come up with the 6 dB difference.

4.7.2 Narrowband Spurious Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution.

The narrowband spurious performance results did not produce any noticeable differences compared to the previous narrowband experiments. Hence, it reinforces the concept that for the current LUT configuration and the FTW value chosen for this architecture, the spurious performance of the GWPP architecture close to the carrier is not directly impacted by finite amplitude resolution. Further details and results from this experiment are described in Appendix G Section 7.2.

4.8 Spur Performance vs FPGA Resource Utilization

This section compares the results obtained from the performance metrics with the amount of hardware consumed to realize the desire functionality. For this architecture, the resources inside the FPGA that are the most valuable for this analysis are the slices and CLBs. Slices are sub-units within the CLB, which contain resources for implementing combinatorial and sequential logic circuits. The slices inside the Virtex[®]-6 FPGA are composed of 4 LUTs, 8 FFs, and additional logic. The LUTs can be reconfigured as one 6-input LUT or two 5-input LUTs. In addition, specific slices such as SLICEMs can be used to implement memory operations. There are two slices inside the CLB. Therefore, this analysis studies the spurious performance with respect to slice and CLB utilization. The analysis is done by considering the

performance metrics used to study the different LUT configurations.

The first performance metric used to study the spurious performance due to the combination of phase truncation and finite amplitude resolution is the comparison between the power contained in the carrier and the power contained in the spurs/harmonics in the spectrum. Table 25 compares the results obtained from this performance metric with the amount of FPGA resources used to implement each LUT. Hence, it can be seen that the best spectral performance with respect to this metric is the case for when the LUT consists of 512 phase points and a 6-bit amplitude resolution. This implementation only occupies 4,063 slices out of a total of 37,680. Therefore, this implementation only uses 10% of the slices available on the Virtex[®]-6 FPGA. Since there are two slices in each CLB the 512 phase point LUT with 6-bit amplitude resolution utilizes 2,032 CLBs.

Table 25. Power_{CSE} vs FPGA Resource Utilization

Phase Pts	AR	Power _{CSE}	FPGA Slices	FPGA CLBs	Slice Utilization
32	5 bits	25.77 [dB]	3,637	1,819	9%
32	6 bits	25.79 [dB]	4,136	2,068	10%
64	5 bits	33.04 [dB]	4,194	2,097	11%
64	6 bits	33.24 [dB]	3,871	1,936	10%
512	6 bits	40.04 [dB]	4,063	2,032	10%

Table 26 analyzes the SFDR achieved by each LUT configuration tested in this thesis with the amount of FPGA resources used to implement each LUT. However, this table only analyzes the SFDR in a 100 MHz span centered around the fundamental carrier, which is the GPS L1 carrier. Again, the best SFDR in a 100 MHz span is the implementation of the LUT with 512 phase points and 6 bits of amplitude resolution. The SFDR achieved in this configuration is -46.2390 dBc and consumes 4,063 slices, which translates to 2,032 CLBs. Therefore, this implementation only consumes 10% of FPGA slices and CLBs, respectively.

Table 27 analyzes the SFDR achieved by each LUT configuration tested in this

Table 26. SFDR 100 MHz Span vs FPGA Resource Utilization

Phase Pts.	AR	SFDR [dBc]	FPGA Slices	FPGA CLBs	Slice Utilization
32	5 bits	-30.66	3,637	1,819	9%
32	6 bits	-30.71	4,136	2,068	10%
64	5 bits	-36.86	4,194	2,907	11%
64	6 bits	-36.87	3,871	1,936	10%
512	6 bits	-46.32	4,1063	2,032	11%

thesis with the amount of FPGA resources used to implement each LUT. However, this table only analyzes the SFDR in a 20 MHz span centered around the fundamental carrier, which is the GPS L1 carrier. Again, the best SFDR in a 20 MHz span is the implementation of the LUT with 512 phase points and 6 bits of amplitude resolution. The SFDR achieved in this configuration is -68.9716 dBc and consumes 4,063 slices, which translates to 2,032 CLBs. Therefore, this implementation only consumes 10% of FPGA slices and CLBs, respectively.

Table 27. SFDR 20 MHz Span vs FPGA Resource Utilization

Phase Pts.	AR	SFDR [dBc]	FPGA Slices	FPGA CLBs	Slice Utilization
32	5 bits	-65.82	3,637	1,819	9%
32	6 bits	-64.73	4,136	2,068	10%
64	5 bits	-55.22	4,194	2,097	11%
64	6 bits	-63.44	3,871	1,936	10%
512	6 bits	-68.97	4,063	2,032	10%

4.9 Experimental Spurious Performance vs Theoretical Spurious Performance

In this chapter, the spurious performance was analyzed with respect to errors due to the implementation of the LUT. These errors are mainly created due to the phase truncation of the accumulator output and the finite amplitude resolution of the amplitude samples stored in the LUT. These are typical error sources associated with DDS-based systems. Since DDS is an emerging and maturing signal generation

technology, it has become an interesting field of study. Furthermore, the literature presented on DDS covers many aspects of DDS from typical error sources to implementation platforms to different phase-to-amplitude converters. One particular dissertation that was used to verify the spurious performance of this platform with respect to the error sources mentioned above was reference [23]. In reference [23], the author derives a closed-form equation to quantify the theoretical spur level as a result of phase truncation. If the quantity $A - P$, where A is the accumulator size in bits and P is the phase word size in bits, is 4 or more, then the maximum spur level turns out to be very closely approximated by $-6.02P$ [23]. Since the accumulator used in this architecture is greater than 32 and the maximum word size used in experimentation was 9 bits, this equation can be used to approximate the maximum spur level generated by the DDS architecture built inside the GWPP platform. Thus, the quantity $-6.02P$, where P represents the phase resolution, is a theoretical metric used as a comparison with the experimental maximum spur level generated by the GWPP platform. The comparison is presented in Table 28.

Table 28. SFDR 100 MHz Span vs Theoretical

Phase Resolution	Amplitude Resolution	SFDR [dBc]	Theoretical SFDR [dBc]
5 bits	5 bits	-30.66	-30.10
5 bits	6 bits	-30.71	-30.10
6 bits	5 bits	-36.86	-36.12
6 bits	6 bits	-36.87	-36.12
9 bits	6 bits	-46.32	-54.18

Based on Figure 28, each experimental SFDR calculation corresponds with theory except for the last configuration where the phase resolution comprises 9 bits. The experimental SFDR is -46.32 dBc, whereas the theoretical is -54.1800 dBc. The difference between the two values could possibly be attributed to intermodulation distortion caused by the spectrum analyzer. However, additional analysis will need to be performed to prove this theory. However, this equation provides a closed-form

method of quantifying the maximum spur level due to phase truncation. Therefore, this equation can be used in further development to compare experimental results with theoretical results.

4.10 Summary of Results

The summary of results obtained in this chapter is shown in Table 29. It can be seen from Table 29 that the optimal choice for LUT configuration based on these experiments is a LUT with 512 phase points and 6-bit amplitude resolution. This implementation yields the best spurious performance and only consumes one additional percentage point of slices in the FPGA. It is interesting to note that the 64 phase point LUT utilizes less slices than the 32 phase LUT. From a closer look into the resource summary provided by the FPGA tools showed how the synthesis the LUTs with 32,64, and 512 phase points with 5-,6-, and 6-bit AR, respectively, were able to utilize more fully used LUT-FF pairs resulting in better optimization of resources compared to the other LUT implementations.

Table 29. Summary of Results

Phase Points	AR	SFDR 20 MHz	SFDR 100 MHz	P _{CSE}	Slice Utilization
32	5 bits	-65.8202	-30.6627	25.7771	9%
32	6 bits	-64.7334	-30.716	25.7932	10%
64	5 bits	-55.2238	-36.8635	33.0480	11%
64	6 bits	-63.4476	-36.8756	33.2454	10%
512	6 bits	-68.9716	-46.3290	40.0423	10%

V. Conclusion and Future Work

THIS chapter summarizes the research and presents the conclusion to this thesis. Recommendations for further research are also presented.

5.1 Summary

GPS has become the de-facto standard for precision time transfer. By definition, all signal structure details of the open civilian GPS signals are known and published. This makes them vulnerable to numerous threats. As GPS signals continue to operate at the heart of critical infrastructures, there is a need to mitigate these threats by incorporating signal authentication features. However, the challenge in doing so is the lack of a hardware platform to verify the features. This thesis presented a FPGA-based GPS waveform prototyping platform that features a software-defined all-digital signal synthesis architecture in conjunction with a high-speed DAC as a solution.

The architecture of this platform is based on the emerging and promising signal generation technique known as DDS. DDS adds dimensions not possible with analog designs in the sense that the fundamental nature of the waveform and its features are generated digitally; hence, providing precise control over the amplitude, frequency, phase, and modulation scheme of the output waveform. The inclusion of a high-speed DAC enables the ability to synthesize RF carriers without the need for PLLs and analog mixers.

The goal of this research is to design, implement, and integrate the GWPP architecture into an FPGA framework. This work demonstrates the feasibility in using a DDS-based architecture to generate GPS waveforms. However, the limited resources in the FPGA presents a trade-off between signal quality and FPGA resource utilization that must be considered. For DDS-based systems, the trade-off is a result of

the phase resolution and amplitude resolution of the LUT implementation. Both of these factors greatly impact the spectral purity of the GWPP architecture as well as the FPGA resource utilization. This work analyzes the experimentation of different LUT configurations, which consisted of varying the phase and amplitude resolutions of the architecture. From the results, it was shown that the LUT configured with 9-bit phase resolution and 6-bit amplitude resolution yields the best performance: SFDR of -46 dBc with only 10 % of FPGA logic slices consumed. Furthermore, this work verified the SFDR matches the theoretical SFDR for DDS-based systems.

5.2 Recommendations for Future Work

The recommendations for future work can be broken down into two categories: further experimentation and architecture expansion. Further experimentation refers to additional testing with different LUT configurations. In this thesis, the spectral performance was analyzed with five different configurations. The last configuration in which the phase resolution is set to 9 bits and the amplitude resolution set to 6 bits yielded the best spectral performance. This configuration only consumed 10% of FPGA slices. Consequently, there is a considerable amount of resources left on the FPGA to experiment with different LUT configurations. Therefore, a future recommendation for experimentation is to ultimately experiment with different LUTs to find a point of diminishing returns. In addition, this would effectively create a profile in which designers can choose the LUT configuration based on the desired spectral performance. Another recommendation for further experimentation is to assess signal quality and measurement performance with respect to testing GWPP architecture with a GPS receiver, either standalone or software-defined. The purpose of this task would be demonstrate acquisition and tracking as well as verify correct PRN generation.

