FPGA Implementation of OFDM SDR

Prabhudev B L*, Shashidhara H R** and Dr.Siddesh G K***
*Research Scholar, VTU, Belgaum, Assistant Professor, DYPSOEA, Pune
Email:blprabhu14@gmail.com
**Assistant Professor, JSSATE, Bangalore
Email:shahidharahr@gmail.com
***Associate Professor, JSSATE, Bangalore
Email: siddeshgundagatti@gmail.com

Abstract: A Software Defined Radio (SDR) is a transmitter and receiver system that uses digital signal processing (DSP) for coding, decoding, modulating, and demodulating data. This paper presents the framework for hardware implementation of SDR using Orthogonal Frequency Division Multiplexing (OFDM). The framework comprises of VLSI mapping of algorithms, Orthogonal Frequency Division Multiplexing (OFDM), Quadrature Phase Shift Keying (QPSK), Fast Fourier Transform (FFT) Algorithms and most importantly, the algorithm for Direct Digital Frequency Synthesis (DDFS). A digital frequency synthesizer with optimized time and area resources has been proposed for the SDR. This VLSI implementation of the DDFS computes the sine and cosine function on a single edge of clock, thus proving to be optimized in terms of area and speed. Fixed-Point implementation was accomplished with ModelSim simulator. Verilog HDL was used as a description language for mapping Algorithms in VLSI. Xilinx Spartan 3 XC3S200 Field Programmable Gate Array (FPGA) was chosen as a Hardware Platform for the System Implementation.

Keywords: Software Defined Radio, Direct Digital Frequency Synthesis, Orthogonal Frequency Division Multiplexing.

Introduction

A Software Defined Radio (SDR) is defined as a radio in which the receive digitization is performed at some stage downstream from the antenna, typically after wideband filtering, low noise amplification, and down conversion to a lower frequency in subsequent stages - with a reverse process occurring for the transmit digitization. In an SDR, Digital Signal Processing in flexible and reconfigurable functional blocks define the characteristics of the radio. As technology progresses, an SDR can move to an almost total Software Radio (SR), where the digitization is at (or very near to) the antenna and all of the processing required for the radio is performed by software residing in high-speed digital signal processing elements. The SDR will occur in the near term, migrating to the SR in the longer term, subject to the progression of core technologies. The need for such progression will be a function of the application. For example, a base station application may require and/or be able by virtue of technology advances and design latitude to move to an SR.

But a handset or portable terminal, because of numerous constraints, may not need or be able to progress beyond an SDR [1]. Researchers in the area of Communication and VLSI have a growing tendency to develop state-of-the-art architectures for SDR. Recent developments include the frameworks proposed in [2][3][4][5].

The backbone of SDR framework in the digital domain is a direct digital frequency synthesizer (DDFS) . Using DDFS, we can generate the high frequency carrier waves in digital domain and modulate the message on it, and then convert it to analog form using a digital to analog converter (DAC) before antenna. This is referred to be the digital up-conversion (DUC) [6][7]. The counter-part of digital up-converter, on the receiver end, is digital down converter (DDC). VLSI has been a key technology for mapping communication and signal processing algorithms in hardware. A key challenge for researchers has been the optimization of the VLSI circuit in terms of area as well as time. Conventional methods include pipelining, retiming, parallel processing etc. Unconventional methods refer to evolutionary and genetic algorithms, genetic programming and Cartesian genetic programming [8][9][10][11]. The optimization of the DDFS includes Lookup Table based designs and Coordinate Rotation Digital Computer (CORDIC) based designs. While [12][13][14][15][16][17] have used Lookup Table based approaches, CORDIC based approaches have been used in [18][19][20] for the generation of Sine and Cosine. Janiszewski [21] has adapted a hybrid scheme for the design of Numeric Controlled Oscillator (NCO) using both the Lookup Table based and CORDIC based approach.

