Master thesis in Electronics and Computer Science, Instrumentation

A Survey of DDS Implementations for RIMFAX

Kim Østenfor Spildrejorde

Department of Physics
Faculty of Mathematics and Natural Science
UNIVERSITY OF OSLO

01. December 2016
Abstract

RIMFAX is a ground penetrating radar currently under development by The Norwegian Defence Research Establishment (FFI). The radar signal is a sine wave that will be generated digitally by a Direct Digital Synthesizer, which is implemented on a FPGA. A sine wave DDS basic building blocks are a phase accumulator and a ROM of stored sine wave samples. The ROM is potentially very resource intensive and methods exist to reduce the resource consumption of a DDS on a FPGS. This thesis explores two methods for reducing consumption. These have been implemented on a Kintex-7 FPGA, in addition to a IP based DDS from the Vivado Design Suite and a DDS resembling the current RIMFAX prototype. Output signal has been characterized using a spectrum analyzer, for various DDS configurations adjusting bit lengths of output amplitude, phase address bits used internally and the input word defining the output frequency, for characterizing changes in behaviour, set up against a requirement of having integrated phase noise performance of -80 dBC or better for the frequency offset range 1 kHz - 30 kHz. Best performance with regards to both resource utilization and output characteristics was the Nicholas Compression DDS, which offered phase noise within requirements and had the lowest resource utilization.
Acknowledgements

The work in this thesis was conducted in the time period from September 2014 to December 2016, under the supervision of RIMFAX Principal Investigator Svein-Erik Hamran at The Norwegian Defence Research Establishment and the Nanoelectronics group, and associate professor Ketil Røed at the Electronics group at the University of Oslo.

I would like to thank Svein-Erik Hamran for the opportunity to write my thesis on such an exciting subject, fulfilling a goal that stretches many years back, of getting to work on a project related to space flight. I have learned a great deal over the last two years. Your input and guidance with regards to DDS - and radar theory has been greatly appreciated and has been an essential part of getting this thesis in the right direction.

I would also like to thank Ketil Røed for his help and guidance. Your help in reviewing and providing constructive feedback throughout the process has been invaluable and I am grateful for all our discussions, helping me to ask the right questions in order to continuously improve my work and understanding.

A big thank you as well to Mats Jørgen Øyan of the RIMFAX team The Norwegian Defence Research Establishment, for all technical guidance throughout this process, always being available for questions and for many good conversations and laughs, helping to keep up motivation.

Marius, Vegard and Ivar; thank you for all your support and for all the fun during these years of studying together. Also, thank you to everyone at 333 and at SEF for all the good memories.

Finally, thank you to my wonderful girlfriend Helen for your unwavering support and encouragement during my work on this thesis.
Contents

1 Introduction 1
  1.1 Background ........................................ 1
    1.1.1 NASA ......................................... 1
    1.1.2 FFI and Radar Research ....................... 2
    1.1.3 Mars .......................................... 3
    1.1.4 Robotic Exploration of Mars ................... 3
    1.1.5 Mars 2020 .................................... 6
    1.1.6 RIMFAX ...................................... 6
  1.2 Motivation ........................................ 8
  1.3 Goals of this thesis ............................... 9
  1.4 Outline .......................................... 9

2 DDS and Radar Technology 11
  2.1 Direct Digital Synthesizers ....................... 11
    2.1.1 Basic DDS structure .......................... 11
    2.1.2 Sine wave look-up table ...................... 13
  2.2 Alternatives to DDS .............................. 15
    2.2.1 Phase Locked Loop ............................ 15
  2.3 FPGA ............................................ 16
    2.3.1 FPGAs in space .............................. 17
    2.3.2 VHDL ........................................ 17
  2.4 DDS in Hardware .................................. 18
    2.4.1 CMOS DDS .................................... 18
    2.4.2 DDS in FPGA ................................ 18
  2.5 Radar ............................................. 19
    2.5.1 Ground Penetrating Radar ..................... 20
    2.5.2 Frequency Modulated Continuous Wave Radar .. 20

3 Methods of Resource Compression in FPGA-based DDS 25
  3.1 Overview ........................................ 25
  3.2 Reducing phase accumulator size ................... 25
    3.2.1 Sine wave Look-Up Table ..................... 26
  3.3 Compression methods and algorithms ............... 26
    3.3.1 Quarter Sine Symmetry ....................... 26
    3.3.2 Sunderland compression ...................... 27
    3.3.3 Nicholas’ compression algorithm ............. 29
    3.3.4 Phase Difference Algorithm .................. 31
3.3.5 Kenji Terai Algorithm .............................. 32

4 Sources of noise in DDS ............................... 35
  4.1 Phase noise ........................................ 35
  4.1.1 RMS Phase error and Jitter .................... 37
  4.2 Spurious noise .................................... 39
  4.2.1 Spurs due to phase bit truncation ............... 39
  4.3 DAC noise and aliasing .............................. 42
  4.3.1 Aliasing ....................................... 43
  4.4 Quantization noise and finite precision of sine samples ... 43

5 Implementation ......................................... 45
  5.1 Hardware ............................................ 45
     5.1.1 Kintex 7 KC705 Evaluation Board ............... 45
     5.1.2 Texas Instruments DAC5675A .................... 45
  5.2 Phase accumulator structure ......................... 46
  5.3 Compression of Look-Up Table ....................... 46
     5.3.1 Matlab ....................................... 48
     5.3.2 VHDL Code Overview ........................... 48
     5.3.3 Sunderland and Nicholas DDS VHDL-file ........ 51
     5.3.4 Phase accumulator implementation .............. 52
  5.4 Quarter Sine DDS ................................... 53
  5.5 LogiCORE IP DDS .................................... 53
  5.6 Generating sine wave look-up tables ................. 53
  5.7 Test 1: Amplitude bit length test ................... 54
  5.8 Adjusting sine ROM address length .................. 56
  5.9 Adjusting the Frequency Tuning Word ................. 57

6 Tests and results ........................................ 61
  6.1 Test setup .......................................... 61
  6.2 Output analysis ..................................... 61
     6.2.1 Test outline .................................. 61
  6.3 Test scenario 1: Varying output amplitude bit length ... 64
  6.4 Test 1: Quarter Sine DDS ............................. 65
     6.4.1 4 bit output .................................. 65
     6.4.2 8 bit output .................................. 66
     6.4.3 14 bit output and 18 bit output ................ 67
     6.4.4 Phase Noise analysis ......................... 69
  6.5 Test 1: LogiCORE IP DDS ............................. 71
     6.5.1 4 bit amplitude ............................... 71
     6.5.2 8 bit amplitude ............................... 72
     6.5.3 14 bit ......................................... 73
     6.5.4 18 bit ......................................... 73
     6.5.5 Phase noise analysis ........................... 74
  6.6 Test 1: Sunderland compression DDS ................... 74
     6.6.1 Phase noise analysis ........................... 76
  6.7 Test 1: Nicholas Compression DDS ..................... 79
     6.7.1 4 bit and 8 bit amplitude ...................... 79
7 Summary and conclusion 107
  7.1 Summary of work done . . . . . . . . . . . . . . . . . . . . 107
  7.2 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . 108
  7.3 Future work . . . . . . . . . . . . . . . . . . . . . . . . . . 108

Appendices 115

A Permissions 117

B Matlab Code 119
  B.1 Script Generating Quarter Sine ROM . . . . . . . . . . . 119
  B.2 Script Generating Sunderland Compression ROMs . . . . 121
  B.3 Script Generating Nicholas Compression ROMs . . . . . . 125
  B.4 Phase Noise Calculation Function . . . . . . . . . . . . . . 129
  B.5 Frequency Spectrum Plotting . . . . . . . . . . . . . . . . . 131

C VHDL Code 133
  C.1 Top Level . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
  C.2 Phase Accumulator . . . . . . . . . . . . . . . . . . . . . . . 135
  C.3 DDS (Sunderland/Nicholas) . . . . . . . . . . . . . . . . . . 137
  C.4 DDS (Quarter Sine Symmetry) . . . . . . . . . . . . . . . . 142

D LogiCORE IP DDS Block Schematic 147

E Test Result Graphs 149
  E.1 Test 1: Varying Amplitude Bit Length . . . . . . . . . . . 149
    E.1.1 Quarter Sine DDS . . . . . . . . . . . . . . . . . . . . 149
    E.1.2 LogiCORE IP DDS . . . . . . . . . . . . . . . . . . . . 149
    E.1.3 Sunderland Compression DDS . . . . . . . . . . . . . 150
    E.1.4 Nicholas Compression DDS . . . . . . . . . . . . . . . 152
  E.2 Test 2: Varying LUT Address Bit Length . . . . . . . . . . . 156
    E.2.1 Quarter Sine DDS . . . . . . . . . . . . . . . . . . . . 156
    E.2.2 LogiCORE IP DDS . . . . . . . . . . . . . . . . . . . . 157
    E.2.3 Sunderland Compression DDS . . . . . . . . . . . . . 158
    E.2.4 Nicholas Compression DDS . . . . . . . . . . . . . . . 159
  E.3 Test 3: Varying FTW Bit Length . . . . . . . . . . . . . . . . 163
    E.3.1 Quarter Sine DDS . . . . . . . . . . . . . . . . . . . . 163
    E.3.2 LogiCORE IP DDS . . . . . . . . . . . . . . . . . . . . 164
    E.3.3 Sunderland Compression DDS . . . . . . . . . . . . . 165
    E.3.4 Nicholas Compression DDS . . . . . . . . . . . . . . . 168
List of Figures

1.1 Pictured is the Vehicle Assembly Building at NASA’s Kennedy Space Center in Florida. Kennedy Space Center and the adjacent Cape Canaveral Air Force Station has launched all American manned space flights and the majority of unmanned scientific probes. *Photo Credit: NASA.*

2

1.2 Mars, the fourth planet of the Solar system. It has fascinated mankind for thousands of years, but so far has not revealed any current or former life forms that people in the past thought might exist on the planet. However new science has indicated that conditions once might have supported life. Today though, the planet is a barren wasteland with no magnetic field, a thin atmosphere and low surface temperatures. *Photo Credit: NASA.*

4

1.3 A comparison of all Mars rovers sent by NASA. To the left we have the Mars Exploration Rover, of which two were sent (Opportunity and Spirit). They landed in 2004 and Opportunity is still functioning, while Spirit was lost in 2010. In the lower left corner is the Sojourner rover, which was deployed from the Pathfinder lander in 1997. The 2012 Mars Science Laboratory - known as Curiosity, is pictured on the right. This is the most advanced and largest rover yet. The Mars 2020-rover will be structurally identical to Curiosity, but with new instruments. *Image Credit: NASA.*

5

1.4 Schematic of the Mars 2020-rover and the placement of the seven science instruments. The antenna of RIMFAX is installed in the rear of the rover, with the electronics adjacent to the Radioactive Thermal Generator (RTG), a plutonium based power generator.

7

2.1 The basic blocks of a DDS include a Phase Accumulator, consisting of a register and an adder summing the accumulated phase with the FTW every clock cycle, and a Phase-to-Sine Look-up table which store sine amplitude samples. The output from the phase register functions as an address to the corresponding look-up table sine sample. The sine sample is then sent to a Digital-to-analog converter.

12
2.2 Phase circle showing the possible phase values of a 4-bit phase accumulator. ........................................ 13

2.3 Phase circle showing the possible phase values of a 5-bit phase accumulator. Compared to figure 2.2, we now see that by increasing the accumulator size by a single bit, the minimum phase increment is reduced by half, also doubling frequency resolution (see equation 2.2) .................. 14

2.4 Basic building blocks in a PLL. ©D. Banerjee [14]. See appendix A for permit from author and publisher for reuse of figures. .......................................................... 15

2.5 Graph illustrating the concept of a FMCW radar. The transmitted sweep is shown in blue, with the received, reflected sweep shown in red. The dashed lines over and under the solid lines indicate the frequencies $f_{start}$ and $f_{stop}$ respectively. The receiver mixes these signals, which creates two new signals due to the heterodyne principle[21]: One signal with frequency equal to the sum of the two signals, and one signal with frequency equal to the difference. The former is filtered, using a low-pass filter, while the difference signal is processed to produce the radar image. This signal is referenced in [22] as the Intermediate Frequency (IF)-signal. 22

2.6 Illustration of the main components in a FMCW radar. A variable frequency signal generator transmits a signal, which is reflected and received by the Rx-antenna. The received signal is a time delayed version of the transmitted signal, which also leads to a frequency difference. These signals are mixed, creating a signal with frequency equal to the sum and one signal with frequency equal to the difference. The sum signal is filtered using a low pass filter. ......................... 23

3.1 As shown in this figure, each quarter of the sine wave is symmetrical. We can therefore reduce the look-up table size by a factor of four, by simply storing the first quarter, and generating the other three by manipulating the address - and output signals. ................................................ 27

3.2 This figure displays a block schematic of how a DDS utilizing either Sunderland’s compression method [23] or Nicholas’ compression method [25], the latter of which is largely similar to the first, but differs in how the sine samples are selected. ........................................... 28

3.3 Figure demonstrating the concept of Nicholas’ compression algorithm. Here shown with 6 bit phase and segmentation of 2 bits each for A, B and C. (©1988 IEEE. Permitted for thesis reuse.) .............................. 30
3.4 Block schematic of the phase difference algorithm proposed by Nicholas in [25]. The truncated phase bits are added to the final LUT-output, to produce the output sine sample. Compared to Sunderland - and Nicholas’ compression this requires only a single extra adder in terms of extra resources. 32

4.1 Graph illustrating the concept of integrating phase noise between two frequencies. © D. Banerjee Originally published in [14], and used with permission from the author and publisher (see appendix A). 36

5.1 Figure showing the entity of the phase accumulator. The output is used for addressing the sine ROM, after being truncated to appropriate length. 46

5.2 Figure showing the entity of the top level file. Inputs are the two differential clock signals, the reset signal and the FTW switch signal. The output of the top level file is the final sine sample which is then passed to the DAC. 50

5.3 Illustration of the DDS entity. The inputs include a clock signal, reset, an enable signal (always held high) and the selected FTW, which is selected in a process in the top level file. Output is the final sine sample, which is passed to a top level register before being sent to the DAC. 51

6.1 Block schematic showing an outline of the test configuration for all tests conducted in this thesis. Not shown is the low-pass filter which was connected directly between the wire from the DAC and the spectrum analyzer. 62

6.2 Frequency spectra for Quarter Sine Symmetry DDS at 4 bits amplitude length. 66

6.3 Frequency spectra for Quarter Sine Symmetry DDS at 8 bits amplitude length. 67

6.4 Frequency spectra for Quarter Sine Symmetry DDS at 14 bits amplitude length. 68

6.5 Frequency spectra for Quarter Sine Symmetry DDS at 18 bits amplitude length. 69

6.6 Frequency spectra for IP DDS. Amplitude bit length is 4 bits. Note the similarities to the Quarter Sine DDS in figure 6.6. 71

6.7 Frequency spectra for IP DDS. Amplitude bit length is 8 bits. 72

6.8 Frequency spectra for IP DDS. Amplitude bit length is 14 bits. 73

6.9 Frequency spectra for DDS using Sunderland’s compression method. Amplitude bit length is 14 bits. 76

6.10 These plots show measured phase noise in the range 1 kHz - 30 kHz for all test frequencies using the Sunderland Compression DDS, with amplitude bits in the range 4, 8, 14 and 18 bits. 77
6.11 These plots show emulated phase noise in the IF signal after correction factor has been applied. Frequency offset is in the range 1 kHz - 30 kHz for all test frequencies. Sunderland Compression DDS, with amplitude bits in the range 4, 8, 14 and 18 bits. ................................. 78

6.12 This figure shows frequency spectrum for all four test frequencies, for Nicholas Compression DDS with 14 bit amplitude. .................................................. 80

6.13 Frequency Spectrum for Quarter Sine DDS for Test 2, with 9 phase bits for addressing the sine ROM. .................. 82

6.14 Frequency Spectrum for Quarter Sine DDS for Test 2, with 12 phase bits for addressing the sine ROM. .................. 83

6.15 Frequency Spectrum for LogiCORE DDS IP for Test 2, with 9 phase bits for addressing the sine ROM. .................. 85

6.16 Frequency Spectrum for Nicholas Compression DDS for Test 2, with 9 phase bits for addressing the sine ROM. ........ 89

6.17 Frequency Spectrum for Nicholas Compression DDS for Test 2, with 12 phase bits for addressing the sine ROM. ........ 90

6.18 Frequency Spectrum for Quarter Sine DDS for Test 3 with 18 bit FTW. ..................................................... 93

6.19 Frequency Spectrum for Quarter Sine DDS for Test 3, with 32 bit FTW. ..................................................... 94

6.20 Frequency Spectrum for the IP DDS for Test 3, with 18 bit FTW. 96

6.21 Frequency Spectrum for the IP DDS for Test 3, with 18 bit FTW. 97

6.22 Screenshot from spectrum analyzer showing Sunderland DDS with 9 bit phase, at 30 MHz output frequency, 100 MHz measurement span, with resolution bandwidth at 200 kHz. . 102

6.23 Illustration of a sine wave with inverted sign bit. ....... 103

6.24 The plot on the left shows the measured phase noise at $f = 18.75$ MHz, for varying amplitude bit lengths using a Quarter Sine DDS. Take note of the consistency between the four configurations, apparently unaffected by the change in amplitude bit length. To the right the same measurement after being multiplied with the correction factor for close targets. ................................. 104

6.25 The plot on the left shows the measured phase noise at $f = 18.75$ MHz, for varying phase bit lengths using a Nicholas Compression DDS. Phase noise appears comparatively equal, and also do not diverge from the Quarter Sine DDS plots in figure 6.24. To the right the same measurement after being multiplied with the correction factor for close targets. ................................. 105

E.1 Frequency spectra for Quarter Sine DDS with 4 bit amplitude output ..................................................... 150

E.2 Frequency spectra for Quarter Sine DDS with 8 bit amplitude output ..................................................... 151
E.3 Frequency spectra for Quarter Sine DDS with 14 bit amplitude output ................................. 152
E.4 Frequency spectra for Quarter Sine DDS with 18 bit amplitude output ................................. 153
E.5 Phase noise measurements of Quarter Sine DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. ................... 154
E.6 Phase noise measurements of Quarter Sine DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing. ................... 155
E.7 Frequency spectra for LogiCORE IP DDS, 4 bit amplitude output ........................................ 156
E.8 Frequency spectra for LogiCORE IP DDS, 8 bit amplitude output ........................................ 157
E.9 Frequency spectra for LogiCORE IP DDS, 14 bit amplitude output ........................................ 158
E.10 Frequency spectra for LogiCORE IP DDS, 18 bit amplitude output ....................................... 159
E.11 Phase noise measurements of LogiCORE IP DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. 160
E.12 Phase noise measurements of LogiCORE IP DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing. ........... 161
E.13 Frequency spectra for Sunderland Compression DDS, 4 bit amplitude output ......................... 162
E.14 Frequency spectra for Sunderland Compression DDS, 8 bit amplitude output ......................... 163
E.15 Frequency spectra for Sunderland Compression DDS, 14 bit amplitude output ....................... 164
E.16 Frequency spectra for Sunderland Compression DDS, 18 bit amplitude output ....................... 165
E.17 Phase noise measurements of Sunderland Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. 166
E.18 Phase noise measurements of Sunderland Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing. 167
E.19 Frequency spectra for Nicholas Compression DDS, 4 bit amplitude output ............................. 168
E.20 Frequency spectra for Nicholas Compression DDS, 8 bit amplitude output ............................. 169
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.21</td>
<td>Frequency spectra for Nicholas Compression DDS, 14 bit amplitude output</td>
<td>170</td>
</tr>
<tr>
<td>E.22</td>
<td>Frequency spectra for Nicholas Compression DDS, 18 bit amplitude output</td>
<td>171</td>
</tr>
<tr>
<td>E.23</td>
<td>Phase noise measurements of Nicholas Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz.</td>
<td>172</td>
</tr>
<tr>
<td>E.24</td>
<td>Phase noise measurements of Nicholas Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing.</td>
<td>173</td>
</tr>
<tr>
<td>E.25</td>
<td>Quarter Sine Symmetry DDS with 9 phase bits for sine ROM addressing</td>
<td>174</td>
</tr>
<tr>
<td>E.26</td>
<td>Quarter Sine Symmetry DDS with 12 phase bits for sine ROM addressing</td>
<td>175</td>
</tr>
<tr>
<td>E.27</td>
<td>Quarter Sine Symmetry DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for Quarter Sine Symmetry DDS with 9, 12 and 15 phase bits used for sine ROM addressing.</td>
<td>176</td>
</tr>
<tr>
<td>E.28</td>
<td>Quarter Sine Symmetry DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz.</td>
<td>177</td>
</tr>
<tr>
<td>E.29</td>
<td>Frequency spectra for LogiCORE IP DDS with 9 phase bits used for sine ROM addressing</td>
<td>178</td>
</tr>
<tr>
<td>E.30</td>
<td>Frequency spectra for LogiCORE IP DDS with 12 phase bits used for sine ROM addressing</td>
<td>179</td>
</tr>
<tr>
<td>E.31</td>
<td>LogiCORE IP DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 9, 12 and 15 phase bits used for sine ROM addressing.</td>
<td>180</td>
</tr>
<tr>
<td>E.32</td>
<td>LogiCORE IP DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz.</td>
<td>181</td>
</tr>
<tr>
<td>E.33</td>
<td>Frequency spectra for Sunderland Compression DDS with 9 phase bits for sine ROM addressing.</td>
<td>182</td>
</tr>
<tr>
<td>E.34</td>
<td>Frequency spectra for Sunderland Compression DDS with 12 phase bits for sine ROM addressing.</td>
<td>183</td>
</tr>
<tr>
<td>E.35</td>
<td>Sunderland Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 9, 12 and 15 phase bits used for sine ROM addressing.</td>
<td>184</td>
</tr>
</tbody>
</table>
E.36 Sunderland Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz. 185

E.37 Frequency spectra for Nicholas Compression DDS with 9 phase bits for sine ROM addressing. 186

E.38 Frequency spectra for Nicholas Compression DDS with 12 phase bits for sine ROM addressing. 187

E.39 Nicholas Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 9, 12 and 15 phase bits used for sine ROM addressing. 188

E.40 Nicholas Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz. 189

E.41 Quarter Sine Symmetry DDS with 18 bit FTW. 190

E.42 Quarter Sine Symmetry DDS with 32 bit FTW. 190

E.43 Quarter Sine Symmetry DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for Quarter Sine Symmetry DDS with 18, 32 and 48 bit FTW. 191

E.44 Quarter Sine Symmetry DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit FTW. Frequency range 1 kHz - 30 kHz. 192

E.45 Frequency spectra for LogiCORE IP DDS with 18 bit FTW. 193

E.46 Frequency spectra for LogiCORE IP DDS with 32 bit FTW. 193

E.47 LogiCORE IP DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 18,32 and 48 bit FTW. 194

E.48 LogiCORE IP DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18,3 2 and 48 bit FTW. Frequency range 1 kHz - 30 kHz. 195

E.49 Frequency spectra for Sunderland Compression DDS with 9 phase bits for sine ROM addressing. 196

E.50 Frequency spectra for Sunderland Compression DDS with 12 phase bits for sine ROM addressing. 197

E.51 Sunderland Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 18, 32 and 48 bit FTW. 198

E.52 Sunderland Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit FTW. Frequency range 1 kHz - 30 kHz. 199
E.53 Frequency spectra for Nicholas Compression DDS with 18 bit FTW. ................................. 200
E.54 Frequency spectra for Nicholas Compression DDS with 32 bit FTW ................................. 201
E.55 Nicholas Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for Nicholas Compression DDS with 18, 32 and 48 bit FTW. .............. 202
E.56 Nicholas Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit Frequency Tuning Word. Frequency range 1 kHz - 30 kHz. ........................................... 203
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Table demonstrating the function of the two most significant bits of the phase output in a DDS utilizing quarter sine wave symmetry compression.</td>
</tr>
<tr>
<td>5.1</td>
<td>My caption</td>
</tr>
<tr>
<td>5.2</td>
<td>My caption</td>
</tr>
<tr>
<td>5.3</td>
<td>My caption</td>
</tr>
<tr>
<td>5.4</td>
<td>Table showing the test parameters for Test 1: Varying amplitude bit length.</td>
</tr>
<tr>
<td>5.5</td>
<td>Sine ROM size for Quarter Sine DDS with 15 address bits, and amplitude bits varying from 4-18 bits.</td>
</tr>
<tr>
<td>5.6</td>
<td>Number of bits required to store sine ROMs for Sunderland Compression DDS for varying amplitude bit lengths, and 15 address bits.</td>
</tr>
<tr>
<td>5.7</td>
<td>Number of bits required to store sine ROMs for Nicholas Compression DDS for varying amplitude bit lengths, and 15 address bits.</td>
</tr>
<tr>
<td>5.8</td>
<td>Resource utilization as reported by Vivado post-implementation for Test 1: Varying amplitude bit lengths.</td>
</tr>
<tr>
<td>5.9</td>
<td>Parameters for Test 2: Varying phase address bit length for sine ROM addressing.</td>
</tr>
<tr>
<td>5.10</td>
<td>Sine ROM size for Quarter Sine DDS with varying phase address bits, and 14 amplitude bits (13 sine ROM amplitude bits, 1 sign bit from phase accumulator)</td>
</tr>
<tr>
<td>5.11</td>
<td>My caption</td>
</tr>
<tr>
<td>5.12</td>
<td>Sine ROM size in no of bits for Nicholas Compression DDS for Test 2: Varying phase address bit lengths.</td>
</tr>
<tr>
<td>5.13</td>
<td>Resource utilization as reported by Vivado for Test Scenario 2, adjusting sine ROM address bit length. As is clearly evident from the table, Nicholas compression offers the least utilization, which is in accordance with previously described theory.</td>
</tr>
<tr>
<td>6.1</td>
<td>Worst case Carrier-to-spur ratio based on amplitude bit length. Valid for all DDS implementations in Test 1</td>
</tr>
</tbody>
</table>
6.2 Test 1: Table showing integrated phase noise for Quarter Sine DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.3 Test 1: Table showing integrated phase noise for Quarter Sine DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1 kHz-30 kHz.  
6.4 Test 1: Table showing integrated phase noise for LogiCORE IP DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.5 Test 1: Table showing integrated phase noise for LogiCORE IP DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.6 Test 1: Table showing integrated phase noise for Sunderland Compression, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.7 Test 1: Table showing integrated phase noise for Sunderland Compression, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.8 Test 1: Table showing integrated phase noise for Nicholas Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.9 Test 1: Table showing integrated phase noise for Nicholas Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.10 Test 2: Table showing integrated phase noise for Quarter Sine DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.11 Test 2: Table showing integrated phase noise for Quarter Sine DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.12 Test 2: Table showing integrated phase noise for LogiCORE DDS IP, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.  
6.13 Test 2: Table showing integrated phase noise for LogiCORE DDS IP, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.
6.14 Test 2: Table showing integrated phase noise for Sunderland Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . 88
6.15 Test 2: Table showing integrated phase noise for Sunderland Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . . . . . . . . . . . . . . . . . . . . . . 88
6.16 Test 2: Table showing integrated phase noise for Nicholas Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . 91
6.17 Test 2: Table showing integrated phase noise for Nicholas Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . . . . . . . . . . . . . . . . . . . . . . 91
6.18 Test 3: Table showing integrated phase noise for Quarter Sine DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . 94
6.19 Test 3: Table showing integrated phase noise for Quarter Sine DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . . . . . . . . . . . . . . . . . . . . . . 95
6.20 Test 3: Table showing integrated phase noise for Sunderland Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . 98
6.21 Test 2: Table showing integrated phase noise for Sunderland Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . . . . . . . . . . . . . . . . . . . . . . 98
6.22 Test 3: Table showing integrated phase noise for Nicholas Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . 99
6.23 Test 3: Table showing integrated phase noise for Nicholas Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz. . . . . . . . . . . . . . . . . . . . . . . . 100
## Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_{ck} )</td>
<td>System Clock Frequency</td>
</tr>
<tr>
<td>( j )</td>
<td>FTW bit length</td>
</tr>
<tr>
<td>( k )</td>
<td>Number of sine ROM address bits</td>
</tr>
<tr>
<td>( m )</td>
<td>Number of sine ROM amplitude bits</td>
</tr>
<tr>
<td>BRAM</td>
<td>Block RAM</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to analog converter</td>
</tr>
<tr>
<td>dBc</td>
<td>Decibels relative to Carrier</td>
</tr>
<tr>
<td>dBm</td>
<td>Decibels relative to 1 mW</td>
</tr>
<tr>
<td>DDS</td>
<td>Direct Digital Synthesizer</td>
</tr>
<tr>
<td>FF</td>
<td>Flip-Flop</td>
</tr>
<tr>
<td>FFI</td>
<td>The Norwegian Defence Research Establishment</td>
</tr>
<tr>
<td>FMCW</td>
<td>Frequency Modulated Continuous Wave</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>FTW</td>
<td>Frequency Tuning Word</td>
</tr>
<tr>
<td>IF-signal</td>
<td>Intermediate Frequency Signal</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit</td>
</tr>
<tr>
<td>LUT</td>
<td>Look-Up Table</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautic and Space Administration</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase-Locked Loop</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RIMFAX</td>
<td>Radar IMager For MARs subsurface eXperiment</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
</tbody>
</table>
ROM  Read-Only Memory
SFDR  Spurious Free Dynamic Range
SNR  Signal-to-noise Ratio
Chapter 1

Introduction

This thesis describes the design and implementation of a Direct Digital Synthesizer for generating sine waves. The has been envisioned as part of the RIMFAX ground penetrating radar currently in development at The Norwegian Defence Research Establishment (NFR), and has been implemented on a FPGA. RIMFAX is one of nine instruments for NASA’s next Mars Rover, currently known as Mars 2020. Project management and operations are conducted at the Jet Propulsion Laboratory in Pasadena, California.

A DDS consists of two main blocks, a phase accumulator and a look-up table containing sine wave amplitude values. The FPGA resource consumption of a DDS can be considerable, with especially the look-up table claiming a large amount of resources. Objectives in this thesis revolve around exploring concepts for reducing resource consumption, and characterising the output signal of these alternative DDS implementations. The signal characteristics are then compared to an IP based DDS design provided by Xilinx as well as a standard non-resource optimized version of a DDS.

The RIMFAX DDS operates in the range of 18.75 MHz up to 37.5 MHz. Two approaches for reducing look-up table size have been designed and implemented on a Kintex-7 KC705 Evaluation Board, while the output characteristics were determined by spectrum analyzer and Matlab calculations.

1.1 Background

During the following section, we will give a short introduction to the institutions involved in RIMFAX and a summary of rover exploration of Mars as of today.

1.1.1 NASA

The National Aeronautics and Space Administration is the United States’ government agency responsible for aeronautical research, space technology,
Earth - and space science, and human space exploration. [1, 2] Founded in 1958 by President Dwight D. Eisenhower, the administration has a remarkable number of achievements during its history, being responsible for sending 12 men to the moon during the Apollo-program, the Space Transportation System (commonly known as the Space Shuttle) and having visited all planets in the solar system by robotic, unmanned orbiters and landers. Its future projects include the Mars 2020 rover and new manned space vehicles which one day will support manned missions to Mars, as well as support of new commercial providers of space access, among other places from the Kennedy Space Center in Florida, see figure 1.1.1.

1.1.2 FFI and Radar Research

The Norwegian Defence Research Establishment (Forsvarets forskningsinstitutt, FFI) was founded on 11th April 1946, when the Norwegian parliament Storting unanimously approved the establishing of a Defence Research Institute [3]. Five divisions were initially formed and spread across the country. Department R, the Radar Department, was located in Bergen.

Already in 1947 some of the departments relocated to Kjeller, which now is the sole location of the FFI, except for a research facility at Karljohansvern in Horten. The Radar Department was relocated when the new Electronics

\[1\] NASA images are permitted for republishing as long as NASA is credited.
Building was finished in 1962.