The next recommendation for future work is to optimize and expand the current GWPP architecture. First, the C/A-code generator needs to incorporate the capability of starting the PRN sequence at any fractional code phase offset. Recall that during the Code NCO initialization, each code NCO is initialized with an offset since 16 C/A-code samples need to be generated every clock cycle. The overflow behavior of the Code NCO controls the LFSRs in the C/A-Code Generator. It essentially represents the transition to the next chip. However, a special situation in initialization occurs when one or more code NCOs, after the first NCO is initialized, overflows. In this case, their respective LFSRs shift to the next chip, whereas the LFSRs associated with the preceding NCOs do not. As a result, the C/A-Code Generators are not in sync which results in the incorrect generation of the C/A-code. Currently, this procedure is limited and controlled by software. In other words, in software, the capability to program the Code NCO to begin at any fractional code phase after a certain threshold is not supported. A recommendation for future work is to handle this situation in hardware. This task is importance since it provides greater flexibility in the C/A-code generation in the architecture. Next, the architecture can be enhanced by replacing the half wave LUT with a quarter-wave LUT. This implementation will further improve FPGA resource utilization. In addition, the architecture can be enhanced by incorporating the capability to modulo-2 add the navigation message to the C/A-code. This implementation would require the shift from a reference oscillator that has great short-term phase noise performance to one with great long-term phase noise performance.

Through design, implementation, and integration of the GWPP architecture, this research has laid out the groundwork for achieving the goal of an FPGA-based GPS waveform prototyping platform. Further expansions of this architecture with the recommendations presented in the previous paragraphs in conjunction with the ability

to generate additional GPS waveforms will elevate the performance and capabilities of GPS in the field of advanced signals research.

Appendices

DEVELOPMENT OF A SOFTWARE-DEFINED GNSS SIMULATION
PROTOTYPE FOR ADVANCED SIGNALS RESEARCH

A. FMC230 Digital to Analog Converter Configuration Fundamentals

The FMC230 is a dual-channel DAC connected to the FPGA through the FMC HPC interface. The functional block diagram for this DAC is shown in Figure 42. For this present discussion, we will bypass the overall theory of operation and only discuss several components/signals of the DAC functional block diagram that are valuable to understanding the process in which the DAC converts the GPS L1 C/A digital waveform to analog. These components/signals include the following: $P_0D[13:0]P$, $P_0D[13:0]N$, DCI_X , DCO_x , $DACCLK_x$, and DAC core. First, the 14-bit LVDS data port associated with DAC 0: $P_0D[13:0]P$ and $P_0D[13:0]N$ and $P_1D[13:0]P$ and $P_1D[13:0]N$. This port enables a DDR LVDS interface to the FPGA. DDR simply means that data is transferred on both falling and rising edges of the clock as shown in Figure 43. LVDS simply means that the port transmits or receives information as the difference between the voltages on a pair of wires. This differential signaling system provides high data rate transmission, along with low power and improved noise performance [25]. DDR in conjunction with LVDS creates a high data rate, simplified interface between the FPGA and DAC. From the perspective of the DAC, it expects the FPGA to send both de-interleaved data to the LVDS port, along with an embedded *Data Clock Input (DCI)* signal; both in a DDR format. The suffix also means it is a differential signal.

The DCI clock operates at one-fourth the data rate to effectively reduce the bus

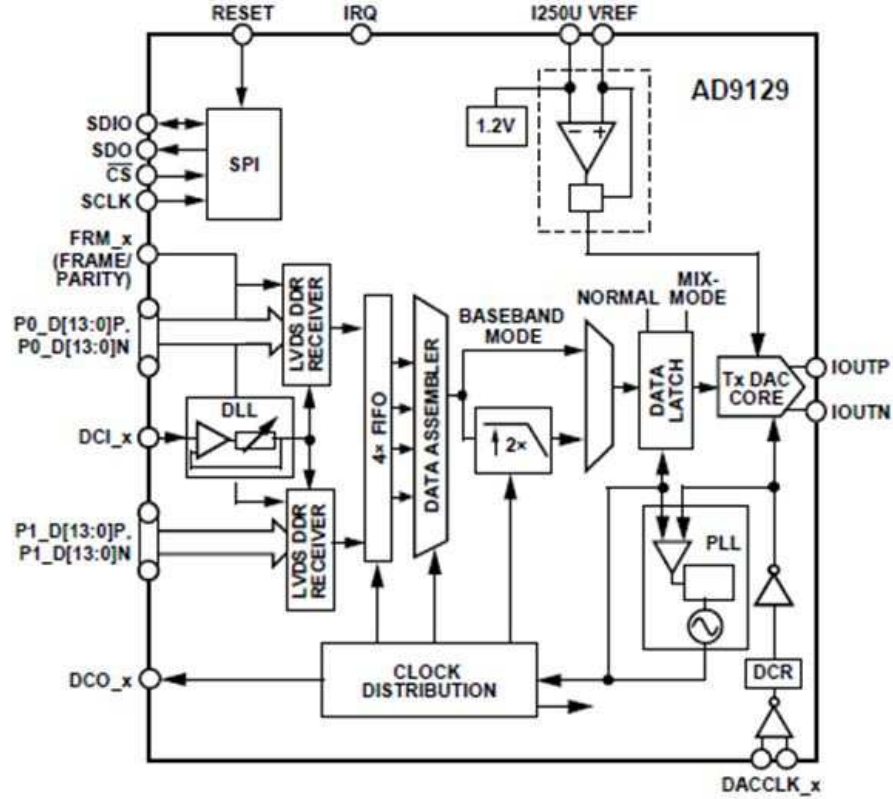


Figure 42. DAC (AD9129) Functional Block Diagram [19]. The DACCLK_x is the 2400 MHz clock provided from Keysight N5182B MXG Vector Signal Generator; DCO_x is the clock from the DAC provided to the FPGA; DCI_x is the clock from the FPGA provided to the FPGA; and P0 are the 14 bits samples generated from the GWPP architecture.

interface speed to half the data rate. Since it is also a DDR signal, each edge of the DCI transfers data into the DAC. It is important to note that the data must also be aligned with the DDR DCI signal to sample the de-interleaved data coming from the FPGA on both the rising and falling edge of the DCI clock. The DCI clock is generated with the same frequency as the *Data Clock Output (DCO)* provided by the DAC to effectively establish synchronous operation between the DAC and FPGA. Nonetheless, this information is critical to understanding the rate at which the DAC requires the FPGA to transfer data. If these timing requirements are not met, the DAC will incorrectly sample the digital GPS L1 C/A waveform and ultimately produce a distorted analog waveform. The DCO is an output resulting from

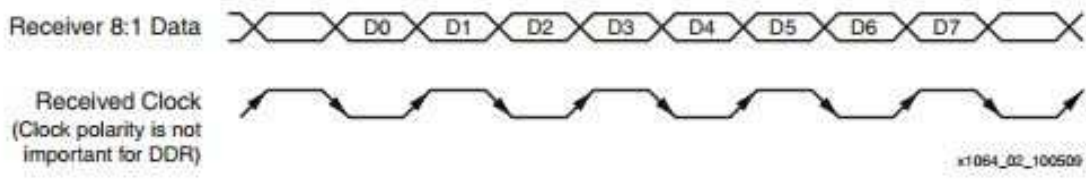


Figure 43. 8:1 Data Stream Using DDR [24]. Data is clocked on both rising and falling edges of the received clock.

the $DACCLK_x$ input going through a Duty Cycle Restorer (DCR) and then being divided by four; Again, this division is to support DDR transfer with a high sampling rate. The $DACCLK_x$ input directly drives the DAC core to minimize clock jitter and sets the sampling rate of the DAC. It is crucial that this signal is near 50% duty cycle because this is expected by the interfacing hardware. Duty cycle refers to the fraction of one period in which a signal is active. Hence, 50% duty cycle means the signal is high for half the period and low for the other half. Not adhering to this duty cycle will bring challenges when communicating between hardware on the DAC and FPGA. Therefore, the DCR ensures near 50% duty cycle to the subsequent circuitry. It is also important to note that the output of the DCR serves as the master clock to the subsequent circuitry. It is routed directly to the DAC core, as well as to a clock distribution block that generates all critical internal and external clocks [19]. The clock source quality of $DACCLK_x$, as defined by its phase noise characteristics and jitter, is an important consideration in maintaining optimum DAC performance. However, this is further explained in experiment setup.

Finally, the DAC core is where the digital to analog conversion takes place. It is based on a quad-switch architecture that enables dual-edge clocking operation along with additional operating modes such as mixed-mode. When configured in either mode, this effectively allows the DAC to support a sampling rate up to 2.85 Gsps. According to the *Sampling Theorem* [26], the max frequency that can be reconstructed to avoid ambiguities resulting from aliasing must follow the relationship shown in

Equation (1.0.1) below,

$$F_s > 2F_{max} \quad (1.0.1)$$

where F_s is the sampling rate and F_{max} is the maximum frequency in the signal to be sampled to prevent aliasing. In order to determine the max frequency that is reconstructible without aliasing, we simply rearrange Equation (1.0.1) to develop an expression for F_{max}

$$F_{max} < \frac{F_s}{2} \quad (1.0.2)$$

Hence, it is shown that the max frequency able to be reconstructed to avoid aliasing must be less than half the sampling rate. Therefore, with a 2.85 Gsps sampling rate, the highest frequency we can reconstruct without aliasing is 1.425 GHz. Since the GPS L1 C/A frequency is 1.57542 GHz, Mix-Mode operation is required to reconstruct the carrier in the second Nyquist zone. Equation (1.0.1) must be obeyed to keep the signal in the first Nyquist zone. With this signal, there is another alias in the second Nyquist zone which is of interested. Mix-Mode provides the ability to use this signal. Overall, this section covered a high level overview of the processes that take place inside the DAC to convert the GPS digital waveform to a real-world analog waveform to be tested with receivers and other test equipment. Here are some of the main take aways from the configuration of the DAC:

- The DAC has a sample rate up to 2.85 Gsps. From the Sampling Theorem, this is not efficient enough to reconstruct the GPS L1 C/A carrier without aliasing. Therefore, Mixed-mode is required for this DAC.
- Mixed-mode operation enables the ability to reconstruct RF carriers in the second Nyquist zone. This mode enables direct RF synthesis to the GPS L1 C/A center frequency. The DAC must be changed from normal mode to Mix-Mode by changing the DAC initialization firmware.

- The DAC interface has two DDR LVDS data ports that create a high speed, simplified interface for the FPGA to transfer the digital waveform. Although only one is used in this research to demonstrate proof of concept. These are the ports that receive the GPS L1 C/A samples from the FPGA.
- DAC provides an output DCO signal to synchronize the FPGA to DAC. Inside the FPGA, this signal is called DCI. It is sent to the DAC along with the 14-bit data bus. The falling and rising edges of the DCI signal essentially samples the de-interleaved data coming from the FPGA.
- FPGA must transfer data in DDR format at a specific rate aligned with the DCI signal. If not, the DAC will inaccurately sample the digital waveform and generate a distorted output waveform.

