IMPLEMENTATION OF GENERALIZED DFT ON FIELD PROGRAMMABLE GATE ARRAY

*Wes P. Weydig, Mustafa U. Torun†, and Ali N. Akansu†**

Qualcomm
New Jersey Research & Development Center
500 Somerset Corporate Boulevard, Bridgewater, NJ 08807, USA

†Department of Electrical and Computer Engineering
New Jersey Institute of Technology
University Heights, Newark, NJ 07102 USA

ABSTRACT

We introduce the implementation of Generalized Discrete Fourier Transform (GDFT) with nonlinear phase on a Field Programmable Gate Array (FPGA.) After briefly revisiting the GDFT framework, we apply the framework to a channel equalization problem in an Orthogonal Frequency Division Multiplexing (OFDM) communication system. The block diagram of the system is introduced and detailed explanations of the implementation for each block are given along with the necessary VHDL code snippets. The resource usage and registered performance of the design is reported and alternatives to improve the design in terms of performance and resolution are provided. To the best of our knowledge, this is the first hardware implementation of GDFT reported in the literature.

Index Terms— GDFT, FPGA, OFDM

1. GENERALIZED DFT

An N^{th} root of unity is a complex number satisfying the equation

$$z^N = 1 \quad N = 1, 2, \ldots$$

If $z_p^m \neq 1$ with $m = 1, 2, \ldots, N-1$, then z_p is defined as the p^{th} *primitive* N^{th} *root of unity* and m and N must be coprime integers. The complex number $z_1 = e^{j(2\pi/N)}$ is the primitive N^{th} root of unity with the smallest positive argument. There are N distinct N^{th} roots of unity for any primitive and expressed as $z_k = (z_p)^k$ where $k = 1, 2, \ldots, N\ \forall p$, z_p is any of the primitive N^{th} root of unity. As an example, $z_1 = e^{j2\pi/4}$ and $z_2 = e^{j3\pi/2}$ are the two primitive N^{th} roots of unity for $N = 4$. The summation of a *primitive* N^{th} *root of unity* in a geometric series is expressed as follows

$$\sum_{n=0}^{N-1} (z_p)^n = \frac{(z_p)^N - 1}{z_p - 1} = \begin{cases} 1 & N = 1 \\ 0 & N > 1 \end{cases} \forall p \qquad (1)$$

*Corresponding author: akansu@njit.edu

Now, let's define a periodic, constant modulus, complex sequence $\{e_r(n)\}$ as the r^{th} power of the first *primitive* N^{th} *root of unity* z_1 raised to the n^{th} power as expressed in

$$e_r(n) \triangleq (z_1^r)^n = e^{j(2\pi r/N)n}$$

where $n, r = 0, 1, \ldots, N-1$. The sum of its geometric series is expressed according to Eq. 1 as follows [1]

$$\frac{1}{N}\sum_{n=0}^{N-1} (z_1^r)^n = \frac{1}{N}\sum_{n=0}^{N-1} e^{j(2\pi r/N)n} = \begin{cases} 1 & r = mN \\ 0 & r \neq mN \end{cases} \qquad (2)$$

where *m* is an integer. Let's generalize Eq. 2 by rewriting the phase as the difference of two functions $\phi_{kl}(n) = \phi_k(n) - \phi_l(n) = r$ and expressing a constant modulus orthogonal set as follows

$$\begin{aligned} &\frac{1}{N}\sum_{n=0}^{N-1} e^{j(2\pi r/N)n} = \frac{1}{N}\sum_{n=0}^{N-1} e^{j[2\pi\phi_{kl}(n)/N]n} \\ &= \begin{cases} 1 & \phi_{kl}(n) = \phi_k(n) - \phi_l(n) = k - l = r = mN \\ 0 & \phi_{kl}(n) = \phi_k(n) - \phi_l(n) = k - l = r \neq mN \end{cases} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} e^{j[2\pi\phi_k(n)/N]n} e^{-j[2\pi\phi_l(n)/N]n} \\ &= \langle e_k(n), e_l^*(n) \rangle \end{aligned} \qquad (3)$$

and k, l, $n \in \{0, 1, \ldots, N-1\}$. Hence, the basis functions of the new orthogonal set are defined as

$$\{e_k(n)\} \triangleq e^{j(2\pi/N)\phi_k(n)n} \qquad (4)$$

This orthogonal set is called as the *Generalized Discrete Fourier Transform* (GDFT) with *nonlinear phase* [1]. It is observed from Eq. 2 and Eq. 3 that it is an *uncountable set,* and there are infinitely many sets of constant modulus and *nonlinear phase functions* available.