While the name suggested the department focused on radars, at first they actually largely focused on telecommunications, as this was deemed more relevant for Norwegian interests than radar technology, which in the mid-40’s was already well understood by both British and American researchers.

Radar research was again at the forefront starting in the mid-70s with Norway becoming an oil nation, something that spurred interest in satellite based surveillance of Norwegian territories[4].

Apart from the long range radars on satellites, FFI now also possess extensive experience in ground penetrating radars. These radars have previously been used by FFI on Svalbard to map geological - and glacial structures. Using this experience, a team led by Svein-Erik Hamran at FFI developed a proposal for a ground penetrating radar for use on the Mars 2020-rover.

1.1.3 Mars

Mars is the fourth planet of the Solar system and one of Earth’s neighbour planets. The planet has been the source of curiosity and wonder from ancient times and was long thought to harbour life and civilisations of its own. With the advent of modern astronomy it quickly became clear that Mars was far from the oasis of life imagined over the centuries. The barren, desert-like landscape, as seen in figure 1.2, was quickly deemed inhospitable to any life forms.

1.1.4 Robotic Exploration of Mars

While the main focus of the public during the Space Race of the 60’s was the manned flights that eventually led to the moon landings, races were also underway to be the first country to send probes to the planets of the solar system - Mars included. The first probe to arrive safely at Mars was Mariner 4 [5] which flew by the planet on July 14. 1965, taking the first ever close-up photos of another planet. Other major discoveries were the absence of a planetary magnetic field - and radiation belts, further reducing expectations of life on the planet. Other American probes that successfully flew by Mars during this decade were Mariner 6, 7 and 9, the latter also getting the distinction of being the first man made object to enter orbit of another planet. The Soviet Union fared worse in their attempts to get to Mars. During the 1960’s, all probes either were destroyed during launch or failed on their way to Mars.

The first probe to actually perform a successful landing on Mars was Viking 1 in 1976. This constituted a big leap in both technical skills regard-
Figure 1.2: Mars, the fourth planet of the Solar system. It has fascinated mankind for thousands of years, but so far has not revealed any current or former life forms that people in the past thought might exist on the planet. However new science has indicated that conditions once might have supported life. Today though, the planet is a barren wasteland with no magnetic field, a thin atmosphere and low surface temperatures. Photo Credit: NASA
ing unmanned spacecraft, but even more when it came to knowledge about the planet Mars.

Viking was stationary and had no means of travel. This changed with Pathfinder which landed on Mars in 1997[6]. In addition to being a stationary lander, it also contained a small rover called Sojourner. It was not in any way a large vehicle, around the size of a microwave oven. But it still provided for increased flexibility in the scientific possibilities around the landing site.

Figure 1.3: A comparison of all Mars rovers sent by NASA. To the left we have the Mars Exploration Rover, of which two were sent (Opportunity and Spirit). They landed in 2004 and Opportunity is still functioning, while Spirit was lost in 2010. In the lower left corner is the Sojourner rover, which was deployed from the Pathfinder lander in 1997. The 2012 Mars Science Laboratory - known as Curiosity, is pictured on the right. This is the most advanced and largest rover yet. The Mars 2020-rover will be structurally identical to Curiosity, but with new instruments. Image Credit: NASA

Opportunity and Spirit landed on Mars in 2004 and ushered in a new era of Mars rover science. The two rovers were slated to conduct a 90 day mission. However Opportunity is still functioning, while Spirit was officially declared lost in 2011 after no communication had been received since the previous year. A harsh Martian winter likely damaged on board electronics to the point where the rover’s systems stopped functioning [7]. These two rovers, among other things, unraveled evidence of long-gone Martian lakes, and relatively recent interaction between surface material and water. This opened up the possibilities of habitable conditions once having existed on the planet, and laid the groundwork for the future
science to be conducted by the next Mars-rover, Curiosity. Curiosity landed in 2012, in what was not only a milestone for the largest rover yet, but also a technological breakthrough in terms of landing technique. Due to its mass, Curiosity could not simply land by using parachutes and airbags like its predecessors. Instead a complex rocket-powered device called the Sky Crane, would hover several meters above the ground and lower the rover down onto the surface before flying off to a crash landing in the distance.

The new rover, which is still fully functional, has made further discoveries that the ancient chemistry of Mars could have provided conditions suitable to life, that a water stream about knee deep once was present where the rover landed, just to name a few. This accumulated knowledge formed the baseline for the successor to Curiosity; Mars 2020.

### 1.1.5 Mars 2020

The 2012 Mars Science Laboratory-mission, with the rover Curiosity, proved that conditions on Mars once were habitable enough that life could have existed on the planet [8].

On December 4th 2012 NASA announced the Mars 2020-mission [8]. It is part of NASA’s Mars Exploration Program and will be the fifth American rover to land on Mars. To keep costs to a minimum the rover design will be similar to Curiosity, the previous rover the space agency sent to the red planet, but with a new set of scientific instruments in addition to engineering changes based on experiences with previous rovers.

Following the discoveries by Curiosity, the science goals for Mars 2020 are focused on searching for direct signs of extant life on Mars, exploring the geological features of the area surrounding the landing site and preparing soil samples for a future sample-return mission.

Seven instruments were chosen to be part of the Mars2020 rover. The selected instruments were announced in July 2014 [9], and in figure 1.4 an illustration shows an outline of the rover and how the instruments will be placed on the vehicle.

### 1.1.6 RIMFAX

RIMFAX (Radar Imager for Mars’ Subsurface Experiment) is a ground penetrating radar in development for the Mars2020-rover. It is being developed at FFI, building on experience gathered through previous ground penetrating radar projects, some of which have been tested on Svalbard. [10]

During operations, the radar will be able to penetrate down to over 10 meters depth, depending on subsurface material composition. This is the first time an instrument of this kind has been made for a NASA-rover, and for
Figure 1.4: Schematic of the Mars 2020-rover and the placement of the seven science instruments. The antenna of RIMFAX is installed in the rear of the rover, with the electronics adjacent to the Radioactive Thermal Generator (RTG), a plutonium based power generator.
the first time the upper layers of the Martian geology can be investigated.

The objectives of RIMFAX are to image subsurface structures and to provide information regarding the composition of the upper layers of Mars’ surface. The instrument will image down to a depth of more than 10 meters, with vertical resolution of better than 30 cm and a horizontal sampling distance of 10 cm as the rover travels along the surface. Moreover, data gathered from the instrument will be used to assess one of the main science goals of Mars2020, regarding whether habitable conditions were once present at the landing site of the rover. It also holds the distinction of being the first instrument that will be active while the rover is in motion. All other instruments are turned on only when the rover is stationary at a new destination.

RIMFAX utilizes a type of radar known as Frequency Modulated Continuous Wave radar, which is explained in more detail later in this thesis. The starting point of the radar signal is the DDS which creates the basic signal which is multiplied after digital-to-analog conversion to the wanted output frequency, which spans the range from 150 MHz to 1.2 GHz.

1.2 Motivation

To generate the sine wave output of the radar, a number of steps are performed. The first and one of the most important components of the synthesizer is the DDS. A DDS can be implemented using a dedicated integrated circuit, as these are readily available from a multitude of vendors. Several systems of RIMFAX will implemented on a FPGA, therefore it has also been decided to implement the DDS on the same FPGA to reduce the number of dedicated components, as this saves board area and power consumption.

Going to Mars sets restrictions when it comes to power consumption, size and radiation tolerance. This limits the choice of what FPGA is to be used and sets a limit to available logic due to extra redundancy that has to be added for the FPGA to function in space. RIMFAX will use a RTAX2000SL FPGA [11]. Different algorithms exist to reduce the footprint of the DDS, by e.g. utilizing symmetries in the sine wave, numeric approximations and trigonometric identities. If such a technique could be implemented while still retaining output characteristics that are sufficient for RIMFAX’s scientific objectives this could free up considerable resources on the FPGA, meaning more available logic for other functions. In particular digital filtering of the received signal could be improved by dedicating less space to the DDS and more to the RX filters. A detailed explanation of the DDS concept and its’ building blocks is provided in chapter 2.
1.3 Goals of this thesis

Goals of this thesis are to implement a sine wave DDS in FPGA and to explore concepts that possible can reduce the footprint and resource consumption of the DDS on the FPGA.

Methods for doing this have to be chosen from available literature and already published papers. Alterations have been done in terms of varying bit lengths of different signals in order to explore how this might impact the output signal characteristics.

The methods will be implemented on a Kintex-7 KC705 FPGA evaluation board and the output signal will be characterized. This includes analyzing the frequency spectra and doing phase noise analyses, and also looking at spurious noise and interactions from harmonic signals. Implementations using compression algorithms will be compared to a LogiCORE IP DDS provided in the Xilinx Vivado design tool and a DDS using a simpler compression method utilizing quarter sine wave symmetry, that is the current method of choice on the RIMFAX prototype.

Several test scenarios have been designed to provide a picture of the performance over a range of both DDS parameters as well as ranges in frequency between 18.75-37.5 MHz, as this corresponds to the frequency range of the RIMFAX DDS.

RIMFAX phase noise requirements state that the radar must have integrated phase noise performance of -80dBc or better (Based on personal communication with S-E.Hamran, Principal Investigator on RIMFAX). This is in order to ensure radar imaging of adequate quality at the minimum depth requirement of 10 meters underground. The dynamics per sample at the receiver is at -80 dBc, and the bandwidth of the received sample is 30 kHz. Therefore the phase noise measurements are focused on the range up to 30 kHz.

1.4 Outline

The work in this thesis has been split up in seven chapters. An outline of each chapter, giving an overview over the thesis, is presented below.

- Chapter 1: Introduction
- Chapter 2: An overview to basic radar principles, Frequency Modulated Continuous Wave radars and an introduction to Direct Digital Synthesizers is presented.
- Chapter 3: Various methods for compressing a sine wave Look-Up Table in a DDS is presented. The advantages and disadvantages of each method are discussed.
• Chapter 4: A thorough look at noise in Direct Digital Synthesizers, to give an overview and perspective for discussing the later test results.

• Chapter 5: This chapter covers the implementation of the DDS configurations on the FPGA. Furthermore, utilized hardware is presented, along with an overview of the VHDL-structure and associated Matlab-scripts.

• Chapter 6: This chapter explains the execution of the test scenarios and presents the results. A discussion section is also included.

• Chapter 7: Contains a summary of work done, conclusions and suggestions for future work.
Chapter 2

DDS and Radar Technology

This chapter covers basic theory about DDS, radar and the phase-locked loop which is a commonly used alternative to the DDS.

2.1 Direct Digital Synthesizers

Direct Digital Synthesizers were first described by Tierney et. al [12] in 1971. It comprises a phase accumulator and a phase-to-amplitude look-up table containing stored samples of the waveform we would like to create. A basic schematic for a sine output DDS is shown in figure 2.1. The frequency generated by the DDS is dependent on three parameters; the input Frequency Tuning Word (FTW), the system clock speed \( f_{\text{clk}} \) and the size of the phase accumulator, which is explained further below. This gives us the following expression for the output frequency, as shown in equation 2.1. In this equation \( N \) is the bit length of the FTW and the phase accumulator.

\[
 f_{\text{out}} = \frac{\text{FTW} \cdot f_{\text{clk}}}{2^N} \tag{2.1}
\]

2.1.1 Basic DDS structure

A DDS comprises a phase accumulator and a look-up table containing sampled sine wave amplitude values. The previously mentioned FTW serves as input to the phase accumulator. The phase of a sine wave runs from \( 0 - 2\pi \) in a repeating, periodic fashion.

For us to understand how the phase accumulator works, it is useful to think of the phase as a unit circle. The accumulator consists of a N-bit full-adder. For each clock cycle, the FTW is added to the phase sum of the previous cycle. Relating this to our phase unit circle, we can insert \( 2^N \) evenly spaced dots around the circle, giving the phase increment size and showing the total number of possible phase values. For each clock cycle, the FTW is added to the accumulated phase.
The basic blocks of a DDS include a Phase Accumulator, consisting of a register and an adder summing the accumulated phase with the FTW every clock cycle, and a Phase-to-Sine Look-up table which stores sine amplitude samples. The output from the phase register functions as an address to the corresponding look-up table sine sample. The sine sample is then sent to a Digital-to-analog converter.

When the accumulator overflows the cycle starts again, making the accumulator function as a modulo $2^N$ counter. As illustrated in figures 2.2 and 2.3, we see that by increasing the size of the accumulator, the frequency resolution increases. Frequency resolution can be calculated by setting the $FTW = 1$, giving the following expression for frequency resolution:

$$\Delta f = \frac{f_{\text{clk}}}{2^N}$$

From the phase accumulator we don’t initially get a complete sine value, only a linearly increasing sawtooth shaped signal. The conversion to a sine wave is done by connecting the output of the accumulator to a corresponding sine wave look-up table containing stored samples of a sine wave. Each phase value corresponds to an amplitude value in the LUT. Amplitude resolution is dependent on the number of bits in the stored sine values. A larger bit size will increase the number of possible amplitude steps vertically, allowing for finer resolution and lessen the impact of quantization error and phase errors, as explained further in Chapter 4.

The output from the look-up table is also the digital output of the DDS. Following the Look-Up Table, the signal is passed to a Digital-to-Analog Converter (DAC) which converts the signal into analog form. The signal also needs to be filtered after this stage prior to use by the specific application the DDS is supporting.

There are several advantageous characteristics in a DDS that makes it a desired component in various systems. Often used in digital communications systems, it allows for frequency adjustments in the milli-Hz range, low phase noise and very quick and agile frequency switching characteristics.
2.1.2 Sine wave look-up table

An analog sine wave is continuous with no discrete points. In the digital domain we don't have this luxury and therefore have to make do with a sampled version of the sine wave. This means discrete points with an amplitude corresponding to a given phase value.

Sampling signals come with their own challenges, related to sampling frequency, quantization errors due to the limited amounts of discrete amplitude levels for the samples to name a few. Theoretically we can produce frequencies up to half of the system clock. This follows from the Nyquist–Shannon sampling theorem [13]. This theorem explains that in order to fully replicate a continuous signal, the samples have to be a maximum of \( \frac{1}{2f} \) seconds apart, where \( f \) is the highest frequency component of the signal to be sampled. Statet another way, the
Figure 2.3: Phase circle showing the possible phase values of a 5-bit phase accumulator. Compared to figure 2.2, we now see that by increasing the accumulator size by a single bit, the minimum phase increment is reduced by half, also doubling frequency resolution (see equation 2.2)
sampling frequency has to be a minimum of twice the highest frequency component of the signal that is to be sampled, in order to recreate the original signal. This frequency is known as the sampling frequency $f_s$. Sampling with a lower frequency leads to aliasing, which means higher frequency components are aliased down and appear in the sampled signal as lower frequency noise.

2.2 Alternatives to DDS

2.2.1 Phase Locked Loop

One of the most prevalent alternatives for frequency synthesis apart from a DDS is the phase locked loop, or PLL. The concept was initially described in the 1930s. In essence, a PLL is a feedback system, and the principal building blocks are a phase detector, a loop filter, a voltage controlled oscillator (VCO) and a divide-by-N counter. A block schematic is shown in figure 2.4.

In a PLL the idea is to start off with a stable input frequency and from that generate a tunable output frequency [14]. The feedback loop allows the PLL to have both an accurate output frequency with relatively low noise.

We’ll use the block schematic in figure 2.4 to explain the concept. For a more detailed description, see chapter 1 in [14]. From the left we first have a stable input frequency $f_{osc}$, e.g. based on a crystal oscillator. This signal is passed through a divider to generate the phase detector frequency, $f_{PD}$. Next we have the phase detector and charge pump. Essentially this module compares the phase of the phase detector frequency with the phase of the signal $f_N$ from the negative feedback loop. Depending on the phase error, this triggers current correction pulses $K_{PD}$, which pass to a loop-filter (a low pass filter), which has a current-to-voltage transfer function which [14] denotes as $Z(s)$. The output of the filter is used as a control signal for the
voltage controlled oscillator (VCO). The final output of the PLL, stemming from the VCO, is determined based on the interaction of the loop filter output with the VCO.

The output of the VCO is connected to the phase detector through a feedback loop, and on the way passing through the $1/N$-divider. The VCO frequency is adjusted until there is no phase error detected at the phase detector. At this point the phase detector inputs both have the same frequency and phase. To adjust the frequency of a PLL, we can adjust the value $N$. The output of the VCO is given by equation 2.3.

$$f_{VCO} = \frac{N}{R} \cdot f_{osc}$$

(2.3)

The output frequency therefore can be adjusted in integer increments of $N$. This creates a spacing between each frequency, or channel.

The fact that the PLL relies on a feedback loop to control the output frequency means that when the desired frequency changes, a time delay will occur before the system settles on the correct frequency. This implies that rapid changes in frequency are more difficult to achieve than with a DDS, which can change output frequency in one single clock cycle of the system clock. Another advantage of the DDS is the higher frequency resolution. In a PLL the output resolution will always be limited by the ratio $N/R$ as $R$ is locked and $N$ is adjustable, while in a DDS the frequency step is limited only by the size of the phase accumulator, which generally is very large (often 32 or 48 bits).

### 2.3 FPGA

FPGAs (Field-Programmable Gate Arrays) are programmable circuits that contain programmable logic blocks, interconnects and memory elements. Many FPGAs are also reprogrammable and can be reconfigured simply by downloading a new bitstream file to the chip. This allows for easy studies and testing of concepts rather than having to develop an application-specific integrated circuit (ASIC) from the start. Not all FPGAs are reprogrammable. For instance is the RTAX2000SL FPGA used on RIMFAX a so-called anti-fuse FPGA [11]. This means that when the bitstream file is downloaded, you cannot reprogram the chip as fuses on the chip are burned to lock the configuration in place. Reprogrammable FPGAs have memory made out of either SRAM-cells or Flash memory. They differ in that SRAM-cells need to be reconfigured after power-down, while Flash memory retains the stored configuration.
2.3.1 FPGAs in space

Anti-fuse solutions are preferred for space applications where an SRAM-based FPGA, which can be re-programmed, thereby implicitly also is open for “reprogramming” due to single-event upsets caused by the present radiation environment.

FPGAs are very power consuming and this is one of the main drawbacks compared to an ASIC. However, as in the case of RIMFAX, an FPGA will already be utilized for many of the on-board functions of the instrument. Being able to include the DDS in the FPGA therefore causes less increase in power consumption compared to adding an extra integrated circuit.

Configuring a DDS in an FPGA is a task the FPGA is well suited for. As it contains logic blocks and memory, the only task needed is to tell the FPGA what data should be stored in the memory (in this case the sine wave samples) and how the logic should be connected to form a DDS.

The configuration of the FPGA is written in a hardware description language, in this case VHDL (VHSIC Hardware Description Language).

2.3.2 VHDL

VHDL is a hardware description language that originated from the U.S. Department of Defence, with development starting in 1981. The goal was to standardize and simplify descriptions of hardware circuits for when they needed replacement[15]. The first standard was published in 1987 and the rights to VHDL were given away fully to IEEE. In 1993, a revised version was published and is the version mostly supported to this day. In VHDL, digital designs can be described in a systematic way and by utilizing appropriate software, we can synthesize a netlist and moreover a ready configuration bit stream file which can be downloaded onto an FPGA to test the described design.

Only a short introduction will be provided in this section. There are numerous resources available for further study. One recommendation in particular is Mark Zwolinski’s book “Digital System Design with VHDL” [16].

A VHDL-file starts with an entity description. Basically this means describing the design as a black box, where the inputs and outputs of the design are declared.

The next part is the actual architecture of the design. In this section all the inner workings of the design are explained. A declaration area is dedicated for declaring internal signals. In VHDL we are describing hardware, which is generated based on the VHDL description[16]. As this is not a software code being executed, instead a description of multiple signals and their behaviour, the designer has to bear in mind that behaviour of all lines happen
concurrently. This means that all lines of code after the declaration section runs in parallel, not sequentially as in regular software code. This part of the VHDL code is known as the concurrent part of the code. In itself not dependent on being synchronised to a system clock. Any change in the input results in an immediate change on the output. However, synchronous behaviour can be achieved by using processes, which can be told to trigger on an event by designated signals, e.g. a clock signal. Within a process each line of code is executed sequentially. Processes are useful for implementing state machines, which may execute a different behaviour depending on specific signals or conditions, which may trigger a specific state. This thesis did not implement the DDS explicitly as a state machine, and I will therefore not go into further detail.

2.4 DDS in Hardware

Here follows a short description of how DDS can be implemented in FPGA as well as in a designated integrated circuit.

2.4.1 CMOS DDS

If you need a DDS for an application, they can easily be purchased as dedicated integrated circuits from a number of vendors, such as Analog Devices. This has the advantage of giving the user a thoroughly tested and documented device which is good for predictability.

The drawback of such a solution is that it gives the user an extra integrated circuit to handle. This means allocating enough area on the circuit board, routing signals, power and ground plane to the chip, consuming even more area, and finally there is the concern of power consumption.

Especially if your design already will include an FPGA and power is a very limited resource, adding an extra integrated circuit into the design may be putting too much strain on the power budget of the design. Combined with the extra area needed, this is a serious drawback for applications such as RIMFAX which has very tight requirements and constraints both in terms of power consumption and physical area consumption of the circuit board. While a dedicated DDS in itself might outperform a FPGA based design, the fact that FFI anyway are using a FPGA for other on board tasks, meant that adding a dedicated DDS circuit also included added board area to make room for the extra IC and routing. It was therefore desirable to look at solutions where the DDS could be included on the FPGA along with the rest of the digital electronics.

2.4.2 DDS in FPGA

Implementing a DDS on an FPGA is a quite common application and can be readily found available as an IP for quick implementation. Several pa-
pers have also been written on the subject [17][18].

FPGAs are suitable for prototyping and applications where small number of units are being made. For large scale products, the usual approach is to develop an ASIC which contains the application specific circuit instead of using a FPGA. FPGAs are more power consuming than an ASIC and are also often more expensive due to including peripherals and additional resources beyond the specific need of the application.

### 2.5 Radar

In the scope of this thesis, radar is simply the application for which the DDS is intended, and is not directly related to the work in the thesis. Therefore only a short overview of radar technology will be given.

During the first decades of the 20th century, much research was done in the field of using radio waves to detect and measure the distance of foreign objects. The term RADAR was coined by the U.S. Navy, and stood for RAdio Detection And Ranging, a term which in time has entered the dictionary as a noun in its own right, without capital letters.[19]

In short the radar functions by transmitting radio waves which then are reflected when the waves interact with other objects, due to differing dielectric constants in the objects compared to the medium in which the wave is travelling. The reflected waves are then detected by the radar system, and by processing this signal, a variety of information can be extracted, such as speed relative to the radar or distance, depending on the desired application in question.

The radar equation (equation 2.4) calculates the power of the received signal [20]. Assuming that both transmitter and receiver are in the same location relative to the measured object, the equation shows that the power of the received signal declines rapidly as the distance increases.

\[
P_r = \frac{P_t G_t A_r \sigma F^4}{(4\pi)^2 R_t^2 R_r^2}
\]  

(2.4)

\(P_t\) is the transmitter power, \(G_t\) is the antenna gain (meaning the antennas ability to transform electrical energy into radio waves), \(A_r\) is the area of the receiver antenna, \(\sigma\) is the radar cross section, which means the measured object’s ability to reflect radio waves, \(R_t\) is the distance from the target to the transmitter, while \(R_r\) is the distance from the target to the receiver antenna.

A radar signal is usually transmitted as pulse. The transmitted energy in a transmission is expressed in equation 2.5.
\[ E_t = \tau \cdot P_t \] (2.5)

Transmitted energy \( E_t \) is given by the product of the pulse length \( \tau \) and the transmitted signal power \( P_t \). Increasing transmitted energy, increases the range of the radar. This entails either lengthening the pulses or increasing the power of the antenna. Neither of these may be feasible in all scenarios, something that is the case for RIMFAX. There is, however, another way of yielding better results, which will be described later.

### 2.5.1 Ground Penetrating Radar

Ground penetrating radars is a subtype of radar commonly used for imaging subsurface structures. It is a ultra wide band radar, meaning that the bandwidth of the radar is equal or close to the center frequency of the radar. When sending a radar pulse into the ground, you have major losses in received signal quality due to the conductivity of the ground and scattering of the transmitted signal. Signal to noise ratio of a ground penetrating radar is given by the equation:

\[ \text{SNR} = \frac{2E_t}{N} \] (2.6)

To increase the SNR, the transmitted energy has to increase. One way of doing that is increasing the power, as we know from equation 2.5. However, additional power may not be readily available and therefore this is often not a viable option, especially for space applications where power budgets are already stretched. The second option is to increase the pulse length \( \tau \). But this comes at the expense of resolution. We want the best resolution possible, which means that \( \tau \) has to be small.

With the restrictions of not being able to increase signal power and having to keep a short pulse length, alternatives had to be found. The solution was to abandon a single pulse radar, transmitting at a set frequency, and instead create a sweeping radar system.

### 2.5.2 Frequency Modulated Continuous Wave Radar

One of these solutions is the Frequency Modulated Continuous Wave radar. Consider a regular radar pulse. It consists of a single pulse at a set frequency, transmitted at a given power. In a FMCW-radar, the pulse consists of a frequency sweep, sweeping from a start frequency \( f_1 \) up to a stop frequency \( f_2 \). The frequency increases linearly. With this approach, the resolution of the radar is no longer linked to the pulse length \( \tau \). In a FMCW-radar each pulse is in the millisecond range, far longer than the typical nanosecond range of a regular single pulse radar. The resolution is now given by the bandwidth of the sweep, defined by the frequency difference \( \Delta f \) between \( f_1 \) and \( f_2 \).
The FMCW consists of a signal generator capable of varying the frequency. A pulse is transmitted, and when hitting an object it is reflected back to the antenna. The receiving antenna picks up the reflected signal. We now mix the received signal with the transmitted signal, by way of multiplying the two. This is illustrated in figure 2.6.

Depending on the time it has taken for the signal to bounce back, the transmitted frequency has changed leading to a frequency difference between the two signals, as shown in figure 2.5. When multiplying the signals we are left with two signals; the difference between the two signals, and the sum [21]. This is due to the heterodyne principle, which is most easily described by the trigonometric identity of multiplying two sine waves, which is described further in chapter 4 and equation 4.23. The latter is effectively attenuated by low pass filtering, leaving a single sine wave with a frequency equal to the difference between the transmitted and received signal. The difference signal will be constant throughout the sweep. A larger frequency difference equals a longer distance to the object that reflected the signal. The difference signal that results from mixing the transmitted and received signal is called the Intermediate Frequency signal, or IF signal.

Further processing of the signal is done by doing an inverse Fourier transformation. With a single frequency this will result in a sinc-shaped plot. The peak of the plot will be at the x-value corresponding to the time $T_0$ which is the time from transmission to the signal bounced back.

In RIMFAX, the signal generator of the FMCW-radar is comprised of the DDS and analog multipliers and switches. There are several reasons for applying this approach, most importantly, by having multipliers in a sequential fashion, each covering one decade of frequency increase, you mitigate the second harmonic of the signal, as that frequency will, even from the beginning, lie outside the bandwidth of the system. This greatly increases signal performance and reduces the need for further filtering of the transmitted radar pulse before transmission.
Figure 2.5: Graph illustrating the concept of a FMCW radar. The transmitted sweep is shown in blue, with the received, reflected sweep shown in red. The dashed lines over and under the solid lines indicate the frequencies $f_{\text{start}}$ and $f_{\text{stop}}$ respectively. The receiver mixes these signals, which creates two new signals due to the heterodyne principle[21]: One signal with frequency equal to the sum of the two signals, and one signal with frequency equal to the difference. The former is filtered, using a low-pass filter, while the difference signal is processed to produce the radar image. This signal is referenced in [22] as the Intermediate Frequency (IF)-signal.
Figure 2.6: Illustration of the main components in a FMCW radar. A variable frequency signal generator transmits a signal, which is reflected and received by the Rx-antenna. The received signal is a time delayed version of the transmitted signal, which also leads to a frequency difference. These signals are mixed, creating a signal with frequency equal to the sum and one signal with frequency equal to the difference. The sum signal is filtered using a low pass filter.
Chapter 3

Methods of Resource Compression in FPGA-based DDS

3.1 Overview

One of the main goals of this thesis is to explore techniques that may allow for reduced resource consumption on the FPGA, with either equal or improved output signal characteristics. Various methods exist in order to achieve this. In this chapter we will examine a selection of these methods.

As explained earlier, the two main blocks of the DDS are the phase accumulator and the sine wave LUT. Intuitively, we see that if we want to reduce the footprint of the DDS, we either have to make the phase accumulator smaller, or reduce the size of the look-up table.

3.2 Reducing phase accumulator size

One option for reducing the DDS footprint is to make the phase accumulator smaller. We know from equation 2.2 that the length of the FTW and by extension the phase accumulator, controls the frequency resolution of the DDS. By increasing the phase accumulator size, the number of possible phase values increases, which means greater frequency resolution. RIMFAX is supposed to be a FMCW-radar, as we know. Due to the sweeping profile of the radar, a minimum of 29 bits is needed to achieve small enough frequency steps.

Nevertheless, in the interest of characterizing changes in the output when adjusting the size of the phase accumulator and FTW, we will actually test an even smaller accumulator size.
3.2.1 Sine wave Look-Up Table

By far the most area consuming part of the DDS, is the sine wave look-up table. If we consider a basic DDS, the phase accumulator output is directly connected to the sine wave LUT. Each phase value has a corresponding sine amplitude value. Now consider a 20 bit phase accumulator, which is still rather small compared to commercially available DDS-circuits [ad9910], which often have as much as 48 bit accumulators. The bit length of the sine wave samples determine the amplitude resolution. Longer bit lengths lead to higher resolution. In our example, say we have an amplitude bit length of 12 bits.

If we simply use the phase values directly, this requires a look-up table of the following size:

\[ 2^{20} \cdot 12 = 12582912 \text{ bits} \approx 12.58 \text{ Mb} \]  

(3.1)

Even at this modest accumulator length the look-up table is very large and requires a lot of memory on the FPGA. In practice though, the phase bits are not used as is. To reduce the required size of the look-up table, we can truncate one or more of the least significant bits of the phase output. For each bit truncated the number of look-up table entries is reduced by half. Let's continue using the same example as before, but now truncating the phase bits down to 12 bits, same as the amplitude resolution.

\[ 2^{12} \cdot 12 = 49152 \text{ bits} = 49.15 \text{ kb} \]  

(3.2)

However, there are trade-offs by increasing or decreasing bit lengths of the accumulator and amplitude samples. This affects the noise characteristics of the output and is explained in more detail in chapter 4.

In addition to adjusting these parameters there are several other methods that reduce the necessary memory for storing the sine wave samples. A subset of these methods are presented below.

3.3 Compression methods and algorithms

In this section various methods for compressing the sine ROM are explained.

3.3.1 Quarter Sine Symmetry

If we look at a sine wave and split it in half, we see that the positive and negative half periods are symmetric. Additionally it can also be seen that each half, when divided by two again, is also symmetric. In short, a sine wave is symmetric for each quarter. This is also apparent from the unit circle when plotting the phase of a sine wave.
CHAPTER 3. METHODS OF RESOURCE COMPRESSION IN FPGA-BASED DDS

Figure 3.1: As shown in this figure, each quarter of the sine wave is symmetrical. We can therefore reduce the look-up table size by a factor of four, by simply storing the first quarter, and generating the other three by manipulating the address - and output signals.

Table 3.1: Table demonstrating the function of the two most significant bits of the phase output in a DDS utilizing quarter sine wave symmetry compression.