B. Distributed Clock Tree Configuration Fundamentals

The **DCT** is essentially a multi-output clock distribution component. It generates the sampling clock for both DACs in addition to other clocks used for test and monitoring purposes. The sampling clock is provided through the Low-Voltage Positive-Emitter Coupled Logic (LVPECL) ports: *OUT0* and *OUT1*, which are shown in the DCT functional block diagram in Figure 44. However, we are only interested in the *OUT0* port since it connects to *DAC 0*. However, it is important to discuss some key components that play a vital role in generating the sampling clock. Recall that the clock source quality, as defined by its phase noise characteristics and jitter, is an important consideration in maintaining optimum DAC performance. Since the end goal is to synthesize the GPS L1 C/A signal, any jitter or noise in the sampling clock will appear in the spectrum of the GPS L1 C/A signal. This can potentially result in failure to acquire/lock on to the signal or incorrect pseudorange measurements. Although this is further discussed in experiment setup (Section 3.3), it is important to get an idea of what components affect the quality of the sampling clock. As shown in Figure 44, these components are *Ref*, *Clock*, *TCXO*, and *VCO*. Both *Ref* and *TCXO* are off-chip and on-chip reference oscillators, respectively, used inside the PLL to maintain a constant phase angle relationship between the VCO and reference oscillator. This relationship is key for generating a stable, high-frequency oscillator greater than 2 GHz; which is required to accurately sample the GPS carriers. Consider the following: VCOs use the combination of inductors, capacitors, and diodes (also called a varactor circuit) to create a frequency of oscillation. Any slight voltage variation in the circuit will cause a shift in frequency, meaning VCOs are not stable. On the other hand, crystal oscillators are known for their stability. However, to acquire a crystal oscillator over 100 MHz would be extremely expensive considering we only need the crystal oscillator for its stability, not necessarily frequency. Therefore, the

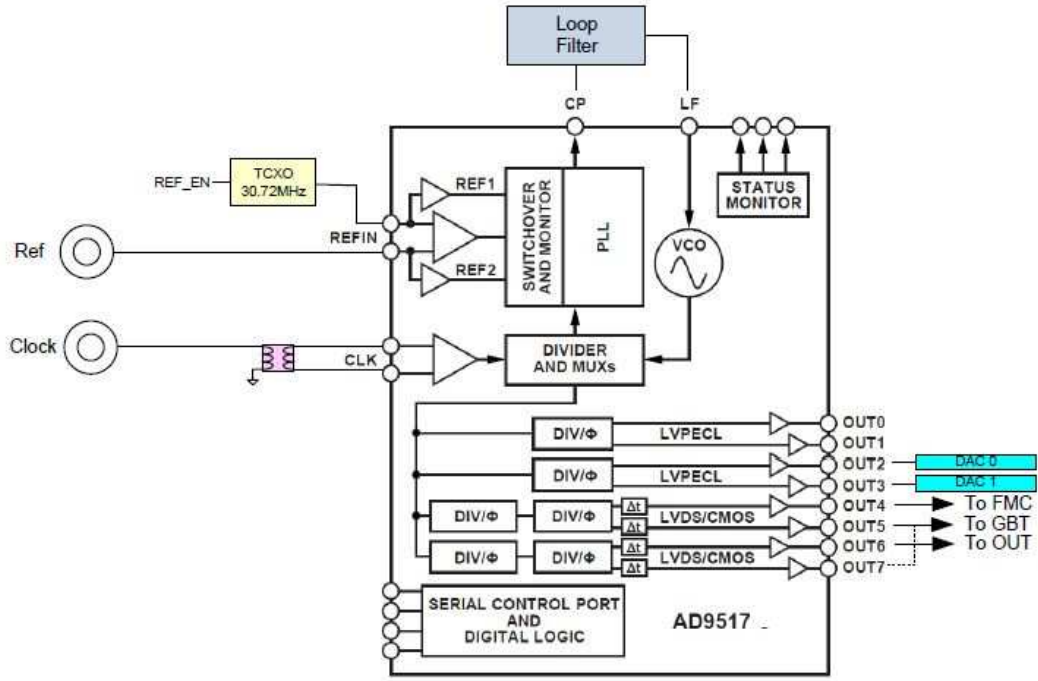


Figure 44. DCT (AD9517-1) Functional Block Diagram [18]. Only DAC0 is used in this research.

PLL locks the VCO to the reference oscillator; effectively combining the flexibility of the VCO with the stability of the reference oscillator. After all, this is just one way to generate the DAC sampling clock in the DCT. The other way is by providing an external sampling clock to the *Clock* input of the DCT. Like the VCO, the external sampling clock is generated by an oscillator. This oscillator typically is not as stable as crystal reference oscillators. For this reason, they too are phase locked to an external oscillator. In that case, the external sampling clock is considered a PLO. In spite of having two different methods of generating the sampling clock used in the DAC, only one method can be configured at once. The design decisions made in determining which method to use are further explained in experiment setup. Here are the main take aways from this section:

- DCT is responsible for generating the sampling clock provided to the DAC.

- Inside the DCT, there are two methods for generating the sampling clock: supplying an external sampling clock or using the PLL.
- Both external sampling clock and on-chip VCO are phase locked to reference oscillators to take advantage of their excellent stability.
- Clock source quality, as defined by its phase noise characteristics and jitter, is an important consideration in maintaining optimum DAC performance. Any noise or jitter inside the clock source will reflect in the quality of the sampling clock, in turn affecting the performance of the DAC.
- Since the end goal is to synthesize the GPS L1 C/A signal, any jitter or noise in the sampling clock will appear in the spectrum of the GPS L1 C/A signal; potentially resulting in the failure to acquire/lock on to the signal or calculate incorrect pseudorange measurements.

C. Serializer/Deserializer Configuration Fundamentals

The OSERDES primitive is used to transmit data from the FPGA interface to the DAC while minimizing the number of IO pins and interconnects. The SERDES primitive supplies serial streams of parallel data in DDR format to the LVDS ports on the DAC interface. A high level representation of this process is shown in Figure 45.

In the hardware implementation of the GWPP architecture, this is done by the built-in OSERDES core in the Virtex[®]-6. As previously mentioned, it is a dedicated parallel-to-serial converter with specific clocking and logic resources designed to ease the implementation of high-speed digital interfaces (source-synchronous interfaces) [20]. A block diagram of the OSERDES component is shown in Figure 46. In Figure 46, it is important to understand the *CLK* and *CLKDIV* signals as well as the *Data Parallel-to-Serial Converter*. These components not only give insight to how the data is transferred to the DAC, but also the transfer rate. The data converter receives two to six bits of parallel data from the FPGA fabric, but can support more with OSERDES width expansion (basically chaining two OSERDES primitives together). Next, it serializes the data and presents it to the IOB via the OQ outputs. It is important to note that the parallel data is serialized from the lowest order data input pin to highest. In other words, D1 is the first bit transmitted. The converter is available in two modes: SDR and DDR, however in this research it is configured for DDR due to the DAC requirements. With a DDR configuration, the width to the parallel-to-serial converter can be expanded. The available widths are 4:1, 6:1, 8:1, and 10:1 [20]. Widths larger than six requires a pair of OSERDES to be configured into a master-slave configuration. Since the GWPP platform is generating 14-bit samples, 8:1 data width configuration is chosen. With this configuration, it takes four CLK cycles when the rising edge of CLKDIV clocks the data at the inputs of the

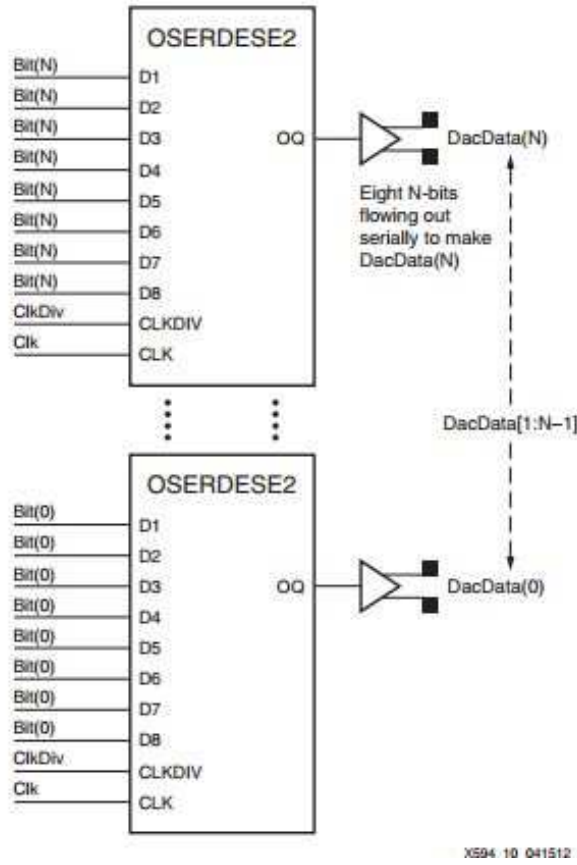


Figure 45. SERDES Parallel-to-Serial Interface [20]

converter into the OSERDES to when the first bit of the serial stream appears at OQ [20]. This latency determines the data rate required in the OSERDES block to meet the transfer rate requirements of the DAC. Thus, it is taken into consideration when distributing clocks to this component. *CLK* and *CLKDIV* are the two clocks used in OSERDES for data rate conversion. *CLK* is the high speed serial clock, which drives the serial side of the parallel-to-serial converter; it is essentially the DCI/DCO clock divided by two. *CLKDIV* is the divided version of the high-speed clock input and it drives the parallel side of the converter; it is essentially the divided version of the clock connected to the *CLK* port. It is crucial that these signals remain phase-aligned to ensure that data is being transferred to the DAC at the correct rate.

