EXPERIMENTAL RESULTS ON ALL-DIGITAL IMPLEMENTATION OF THE PHASE LOCKED-LOOP FOR SOFTWARE-DEFINED RADIO

Alexandre Marsolais (Ultra Electronics TCS, Montreal, Canada;
alexandre.marsolais @ultra-tcs.com);
Doan Nguyen Vo (doan-nguyen.vo @ultra-tcs.com)

ABSTRACT

Flexibility in terms of frequency tracking and generation is paramount to any SDR design. This paper presents the implementation and simulation of a highly flexible Digital PLL (DPLL). Experimental results of the proposed technique are presented. The integrated phase noise of the generated clocks is better than -74 dBc in a 10Hz-5MHz bandwidth. The spurious free dynamic range (SFDR), up to 100MHz, is at least 72 dBc and over a 200 kHz bandwidth the SFDR is better than -77 dBc. In terms of jitter performance, a very low jitter (1-2 ps) is attractive to engineers building transceiver systems with DAC and ADC. An Altera Stratix 3 device and a 14 bit Texas Instrument DAC were used for the implementation. This paper proposes a feature rich DPLL and a novel technique which improves close-in phase noise performance without impacting transient behavior.

1. INTRODUCTION

The proposed structure is meant to be used as an ultra flexible all digital phase lock loop (DPLL) core. Such a core should be part of the next generation of high performance high density FPGAs. The core software defined architecture is well aligned with the SDR philosophy. Each variable aspect of the core shall be programmable. For example; the loop filter coefficient and scaling will all be software defined and may be changed at any given time. Also, every register loading interface may be designed to be flexible to adapt to any application requirement.

Figure 1 suggests a simple implementation of an all digital PLL. In the proposed DPLL circuitry, contrary to the typical analog approach, a phase and frequency detector (PFD) [6] is no longer needed since the NCO count value "μ", represent the current phase of the NCO and a D Flip-Flop (DFF) will latch in the phase error w.r.t the reference phase on the rising edge of the reference clock. Instead of converting the phase error into a pulse and then integrating with the loop filter we obtain, directly, the exact phase error from the NCO "μ".

2. THE PROPOSED CIRCUITRY

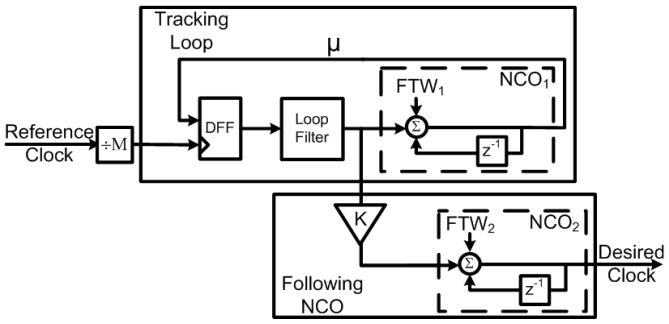

Figure 1: A proposed Rational-K PLL, dual NCO structure

The proposed approach uses two NCOs in one DPLL circuit. The primary or tracking NCO (NCO_1) is in phase lock with the input reference frequency. By loading the secondary NCO (NCO_2) with a scaled version of the error signal and a scaled frequency tuning word (FTW), NCO_2 can generate a signal phased locked to the reference. Note that the only limit in terms of frequency resolution is the number of bits of the FTW and error signal driving the two NCOs. Both scaling factors of the error and FTW are the same value, such that the phase accumulator values in both NCO_1 and NCO_2 only differ by the scaling factor K. K can be any rational number limited by the accumulator number of bits.

2.1. Tracking Loop

The tracking loop uses an approach that was previously proposed and analyzed in [1]. This primary loop only serves as a reference tracking loop. A D-Flip-flop is used as a phase error detector. The reference input is buffered and converted into a CMOS compatible square wave before being fed to the DPLL. This implies that only the zero crossings are used for phase error detection.