Lookup Table based designs require huge ROMs for implementation and are declared to be area-hungry. On the other hand, CORDIC based techniques use iterative algorithms for the computation of Sine and Cosine functions and are computationally inefficient. Singleton [22] used basic trigonometric identities to compute the values of Sine and Cosine. Each sample of Sine and Cosine requires 4 multiplications and 2 additions. However, the design in [23] is even simpler and faster. It utilizes 2 adders and 2 multipliers to generate a sample of sine and cosine. This design is more area and time efficient than CORDIC

and look-up table based approaches. But the Architecture for VLSI implementation proposed by [23], as shown in Fig. 1 has utilized registers being triggered on the positive and negative edges of the same clock. Such designs are not synthesizable on Field Programmable Gate Arrays (FPGA). So, in this work, a novel architecture has been proposed which generates the Sine and Cosine function using the single clock edge. The proposed architecture reduces the required hardware resources while realizing a synthesizable efficient design.

The remaining of the paper is organized as; Section II briefly describes the overall framework and the functionality of each block of the framework. Detailed discussion on the implementation of the blocks and especially the FFT block has been made in section III. DDFS is the key component of the DUC and DDC blocks and is the focus of this paper. Thus, a separate section i.e. section IV has been dedicated for the discussion of DDFS. Finally, section V concludes the paper.

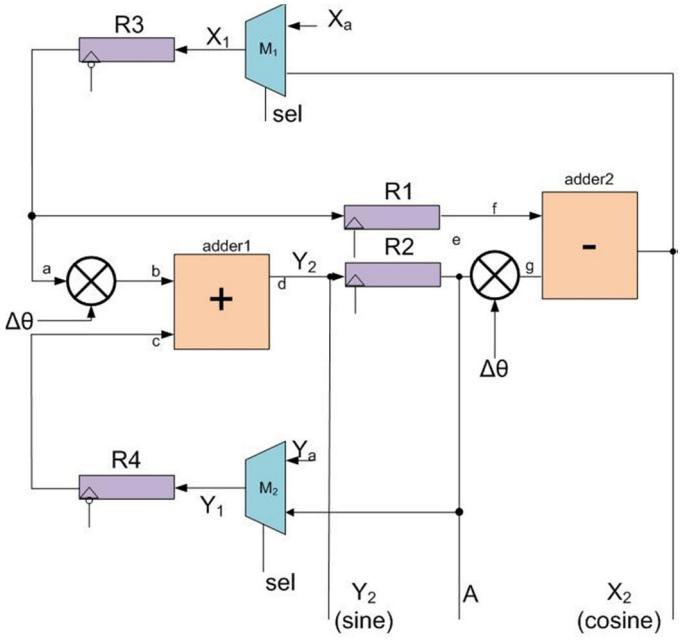


Fig. 1 Architecture proposed in [23]

Methodology

This work contains simulation and Verilog HDL implementation of OFDM based transmitter and receiver system. After floating point simulation of the framework, Verilog HDL has been used for fixed point simulation and description of hardware details. The transmitter first converts the input data from a serial stream to parallel sets. Each set of data contains one information bit for each carrier frequency. Then, parallel data are modulated to the orthogonal carrier frequencies. The IFFT converts the parallel data into time domain waveforms. Finally, these waveforms are combined to create a single time domain signal for transmission. The channel simulation will allow for us to examine the effects of noise and multipath on the OFDM scheme. By adding small amount of random data to the transmitted signal, simple noise can be simulated. Multipath simulation involves adding attenuated and delayed copies of the transmitted signal to the original. This simulates the problem in wireless communication when the signal propagates on many paths. For example, a receiver may see a signal via a direct path as well as a path that bounces-off of a building. The receiver basically performs the inverse of the transmitter by first separating the data into parallel streams. The FFT converts the parallel data streams into frequency domain data. Demodulation down-converts this information back to the baseband. Finally, the parallel data are converted back into a serial stream to recover the original signal. The overall block diagram of the system has been shown in Fig. 2.

Block description and implementation

Serial-to-parallel Conversion and Parallel-to-Serial Conversion

The Serial-to -Parallel converter takes 8 bits at a time and produces four parallel streams of two bits each. It takes 8 clock cycles. The ASM chart for serial to parallel conversion has been shown in Fig. 3. On the receiver end, the parallel-to-serial converter takes four parallel streams, each one consisting of two bits and transforms it into a stream of 8 bits serial data.