<table>
<thead>
<tr>
<th>MSB</th>
<th>MSB-1</th>
<th>Sine Quadrant</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>First quadrant</td>
<td>No inversion of signals</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Second quadrant</td>
<td>Invert LUT address bits</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Third quadrant</td>
<td>Invert amplitude output bits</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Fourth quadrant</td>
<td>Invert LUT address bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>and amplitude output bits</td>
</tr>
</tbody>
</table>

What this means is that the size of the sine wave look-up table can be reduced to only a quarter of the size when storing the entire sine wave, as it is only necessary to store the first quadrant of the wave form [23][24]. This comes at the expense of some logic, to keep track of which quadrant we are in, adding the phase increment for each clock cycle when in the first and third quadrant, subtracting the increment while in the second and fourth quadrant, and inverting the output when in the third and fourth quadrant. Still, the reduction in LUT size is considerable enough that even with the additional logic, a substantial amount of resources is saved, the exact numbers depending on the phase length and amplitude length.

The easiest way of implementing this is by using the two MSBs of the phase accumulator output as control bits. This means that bit MSB-2 down to 0 is passed to the LUT as address bits, while MSB and MSB-1 is used to control inverting of phase and/or amplitude output. See table 3.1 for more details.

3.3.2 Sunderland compression

First described by Sunderland et.al. in 1984 [23], the Sunderland compression algorithm utilizes trigonometric identities to approximate a sine wave.
CHAPTER 3. METHODS OF RESOURCE COMPRESSION IN FPGA-BASED DDS

Figure 3.2: This figure displays a block schematic of how a DDS utilizing either Sunderland’s compression method [23] or Nicholas’ compression method [25], the latter of which is largely similar to the first, but differs in how the sine samples are selected.

By using two smaller look-up tables and combining the output of the two LUTs the total memory usage can be reduced considerably compared to implementing a single LUT containing all sine wave samples.

Sunderland splits the phase into smaller components and by combining these parts using the aforementioned trigonometric identities, is able to recreate an approximation of a complete sine wave. In [23], Sunderland explains how this approach usually would entail using multipliers, but by using suitable approximations this can be avoided. Multipliers are hardware intensive and therefore it is highly advantageous not being dependent on them.

Considering a quarter sine wave look-up table, with X number of phase bits and a word length in the look-up table of Y bits. This gives a look-up table size of $Y \cdot 2^X$ bits. By using trigonometric identities and approximations, we can greatly reduce this number.

The Sunderland algorithm is explained using an example with a 14 bit phase output from the phase accumulator. The output word length is 12 bits. The two MSB bits are used later in the circuit to indicate which quadrant we are in and if the sine wave is increasing or decreasing in amplitude, as previously described in table 3.1. This leaves us with 12 remaining phase bits, which are further divided into three fractions $A, B, C$, each consisting of four bits such that $A < (\pi/2), B < (\pi/2)2^{-4}$ and $C < (\pi/2)2^{-8}$ [23]. The sine wave can now be approximated using trigonometric identities, and we get the following function as a result:

$$\sin(A + B + C) = \sin(A + B)\cos C + \cos A \cos B \sin C - \sin A \sin B \sin C$$  \hspace{1cm} (3.3)

The paper further approximates this, due to the relative size of the three fractions, to:

$$\sin(A + B + C) = \sin(A + B) + \cos A \sin C$$  \hspace{1cm} (3.4)
Using equation 3.4, we now create two smaller look-up tables, one for each of the terms in the approximation. The first term gives us a coarse output value, while the second term provides fine values for interpolating between each coarse LUT-value.

The coarse LUT has an 8 bit input address, given by concatenating A and B. The output is 11 bits. The fine LUT has an 8 bit input address from concatenating A and C, with a 4 bit output as the upper 7 bits will always be 0. The output values from each look-up table is added to produce the final output amplitude.

With a state machine consisting of one state for each quadrant of the sine wave, we then take the 1’s complement of the value when sine is in the negative half. To accomplish the second and fourth quadrant, where the amplitude is decreasing towards 0, the 1’s complement of the lowermost bits of the phase accumulator output is used for the input addresses of the look-up tables.

3.3.3 Nicholas’ compression algorithm

In 1988, Nicholas et.al. described a modified version of the Sunderland compression [25].

This method still relies on the usage of two look-up tables, and the quarter sine symmetry as described in section 3.3.1.

The two look-up tables are divided in a course resolution look-up table and a fine resolution look-up table, just as we are familiar with from the Sunderland compression method. However rather than utilizing Sunderland’s method of trigonometric approximations, Nicholas describes a course ROM with low phase resolution, where the fine ROM contains values used for interpolation between the coarse ROM samples.

These samples have been numerically calculated beforehand, using one of two optimization criteria that will be explained later. The end result is improved spurious performance, due to optimizing the choice of fine ROM values.

As in Sunderland’s method, we have the phase $\Phi$ which is split in three parts, shown in 3.5

$$\Phi = \alpha + \beta + \chi$$  \hspace{1cm} (3.5)

The bit lengths of $\alpha$, $\beta$ and $\chi$ are A, B and C, respectively. For inputs to the two look-up tables we have $\alpha$ and $\beta$ as input value to the coarse ROM, while the fine ROM input value consists of $\alpha$ and $\chi$. To select the samples, first we split the range from 0 to $\frac{\pi}{2}$ into $2^A$ separate regions. Within each region there are $2^B$ coarse ROM values distributed evenly. This concept is
CHAPTER 3. METHODS OF RESOURCE COMPRESSION IN FPGA-BASED DDS

Figure 3.3: Figure demonstrating the concept of Nicholas’ compression algorithm. Here shown with 6 bit phase and segmentation of 2 bits each for A, B and C. (©1988 IEEE. Permitted for thesis reuse.) shown in figure 3.3.

In the figure the dots along the sine function represent the coarse ROM sample within each quadrant, corresponding to the values of $\alpha$ and $\beta$ at that point. The error bar represents the area where the correcting values supplied by the fine resolution ROM is added.

For each quadrant, one correction values is used between the error bars and the corresponding coarse ROM sample. This value is used for all $\chi$ offsets from the coarse sample. In the case of figure 3.3, this means four offset values for each coarse ROM sample. The value of the offset, which is stored in the fine resolution ROM, has been pre-calculated from one of two optimization criteria.

First the coarse ROM values are calculated. This is done by the following algorithm, as presented by Nicholas et.al:

$$F_c(\alpha, \beta) = \sin\left(\pi \left(\frac{\alpha^B + \beta}{2^{A+B}} + \frac{1}{2^{A+B+C}}\right)\right)$$  \hspace{1cm} (3.6)

The first possible optimization criteria is based on reducing the mean square error of the samples. The mean square error can be described as a measure of how close a fitted line is to the actual data points. The algorithm for calculating fine ROM samples using this criteria is shown in 3.7

$$F_f(\alpha, \chi) = \sum_{n=0}^{N-1} \frac{1}{2^B} \left[\sin\left(\pi \left(\frac{\alpha^B + \beta^C + \chi}{2^{A+B+C}} + \frac{1}{2^{A+B+C+1}}\right)\right) - F_c(\alpha, \beta)\right]$$  \hspace{1cm} (3.7)
The second option is a criteria based on minimizing the absolute error of the correction values. This means calculating the maximum and minimum possible value of a fine ROM-sample for a given coarse ROM value. The difference between these two outer borders gives the least absolute error for each value of \( \beta \) and is the value that is stored. The algorithm for choosing fine ROM values when optimizing the absolute error is given as 3.8

\[
F_f(\alpha, \chi) = \frac{1}{2} \text{MAX}
\]

Nicholas states that in simulations, the latter option appears to offer a lower power of the maximum noise spur, but this has not been proven. The mean square error optimization does however reduce the total spur energy, according to [25].

This means that choosing an optimization criteria can achieve either of the following:

- Reducing the total noise energy, meaning a reduced noise floor.
- Reducing the power of the maximum spur, meaning an improved SFDR.

Which of these criteria that ultimately are chosen will depend on the application and whether a lower noise floor or a better SFDR is most critical.

### 3.3.4 Phase Difference Algorithm

Nicholas et.al [25] also suggested implementing another function in the look-up tables. Instead of storing the explicit sine value, we can instead store the difference in phase from one sample to the next, and utilize the truncated phase bits in calculating the final output.

We can store the following function in the look-up table, instead of the sine function:

\[
f(x) = \sin(\pi x/2) - \pi x/2
\]

Since the final full scale output requires adding the truncated phase bits, the maximum amplitude stored in the look-up table is less than when implementing the sine function. Because of this, the size of the look-up table shrinks accordingly. This effectively will reduce the necessary look-up table size by an extra 15% according to Nicholas’ estimation.

In terms of added logic, this only leads to the addition of an extra adder. A block schematic of this approach is shown in figure 3.4.
Figure 3.4: Block schematic of the phase difference algorithm proposed by Nicholas in [25]. The truncated phase bits are added to the final LUT-output, to produce the output sine sample. Compared to Sunderland - and Nicholas' compression this requires only a single extra adder in terms of extra resources.

3.3.5 Kenji Terai Algorithm

Finally, a method proposed by Kenji Terai [17] was explored. Just like the phase difference algorithm and Nicholas' compression, it carries much of its basic architecture from the Sunderland compression method. While all the aforementioned methods involve approximating a sine wave, thereby always containing a quantifiable approximation error, the new method involves no such thing. In theory this means Kenji Terai’s method is able to describe a perfect sine wave.

The article proposes a configuration where the phase value, denoted as \( P \) is split into two parts \( X \) and \( Y \). This differs from the previously explained methods where the phase was divided in three, which were then combined into two separate addressing signals for the look-up table. Terai still uses two look-up tables. Phase value \( X \) addresses LUT1 and \( Y \) addresses LUT2.

Each look-up table contains the values for both the sine and cosine of \( X \) and \( Y \). Each phase address therefore is linked to a corresponding amplitude bit value which has to be split into two output signals, each containing half of the bits at the address, for a \( \sin(X) \) - and \( \cos(X) \) output from ROM1 and \( \sin(Y) \) - and \( \cos(Y) \) from ROM2.

The expression for calculating the output sine amplitude is given by equation 3.10.

\[
\sin(X + Y) = \sin(X)\cos(Y) + \cos(X)\sin(Y) \quad (3.10)
\]

Due to the phase being split in two components, the necessary resource requirement for the look-up tables are:

\[
2(2^X Wx + 2^Y Wy) \quad (3.11)
\]

Where \( Wx \) and \( Wy \) are the word lengths of the output amplitude, while
the multiplication by 2 is due to us having to store both the sine and cosine of the phase value. In the paper, Terai calculates the requirements for an example DDS with 12 bits phase amplitude. The new method requires a total of 3072 bits, while traditional Sunderland-compression results in 6144 bits. The reduction is due to splitting the phase into two 6 bit components, rather than three 4 bit components which are then concatenated into two 8 bit addressing signals for the look-up tables.

Terai’s method utilizes the quarter sine symmetry for reducing the size of ROM1. This ROM gets its phase value from the X most significant bits of the phase output. ROM2 can be reduced in size depending on the relation between X and Y. As stated in Terai’s paper [17], the phase value Y takes the phase values in the range $0 - \frac{\pi}{2}$. When X is longer than 3 bits, it is simpler to store the value $1 - \cos(Y)$ instead of $\cos(Y)$. Terai further describes how we then can reconstruct the value of $\cos(Y)$ by adding a subtractor, removing the stored number from 1.

As X grows, intuitively Y shrinks. Moreover, this gives us the chance of reducing the amplitude values stored in the second ROM. Terai explains how we can calculate this by introducing the factors $W_s$ and $W_c$, which are defined as:

$$W_s = -\ln(\max(\sin(Y)))$$

$$W_c = -\ln(\max(1 - \cos(Y)))$$

Figure 5 in [17] shows a graph detailing how the number of phase bits X determine how many bits can be reduced in the Y-phase LUT. If $W_c$ is larger than the number of output bits from the X-phase LUT, the term $1 - (\cos(Y))$ will always be 0 and can be dropped from the look-up table, reducing size further.

Table 2 in [17] shows how this method can reduce the DDS-footprint compared to a Xilinx DDS core IP.
Chapter 4

Sources of noise in DDS

In this chapter we will take a look at noise sources in a DDS system. The main focus is on phase noise and spurious noise. There are several factors that influence the noise characteristics in a DDS system and several design choices can make an impact to the final output noise. We will first describe phase noise before continuing with spurious noise and other noise components.

4.1 Phase noise

Phase noise is one of the most important characteristics of a DDS system. It is a type of noise caused by short, random anomalies from the theoretical phase of a given perfect sine value.

IEEE defines phase noise as “One-half of the phase spectrum \( S_\phi(f) \) (...) defined as [the] one-sided spectral density of the phase fluctuations”[26]:

\[
\mathcal{L} = \frac{S_\phi(f)}{2}
\]  

A perfect oscillator would output only a single frequency, without any anomaly. In the real world this is not attainable. Imperfections in the circuit, components and the surrounding environment will lead to fluctuations that cause the frequency of a signal to slightly vary in a random pattern. Due to the randomness of these fluctuations they do not cause spurious noise, which stems from periodic noise, appearing as a separate frequency component. Phase noise instead is spread around the carrier and trails off as we study frequencies further and further from the carrier.

The phase noise is typically given as a unit of dBc/Hz, which means it is measured in decibels relative to the carrier, also known as the desired frequency, for a bandwidth of 1 Hz. Phase noise is measured at a given frequency from the carrier frequency and this is also important to include when measuring phase noise.
Phase noise is often measured using spectral analyzers. They allow for quick and easy calculation of the phase noise using single sideband analyses of the signal. The result is a plot with frequency offset on the x-axis and magnitude of the phase noise on the y-axis.

In addition to presenting phase noise at a specific frequency offset, what often is done is to integrate the phase noise. This entails integrating the phase noise within a desired frequency range. From this measurement it is possible to calculate a signal-to-noise ratio, and to calculate the RMS phase error and by extension time jitter of the signal [14].

Integrated phase noise $A$ is calculated as described in equation 4.2, based on the description of integrated phase noise in [14]. In the equation, $A$ is the integrated phase noise, $\mathcal{L}(f)$ is measured phase noise at frequency offset $f$, and $f_a$ and $f_b$ are the lower and upper boundaries of the integral, respectively. Figure 4.1 illustrates the concept of integrated phase noise.

$$A = 2 \cdot \int_{f_a}^{f_b} \mathcal{L}(f) \cdot df \quad (4.2)$$

This can in turn be used to calculate the SNR of the signal. This is calculated by equation 4.3, and in dB as calculated in equation 4.4.

$$\text{SNR} = \frac{1}{A} \quad (4.3)$$

$$\text{SNR}_{\text{dB}} = 10 \cdot \log\left(\frac{1}{A}\right) \quad (4.4)$$
In FMCW radar applications we also have to consider the range to the target, when measuring phase noise [27] [22]. Phase noise in the received signal, denoted $L_{Rx}$, is simply a delayed version of the phase noise $L_{Tx}$ of the transmitted signal.

In [22], the authors introduce the variable $\tau_d$, which represents the delay time between the transmitted and received signal, which is given by equation 4.5. $R$ is the distance from the antenna to the target, while $c$ is the speed of light.

$$\tau_d = \frac{2R}{c} \quad (4.5)$$

In an FMCW-radar, the received signal is mixed with the signal to be transmitted, and the resulting signal, in [22] called the Intermediate Frequency signal (IF-signal), has phase noise given by equation 4.6.

$$L_{IF}(f_m) = L_{Tx}(f_m) \cdot 4\sin^2(\pi f_m \tau) \quad (4.6)$$

In equation 4.6, $L_{IF}(f_m)$ is the phase noise of the IF-signal at offset frequency $f_m$ from the carrier frequency, while $L_{Tx}(f_m)$ is the same for the transmitted signal. The term after the multiplication sign is the correction factor, to correct for correlating phase noise at short distances. By plotting phase noise plots using the above equation, we see that the correction does in fact not function at long distances, but improve the phase noise considerably for short distances, which is demonstrated in the results chapter later in this thesis. A clear definition of what constitutes short range is not given by the authors, but an example using a target distance of 600 meters is used, which indicates the correction factor is valid for the distances considered for RIMFAX.

### 4.1.1 RMS Phase error and Jitter

Jitter is noise caused by irregularities in the periodicity of a signal. Due to effects caused by circuit instability, clock irregularities etc a signal will never have a perfect periodicity, but will have small fluctuations which cause noise in the signal. [24]

A main source of jitter is thermal noise in the circuit. This noise is totally random and therefore has a Gaussian distribution.

Additionally, jitter noise will appear in the digital-to-analog conversion. Theoretically each sample to be converted is done at precise intervals. However, due to instabilities in the clock source, the timing between each clock pulse will be ever so slightly off, causing jitter noise.

Jitter in oscillators is caused by thermal noise, instabilities in the oscillator electronics, external interference through the power rails, ground, and even the output connections. Other influences include external magnetic or...
electric fields, such as RF interference from nearby transmitters, which can contribute jitter affecting the oscillator’s output. Even a simple amplifier, inverter, or buffer will contribute jitter to a signal.

Thus the output of a DDS device will add a certain amount of jitter. Since every clock will already have an intrinsic level of jitter, choosing an oscillator with low jitter is critical to begin with. Dividing down the frequency of a high-frequency clock is one way to reduce jitter. With frequency division, the same amount of jitter occurs within a longer period, reducing its percentage of system time.

In the previous section we looked at how integrated phase noise could be used to calculate the SNR of a signal. We can also use this information to calculate RMS phase error which in the time domain is the same as jitter. Phase error and jitter are in other words two sides of the same concept, phase error when in the frequency domain and jitter when in the time domain.

The reader is encouraged to read chapter 26 of [14] for the full elaboration on the relation between integrated phase noise and jitter. A condensed explanation is offered below.

Banerjee in chapter 26, page 239 of [14] explains that power is related to the square of the voltage. Phase noise is relative to the carrier power, and the same therefore applies to noise voltage versus carrier voltage. Since are discussing this relative to the carrier, we do not require knowing the resistance value when discussing the power, due to the same resistance existing in both terms, thereby cancelling each other. Banerjee continues to explain how integrated phase noise equals the variance of the noise voltage. Standard deviation of the noise voltage (the result is a relative voltage, relative to the carrier), is shown in 4.7.

\[ \sigma_v = \sqrt{2 \cdot \int_{f_a}^{f_b} L(f) \cdot df} = \sqrt{A} \] (4.7)

In order to get to phase error calculations, Banerjee continues to show that the standard deviation for noise voltage can be equated to standard deviation of the phase error \( \sigma_\theta \). The standard deviation in equation 4.7 is in number of radians when read as the phase error. As RMS phase error is usually displayed in degrees, we have to convert the number as shown in equation 4.8.

\[ \sigma_{\theta(\text{deg})} = \frac{180}{\pi} \cdot \sigma_{\theta(\text{rad})} = \frac{180}{\pi} \cdot \sqrt{A} \] (4.8)

Now, finally we can calculate the RMS jitter this corresponds to. RMS phase error and RMS jitter are, as we now know, similar, just viewed in different domains. The conversion is shown in equation 4.9

\[ \sigma_t = \frac{1}{f} \cdot \frac{\sigma_{\theta(\text{deg})}}{360} \] (4.9)
4.2 Spurious noise

A DDS will never reproduce an exact sine wave. This is inescapable and is due to the fact that to create a sine wave digitally, there is no getting away from basing the signal on a sampled waveform. The resolution can be increased, which also increases precision. But due to the stepwise fashion the sine wave is constructed from, there will always be irregularities. When an error in the signal appears periodically, it takes on the shape of a spur (spike) in the frequency spectrum. This is because a periodic error can be thought of as a signal with its own inherent frequency, and period.

4.2.1 Spurs due to phase bit truncation

The effects of phase accumulator truncation have a potentially large impact on the spurious noise characteristics. As explained in [24], an ideal DDS with no truncation of neither phase or amplitude has an output sequence described by:

\[ s(n) = \sin\left(2\pi \frac{\text{FTW}}{2^j} n\right) \]  \hspace{1cm} (4.10)

FTW is the Frequency Tuning Word and \( j \) is the bit length of the phase accumulator. However, to use the entire phase accumulator output will as explained earlier, lead to an impractically large sine wave LUT. Instead we truncate the phase accumulator output into such a size that is more practically feasible while still satisfying needs for resolution.

Taking into consideration the truncation of phase bits, the modified version of equation 4.10 becomes:

\[ s(n) = \sin\left(2\pi \frac{\text{FTW}}{2^k} \left[ \frac{\text{FTW}}{2^{j-k}} \right] \right) \]  \hspace{1cm} (4.11)

Where the square brackets indicate truncation to integer values, further rewritten this becomes (as explained in [24]):

\[ s(n) = \sin\left(\frac{2\pi}{2^j} (\text{FTW}n - e_p(n))\right) \]  \hspace{1cm} (4.12)

Where \( e_p(n) \) is the phase truncation error. As we utilize the \( k \) topmost bits for LUT addressing, this means the magnitude of the phase error sequence is limited to:

\[ e_p(n) < 2^{j-k} \]  \hspace{1cm} (4.13)

It should also be noted that phase truncation errors only when the \( \text{GCD}(\text{FTW}, 2^j) \) is smaller than \( 2^{j-k} \). A GCD that is equal to or greater than \( 2^{j-k} \) means that the FTW has no non-zero bits which get truncated. When only zeros are truncated, no error is introduced either. This is an important point in this particular DDS due to the frequency range that will be used,
and I will return to this point in chapter 5 about the implementation.

In [28], Jenq considers the phase accumulator output that addresses the sine LUT as a form of non-uniform sampling, due to the inherent jitter and inaccuracies in the operation of the system clock that leads to each sampling not being uniformly spaced in time. This concept is summarized in [24] by Vankka.

The truncation effects are periodic. Jenq considers the FTW as a number consisting of an integer part $W$ and a fractional part $L/M$. Here $W$ is the $k$ topmost bits which are used for LUT-addressing, while the fractional part is the $2^{j-k}$ remaining bits which are truncated.

$$FTW = W + \frac{L}{M} \quad (4.14)$$

In equation 4.14 the ratio of $M$ to $L$ is a prime number, and $M$ is the smallest integer that makes $M \cdot FTW = M(W + \frac{L}{M})$ an integer. As stated in [24], the overall period of the output sine wave is $MT_{clk}$ where $T_{clk} = \frac{1}{f_{clk}}$, and $M$ is calculated by the following equation:

$$M = \frac{2^{j-k}}{GCD(FTW, 2^{j-k})} \quad (4.15)$$

Furthermore, this number can be used to calculate the number of spurs caused by phase truncation, as demonstrated by Nicholas et.al. in [29]. The location of these spurs are all in the first Nyquist zone, meaning in the bandwidth of 0 to $f_s/2$, where $f_s$ is the sampling frequency. The number of spurs $Y$ can be calculated by:

$$Y = \frac{2^{j-k}}{GCD(FTW, 2^{j-k})} - 1 = M - 1 \quad (4.16)$$

This gives us insight into some interesting corner cases. We already know that when all truncated bits are zero, there is no phase error. In such a case $M = 1$. Cross-checking with equation 4.16, we see that this is correct. It also demonstrates the worst case and best case scenarios.

The worst case scenario is when $M = 2$. Again by looking at equation 4.16, such a case will give 1 single spur. This also means that all spurious noise energy generated by phase truncation will be focused in one single spur.

A best case scenario, when there are phase errors, is when $GCD(FTW, 2^{j-k}) = 1$. This gives us $2^{j-k} - 1$ spurs, which also means sharing the noise energy across all these spurs. In the frequency spectrum this will then appear almost as white noise, depending on the size of the FTW.

The placement of these spurs can be calculated. In particular, what is known as the Primary Phase Truncation spur, which is the first order spur
caused by phase truncation with the highest magnitude.

Chapter 5 of [24] gives a thorough overview of calculating signal-to-noise ratio (SNR) and carrier-to-spur level to be expected based on the DDS architecture in itself. It is important to remember that this does not factor in other noise sources. Therefore equation 4.17 can be used for comparison, when comparing to measured SNR which we later will calculate from phase noise measurements using equation 4.4. In summary, to calculate the Signal-to-Noise ratio, the following expression can be used:

$$S/N = 10\log\left[\frac{\sin^2(\frac{\pi}{N}\left(\frac{\pi}{MN}\right)^2}{(\frac{\pi}{N})^2 \sin^2(\frac{\pi}{MN})}\right]$$

(4.17)

In the equation \(N = 2^k\) while \(M\) is from 4.15. Furthermore, the author remarks that this establishes the following:

- For a fixed \(N\), equation 4.17 decreases when \(M\) is smaller, worsening the SNR. This is exactly as expected from earlier theory, the exception being when \(M = 1\) which yields no phase truncation spurs.

- For a fixed \(M\), equation 4.17 increases when \(N\) is increased, meaning that we can improve the SNR to the point of theoretically setting our desired SNR based purely on the number of sine samples in the look-up table.

Continuing, when examining the corner cases for a fixed \(N\), meaning \(M = 2\) and \(M = \infty\), we get the following expressions for calculating SNR. This requires a sufficiently large \(N\), which in the book is defined as \(N > 10\):

$$S/N_{\text{max}} = 20\log[\cot(\frac{\pi}{2N})]$$

\[\approx 20\log(N) - 10\log(\pi/2)\]

\[\approx 6.02k - 3.92\text{dB} \] (4.18)

$$S/N_{\text{min}} = \left[\frac{[\sin(\pi/N)/(\pi/N)]^2}{1 - (\sin(\pi/N)/(\pi/N))^2}\right]$$

\[\approx 20\log(N) - 10\log(\pi^2/3)\]

\[\approx 6.02k - 5.17\text{dB}\] (4.19)

Where equation 4.19 is the theoretical SNR when \(M = 2\) while equation 4.18 is for best case scenarios when \(M = \infty\).

The Carrier-to-spur level can be calculated. This is also explained in chapter 5 of [24] and I refer to the source for a full elaboration. The following simplification is valid for all cases where the number of sine samples \(N\) is sufficiently large. In the book, the author equates any number
larger than 10 as sufficient and is therefore valid for all cases explored in this thesis. Keep in mind that \( N = 2^k \). Equation 4.20 gives the worst case spur, where the spur energy is consolidated in one single spur, while equation 4.21 gives the Carrier-to-spur level for the best case scenario.

\[
\frac{C}{S_{\text{max}}} = 20 \cdot \log\left[\cot\left(\frac{\pi}{2N}\right)\right] \\
\approx 20 \log(N) - 20 \log\left[\frac{\pi}{2}\right] \\
\approx 6.02k - 3.92\text{dB} 
\]

Equation 4.20

\[
\frac{C}{S_{\text{max}}} = 20 \cdot \log[N + 1] \\
\approx 20 \log(N) \\
= 6.02k 
\]

Equation 4.21

The final approximations are, according to [24], a result of expanding the argument of the logarithmic function in 4.20, retaining only the first significant term\(^1\). The carrier-to-spur is also known as Spurious Free Dynamic Range, which in dB is calculated by subtracting the magnitude of the largest spur from the carrier magnitude, as shown in equation 4.22

\[
\text{SFDR(dB)} = \text{Carrier Amplitude(dB)} - \text{Largest Spur Amplitude (dB)} 
\]

Equation 4.22

4.3 DAC noise and aliasing

The digital-to-analog converter also has its own inherent noise characteristics which have to be accounted for. The DAC is not a perfect converter and the voltage related to each conversion level is not exact. In other words the DAC is not linear. The linearity error of the DAC is usually provided by the manufacturer in the data sheet. According to [24], the largest noise source in a DDS is not the digital errors caused by phase truncation and quantization noise, but analog errors introduced by the DAC. Choosing a DAC with good characteristics for the application in question is therefore very important.

For all scenarios where the output amplitude is less than 14 bits, the output is connected to the uppermost bits of the DAC, while the rest is left as 0, this is done to ensure that the full scale amplitude of the DAC is utilized, despite the output signal not making use of all the DAC-bits. This also means that the LSB absolute value increases accordingly.

\(^1\)See page 37 of [24]
4.3.1 Aliasing

Errors caused by the DAC relate to the spurious errors that appear. The DDS is a sampled system and therefore generates images at multiples of the carrier frequency, but also generate spurs at frequencies that equal the sum and difference of the carrier and the DAC sample frequency. Spurs that appear outside the Nyquist zone are aliased into the zone and appear as spurs. This follows from the trigonometric identity of adding two sine waves, as described in equation 4.23. A more thorough explanation is offered in [21]:

\[
\sin \theta_1 \sin \theta_2 = \frac{1}{2} \cos(\theta_1 - \theta_2) - \frac{1}{2} \cos(\theta_1 + \theta_2)
\] (4.23)

Applying this to our signals, we get the sum of the carrier and clock, and the difference of the carrier and clock. This also holds for harmonics of the carrier frequency. These may be located outside the Nyquist zone and will thereby create aliases. The location of these alias spurs can be estimated by equation 4.24, where \( N \) is the integer of the harmonic, \( f_{out} \) is the DDS carrier frequency and \( f_s \) is the sampling clock rate.

\[
f_{alias} = N \cdot f_{out} - f_s
\] (4.24)

4.4 Quantization noise and finite precision of sine samples

Quantization noise occurs as a result of the discrete sampling of the original continuous sine wave. When calculating and creating the sine wave LUT-samples, discrete points in the sine wave are sampled and stored. These points may not correspond fully with a bit value within the bit range of the output. This in turn leads to the value either being rounded up or down to the closest bit value. This round-off error is the quantization error.

The size of the quantization error for a given sample is always within this criteria:

\[
e_q \leq \pm \frac{\Delta_A}{2}
\] (4.25)

Where \( \Delta_A \) is the quantization step size, otherwise known as 1 LSB (least significant bit), and is related to the amplitude bit length as shown in equation 4.26, where \( m \) is the bit length of the sine ROM.

\[
\Delta_A = \frac{1}{2^m}
\] (4.26)

Later we will see from Matlab calculations that this is correct, giving rectangular error plots. Moreover, this implies that the absolute error of the quantization noise is directly linked to the precision and resolution of the
sine samples.

With increasing amplitude resolution, the absolute value of the quantization noise decreases. As we get an increasing number of possible amplitude values with increased bit length of the stored sine sample, the amount of round-off error decreases, decreasing quantization noise.

We already have described equations that describe the SNR and carrier-to-spur ratios caused by the number of addresses in the sine ROM. In [24], Vankka describes equations that allow us to calculate the carrier-to-spur ratio caused by the number of amplitude bits.

First, it is important to note that no quantization error occurs when the output frequency is a multiple of the system clock, as this means the output address is at the correct quantization level each time. Frequencies that satisfy criteria are for example \( f_{\text{clk}}/4 = 25 \text{ MHz} \).

Vanka further describes that if the maximum quantization error occurred for each sample, and that the energy is concentrated in one spur, the carrier-to-noise ratio would be given by equation 4.27.

\[
\frac{C}{S} = -3.01 + 6.02m \text{ dBC}
\]  

(4.27)

We will come back to this during the tests later in chapter 6.
Chapter 5

Implementation

5.1 Hardware

5.1.1 Kintex 7 KC705 Evaluation Board

The Kintex 7 FPGA family is part of Xilinx’ 7 Series FPGAs. In the hierarchy of the 7 Series, the Kintex 7 is the second-most advanced FPGA, second only to the high-performance Virtex 7 [30]. This FPGA was chosen for use in this thesis due to the RIMFAX FPGA prototype using an older Virtex 5, which at the outset of this thesis was considered for the flight version of the radar.

The KC705 Evaluation Board contains a XC7K235T device, which has the following features [31]:

- 326 080 Logic Cells
- 50 950 Slices
- 4000 Kb Block RAM\(^1\)

The Evaluation Board also features USB/JTAG [32] connectivity in order to communicate with a computer running the Vivado Design Suite, which was used for all parts of development, from writing code, synthesis, implementation and finally generating and uploading the bitstream file to the FPGA.

5.1.2 Texas Instruments DAC5675A

The 14 bit Texas Instruments DAC5675A which is used on RIMFAX and in this thesis, offers up to 400 MSPS (MegaSamples Per Second), and LVDS input. In this thesis the DAC has been driven by a 100MHz external clock provided from the FPGA[33]. 14 bit output range means the least significant bit that can be detected is equal to an amplitude change of \(2^{-14}\). This is larger than all but one test scenario, which uses a 18 bit output, which is

\(^1\)Kilobits
truncated to 14 bits on the output.