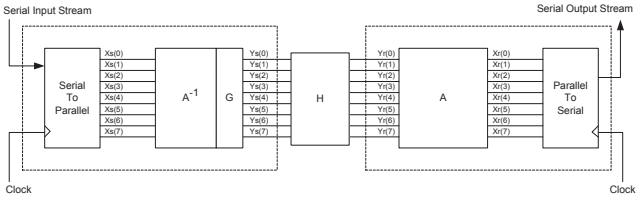

Fig. 1. Block diagram of the GDFT based OFDM communication system.

2. GDFT BASED OFDM SYSTEM

The block diagram of the GDFT based OFDM system under consideration is given in Fig. 1. It can be observed from the block diagram that the transmitter output is an $N \times 1$ vector as given

$$\mathbf{Y}_s = \mathbf{G}\mathbf{A}^{-1}\mathbf{x}_s \tag{5}$$

where $\mathbf{x}_s$ is an $N \times 1$ input data vector, $\mathbf{A}$ is the $N \times N$ discrete Fourier transform (DFT) matrix, and $\mathbf{G}$ is the matrix that provides the non-linearity in the phase domain as introduced by the GDFT framework. The channel response is modeled such that the channel output is equal to

$$\mathbf{Y}_r = \mathbf{H}\mathbf{Y}_s \tag{6}$$

where $\mathbf{H}$ is an $N \times N$ channel response matrix. The receiver multiplies the channel output with the DFT matrix from left as given

$$\mathbf{x}_r = \mathbf{A}\mathbf{Y}_r \tag{7}$$

Substitution of Eqs. 5 and 6 into Eq. 7 yields

$$\mathbf{x}_r = \mathbf{A}\mathbf{H}\mathbf{Y}_s = \mathbf{A}\mathbf{H}\mathbf{G}\mathbf{A}^{-1}\mathbf{x}_s \tag{8}$$

Since the purpose of this study is to implement GDFT on an FPGA, for the sake of simplicity of the discussion, it is assumed that the channel response, $\mathbf{H}$ is known a priori, thus it is possible to choose $\mathbf{G} \perp \mathbf{H}$. Given that and given $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix, it follows from Eq. 8 that

$$\mathbf{x}_r = \mathbf{x}_s \tag{9}$$

which is a desired property for a communication system. In order to further simplify the discussion, it is assumed that the channel is single-path and it only introduces phase distortion, i.e. $\mathbf{H}$ is in the form

$$[H]_{ij} = \begin{cases} e^{j\theta_i} & i = j \\ 0 & i \neq j \end{cases} \tag{10}$$

Also note that, the system under consideration is implemented on a single FPGA chip, and no noise generators are implemented. In other words no noise is introduced in the channel in the system. This is not realistic, but again, our focus in this study is to implement GDFT and it is straightforward for a system engineer to further develop the design we present for real-world applications.

3. IMPLEMENTATION

Details of the several blocks given in Fig. 1 and corresponding implementation techniques employed are given in this section.

3.1. Serial to Parallel and Parallel to Serial Blocks

The first block of the system is the serial to parallel block which receives a serial stream of binary data and delivers it to the next block, the IFFT block, as an $N \times 1$ binary column vector where N is the FFT size used in the OFDM system. This operation is done in hardware by first identifying the start of data transmission. At every N*th* rising edge of the clock, a frame signal is used to reload the N bit register with the next set of values. Moreover, at each rising edge of the clock, a new data bit is being shifted into the register. The frame signal is generated from an $M = \log_2 N$ bit counter that begins counting once the `reset` signal goes from logic high to logic zero.