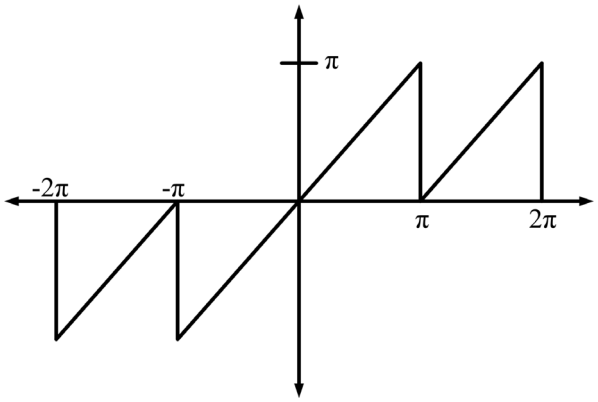

Figure 2: PFD Transfer Function

Figure 2 shows the transfer function of the DFF used as a phase and frequency detector (PFD). The gain of the detector is unity, for a given sampled phase error, the exact same error signal will be fed to the loop filter. The transfer function shows a range of ±π.

The phase of the tracking NCO_1 will get sampled on the rising edge of the reference clock. In the case of phase lock the sampled phase will be near zero and the generated error and correction will also be near zero. Otherwise if there is a phase or frequency error it will be sent to the loop filter and the correction will be applied to NCO_1. From this you can see that close-in reference referred jitter will be introduced by any external buffers used to convert the reference clock to the appropriate CMOS level required by the FPGA input buffer and any internal circuitry up to the DFF used for the NCO_1 phase sampling instant. Usually, the FPGA vendor will use dedicated buffers for such a DPLL core and their jitter performances would be well characterized by the vendor. The out of loop phase noise will be driven by system clock phase noise. With this tracking approach the transient response is limited to a single DPLL loop transfer function, effectively there is no dependency of the transfer function to scaling factor K. Therefore the frequency of operation will not affect loop filter coefficients during change in output frequency in order to maintain the loop bandwidth, or PLL noise shaping. The tracking loop incorporates a loop filter that needs to be designed to ensure proper loop operation to meet the requirements of the application. In the SDR concept it is of the utmost importance that the filter coefficients and frequency may be changed at any given time.

2.2. Secondary NCO: Type 1

The error signal out of the tracking NCO loop filter is sent directly to the scaling factor K, and applied to NCO_2. Like the well known reference phase noise scaling due to frequency multiplication, this approach will also have 20LogK reference phase noise degradation.

2.3. Secondary NCO: Type 2 Improved Phase Noise

In the type 2 implementation a filter is included either before or after the K factor scaling. Effectively, NCO_2 will differ in terms of transfer function since its response will be a filtered version of that of NCO_1. The reason for this added filtering is to further improve the phase noise (PN) performance of the overall DPLL. Note that such added filtering will need to be designed in accordance with the system requirements. The key attribute of NCO_2 FIR low pass filter is that the sum of its coefficient must be equal to unity. This way once the loop has reached steady state, the error signal remains unchanged except for the noise filtering. Note that the added filtering can be added, removed and modified by simply changing the filter coefficients in accordance with the SDR architecture philosophy.

