A CORDIC Based FFT Processor for MIMO Channel Emulator

Yanwei Xiong, Jianhua Zhang, Ping Zhang
Key Lab of Universal Wireless Communications, Ministry of Education
Beijing University of Posts and Telecommunications, Beijing, China

ABSTRACT

With the advent of Multi Input Multi Output (MIMO) systems, the system performance is highly dependent on the accurate representation of the channel condition that causes the wireless channel emulation to become increasingly important. The conventional Finite Impulse Response (FIR) based emulator has a high real-time but the complexity rapidly becomes impractical for larger array sizes. However, the frequency domain approach can avoid this problem and reduce the complexity for higher order arrays. The complexity comparison between in time domain and in frequency domain is made in this paper. The Fast Fourier Transform (FFT) as an important component of signal processing in frequency domain is briefly introduced and an FGPA system architecture based on CORDIC algorithm is proposed. The full design is implemented in Xilinx's Virtex-5.
Keywords: channel emulation, CORDIC, FFT, FPGA, MIMO

1. INTRODUCTION

The MIMO system which has multiple antennas at both the transmitter and receiver can greatly increase spectral efficiency [1]-[3] and is considered the key technology of the wireless communications. In the MIMO transmission system, system performance is highly dependent on the nature of the wireless channel [4]-[6]. Wireless channel emulation shortens the test and validation cycles by replicating channel characteristics in a controllable and repeatable laboratory environment. Thus a real-time channel emulator to emulate the realistic MIMO propagation channel is indispensable to develop products of MIMO systems.

Typically, wireless channels are emulated in time domain using Finite Impulse Response (FIR) filters. In [7] authors reported in detail an implementation of a Single Input Single Output (SISO) FIR based channel emulator with 5 MHz bandwidth. Later, other prototypes for SISO and MIMO channel emulator with wider bandwidth were reported by replicating the same architecture [8]-[10]. However, this architecture suffers from a tradeoff between the number of filter taps per sub-channel and the number of channels supported as a result of the limited computation resources.

Thus, FIR based architecture is well suited for SISO systems with short delay spread. However, the complexity scales quadratic with the array size for MIMO system. This demand for computational resources is a limitation of the FIR approach and promotes studying the other methods, one of which is frequency domain emulation [11].

The implementation of FFT processor as a component of the MIMO channel emulator in frequency domain becomes crucial and influences the emulator performance. The processor should occupy as less resources as possible as to the limited resources and have higher computing efficiency in order to emulate the true channel condition in real time.

This paper introduces in detail the system design of a CORDIC based FFT processor which is implemented using FPGA.

The system architecture for the emulator is presented in section II and the FFT algorithm is described in section III. Section IV presents the FFT processor architecture. Finally, section V concludes the paper.

2. DESIGN OF CHANNEL EMULATOR

2.1. Time domain processing

Typically, the Channel Impulse Response (CIR) of the wireless channel can be expressed as

$$h(t) = \sum_{i=1}^{L} c_i \cdot \delta(t - \tau_i) \tag{1}$$

where L represents the number of multipath, c_i is the complex coefficient associated with the i^{th} path and τ_i is the delay. Coefficients c_i are generated according to the channel conditions and are updated every coherence time [12]-[14].

The received signal can be then expressed as

International Conference on Graphic and Image Processing (ICGIP 2012),
edited by Zeng Zhu, Proc. of SPIE Vol. 8768, 876820 · © 2013 SPIE
CCC code: 0277-786X/13/$18 · doi: 10.1117/12.2010856

$$r(t) = s(t) * h(t) \tag{2}$$

where $*$ represents the convolution operation and $s(t)$ represents the transmitted signal.

According to (1), the channel emulator can be implemented by means of a FIR filter with time variant coefficients as shown in Fig.1.

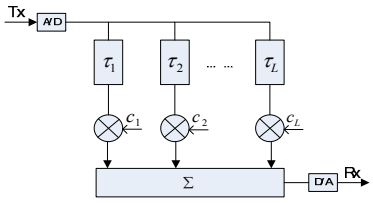

Fig.1 FIR filter with channel impulse response

In case of the M×N MIMO system, where M antennas and N antennas are deployed respectively at the transmitter and the receiver ends, there are M×N sub-channels and each sub-channel can be represented by a separate impulse response as described in (1).