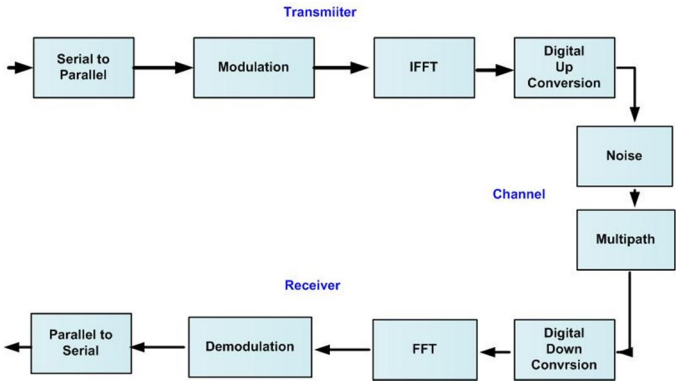


Fig. 2 Block Diagram of OFDM Software Defined Radio

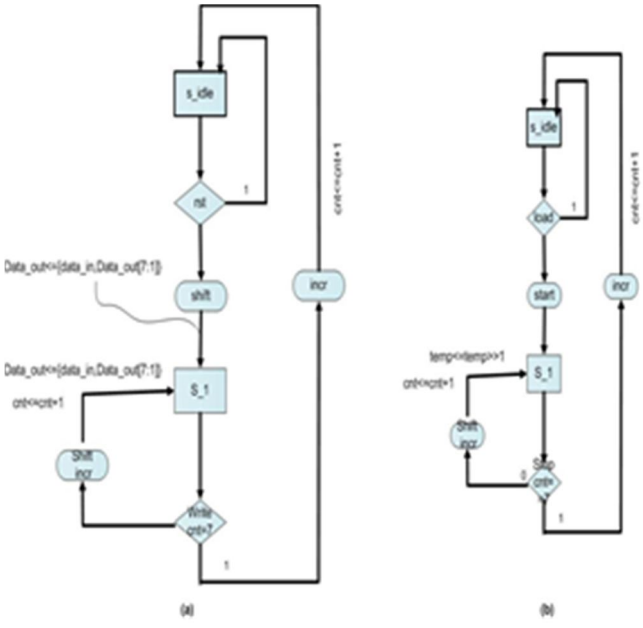


Fig 3. ASM Chart for (a) Serial to Parallel Conversion (b) Parallel to Serial Conversion

Modulation and Demodulation

In this work, we have used Quadrature Phase Shift Keying (QPSK) Modulation. The data is divided into chunks of a set of bits, each containing 2 bits of data, leading to 4 distinct combinations. Each combination is assigned a different symbol [24]. In order to ensure maximum likelihood, the separation between angles should be as large as possible. Hence, an angle assigned to each symbol is **45^o, 135^o, 225^o and 315^o**.

Table.1. Corresponding Angle of Qpsk Modulation

Bit Sequence	Modulation Angle	Mapped Sequence
00	45^o	$+1+j$
01	135^o	$-1+j$
10	225^o	$-1-j$
11	315^o	$+1-j$

Fast Fourier Transform

The FFT machine is partitioned into data path and controller i.e. the Finite State Machine (FSM). It simplifies the architecture design and understanding. The data path presented in this work contains a single butterfly stage, consisting of N/2 butterflies for N -order FFT, registers for holding real and imaginary data as well as the intermediate results of the various stages of FFT, real and imaginary twiddle memories and buses. The 8 -point FFT design, in this work, consists of butterfly stage,