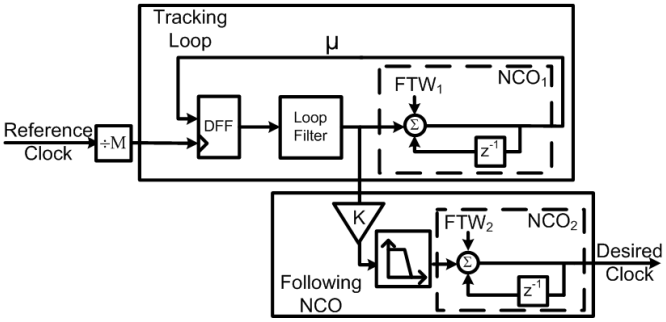

Figure 3: Type 2: Proposed Phase Noise Improvement Method

2.3.1 Secondary NCO: Phase noise and transient optimization algorithm

The introduced phase noise filtering could initially be bypassed but enabled; once lock is detected, the pre-loaded filter could be engaged, see Figure 4. All of its delay element would have the delayed version of the lock state but with less noise. The only phase hit introduced by such algorithm would be equivalent to the added noise rejection; this requires the lock detection trigger to have some build in margin. Such a scheme would maintain almost identical transient behavior of both NCOs, and once locked, we could obtain a considerable improvement in phase noise characteristics of the NCO_2 output.

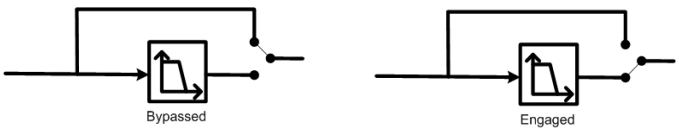

Figure 4: Bypassing and engaging of the phase noise filter

Considering that this method permits fast settling time and optimized phase noise simultaneously, where as typical PLL implementation only offers only one of the two, this novelty could also make its way into future discrete component

offerings, just like current DDS devices. If this method is combined with process like InP DHBT [5], it could enable an RF synthesizer that offers fast settling time, optimized phase noise and high frequency resolution simultaneously.

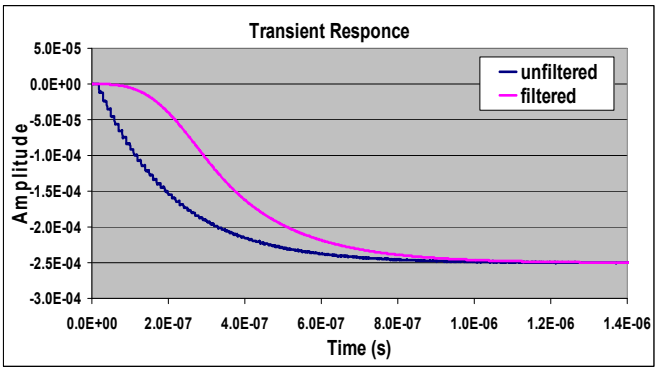

Figure 5: Transient Response of the Tracking and Secondary NCO

Figure 5 presents the effect of the filtering on NCO_2 error signal transient. We can observe that the time response is delayed. Note that the Simulink simulation did not include the lock detection algorithm which obviously should make both responses almost identical.

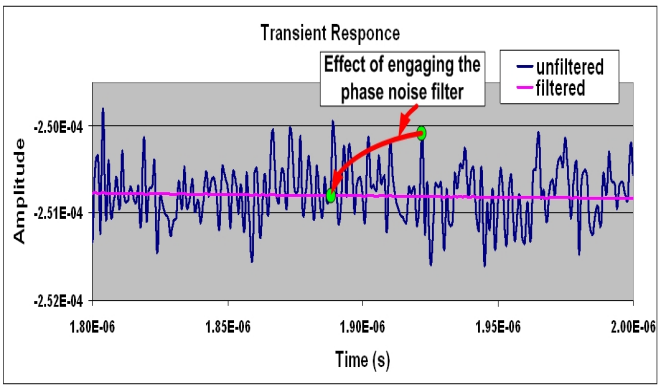

Figure 6: Filtered and unfiltered steady state responses and noise filtering engaging algorithm detail

Figure 6 presents the filtered error signal for both NCO_1 and NCO_2. Some AWGN noise was added before the tracking loop filter of NCO_1 to mimic noise introduced by the reference clock. The observed reduction in error signal noise converts directly to a reduction in reference induced jitter. Note that the in band noise will still have the 20logK degradation, but the added delay of the extra filtering will be the only settling time increase from the locked state of NCO_1. Also, Figure 6, presents the change from bypassed to engaged after lock detection. The error signal changes from the noisy signal to the delayed filtered version. The delay is equal to the number of coefficient in the filter times the FPGA system clock period.