5.2 Phase accumulator structure

The phase accumulator was implemented as outlined in the theory in chapter 2. It consists only of an adder and a register storing the current accumulated phase, which for each clock cycle is added with the input frequency tuning word. Furthermore, the accumulator adds an extra LSB for every second clock cycle, thereby for a $M$ bit accumulator emulating a $M + 1$ bit accumulator, where the LSB always is 0 giving a constant odd FTW. This is done in accordance with theory in chapter 4, to reduce phase truncation spurs. Figure 5.1 shows a block schematic of the entity with inputs and outputs.

![Figure 5.1: Figure showing the entity of the phase accumulator. The output is used for addressing the sine ROM, after being truncated to appropriate length.](image)

5.3 Compression of Look-Up Table

The existing DDS prototype currently in use by the RIMFAX team utilizes no specific compression method, other than exploiting the quarter sine symmetry. Depending on the results of this thesis, some elements may influence the final flight design of the RIMFAX DDS. Therefore the tested methods were consciously chosen for potential easy implementation into the RIMFAX system. This meant avoiding methods that required large changes in structure.

Sunderland’s compression method is the simplest compression method described in this thesis. It adds relatively little additional logic compared to a DDS utilizing standard quarter sine wave symmetry. The additional logic comes in the form of an extra adder for summing the two LUT outputs,
and registers associated with splitting the phase output into the subcomponents A, B and C.

Likewise, the Nicholas-method, which as we know basically is an extension of the Sunderland-method, also was a suitable candidate. The difference between these two methods are in the selection of the sine samples, which is a process done by Matlab and therefore does not impose any extra complexity. Nicholas’ method also does not exclusively demand the same bit lengths for the phase subcomponents, however this extra flexibility was not explored further in this thesis.

Kenji Terai’s compression method, which in theory abolishes the approximation error of the sine waves is promising as well, at least on paper. However in this thesis the method was not selected for further testing and implementation. Work was performed to try and generate the VHDL-code for this DDS-configuration, but was stopped prior to completion.

While the previous two methods both only need an additional adder to produce the sine output, in Kenji Terai’s method we need a multiplier. In the paper, he proposes utilizing an 18x18 multiplier, which is a dedicated multiplier commonly found in FPGAs. In RTAX2000SL this is not the case [11], requiring the multiplier to be implemented in logic. Multipliers are very logic-intensive, and therefore is a possible source of considerable resource consumption. As one of the goals of this thesis is to identify methods of reducing the footprint of the DDS, this counted in favor of not pursuing this method. Avoiding multipliers in the design has also already been used as a specified advantage of the Sunderland - and Nicholas-methods.

Furthermore, even if FFI had used an FPGA which had 18x18 multipliers, such as the related RTAX2000D from the RTAX-DSP family [11], these multipliers are generally intended for digital signal-processing tasks. In RIMFAX’ case they would therefore be very desirable for digital filtering of the received signals and is yet another reason for why this method was deemed not fit for further exploration in this thesis.

The final choice on compression methods were:

- Sunderland based compression
- Nicholas’ modified Sunderland compression

These were compared against a regular quarter sine wave DDS and a DDS based on the LogiCore IP DDS found in Vivado.

Custom code was written for all implementations, except the IP based solution. For this configuration only minor alterations were done to the VHDL-wraper. Specific alterations are listed in the subsection for each corresponding test scenario.
5.3.1 Matlab

Matlab was used for calculating the sine samples. Several scripts were made, each generating sine samples for a given compression method. The script was also extended to include calculations of quantization errors, confirming the theory stated in chapter 4. Additionally, the mean error was also calculated.

The scripts were written as functions. Input arguments were the bit size of the respective phase subsets in accordance with the given compression method. The number of amplitude output bits was also an argument, as well as the filenames of the final VHDL-files for each look-up table. After generating the sine samples and calculating the error values, the script autogenerated VHDL code for each look-up table, which was ready for use and synthesis by Vivado. This saved a lot of time and allowed for a wide range of possible LUT permutations as opposed to writing the VHDL files more manually for each permutation of the DDS.

A simple emulation script was also implemented which gave a preview of the time domain output from the LUTs given a specific FTW.

Matlab was also used for collating results from the spectrum analyzer into plots for easier analysis. These scripts also performed calculations on the measured signals. Matlab scripts used in this thesis are enclosed in appendix B.

5.3.2 VHDL Code Overview

The custom written VHDL code was written to allow for easy switching between configurations within the same compression method, as well as when changing the method. For the Sunderland - and Nicholas-compression methods this meant using generics to easily switch between the various bit lengths for amplitude and look-up table address bits. For these two methods the DDS-code was otherwise identical, as both utilized two look-up tables and used the same partial phase information for look-up table addressing.

A modified version was used for the DDS configuration using a quarter sine wave symmetry approach only. This basically entailed reducing the complexity of the Sunderland-Nicholas code to only use one look-up table. Otherwise the code was left as similar as possible.

LogiCore IP VHDL-files were automatically generated by the Vivado design tool, based on the desired settings provided to the tool. For some cases small custom changes had to be made to acquire the exact wanted design. This is specified for all cases in the corresponding subsections later in the thesis.
In addition to the modules relevant to the DDS itself, a debouncer-circuit was written. When pressing the pushbuttons on the Kintex 7 evaluation board, a small circuit is required to ensure only a single keypress is detected. A spring loaded pushbutton will, on a microscopic scale, actually bounce back and forth for a while after the user pushes the button. Instead of a single keypress, many thousands may be detected before the switch settles. We can counter this

**Top Level VHD-file**

In this section an overview of the top level VHDL-file for the custom designed DDS-configurations is given. This file was largely left unchanged in between configurations.

It contained the following main items:

- Custom DDS design, imported as component
- Xilinx Clock Manager IP for generating correct clock frequency to the DDS
- Frequency Tuning Word selection logic, for easy switching between output frequencies

Figure 5.2 shows the entity of the top level file. This file was for the most part constant for all tests. The only parameter that was changed was the output bit length, which was changed as a generic in the entity of the code.

The Kintex-7 has a differential system clock generated by a 200 MHz on board oscillator. As previously mentioned these signals are sent to the clock manager IP which in turn creates a single clock signal driving the rest of the DDS-system.

Output signals from the top level is the sine amplitude from the look-up table, which can be adjusted in length by changing the generic. Last we have the clock signal for the digital-to-analog converter, which is connected to the output of the clock manager. The clock frequency of the DAC matches the system clock frequency of the DDS.

The Clock Manager IP was instantiated as a component, using the differential clock signals as input signals. Output from the IP was the single clock signal used as the system clock for the DDS, which also was connected to the `dac_clk` output functioning as the input clock to the DAC5675 digital-to-analog converter.

Furthermore, the top level file also takes in a debouncer which is connected to the `ftw_sw`. 
Figure 5.2: Figure showing the entity of the top level file. Inputs are the two differential clock signals, the reset signal and the FTW switch signal. The output of the top level file is the final sine sample which is then passed to the DAC.

| GENERICS | |
|---|---|---|---|
| Generic | Type | Description | Size |
| OUTPUT_LENGTH | Integer | Define output length | 8-14 |

| SIGNAL DECLARATIONS | |
|---|---|---|---|
| Signal | IN/OUT | Description | Size |
| clk_in_p | IN | Positive differential clock | 1 |
| clk_in_n | IN | Negative differential clock | 1 |
| reset | IN | Active high reset | 1 |
| ftw_sw | IN | Switch FTW | 1 |
| amplitude_out | OUT | Sine amplitude output | OUTPUT_LENGTH |
Table 5.2: My caption

<table>
<thead>
<tr>
<th>Generic</th>
<th>Type</th>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTW_LENGTH</td>
<td>Integer</td>
<td>Define output length</td>
<td>16, 32, 48</td>
</tr>
<tr>
<td>PHASE_LENGTH</td>
<td>Integer</td>
<td>Total phase for LUT-addressing</td>
<td>9, 12, 15</td>
</tr>
<tr>
<td>AMPL_LENGTH</td>
<td>Integer</td>
<td>Output length</td>
<td>4, 8, 14, 18</td>
</tr>
<tr>
<td>LUT_ADDR_LENGTH</td>
<td>Integer</td>
<td>Length of LUT input address</td>
<td>6, 8, 10</td>
</tr>
<tr>
<td>PARTIAL_ADDR_LENGTH</td>
<td>Integer</td>
<td>Length of A, B and C</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>COARSE_OUT_LENGTH</td>
<td>Integer</td>
<td>Length of ROM1 output</td>
<td>3, 7, 13, 15</td>
</tr>
<tr>
<td>FINE_OUT_LENGTH</td>
<td>Integer</td>
<td>Length of ROM2 output</td>
<td>0, 0, 4, 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal</th>
<th>IN/OUT</th>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>IN</td>
<td>System clock</td>
<td>1</td>
</tr>
<tr>
<td>reset</td>
<td>IN</td>
<td>Active high reset</td>
<td>1</td>
</tr>
<tr>
<td>ena</td>
<td>IN</td>
<td>Enable for phase accumulator</td>
<td>1</td>
</tr>
<tr>
<td>ftw_in</td>
<td>IN</td>
<td>FTW input</td>
<td>FTW_LENGTH</td>
</tr>
<tr>
<td>amplitude_out</td>
<td>OUT</td>
<td>Sine amplitude output</td>
<td>AMPL_LENGTH</td>
</tr>
</tbody>
</table>

5.3.3 Sunderland and Nicholas DDS VHDL-file

For both the Sunderland and Nicholas based DDS-configurations, the actual DDS VHDL component was the same, due to both functioning in the same manner, the difference lying in how the samples for the look-up table were calculated.

Figure 5.3 shows the entity of the DDS VHDL file.

![Figure 5.3: Illustration of the DDS entity. The inputs include a clock signal, reset, an enable signal (always held high) and the selected FTW, which is selected in a process in the top level file. Output is the final sine sample, which is passed to a top level register before being sent to the DAC.](image)

In the same manner as the top level file, reusability has been an important
factor in writing this file. Therefore the only changes in between configuration changes is to change the necessary generics. Otherwise the file remains the same.

Internal signals were added to create the circuit structure that was outlined in chapter 3 and figure 3.2, describing Sunderland and Nicholas Compression. Worth noticing are synchronizing registers which turned out to be necessary to synch the MSB bit from the phase accumulator to the corresponding sine value, which had to propagate through the look-up table and adders for summing the sine ROM samples, before being concatenated together. Not synchronizing the signals lead to the sign bit being associated with the wrong sine sample, possibly creating problems in the sections where the phase accumulator rolls over. This could lead to added noise due to a small number of clock cycles each period having the wrong sign bit.

The handling of addresses and sine samples were split into two processes, one which was tasked with handling the phase accumulator output and directing the various partial phase addresses to the correct sine ROM. The second process handled the output from the sine ROMs, summing them and inverting when the output was in the third or fourth quadrant of the sine wave. The DDS takes in three components: The coarse sample LUT, the fine sample LUT and the phase accumulator component. Two synchronous processes were responsible for correct addressing of the look-up tables and for finalising the sine output.

For more details, the reader is referred to appendix C for the full VHDL code.

5.3.4 Phase accumulator implementation

As described earlier the basic phase accumulator works by adding the frequency tuning word to the previously calculated phase.

The truncation of the phase bits prior to the look-up table is known to be a source of potential noise in a DDS. This is explained in more detail in chapter 4.

With this knowledge we set out to write a phase accumulator that can ensure the greatest common divisor will always be 1 as this theoretically will provide the best output performance.

To ensure this, Nicholas proposed adding a D-flip flop and an inverter, connected to the carry-in input of the adder in the accumulator, adding a single LSB for every second clock pulse. This has the effect of emulating an N+1 bit adder, where we see the N topmost bits, with an LSB always being 1, effectively ensuring an odd numbered FTW regardless of the value being present on the input. This adds only a 0.5LSB error to the phase accumulator output, or in different terms $0.5\Delta f$, where $\Delta f$ is the smallest frequency
increment possible with a given phase accumulator bit length.

In the VHDL-code, the functionality is easily implemented by means of a simple enable signal, which for every second clock period adds an LSB to the FTW-input which is summed with the accumulated phase.

### 5.4 Quarter Sine DDS

The Quarter Sine DDS was built on the Sunderland/Nicholas VHDL-file. They function in the same manner, except for just having one sine ROM instead of two. Otherwise the functionality was the same.

### 5.5 LogiCORE IP DDS

The IP DDS was created using block schematics in Vivado and connecting the blocks together in the graphical user interface. Menus allowed for modifying each block to have the desired parameters. A block schematic of the IP DDS is enclosed in appendix D.

### 5.6 Generating sine wave look-up tables

The sine ROM samples were generated in Matlab, based on the respective compression method. Below the size of look-up tables in each test scenario is presented. The quarter sine symmetry DDS is used as a base line which the Sunderland - and Nicholas compression implementations are compared against.

The IP DDS also utilizes the symmetry of the quarter sine wave [34], but also offers additional compression by using Taylor series correction. However, this makes use of DSP slices on the Kintex 7 FPGA. This is not an option on the RTAX2000SL [11], and therefore was not implemented.
Table 5.4: Table showing the test parameters for Test 1: Varying amplitude bit length.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Fixed/Adjusted</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Tuning Word</td>
<td>Fixed length</td>
<td>48 bits</td>
</tr>
<tr>
<td>Amplitude</td>
<td>Adjusted</td>
<td>4, 8, 14, 18 bits</td>
</tr>
<tr>
<td>LUT address</td>
<td>Fixed length</td>
<td>15 bits</td>
</tr>
</tbody>
</table>

Table 5.5: Sine ROM size for Quarter Sine DDS with 15 address bits, and amplitude bits varying from 4-18 bits.

<table>
<thead>
<tr>
<th>QUARTER SINE SYMMETRY DDS</th>
<th>Amplitude length</th>
<th>ROM size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 bit</td>
<td>98304</td>
</tr>
<tr>
<td></td>
<td>8 bit</td>
<td>229376</td>
</tr>
<tr>
<td></td>
<td>14 bit</td>
<td>425984</td>
</tr>
<tr>
<td></td>
<td>18 bit</td>
<td>557056</td>
</tr>
</tbody>
</table>

5.7 Test 1: Amplitude bit length test

In this first test we adjusted the bit length of the output amplitude, which reduces the size of the sine sample ROM.

The samples were generated in Matlab using scripts enclosed in appendix B. Table 5.4 shows the parameters for this test.

Quarter Sine Symmetry LUT

FFI’s current RIMFAA DDS prototype utilizes the quarter sine symmetry described in chapter 3. Compared to a straight-forward approach where the entire sine wave is stored in the ROM, this requires only a quarter of the resources.

Table 5.5 shows the size of the sine ROM for the test with adjusted amplitude length. For a N bit output, the ROM stores a N-1 bits long amplitude, as the MSB, functioning as a sign bit, comes straight from the phase accumulator output.

Size of the look-up table was calculated using equation 5.1.

\[ 2^{\text{LUT-input length}} \cdot \text{Amplitude Length} \quad (5.1) \]

Sunderland compression LUT

In chapter 3 we were acquainted with Sunderland’s compression method, which utilizes trigonometric identities to approximate a sine wave [23]. By splitting the sine sample ROM into two smaller look-up tables, the resource
Table 5.6: Number of bits required to store sine ROMs for Sunderland Compression DDS for varying amplitude bit lengths, and 15 address bits.

<table>
<thead>
<tr>
<th>Amplitude Length*</th>
<th>ROM1 Size</th>
<th>ROM2 Size</th>
<th>Total Size</th>
<th>Compression Rate**</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bit</td>
<td>3072</td>
<td>0</td>
<td>3072</td>
<td>32***</td>
</tr>
<tr>
<td>8 bit</td>
<td>7168</td>
<td>0</td>
<td>7168</td>
<td>32</td>
</tr>
<tr>
<td>14 bit</td>
<td>13312</td>
<td>4096</td>
<td>17408</td>
<td>24.47</td>
</tr>
<tr>
<td>18 bit</td>
<td>17408</td>
<td>7168</td>
<td>24576</td>
<td>22.67</td>
</tr>
</tbody>
</table>

Table 5.7: Number of bits required to store sine ROMs for Nicholas Compression DDS for varying amplitude bit lengths, and 15 address bits.

<table>
<thead>
<tr>
<th>Amplitude Length*</th>
<th>ROM1 Size</th>
<th>ROM2 Size</th>
<th>Total Size</th>
<th>Compression Rate**</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bit</td>
<td>3072</td>
<td>0</td>
<td>3072</td>
<td>32</td>
</tr>
<tr>
<td>8 bit</td>
<td>7168</td>
<td>0</td>
<td>7168</td>
<td>32</td>
</tr>
<tr>
<td>14 bit</td>
<td>13312</td>
<td>3072</td>
<td>16384</td>
<td>26</td>
</tr>
<tr>
<td>18 bit</td>
<td>17408</td>
<td>7168</td>
<td>24576</td>
<td>22.67</td>
</tr>
</tbody>
</table>

consumption could be further reduced.

For the test varying the output amplitude bits a configuration of 10 bit look-up tables were chosen, leading to a 15 bit phase-to-amplitude address in addition to the two most significant bits from the phase accumulator as quadrant - and sign control bits. Table 5.6 lists the required resources for this implementation and compares it to a DDS using quarter sine symmetry only, just like FFI’s current prototype DDS, for the various output amplitude bit lengths.

Size of each look-up table was calculated using equation 5.1, where the LUT input length is 1 bit less than the amplitude output.

Nicholas compression LUT

Table 5.7 shows the size of the Nicholas compression sine look-up tables and compares the size to a quarter sine symmetry implementation.

Summary after Synthesis and Implementation

Implemented on the Kintex 7, the resource consumption as stated in the post-implementation report can be seen in table 5.8. Here we compare all four DDS configurations and how much resources Vivado allocates to each.

In terms of quantization errors, we still get values within the ±0.5LSB. However we also need to take in consideration that by reducing the ampli-
56 CHAPTER 5. IMPLEMENTATION

Table 5.8: Resource utilization as reported by Vivado post-implementation for Test 1: Varying amplitude bit lengths.

<table>
<thead>
<tr>
<th>RESOURCE CONSUMPTION 4 BIT AMPLITUDE</th>
<th>Quarter Sine DDS</th>
<th>Xilinx IP DDS</th>
<th>Sunderland DDS</th>
<th>Nicholas DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-flops</td>
<td>107</td>
<td>141</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>LUT</td>
<td>67</td>
<td>87</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>BRAM</td>
<td>3</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>BUFG</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Memory LUT</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESOURCE CONSUMPTION 8 BIT AMPLITUDE</th>
<th>Quarter Sine DDS</th>
<th>Xilinx IP DDS</th>
<th>Sunderland DDS</th>
<th>Nicholas DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-flops</td>
<td>111</td>
<td>153</td>
<td>89</td>
<td>89</td>
</tr>
<tr>
<td>LUT</td>
<td>68</td>
<td>94</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>BRAM</td>
<td>7</td>
<td>2</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>BUFG</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Memory LUT</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESOURCE CONSUMPTION 14 BIT AMPLITUDE</th>
<th>Quarter Sine DDS</th>
<th>Xilinx IP DDS</th>
<th>Sunderland DDS</th>
<th>Nicholas DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-flops</td>
<td>101</td>
<td>172</td>
<td>85</td>
<td>77</td>
</tr>
<tr>
<td>LUT</td>
<td>63</td>
<td>105</td>
<td>52</td>
<td>63</td>
</tr>
<tr>
<td>BRAM</td>
<td>13</td>
<td>3.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>BUFG</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Memory LUT</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESOURCE CONSUMPTION 18 BIT AMPLITUDE</th>
<th>Quarter Sine DDS</th>
<th>Xilinx IP DDS</th>
<th>Sunderland DDS</th>
<th>Nicholas DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-flops</td>
<td>101</td>
<td>176</td>
<td>89</td>
<td>75</td>
</tr>
<tr>
<td>LUT</td>
<td>68</td>
<td>107</td>
<td>54</td>
<td>68</td>
</tr>
<tr>
<td>BRAM</td>
<td>13</td>
<td>4.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>BUFG</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Memory LUT</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

titude bits, the absolute value of 1 LSB increases. This can affect the output noise by increasing the amplitude of spurs, as described in Chapter 4.

5.8 Adjusting sine ROM address length

The second test scenario was designed to see whether adjusting the address length to the look-up table would change the output characteristics. This meant adjusting the amount of truncated phase bits, decreasing phase resolution and accordingly the number of discrete sine samples for a sine wave.

The parameters for the test are shown in table 5.9. For the test using 15 bit phase length, the measurement data from the corresponding test in the previous test scenario were utilized.

It is important to emphasize that the address lengths provided describe the
Table 5.9: Parameters for Test 2: Varying phase address bit length for sine ROM addressing.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Fixed/Adjusted</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Tuning Word</td>
<td>Fixed length</td>
<td>48 bits</td>
</tr>
<tr>
<td>Amplitude</td>
<td>Fixed length</td>
<td>14 bits</td>
</tr>
<tr>
<td>LUT address</td>
<td>Adjusted</td>
<td>9, 12, 15 bits</td>
</tr>
</tbody>
</table>

Table 5.10: Sine ROM size for Quarter Sine DDS with varying phase address bits, and 14 amplitude bits (13 sine ROM amplitude bits, 1 sign bit from phase accumulator)

<table>
<thead>
<tr>
<th>QUARTER SINE SYMMETRY DDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine ROM address length</td>
</tr>
<tr>
<td>9 bit</td>
</tr>
<tr>
<td>12 bit</td>
</tr>
<tr>
<td>15 bit</td>
</tr>
</tbody>
</table>

number of bits used as input to the look-up table (prior to splitting into subsets such as in Sunderland and Nicholas based compression), just as was the case in the previous test scenario.

Table 5.11 displays the resource consumption of Sunderland, 5.12 for Nicholas and 5.10 for Quarter Sine configuration. each configuration with the parameters listed in 5.9. Post-Implementation utilization numbers from Vivado are presented in table 5.13. 15 bit phase is the same configuration as the 14 bit amplitude configuration in test 1. For the resource utilization for this configuration the reader is referred to table 5.8. What is evident is that the Nicholas Compression DDS offers the lowest resource requirements of the implemented configurations, as expected from theory.

5.9 Adjusting the Frequency Tuning Word

When adjusting the FTW bit length, we don’t earn any reduction in ROM size. However, the phase accumulator is reduced in size and thus will provide some reduction in overall design size. The configuration for this

<table>
<thead>
<tr>
<th>Table 5.11: My caption</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUNDERLAND COMPRESSION</td>
</tr>
<tr>
<td>Phase Address Length*</td>
</tr>
<tr>
<td>9 bit</td>
</tr>
<tr>
<td>12 bit</td>
</tr>
<tr>
<td>15 bit*</td>
</tr>
</tbody>
</table>
Table 5.12: Sine ROM size in no of bits for Nicholas Compression DDS for Test 2: Varying phase address bit lengths.

<table>
<thead>
<tr>
<th>LUT Address Length</th>
<th>Coarse ROM Size (bits)</th>
<th>Fine ROM Size (bits)</th>
<th>Total ROM size (bits)</th>
<th>Compression ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 bits</td>
<td>832</td>
<td>448</td>
<td>1280</td>
<td>5.2</td>
</tr>
<tr>
<td>12 bits</td>
<td>3328</td>
<td>1280</td>
<td>4608</td>
<td>11.56</td>
</tr>
<tr>
<td>15 bits</td>
<td>13312</td>
<td>3072</td>
<td>16384</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 5.13: Resource utilization as reported by Vivado for Test Scenario 2, adjusting sine ROM address bit length. As is clearly evident from the table, Nicholas compression offers the least utilization, which is in accordance with previously described theory.

<table>
<thead>
<tr>
<th>RESOURCE CONSUMPTION 9 BIT PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Quarter Sine DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
<tr>
<td>Xilinx IP DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
<tr>
<td>Sunderland DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
<tr>
<td>Nicholas DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESOURCE CONSUMPTION 12 BIT PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Quarter Sine DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
<tr>
<td>Xilinx IP DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
<tr>
<td>Sunderland DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
<tr>
<td>Nicholas DDS</td>
</tr>
<tr>
<td>Flip-flops</td>
</tr>
<tr>
<td>LUT</td>
</tr>
<tr>
<td>BRAM</td>
</tr>
<tr>
<td>BUFG</td>
</tr>
<tr>
<td>Memory LUT</td>
</tr>
</tbody>
</table>
test scenario utilized 15 bits of the phase accumulator output and 14 bit amplitude output. FTW has been implemented in versions of 18 bits, 32 bits and 48 bits. Reducing the FTW bit length leads to less frequency resolution and larger frequency offset from the fact that the FTW is truncated to an integer value during the conversion from a decimal number to the binary integer word. Sine ROM size is in this test the same as described for 14 bit output in tables 5.7, 5.6 and 5.5. As no major reduction in resource utilization is expected, the utilization reports are not included in the thesis.  

\footnote{Some reduction is likely, especially as with 18 bit FTW the adder structure gets less complex. However, with 15 phase bits and 14 amplitude bits, the dominating factor is still the sine ROM.}
Chapter 6

Tests and results

6.1 Test setup

All DDS implementations were synthesized and implemented on the KC705 Evaluation Board. The board was connected to the DAC via an FMC-XM105 debug board\[fmc\]. Power to the DAC was supplied by a 3.3V power supply, while the external clock input to the DAC was supplied by the dac\_clk-signal from the FPGA, through one of the GPIO SMA-connectors. The DAC output was connected to a spectrum analyzer, via a low-pass filter[35] with cut-off frequency at 50 MHz. Settings on the analyzer were set to the default settings, which for frequency spectrum measurements meant a frequency range of 100 MHz, and a resolution bandwidth of 2 MHz. For phase noise measurements, the analysis ran from a frequency offset of 1 kHz - 5 MHz, of which the section from 1 kHz - 30 kHz is examined below. A block schematic of the test layout is shown in figure 6.1

6.2 Output analysis

6.2.1 Test outline

Based on the implemented DDS variations explained in the previous chapter a specific test plan was designed. Test frequencies were set based on the RIMFAX DDS frequency requirements:

- First test at 18.75 MHz. Lowest frequency of RIMFAX DDS
- Second test at 24.5 MHz
- Third test at 30.00 MHz
- Fourth and final test at 37.5 MHz. Highest frequency of RIMFAX DDS

Test frequency number two and three were chosen due to their respective Frequency Tuning Words, which will lead to phase truncation errors, and therefore will have more noise components than frequency one and four which do not show these effects, due to all bits that are truncated will be
Figure 6.1: Block schematic showing an outline of the test configuration for all tests conducted in this thesis. Not shown is the low-pass filter which was connected directly between the wire from the DAC and the spectrum analyzer.
zero.

Three test scenarios were designed:

- Constant length of FTW and phase bits. Amplitude bits are altered and tested at 4, 8, 14 and 18 bits.
- Constant length of FTW and amplitude bits. Phase addressing bits are altered and tested at 9 bits, 12 bits and 15 bits.
- Constant length of phase addressing bits and amplitude bits. FTW is altered and tested at 24 bits, 32 bits and 48 bits.

During each test, the configuration bit stream was uploaded to the Kintex 7 FPGA which in turn was connected to the DAC. The DAC output was connected to the spectrum analyzer through a low-pass filter and the following output characteristics were recorded and stored for further analysis in Matlab:

- Frequency spectrum spanning the range 0 Hz - $f_s$ (100 MHz).
- Single-sideband phase noise analysis from 1 kHz-30 kHz.
- Vivado post-implementation reports of on-board resource utilization

Integrated phase noise is calculated by the spectrum analyzer, for the full measurement range. This value is for the original, transmitted signal without correction for phase noise correlation. This value is used to calculate the SNR of the signal, based on equation 4.4.

Integrated phase noise as experienced in the final measurement of the radar has been calculated for the range 1 kHz-30kHz, per the objectives of the thesis. The measured phase noise has been multiplied with the correction factor due to short range to the desired target, as described in equation 4.6. For RIMFAX the target is ground structures at a depth of at least 10 meters, however this is a minimum requirement in the specification for the instrument and depending on the material in the ground it is quite possible for the radar to image structures at even greater depths. We have in this thesis made a conservative estimate of the phase noise by using a time delta factor $\tau_d = 1 \cdot 10^{-6}$ seconds. This corresponds to a target distance of 150 meters, which can be stated to be a conservative estimate of the noise performance, as shorter distances increases the correlation of transmitted and received phase noise and thereby increases noise performance even further.

We can expect harmonics of the fundamental to appear as spurs. This also included aliases of harmonics outside the Nyquist zone, which fold back into the zone, due to the sampled nature of the system and heterodyning. Using equation 4.24 we can predict the locations of possible spurs. Spur locations in the results that refer to harmonics have been calculated by this equation.
Table 6.1: Worst case Carrier-to-spur ratio based on amplitude bit length. Valid for all DDS implementations in Test 1

<table>
<thead>
<tr>
<th>No of Amplitude Bits</th>
<th>Worst Case Spur (dBc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bits</td>
<td>21.07</td>
</tr>
<tr>
<td>8 bits</td>
<td>45.15</td>
</tr>
<tr>
<td>14 bits</td>
<td>81.27</td>
</tr>
<tr>
<td>18 bits</td>
<td>105.35</td>
</tr>
</tbody>
</table>

6.3 Test scenario 1: Varying output amplitude bit length

This test was designed to analyze the different characteristics gained from altering the number of output amplitude bits. Fewer bits reduces the number of possible amplitude values, reducing the resolution of the sine wave. From chapter 4 we know this reduces the theoretical best case SNR caused by the DDS. The reduced amplitude resolution increases quantization noise, which increases amplitude of spurs.

With only a 4 bit output, the maximum number of unique amplitude values in a single quarter sine wave were limited to 16. Therefore this was the least promising candidate from a performance standpoint. Few possible amplitude values increased the chance of phase error due to round-off to an amplitude value that had a significant offset from the desired value.

Theoretical SNR and SFDR from this is given by equation 4.18, as the phase accumulator has been designed to emulate an always odd FTW, which ensures that the number of phase truncation spurs \( Y \) is as large as possible, as described in equations 4.15 and 4.16. It should be noted that these values are based upon the number of samples in the sine ROM, and do not account for other noise sources. For example the sine amplitude resolution may be the dominating factor in terms of spur noise.

\[
\text{SNR} \approx 6.02k - 5.17 \text{ dBC} = 6.02 \cdot 15 - 3.92 \text{ dBC} = 86.38 \text{ dBC} \quad (6.1)
\]

Carrier-to-spur ratio caused by the number of addressing bits is:

\[
\frac{C}{S_{\text{Address}}} = 6.02k = 90.3 \text{dBC} \quad (6.2)
\]

This test is to study the effects of shortening amplitude bit length. Our next step therefore is to calculate predicted worst case carrier-to-spur ratio due to amplitude bit length. This is done as described in equation 4.27. Calculated values are shown in table 6.1.
From table 6.1, it is clear that for 4, 8 and 14 bit amplitude length, this is the dominating factor compared to the address bit length. These are all worst case scenarios, and the reader is encouraged to read [24] chapter 5.2 for more details.

By increasing the number amplitude bits to 8, the amount of possible amplitude values grow to 256, meaning a much finer resolution than in the previous scenario. This implies smaller quantization errors, which is supported by the values in table 6.1 predicting smaller spurs with increasing amplitude length. With 14 bits of amplitude output bit length the number of possible amplitude values is 16384. This is also the range of the DAC5675 digital-to-analog converter.

Finally, we have performed a test at 18 amplitude output bits. The number of possible amplitude values in this case is 262144. As the DAC only supports 14 bits output, the four lowermost bits are truncated before being sent to the DAC. This is included as a control measure, to see whether any additional performance can be harnessed from the finer resolution. The expectation is that this will produce the same results as for 14 bit output.