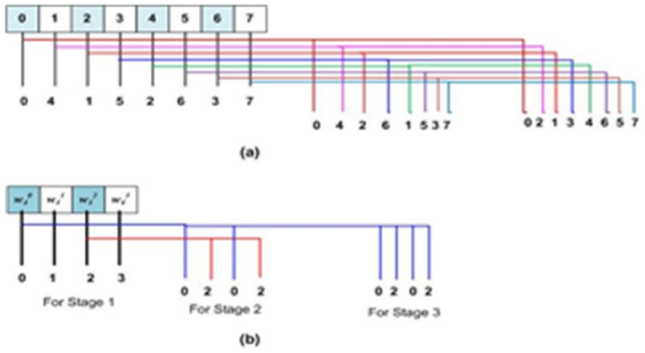


Fig. 4 Memory Representation. (a) Input Memory. (b) Twiddle Memory

twiddle memories, input register banks that also store the values calculated during the various stages of FFT, multiplexers for steering signal. There are 3 stages in 8-point FFT/IFFT, so input memories and twiddle memories give 3 outputs, one for each stage, as shown in the Fig. 4. In this way, the design of FSM is simplified so that it keeps track of the FFT current stage only, i.e. the stage being computed at that particular time. The butterfly stage comprises of N/2 butterflies. The FFT data path has been shown in Fig. 5. Each sub-stage on the FFT takes input data in different order, and the twiddle values used in each sub -stage are also different. Thus, this FFT design simultaneously generates output for each sub-stage. The FSM selects one out of the three outputs, depending upon the stage of FFT that is being calculated. The state transition graph for the FFT FSM has been shown in Fig. 6. The number of states in FSM depends upon the FFT order. Initially, when the FFT machine is reset, the FSM enters the IDLE state, which means that the FFT machine is waiting for data to be processed. Once 8 samples of data are received by the serial-to-parallel converter before the FFT, a signal Readyin is asserted.

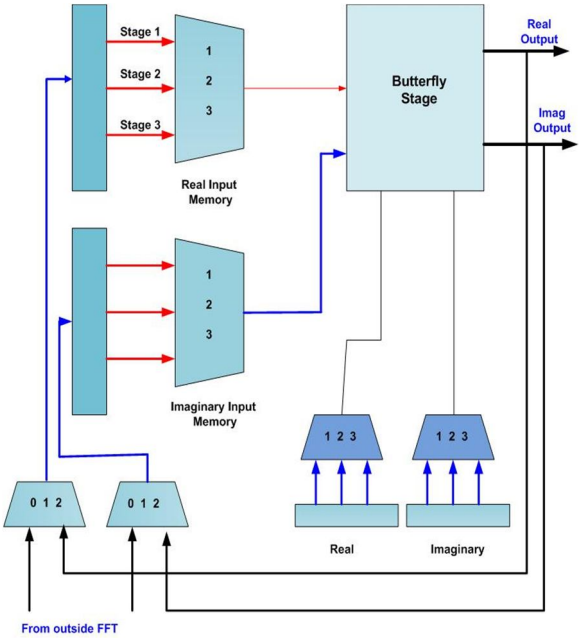


Fig. 5 FFT Data Path

This signal is originated by serial-to -parallel converter. After the assertion of Readyin signal, FFT FSM goes to the LOAD state; otherwise it stays in the IDLE state. In the LOAD state,
the FSM loads the input memories (real and imaginary) with input, and starts the FFT operation. On the next positive edge of the clock cycle, the FSM enters STAGE-1 state, which means that stage 1 of the 8-point FFT is calculated. The number of clock cycles the FSM stays in STAGE-1 state depends upon the specific implementation of the multipliers in the butterfly. Faster multiplier implementation means reduced clock cycles per computation stage of the FFT. After the first stage of FFT computation is calculated, the FSM enters STAGE-2 state, which is similar in operation to the STAGE-1. Following this, the FSM enters STAGE-3 , and returns to the IDLE state after completion of STAGE-3, and the whole cycle starts again.