Thus, a FIR filter is needed for every sub-channel and the consumption of resources will become larger with a higher order MIMO configuration.

2.2. Frequency domain processing

Equation (2) can be written in frequency domain as

$$R(f) = S(f) \cdot H(f) \tag{3}$$

In case of the M×N MIMO system, the spectrum at the n^{th} receiver antenna can be written as

$$R_n(f) = \sum_{m=1}^{M} S_m(f) \cdot H_{m,n}(f) \tag{4}$$

where $S_m(f)$ is the spectrum of the transmit data from the m^{th} transmit antenna, $H_{m,n}(f)$ is the spectrum of the sub-channel between the m^{th} transmitter and the n^{th} receiver.

In order to ensure equivalence between the linear convolution in (2) and circular convolution created by spectrum multiplication, the frequency domain processing applies the overlap-and-add method [15]. An FFT of length N_{FFT} is taken from a data vector of length N_d with the condition that: $N_{FFT} = N_d + \tau / T_s$ where τ is the delay spread in frequency domain and T_s is the sampling time. Append the N_d sample data vector by τ / T_s zeros.

In case of M×N MIMO, the system diagram of the channel emulator in frequency domain can be depicted in Fig.2.

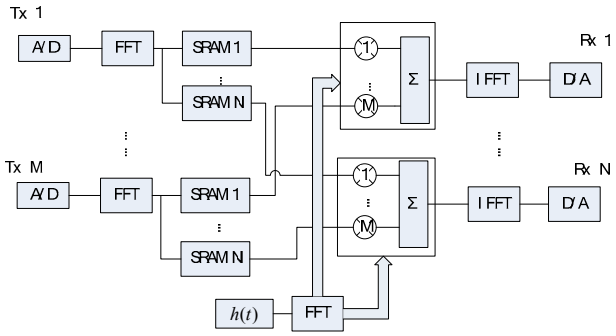

Fig.2 system diagram of frequency domain channel emulator

The frequency domain implementation exhibits initial complexity that grows at a reduced rate as a function of the array size. This will result in significant savings in complexity for higher order arrays. Furthermore, the frequency domain performing solves the classical problem of trading off the number of filter taps per sub-channel versus the number of emulated channels.

2.3. Complexity comparison

Table 1 presents a basic complexity comparison between time domain and frequency domain emulation assuming the channel with L multipath, the impulse response length of P samples and the data vector length is $2P$. IFFT is taken over P data samples to account for the overlap and add approach. The results are reported for a radix-2 FFT/IFFT implementation for frequency domain emulation and an FIR filter with L taps for time domain.

It is important to note that in term of the average complexity per output sample, both systems exhibit a linear dependency on the array dimension. However, the frequency domain processing which exhibits a fixed initial cost makes less attractive for SISO system. This complexity will be amortized over the entire array size for MIMO system and the frequency approach presents higher efficiency than the time domain processing. On the other hand, the time approach exhibits a linear cost in terms of the array size but also the number of taps supported. This is the reason why there is a trade-off between the MIMO array dimensions and the number of taps supported.