Below follows the results for each of the DDS configurations, with the test parameters. Noise characteristics have also been analyzed. To reduce the number of graphs in the thesis, a subset is presented, while the rest are enclosed in appendices E.1, E.2 and E.3.

### 6.4 Test 1: Quarter Sine DDS

Here follows the test results from the Quarter Sine Symmetry DDS, for varying amplitude bit lengths. Phase noise data is summarized at the end of the section.

#### 6.4.1 4 bit output

At 4 amplitude bits the largest spur is predicted to be located at, at worst, -21.07 dB from the carrier. Figure 6.2 shows the frequency spectra for the four test frequencies. Frequency range of the measurement is from 0-100 MHz.

Several spurs are evident in the frequency spectra. For 18.75 MHz, 30.0 MHz and 37.5 MHz the spurs are all clearly at locations we can predict by using equation 4.24, by testing for various harmonics, meaning they are either harmonics or aliases folded back into the Nyquist zone. Spurs that appear outside this zone in the spectrum are not included. For 24.5 MHz the calculated spur locations are too close to the carrier to be clearly visible at the resolution bandwidth of the measurement (2 MHz). SFDR is calculated as described in chapter 4 and equation 4.22

- 18.75 MHz: Spurs at 6.25 MHz (5th harmonic), 31.25 MHz (7th harmonic), 43.25 MHz (3rd harmonic and largest). The largest spur is
at -30.93 dBm\(^1\). SFDR: 26.76 dB

- **24.75 MHz**: Individual spurs are hard to discern. Calculations reveal spurs of 3rd and 5th harmonic should appear at 26.5 MHz and 22.5 MHz respectively. Largest visible spur peak at 14.26 MHz at -34.38 dBm. SFDR: 29.98 dB

- **30.0 MHz**: Two distinct spurs at 10 MHz (3rd harmonic spur) and 50.0 MHz (5th harmonic). Largest spur is caused by the 3rd harmonic, at -29.69 dBm. SFDR: 23.6 dB

- **37.5 MHz**: One spur within the Nyquist zone, at 12.5 MHz caused by the 3rd harmonic of the fundamental. Spur amplitude at -32.09 dBm. SFDR: 26.28 dB

### 6.4.2 8 bit output

At 8 amplitude bits the largest spur is predicted to be located at, at worst, -21.07 dB from the carrier. Figure 6.3 shows the frequency spectra for the four test frequencies. Frequency range is the same as in the previous section.

The spectrum for all four frequencies are in general of better performance than at 4 bit. However, it should be noted that this also entails an increase

\(^1\)dBm: Decibels relative to 1 mW
Figure 6.3: Frequency spectra for Quarter Sine Symmetry DDS at 8 bits amplitude length from 16 to 256 possible amplitude values.

There are still some visible spurs:

- **18.75 MHz**: Indications of a spur at 6.25 MHz (-52.74 dBm), 31.25 MHz (-56.84 dBm) and a clear spur at 43.75 MHz (-52.03) caused by the 3rd harmonic. SFDR: 48.41 dB
- **24.5 MHz**: No clear spur visible, however 3rd and 5th harmonic spur are hidden due to resolution bandwidth limitations in the measurement. SFDR based on noise floor of $\approx -59$ dBm: 54.96 dB (uncertainty connected with the possible spurs at 26.5 MHz and 22.5 MHz)
- **30.0 MHz**: One clear spur, at 10.0 MHz, caused by the 3rd harmonic. amplitude: -47.08 dBm. SFDR: 42.53 dB
- **37.5 MHz**: Indications of spur at 12.5 MHz caused by the 3rd harmonic. amplitude: -55.08 dBm. SFDR: 49.61 dB

### 6.4.3 14 bit output and 18 bit output

14 bit output should theoretically give the best performance possible given the DAC max output bit length of 14 bits, with a predicted worst case
Quarter Sine Symmetry, 14 bit amplitude

![Graphs showing frequency spectra for different frequencies](image)

**Figure 6.4:** Frequency spectra for Quarter Sine Symmetry DDS at 14 bits amplitude length

carrier-to-spur ratio of 81.27 dB. From figure 6.4 it is clear that this is not the case. These spectra do not agree with this number as the amplitude of the spurs are much larger:

- **18.75 MHz:** Largest spur at 43.75 MHz, from the 3rd harmonic. amplitude -43.64 dBm. SFDR: 40.14 dB
- **24.5 MHz:** Largest spur at 49.2 MHz, indicating it is caused by the 2nd harmonic. amplitude: -39.97 dBm. SFDR:35.98 dB
- **30.0 MHz:** Largest spur at 10.0 MHz (3rd harmonic), amplitude -29.69 dBm. Also spur at 50 MHz, amplitude -39.5 dBm. SFDR: 23.92 dB
- **37.5 MHz:** No spurs inside Nyquist zone

For the 18 amplitude bit measurement, the spurs are not only connected to the harmonics of the carrier. Possible error sources causing these readings are elaborated on in the discussion section.

18 bit output should in theory not provide any performance improvements over 14 bits due to DAC limitations. The 4 lowermost bits will here be truncated. This measurement was done as a control check regarding the 14 bit output to verify that additional resolution would not yield better results.
However also this measurement exhibits characteristics that are not expected. The worst case spur caused by the finite amplitude bit length is for 18 bits: -105.35 dBc as described in table 6.1. The address length causes a worst case spur of -86.38 dBc. Largest spurs are listed below:

- 18.75 MHz: Spur at 29.97 MHz, amplitude -36.65 dBm. SFDR = 33.1 dB
- 23.5 MHz: Spur at 17.15 MHz, amplitude -36.72 dBm. SFDR = 32.7 dB
- 30 MHz: Spur at 38.6 MHz, amplitude -36.77 dBm. SFDR: 32.23 dB
- 37.5 MHz: Spur at 24.5 MHz, amplitude -37.69 dBm. SFDR = 32.29 dB

6.4.4 Phase Noise analysis

Integrated phase noise and SNR are shown in 6.2 for the original measurement phase noise, while table 6.3 shows the corresponding phase noise and SNR after calculating with the correction factor. Plots associated with these measurements are enclosed in appendix E.1, as figures E.5 for the original measurement and E.6 for the calculated phase noise for the IF signal.
Table 6.2: Test 1: Table showing integrated phase noise for Quarter Sine DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-50.58</td>
<td>-50.81</td>
<td>-51.21</td>
<td>-50.80</td>
</tr>
<tr>
<td>24.5</td>
<td>-48.27</td>
<td>-48.79</td>
<td>-48.76</td>
<td>-48.27</td>
</tr>
<tr>
<td>30.0</td>
<td>-46.95</td>
<td>-46.81</td>
<td>-46.45</td>
<td>-46.86</td>
</tr>
<tr>
<td>37.5</td>
<td>-44.84</td>
<td>-44.21</td>
<td>-44.28</td>
<td>-44.45</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>50.58</td>
<td>50.81</td>
<td>51.21</td>
<td>50.80</td>
</tr>
<tr>
<td>24.5</td>
<td>48.27</td>
<td>48.79</td>
<td>48.76</td>
<td>48.27</td>
</tr>
<tr>
<td>30.0</td>
<td>46.95</td>
<td>46.81</td>
<td>46.45</td>
<td>46.86</td>
</tr>
<tr>
<td>37.5</td>
<td>44.84</td>
<td>44.21</td>
<td>44.28</td>
<td>44.45</td>
</tr>
</tbody>
</table>

Table 6.3: Test 1: Table showing integrated phase noise for Quarter Sine DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1 kHz-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-95.09</td>
<td>-95.33</td>
<td>-95.27</td>
<td>-95.13</td>
</tr>
<tr>
<td>24.5</td>
<td>-92.50</td>
<td>-93.02</td>
<td>-92.90</td>
<td>-92.79</td>
</tr>
<tr>
<td>30.0</td>
<td>-91.08</td>
<td>-91.18</td>
<td>-91.18</td>
<td>-91.10</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.18</td>
<td>-89.24</td>
<td>-89.12</td>
<td>-89.22</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>95.09</td>
<td>95.33</td>
<td>95.27</td>
<td>95.13</td>
</tr>
<tr>
<td>24.5</td>
<td>92.50</td>
<td>93.02</td>
<td>92.90</td>
<td>92.79</td>
</tr>
<tr>
<td>30.0</td>
<td>91.08</td>
<td>91.18</td>
<td>91.18</td>
<td>91.10</td>
</tr>
<tr>
<td>37.5</td>
<td>89.18</td>
<td>89.24</td>
<td>89.12</td>
<td>89.22</td>
</tr>
</tbody>
</table>
Figure 6.6: Frequency spectra for IP DDS. Amplitude bit length is 4 bits. Note the similarities to the Quarter Sine DDS in figure 6.6.

### 6.5 Test 1: LogiCORE IP DDS

The IP based DDS utilizes quarter sine symmetry for minimizing resource utilization on the FPGA. The implementation is therefore largely similar to the custom Quarter Sine DDS for which test results were just presented.

#### 6.5.1 4 bit amplitude

For 4 bit amplitude the frequency spectrum is comparatively similar to the Quarter Sine DDS in figure 6.2. The IP DDS spectra for the same output bit length is shown in figure 6.6. However on closer examination it is discovered that the spurs are of lower amplitude than in the Quarter Sine DDS for the three first frequencies, while at 37.5 MHz the IP DDS performs worse than the Quarter Sine DDS.

- 18.75 MHz: Spurs at 6.25 MHz (5th harmonic), 31.25 MHz (7th harmonic), 43.25 MHz (3rd harmonic and largest). The largest spur is at -38.03 dBm. SFDR: 32.17 dB
- 24.75 MHz: Individual spurs are hard to discern. Calculations reveal spurs of 3rd and 5th harmonic should appear at 26.5 MHz and 22.5 MHz respectively. Largest visible spur peak at 12.18 MHz at -36.07 dBm. SFDR: 29.69 dB
Figure 6.7: Frequency spectra for IP DDS. Amplitude bit length is 8 bits.

- 30.0 MHz: Two distinct spurs at 10 MHz (3rd harmonic spur) and 50.0 MHz (5th harmonic). Largest spur is caused by the 3rd harmonic, at -31.31 dBm. SFDR: 24.35 dB

- 37.5 MHz: One spur within the Nyquist zone, at 12.5 MHz caused by the 3rd harmonic of the fundamental. Spur amplitude at -30.6 dBm. SFDR: 22.78 dB

6.5.2 8 bit amplitude

For amplitude length of 8 bits, the frequency spectra are shown in figure 6.7. Spurious noise is markedly lower than it was for 4 bit amplitude.

- 18.75 MHz: One spur visible, at 31.25 MHz, related to the 7th harmonic. Amplitude: -55.85 dBm. SFDR: 52.17 dB

- 24.5 MHZ: Spurs possibly hidden due to closeness to carrier and too low resolution on the measurement. SFDR based on noise floor of approximately -58 dBm: 53.87 dBm.

- 30.0 MHz: Spur at 10.0 MHz (3rd harmonic alias, and largest) and 50.0 MHz (5th harmonic alias). Amplitude largest spur: -51.52 dBm. SFDR: 46.92 dB
Figure 6.8: Frequency spectra for IP DDS. Amplitude bit length is 14 bits.

- 37.5 MHz: Spur at 12.5 MHz from 3rd harmonic. amplitude: -50.28 dBm. SFDR: 44.76 dB

### 6.5.3 14 bit

At 14 bits the spurs are now too low in amplitude to discern in the frequency spectra, as seen in figure 6.8.

SFDR is estimated based on the apparent noise floor at -59 dBm.

- 18.75 MHz: 55.43 dB
- 24.5 MHz: 55.00 dB
- 30.0 MHz: 55.5 dB
- 37.5 MHz: 54.61 dB

### 6.5.4 18 bit

18 bits does not offer any improved performance over 14 bit amplitude length, based on the measurements performed. Frequency spectra can be seen in appendix E.1, as figure E.10.
Table 6.5: Test 1: Integrated phase noise (dBc/Hz) for LogiCORE IP DDS. Frequency range 1 kHz - 30 kHz

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-50.63</td>
<td>-50.28</td>
<td>-50.97</td>
<td>-50.74</td>
</tr>
<tr>
<td>30.0</td>
<td>-46.76</td>
<td>-46.87</td>
<td>-47.06</td>
<td>-46.40</td>
</tr>
<tr>
<td>37.5</td>
<td>-44.71</td>
<td>-44.43</td>
<td>-44.39</td>
<td>-44.95</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>18.75</th>
<th>24.5</th>
<th>30.0</th>
<th>37.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bit</td>
<td>50.63</td>
<td>48.27</td>
<td>46.76</td>
<td>44.71</td>
</tr>
<tr>
<td>8 bit</td>
<td>50.28</td>
<td>48.40</td>
<td>46.87</td>
<td>44.43</td>
</tr>
<tr>
<td>14 bit</td>
<td>50.97</td>
<td>48.40</td>
<td>47.06</td>
<td>44.39</td>
</tr>
<tr>
<td>18 bit</td>
<td>50.74</td>
<td>48.09</td>
<td>46.40</td>
<td>44.95</td>
</tr>
</tbody>
</table>

6.5.5 Phase noise analysis

Phase noise measurements for the IP DDS is presented in this section, along with SNR based on the calculated integrated phase noise. Table 6.4 shows the integrated phase noise based on the original measurements from the spectrum analyzer while table 6.5 shows the associated values after applying the correction factor for correlating phase noise. Associated plots are enclosed in appendix E.1 as figures E.11 for the original phase noise measurement and E.12 for the IF signal.

6.6 Test 1: Sunderland compression DDS

Here follows the results from the Sunderland-based DDS, from Test 1, with varying amplitude bit lengths.

4 and 8 bit amplitude

At these amplitude lengths the Sunderland algorithm breaks down, as seen from table 5.6. The algorithm is based on splitting the sine values into a coarse and fine ROM, but at these amplitude lengths, there is not enough amplitude resolution for the fine ROM to contain any values. This makes the DDS function simply as a regular Quarter Sine DDS with phase resolution given by the number of address bits in the coarse ROM. In this case this means a 10 bit Quarter Sine DDS, with 4 or 8 output bits, respectively. These measurements are therefore not analysed further, however the phase noise characteristics do appear in the corresponding phase noise plots in figures 6.10 and 6.11. Integrated phase noise and SNR calculations are not included. Frequency spectra from these measurements are available to the reader in appendix E.1.
Table 6.5: Test 1: Table showing integrated phase noise for LogiCORE IP DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-95.08</td>
<td>-95.27</td>
<td>-95.25</td>
<td>-95.35</td>
</tr>
<tr>
<td>24.5</td>
<td>-92.88</td>
<td>-92.91</td>
<td>-92.89</td>
<td>-92.71</td>
</tr>
<tr>
<td>30.0</td>
<td>-91.17</td>
<td>-91.25</td>
<td>-91.06</td>
<td>-91.10</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.17</td>
<td>-89.20</td>
<td>-89.09</td>
<td>-89.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
</tr>
<tr>
<td>24.5</td>
</tr>
<tr>
<td>30.0</td>
</tr>
<tr>
<td>37.5</td>
</tr>
</tbody>
</table>

14 bit amplitude

At 14 amplitude bits we utilize the full range of the DAC and the Sunderland method functions according to theory. Frequency spectra are displayed in figure 6.9. Spurious noise is listed below, with spurs appearing at harmonics of the fundamental in accordance with equation 4.24.

From figure 6.9 we see that within the Nyquist zone, no spurs appear above the noise floor for any test frequency, except for one in the plot of $f = 24.5$ MHz, with a frequency of 43.9 MHz and an amplitude of -54.32 dBm. This does not correspond to a harmonic or alias appearing within the Nyquist zone.

SFDR for these measurements are therefore, based on a noise floor of -60dBm:

- 18.75 MHz: 56.44 dB
- 24.5 MHz: 55.89 dB
- 30.0 MHz: 55.5 dB
- 37.5 MHz: 54.44 dB

It should be noted that this is based on the current resolution bandwidth of the measurement and might need revision with a finer resolution.

18 bit amplitude

This test was done as a control measure to compare with 14 bit amplitude to see whether any more performance gain could be harnessed despite the
truncation of the four lowermost bits. As in the case of 14 bits amplitude, there are no discernable spurs within the Nyquist zone. Based on a noise floor of -60 dBm the SFDR gets the same values as described above for 14 bit amplitude. Phase noise and calculated SNR based on these analyses are discussed in the next section. The reader is referred to appendix E.1 and figure E.16 for the frequency spectra from this measurement.

6.6.1 Phase noise analysis

Figures 6.10 display phase noise plots for the Sunderland DDS, with varying amplitude bit length. These are the original phase noise measurements from the spectrum analyzer. Integrated phase noise and SNR is presented in table 6.6. Figure 6.11 and table 6.7 show the related phase noise values after being multiplied by the correction factor for correlating phase noise between transmitted and received signal for short target distances. As these figures and tables show, the phase noise is not affected by the changes in amplitude bit length, from an offset of approximately 2 kHz.
Figure 6.10: These plots show measured phase noise in the range 1 kHz - 30 kHz for all test frequencies using the Sunderland Compression DDS, with amplitude bits in the range 4, 8, 14 and 18 bits.

Table 6.6: Test 1: Table showing integrated phase noise for Sunderland DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

| Test 1: Integrated phase noise (dBc/Hz) for Sunderland DDS. Frequency range 1 kHz - 30 kHz | No. of amplitude bits |
|---|---|---|---|---|
| Frequency (MHz) | 4 bit | 8 bit | 14 bit | 18 bit |
| 18.75 | -51.24 | -49.95 | -50.79 | -50.75 |
| 24.5 | -48.88 | -47.73 | -48.87 | -48.59 |
| 30.0 | -46.24 | -46.70 | -46.56 | -46.71 |
| 37.5 | -44.77 | -44.83 | -44.89 | -44.24 |

Difference in phase noise from lowest to highest frequency:

| 6.47 | 5.12 | 5.9 | 6.51 |

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>51.24</td>
<td>49.95</td>
<td>50.79</td>
<td>50.75</td>
</tr>
<tr>
<td>24.5</td>
<td>48.88</td>
<td>47.73</td>
<td>48.87</td>
<td>48.59</td>
</tr>
<tr>
<td>30.0</td>
<td>46.24</td>
<td>46.70</td>
<td>46.56</td>
<td>46.71</td>
</tr>
<tr>
<td>37.5</td>
<td>44.77</td>
<td>44.83</td>
<td>44.89</td>
<td>44.24</td>
</tr>
</tbody>
</table>
Figure 6.11: These plots show emulated phase noise in the IF signal after correction factor has been applied. Frequency offset is in the range 1 kHz - 30 kHz for all test frequencies. Sunderland Compression DDS, with amplitude bits in the range 4, 8, 14 and 18 bits.

Table 6.7: Test 1: Table showing integrated phase noise for Sunderland Compression, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Test 1: Integrated phase noise (dBc/Hz) for Sunderland DDS</th>
<th>No. of amplitude bits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Frequency (MHz)</strong></td>
<td>4 bit</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
</tr>
<tr>
<td>18.75</td>
<td>-95.34</td>
</tr>
<tr>
<td>30.0</td>
<td>-90.99</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Difference from lowest to highest frequency:</th>
<th>6.0672</th>
<th>5.8273</th>
<th>5.9502</th>
<th>5.9888</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal-to-Noise Ratio (dB)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.75</td>
<td>95.34</td>
<td>95.04</td>
<td>95.12</td>
<td>95.23</td>
</tr>
<tr>
<td>24.5</td>
<td>92.99</td>
<td>92.80</td>
<td>92.92</td>
<td>92.92</td>
</tr>
<tr>
<td>30.0</td>
<td>90.99</td>
<td>91.06</td>
<td>91.09</td>
<td>91.14</td>
</tr>
<tr>
<td>37.5</td>
<td>89.27</td>
<td>89.22</td>
<td>89.17</td>
<td>89.24</td>
</tr>
</tbody>
</table>
6.7 Test 1: Nicholas Compression DDS

Here follows test results for the Nicholas compression method DDS, for varying amplitude bit lengths. Calculations of spur frequencies are performed as for Sunderland compression. No further explanation for the calculations are thereby given.

6.7.1 4 bit and 8 bit amplitude

As seen in table 5.7 in chapter 5, the Nicholas algorithm does not work at 4 and 8 bit amplitude length, just like was the case with the Sunderland algorithm at the same values. The reader is referred to the previous section for further elaboration. The frequency spectra are attached in Appendix E.1. The configurations do however appear in the phase noise analyses in the end of this section.

6.7.2 14 bit amplitude

At 14 bit amplitude, the compression method works as intended, which is evident from table 5.7 in chapter 5, showing both the fine ROM and coarse ROM contain actual data. No spurs are apparent in the spectra of any of the four tested frequencies. Noise floor is at approximately -60 dBm, giving an estimate of the SFDR:

- 18.75 MHz: 56.44 dB
- 24.5 MHz: 55.99 dB
- 30.0 MHz: 55.50 dB
- 37.5 MHz: 54.60 dB

6.7.3 18 bit amplitude

Just as was the case at 14 bit amplitude, no discernable spurs are present. In both cases it should be noted that a narrower resolution bandwidth would have resulted in a lower noise floor, which could reveal additional spurs. The reader is referred to appendix E.1, figure E.22. As expected there are no visible differences from 14 bit amplitude.

An example of this is estimating the SFDR of 18.75 MHz. The noise floor is also here at approximately -60 dBm. SFDR, based on the amplitude of the carrier calculates to 56.44 dBm, which is identical with the 14 bit calculation.

6.7.4 Noise analysis

Phase noise plots were generated from 1 kHz-30 kHz. Table 6.8 shows the original measurement from the spectrum analyzer and figure 6.9
Figure 6.12: This figure shows frequency spectrum for all four test frequencies, for Nicholas Compression DDS with 14 bit amplitude.

Table 6.8: Test 1: Table showing integrated phase noise for Nicholas Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Test 1: Integrated phase noise (dBc/Hz) for Nicholas Compression DDS. Frequency range 1 kHz - 30 kHz</th>
<th>No. of amplitude bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (MHz)</td>
<td>4 bit</td>
</tr>
<tr>
<td>18.75</td>
<td>-51.24</td>
</tr>
<tr>
<td>24.5</td>
<td>-48.34</td>
</tr>
<tr>
<td>30.0</td>
<td>-46.72</td>
</tr>
<tr>
<td>37.5</td>
<td>-44.38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
</tr>
<tr>
<td>24.5</td>
</tr>
<tr>
<td>30.0</td>
</tr>
<tr>
<td>37.5</td>
</tr>
</tbody>
</table>
Table 6.9: Test 1: Table showing integrated phase noise for Nicholas Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>4 bit</th>
<th>8 bit</th>
<th>14 bit</th>
<th>18 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-95.19</td>
<td>-95.06</td>
<td>-95.12</td>
<td>-95.21</td>
</tr>
<tr>
<td>24.5</td>
<td>-92.81</td>
<td>-92.94</td>
<td>-92.95</td>
<td>-92.93</td>
</tr>
<tr>
<td>30.0</td>
<td>-91.21</td>
<td>-91.27</td>
<td>-90.97</td>
<td>-91.32</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.30</td>
<td>-89.39</td>
<td>-89.22</td>
<td>-89.03</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>18.75</th>
<th>24.5</th>
<th>30.0</th>
<th>37.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bit</td>
<td>95.19</td>
<td>92.81</td>
<td>91.21</td>
<td>89.30</td>
</tr>
<tr>
<td>8 bit</td>
<td>95.06</td>
<td>92.94</td>
<td>91.27</td>
<td>89.39</td>
</tr>
<tr>
<td>14 bit</td>
<td>95.12</td>
<td>92.95</td>
<td>90.97</td>
<td>89.22</td>
</tr>
<tr>
<td>18 bit</td>
<td>95.21</td>
<td>92.93</td>
<td>91.32</td>
<td>89.03</td>
</tr>
</tbody>
</table>

6.8 Overview test scenario 2: Varying Look-Up Table address length

In this test scenario, the number of phase bits that were used for sine ROM addressing were adjusted. Amplitude output was set at 14 bit, while phase bits were tested at 9 bits, 12 bits and 15 bits. The latter is the same configuration as for 14 bit output in Test 1. Therefore, these data are reused and not based on new measures. In these sections, the reader is referred to the appropriate section in Test 1.

Based on equation 4.21 we can predict the worst case spur amplitude for spurs dictated by the number of samples that are stored in the sine ROM i.e. how many address bits are available:

\[
\frac{C}{S_m} ax = 6.02k = 6.02 \cdot 9 = 54.18 dB
\]  (6.3)

\[
\frac{C}{S_m} ax = 6.02k = 6.02 \cdot 12 = 72.24 dB
\]  (6.4)

\[
\frac{C}{S_m} ax = 6.02k = 6.02 \cdot 15 = 90.3 dB
\]  (6.5)

Comparing this to table 6.1, it is shown that the largest spur will be dictated by the address bit length for 9 and 12 bit address length, while at 15 bits, the largest spur is dictated by the number of amplitude bits in the sine samples.
6.9 Test 2: Quarter Sine DDS

6.9.1 9 bit address length

Frequency spectra for this measurement series is shown in figure 6.13. From the calculations in the introduction to this test the worst case spur was estimated at -54.18 dB. The noise floor of the measurements, which lie close to this value, makes it difficult to discern spurs that may exist in the spectrum. The exception is 37.5 MHz, at which a small spur is visible at an amplitude of -56.82 dBm.

Based on the noise floor which lies at approximately -59 dBm, SFDR is approximated:

- 18.75 MHz: 55.43 dB
- 24.5 MHz: 54.98 dB
- 30.0 MHz: 54.51 dB
- 37.5 MHz: 51.4 dB
Figure 6.14: Frequency Spectrum for Quarter Sine DDS for Test 2, with 12 phase bits for addressing the sine ROM.

### 6.9.2 12 bit address length

Frequency spectra for this measurement series is shown in figure 6.14. From the calculations in the introduction to this test the worst case spur was estimated at -72.24 dB. The measured noise floor lies approximately 12 dB higher, which means this spur will be drowned out by other noise sources in the spectrum. Figure 6.14 supports this as no spurs are visible.

Based on the noise floor which also here lies at approximately -59 dBm, SFDR is approximated:

- 18.75 MHz: 55.6 dB
- 24.5 MHz: 55.72 dB
- 30.0 MHz: 54.24 dB
- 37.5 MHz: 52.6 dB

### 6.9.3 15 bit address length

Results in this section are based on the original measurements done in Test 1, as these configurations are identical. The reader is referred to section 6.4.3.
Table 6.10: Test 2: Table showing integrated phase noise for Quarter Sine DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-51.24</td>
<td>-49.94</td>
<td>-51.19</td>
</tr>
<tr>
<td>24.5</td>
<td>-48.54</td>
<td>-48.68</td>
<td>-48.76</td>
</tr>
<tr>
<td>30.0</td>
<td>-46.98</td>
<td>-46.16</td>
<td>-46.45</td>
</tr>
<tr>
<td>37.5</td>
<td>-44.40</td>
<td>-44.78</td>
<td>-44.28</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>51.24</td>
<td>49.94</td>
<td>51.19</td>
</tr>
<tr>
<td>24.5</td>
<td>48.34</td>
<td>48.03</td>
<td>48.76</td>
</tr>
<tr>
<td>30.0</td>
<td>46.98</td>
<td>46.16</td>
<td>46.45</td>
</tr>
<tr>
<td>37.5</td>
<td>44.40</td>
<td>44.78</td>
<td>44.28</td>
</tr>
</tbody>
</table>

6.9.4 Noise analysis

Calculated integrated phase noise is shown in table 6.18 for calculations based on the spectrum analyzer. In table 6.19.

6.10 Test 2: LogiCORE DDS IP

6.10.1 9 bit address length

At 9 bit of phase bits for sine ROM addressing, the frequency spectra shown in figure 6.15 shows spurs in the spectra for 24.5 MHz and 30 MHz. We know that the largest spur in the worst case scenario should have a max amplitude of -54.18 dBc.

The spurs are located at:

- 24.5 MHz: One spur at 12.5 MHz, amplitude of -55.5 dBm. SFDR: 50.05 dB
- 30.0 MHz: One spur at 10.0 MHz, amplitude: -53.6 dBm. SFDR: 49.07 dB

For the 18.75 MHz and 37.5 MHz measurements the noise floor is at approximately -60 dBm, giving an approximated SFDR of:

- 18.75 MHz: 56.4 dB
- 37.5 MHz: 54.55 dB
Table 6.11: Test 2: Table showing integrated phase noise for Quarter Sine DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of phase bits</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td></td>
<td>-95.19</td>
<td>-95.06</td>
<td>-95.12</td>
</tr>
<tr>
<td>24.5</td>
<td></td>
<td>-92.81</td>
<td>-92.94</td>
<td>-92.95</td>
</tr>
<tr>
<td>30.0</td>
<td></td>
<td>-91.21</td>
<td>-91.27</td>
<td>-90.97</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-89.30</td>
<td>-89.39</td>
<td>-89.22</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>95.19</td>
<td>95.06</td>
<td>95.12</td>
</tr>
<tr>
<td>24.5</td>
<td>92.81</td>
<td>92.94</td>
<td>92.95</td>
</tr>
<tr>
<td>30.0</td>
<td>91.21</td>
<td>91.27</td>
<td>90.97</td>
</tr>
<tr>
<td>37.5</td>
<td>89.30</td>
<td>89.39</td>
<td>89.22</td>
</tr>
</tbody>
</table>

Figure 6.15: Frequency Spectrum for LogiCORE DDS IP for Test 2, with 9 phase bits for addressing the sine ROM.
Table 6.12: Test 2: Table showing integrated phase noise for LogiCORE DDS IP, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of phase bits</th>
<th>9</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td></td>
<td>-50.50</td>
<td>-51.16</td>
<td>-50.97</td>
</tr>
<tr>
<td>24.5</td>
<td></td>
<td>-48.16</td>
<td>-48.84</td>
<td>-48.40</td>
</tr>
<tr>
<td>30.0</td>
<td></td>
<td>-46.87</td>
<td>-46.77</td>
<td>-47.06</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-44.87</td>
<td>-44.44</td>
<td>-44.39</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>50.50</td>
<td>51.16</td>
</tr>
<tr>
<td>24.5</td>
<td>48.16</td>
<td>48.84</td>
</tr>
<tr>
<td>30.0</td>
<td>46.87</td>
<td>46.77</td>
</tr>
<tr>
<td>37.5</td>
<td>44.87</td>
<td>44.44</td>
</tr>
</tbody>
</table>

6.10.2  12 bit address length

12 bit means the worst case spur should be at -72.24 dB. No visible spurs are present in the spectra. They are included in appendix E.2 as figure E.30.

SFDR can be approximated. The noise floor is at approximately -60 dBm.

- 18.75 MHz: 56.4 dB
- 37.5 MHz: 54.54 dB

6.10.3  15 bit address length

This is the same configuration as in Test 1 for 14 amplitude bits. The reader is referred to section 6.5.3 for more details.

6.10.4  Phase noise analysis

Phase noise calculations are presented in this section. Table 6.12 shows integrated phase noise calculated from spectrum analyzer measurements. SNR is also calculated, based on equation 4.4. Table 6.13 shows the corresponding value for integrated phase noise after multiplying with the correction factor for correlating phase noise.