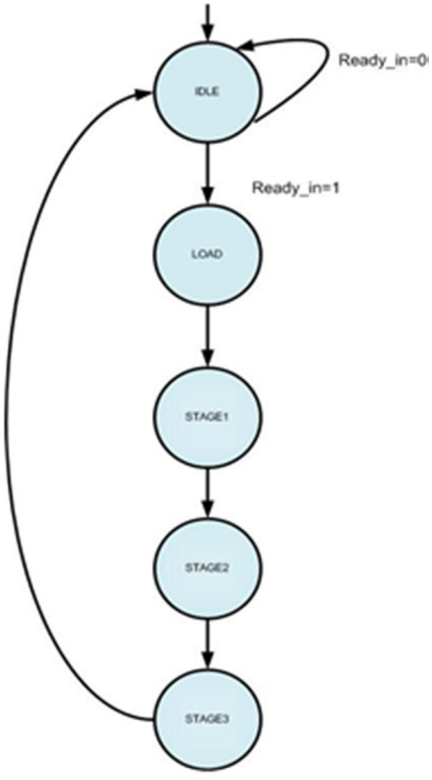


Fig. 6 State Transition Graph for FFT FSM

The Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) are derived from the main function which is called Discrete Fourier Transform (DFT). The equation 1 shows the DFT.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k/n} \qquad \text{Eq. 1}$$

X(k) represent the DFT frequency output at the k-the spectral point where k ranges from 0 to N-1. The quantity N represents the number of sample points in the DFT data frame. The quantity x(n) represents the n-th time sample, where n also ranges from 0 to N-1. In general equation, x(n) can be real or complex.

The single butterfly model used in this work saves hardware resources. The butterfly stage takes input in the Q8.8 format, and gives output in Q8.8 format. There are two input memories for storing real and imaginary parts of the complex inputs. As FFT is a parallel process, so the memories do not give or store data serially but rather store or provide the data in chunks of particular number of bits, depending upon the order of FFT. The input memories are not only used to store the inputs at the beginning of the FFT operation, but also used to store output of the butterfly stage at the completion of the subsequent FFT stages. This helps in reducing the memory required for the FFT machine. The input memories are designed so that they can be read at the positive edge of the clock and data can be written to them at the negative edge of the clock. This helps in synchronization and ensures that data is not read before the completion of the modification of data in the memories. The twiddle memories are designed to give output at the positive edge of the clock. IFFT is defined as the equation below

$$X(k) = 1/N \sum_{n=0}^{N-1} x(n) W^{-nk} \qquad \text{Eq. 2}$$

The architecture for IFFT is almost similar to that of FFT except the division by 8 after completion of IFFT operation. The division by 8 in IFFT architecture is accomplished by shift registers and a change in twiddle factors. The rest of architecture of the data path and the FSM is the same.

Direct Digital Frequency Synthesis (DDFS)

The DUC is a digital circuit which implements the conversion of a complex digital baseband signal to a real pass band signal. The input complex baseband signal is sampled at a relatively low sampling rate, typically the digital modulation symbol rate. The Digital Down Converter (DDC) is the counter -part on the receiver end. The detailed description on DUC and DDC can

be found in [25][26]. This section focuses on the efficient hardware implementation of DDFS, which is backbone of the DUC and DDC.

A. Proposed Architecture

The proposed architecture for VLSI implementation of the DDFS is shown in Fig. 7. This architecture utilizes two adders, two multiplexers, two multipliers and two registers. The presentation is general and applicable to any bit length. The data path elements are 32 bits wide. As compared to the architecture proposed in [23], the required number of registers has been reduced to two, instead of four. Depending upon the number of bits used, it results in considerable reduction of hardware resources.

Initially Yi and Xi are fed to the Registers because sel is 1. On the next positive edge of clk, these seed values are used to compute the values of Sine and Cosine. After the first clock cycle, sel is 0 and now the multiplexers only act as simple wires for rest of the clock cycles. The previous value of Cosine is multiplied by Δθ and added with the previous value of Sine to generate the current value of Sine, Yc. Similarly, the previous value of Sine, Yp is multiplied by Δθ and subtracted from the previous value of Cosine, Xp to generate the current value of Cosine, Xc.