2.4. Tracking Loop Filter Design and Implementation

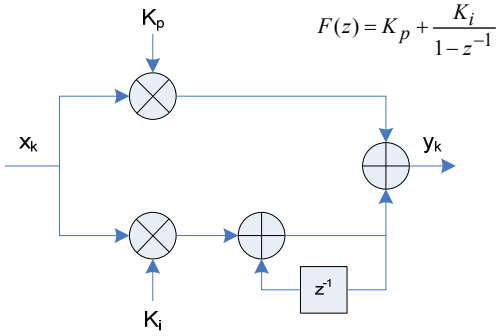

Figure 7: Lead-Lag Loop Filter

Figure 7 present the loop filter used in the implementation. Lead-Lag filter implementations are well known and simple to design.

$$N = 1$$
$$W_n = loop\_natural\_frequency$$
$$T = FPGA\_Clock\_Period$$
$$K = \frac{1}{2\pi}$$
$$\zeta = 3$$
$$K_i = \frac{N \cdot W_n^2 \cdot T^2}{2\pi}$$
$$K_p = \frac{2 \cdot N \cdot \zeta \cdot W_n \cdot T}{2\pi}$$

Equation 1: Design equations for loop coefficients calculation, Ref [2]

From Equation 1 we can observe that a change in the two coefficients can be easily obtained and if the change is made by small increment no major phase discontinuity will be introduced.

3. IMPLEMENTATION

3.1. Hardware Setup and Implementation

We implemented the proposed 2nd order DPLL type 1 scheme on an FPGA platform. Figure 8 presents the hardware setup used to implement and test. The FPGA design and simulation software used were Quartus, Synplify Pro and ModelSim. The FPGA device is a Stratix III and the DAC is 14 bit Texas Instrument. The measurements of spectrum, phase noise, and jitter are done with an E5052B signal source analyzer from Agilent and an FSQ-26 spectrum analyzer from Rohde and Schwarz.