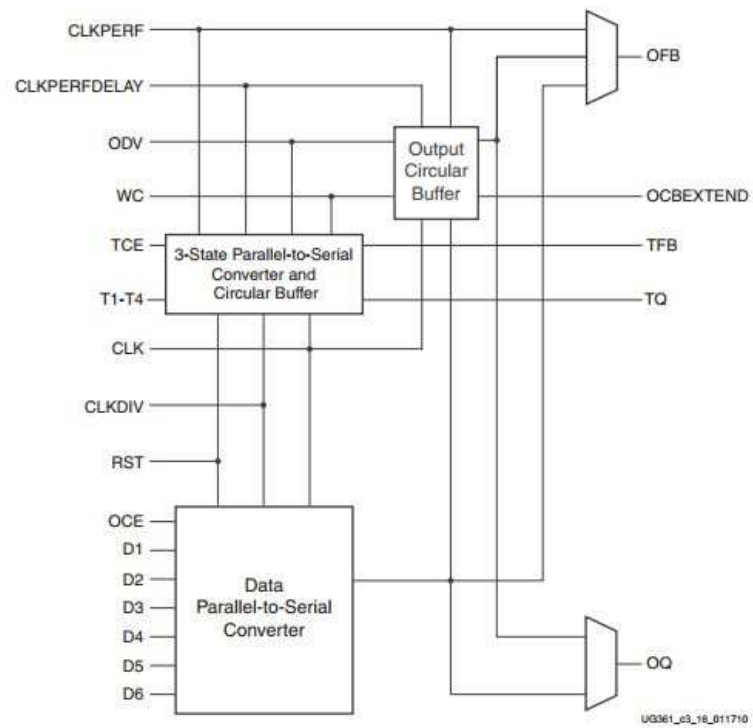


Figure 46. OSERDES Block Diagram [20]

D. Universal Asynchronous Receiver Transmitter to L1 Course Acquisition 16 Sample Generator Interface

This component serves as an interface between *UART receiver* and *L1 C/A 16 Sample Generator*. An interface component is needed between the UART receiver and L1 C/A 16 Sample Generator blocks for several reasons. One, both components operate in different clock domains; all UART operations are clocked by a 66 MHz clock, whereas all operations inside L1 C/A 16 Sample Generator block are clocked by an approximate 150 MHz clock. Recall that the 150 MHz clock is the calculated clock rate needed for the SERDES primitive to transmit 224 bits to the DAC to ensure accurate sampling in the DAC. Interfacing two components working at two different clocks introduces challenges in the sense that the design becomes asynchronous at the boundary of each interface. Therefore, sending data through different clock domains could result in unreliable data transfer. Furthermore, it can also result in setup and hold violations as well as metastability issues. Another reason for the need of an interface component is that the UART receiver is built to receive one byte; the architecture of this block is discussed in Appendix E. However, for this present discussion, only consider that the host PC, via MATLAB[®] scripts, calculates the initialization values to be routed to the configuration registers. The configuration registers are several bytes in size to accommodate for the size of the phase accumulator in both Carrier and Code NCOs. Therefore, the data being sent from the host PC are arranged as a string of bytes, where the first byte corresponds to the address of the configuration register, while the remaining bytes correspond to the data (initialization value) to be stored in the configuration register. After the calculation is arranged in this format, it is ready to be transferred to the FPGA through the established communication interface. In this architecture, for reasons explained in Appendix E, UART serial communication is chosen as the communicating interface. Therefore,

UART inside the host PC disassembles the arranged string of bytes, which contains the configuration register address and data, into individual bits and transmits them at a specific baud. However, the UART receiver in this architecture is only designed to decode one byte. Therefore, if multiple bytes are sent, each byte received will be replaced with the incoming byte. Therefore, this design component is needed to collect the UART bytes and reassemble the bytes into the original format calculated in software; which is the first most significant byte is the address and the following least significant bytes are the data (initialization values for the configuration registers). This is essential because the address-data format is what enables the ability to not only program multiple configuration registers, but also allows the address decoding to accurately route the data to its appropriate register. Another reason for this design component is to accurately transfer data between the clock domain used for the DAC sample computations (16 Sample Generator) and the UART receiving (UART receiver). In hardware design, a common technique to convert serial data into parallel form is through the implementation of shift registers. In addition, to perform data transfer across CDC boundaries, a First In First Out (FIFO) buffer is commonly used. Both of these elements make up the architecture of this interface component. This is shown in the functional block diagram for this interface component in Figure 47.

The same notation used to represent bus widths is applied to this block diagram as well. It is also important to note that some signals have variables as their size due to the fact that these are chosen by the user. These variables are: n , which is the number of shift registers in the interface component; z , which is the number of bits needed to count n bytes; and x , which is the number of bits for the data entering the FIFO buffer. For example, if the host PC sends five bytes (40 bits), five shift registers are needed to collect the five bytes and put the 40-bit data into the

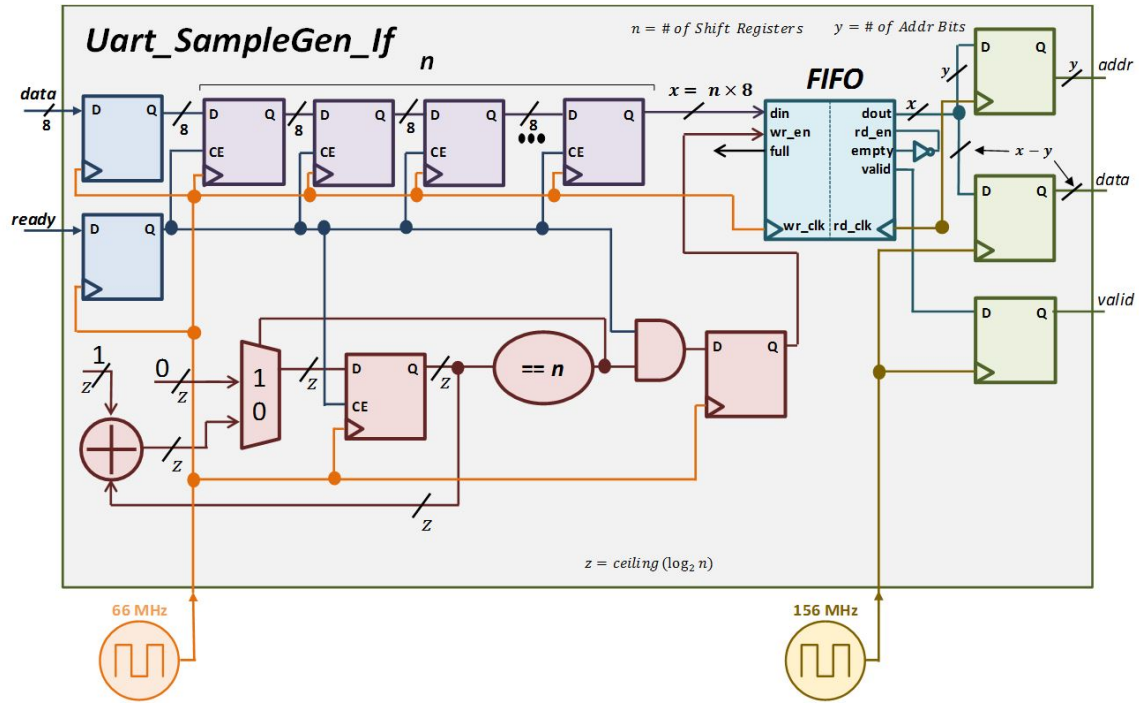


Figure 47. UART to Sample Generator Interface Functional Block Diagram

FIFO buffer. The architecture is designed as a source-sink flow network. The source covers all the operations involved in shifting in the UART bytes, counting the number of bytes sent, concatenating each byte into a data bus and sending the bus to the FIFO. These operations are clocked by the 66 MHz clock. The sink covers all the operations involved in reading the data from the FIFO, constructing the separate data and address signals, and outputting these signals along with a valid signal. These operations are clocked by the 150 MHz clock. On the source side, there are two input registers that register the data and ready signals coming from the UART receiver. The input registers are there to ensure that timing requirements are met. This is a design practice that will also be seen in other UDBs. As mentioned previously, the source side also contains the shift registers used to collect UART bytes. The host PC sends several bytes in sequence to program the configuration registers since these are also several bytes in size. Since the UART receiver block only receives one-byte, without the shift registers, each decoded byte will be replaced by the next incoming

byte. Therefore, the shift registers are used to collect the bytes sent from the host PC. The last part of the source side is the byte counter. This is shown by the maroon blocks in Figure 47. The byte counter is designed to keep track of the number of bytes sent from the UART receiver. The counter is needed to ensure that all of the bytes sent from the host PC have been recovered before the data enters the FIFO.

On the sink side of the interface, the main component is the dual, asynchronous FIFO. Dual, asynchronous is a characteristic given to FIFOs that have two clocks—one for writing, one for reading—that operate independently. The FIFO is used in the interface to transfer data between registers that operate in two different clock domains. Hence, performing clock domain synchronization. Interfacing two logic blocks working at two different clocks is challenging in the sense that the design becomes asynchronous at the boundary of each interface. This often results in setup/hold violations, metastability, and unreliable data transfer [27]. Metastability refers to signals that do not assume stable 0 or 1 states during the setup/hold times in the FFs. Although the effects of metastability are further explained in [27], for the present discussion, we need only consider that metastability issues result in the propagation of erroneous signals into the receiving clock domain. This issue typically results from the receiving clock domain sampling the signal when it is not stable.

Therefore, important design considerations require that multi-clock designs be carefully constructed at CDC boundaries; in this case, the source and sink boundaries. Several methods for mitigating metastability are described in [27] such as two FFs synchronizers and three FF synchronizers. However, when implemented using the clock domains specific to this block, the Mean Time Before Failure (MTBF) was too short. MTBF in this sense means a signal that passes through a synchronizing FF, goes metastable on the first stage synchronizer FF, and continues to be metastable one clock cycle later when it is sampled in the next stage or stages of the

synchronizer FFs. Due to short MTBF times with FF synchronizers, we decided to use a third method for clock domain synchronization presented in [27]: asynchronous FIFO. In the asynchronous FIFO, data is inserted into the FIFO from the write clock domain (66 MHz), and removed from the FIFO from the read clock domain (150 MHz). With debugging using Xilinx Chipscope along with General Purpose Input / Output (GPIO) LEDs, we found this approach to be a safe way to pass multi-bit values between clock domains. However, this technique results in additional usage of memory, which is a disadvantage. The FIFO was generated using Xilinx CORE Generator and is built to store 16 words. To facilitate error free operations, we added FIFO-specific full, empty, and valid signals generated by their respective clock. The full signal is used to indicate that the FIFO is at max capacity. However, since the data getting inserted into the FIFO ultimately comes from the UART receiver and is transmitted at a low data rate, it is unlikely that the FIFO will ever reach max capacity. The empty signal is used to indicate whether there are contents stored in the FIFO. With this specific FIFO, it takes four clock cycles for the empty signal to de-assert after the first word is inserted into the FIFO. As a result, it takes four clock cycles for data to be available on the output of the FIFO. Hence, with the FIFO, we gain reliability in ensuring accurate data transfer while sacrificing latency. Finally, the valid signal indicates whether the receiving component successfully read from the FIFO. This signal is used as a data valid signal once data is successfully read from the FIFO. Once, the data is successfully read from the FIFO, both the data and valid signals are passed through output registers. Again, the output registers serve in the same manner as the input registers to prevent timing violations and ensure synchronization. It is important to note that the data read from the FIFO is split into two signals, data and address. The bus-width for both of these signals depend on the number of shift registers as well as the number of address and data bits used

in the UART transmission from the host PC.