TABLE I. COMPLEXITY FOR TIME DOMAIN VS FREQUENCY DOMAIN

complexity		
	Time domain	**Frequency domain**
Add	$MN(L-1)$	$M\cdot 2P\log_2 2P+MN\cdot P\log_2 P+2P\cdot MN+N\cdot P\log_2 P$ (space multiplexing)
		$2P\log_2 2P+MN\cdot P\log_2 P+2P\cdot MN+N\cdot P\log_2 P$ (spatial diversity)
Mul	MNL	$M\cdot\frac{2P}{2}\log_2 2P+MN\cdot\frac{P}{2}\log_2 P+P\cdot MN+N\cdot\frac{P}{2}\log_2$ (space multiplexing)
		$\frac{2P}{2}\log_2 2P+MN\cdot\frac{P}{2}\log_2 P+P\cdot MN+N\cdot\frac{P}{2}\log_2 P$ (spatial diversity)
Average complexity per output sample		
	Time domain $(\frac{complexity}{N})$	**Frequency domain** $(\frac{complexity}{PN})$
Add	$M(L-1)$	$(\frac{2M}{N}+M+1)\log_2 P+2M(1+\frac{1}{N})$ (space multiplexing)
		$(\frac{2}{N}+M+1)\log_2 P+2(M+\frac{1}{N})$ (spatial diversity)
Mul	ML	$(\frac{M}{N}+\frac{M}{2}+\frac{1}{2})\log_2 P+M(1+\frac{1}{N})$ (space multiplexing)
		$(\frac{1}{N}+\frac{M}{2}+\frac{1}{2})\log_2 P+(M+\frac{1}{N})$ (spatial diversity)
$MN\cdot P\log_2 P$ and $MN\cdot\frac{P}{2}\log_2 P$ are the number of addition and multiplication for the FFT processing of CIR which is calculated once in coherence time. The complexities are different when different MIMO technologies are adopted.		

3. THE FFT ALGORITHM

The essence of Discrete Fourier Transform (DFT) is limited discrete sampling of the limited long sequence Fourier transformation. DFT introduces the method of discretization in the frequency domain that makes the digital signal processing adopt digital operation methods and greatly increases the flexibility of digital signal processing.

The N-points DFT $X(k)$ for a given sequence $x(k)$ is defined as

$$X(k)=\sum_{n=0}^{N-1}x(k)\cdot W_N^{nk} \quad (5)$$

where $W_N^{nk}=\exp(-j\frac{2\pi}{N}kn)$, $k=0,1,2,\ldots,N-1$.

The FFT is based on the decomposition of a sequence of the DFT into lower order computations that results in a reduction in the number of operations. The FFT algorithm increases the operation efficiency by 1~2 orders of magnitude and creates conditions for signal real-time processing.

There are two basic approaches to implement the FFT algorithm: decimation in time (DIT) or decimation in frequency (DIF). The difference between them is the way to decompose the DFT into lower order DFTs, resulting in a different sequence of operations.

In case of a DIF implementation with radix-2, the resulting decomposition is achieved

$$\begin{aligned} X(2r) &= \sum_{n=0}^{N/2-1}[x(n)+x(n+N/2)]W_N^{2rn} \\ &= \sum_{n=0}^{N/2-1}[x(n)+x(n+N/2)]W_{N/2}^{rn} \end{aligned} \quad (6)$$

with $r=0,1,\ldots\ldots,N/2$;

$$\begin{aligned} X(2r+1) &= \sum_{n=0}^{N/2-1}[x(n)-x(n+N/2)]W_N^{n(2r+1)} \\ &= \sum_{n=0}^{N/2-1}[x(n)-x(n+N/2)]W_N^{n}\cdot W_{N/2}^{nr} \end{aligned} \quad (7)$$

with $r=0,1,\ldots\ldots,N/2$.

In this paper, a radix-2 FFT implementation based DIF is chosen because it has advantages in terms of regularity of hardware, ease of computation and number of processing elements.

4. THE ARCHITECTURE FOR FFT PROCESSOR

There are many implementations for FFT algorithms: digit-serial or bit-parallel arithmetic, pipelined or iterative implementations. Among these different methods, there are two methods can be seen as two extremes: the first one uses only a single butterfly arithmetic unit and one memory unit. This method leads to a small hardware resources but has a long computation time to execute the transform. The second one completes the entire FFT structure composed of $\log_2 N$ and $N/2$ arithmetic unit each stage, where N is the transform length. This method leads to a very high speed computation at a very large and expensive area.

In order to reflect the actual channel condition, the emulator must have a good real-time performance. Thus the speed of the FFT processer becomes the main requirement though there is a trade-off between speed and area. The implementations that fit these requirements are bit-parallel and pipelined architectures, where the processing in several cascaded stages, as depicted in Fig.3.