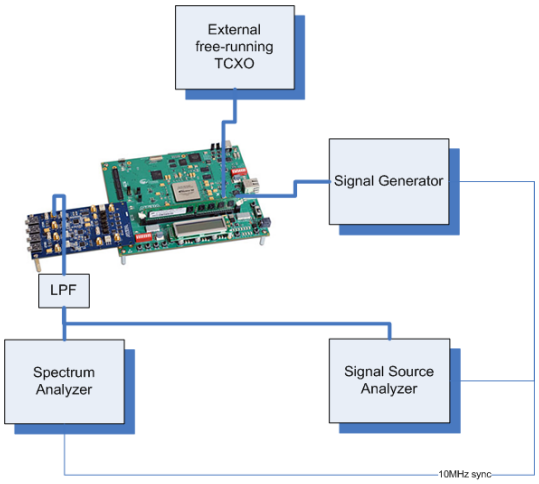

Figure 8: Implementation with Altera Evaluation Boards

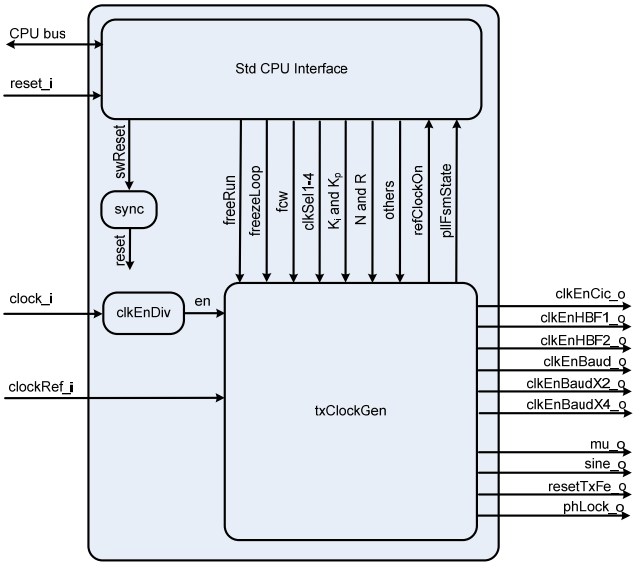

Figure 9: Block diagram of VHDL implementation

Figure 9 shows the block diagram of the DPLL core. We can see that a flexible CPU interface is included to enable easy core reconfiguration. The FPGA used in the implementation was an EP3SE80F780I4L. The speed optimization option was selected during the place and route routine. Table 1 results were obtained using Quartus II v9.0. The loop filter accumulator and the NCOs accumulator bit widths were all selected to be 52 bits.

Max Freq (MHz)	FF	ALUT	Memory		Multipliers (18 bits)
			Bits (Kb)	M9K	
205	3868/64000	3779/64000	0/6331	0/495	2

Table 1: Resources allocation and speed

3.2. Phase Noise Measurements

Phase noise measurement was performed to characterize the performance of the proposed scheme. The measurements were done with many generated frequencies. Figure 10 presents an overlay of both the reference (10MHz OCXO from Vectron Int.) and DPLL measured phase noise for one typical frequency (27.3MHz). We can observe that above 100Hz the reference phase noise is filtered out by the DPLL process. Above this point the system clock (100MHz TCXO from Vectron Int.) phase noise is the main contributor. The DPLL output phase noise @ 1 kHz offset is -125dBc/Hz and 1.5ps rms of integrated jitter, which converts into 14.9 mdeg rms or -74.7dBc. All integrated measurements are from 10 Hz to 5 MHz. Also, a measured residual FM of 515Hz is present. We can observe that the broadband noise introduced by the FPGA, DAC, and crystals is on average ~-143dBc/Hz. We can see that one of the major contributors is the reference crystal. The performances are similar over all tested frequencies.

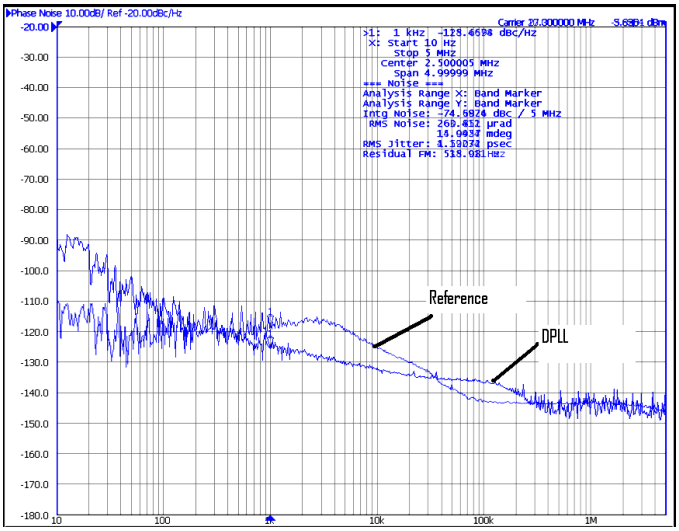

Figure 10: Phase Noise Measurement of the Reference (10MHz) and DPPL Output (27.3MHz).

3.3. Spurious Emission

Figure 11 presents the close in spurious free dynamic range (SFDR) over a span of 200 kHz. The SFDR is 77 dB.