In conclusion, this component is used in a two-fold manner: reassemble the UART bytes into address-data format and transfer the data and address components through two different clock domains to the address decoder inside the *GPS L1 C/A 16 Sample Generator*. The assembling of the n UART bytes is done by using n shift registers. Thus, a counter is used to count the number of bytes received to ensure that all of the bytes that were transmitted were received. Once all of the bytes are received, this data is inserted into the FIFO from the 66 MHz clock domain. The dual, asynchronous FIFO is used to perform clock domain synchronization to ensure reliable data transfer between the two clock domains. Once the FIFO is not empty, data is removed from the FIFO from the 156 MHz clock domain. The valid signal indicates whether the read was successful. Therefore, this enables the FIFO to provide the receiving component, *GPS L1 C/A 16 Sample Generator* with an address, data, and valid signal, to be sent to the address decoder. The valid signal signifies the address and data signals are correct. It is important to note that before the data enters the shift registers, it must be accurately sampled and decoded by a UART receiver. For this reason, this design architecture requires a UART receiver component, which is discussed next, to establish a communication channel between the host PC and FPGA.

E. Universal Asynchronous Receive/Transmit Receiver

Serial communication between the host PC and FPGA is a vital part of this architecture. After all, it is the part that deals with configuring the FPGA to synthesize various waveforms; more specifically, the GPS L1 C/A signal as the goal of this research. It essentially enables the ability to program the configuration registers inside our logic blocks with the goal of altering their current or initialization contents. Recall, serial communication is the process of sending data one bit at a time, sequentially over a bus. This is in contrast to parallel communication, where several bits are sent as a whole. Although serial communication may seem inferior to parallel since it can transmit less data per clock cycle, it is often the case that serial communication can achieve a higher data rate. This is often due to the fact that parallel interfaces suffer more from clock skew and cable length. This is one reason why serial communication is favored over parallel. Another is that serial interfaces have fewer pins and are therefore less expensive to implement. In this architecture, we are programming a phase accumulator greater than 20 bits. The phase accumulator is implemented using an adder and register. A 20-bit register is made with 20 FFs. To initialize 20 FFs through parallel communication would require 20 pins on the FPGA. As a result, design costs will increase as well as area. For this reason, parallel communication is impractical for this project and hence serial is the preferred choice for communication.

As previously mentioned, UART is the serial communication protocol chosen in this architecture. Mainly, due to it being relatively inexpensive in terms of hardware to implement. The block diagram is shown in Figure 48. In addition, the UART protocol has a simple interface since it consists of two wires: *TX* and *RX*, where *TX* corresponds to transmit and *RX* to receive. For this architecture, the design requirements only required simplex communication between the host PC and FPGA. Hence, the host PC is responsible for transmitting the data bits while the FPGA is

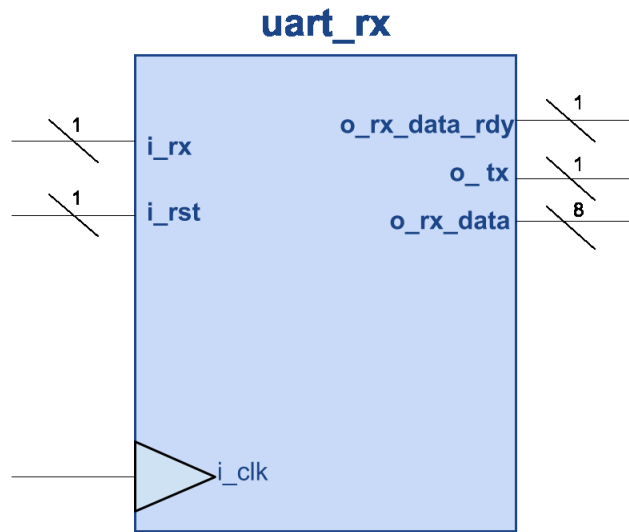


Figure 48. UART Interface Functional Block Diagram

responsible for receiving the data bits and re-assembling the bits into complete bytes. Since there is no requirement to transmit back to the host PC, this component is only designed to receive. Implementing a transmitter in this block would introduce additional logic and increase design costs. Therefore, adding logic to support transmit is impractical and a waste of hardware resources. There were several design considerations made before implementing this component. One key design consideration was choosing what timing parameters to use in this block. Recall that in UART protocol, both transmitting and receiving devices must agree on the baud rate. Baud rate can be thought of as the language between the two devices. If two devices are not speaking at the same speed, it often results in data being either misinterpreted or completely missed. While there is no *theoretical* limit to UART, most devices only support a certain set of fixed baud rates. Since UART reception deals with oversampling the UART bits, it is a common practice to have a UART input clock to your receiver that is 16x the baud rate [28]. With very high baud rates, it would require higher frequency clocks, which may not be available on some FPGAs. Therefore, it is important to understand the clocking resources on the FPGA before choosing the

baud rate to ensure that the input clock is able to perform the 16x oversampling. However, with the Virtex[®]-6, the on-board user clock is 66 MHz, which is already greater than 16x the highest baud rate allowed on the host PC used in this research (115200 baud). Since the highest baud allowed for the host PC is 115200, we designed the UART receiver around this baud rate to achieve the highest data rate possible. 115200 baud simply means that the serial port is capable of transferring a maximum of 115200 bits per second. Another design consideration made was the structure of the UART frame. In other words, how many data, parity, and stop bits will the sending UART transmit. With UART protocol, the start and stop bits are required to synchronize the receiving and transmitting UART. This process allows the receiver to know how many data bits and parity bits to expect before the stop bit arrives. Again, the goal is to demonstrate proof of concept with the fewest bits. Therefore, we chose the UART frame structure illustrated in Figure 49. 8N1 simply means the frame consists of eight data bits, no parity bit, and one stop bit.

The receiver is built to receive a 8N1 UART frame (refer to Figure 49) at 115200 baud. The decision to build a one-byte UART has several advantages. For one, it allows for coarse-grain reconfigurability. Coarse-grain reconfigurability is an architecture in which larger components are built from smaller subcomponents. Hence, building a UART receiver to receive 8 bits creates a UART subcomponent which can be used to build larger UART receiver components. For example, this component can be used with 8-bit shift registers to technically build a UART receiver that receives more than 8 bits (technically, any multiple of one-byte). Consider the following: this component connected to four 8-bit shift registers technically simulates a 32-bit UART receiver. This phenomenon was explained in the discussion of the *UART_SampleGenerator_If* UDB implementation. All operations of the UART receiver are controlled by a 66 MHz clock signal available on the Virtex[®]-6. This clock

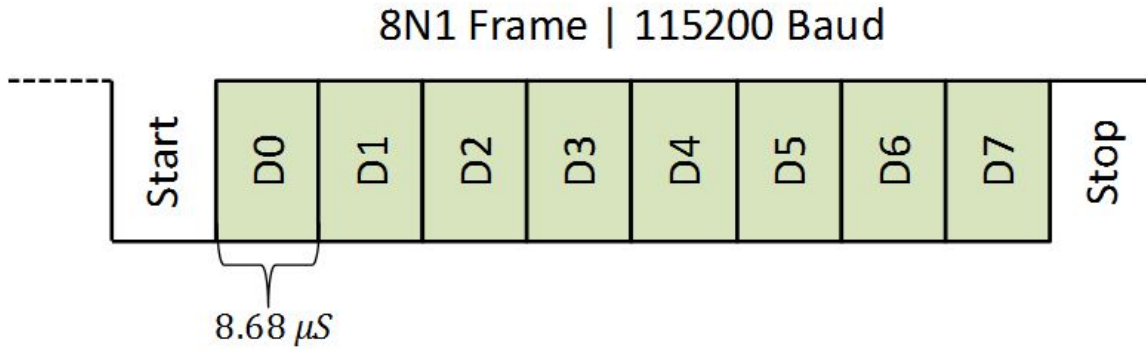


Figure 49. UART 8N1 Frame, 115200 Baud. 8 Data Bits, No Parity Bit, 1 Stop Bit

runs at a multiple of the data rate, approximately 573 times the bit rate, which is sufficient for 16x oversampling. The DCO clock provided to the FPGA from the DAC is also an available clocking resource; however, since this clock is a divided clock based on the sampling rate, it is often generated as a non-integer clock source and minimally changes with the sampling rate. Since this architecture uses the input clock and the baud rate to calculate predetermined sampling times for each bit, the constant changes in the DCO clock frequency would require the designer to change the sampling times based on the new DCO frequency. To eliminate this overhead, we chose to implement this block with a stable 66 MHz user clock that is known to operate deterministically. This clock was also verified using an oscilloscope to ensure 66 MHz frequency. In addition, this clock source travels through a MMCM primitive to ensure operation at a defined phase and frequency.

The architecture of this receiver block does not follow the typical UART receiver design where shift registers are used to convert the serial bits to parallel data. Instead, the receiver samples each bit at specific sampling times that correspond to half its period based on two assumptions: 115200 baud rate and 66 MHz input clock. For example, based on these assumptions, the period of one UART bit is equal to 574 clock cycles. Based on UART knowledge, the start bit is the first bit to be received and is detected when the RX line changes from logic high to logic low. A pipelined

falling-edge detection algorithm is used to perform this detection. Once the start bit is detected, it is assumed that this bit will have a period of 574 clock cycles. Therefore, to validate that this is indeed the start bit and not a spike, it is sampled at half its period, 287 clock cycles. Sampling at half the period of a UART bit transmission is a typical design practice for several reasons. One, if sampling at the beginning of the period, the bit could be a spurious spike and not the actual bit. On the other hand, sampling at the end of the period could possibly lead to sampling the next bit, not the current bit of interest. For these reasons, UART bit sampling occurs at half the period of transmission. If the start bit changed to logic high from the time it was first received to being sampled at half its period, it is considered to be a spurious pulse and invalid. At that point, the receiver generates a start bit error, which notifies the control logic to return to the idle state and wait for a new start bit. On the other hand, if the start bit is unchanged (still logic low), it is considered valid and placed on a 10-bit bus. After waiting 574 more clock cycles, the first data bit is sampled and placed on the bus. Since the start bit was sampled at half its period, waiting 574 clock cycles would result in sampling the first data bit at half its period. After the required number of bit periods have elapsed (10 bits have been received), the control logic verifies the received UART frame. If the first bit received, which is the start bit, is low and the last bit received, which is the stop bit, is high, the received UART frame is considered valid. At this point, the contents on the bus that correspond to the data bits are made available to the receiving component, *Uart_SampleGenerator_If*. The UART will set a flag to indicate new data is available and valid. Both valid and data signals are sent to *Uart_SampleGenerator_If*. Once these signals are sent, the UART receiver returns to the idle state and waits for the next transmission from the host PC. The infrastructure of this block is controlled by a mealy Finite State Machine (FSM) and a running counter. Mealy state machine was chosen over moore

FSM since it is typically faster and reduces glitchy outputs. The state machine and implementation is shown in Figure 50. The state machine handles the control logic of the implementation. It controls whether the receiver is idling (waiting for a start bit), receiving (sampling the bits), or clearing (notifying receiving component that data is not ready). In addition, it performs the falling-edge detection and determines when the received UART data is available to the receiving component. The running counter begins at zero and increments at each rising edge of the clock cycle. Since the number of UART bits are known and the period of each bit is known, this counter counts to a predetermined value that corresponds to the length of transmission of the entire UART frame. As this counter increments, it only saves in the UART bits after the start bit is ascertain to be valid and the counter reaches the specific sampling times for each UART bit. Once the counter samples each bit, it asserts the flag to notify the control FSM to verify the UART frame. After verification, the data bits are reassembled into a byte and transferred, along with a ready flag, to the receiving component *UART_SampleGenerator_If*.