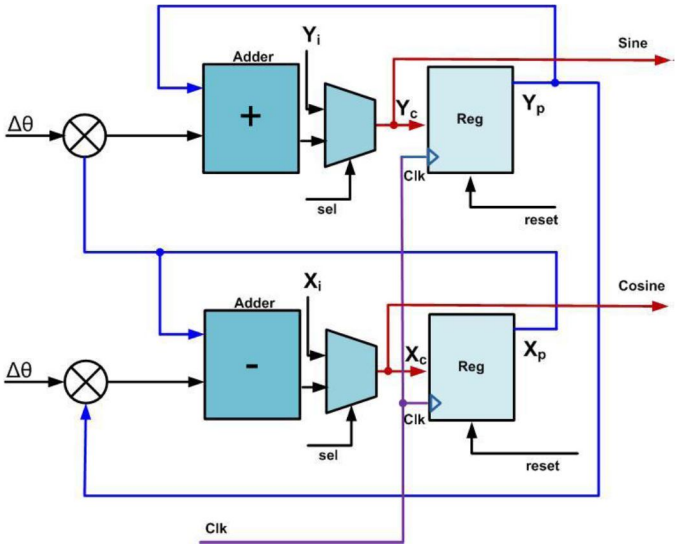


Fig. 7 The Proposed Architecture for DDFS

Simulation Results

To verify correct functionality of the filters in the digital down conversion processing chain, the spectrum of the incoming signal is defined with the following specifications

Sampling frequency of the ADC = 6 MHz

Carrier frequency of the desired signal = 3 MHz

One sided bandwidth of the desired signal = 10 KHz

Frequency where an unwanted signal in the spectrum exists = 15 KHz

Using these specifications a test signal is generated in MATLAB. The discrete values of the test signal are stored in the text file to give it as input to the VHDL Design. The test signal contains two desired frequencies- (3 MHz + 10 KHz), (3 MHz - 10 KHz), and two unwanted frequencies-(3 MHz + 15 KHz), (3 MHz - 15 KHz).

The output waveform of the Digital down conversion is shown in the figure 8. The results are then compared with the simulated result got out of MATLAB shown in fig 9.

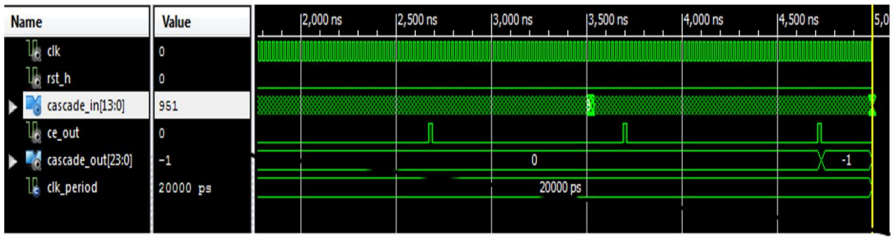
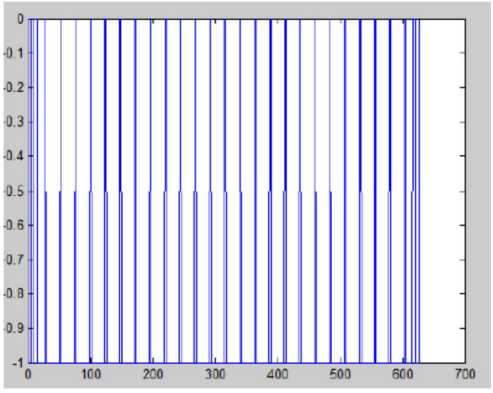


Fig 8: Simulation results



X-Axis:- FFT in db ; Y-Axis:- Frequency in Hz

Fig 9: Simulation result of MATLAB

Conclusion

In this paper, we presented simplified hardware architecture for OFDM Software Defined Radio. This paper also presented an area and speed optimized architecture for Direct Digital Frequency Synthesis, one of the backbone for SDR. The proposed framework consumes lesser silicon area and is realizable on FPGA. This is due to the fact that it generates the sine and cosine values on a single edge of the system clock. The required memory resources are extremely less as only two 32-bit registers have been used in the architecture. The future goals include the reduction of the critical path, so that the framework may work at even higher clock rate and the range of frequency generated may be increased. Besides, the design may be optimized leading to a more intelligent framework to help in realization of a more ideal Cognitive Radio.