The FFT block, delivers data in $N \times 1$ size vectors. The last block in the system, parallel to serial block, receives the data from FFT block in vector form and serializes it for the delivery to the user. The start of frame is identified by two signals that are derived from the FFT block. These two signals, when active together, indicate the start of a new frame. The output of the FFT block is loaded into a register at a frame signal. A logic zero is shifted into the least significant bit at every clock cycle. To clock the data serially, $N - 1$ indexed bit is fed with a new value at every clock cycle by virtue of the logic zero being shifted into the register.

3.2. FFT and IFFT Blocks

The FFT and IFFT blocks are designed to perform an N length, radix-2 Cooley-Tukey algorithm [2]. Each butterfly processor that form the FFT and IFFT blocks contains a complex multiplication block that instantiates the multipliers, adders, subtractors for the complex mathematical calculations [3]. The FFT block was designed for the graduate FPGA laboratory manual at NJIT [4]. The IFFT block is basically an FFT block except for a sign change and a scale factor of N, i.e. the output of the block is divided by N. Since the division is by a power of two, it does not consume additional resources in the FPGA. Ignoring the least significant $\log_2 N$ bits at the output does the job. However, this operation results in a loss of precision. In order to partially compensate for this loss, an additional stage is added to the output of the IFFT which rounds the result up or down based on $\log_2 N - 1$ indexed bit

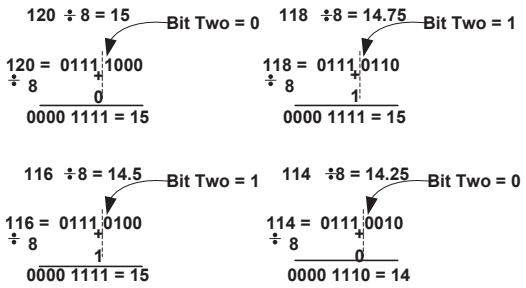

Fig. 2. Examples of rounding operation.

of the IFFT output. A simple example of the operation of this block is illustrated in Fig. 2.

Implementation of this block consists of registering the original bit locations, i.e. $N-2$ through $M = \log_2 N$, and sign extending the pre-rounded result as displayed in Alg. 1. The $M-1$ least significant bits are discarded at this point. Bit $M-1$ is preserved in its own register. Then, bit $M-1$ which was registered at the previous clock cycle is now used to decide whether to round up or not (See Alg. 1.) The whole process incurs two cycles of latency for each rounding process in the design.

3.3. GDFT and Channel Blocks

The output of the IFFT block consists of N complex numbers. In our case, $\mathbf{G}$ is a diagonal matrix with complex elements, the first matrix multiplication in Eq. 5 boils down to a number of N complex number multiplications. Multiplication of two complex numbers requires First Outer Inner Last (FOIL) operation to be performed which consists of four multiplications and two additions. These operations were performed by instantiating an Altera library of parametrized modules (LPM), multiplier, and an LPM add/subtract block [5]. These blocks are configurable and implement the specified operation with either no pipeline delay, i.e. purely combinatorial, or the latency that may be specified in terms of clock cycles in order to stage pipeline operations. The current design is configured to synthesize these modules combinatorially. The channel response block, i.e. Eq. 6, is implemented in the same fashion as described above.

It is also worthy to mention that multipliers implemented in this section require the results to be divided by 2^N since the elements of $\mathbf{G}$ and $\mathbf{H}$ matrices are normalized to N bits. Therefore, in an attempt to reduce the loss of precision, the special rounding operator introduced in the previous section is also utilized in these blocks.