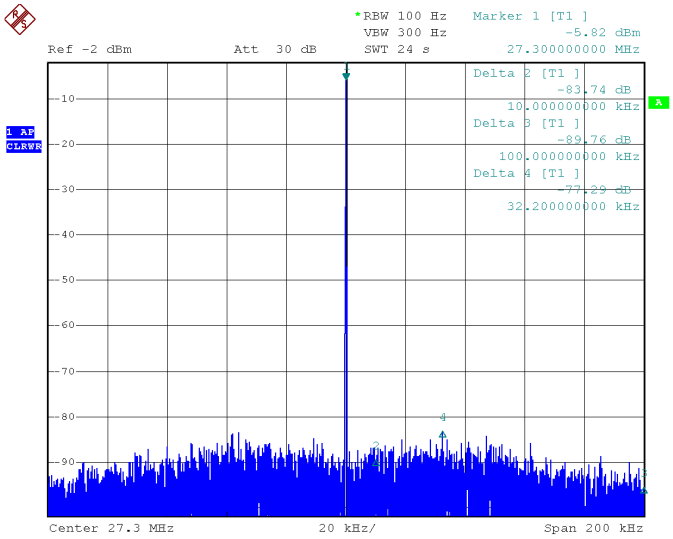

Figure 11: ±100kHz SFDR=77dB

Figure 12 shows a typical spectrum at the output of the DAC before the image filter. The markers 2 and 3 are sysClk ± synthesized frequency, (i.e. 100±27.3MHz). Marker 4 and 5 are (2*sysClk) ± synthesized frequency, (i.e. 200±27.3MHz) where sysClk is the frequency of the system clock driving the FPGA circuitry. This compares well to typical performance of a stand-alone DDS [4], but with the added benefit of being phase and frequency locked to a reference clock.

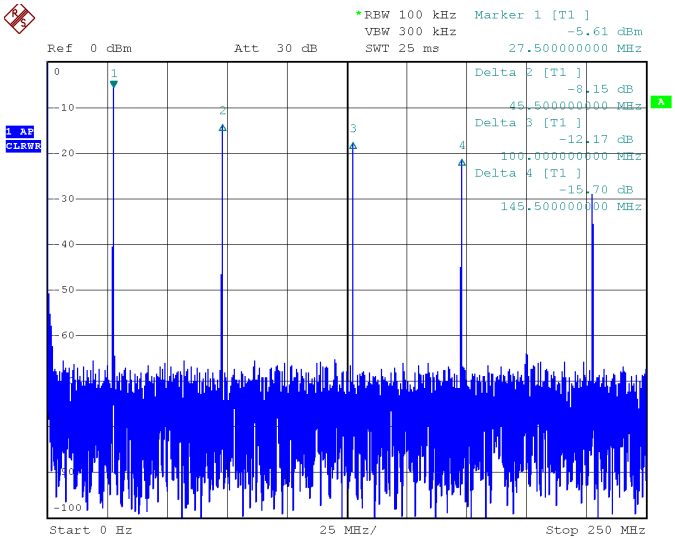

Figure 12: Spectrum at DAC without Filtering

4. EXTENSIONS AND DERIVED CIRCUITS

4.1 Multiple Synchronous Frequency Generation

Figure 13 presents an efficient way of generating multiple frequencies that are all phased locked to the same reference. In some case it maybe desirable to have a completely phase synchronous system, i.e. Tx and Rx. Also, it is possible to ensure static phase offset between individual frequencies such that they would not add constructively. Since the different NCOs are independent each one is highly frequency agile and can be reconfigured extremely rapidly. Therefore the switching between two desired frequencies can be done within a system clock cycle, with only a change in the frequency tuning word (FTW) and a synchronous load of NCO_2 with a scaled version of the current phase of NCO_1. All frequencies are already synchronized to the reference clock via the first NCO as always.