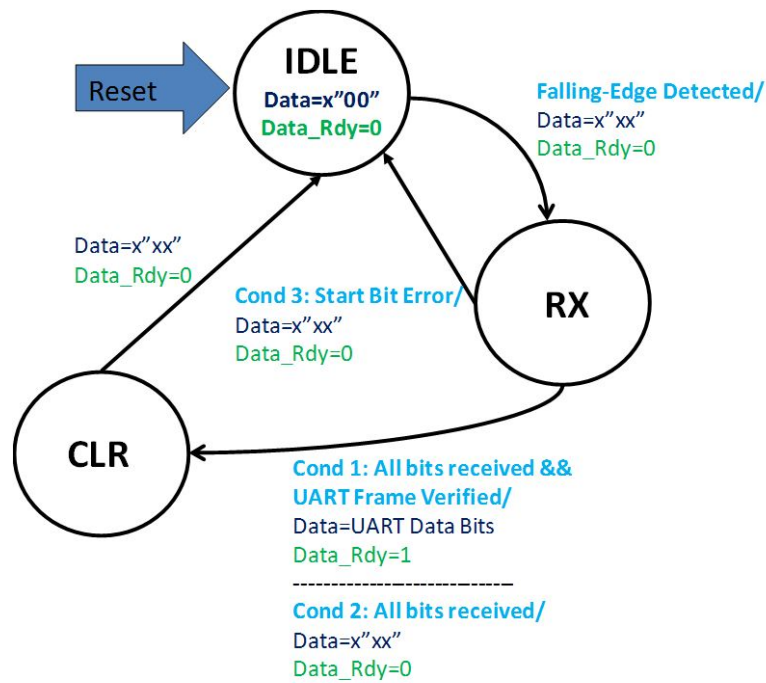


Figure 50. UART Algorithm FSM. In RX State, Condition 1 is tested first, then Condition 2, finally Condition 3.

F. User Defined Blocks RTL Verification

The purpose of Appendix A is to demonstrate verification of the user defined blocks designed in RTL. The verification is performed in ModelSim® and therefore only verifies functionality. It is important to note that the parameters used to verify this component are not necessarily the correct parameters to generate the GPS L1 C/A signal. Instead, the parameters are chosen to intuitively demonstrate proof of concept. The verification of the following blocks are provided: Carrier NCO, Code NCO, C/A-Code Generator, and Sine LUT. The verification of each block is done with a simulated system clock of 200 MHz.

6.1 Verification of Carrier NCO

Figure 51 shows the results from the verification of the Carrier NCO design block. Again, the simulated system clock is 200 MHz. To show proof of concept, the FTW is chosen to produce a 50 MHz output waveform. Recall that the Carrier NCO design element employs DDS, which comprises a phase accumulator. Typically, this design block is fed a BPSK offset and is connected directly to the sine LUT; however, these elements are disconnected from the design to prove correction functionality of the phase accumulation. The output of the phase accumulator increments in a staircase-like manner, which its analog representation without the Sine LUT is a ramp waveform. This is verified in Figure 51 and the two markers show the period of the staircase is 20 ns; hence, the frequency of the ramp is 50 MHz.

6.2 Verification of Code NCO

Figure 52 shows the results from the verification of the Code NCO design block. Again, the simulated system clock is 200 MHz. The FTW value chosen for this

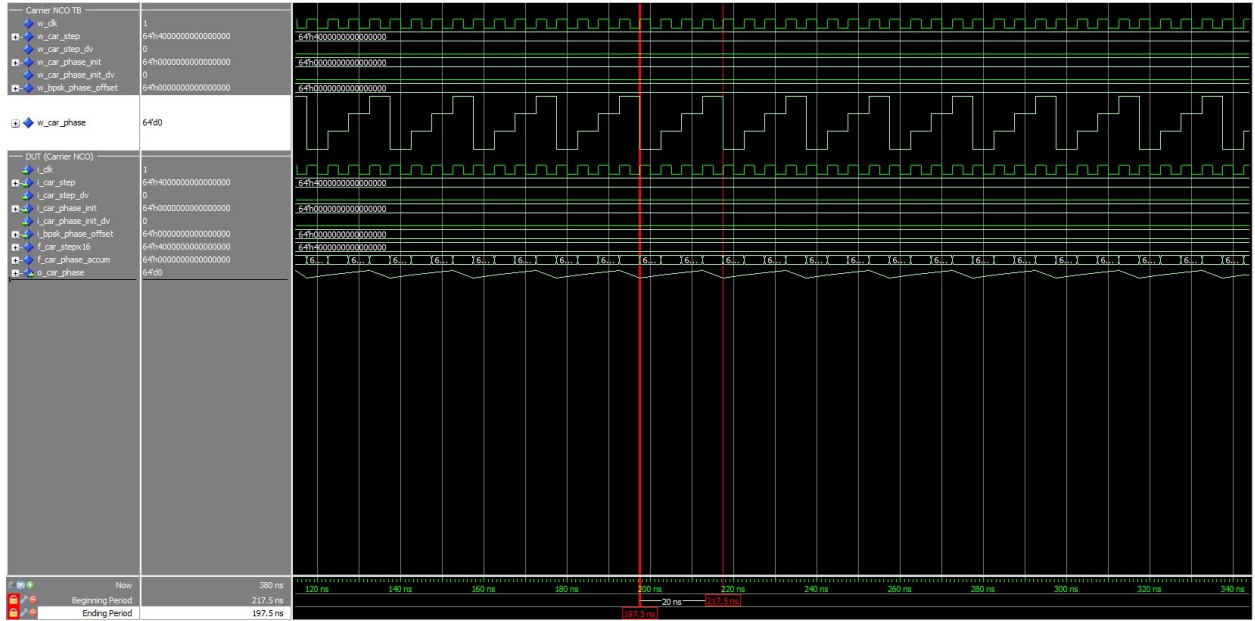


Figure 51. Verification of Carrier NCO Block

verification produces an output waveform with a frequency equal to 1.023 MHz. Recall that this is the nominal chipping rate for the C/A-code. The Code NCO design element also employs DDS, which comprises a phase accumulator. In this verification, the ramp output, which is the analog interpolated representation of the staircase function, is shown. This is verified in Figure 52 and the two markers show the period of the staircase is 980 ns. The actual period of the C/A-code is 977.5 ns. The 2.5 ns difference is due to the marker placement and the time scale on the waveform window. However, it verifies that the overflow signal has a frequency equivalent to the C/A-code chipping rate.

6.3 Verification of C/A-Code Generator

Figure 53 shows the results from the verification of the C/A-Code Generator design block configured for PRN 1 and integer code phase set to 0. Therefore, the expected result of this configuration is the generation of the PRN 1 C/A-code beginning with the first chip, Chip 0. In order to do so, the G1 and G2 LFSRs are initialized

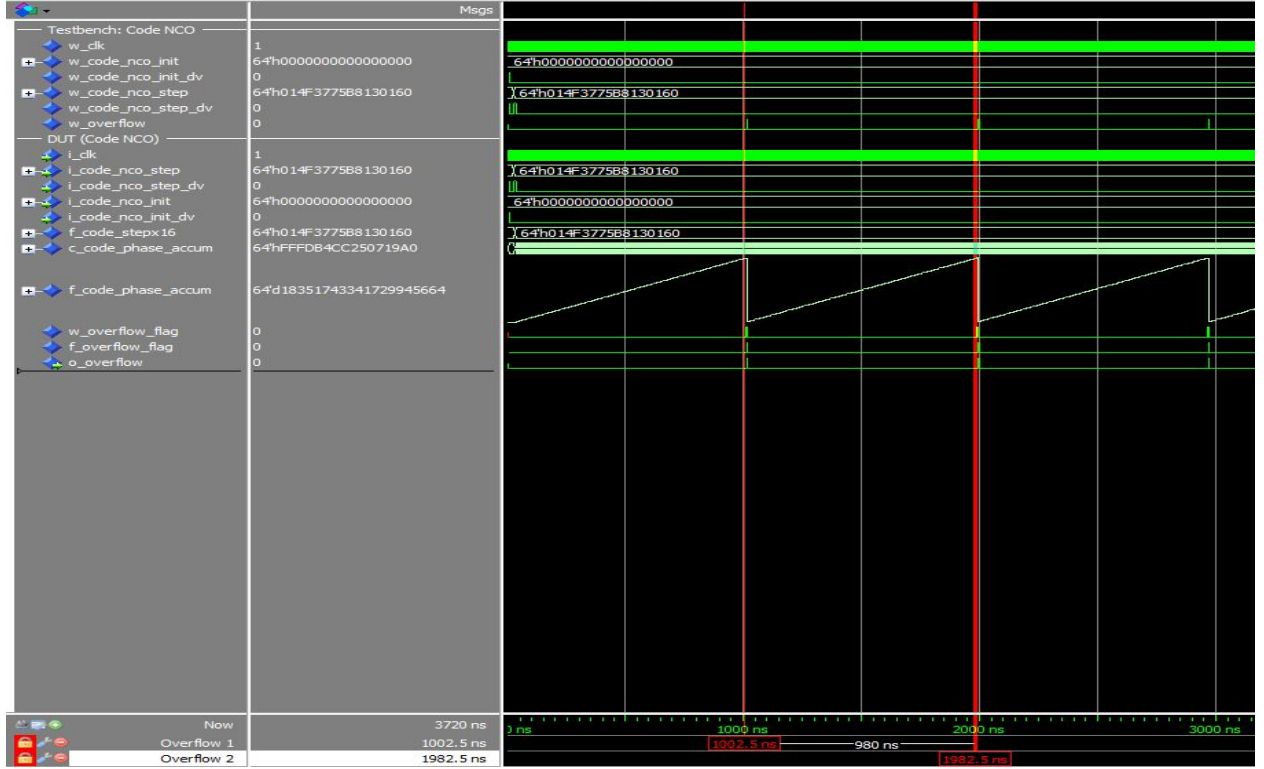


Figure 52. Verification of Code NCO Block

accordingly. The waveform window captures the first 10 PRN chips for PRN 1 with integer code phase set to 0, which are 1,1,0,0,1,0,0,0,0,0. Based on [5], the results match the first 10 chips for PRN 1 with integer code phase set to 0. Figure 54 shows the results from the verification of the C/A-Code Generator design block configured for PRN 1 and integer code phase set to 1. Therefore, the expected result of this configuration is the generation of the PRN 1 C/A-code beginning with the second chip, Chip 1. In order to do so, the G1 and G2 LFSRs are initialized accordingly. It is important to note that the initialization settings for the G1 and G2 LFSRs in Figure 54 are different from Figure 54. The waveform window captures the first 10 PRN chips for PRN 1 with integer code phase set to 1, which are 1,0,0,1,0,0,0,0,1,1. Based on [5], the results match the first 10 chips for PRN 1 with integer code phase set to 1.