Algorithm 1 Rounding operation.

```
yre_pre_round <= yre(N-1) & yre(N-1) & yre(N-1) &
yre(N-1) & yre(N-2 downto M);
yre_bit_two <= yre(M-1);
IF (yre_bit_two = '1') THEN
  yre_post_round <= yre_pre_round + '1';
ELSE
  yre_post_round <= yre_pre_round;
END IF;
```

4. RESOURCES AND PERFORMANCE

For simulation, resource, and performance analysis studies, the FFT length parameter of the system is selected as $N = 8$. The length-8 FFT and length-8 IFFT require 12 multipliers, each along with 12 adders/subtractors. These elements are required to implement the Cooley-Tukey algorithm [2]. Along the 3 stages of the algorithm there are 4 multipliers required along with the 4 add/subtract functions. The $\mathbf{G}$ and $\mathbf{H}$ matrix multipliers require 4 DSP elements each and 2 adders/subtractors due to the complex calculations required to perform the matrix multiplication.

The system is implemented on an Altera DSP development platform [6] coded in VHDL. The FPGA on the platform is a Stratix II EP2S60F672 [7]. Overall, the implementation of the system consumes 8% of the logic resources (1,745 out of 48,352 combinational look up tables. i.e. ALUT's and 3,155 out of 48,352 dedicated logic registers) and 11% (32 out of 288) of the DSP elements available on the device. The resource usage report indicates that it is possible to improve the design in order to accommodate larger length FFT blocks which may add more granularity at the output of each stage in the design at the expense of additional resources.

The system performs at a maximum frequency of 150 MHz with no setup or hold violations. The maximum operating frequency of the design may be improved upon by using the pipeline settings in the multipliers and adder blocks. This will ensure that there are clocked stages in these blocks thereby reducing the amount of combinatorial logic between clocked stages. The design uses these modules with the latency parameter set to zero.

5. SIMULATIONS

As stated in the previous section, the FFT length of the system is selected as $N = 8$. Further, the channel response matrix defined in Eq. 10 used in the simulation is selected to be

$$H = diag\left(e^{j\frac{\pi}{6}}, e^{j\frac{\pi}{3}}, e^{j\frac{\pi}{2}}, e^{j\frac{2\pi}{3}}, e^{j\frac{5\pi}{6}}, e^{j\pi}, e^{j\frac{7\pi}{6}}, e^{j\frac{4\pi}{3}}\right)$$

Note that, H, is an 8×8 diagonal matrix consisting of 8 constant modulus complex numbers. Therefore in order to ensure that $\mathbf{G} \perp \mathbf{H}$, it is enough to choose $\mathbf{G} = \mathbf{H}^*$ where superscript $*$ is the complex conjugation operator. In the implementation, the elements of $\mathbf{H}$ and $\mathbf{G}$ are converted to Cartesian form and real and imaginary parts are represented with 8 bits each. For instance, the element on the first row and the first column of matrix $\mathbf{H}$, $e^{j\frac{\pi}{6}}$, is equal to $0.866 + j0.5$ or approximately $110/128 + j\,(64/128)$. Hence, it is possible to represent these

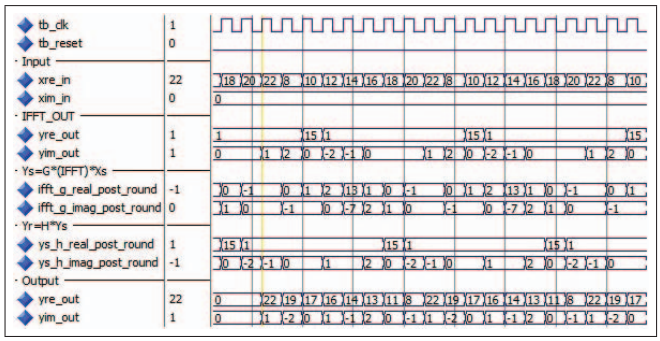

Fig. 3. Simulation results obtained with Altera ModelSim.