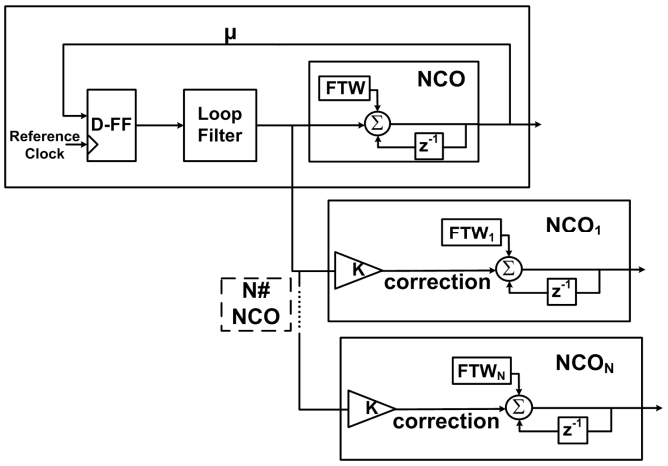

Figure 13: Multiple Synchronous NCOs

4.2 Asynchronous Interpolation for Multi-rate Systems

Figure 14 depicts the use of the DPLL in a modem implementation. In this scheme, the Farrow filter is a device to perform a rate translation via interpolation based on the value of the current symbol phase μ. The proposed NCO is to provide a symbol timing phase and strobe for this Farrow interpolator. With the use of such DPLL one can create any symbol clock and still be phase locked to the reference clock. This is paramount to flexible SDR platform design.

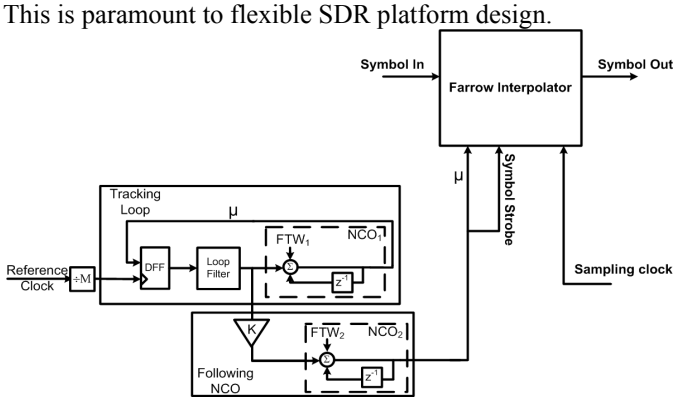

Figure 14: Asynchronous Interpolation for SDR, Ref [3]

5. CONCLUSIONS

The work presented in this paper covers a novel method of Rational-K frequency synthesis, an implementation and

measurement results are presented for a type one NCO. Also, a novel technique which permits the optimization of both phase noise and transient behavior independently is presented with an algorithm that permits optimized transient and phase noise performances simultaneously.

6. ACKNOWLEDGMENTS

We would like to thank Ultra Electronics TCS for supplying lab equipment and supporting us during this project. Special thanks to the members of the DSP and RF group for sharing their opinions and experience.

8. REFERENCES

[1] "All-Digital Phase Locked-Loop Part 1- Fundamentals", Doan Nguyen Vo, Alexandre Marsolais, 5th International Waveform Diversity & Design Conference, 2010.

[2] "Discrete-Time Signal Processing", Alan V. Oppenheim, Ronald W. Schafer, Prentice Hall.

[3] "Interpolation in Digital Modems-Part 11: Implementation and Performance", Lars Erup, Member, IEEE, Floyd M. Gardner, Fellow, IEEE, and Robert A. Harris, Member, IEEE transactions on communications, vol. 41, no. 6, June 1993, pp 998.

[4] "AD9956 Datashet, 2.7 GHz DDS-Based AgileRFTM Synthesizer", Analog Devices AD9956 Datasheet Rev. A , www.analog.com, 2004.

[5] "Direct Digital Synthesizer With Sine-Weighted DAC at 32-GHz Clock Frequency in InP DHBT Technology", Steven Eugene Turner, Student Member, IEEE, and David E. Kotecki, Senior Member, IEEE, IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 41, NO. 10, OCTOBER 2006.

[6] Alexandre Marsolais, N.M. El-Gamal, M. Sawan, "A CMOS frequency synthesizer covering the lower and upper bands of 5 GHz WLANs," Micro-Mechatronics and Human Science, 2005 IEEE International Symposium, vol. 3, pp 1146-1149, 27-30 Dec. 2003