6.11  Test 2: Sunderland Compression DDS

6.11.1  9 bit address length

For 9 bit phase address length, the Sunderland partial ROMs both have a 6 bit input address, allowing for reduction of resource utilization. Performance should however be on par with a regular DDS with 9 phase bits for
Table 6.13: Test 2: Table showing integrated phase noise for LogiCORE DDS IP, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Test 2: Integrated phase noise (dBc/Hz) for LogiCORE IP DDS</th>
<th>No. of phase bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency (MHz)</td>
<td>9 bit</td>
</tr>
<tr>
<td>18.75</td>
<td>-95.17</td>
</tr>
<tr>
<td>24.5</td>
<td>-92.93</td>
</tr>
<tr>
<td>30.0</td>
<td>-91.07</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.13</td>
</tr>
</tbody>
</table>

| Signal-to-Noise Ratio (dB)                                  |       |       |       |
| 18.75                                                       | 95.17  | 95.50  | 95.25  |
| 24.5                                                        | 92.93  | 92.91  | 92.89  |
| 30.0                                                        | 91.07  | 91.14  | 91.06  |
| 37.5                                                        | 89.13  | 89.10  | 89.09  |

Sine ROM addressing.

Frequency spectra reveal no large discrepancies compared to the Quarter Sine DDS with 9 bit address length. One spur is apparent in the spectrum at 30.0 MHz at 1.0 MHz, caused by the 3rd harmonic, with an amplitude of -52.57 dBm. This lead to a SFDR at 30.0 MHz of 47.88 dBm.

Frequency spectra are enclosed in appendix E.2.

6.11.2 12 bit address length

Frequency spectra for 12 bit address length reveal no spurs visible above the noise floor. SFDR is therefore comparable to previously calculated values for Quarter Sine DDS for 12 bits.

Frequency spectra are enclosed in appendix E.2.

6.11.3 15 bit address length

Results in this section are based on the original measurements done in Test 1, as these configurations are identical as the 14 amplitude bit DDS in section 6.6.

6.11.4 Noise analysis

Here follows integrated phase noise calculations for varying phase bits in Sunderland DDS. Table 6.14 shows the integrated phase noise of the measured output from the DDS, while table 6.15 shows the integrated phase noise after multiplying with correction factor for close targets. Associated plots are enclosed in appendix E.2.
Table 6.14: Test 2: Table showing integrated phase noise for Sunderland Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-50.60</td>
<td>-50.11</td>
<td>-50.79</td>
</tr>
<tr>
<td>24.5</td>
<td>-48.23</td>
<td>-48.50</td>
<td>-48.87</td>
</tr>
<tr>
<td>30.0</td>
<td>-46.42</td>
<td>-46.72</td>
<td>-46.56</td>
</tr>
<tr>
<td>37.5</td>
<td>-44.74</td>
<td>-44.97</td>
<td>-44.88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
</tr>
<tr>
<td>24.5</td>
</tr>
<tr>
<td>30.0</td>
</tr>
<tr>
<td>37.5</td>
</tr>
</tbody>
</table>

Table 6.15: Test 2: Table showing integrated phase noise for Sunderland Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-95.16</td>
<td>-94.98</td>
<td>-95.11</td>
</tr>
<tr>
<td>24.5</td>
<td>-92.83</td>
<td>-92.86</td>
<td>-92.91</td>
</tr>
<tr>
<td>30.0</td>
<td>-91.23</td>
<td>-91.22</td>
<td>-91.08</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.16</td>
<td>-89.31</td>
<td>-89.16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
</tr>
<tr>
<td>24.5</td>
</tr>
<tr>
<td>30.0</td>
</tr>
<tr>
<td>37.5</td>
</tr>
</tbody>
</table>
Figure 6.16: Frequency Spectrum for Nicholas Compression DDS for Test 2, with 9 phase bits for addressing the sine ROM.

6.12 Test 2: Nicholas Compression DDS

6.12.1 9 bit address length

The Nicholas Compression DDS reveals one spur in the spectra provided, and as with the Sunderland configuration it is located at 30.0 MHz. However the amplitude is markedly higher, with an amplitude of -45.22 dB, giving a SFDR of 40.66 dB, when compared to the carrier amplitude. The noise floor for 24.5 MHz is also higher than the rest of the frequency spectra, at approximately -53 dBm, instead of closer to -60 dBm for the others. SFDR for 24.5 MHz is approximated to be: -49 dB. For 18.75 MHz and 37.5 MHz the SFDR is comparable to Sunderland Compression and Quarter Sine DDS for 9 bit address length.

6.12.2 12 bit address length

For 12 address bits, we still have some spurs, just like at 9 phase bits, in the frequency spectrum for 24.5 MHz and 30.0 MHz. For 24.5 MHz the spur appears at 12.25 MHz with an amplitude of -56.57 dBm, giving a SFDR of 52.50 dB. The location does not match any alias for the fundamental, but it is of note that the frequency is exactly half of the carrier frequency. For 30 MHz the spur is linked to the 3rd harmonic, at 10.0 MHz with an amplitude of -56.15 dBm, giving a SFDR of: 51.61 dB. SFDR for 18.75 MHz and 37.5
MHz are comparable to Sunderland and Quarter Sine DDS for the same configuration, with no discernable spurs in the spectra.

6.12.3 15 bit address length

Results in this section are based on the original measurements done in Test 1, for 14 bit amplitude, as these configurations are identical. For results, see section 6.7.2.

6.12.4 Phase noise analysis

Table 6.16 display the integrated phase noise for the Nicholas Compression DDS for 9, 12 and 15 bit long phase output. SNR is also calculated. The frequency range of the integrated phase noise is 1 kHz - 30 kHz. Table 6.17 shows the phase noise for the IF signal.

6.13 Test scenario 3: Varying Frequency Tuning Word length

The final test scenario entailed adjusting the length of the input FTW. According to theory this reduces the frequency resolution. Each bit removed
Table 6.16: Test 2: Table showing integrated phase noise for Nicholas Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of phase bits</th>
<th>9 bits</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.75</td>
<td>9 bits</td>
<td>-50.59</td>
<td>-50.40</td>
<td>-50.92</td>
</tr>
<tr>
<td>24.5</td>
<td>12 bit</td>
<td>-48.47</td>
<td>-48.77</td>
<td>-48.70</td>
</tr>
<tr>
<td>30.0</td>
<td>15 bit</td>
<td>-46.77</td>
<td>-46.89</td>
<td>-46.90</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-44.52</td>
<td>-46.14</td>
<td>-44.36</td>
</tr>
</tbody>
</table>

Table 6.17: Test 2: Table showing integrated phase noise for Nicholas Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of phase bits</th>
<th>9 bit</th>
<th>12 bit</th>
<th>15 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>9 bit</td>
<td>-95.26</td>
<td>-95.26</td>
<td>-95.12</td>
</tr>
<tr>
<td>24.5</td>
<td>12 bit</td>
<td>-92.88</td>
<td>-92.95</td>
<td>-92.95</td>
</tr>
<tr>
<td>30.0</td>
<td>15 bit</td>
<td>-91.18</td>
<td>-91.19</td>
<td>-90.97</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-89.28</td>
<td>-89.60</td>
<td>-89.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>95.26</td>
</tr>
<tr>
<td>24.5</td>
<td>92.88</td>
</tr>
<tr>
<td>30.0</td>
<td>91.18</td>
</tr>
<tr>
<td>37.5</td>
<td>89.28</td>
</tr>
</tbody>
</table>
halves the frequency resolution, something demonstrated by equation 2.2, where \( N \) is the bit length of the FTW and phase accumulator. This means that the step between each possible frequency also doubles by reducing the bit length of the FTW by 1.

Shortening the FTW means the integer value also decreases. Output frequency is given by equation 2.1. However, this value has to be truncated to an integer value in order to be used in a digital system. For a given frequency this means that the remainder after the decimal sign is discarded, leading to a frequency error in the output frequency compared to the desired frequency. This error is however even at 18 bit FTW negligible compared to the frequencies that is tested. E.g. for 24.5 MHz, the FTW has been calculated in decimal form, converted into an 18 bit integer, and then compared to the decimal number. This \( \Delta \text{FTW} \) when put into equation 2.1, gave an frequency offset of 20 mHz. This is therefore not of any concert to the measurements performed.

From equations 4.15 and 4.16 we know that the number of spurs \( Y \) in the first Nyquist zone can be determined as seen below for 18 bits and 32 bits FTW, when we use 15 bits of the phase accumulator output:

\[
Y = \frac{2^{j-k}}{\text{GCD}(\text{FTW}, 2^{j-k})} - 1 = \frac{2^{18-15}}{1} - 1 = 2^3 - 1 = 7
\]

\[
Y = \frac{2^{j-k}}{\text{GCD}(\text{FTW}, 2^{j-k})} - 1 = \frac{2^{32-15}}{1} - 1 = 2^{17} - 1 = 131071
\]

This is also as expected from equation 4.17, where it was stated that a lower number \( M \), thereby a lower number \( Y \) would yield a worse noise characteristic due to the noise energy being consolidated into fewer spurs. At 18 bit FTW, the period of the truncated three bits is at the shortest 7 clock cycles, meaning the truncated bits, which have a sawtooth shaped period, has a higher frequency and appears as more high frequent noise, than in the case of 32 bit or 48 bit FTW where the maximum period of the phase truncation bits is considerably higher and thereby providing a low level noise, which is distributed into many smaller spurs. In this test scenario we can therefore expect the 18 bit FTW to cause added noise compared to 48 bit FTW, while the 32 bit FTW most likely will not exhibit much different characteristics from the 48 bit versions.

### 6.14 Test 3: Quarter Sine DDS

#### 6.14.1 18 bit FTW

As seen in figure 6.18, there are a number of spurs surrounding the carrier in all test frequencies. From the calculations above we expected 7 spurs due to phase truncation and the low number of bits being truncated.
Figure 6.18: Frequency Spectrum for Quarter Sine DDS for Test 3 with 18 bit FTW.

6.14.2 32 bit FTW

At 32 bit FTW the Quarter Sine DDS still exhibits behaviour that is unexpected. From the calculations in the introduction to this test the phase truncation noise should be spread over a large number of spurs, appearing almost as white noise. Bit length of sine ROM address and amplitude output also suggest a less noisy signal than what is seen in figure 6.19. This is indicative of other errors in the system. Some possible candidates that may have caused the error will be explained in the discussion section.

6.14.3 48 bit FTW

Results in this section are based on the original measurements done in Test 1 for 14 bit amplitude. The reader is referred to figure 6.4 for frequency spectrum information.

6.14.4 Phase Noise Analysis

Table 6.18 display the integrated phase noise for the Quarter Sine DDS for varying FTW bit length. SNR is also calculated. The frequency range of the integrated phase noise is 1 kHz - 30 kHz. For phase noise as experienced by the IF signal, see table 6.19.
Table 6.18: Test 3: Table showing integrated phase noise for Quarter Sine DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>No. of FTW bits</th>
<th>Frequency (MHz)</th>
<th>18 bits</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18.75</td>
<td>-50.62</td>
<td>-48.98</td>
<td>-50.79</td>
</tr>
<tr>
<td></td>
<td>24.5</td>
<td>-48.42</td>
<td>-47.96</td>
<td>-48.76</td>
</tr>
<tr>
<td></td>
<td>30.0</td>
<td>-46.28</td>
<td>-45.84</td>
<td>-46.45</td>
</tr>
<tr>
<td></td>
<td>37.5</td>
<td>-43.99</td>
<td>-45.00</td>
<td>-44.28</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>18 bits</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>50.62</td>
<td>48.98</td>
<td>50.79</td>
</tr>
<tr>
<td>24.5</td>
<td>48.42</td>
<td>47.96</td>
<td>48.76</td>
</tr>
<tr>
<td>30.0</td>
<td>46.28</td>
<td>45.84</td>
<td>46.45</td>
</tr>
<tr>
<td>37.5</td>
<td>43.99</td>
<td>45.00</td>
<td>44.28</td>
</tr>
</tbody>
</table>
Table 6.19: Test 3: Table showing integrated phase noise for Quarter Sine DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>18 bit</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-92.86</td>
<td>-92.90</td>
<td>-92.90</td>
</tr>
<tr>
<td>24.5</td>
<td>-94.95</td>
<td>-94.90</td>
<td>-95.11</td>
</tr>
<tr>
<td>30.0</td>
<td>-91.11</td>
<td>-91.00</td>
<td>-91.18</td>
</tr>
<tr>
<td>37.5</td>
<td>-89.14</td>
<td>-89.29</td>
<td>-89.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
</tr>
<tr>
<td>24.5</td>
</tr>
<tr>
<td>30.0</td>
</tr>
<tr>
<td>37.5</td>
</tr>
</tbody>
</table>

6.15 Test 3: LogiCORE DDS IP

6.15.1 18 bit and 32 bit FTW

Due to errors in the DDS implementation for these configurations, the measurements are not valid for comparison with the other DDS configurations. Figure 6.20 shows the frequency spectra from the 18 bit implementation which closely mirrors the 32 bit implementation (the latter is available in appendix E.3).

As is shown in figure 6.20 the measurements have large amounts of noise and spurs. Harmonics are of equal or higher amplitude in e.g. 30.0 MHz and 18.75 MHz. Because of this, no phase noise calculations were performed for the IP DDS with either 18 bit FTW or 32 bit FTW. The phase noise plots were generated and are enclosed in appendix E.3. The discussion section will elaborate on the cause of the observed errors.

6.15.2 48 bit FTW

This configuration is the same as the 14 bit amplitude length DDS from Test 1. Spectra can therefore be seen in figure 6.8 and phase noise characteristics are shown under 14 bit in tables 6.4 and 6.5 in the column for 14 bit amplitude.

6.15.3 Phase noise analysis

Due to the errors in the 18 bit and 32 bit configurations, the phase noise plots and tables do not provide any valuable information regarding actual DDS performance of a functioning circuit with this number of FTW bits.
The frequency spectra for this test are shown in figure 6.21. At 18 bit FTW the Sunderland Compression DDS exhibits no spurs within the Nyquist zone, except for an indicated spur at 4.67 MHz in the spectrum for \( f=18.75 \) MHz, with an amplitude of -52.04 dBm. It also appears in the spectra for output frequencies 24.5 MHz and 30.0 MHz with amplitudes of -46.64 dBm and -52.66 dBm respectively.

SFDR in these cases is thereby:

- 18.75 MHz: 48.49 dB
- 24.5 MHz: 42.62 dB
- 30.0 MHz: 48.16 dB
- 37.5 MHz approximated based on approximate noise floor level of -60 dBm: 54.54 dB

These are therefore not included in the thesis, however the plots are enclosed in appendix E.3.

6.16 Test 3: Sunderland Compression DDS

6.16.1 18 bit FTW

The frequency spectra for this test are shown in figure 6.21. At 18 bit FTW the Sunderland Compression DDS exhibits no spurs within the Nyquist zone, except for an indicated spur at 4.67 MHz in the spectrum for \( f=18.75 \) MHz, with an amplitude of -52.04 dBm. It also appears in the spectra for output frequencies 24.5 MHz and 30.0 MHz with amplitudes of -46.64 dBm and -52.66 dBm respectively.

SFDR in these cases is thereby:

- 18.75 MHz: 48.49 dB
- 24.5 MHz: 42.62 dB
- 30.0 MHz: 48.16 dB
- 37.5 MHz approximated based on approximate noise floor level of -60 dBm: 54.54 dB
6.16.2 32 bit FTW

The spectra at 32 bit FTW is comparable to the spectra for 18 bits. No new spurs appear in the Nyquist zone, and the small spur visible in figure 6.21 in the plots for 18.75 MHz, 24.5 MHz and 30.0 MHz is still present. For this measurement it is also seen in the spectrum for 37.5 MHz, at an amplitude of -49.6 dBm, giving a SFDR of: 44.22 dB.

The spectra are otherwise comparable to figure 6.21. Plots for this DDS configuration are available in appendix E.3.

6.16.3 48 bit FTW

Results in this section are based on the original measurements done in Test 1, as these configurations are identical. For results, see tables 6.4 and 6.5 for integrated phase noise and SNR-calculations.

6.16.4 Phase noise analysis

Here follows tables showing integrated phase noise for the Sunderland Compression DDS in Test 3. Table 6.20 shows the measured and calculated integrated phase noise from the spectrum analyzer while table 6.21 shows the measurements after being multiplied by the correction factor.
Table 6.20: Test 3: Table showing integrated phase noise for Sunderland Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of FTW bits</th>
<th>18 bits</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>18 bits</td>
<td>-50.78</td>
<td>-50.72</td>
<td>-50.79</td>
</tr>
<tr>
<td>24.5</td>
<td>32 bit</td>
<td>-48.66</td>
<td>-48.11</td>
<td>-48.87</td>
</tr>
<tr>
<td>30.0</td>
<td>48 bit</td>
<td>-46.87</td>
<td>-46.84</td>
<td>-46.56</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-44.24</td>
<td>-43.91</td>
<td>-44.88</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of FTW bits</th>
<th>18 bits</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>18 bits</td>
<td>50.78</td>
<td>50.72</td>
<td>50.79</td>
</tr>
<tr>
<td>24.5</td>
<td>32 bit</td>
<td>48.66</td>
<td>48.11</td>
<td>48.87</td>
</tr>
<tr>
<td>30.0</td>
<td>48 bit</td>
<td>46.87</td>
<td>46.84</td>
<td>46.56</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>44.24</td>
<td>43.91</td>
<td>44.88</td>
</tr>
</tbody>
</table>

Table 6.21: Test 2: Table showing integrated phase noise for Sunderland Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of FTW bits</th>
<th>9 bit</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>9 bit</td>
<td>-94.99</td>
<td>-95.06</td>
<td>-95.11</td>
</tr>
<tr>
<td>24.5</td>
<td>32 bit</td>
<td>-93.01</td>
<td>-92.84</td>
<td>-92.91</td>
</tr>
<tr>
<td>30.0</td>
<td>48 bit</td>
<td>-91.10</td>
<td>-91.04</td>
<td>-91.08</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-89.23</td>
<td>-89.32</td>
<td>-89.16</td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of FTW bits</th>
<th>9 bit</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>9 bit</td>
<td>94.99</td>
<td>95.06</td>
<td>95.11</td>
</tr>
<tr>
<td>24.5</td>
<td>32 bit</td>
<td>93.01</td>
<td>92.84</td>
<td>92.91</td>
</tr>
<tr>
<td>30.0</td>
<td>48 bit</td>
<td>91.10</td>
<td>91.04</td>
<td>91.08</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>89.23</td>
<td>89.32</td>
<td>89.16</td>
</tr>
</tbody>
</table>
Table 6.22: Test 3: Table showing integrated phase noise for Nicholas Compression DDS, measured by spectrum analyzer. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>No. of FTW bits</th>
<th>18 bit</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td></td>
<td>-50.75</td>
<td>-51.15</td>
<td>-50.92</td>
</tr>
<tr>
<td>24.5</td>
<td></td>
<td>-48.61</td>
<td>-48.71</td>
<td>-48.70</td>
</tr>
<tr>
<td>30.0</td>
<td></td>
<td>-46.66</td>
<td>-46.91</td>
<td>-46.90</td>
</tr>
<tr>
<td>37.5</td>
<td></td>
<td>-44.78</td>
<td>-45.34</td>
<td>-44.36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal-to-Noise Ratio (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
</tr>
<tr>
<td>24.5</td>
</tr>
<tr>
<td>30.0</td>
</tr>
<tr>
<td>37.5</td>
</tr>
</tbody>
</table>

6.17 Test 3: Nicholas Compression DDS

6.17.1 18 bit FTW

Frequency spectra are enclosed in appendix E.3, as figure E.53. Performance is comparable to Sunderland Compression DDS as seen in figure 6.21, and no spurs are visible in the Nyquist zone.

6.17.2 32 bit FTW

Frequency spectra are enclosed in appendix E.3, as figure E.54. Also in this case the performance is comparable to Sunderland Compression DDS. Still no spurs in the Nyquist zone and noise floor is comparable to both 18 bit FTW Nicholas Compression DDS and Sunderland DDS for the same configuration.

6.17.3 48 bit FTW

Results in this section are based on the original measurements done in Test 1 at 14 bit amplitude. Frequency spectra are shown in figure 6.12, along with noise characteristics in the corresponding section.

6.17.4 Phase noise analysis

6.18 Discussion

In this section a discussion of the results is presented, based on the partial results from each test.
Table 6.23: Test 3: Table showing integrated phase noise for Nicholas Compression DDS, after applying correction factor for correlating phase noise in transmitted and received signal, for a target distance of 150 meters. Also shown, SNR measured in dB. Frequency range 1-30 kHz.

<table>
<thead>
<tr>
<th>No. of FTW bits</th>
<th>Frequency (MHz)</th>
<th>18 bit</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>-95.27</td>
<td>-95.22</td>
<td>-95.12</td>
<td></td>
</tr>
<tr>
<td>24.5</td>
<td>-92.80</td>
<td>-92.90</td>
<td>-92.95</td>
<td></td>
</tr>
<tr>
<td>30.0</td>
<td>-91.14</td>
<td>-91.26</td>
<td>-90.97</td>
<td></td>
</tr>
<tr>
<td>37.5</td>
<td>-89.31</td>
<td>-89.38</td>
<td>-89.22</td>
<td></td>
</tr>
</tbody>
</table>

Signal-to-Noise Ratio (dB)

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>18 bit</th>
<th>32 bit</th>
<th>48 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.75</td>
<td>95.27</td>
<td>95.22</td>
<td>95.12</td>
</tr>
<tr>
<td>24.5</td>
<td>-2.80</td>
<td>92.90</td>
<td>92.95</td>
</tr>
<tr>
<td>30.0</td>
<td>91.14</td>
<td>91.26</td>
<td>90.97</td>
</tr>
<tr>
<td>37.5</td>
<td>89.31</td>
<td>89.38</td>
<td>89.22</td>
</tr>
</tbody>
</table>

In terms of resource utilization, it is of note that the LogiCORE IP would often require more resources on the FPGA than the Quarter Sine DDS. This is presented in tables 5.8 and 5.13 in chapter 5. The probable cause of this is the overhead caused by implementing the entire system using only IP blocks, which may contain more logic than necessary had each piece been assembled together as a whole. It was implemented using quarter sine symmetry as the only method of reducing resource consumption and would probably have taken less resources if it had been generated with area optimization in mind.

The Nicholas and Sunderland DDS’ were throughout the configurations that used the least resources, which agrees with presented theory, with Nicholas Compression DDS using slightly less resources than the Sunderland Compression DDS.

From a utilization standpoint, Nicholas compression presents itself as the preferred solution for a DDS application requiring minimal resource utilization, compared to the other methods explored in this thesis. Sunderland Compression also presents itself as a good alternative, as both offer good compression and easy implementation.

For short amplitude lengths (4 and 8 bit) we see from tables 5.6 and 5.7 that the method for both Sunderland and Nicholas breaks down. Due to the low number of possible amplitude values the second look-up table, which functions as a fine tuning device, simply does not contribute. This is due to the fact that the offset from the actual sine amplitude at a given phase value is less than the value of 1 LSB in those cases. The DDS as a result functions as a regular DDS utilizing quarter sine symmetry with phase length equal to the address length of the coarse ROM. From the partial phase compo-
nents, which in this test scenario was 5 bits, for a total address length to each look-up table of 10 bits. For these cases the DDS should then actually have less amplitude resolution than the quarter sine DDS in the same test scenario. From theory we know that this should increase the absolute error of quantization errors and increase the amplitude of spurs generated by the finite address length. This can be a likely cause of the frequency spectrum results for 4 bit amplitude, showing several prominent spurs, both for Sunderland and Nicholas based DDS. At 8 bit, the spurs are reduced drastically, but still are a factor that needs to be addressed.

14 bit amplitude demonstrated good performance for all implementations in Test 1. For a Quarter Sine DDS, the resource requirements of such a large ROM requires a FPGA with ample resources. For an application using the RTAX2000SL, it would be advisable to choose the Nicholas Compression DDS or Sunderland Compression DDS, which both offer similar noise characteristics and dynamic range at a fraction of the resource utilization of a quarter sine symmetry DDS, as seen chapter 5.

An issue that may have affected the results is the resolution bandwidth of the spectrum analyzer. This was preset to 2 MHz and was not changed for any of the measurements. Late in the process, two measurements were done at 200kHz resolution bandwidth to see how this might affect the results. A screenshot is shown in figure 6.22, here the output frequency is at 30 MHz. In the image the reader will see that increasing the resolution allows for detecting spurs and noise characteristics closer to the carrier. Especially for 24.5 MHz measurements, where the 3rd and 5th harmonic are very close to the carrier, this would be of great value in order to clearly distinguish the harmonics from the fundamental.

In section 6.4.3 the Quarter Sine DDS exhibits unexpected noise characteristics. No immediate cause is clear, however it may stem from problems in the code regarding the synchronization registers for the sign bit. In a few cases during the work with this thesis there were issues with the sign bit being too early relative to its corresponding sine sample. This led to the sign switching early at two to four places for each period of one sine wave, either at each quadrant change, or between the first and second, and third and fourth quadrants. However, by examining the frequency spectra in Matlab, the spurs are not located at spur locations that may be related to such a signal error either. Another possibility is a missing rising_edge statement in one of the processes that was discovered in the dds.vhd-file. Uncertainty regarding when the signal is updated could be a cause of timing issues when synthesis and place-and-route has been performed. The similarities between the plots in figures 6.4, 6.5 from Test 1 and 6.18 and 6.19 from Test 3 may indicate they both are impaired by a design error, rather than effects from the expected noise sources in the system. Spurs were found at frequencies aligning with harmonics and aliases thereof, and spur amplitudes relative to the carrier were within predicted amplitudes.
Figure 6.22: Screenshot from spectrum analyzer showing Sunderland DDS with 9 bit phase, at 30 MHz output frequency, 100 MHz measurement span, with resolution bandwidth at 200 kHz.
In Test 3, the IP DDS was also ridden with excessive noise. This noise exhibited characteristics of strong harmonics related to the fundamental. This error occurred for both 18 and 32 bit FTW. Upon examining the code, an error was discovered in the output signal. The TI DAC5675A works on unsigned numbers, meaning that all 0’s represents the bottom of the negative half of the sine wave, while all 1’s represent the top of the positive half. The IP does however work with signed notation, and therefore manual alterations had to be performed to make sure the MSB was inverted to work correctly with the DAC. At the same time, Vivado would regenerate the wrapper file each time any changes were made to the IP configuration, necessitating a redo of the manual alteration. Failure to do so creates a signal resembling a square wave with a curved top and bottom, as illustrated in figure 6.23. Square waves consist of a fundamental and strong odd harmonics. The frequency spectrum in figure 6.20 demonstrates strong harmonics, especially visible in the 18.75 MHz plot, which indicates that this may be the reason for the erroneous signals.

It is of note that the phase noise measurements are almost unchanged between all different test scenarios in this thesis. Figure 6.24 represents the phase noise (both original measurement, and after applying correction factor) for Quarter Sine DDS in Test 2 at 18.75 MHz while figure 6.25 represent the equivalent for Nicholas 14 bit amplitude, 15 bit phase from Test 1 to give an example. Comparing with their respective tables for integrated
Figure 6.24: The plot on the left shows the measured phase noise at \( f = 18.75 \) MHz, for varying amplitude bit lengths using a Quarter Sine DDS. Take note of the consistency between the four configurations, apparently unaffected by the change in amplitude bit length. To the right the same measurement after being multiplied with the correction factor for close targets.

One of the goals of this thesis was to see if reducing the resource consumption of the DDS still allowed for phase noise performance of minimum -80 dBc. From the basic measurements themselves we see across the board that they do not live up to this requirement. However, by factoring in the correlation of phase noise in transmitted and received signal, after mixing, we have calculated that all measured DDS-configurations comply with the requirement.

After work on this thesis had commenced, it was made known that the flight DDS for RIMFAX will utilize a 29 bit FTW and phase accumulator, 12 phase bits for sine ROM addressing and 12 amplitude bits. Both the Nicholas and Sunderland Compression methods will function with those parameters, as the papers where they are described actually do that with...
Figure 6.25: The plot on the left shows the measured phase noise at \( f = 18.75 \, \text{MHz} \), for varying phase bit lengths using a Nicholas Compression DDS. Phase noise appears comparatively equal, and also do not diverge from the Quarter Sine DDS plots in figure 6.24. To the right the same measurement after being multiplied with the correction factor for close targets.

12 phase bits and 12 amplitude bits. With a regular Quarter Sine Symmetry DDS, the sine ROM will have a size of 45056 bits, if the MSB of the amplitude output comes from the phase accumulator, meaning 11 bits are stored in the ROM. Even with retaining 14 amplitude bits and 15 bit phase input with Nicholas Compression, the ROM size will be: 16384 bits, meaning a 2.75 times smaller ROM than the 12 phase bit 11 amplitude bit Quarter Sine ROM. With 12 bit phase and 14 bit output the resource reduction gets even larger, based on table 5.12, the reduction in ROM size is 9.78 times.

From this information it should be of interest to explore whether a Nicholas Compression DDS could be utilized either in RIMFAX or future radar applications that call for FPGA based DDS with minimal footprint, while still providing good output characteristics.
Chapter 7

Summary and conclusion

In this chapter a summary of the work done in this thesis is presented and a conclusion is provided, before suggestions for future work is presented.

7.1 Summary of work done

In the course of this thesis, several methods for reducing the footprint of a FPGA-based Direct Digital Synthesizer have been explored. By looking at pros and cons of each method, a subset was chosen for further study and was implemented on a Kintex 7 FPGA Evaluation Board. The DDS was developed as a module for use on the RIMFAX FMCW-radar currently being developed at FFI for NASA’s Mars 2020-rover. When choosing the methods for further study, the specifications of the actual flight FPGA of RIMFAX, an RTAX2000, where taken into account as well and was the basis of not proceeding with one of the methods.

Additionally, a regular DDS utilizing no other compression method than the quarter symmetry of a sine wave was used. This was done as this is the approach currently used by FFI in the prototype for RIMFAX. No actual prototype code was used, and all code was written by the author. These custom written DDS-configurations were also tested against a Xilinx LogiCore DDS IP supplied by the Vivado development tool.

Three test scenarios were developed and focused on varying three attributes of the DDS, to see what effects, if any, this would have on the output sine wave signal. The first test varied the bit lengths of the sine wave output, the second test varied the bit lengths of the phase value used for look-up table addressing, while the final test varied the length of the input Frequency Tuning Word.

Results from these test were plotted in Matlab, integrated phase noise and signal characteristics were calculated and compared.

Implementations for adjusted FTW bit length for the LogiCORE IP experienced design errors, most likely caused by an inverted MSB on the
output.

7.2 Conclusions

From the work done in this thesis, we see that the Nicholas Compression DDS offers the best performance and also has the largest compression ratio of the sine ROM. This is expected from theory [29][25], which details how the algorithm offers even better compression than the Sunderland-method on which it is based. In [25], a 15% reduction compared to Sunderland is theorized, however this has not been the case in this thesis. It should be noted that the papers all utilized a 12 bit amplitude and phase length, and the discrepancy therefore stems from the different configuration used in this thesis, as this directly impacts the size and ratios between the coarse and fine ROMs in Nicholas-based DDS-configurations.

Phase noise analyses did not yield any clear conclusion with regards to which configuration had the optimal phase noise performance. All configurations, when corrected for target distance, performed better than the -80dBc requirement. However, spurious performance does not correspond with theorized magnitudes, meaning noise from other sources than phase truncation and phase length was dominating. As expected from theory, the performance degraded with lower amplitude bit length, with especially 4 bit amplitude showing the clearest signs of performance degradation, with multiple spurs.

The DAC used on RIMFAX and in this thesis has 14 output bits. In order to utilize the full range of the DAC, it is advisable to select a DDS configuration using 14 amplitude bits, and Nicholas compression on the sine ROM. The limited resolution bandwidth of the frequency spectra from the spectrum analyzer may have influenced the readings and new measurements with higher resolution (e.g. 200 kHz), is recommended before concluding.

7.3 Future work

The work in this thesis has demonstrated several working variations of a FPGA based DDS. However, further improvements can be done to achieve even higher quality output signals than achieved in this thesis. Some alternatives will now be presented.