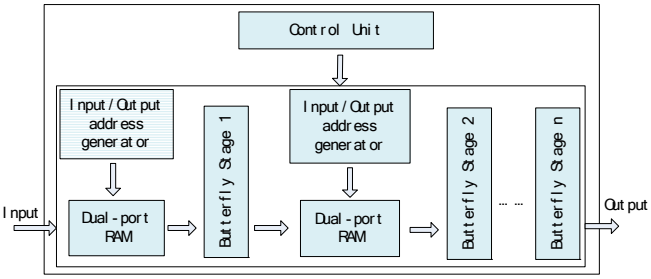

Fig.3 the architecture for FFT processor

As shown in Fig.3, the architecture for FFT is based on a set of stages and each stage performs the decomposition of the input sequence into sub-DFTs which are implemented in later stage. Every stage needs to process an input sequence of length N .

4.1. Control Unit

Before starting work, the control unit resets the overall FFT system. Each butterfly computation has direct data dependency, thus the control unit also needs to coordinate data synchronization in respective butterfly stage. In detail, when the FFT processor begins to work, the first level RAM and the butterfly stage 1 are opened, while the other stages are closed. When the operation of current stage is completed, the later stage is opened and starts to operate. After all stages are opened, the FFT processor enters into the normal work mode and does not need too much control.

4.2. Dual-port RAM

In order to make data computing and storing operation simultaneously, dual-port RAM is introduced. It has two separate sets of data bus, address bus and control bus, equipping FFT operation with time-sharing model of ping-pong type. That is, the dual-port RAM is divided into two parts, when the first part outputs data are computed in current stage, the other part stores the last operation results. The two parts exchange their computing and storing operation after the operation of the current stage is complemented. Suitable for real-time data cache, the dual-port RAM is used here to improve the RAM throughput rate.

4.3. Address Generator

The address generator is to provide the dual-port RAM with the correct input/output addresses. The RAM addresses of each butterfly data storage stage are generated by a virtual counter, which is initialized according to the control signals provided by the control unit.

4.4. Butterfly stage

The core operation of FFT is butterfly unit which directly affects the speed of FFT processer. Therefore how to optimize the butterfly unit is the key point to increase the speed of FFT calculation.

4.4.1 Butterfly computation unit

The design of butterfly unit will be described in detail in the following.

For radix-2 FFT by DIF, the butterfly unit is depicted in Fig.4, having the following form:

$$\begin{cases} C = A + B \\ D = (A - B)W_N^n \end{cases} . \quad (8)$$

where W_N^n is the twiddle factor.

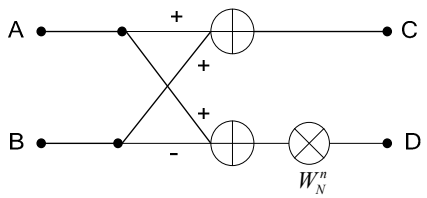

Fig.4 basic two-point butterfly operation

Equation (8) may be written in its expanded form in terms of the real and the imaginary parts as follows:

$$\begin{cases} C_{re} = A_{re} + B_{re} \\ C_{im} = A_{im} + B_{im} \\ D_{re} = (A_{re} - B_{re})\cos(2\pi n / N) + (A_{im} - B_{im})\sin(2\pi n / N) \\ D_{im} = -(A_{re} - B_{re})\sin(2\pi n / N) + (A_{im} - B_{im})\cos(2\pi n / N) \end{cases} \tag{9}$$

The last two in set of (9) essentially represent a plane rotation operation which can be efficiently computed by applying the CORDIC algorithm [16].