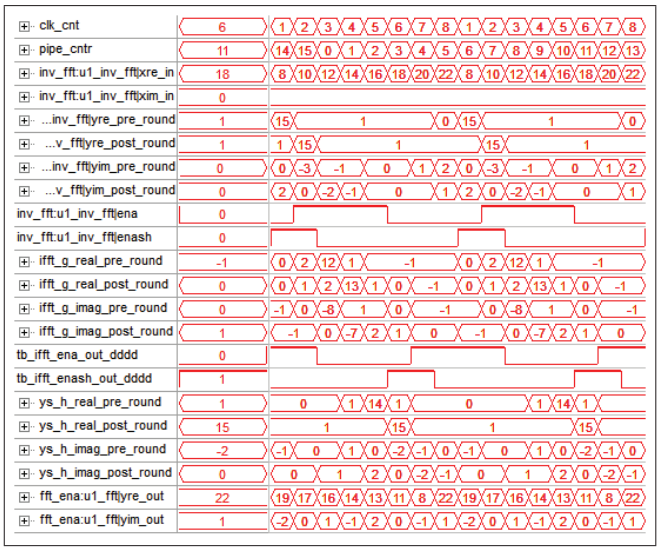

Fig. 4. Actual results running in hardware obtained with Altera SignalTap.

numbers as a fraction of 128. Therefore, the polar number, $e^{j\frac{\pi}{6}}$ is represented as $110 + j64$ in decimal or $6E + j40$ in hexadecimal base. The $\mathbf{H}$ and $\mathbf{G}$ matrices are declared as constant arrays of hexadecimal numbers of size 8×1. The design allows us to use any kind of signal such as a BPSK or M-QAM as an input. However for the test purposes the data source used for simulations is a periodic discrete ramp function where a period is defined as

$$\mathbf{x}_s = \begin{bmatrix} 22 & 20 & 18 & 16 & 14 & 12 & 10 & 8 \end{bmatrix}^{\mathrm{T}}$$

where superscript T is the transpose operator. Simulation results obtained with Altera ModelSim and actual results running in hardware obtained with Altera SignalTap (a software that allows the designer to probe signals in the FPGA and view them in a logic analyzer type display [8]) are given in Fig. 3 and Fig. 4, respectively. Examination of the input and output signals on both figures clearly shows that the simulation results match the actual results with no deviation in the signals. However, the output data vector, $\mathbf{x}_r$, is not exactly equal to the input data vector, $\mathbf{x}_s$, as given by Eq. 9, due to the several quantization errors introduced with the implementation. The number of bits to represent numbers must be increased to improve the precision with the cost of increased resource usage.

6. CONCLUSIONS AND FUTURE WORK

The implementation of GDFT with nonlinear phase on FPGA is introduced. The framework is applied to a channel equalization problem in an OFDM communication system. Implementation details of the main blocks are presented along with the necessary VHDL code snippets. The design only uses about 10% of the resources at the Altera Stratix II EP2S60F672 chip with the registered performance of 150 MHz. Employing pipelining methods and increasing the number of bits to represent numbers are proposed to boost the speed and precision of the design. Authors are planning to implement the proposed methods on top of the current design and to measure the performance for different FFT lengths as future work. Generalized DFT with nonlinear phase is anticipated to be an important component of the wireless communications systems of tomorrow. This is the first hardware implementation of GDFT, and offers engineering community a head start to utilize it as a potential improvement to DFT based technologies in various fields.

7. REFERENCES

[1] A. N. Akansu and H. Agirman-Tosun, "Generalized discrete Fourier transform with nonlinear phase," *IEEE Trans. Signal Processing*, vol. 58, no. 9, pp. 4547–4556, Sep 2010.

[2] James W. Cooley and John W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[3] Uwe Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays (Signals and Communication Technology)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.

[4] A. N. Akansu and M. U. Torun, *Graduate Laboratory Manual for ECE 641 Digital Signal Processing with Field Programmable Gate Arrays*, NJIT, 3 edition, Jan 2011.

[5] Altera, *LPM Quick Reference Guide*, Dec 1996.

[6] Altera, *Nios Development Board Stratix II Edition Reference Manual*, 1.3 edition, May 2007.

[7] Altera, *Stratix II Device Handbook, Volume 1*, May 2007.

[8] Altera, *Quartus II Handbook, Volume 3: Verification*, 10.1 edition, Dec 2010.