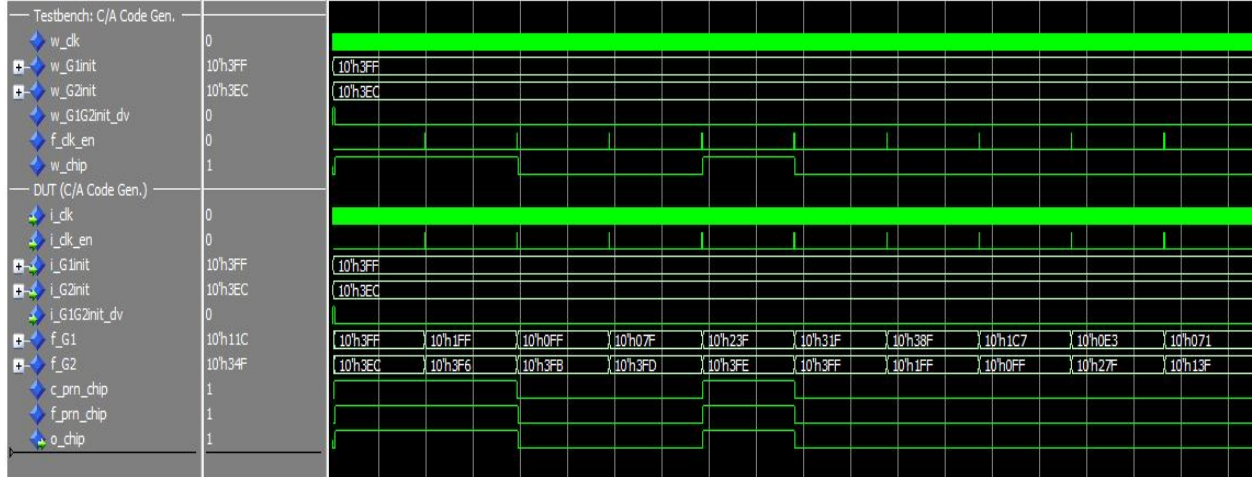


Figure 53. Verification of C/A Code-Generator, Code Phase 0

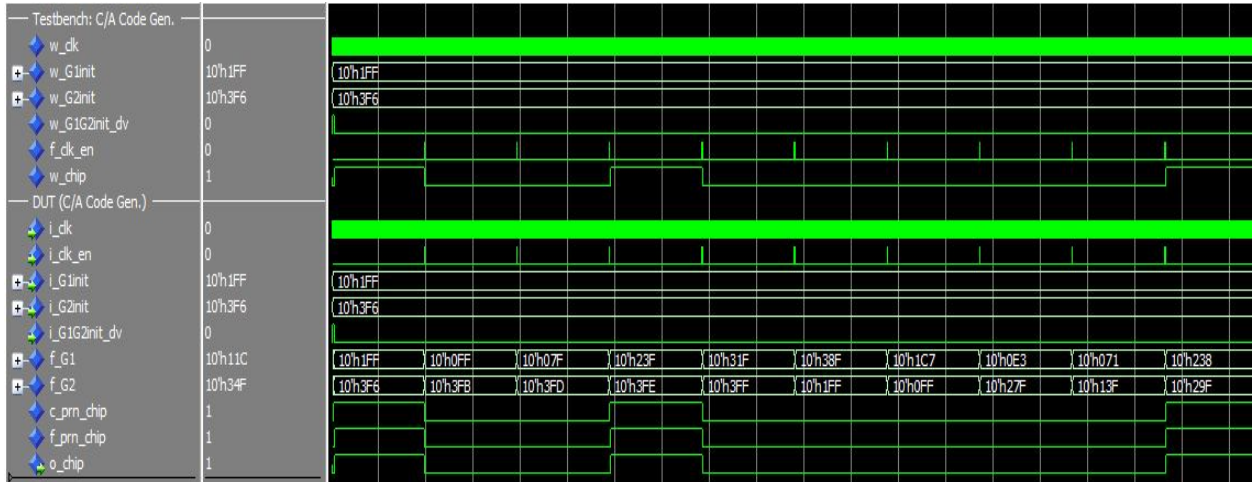


Figure 54. Verification of C/A-Code Generator, Code Phase 1

6.4 Verification of Half-Wave Sine LUT

Figure 55 shows the results from the verification of the Half Wave Sine LUT design block. Signals a, clk, and qspo are associated with the distributed memory CORE provided by Xilinx. The LUT is configured with phase resolution equal to 6 bits and amplitude resolution equal to 5 bits. Figure 55 verifies two important design concepts associated with the sine LUT implementation in the GWPP architecture. First, it verifies that the amplitude resolution stored in the LUT is 1 bit less than the final amplitude resolution, which is 6 bits (signal f.qspo). This represents one optimization

feature made in this architecture to optimize FPGA resource utilization. Second, it verifies tat the LUT only stores half the samples. The phase resolution is 6-bits which generates 64 phase points; however, in the LUT CORE, only 32 samples are stored. This is verified by acknowledging that the address signal to the LUT, a, is only 5 bits. Hence, only 32 phase points are stored in the LUT distributed memory CORE. The additional 32 phase points to complete the period of the sine wave are generated through additional logic.

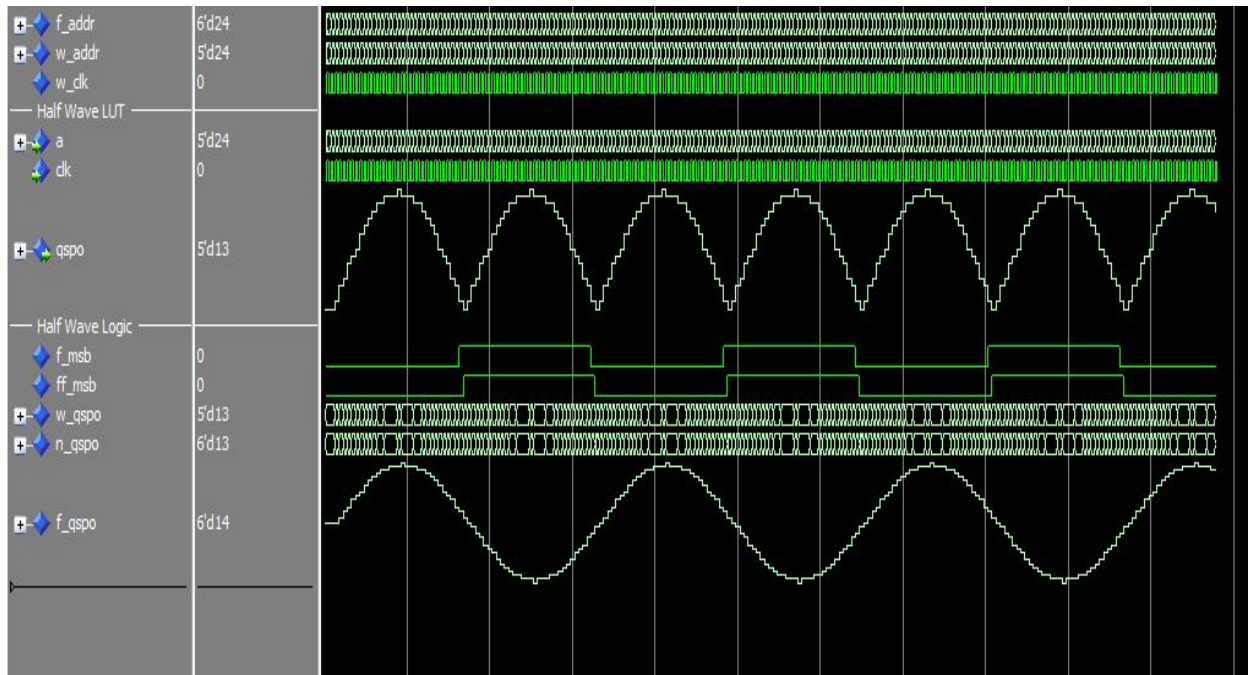


Figure 55. Verification of Half-Wave Sine LUT

G. Narrowband Spurious Performance Results

7.1 Narrowband Spurious Performance Due To Finite Amplitude Resolution: 5-bit Phase Resolution

The narrowband spectrum outputs of the two different LUTs are shown in Figure 56. The LUTs each have 32 phase points, but differ in amplitude resolution. One has 5-bit amplitude resolution, whereas, the other has 6-bit amplitude resolution. The results with respect to the performance metrics are presented in Tables 30 and 31. With consideration in the spectrum output illustrated in Figure 56 as well as the results presented in Tables 30 and 31, it can be reinforced that changing the amplitude resolution results in minimal improvement in spurious performance. In Table 30, the total power contained in the spurs are practically equivalent. This provides further insight on the little effect amplitude resolution has on the spurious performance.

Table 30. Narrowband: 32 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions

AR	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
5 bits	-16.73	-81.05	64.31
6 bits	-16.35	-80.74	64.38

This concept is further verified through the analysis of Table 31. In Table 31, the SFDR for both LUTs are presented are within four-tenths of a dB. Hence, changing the amplitude resolution by a small amount does not result in a noticeable performance improvement.