If the clock frequency could be increased, this would allow for a smaller FTW for the same output frequency, as implied by equation 2.1. From test results conducted in this thesis, doubling the FTW leads to a degradation of -6 dB in phase noise. If the clock frequency could have been doubled, this would allow for a similar change in phase noise. Increasing the clock frequency will however impact power consumption and for applications with a limited power budget this may be an issue. Increased clock frequency
will also move the aliases seen in the test at $f_{clk} - f_{out}$ further away from the output frequency and is thereby an improvement, in addition to also moving aliases of harmonics folding into the Nyquist zone, further away from the fundamental frequency. Measures should also be taken regarding the output low pass filter, which with a steeper roll-off curve (i.e. using a filter of higher order), will attenuate spurs outside the Nyquist-zone better than what is the current case. The filter used in this thesis is not representative of what RIMFAX will use, and this is therefore an observation more aimed at general applications of these DDS-configurations rather than being RIMFAX-specific. A band pass filter could also be useful, to filter out low frequency noise in lower frequencies than the working range of the DDS.

New measurements on the DDS configurations in this thesis is also an item for future work, more specifically with higher resolution bandwidth of the frequency spectra as described in the Discussion section in chapter 6.

The results should also be put through a comparison test with a commercial DDS ASIC, to further explore the performance characteristics of the system. The Analog Devices AD9914 was considered for such a test in this thesis, but due to connectivity issues this was not pursued further.

Spurious noise appeared at predictable locations, except for the Quarter Sine DDS at 18 and 32 bit FTW. In [24] the author describes how the largest noise sources are usually related to the analog part of the system. New tests could therefore be designed to closer characterize the DAC. It would also be of interest to see if the characteristics of the output changes with another DAC designed for similar applications.

To further reduce the footprint, the CORDIC-algorithm could be explored and implemented. It was not considered for this thesis due to being the most disparate DDS-type compared to the other versions considered. Instead of relying purely on a look-up table with sine amplitude values, the CORDIC-algorithm utilizes multiplication and other techniques to approximate a sine wave. According to [24] it offers up to 128 times reduction in resource requirements compared to a regular uncompressed DDS, but at the expense of a more complex circuit. Another interesting method is the phase difference algorithm introduced in chapter 3. This is based on the Nicholas Compression method, but requires an extra adder. In return, the designer can utilize the truncated phase bits as a part of the design instead of discarding them, as well as compressing the sine ROM even further than the regular Nicholas Compression.
Bibliography


Appendices
Appendix A

Permissions
Spildrejorde, Kim Ostenfor

Subject: FW: PLL book figures in thesis

From: Banerjee, Dean
Sent: 23. november 2016 18:59
To: Spildrejorde, Kim Ostenfor
Subject: RE: PLL book figures in thesis

Kim,

I encourage you to look at my 5th edition PLL book at:

As I am the author and publisher of this book, I say it is fine to use a few figures provided you state so in the references.

Regards,
Dean
Appendix B

Matlab Code

B.1 Script Generating Quarter Sine ROM

```matlab
function lut_quartersine ( phase_bits , bits_out , filename )

LUT_len = 2^phase_bits ;
LUT_lim = pi/2;
quarter_sine = linspace (0 , pi/2 , LUT_len);
for i = 1:LUT_len
    sine_ROM(i) = sin (quarter_sine(i));
end;

output_length = bits_out ; % bit output
sine_ROM_int = int32 ( sine_ROM*(2^output_length-1) );

% ERROR PLOTS
figure(3)
error_LUT = sine_ROM*(2^output_length-1) - double ( sine_ROM_int );
title3_str = sprintf ('Error plot of coarse ROM in no. of LSB. 1 LSB = %d', LSB_LUT )
plot ( error_LUT )

% WRITING LUT VHDL-files
num_addresses = LUT_len ;
addr_length = phase_bits ;
out_length = output_length ; % output_length bits for both sine and cosine. Concatenated leads to twice the vector length

entity_name = 'sine_LUT';
import_libraries = 'library ieee;
nuse ieee.std_logic_1164.all;
';
entity_description = 'entity %s is
';
entity_portlist = '	port ( 
		 address : in std_logic_vector (%d downto 0);
		 data : out std_logic_vector(%d downto 0)
);
';
entity_end = 'end entity %s;
';
```

119
arch = 'architecture behavioral of %s is
    type mem is array (0 to
    2**%d -1) of std_logic_vector (%d downto 0);
';
declare_lut = 'constant DDS_LUT : mem := (\n';
process = '    process (address)\n    begin\n    begin
    case_declaration ='	    case address is
    case_end = '	    end case ;
    process_end = '	    end process ;
    arch_end = 'end architecture ;

fileID = fopen ( filename ,'w');
% WRITE ENTITY OF COARSE LUT
fprintf ( fileID , import_libraries );
fprintf ( fileID , entity_description , entity_name );
fprintf ( fileID , entity_portlist , addr_length -1 , out_length -1) ;
fprintf ( fileID , entity_end , entity_name );
% Write architecture of LUT
fprintf ( fileID , arch , entity_name , addr_length , out_length -1) ;
fprintf ( fileID , declare_lut );

for i =1: num_addresses
    bin_value = dec2bin ( sine_ROM_int (i), out_length );
    if i < num_addresses
        line = '		%d => "%s",\t-- Sine value : %d 
';
    elseif i == num_addresses
        line = '		%d => "%s"\t-- Sine value : %d 

);''
    end
    fprintf ( fileID ,line , i -1 , bin_value , double ( sine_ROM_int (i)));
end
fprintf ( fileID , begin );
% Writing process to VHD-file for coarse LUT
fprintf ( fileID , process );
fprintf ( fileID , case_declaration );
for i=1: num_addresses + 1
    i_binary = dec2bin (i-1, addr_length );
    if i < num_addresses + 1
        line = '		\when "%s",\twhen "%s",\twhen others => DDS_LUT(%d);
';
        fprintf ( fileID , line , i_binary , i -1);''
    end
    if i == num_addresses +1
        others_val = dec2bin (0, out_length );
        line = '	\when others => DDS_LUT(%d);
';
        fprintf ( fileID , line , others_val );''
end
fprintf ( fileID , case_end );
fprintf ( fileID , arch_end );
fclose ( fileID );
B.2 Script Generating Sunderland Compression ROMs

```matlab
function lut_sunderland(part_phase_bits, bits_out, coarse_name, fine_name)
% Function for generating sine ROM VHDL file for Sunderland Compression DDS.
% Author: Kim Spildrejorde
abc_len = part_phase_bits;

A_lim = pi/2;
B_lim = (pi/2) * (2^(-abc_len));
C_lim = (pi/2) * (2^(2*abc_len));

A_corr = (1/(2^abc_len)) * A_lim;
B_corr = (1/(2^abc_len)) * B_lim;
C_corr = (1/(2^abc_len)) * C_lim;

A_max = A_lim - A_corr;
B_max = B_lim - B_corr;
C_max = C_lim - C_corr;

A = linspace(0, A_max, 2^abc_len);
B = linspace(0, B_max, 2^abc_len);
C = linspace(0, C_max, 2^abc_len);

coarse_len = abc_len*2;
fine_len = abc_len*2;

ROM1_num_addr = 2^coarse_len;
ROM2_num_addr = 2^fine_len;

output_length = bits_out;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create coarse ROM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a_count = 1;
b_count = 1;
i = 0;
k = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create Coarse LUT, consisting of the term sin(A+B)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:2^coarse_len
    coarse_ROM(i) = (sin(A(a_count) + B(b_count))) * (1-2^-output_length);
    b_count = b_count + 1;
    if b_count > 2^abc_len
        b_count = 1;
        a_count = a_count + 1;
    end;
end;

course_ROM_int = int32(coarse_ROM*(2^output_length-1));
length(coarse_ROM_int)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create Fine LUT, consisting of the term cos(A)*sin(C)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B_avg = mean(B);
a_count = 1;
```

c_count = 1;
for i = 1:2^ fine_len
    fine_ROM(i) = (cos(A(a_count)+B_avg)*sin(C(c_count)))... *\(1-2^{-\text{output_length}}\);
    c_count = c_count + 1;
if c_count > 2^ abc_len
    c_count = 1;
    a_count = a_count +1;
end;
end;

% Value of 1 LSB
LSB_LUT_coarse = 1/(2^ output_length);
LSB_LUT_fine = 1/(2^ output_length);

%ERROR PLOTS
figure(3)
length ( coarse_ROM_int )
error_coarse = coarse_ROM *(2^ output_length -1) - double ( coarse_ROM_int );
title3_str = sprintf ('Error plot of coarse ROM in no. of LSB. 1 LSB = \%d', LSB_LUT_coarse )
title ( title3_str )
plot ( error_coarse )
figure(4)
length ( fine_ROM_int )
error_fine = fine_ROM *(2^8169 -1) - double ( fine_ROM_int )
title4_str = sprintf ('Error plot of fine ROM in no. of LSB. 1 LSB = \%d', LSB_LUT_fine )
title ( title4_str )
plot ( error_fine )

%WRITING LUT VHDL-files
num_addresses_coarse = 2^ coarse_len;
num_addresses_fine = 2^ fine_len;

disp_str = 'Finds bitlength needed for \text{sin}_y output by checking largest number in LUT
out_length_fine = length(dec2bin(max(double(fine_ROM_int))))
entity_name_coarse = 'coarse_LUT';
entity_name_fine = 'fine_LUT';
importibraries = 'library ieee; use ieee.std_logic_1164.all;
entity_description = 'entity %s is \n';
entity_portlist = ' port( \n\taddress : in std_logic_vector(%d downto 0);
\tdata : out std_logic_vector(%d downto 0));\nend
';
entity_end = 'end entity %s;\n';
arch = 'architecture behavioral of %s is \n\ttype mem is array (0 to 2**%d-1) of std_logic_vector(%d downto 0);\n\tconstant DDS_LUT : mem := ();\n\tbegin
\tprocess(address)\n\tbegin
\n';
declare_lut = 'begin \n\n
...
APPENDIX B. MATLAB CODE

124
125 case_declaration = '\t\tcase address is\n';
126 case_end = '\t\tend case;\n';
127 process_end = '\t\tend process;\n';
128 arch_end = 'end architecture;\n';
129
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
130
131
132 fileID = fopen(coarse_name,'w');
133
134
% WRITE ENTITY OF COARSE LUT
135 fprintf(fileID, import_libraries);
136 fprintf(fileID, entity_description, entity_name_coarse);
137 fprintf(fileID, entity_portlist, addr_length_ab-1, out_length_coarse-1);
138 fprintf(fileID, entity_end, entity_name_coarse);
139
% Write architecture of sin(X) & cos(X) LUT
140 fprintf(fileID, arch, entity_name_coarse, addr_length_ab, out_length_coarse-1);
141 fprintf(fileID, declare_lut);
142
143 for i=1: num_addresses_coarse
144 bin_value = dec2bin(coarse_ROM_int(i), out_length_coarse);
145 if i < num_addresses_coarse
146 line = '\t\t\twhen "%s" => data <= DDS_LUT(%d);\n';
147 fprintf(fileID, line, i_binary, i -1) ;
148 end ;
149 if i == num_addresses_coarse +1
150 others_val = dec2bin(0, out_length_coarse);
151 line = '\t\t\twhen others => data <= "%s";\n';
152 fprintf(fileID, line, others_val);
153 end
154 fprintf(fileID, case_end);
155 fprintf(fileID, process_end);
156 fprintf(fileID, arch_end);
157 fclose(fileID);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WRITE FINE LUT
158 fileID2 = fopen(fine_name,'w');
159 fprintf(fileID2, import_libraries);
160 fprintf(fileID2, entity_description, entity_name_fine);
161 fprintf(fileID2, entity_portlist, addr_length_ac-1, out_length_fine-1);
162 fprintf(fileID2, declare_lut);
163
164 for i=1: num_addresses_coarse + 1
165 i_binary = dec2bin(i-1, addr_length_ab);
166 if i < num_addresses_coarse + 1
167 line = '\t\t\twhen "%s" => data <= DDS_LUT(%d);\n';
168 fprintf(fileID2, line, i_binary, i-1);
169 end;
170 if i == num_addresses_coarse+1
171 others_val = dec2bin(0, out_length_coarse);
172 line = '\t\t\twhen others => data <= "%s";\n';
173 fprintf(fileID2, line, others_val);
174 end
175 fprintf(fileID2, case_end);
176 fprintf(fileID2, process_end);
177 fprintf(fileID2, arch_end);
178 fclose(fileID2);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
184 out_length_fine = 1;
185 fprintf(fileID2, declare_lut);
186
for i = 1: num_addresses_fine
187   bin_value = dec2bin(double(fine_ROM_int(i)), out_length_fine);
188   if i < num_addresses_fine
189       line = '\t\t' \%d \t-- Sine value: \%d \n';
190   elseif i == num_addresses_fine
191       line = '\t\t' \%s\t-- Sine value: \%d \n \n';
192   end
193   fprintf(fileID2, line, i - 1, bin_value, double(fine_ROM_int(i)));
194 end
195
% Writing process to VHD-file for sin(y) LUT
196 fprintf(fileID2, process);
197
for i = 1: num_addresses_fine + 1
198   i_binary = dec2bin(i - 1, addr_length_ac);
199   if i < num_addresses_fine + 1
200       line = '\t\t' \when \%s \t\tdata <= DDS_LUT \(\%d\);\n';
201       fprintf(fileID2, line, i_binary, i - 1);
202     end;
203     if i == num_addresses_fine + 1
204       others_val = dec2bin(0, out_length_fine);
205       line = '\t\t\t\when others \t\tdata <= \%s;\n';
206       fprintf(fileID2, line, others_val);
207   end
208 end
209
fprintf(fileID2, case_end);
210 fprintf(fileID2, process_end);
211 fprintf(fileID2, arch_end);
212 fclose(fileID2);
B.3 Script Generating Nicholas Compression ROMs

function rom_nicholas(a,b,c, bits_out , coarse_name , fine_name )
% Function for generating Sine ROM samples for Nicholas Compression DDS
% Author: Kim Spildrejorde
% Date: 22/02/2016

ab_len = a+b;
ac_len = a+c;
ROM1_num_addr = 2^ab_len ;
ROM2_num_addr = 2^ac_len ;
output_length = bits_out ;
%M bit output

% Create ROM samples
% Create a, b, c
a_int = 1;
b_int = 1;
i = 0;
k = 1;

for a_int = 1: 2^a
    for b_int = 1 : 2^b
        coarse_ROM (i+1) = sin((pi/2)*(((a_int-1)*2^b)+(b_int-1))/(2^(a+b))...+
                      (1/(2^(a+b+c))));
    end;
end;

max_term = 0.5* max ( coarse_ROM )
min_term = 0.5* min ( coarse_ROM )

% Create fine ROM

% Create a, b, c
a_int = 1;
b_int = 1;
i = 0;
k = 1;

for a_int = 1: 2^a
    for b_int = 1 : 2^b
        coarse_ROM (i+1) = sin((pi/2)*(((a_int)*2^b)+(b_int-1)))/(2^(a+b))...+
                      (1/(2^(a+b+c))));
    end;
end;

max_term = 0.5*max ( sin (pi/2(a_int)*2^(b+c)+b_int *(2^c)+c_int) )

% Create fine ROM

max_term = 0.5*max (sin(pi/2(a_int*2^(b+c)+b_int *(2^c)+c_int))...+
                      (1/(2^(a+b+c))));

for a_int = 1: 2^a
    for b_int = 1 : 2^b
        coarse_ROM(i+1) = sin((pi/2)*(((a_int)*2^b)+b_int ))/(2^(a+b))...+
                      (1/(2^(a+b+c))));
        i = i+1;
    end;
end;

figure(9)
plot(coarse_ROM)

coarse_ROM_int = int32(coarse_ROM.*(output_length-1));

fine_ROM_int = int32(fine_rom.*(2^output_length-1));

LSB_LUT_coarse = 1/(ROM1_num_addr);

LSB_LUT_fine = 1/(ROM2_num_addr);

% ERROR PLOTS

% START OF WRITING ROM VHDL-files

% Finds bitlength needed for sin_y output by checking largest number in LUT

out_length_fine = length(dec2bin(max(double(fine_ROM_int))))

fileID = fopen(coarse_name,'w');
APPENDIX B. MATLAB CODE

123 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124 % WRITE ENTITY OF COARSE LUT
125 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
126 fprintf(fileID, import_libraries);
127 fprintf(fileID, entity_description, entity_name_coarse);
128 fprintf(fileID, entity_portlist, addr_length_ab-1, out_length_coarse -1);
129 fprintf(fileID, entity_end, entity_name_coarse);
130
131 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
132 % Write architecture of sin(X) & cos(X) LUT
133 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
134 fprintf(fileID, arch, entity_name_coarse, addr_length_ab, out_length_coarse-1);
135 fprintf(fileID, declare_lut);
136
137 for i=1: num_addresses_coarse
138    bin_value = dec2bin(coarse_ROM_int(i), out_length_coarse);
139    if i < num_addresses_coarse
140        if i == num_addresses_coarse
141            line = '\t	%d => "%s" \t-- Sine value: %d
';
142        else
143            line = '\t	%d => "%s" \t-- Sine value: %d
';
144        end
145    end
146    fprintf(fileID, line, i -1, bin_value, double(coarse_ROM_int(i)));
147 end
148 fprintf(fileID, begin);
149
150 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
151 % WRITING PROCESS TO VHD-FILE FOR COARSE LUT
152 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
153 fprintf(fileID, process);
154
155 fprintf(fileID, case_declaration);
156 for i=1: num_addresses_coarse + 1
157    i_binary = dec2bin(i-1, addr_length_ab);
158    if i < num_addresses_coarse + 1
159        if i == num_addresses_coarse + 1
160            others_val = dec2bin(0, out_length_coarse);
161            line = '\t\t\twhen others => data <= DDS_LUT(0);
162        else
163            line = '\t\t\twhen %s => data <= DDS_LUT(%d);
164        end
165    end
166    fprintf(fileID, line, i_binary, i -1);
167 end
168 fprintf(fileID, case_end);
169 fprintf(fileID, process_end);
170 fclose(fileID);
171
172 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
173 % WRITE FINE LUT
174 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
175 fileID2 = fopen(fine_name,'w');
176 % %WRITE ENTITY OF FINE LUT
177 fprintf(fileID2, import_libraries);
178 fprintf(fileID2, entity_description, entity_name_fine);
179 fprintf(fileID2, entity_portlist, addr_length_ac-1, out_length_fine -1);
180 fprintf(fileID2, entity_end, entity_name_fine);
181
182 for i=1: num_addresses_fine
183 ...
bin_value = dec2bin(double(fine_ROM_int(i)), out_length_fine);
if i < num_addresses_fine
    line = '\t\t%d => "%s",
-- Sine value: %d
';
else if i == num_addresses_fine
    line = '\t\t%d => "%s"
-- Sine value: %d
';
end
fprintf(fileID2, line, i-1, bin_value, double(fine_ROM_int(i)));
end
fprintf(fileID2, begin);

% Writing process to VHD-file for sin(y) LUT
fprintf(fileID2, process);

fprintf(fileID2, case_declaration);
for i=1:num_addresses_fine + 1
    i_binary = dec2bin(i-1, addr_length_ac);
    if i < num_addresses_fine + 1
        line = '\t\t\twhen "%s" => data <= DDS_LUT(%d);
    end;
    if i == num_addresses_fine + 1
        others_val = dec2bin(0, out_length_fine);
        line = '\t\t\twhen others => data <= "%s";
    end;
end
fprintf(fileID2, case_end);
fprintf(fileID2, process_end);
fprintf(fileID2, arch_end);
fclose(fileID2);
%% Function information
% a ,b ,c ,d: Input filenames
% end_freq : 1 = 30 kHz , 2 = 5 MHz
% testfreq : 1 = 18.75 MHz 2 = 24.5 MHz 3 = 30.0 MHz 4 = 37.5 MHz
% test_mode : 1 = Amplitude bit length test 2 = LUT address bit length test
% 3 = FTW bit length test
% figname: Name of file to which the phase noise plot is saved

% Author: Kim Spildrejorde
% Date: 22/09/2016
% Parts of function autogenerated by Matlab
% Adapted to include multiple input files, plotting and saving figure, and calculations of integrated phase noise

function noise_plot (a,b,c,d, end_freq , testfreq , test_mode , figname )

%% Initialize variables.
if test_mode == 1
% Only Amplitude bit length test has four different bit lengths
file_list = {a,b,c,d};
else
file_list = {a,b,c};
end
delimiter = ';';
startRow = 156;

if end_freq == 1
% Plot from 1kHz -30 kHz
endRow = 573;
elseif end_freq == 2
% Plot from 1kHz -5 MHz
endRow = 1203;
end

%% Format string for each line of text:
% column1: double (%f)
% column2: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%*s%*s%*s %
[^ 
 r ]';

%% Read file, extract data, perform calculations and plot figure
figure( 'Position', [100 , 100 , 1000 , 600]);
hold 'on'
for i =1:length( file_list )
filename = file_list{1,i};

%% Open the text file.:
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan( fileID , formatSpec , endRow-startRow+1 , 'Delimiter', delimiter , 'HeaderLines', startRow-1 , 'ReturnOnError', false);

%% Close the text file.
close(fileID);

%% Create output variable
data = [dataArray{1: end-1}];
pndata = data(:,2);
fdata = data(:,1);
for k = 1:(endRow-startRow+1)
    fdata(k);
    pnoise_lin_corr(k) = (10^(pndata(k)/20))*(4*(sin(pi*fdata(k)*
        *(1*10^-6)).^2));
    pdata_new(k) = 20*log10(pnoise_lin_corr(k));
end

% Calculating integrated phase noise for the desired frequency range
% Calculated by integrating (summing) the linear phase noise values
% and converting to decibels.
phnoise_int(i) = 20*log10(sum(pnoise_lin_corr));
end

%% Adjusting figure visuals
set(gca,'fontsize',12)
title_text = {'Phase Noise at f=18.75 MHz', 'Phase Noise at f=24.5 MHz'
    'Phase Noise at f=30.0 MHz', 'Phase Noise at f=37.5 MHz'};
title(title_text{1,testfreq})
xlabel('Frequency offset from carrier (Hz)')
ylabel('Magnitude (dBc/Hz)')
if test_mode == 1
    legend('4 bits', '8 bits', '14 bits', '18 bits')
elseif test_mode == 2
    legend('9 bits', '12 bits', '15 bits')
elseif test_mode == 3
    legend('18 bits', '32 bits', '48 bits')
end
grid 'on'
axis([1*10^3 3*10^4 -190 -110])
hold 'off'
savefig(figname)
saveas(gcf,figname,'epsc')

%% Display integrated phase noise values in Command Window
disp(phnoise_int)
clear all
end
B.5 Frequency Spectrum Plotting

```matlab
function freqspec(a,b,c,d, figname , figtitle )
%IMPORTFILE Import numeric data from a text file as a matrix.
% MHZ085 = IMPORTFILE(FILENAME) Reads data from text file FILENAME
for
% the default selection.
% MHZ085 = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from
% rows
% STARTROW through ENDROW of text file FILENAME.
% Example:
% MHZ085 = importfile('100MHZ_085.DAT', 27, 651);
%
% See also TEXTSCAN.
%% Initialize variables.
delimiter = ';';
if nargin<2
startRow = 27;
endRow = inf;
end
%% Format string for each line of text:
column1 : double (%f)
column2 : double (%f)
For more information, see the TEXTSCAN documentation.
file_list = {a,b,c,d};
formatSpec = '%f%f%*s %
[^ \ n \ r ]';
title_texts = {'f =18.75 MHz ','f =24.5 MHz ','f =30.0 MHz ',f =37.5 MHz '};
figure(' Position ', [100, 100, 1000, 650]);
for i =1:length ( file_list )
filename = file_list{1,i};
%% Open the text file
fileID = fopen ( filename ,'r');
%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate
% this
% code. If an error occurs for a different file, try regenerating the
code.
% from the Import Tool.
dataArray = textscan ( fileID , formatSpec , endRow{1}=startRow{1}+1 , 'Delimiter', delimiter, 'HeaderLines', startRow{1}-1, 'ReturnOnError', false);
for block=2:length(startRow)
rewind ( fileID );
dataArrayBlock = textscan ( fileID , formatSpec , endRow{block}=startRow{block}+1 , 'Delimiter', delimiter, 'HeaderLines', startRow{block}-1, 'ReturnOnError', false);
for col=1:length(dataArray)
dataArray{col} = [dataArray{col};dataArrayBlock{col}];
end
end
%% Close the text file.
fclose ( fileID );
%% Create output variable
data = [dataArray{:end-1}];
subplot(2,2,i)
plot (data(:,1), data(:,2), 'LineWidth', 2)
xlabel ('Frequency (Hz)')
ylabel ('Magnitude (dBm)')
axis([0 100*10^6 -70 0])
title ( title_texts{1,i})
```

APPENDIX B. MATLAB CODE 131
58 set(gca,'fontsize',12)
59 grid 'on'
60 end
61 p=mtext(figtitle,...
62 'fontsize',14,...
63 'soff',-.0,'yoff',.05);
64 savefig(figname)
65 saveas(gcf,figname,'epsc')
66 end
Appendix C

VHDL Code

C.1 Top Level

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library UNISIM;
use UNISIM.vcomponents.all;
entity top_level is
    generic (  
        FTW_LENGTH : integer := 48;
        LUT_ADDR_LENGTH : integer := 10;
        OUTPUT_LENGTH : integer := 14;
        DDS_OUT_LEN_INT : integer := 14  
    );
    port (  
        clk_in1_p : in std_logic;
        clk_in1_n : in std_logic;
        reset_n : in std_logic;
        ena : in std_logic;
        ftw_sw : in std_logic;
        amplitude_out : out std_logic_vector(OUTPUT_LENGTH-1 downto 0);
        dac_clk : out std_logic
    );
end entity;
architecture rtl of top_level is
component dds is
    port (  
        clk : in std_logic;
        reset : in std_logic;
        ena : in std_logic;
        ftw_sw : in std_logic;
        amplitude_out : out std_logic_vector(DDS_OUT_LEN_INT-1 downto 0);
    );
end component dds;
component clk_wiz_0 is
    port (  
        clk_in1_p : in std_logic;
        clk_in1_n : in std_logic;
    );
end component clk_wiz_0;
component clk_wiz_1 is
    port (  
        clk_in1_p : in std_logic;
        clk_in1_n : in std_logic;
    );
end component clk_wiz_1;
```

133
clk_out1 : out std_logic;
-- Status and control signals
reset : in std_logic);
end component clk_wiz_0;

component debounce is
PORT(
  clk : IN STD_LOGIC; --input clock
  button : IN STD_LOGIC; --input signal to be debounced
  result : OUT STD_LOGIC); --debounced signal
end component debounce;

signal clk_out : std_logic;

signal button_r : std_logic;
signal button_rr : std_logic;
signal ftw_switch : std_logic := '0';
signal debounce_counter : std_logic_vector (20 downto 0);
signal clk_test : std_logic;
signal odd_phase_ena : std_logic;
signal ftw_in : std_logic_vector (FTW_LENGTH-1 downto 0) :=
  (others => '0');
signal amplitude_out_int : std_logic_vector (DDS_OUT_LEN_INT-1 downto 0);

begin

dac_clk <= clk_out;
amplitude_out <= amplitude_out_int (DDS_OUT_LEN_INT-1 downto
  DDS_OUT_LEN_INT-OUTPUT_LENGTH);

clock_sys: clk_wiz_0
  port map(
    clk_in1_p => clk_in1_p,
    clk_in1_n => clk_in1_n,
    reset => reset_n,
    clk_out1 => clk_out
  );

dds_sys: dds
  port map(
    clk => clk_out,
    reset => reset_n,
    ena => ena,
    ftw_sw => ftw_sw,
    odd_phase_ena => odd_phase_ena,
    amplitude_out => amplitude_out_int
  );
end architecture;
C.2 Phase Accumulator

```vhdl
-- *************************************************
-- Phase Accumulator based on paper by H.T. Nicholas
-- & H. Samueli
-- Ref: An Analysis of the Output Spectrum of DDFS
-- in Presence of Phase-Accumulator Truncation (1987)
-- Date: 11.02.2016
-- Author: Kim Spildrejorde
-- Version 1.0
-- *************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity phase_accumulator is
  generic (FTW_LENGTH : integer := 48);
  port (clk : in std_logic;
         reset : in std_logic;
         ftw_in : in std_logic_vector (FTW_LENGTH-1 downto 0);
         ena : in std_logic;
         phase_out : out std_logic_vector (FTW_LENGTH-1 downto 0);
         switch : in std_logic);
end entity;

architecture beh of phase_accumulator is

-- Internal signals
signal accumulator_sum : std_logic_vector (FTW_LENGTH-1 downto 0);
signal ftw_int : std_logic_vector (FTW_LENGTH-1 downto 0);
signal odd_ftw : boolean := true;

begin
  ftw_int <= ftw_in;

  p_phaseacc: process (clk)
  begin
    if rising_edge (clk) then
      if reset = '1' or switch = '1' then
        accumulator_sum <= (others => '0');
        odd_ftw <= true;
      else
        if ena = '1'
          if odd_ftw = true then
            accumulator_sum <= std_logic_vector(unsigned(accumulator_sum) +
                                             unsigned(ftw_int) + 1);
          elsif odd_ftw = false then
            accumulator_sum <= std_logic_vector(unsigned(accumulator_sum) +
                                             unsigned(ftw_int));
          end if;
        else
          accumulator_sum <= std_logic_vector(unsigned(accumulator_sum) +
                                             unsigned(ftw_int));
        end if;
        phase_out <= accumulator_sum;
        odd_phase_enforcer <= not(odd_phase_enforcer);
      end if;
    end process;
  end if;
end beh;
```
end if;
end process;
end architecture;
C.3 DDS (Sunderland/Nicholas)

```vhdl
-- DDS top level file. Instantiates components consisting of phase accumulator, look-up tables and necessary logic
-- Compression algorithm supported: Sunderland Compression
-- and Nicholas Compression
-- Author: Kim Spildrejorde
-- Date: 28.07.2016
-- Version 2.0
-- *********************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library UNISIM;
use UNISIM.vcomponents.all;
use IEEE.std_logic_unsigned.all;

entity dds is
  generic(
    FTW_LENGTH : integer := 48; -- Varies depending on test scenario
    PHASE_LENGTH : integer := 15; -- Total phase length for LUT-addressing. The rest is truncated. Varies with test scenario
    AMPLITUDE_LENGTH : integer := 14; -- Bit length of final sine amplitude output. Varies with test scenario
    LUT_ADDRESS_LENGTH : integer := 10; -- Total LUT address length
    PARTIAL_ADDR_LENGTH : integer := 5; -- LUT address component length
    COARSE_LUT_OUT_LENGTH : integer := 13; -- Varies by implementation, this case for 14 bit output
    FINE_LUT_OUT_LENGTH : integer := 3 -- Varies by implementation, this case for 14 bit output
  );
  port(
    clk : in std_logic ; -- 100 MHz clock from Clock Manager IP
    reset : in std_logic; -- Active high reset, to comply with active high reset in clock IP, generated in Vivado
    odd_phase_ena : in std_logic; -- '1' ensures the FTW appears to always be odd number. '0' takes the FTW as is
    ftw_sw : in std_logic ; -- Switch signal to change FTW
    amplitude_out : out std_logic_vector (AMPLITUDE_LENGTH-1 downto 0) -- 14 bits max usable output bits due to DAC. LUT output can be truncated down if necessary
  );
end entity;

architecture rtl of dds is
  component phase_accumulator is
    port(
      clk : in std_logic;
      reset : in std_logic;
      ftw_in : in std_logic_vector (FTW_LENGTH-1 downto 0);
      ena : in std_logic;
      switch : in std_logic;
      phase_out : out std_logic_vector (FTW_LENGTH-1 downto 0)
    );
  end component phase_accumulator;