References

[1] W. Tuttlebee, Software Defined Radio Enabling Technologies. John Wiley and Sons, 2002.
[2] X. Qi, L. Xiao, and S. Zhou, “A novel GPP-based Software-Defined Radio architecture,” in 7th Internatioal ICST Conference on Communications and Networking in China (CHINACOM), 2012, pp. 838 – 842.
[3] E. Nicollet, “DSP software architecture for Software Defined Radio,” in IEE Colloquium on DSP enable Radio, 2003, pp. 1 – 9.
[4] V. Barral, J. Rodas, J. A. Garcia-Naya, and C. J. Escudero, “A novel, scalable and distributed software architecture for software-defined radio with remote interactionpp. 80 – 83.,” in 19th International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 80 – 83.
[5] N. Ali, “Novel architecture for software defined radio,” in IEEE International Conference Microwaves, Communications, Antennas and Electronic Systems (COMCAS), pp. 1 – 4.
[6] T. Hentschel, T. Hentschel, Sampling Rate Conversion in Software ConfigurableRadios. Artech House Mobile Communication Series, 2002.
[7] R. B. Staszewski, K. Muhammad, and D. Leipold, “Digital RF Processor DRPTM for Cellular Phones,” Dallas, TX 75243, USA.
[8] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. John Wiley and Sons, 1999.
[9] T. Weise, Global Optimization Algorithms - Theory and Applications. 2008.
[10] W. B. Langdon, Genetic Programming and Data Structures. Springer, 1998.
[11] J. Miller and P. Thomson, “Cartesian genetic programming,” in
[12] Third European Conference on Genetic Programming EuroGP2000, 2000, pp. 121–132.
[13] D. A. Sunderland, S. S. Strauch, H. Wharfield, T. Peterson, and C. R. Cole, “D. A. Sunderland, R. A. Cmos/sos frequency synthesizer lsi circuit for spread spectrum communications,”,” IEEE Journal of Solid-State Circuits, vol. 19, no. 4, pp. 497–566.
[14] P. W. Ruben, E. F. Heimbecher, and D. L. Dilley, “Reduced size phase-to-amplitude converter in a numerically controlled oscillator,” U.S. Patent 4855946.
[15] R. D. McCallister and D. Shearer, “Numerically controlled oscillator using quadrant replication and function decomposition,” U.S. Patent 4486846.
[16] H. T. Nicholas, H. Samueli, and B. Kim, “The optimization of direct digital frequency synthesizer in the presence of finite word length effects performance,” in 42nd Annual Frequency Control Symposium, pp. 357–363.
[17] L. A. Weaver and R. J. Kerr, “High resolution phase to sine amplitude conversion,” U.S. Patent 4905177.
[18] D. D. Caro, E. Napoli, and A. G. M. Strollo, “Direct digital frequency synthesizers using high-order polynomial approximation,” in IEEE International Solid State Circuits Conference, pp. 134–135.
[19] T. Zaidi, Q. Chaudry, and S. A. Khan, “An area and time efficient collapsed modified cordic ddfs architecture for high rate digital receivers,” in IEEE 8th International Multitopic Conference, pp. 677–681.
[20] C. Y. Kang and E. E. Swartzlander, “Digit-pipelined direct digital frequency synthesis based on differential cordic,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 53, no. 5, pp. 1035–1044.
[21] M. Ghariani, N. Masmoudi, M. W. Kharrat, and L. Kamoun, “Design and chip implementation of modified cordic algorithm for sine and cosine functions application: Park transformation,” in Tenth International Conference on Microelectronics ICM 98, 1998, PP 241

– 244.
[22] R. C. Singleton, “A method for computing the fast fourier transform with auxiliary memory and limited high-speed storage,” IEEE Transactions on Audio and Electroacoustics, vol. 15, no. 2, pp. 91 – 98, 1967.
[23] Y. A. Khan, Aneesullah, and H. Ali, “Differential based area efficient rom less quadrature direct digital frequency synthesis,” in IEEE 5th International Conference on Emerging Technologies (ICET), pp. 83 – 88.
[24] B. Sklar, Digital Communications, Fundamentals and Applications, 2nd Ed. University of California.
[25] “Xilinx (Hrsg.): Digital Up Converter (DUC) v1.2. Xilinx,”
[26] “Xilinx (Hrsg.): Digital Down Converter (DDC) v5.0. Xilinx.”
[27] W. S. Gan and S. M. Kuo, Embedded Signal Processing with the Micro Signal Architecture. John Wiley and Sons, 2007.