Table 31. Narrowband: 5-bit Phase Resolution: Spurious Free Dynamic Range

Span [kHz]	Metric	5-bit AR	6-bit AR
200	Carrier Power [dBm]	-16.85	-16.46
	Max Spur [dBm]	-98.25	-98.29
	SFDR [dBc]	-81.40	-81.82

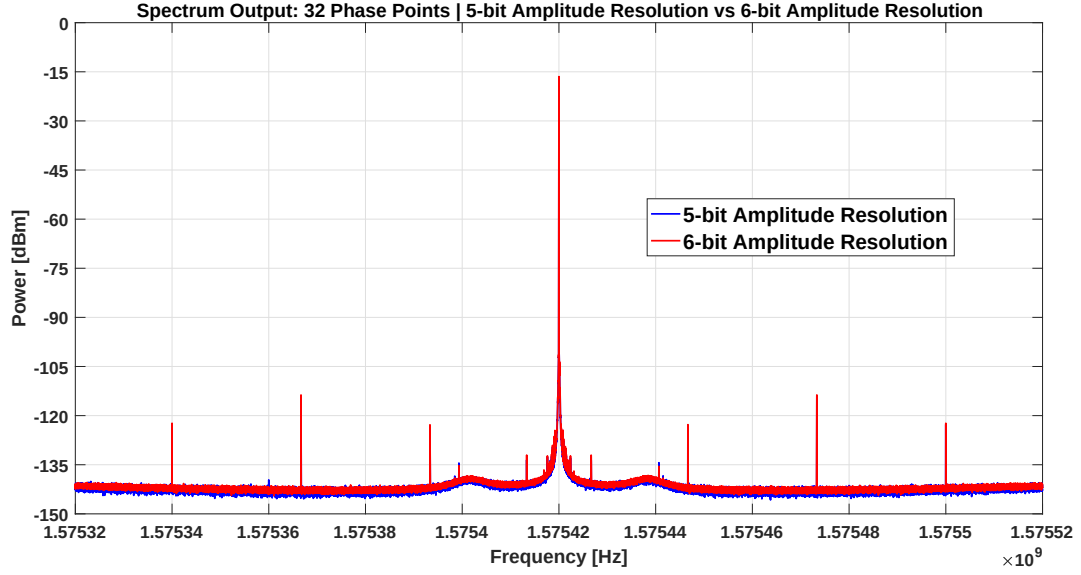


Figure 56. Narrowband Spectrum Output Due To Finite Amplitude Resolution: 5-bit Phase Resolution.

7.2 Narrowband Spurious Performance Due To Finite Amplitude Resolution: 6-bit Phase Resolution

The narrowband spectrum output of the two different LUTs are shown in Figure 57. The LUTs both have 64 points, but differ in the finite amplitude resolution. One LUT implements 5-bit amplitude resolution, whereas, the other implements 6-bit amplitude resolution. The results are shown in Tables 32 and 33. The results in the table verify that for the GWPP architecture, the amplitude resolution as well as the calculated FTW value does not directly impact the spurious performance close to the carrier.

Table 32. Narrowband: 6-bit Phase Resolution, 64 Phase Points: Fundamental Carrier Power vs Power Lost To Spurious Emissions

AR	$Power_{Carrier}$ [dBm]	$Power_{Lost}$ [dBm]	$Power_{CSE}$ [dB]
5 bits	-16.70	-81.15	64.45
6 bits	-16.30	-80.72	64.32

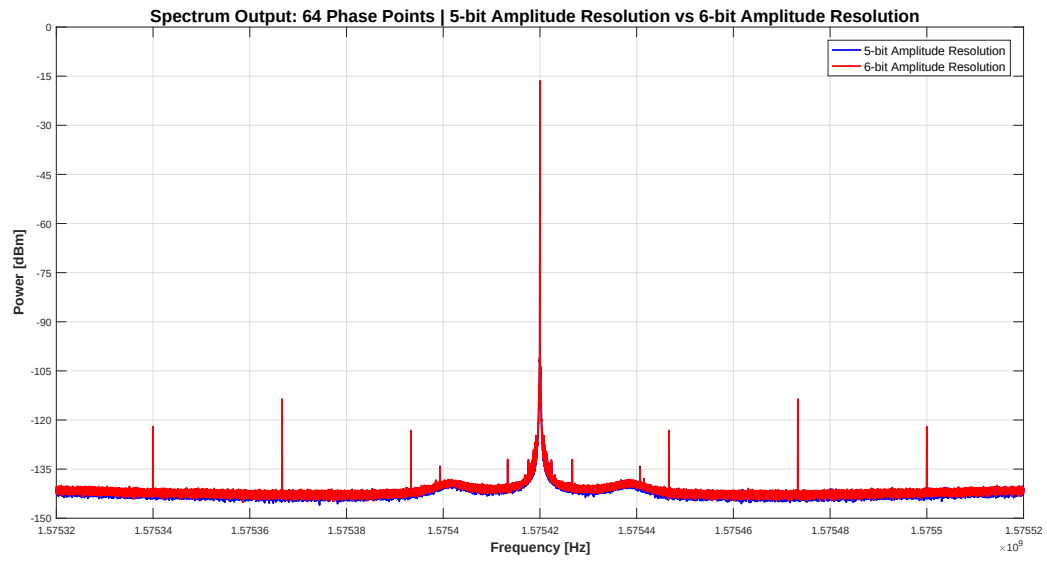


Figure 57. Narrowband Spectrum Output Due To Finite Amplitude Resolution: 6-bit Phase Resolution.

Table 33. Narrowband: 6-bit Phase Resolution: Spurious Free Dynamic Range

Span [kHz]	Metric	5-bit AR	6-bit AR
200	Carrier Power [dBm]	-16.81	-16.42
	Max Spur [dBm]	-98.98	-97.46
	SFDR [dBc]	-82.17	-81.04

Bibliography

1. E. D. Kaplan and C. J. Hegarty, *Understanding GPS Principles and Applications*, 2nd ed. Norwood, MA: Artech House, Inc., 2006.
2. “Fundamentals of GPS Threats,” Tech. Rep., 2016. [Online]. Available: <https://www.spirent.com/-/media/White-Papers/Positioning/Fundamentals-of-GPS-Threats.pdf>
3. P. Misra and P. Enge, *Global Positioning System Signals, Measurements, and Performance*, 2nd ed. Lincoln, MA: Ganga-Jamuna Press, 2006.
4. (2016, Oct) Control Segment. National Coordination Office. [Online]. Available: <http://www.gps.gov/systems/gps/control/>
5. M. J. Dunn, *Interface Specification IS-GPS-200H*, Global Positioning System Directorate, Sept 2013. [Online]. Available: <http://www.gps.gov/technical/icwg/IS-GPS-200H.pdf>
6. B. G. Goldberg, *Digital Frequency Synthesis Demystified*. Eagle Rock, VA: LLH Technology Publishing, 1999.
7. J. Tierney, C. M. Rader, and B. Gold, “A digital frequency synthesizer,” *IEEE Trans. on Audio and Electroacoustics*, vol. 19, pp. 48–57, Mar 1971.
8. R. Cordoso, J. Ribeiro, and M. Silveira, “Direct digital synthesizer Using FPGA,” in *Global Congress on Engineering and Technology Education*, Mar 2005.
9. A. Giuseppe, M. Strollo, D. D. Caro, and P. N. P, “A 630 mhz, 76 mw direct digital frequency synthesizer using enhanced rom compression technique,” *IEEE Journal of Solid State Circuits*, vol. 42, no. 2, pp. 350–360, Feb 2007.
10. (2017, Jan) Virtex-6 FPGA ML605 Evaluation Kit. Xilinx. [Online]. Available: <https://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html>
11. *Virtex-6 Family Overview Product Specification*, Xilinx, Aug 2015. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf
12. Xilinx, *The Programmable Logic Data Book*, 3rd ed. Xilinx, 1994.
13. *Virtex-6 FPGA CLB User Guide*, Xilinx, Feb 2012.
14. *Constraints Guide*, Xilinx, Sep 2010. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx12.3/cgd.pdf

15. S. Gunawardena, "Feasibility study for the implementation of global positioning system block processing techniques in field programmable gate arrays," Master's Thesis, Ohio University, 2000.
16. (2012) Ml605 product brief. Xilinx. [Online]. Available: https://www.xilinx.com/publications/prod_mktg/ml605-product_brief.pdf
17. *FMC230 User Manual*, 4DSP LLC, Sept 2015. [Online]. Available: http://www.4dsp.com/pdf/FMC230_user_manual.pdf
18. *12-Output Clock Generator with Integrated 2.5 GHz VCO Data Sheet AD9517-1*, Analog Devices, Dec 2016.
19. *11-/14-Bit, 5.7 GSPS, RF Digital-to-Analog Converter Data Sheet AD9119/AD9129*, Analog Devices, Dec 2016.
20. *Virtex-6 FPGA SelectIO Resources User Guide*, Xilinx, Nov 2014. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug361.pdf
21. (2017, Feb) Xilinx Website. Xilinx. [Online]. Available: <http://www.xilinx.com>
22. *SA.22c Precision Rubidium Oscillator*, Microsemi, Oct 2016. [Online]. Available: https://www.microsemi.com/document-portal/doc_view/133279-sa-22c
23. J. Vankka, "Direct Digital Synthesizer: Theory, Design, and Applications," Dissertation, Helsinki University of Technology, Nov 2000.
24. N. Sawyer, *Source-Synchronous Serialization and Deserialization (up to 1050 Mb/s)*, Xilinx, Nov 2013. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1064.pdf
25. S. Kempainen, *Low-Voltage Differential Signaling (LVDS)*, National Semiconductor, May 2010. [Online]. Available: <http://cp.literature.agilent.com/litweb/pdf/5988-4797EN.pdf>
26. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms, and Applications*, 4th ed. Pearson Education, Inc., 2007.
27. C. E. Cummings, Ed., *Clock Domain Crossing (CDC) Design & Verification Techniques Using System Verilog*. Boston, MA: SNUG Boston 2008, Apr 2008. [Online]. Available: <http://www.sunburst-design.com/papers/CummingsSNUG2008Boston.CDC.pdf>
28. *KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide*, Texas Instruments, Nov 2010. [Online]. Available: <http://www.ti.com/lit/ug/sprugp1/sprugp1.pdf>

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, 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 Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 23-03-2017		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From — To) Aug 2015 — Mar 2017	
4. TITLE AND SUBTITLE Development of a Software-Defined GNSS Simulation Prototype for Advanced Signals Research				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Ledford, Michael, J., 2LT, USAF				5d. PROJECT NUMBER 16G250/17G250	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENG-MS-17-M-043	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL (Attn: Capt Katherine Carroll, AFRL/RVSE) Comm. Phone: (505) 853-3638 Email: katherine.carroll.3@us.af.mil 3550 Aberdeen Ave SE Bldg 592, Kirtland AFB, NM 87117-5776				10. SPONSOR/MONITOR'S ACRONYM(S) Air Force Research Laboratory (AFRL)/RVSE	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT Global Positioning System (GPS) has become the de-facto standard for precision time transfer. By definition, all signal structure details of the open civilian GPS signals are known and published. This makes them vulnerable to numerous threats One possible solution is to incorporate signal authentication features for future GPS. However, there lacks a hardware platform to verify these features. This thesis proposes a Field Programmable Gate Array (FPGA)-based GPS waveform prototyping platform that features software-defined all-digital signal synthesis architecture in conjunction with a high-speed Digital to Analog Converter (DAC) as a solution. This work explores synthesizing the GPS L1 C/A signal to analyze the spectral performance of this platform with consideration in FPGA resources.					
15. SUBJECT TERMS Global Positioning System, GPS, Global Navigation Satellite System, GNSS, Field Programmable Gate Array, FPGA, Software-Defined, Direct Digital Synthesis, DDS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Richard K. Martin, AFIT/ENG
U	U	U	UU	209	19b. TELEPHONE NUMBER (include area code) (937) 255-3636, x4625; richard.martin@afit.edu