  -- Sunderland/Nicholas compression based look-up tables.
  -- Generics used for bit lengths for easy changeability in LUT-sizes
  component coarse_LUT
    port(address : in std_logic_vector (LUT_ADDRESS_LENGTH-1 downto 0));
  end component coarse_LUT;
```
```vhdl
-- Debounce vhdl file is supplied by Xilinx

component debounce
port (CLK : in STD_LOGIC;
Sig : in STD_LOGIC;
Deb_Sig : out STD_LOGIC);
end component;

signal phase_acc_out : std_logic_vector (FTW_LENGTH -1 downto 0);
signal phase_invert : std_logic := '0'; -- MSB -1 of phase accumulator
output 0 = quadrant 1 and 3, 1= quadrant 2 and 4
signal ampl_invert : std_logic := '0'; -- MSB of phase accumulator
output, used as sign bit of final amplitude output
signal ampl_invert_r : std_logic := '0'; -- Extra register for synch
signal ampl_invert_rr : std_logic := '0'; -- Extra register for synch
signal ampl_invert_rrr : std_logic := '0'; -- Extra register for synch
signal ampl_invert_rrrr : std_logic := '0'; -- Extra register for synch

signal phase_lut_addr : std_logic_vector (PHASE_LENGTH -1 downto 0) :=
(others => '0');
signal lut_address_A : std_logic_vector (PARTIAL_ADDR_LENGTH -1 downto 0);
signal lut_address_B : std_logic_vector (PARTIAL_ADDR_LENGTH -1 downto 0);
signal lut_address_C : std_logic_vector (PARTIAL_ADDR_LENGTH -1 downto 0);

signal coarse_LUT_addr : std_logic_vector (LUT_ADDRESS_LENGTH -1 downto 0);
signal fine_LUT_addr : std_logic_vector (LUT_ADDRESS_LENGTH -1 downto 0);

signal coarse_LUT_output : std_logic_vector (COARSE_LUT_OUT_LENGTH -1 downto 0);
signal fine_LUT_output : std_logic_vector (FINE_LUT_OUT_LENGTH -1 downto 0);

signal LUT_sum : std_logic_vector (COARSE_LUT_OUT_LENGTH -1 downto 0); 
signal switch : std_logic;
signal ftw_sel : std_logic_vector (1 downto 0) := "00";
signal ftw : std_logic_vector (FTW_LENGTH -1 downto 0);

signal amplitude_out_int : std_logic_vector (AMPLITUDE_LENGTH -1 downto 0);

-- Frequency Tuning Words
signal freq_tuning_word1 : std_logic_vector (FTW_LENGTH -1 downto 0) :=
"001100000000000000000000000000000000000000000000"; -- 18.75 MHz
signal freq_tuning_word2 : std_logic_vector (FTW_LENGTH -1 downto 0) :=
"00111101110000001000011111010110001100111001"; -- 24.5 MHz
signal freq_tuning_word3 : std_logic_vector (FTW_LENGTH -1 downto 0) :=
"010011001100110011001100110011001100110011001"; -- 30.0 MHz
signal freq_tuning_word4 : std_logic_vector (FTW_LENGTH -1 downto 0) :=
"011000000000000000000000000000000000000000000"; -- 37.5 MHz
```
signal ftw_int : std_logic_vector(FTW_LENGTH-1 downto 0);

begin

phase_acc : phase_accumulator
port map(
  clk => clk,
  reset => reset,
  ftw_in => ftw,
  ena => odd_phase_ena,
  phase_out => phase_acc_out,
);

LUT1 : coarse_LUT
port map(
  address => coarse_LUT_addr,
  data => coarse_LUT_output
);

LUT2 : fine_LUT
port map(
  address => fine_LUT_addr,
  data => fine_LUT_output
);

DEBOUNCER : debouncer
port map(
  clk => clk,
  sig => ftw_sw,
  Deb_Sig => switch
);

p_lut_addressing : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      phase_invert <= '0';
      ampl_invert <= '0';
      ampl_invert_r <= '0';
      phase_lut_addr <= ( others => '0');
    else
      phase_invert <= phase_acc_out (FTW_LENGTH-2);
      ampl_invert <= phase_acc_out (FTW_LENGTH-1);
      phase_lut_addr <= ( others => '0');
    end if;
  end if;

  lut_address_A <= ( others => '0'); -- Upper half of Coarse ROM address
  lut_address_B <= ( others => '0'); -- Lower half of Coarse ROM address
  lut_address_C <= ( others => '0');

  if phase_invert = '1' then
    lut_address_A <= not ( phase_lut_addr ( PHASE_LENGTH-1 downto
                                       PHASE_LENGTH-( PARTIAL_ADDR_LENGTH)));
    lut_address_B <= not ( phase_lut_addr ( PHASE_LENGTH-( PARTIAL_ADDR_LENGTH+1) downto
                                       PHASE_LENGTH - (2* PARTIAL_ADDR_LENGTH)))
    lut_address_C <= not ( phase_lut_addr ( PHASE_LENGTH- (2* PARTIAL_ADDR_LENGTH)));
  end if;
end process;
```vhdl
PARTIAL_ADDR_LENGTH+1) downto 0));
else
  lut_address_A <= phase_lut_addr(PHASE_LENGTH-1 downto PHASE_LENGTH
    -(PARTIAL_ADDR_LENGTH));
  lut_address_B <= phase_lut_addr(PHASE_LENGTH-(PARTIAL_ADDR_LENGTH
    +1) downto (PHASE_LENGTH - (2*PARTIAL_ADDR_LENGTH +1) downto 0));
end if;
coarse_LUT_addr <= lut_address_A & lut_address_B;
fine_LUT_addr <= lut_address_A & lut_address_C;
end if;
end if;
end if;
end process ;
p_lut_sum : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      LUT_sum <= ( others => '0');
      amplitude_out_int <= ( others => '0');
    else
      if ampl_invert_rr = '1' then
        LUT_sum <= not ( std_logic_vector ( unsigned ( coarse_LUT_output ) +
          unsigned ( fine_LUT_output )));
      else
        LUT_sum <= std_logic_vector ( unsigned ( coarse_LUT_output ) + unsigned
          ( fine_LUT_output ));
      end if;
      amplitude_out_int <= not ( ampl_invert_rrr ) & LUT_sum;
      amplitude_out <= amplitude_out_int;
    end if;
    end if;
    end process ;

-- Process for selecting/switching FTW. When the user presses the push
-- button, the switch signal goes high, incrementing FTW_sel by 1, and thereby
-- changing the FTW
p_sel : process (clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      ftw_sel <= ( others => '0');
    else
      case ftw_sel is
        when "00" =>
          ftw <= freq_tuning_word1;
        when "01" =>
          ftw <= freq_tuning_word2;
        when "10" =>
          ftw <= freq_tuning_word3;
        when "11" =>
          ftw <= freq_tuning_word4;
        when others =>
          ftw <= ( others => '0');
      end case ;
      if switch = '1' and ph_reset = '0' then
        ftw <= ( others => '0');
        ph_reset <= '1';
      elsif ph_reset <= '1' then
        ftw_sel <= '1' then
        ftw <= std_logic_vector(unsigned(ftw_sel + 1));
        ph_reset <= '0';
      end if;
    end if;
  end if;
end process ;
```
228     end if;
229     end if;
230     end if;
231     end process;
232     end architecture;
C.4 DDS (Quarter Sine Symmetry)

```vhdl
-- Implementation of DDS with single sine ROM 
-- utilizing quarter sine wave symmetry.
-- Instantiates phase accumulator, look-up table with 
-- sine values and debouncer for FTW-selection
-- Date: 30.07.2016
-- Author: Kim Spildrejorde
-- Version 1.0
-- *************************************************
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity dds is
  generic(
    FTW_LENGTH : integer := 48; -- FTW bit length, varies by test scenario
    PHASE_LENGTH : integer := 15; -- Topmost-2 phase bits, truncated from accumulator output
    AMPLITUDE_LENGTH : integer := 14; -- Final amplitude length. Dependent on test scenario
    LUT_OUT_LENGTH : integer := 13 -- Output from Sine LUT. Dependent on test scenario
  );
  port(
    clk : in std_logic;  -- 100MHz clock from Clock Manager IP
    reset : in std_logic;  -- Active high reset, to comply with active high reset in clock IP, generated in Vivado
    ena : in std_logic;  -- Enable signal to start frequency generation
    amplitude_out : out std_logic_vector(AMPLITUDE_LENGTH-1 downto 0) -- 14 bits max usable output bits due to DAC. LUT output can be truncated down if necessary
  );
end entity;

architecture beh of dds is
  component phase_accumulator is
    port(
      clk : in std_logic;
      reset : in std_logic;
      ftw_in : in std_logic_vector(FTW_LENGTH-1 downto 0);
      ena : in std_logic;
      phase_out : out std_logic_vector(FTW_LENGTH-1 downto 0)
    );
  end component phase_accumulator;

  component sine_LUT15bit is
    port(
      address : in std_logic_vector(PHASE_LENGTH-1 downto 0);
      data : out std_logic_vector(LUT_OUT_LENGTH-1 downto 0)
    );
  end component sine_LUT15bit;

  component debounce is
    port(
      CLK : in STD_LOGIC;
    );
  end component debounce;
```

-- Sunderland compression based look-up tables. Generics used for bit lengths for easy changeability in LUT-sizes

APPENDIX C. VHDL CODE

55 Sig : in STD_LOGIC;
56 Deb_Sig : out STD_LOGIC
57 );
58 end component;
59
60 DEBOUNCER : debouncer
61 port map(
62 clk => clk,
63 sig => ftw_sw,
64 Deb_Sig => switch
65 );
66
67 signal phase_acc_out : std_logic_vector(FTW_LENGTH-1 downto 0);
68 signal phase_topbits : std_logic_vector(1 downto 0); -- 2 MSB from phase_int
69 signal phase_invert : std_logic; -- LSB of phase_topbits
70 signal ampl_invert : std_logic; -- MSB of phase_topbits
71 signal ampl_invert_rr : std_logic; -- Extra register for synch
72 signal ampl_invert_r : std_logic; -- Extra register for synch
73 signal phase_lut_addr : std_logic_vector ( PHASE_LENGTH -1 downto 0) :=
74 ( others => '0');
75 signal sine_LUT_addr : std_logic_vector (PHASE_LENGTH-1 downto 0);
76 signal sine_LUT_output : std_logic_vector ( LUT_OUT_LENGTH -1 downto 0);
77 signal LUT_sum : std_logic_vector ( LUT_OUT_LENGTH -1 downto 0);
78 signal amplitude_out_int : std_logic_vector (AMPLITUDE_LENGTH-1 downto 0);
79 signal switch : std_logic;
80 signal ftw_sel : std_logic_vector(1 downto 0) := "00";
81 signal ftw : std_logic_vector ( FTW_LENGTH -1 downto 0);
82
83 begin
84
85 phase_acc : phase_accumulator
86 port map(
87 clk => clk,
88 reset => reset,
89 ftw_in => ftw,
90 ena => ena,
91 phase_out => phase_acc_out
92 );
93
94 LUT1: sine_LUT15bit
95 port map(
96 address => sine_LUT_addr,
97 data => sine_LUT_output
98 );
99 p_lut_addressing : process (clk)
100 begin
101  if rising_edge(clk) then
102 if reset = '1' then
103
104

APPENDIX C. VHDL CODE

```vhdl
116  phase_invert <= '0';
117  ampl_invert <= '0';
118  ampl_invert_r <= '0';
119  phase_lut_addr <= (others => '0');
120  else
121     phase_invert <= phase_acc_out(FTW_LENGTH-2);
122     ampl_invert <= phase_acc_out(FTW_LENGTH-1);
123     ampl_invert_r <= ampl_invert;
124     ampl_invert_rr <= ampl_invert_r;
125     ampl_invert_rrr <= ampl_invert_rr;
126     phase_lut_addr <= phase_acc_out(FTW_LENGTH-3 downto FTW_LENGTH-17);
127     if phase_invert = '1' then
128         sine_LUT_addr <= not(phase_lut_addr);
129     else
130         sine_LUT_addr <= phase_lut_addr;
131     end if;
132     end if;
133  end if;
134  end if;
135  end if;
136  end process;
137
138  p_lut_sum : process (clk)
139  begin
140     if rising_edge(clk) then
141         if reset = '1' then
142             LUT_sum <= (others => '0');
143             amplitude_out_int <= (others => '0');
144         else
145             if ampl_invert_rr = '1' then
146                 LUT_sum <= not(std_logic_vector(unsigned(sine_LUT_output))));
147             else
148                 LUT_sum <= std_logic_vector(unsigned(sine_LUT_output));
149             end if;
150             amplitude_out_int <= not(ampl_invert_rrr) & LUT_sum;
151             amplitude_out <= amplitude_out_int;
152         end if;
153     end if;
154  end process;
155
156  -- Process for selecting/switching FTW. When the user presses the push button,
157  -- the switch signal goes high, incrementing FTW_sel by 1, and thereby changing the FTW
158  p_sel : process(clk)
159  begin
160     if rising_edge(clk) then
161         if reset = '1' then
162             ftw_sel <= (others => '0');
163         else
164             case ftw_sel is
165                 when "00" =>
166                     ftw <= freq_tuning_word1;
167                 when "01" =>
168                     ftw <= freq_tuning_word2;
169                 when "10" =>
170                     ftw <= freq_tuning_word3;
171                 when "11" =>
172                     ftw <= freq_tuning_word4;
173                 when others =>
174                     ftw <= (others => '0');
175             end case;
176     end if;
177  end if;
178  end process;
179  -- In case of FTW-switch, the phase accumulator is first flushed to ensure start of count is 0
180  if switch = '1' and ph_reset = '0' then
```
ftw <= (others => '0');
ph_reset <= '1';
elseif ph_reset <= '1' then
  ftw Sel <= std_logic_vector (unsigned (ftw Sel + 1));
  ph_reset <= '0';
end if;
end if;
end if;
end process;
end architecture;
Appendix D

LogiCORE IP DDS Block Schematic
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftw_in[47:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dout[31:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_axis_config</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_axis_config_tdata[47:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_axis_config_tvalid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m_axis_phase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m_axis_phase_tdata[47:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m_axis_phase_tvalid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xlconcat_0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In0[14:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In1[0:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dout[15:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xlconcat_1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In0[14:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In1[0:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dout[15:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xlconstant_0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dout[0:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xlconstant_1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dout[0:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- DDS Compiler
- S_AXIS_CONFIG
- s_axis_config_tdata[47:0]
- s_axis_config_tvalid
- M_AXIS_PHASE
- m_axis_phase_tdata[47:0]
- m_axis_phase_tvalid
- ACLK
- XLSlice
- Din[47:0]
- Dout[31:0]
- ftw_in[47:0]
- DDS Compiler
- S_AXIS_PHASE
- s_axis_phase_tdata[15:0]
- s_axis_phase_tvalid
- M_AXIS_DATA
- m_axis_data_tdata[15:0]
- ACLK
- XLSlice
- Din[15:0]
- Dout[13:0]
- amplitude_out[13:0]
Appendix E

Test Result Graphs

E.1 Test 1: Varying Amplitude Bit Length

E.1.1 Quarter Sine DDS

Frequency Spectra 0 Hz - 100 MHz

- 4 bit amplitude is shown in figure E.1
- 8 bit amplitude is shown in figure E.2
- 14 bit amplitude is shown in figure E.3
- 18 bit amplitude is shown in figure E.4

Quarter Sine Symmetry DDS - Phase Noise 1 kHz - 30 kHz (Original measurement)

Phase noise measurements in the range of 1 kHz - 30 kHz for Quarter Sine Symmetry DDS, with varying amplitude bit length is shown in figure E.5.

Quarter Sine Symmetry DDS Phase Noise 1 kHz - 30 kHz (Final IF signal)

Phase noise measurements in the range of 1 kHz - 30 kHz for Quarter Sine Symmetry DDS, with varying amplitude bit length is shown in figure E.6. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to draw the actual image.

E.1.2 LogiCORE IP DDS

Frequency Spectra 0 Hz - 100 MHz

- 4 bit amplitude is shown in figure E.7
- 8 bit amplitude is shown in figure E.8
- 14 bit amplitude is shown in figure E.9
Figure E.1: Frequency spectra for Quarter Sine DDS with 4 bit amplitude output

- 18 bit amplitude is shown in figure E.10

LogiCORE IP DDS - Phase Noise 1 kHz - 30 kHz (Original measurement)
Phase noise measurements in the range of 1 kHz - 30 kHz for LogiCORE IP DDS DDS, with varying amplitude bit length is shown in figure E.11.

LogiCORE IP DDS - Phase Noise 1 kHz - 30 kHz (Final IF signal)
Phase noise measurements in the range of 1 kHz - 30 kHz for LogiCORE IP DDS, with varying amplitude bit length is shown in figure E.12. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to draw the actual image.

E.1.3 Sunderland Compression DDS
Frequency Spectra 0 Hz - 100 MHz
Figures in this section present frequency spectra of the Sunderland Compression DDS, for varying amplitude bit lengths.

- 4 bit amplitude is shown in figure E.13
Figure E.2: Frequency spectra for Quarter Sine DDS with 8 bit amplitude output

- 8 bit amplitude is shown in figure E.14
- 14 bit amplitude is shown in figure E.15
- 18 bit amplitude is shown in figure E.16

**Phase Noise 1 kHz - 30 kHz (Original measurement)**

Phase noise measurements in the range of 1 kHz - 30 kHz for Sunderland Compression, with varying amplitude bit length is shown in figure E.17. These are the original measurements from the spectrum analyzer.

**Phase Noise 1 kHz - 30 kHz (Final IF signal)**

Phase noise measurements in the range of 1 kHz - 30 kHz for Sunderland Compression, with varying amplitude bit length is shown in figure E.18. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to draw the actual image.
Figure E.3: Frequency spectra for Quarter Sine DDS with 14 bit amplitude output

### E.1.4 Nicholas Compression DDS

#### Frequency Spectra 0 Hz - 100 MHz

Figures in this section present frequency spectra of the Nicholas Compression DDS, for varying amplitude bit lengths.

- 4 bit amplitude is shown in figure E.19
- 8 bit amplitude is shown in figure E.20
- 14 bit amplitude is shown in figure E.21
- 18 bit amplitude is shown in figure E.22

#### Phase Noise 1 kHz - 30 kHz (Original measurement)

Phase noise measurements in the range of 1 kHz - 30 kHz for Nicholas Compression DDS, with varying amplitude bit length is shown in figure E.23. These are the original measurements from the spectrum analyzer.

#### Phase Noise 1 kHz - 30 kHz (Final IF signal)

Phase noise measurements in the range of 1 kHz - 30 kHz for Nicholas Compression DDS, with varying amplitude bit length is shown in figure E.23. These are the original measurements from the spectrum analyzer.
Figure E.4: Frequency spectra for Quarter Sine DDS with 18 bit amplitude output

E.24. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to draw the actual image.
Figure E.5: Phase noise measurements of Quarter Sine DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz.
Figure E.6: Phase noise measurements of Quarter Sine DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing.
Figure E.7: Frequency spectra for LogiCORE IP DDS, 4 bit amplitude output

### E.2 Test 2: Varying LUT Address Bit Length

#### E.2.1 Quarter Sine DDS

**Frequency Spectra 0 Hz - 100 MHz**
- 9 bit phase is shown in figure E.25
- 12 bit phase is shown in figure E.26
- 15 bit phase is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.3.

**Quarter Sine Symmetry DDS - Phase Noise 1 kHz - 30 kHz (Original measurement)**

Phase noise measurements in the range of 1 kHz - 30 kHz for Quarter Sine Symmetry DDS, with varying phase bit length are shown in figure E.27.

**Quarter Sine Symmetry DDS Phase Noise 1 kHz - 30 kHz (Final IF signal)**

Phase noise measurements in the range of 1 kHz - 30 kHz for Quarter Sine Symmetry DDS, with varying phase bit length used for addressing the sine
Figure E.8: Frequency spectra for LogiCORE IP DDS, 8 bit amplitude output

ROM, is shown in figure E.28. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to draw the actual image.

E.2.2 LogiCORE IP DDS

Frequency Spectra 0 Hz - 100 MHz
- 9 bit phase is shown in figure E.29
- 12 bit phase is shown in figure E.30
- 15 bit phase is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.9.

LogiCORE IP DDS - Phase Noise 1 kHz - 30 kHz (Original measurement)

Phase noise measurements in the range of 1 kHz - 30 kHz for LogiCORE IP DDS DDS, with varying phase bit length for sine ROM addressing are shown in figure E.31.
Figure E.9: Frequency spectra for LogiCORE IP DDS, 14 bit amplitude output

LogiCORE IP DDS - Phase Noise 1 kHz - 30 kHz (Final IF signal)

Phase noise measurements in the range of 1 kHz - 30 kHz for LogiCORE IP DDS, with varying phase bit length are shown in figure E.32. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances.

E.2.3 Sunderland Compression DDS

Frequency Spectra 0 Hz - 100 MHz

Figures in this section present frequency spectra of the Sunderland Compression DDS, for varying phase bit lengths used for sine ROM addressing.

- 9 bit phase is shown in figure E.33
- 12 bit phase is shown in figure E.34
- 15 bit phase is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.15.
Phase Noise 1 kHz - 30 kHz (Original measurement)

Phase noise measurements in the range of 1 kHz - 30 kHz for Sunderland Compression, with varying phase bit length, are shown in figure E.35. These are the original measurements from the spectrum analyzer.

Phase Noise 1 kHz - 30 kHz (Final IF signal)

Phase noise measurements in the range of 1 kHz - 30 kHz for Sunderland Compression, with varying phase bit length for sine ROM addressing are shown in figure E.36. This are the measurements after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to generate the actual radar image.

E.2.4 Nicholas Compression DDS

Frequency Spectra 0 Hz - 100 MHz

Figures in this section present frequency spectra of the Nicholas Compression DDS, for varying phase bit lengths used for sine ROM addressing.

- 9 bit phase is shown in figure E.37
- 12 bit phase is shown in figure E.38
Figure E.11: Phase noise measurements of LogiCORE IP DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz.
Figure E.12: Phase noise measurements of LogiCORE IP DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing.
FIGURE E.13: Frequency spectra for Sunderland Compression DDS, 4 bit amplitude output

- 15 bit phase is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.21.

Phase Noise 1 kHz - 30 kHz (Original measurement)

Phase noise measurements in the range of 1 kHz - 30 kHz for Nicholas Compression DDS, with varying phase bit lengths are shown in figure E.39. These are the original measurements from the spectrum analyzer.

Phase Noise 1 kHz - 30 kHz (Final IF signal)

Phase noise measurements in the range of 1 kHz - 30 kHz for Nicholas Compression DDS, with varying phase bit lengths are shown in figure E.40. This is the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to draw the actual image.
APPENDIX E. TEST RESULT GRAPHS

Figure E.14: Frequency spectra for Sunderland Compression DDS, 8 bit amplitude output

E.3 Test 3: Varying FTW Bit Length

E.3.1 Quarter Sine DDS

Frequency Spectra 0 Hz - 100 MHz
- 9 bit phase is shown in figure E.41
- 12 bit phase is shown in figure E.42
- 15 bit phase is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.3.

Quarter Sine Symmetry DDS - Phase Noise 1 kHz - 30 kHz (Original measurement)
Phase noise measurements in the range of 1 kHz - 30 kHz for Quarter Sine Symmetry DDS, with varying FTW bit length are shown in figure E.43.

Quarter Sine Symmetry DDS Phase Noise 1 kHz - 30 kHz (Final IF signal)
Phase noise measurements in the range of 1 kHz - 30 kHz for Quarter Sine Symmetry DDS, with varying FTW bit lengths, are shown in figure E.44.
Sunderland compression, 14 bit amplitude

<table>
<thead>
<tr>
<th>Frequency (Hz) × 10^7</th>
<th>Magnitude (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-80</td>
</tr>
<tr>
<td>5</td>
<td>-60</td>
</tr>
<tr>
<td>10</td>
<td>-40</td>
</tr>
<tr>
<td>15</td>
<td>-20</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

f=18.75 MHz

<table>
<thead>
<tr>
<th>Frequency (Hz) × 10^7</th>
<th>Magnitude (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-80</td>
</tr>
<tr>
<td>5</td>
<td>-60</td>
</tr>
<tr>
<td>10</td>
<td>-40</td>
</tr>
<tr>
<td>15</td>
<td>-20</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

f=24.5 MHz

<table>
<thead>
<tr>
<th>Frequency (Hz) × 10^7</th>
<th>Magnitude (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-80</td>
</tr>
<tr>
<td>5</td>
<td>-60</td>
</tr>
<tr>
<td>10</td>
<td>-40</td>
</tr>
<tr>
<td>15</td>
<td>-20</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

f=30.0 MHz

<table>
<thead>
<tr>
<th>Frequency (Hz) × 10^7</th>
<th>Magnitude (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-80</td>
</tr>
<tr>
<td>5</td>
<td>-60</td>
</tr>
<tr>
<td>10</td>
<td>-40</td>
</tr>
<tr>
<td>15</td>
<td>-20</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

f=37.5 MHz

Figure E.15: Frequency spectra for Sunderland Compression DDS, 14 bit amplitude output

These are the measurement after multiplying by the correction factor for correlating phase noise at short target distances.

### E.3.2 LogiCORE IP DDS

**Frequency Spectra 0 Hz - 100 MHz**

- 18 bit FTW is shown in figure E.45
- 32 bit FTW is shown in figure E.46
- 48 bit FTW is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.9.

**LogiCORE IP DDS - Phase Noise 1 kHz - 30 kHz (Original measurement)**

Phase noise measurements in the range of 1 kHz - 30 kHz for LogiCORE IP DDS DDS, with varying FTW bit length are shown in figure E.47.

**LogiCORE IP DDS - Phase Noise 1 kHz - 30 kHz (Final IF signal)**

Phase noise measurements in the range of 1 kHz - 30 kHz for LogiCORE IP DDS, with varying FTW bit length are shown in figure E.48. This is
Figure E.16: Frequency spectra for Sunderland Compression DDS, 18 bit amplitude output

the measurement after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to generate the actual radar image.

### E.3.3 Sunderland Compression DDS

#### Frequency Spectra 0 Hz - 100 MHz

Figures in this section present frequency spectra of the Sunderland Compression DDS, for varying FTW bit lengths.

- 18 bit FTW is shown in figure E.49
- 12 bit FTW is shown in figure E.50
- 48 bit FTW phase is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.15.

#### Phase Noise 1 kHz - 30 kHz (Original measurement)

Phase noise measurements in the range of 1 kHz - 30 kHz for Sunderland Compression, with varying phase bit length, are shown in figure E.51. These are the original measurements from the spectrum analyzer.
Figure E.17: Phase noise measurements of Sunderland Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz.
Figure E.18: Phase noise measurements of Sunderland Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing.
Phase Noise 1 kHz - 30 kHz (Final IF signal)

Phase noise measurements in the range of 1 kHz - 30 kHz for Sunderland Compression, with varying phase bit length for sine ROM addressing are shown in figure E.52. This are the measurements after multiplying by the correction factor for correlating phase noise at short target distances. This is therefore the phase noise of the IF signal, which is used to generate the actual radar image.

E.3.4 Nicholas Compression DDS

Frequency Spectra 0 Hz - 100 MHz

Figures in this section present frequency spectra of the Nicholas Compression DDS, for varying FTW bit lengths.

- 18 bit FTW is shown in figure E.53
- 32 bit FTW is shown in figure E.54
- 48 bit FTW is the exact configuration as in the Amplitude bit test with 14 amplitude bits, and the reader is therefore referred to section E.1 for the relevant plots, which can be found in figure E.21.
Figure E.20: Frequency spectra for Nicholas Compression DDS, 8 bit amplitude output

**Phase Noise 1 kHz - 30 kHz (Original measurement)**

Phase noise measurements in the range of 1 kHz - 30 kHz for Nicholas Compression DDS, with varying FTW bit length are shown in figure E.55. These are the original measurements from the spectrum analyzer.

**Phase Noise 1 kHz - 30 kHz (Final IF signal)**

Phase noise measurements in the range of 1 kHz - 30 kHz for Nicholas Compression DDS, with varying FTW bit length, are shown in figure E.56. These are the measurements after multiplying by the correction factor for correlating phase noise at short target distances.
Figure E.21: Frequency spectra for Nicholas Compression DDS, 14 bit amplitude output
Figure E.22: Frequency spectra for Nicholas Compression DDS, 18 bit amplitude output
Figure E.23: Phase noise measurements of Nicholas Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz.
Figure E.24: Phase noise measurements of Nicholas Compression DDS with amplitude bit lengths of 4, 8, 14 and 18 bits. Tested for frequencies 18.75 MHz, 24.5 MHz, 30.0 MHz and 37.5 MHz. Here multiplied with correction factor to emulate phase noise for IF signal used for further radar processing.
Figure E.25: Quarter Sine Symmetry DDS with 9 phase bits for sine ROM addressing.
Figure E.26: Quarter Sine Symmetry DDS with 12 phase bits for sine ROM addressing.
Figure E.27: Quarter Sine Symmetry DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for Quarter Sine Symmetry DDS with 9, 12 and 15 phase bits used for sine ROM addressing.
Figure E.28: Quarter Sine Symmetry DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz.
Figure E.29: Frequency spectra for LogiCORE IP DDS with 9 phase bits used for sine ROM addressing.
Figure E.30: Frequency spectra for LogiCORE IP DDS with 12 phase bits used for sine ROM addressing.
Figure E.31: LogiCORE IP DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 9, 12 and 15 phase bits used for sine ROM addressing.
Figure E.32: LogiCORE IP DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz.
Figure E.33: Frequency spectra for Sunderland Compression DDS with 9 phase bits for sine ROM addressing.
Figure E.34: Frequency spectra for Sunderland Compression DDS with 12 phase bits for sine ROM addressing.
Figure E.35: Sunderland Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 9, 12 and 15 phase bits used for sine ROM addressing.
Figure E.36: Sunderland Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz.
Figure E.37: Frequency spectra for Nicholas Compression DDS with 9 phase bits for sine ROM addressing.
Figure E.38: Frequency spectra for Nicholas Compression DDS with 12 phase bits for sine ROM addressing.
Figure E.39: Nicholas Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 9, 12 and 15 phase bits used for sine ROM addressing.
Figure E.40: Nicholas Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 9, 12 and 15 phase bits for sine ROM addressing. Frequency range 1 kHz - 30 kHz.
Figure E.41: Quarter Sine Symmetry DDS with 18 bit FTW.

Figure E.42: Quarter Sine Symmetry DDS with 32 bit FTW.
Figure E.43: Quarter Sine Symmetry DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for Quarter Sine Symmetry DDS with 18, 32 and 48 bit FTW.
Figure E.44: Quarter Sine Symmetry DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit FTW. Frequency range 1 kHz - 30 kHz.
APPENDIX E. TEST RESULT GRAPHS

Figure E.45: Frequency spectra for LogiCORE IP DDS with 18 bit FTW.

Figure E.46: Frequency spectra for LogiCORE IP DDS with 32 bit FTW.
Figure E.47: LogiCORE IP DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 18, 32 and 48 bit FTW.
Figure E.48: LogiCORE IP DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit FTW. Frequency range 1 kHz - 30 kHz.
Figure E.49: Frequency spectra for Sunderland Compression DDS with 9 phase bits for sine ROM addressing.
Figure E.50: Frequency spectra for Sunderland Compression DDS with 12 phase bits for sine ROM addressing.
Figure E.51: Sunderland Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for LogiCORE IP DDS with 18, 32 and 48 bit FTW
Figure E.52: Sunderland Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit FTW. Frequency range 1 kHz - 30 kHz.
Figure E.53: Frequency spectra for Nicholas Compression DDS with 18 bit FTW.
Figure E.54: Frequency spectra for Nicholas Compression DDS with 32 bit FTW
Figure E.55: Nicholas Compression DDS: Original phase noise measurement in the frequency range 1 kHz - 30 kHz for Nicholas Compression DDS with 18, 32 and 48 bit FTW.
Figure E.56: Nicholas Compression DDS: Phase noise for the IF signal after factoring for correlating phase noise in transmitted and received signal due to short target distance, for test with 18, 32 and 48 bit Frequency Tuning Word. Frequency range 1 kHz - 30 kHz.