4.4.2 CORDIC algorithm

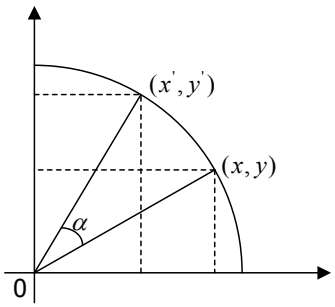

Fig.5 the plane rotation

In the CORDIC technique, the plane rotation through angle α is depicted in Fig.5, where (x, y) is the projections of the original vector in the Cartesian coordinate system and (x', y') is the rotated version. The relationship between them is as

$$\begin{cases} x' = x\cos(\alpha) - y\sin(\alpha) \\ y' = y\cos(\alpha) + x\sin(\alpha) \end{cases}. \tag{10}$$

Equation (10) can be rearranged as

$$\begin{cases} x' = \cos(\alpha)[x - y\tan(\alpha)] \\ y' = \cos(\alpha)[y + x\tan(\alpha)] \end{cases}. \tag{11}$$

Equation (11) can be further simplified by representing $\tan(\alpha) = 2^{-i}$ and dropping the cosine term as

$$\begin{cases} x' = x - y2^{-i} \\ y' = y + x2^{-i} \end{cases} \tag{12}$$

Form (12) we can find that the multiplication by the tangent term is reduced to a simple shift operation.

Many iteration are made in order to get certain angle and (13) can be gotten after one iteration

$$\begin{cases} x_{i+1} = x_i - d_i(2^{-i} y) \\ y_{i+1} = y_i + d_i(2^{-i} x) \end{cases}. \tag{13}$$

The third equation which is called angle accumulator tracking the accumulation of rotating angle is introduced as

$$z_{i+1} = z_i - d_i\alpha_i, \tag{14}$$

where $d_i = \pm 1$ represents the rotation direction and α_i is the rotating angle in the i^{th} iteration. Arbitrary angle of rotation can be completed through a series of smaller elementary angle rotations with n iterations. The output results are

stretched by K_n times because cosine is ignored, where $K_n = \prod 1/\cos\alpha_i$. The scaling factor K_n approaches asymptotically a constant value of $1/0.60725$ when $n \to \infty$.

After n iterations, (14) can be gotten

$$\begin{cases} x_n = K_n[x\cos(\alpha) - y\sin(\alpha)] \\ y_n = K_n[y\cos(\alpha) + x\sin(\alpha)] \end{cases}. \tag{14}$$

As is shown, the last two in set of (9) which are similar with (10) can be calculated with the CORDIC algorithm.

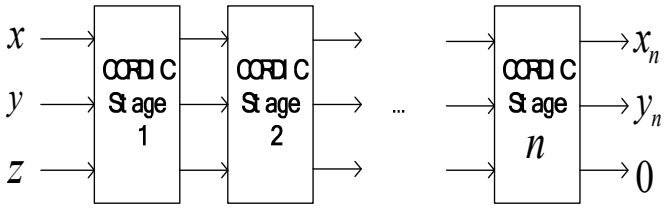

Fig.6 pipelined CORDIC architecture

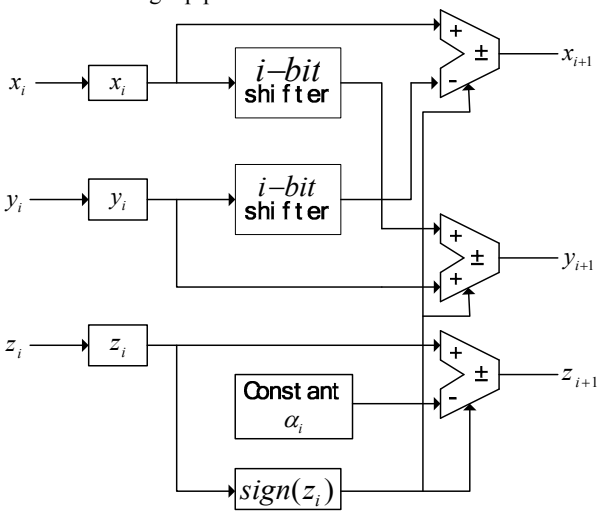

Fig.7 the i^{th} stage of the pipelined CORDIC unit

The design of CORDIC adopts pipelined implementation which saves the computation time and is depicted in Fig.6. Fig.7 describes the i^{th} stage of the pipelined CORDIC unit in detail.

To validate correctness of CORDIC on this architecture design, a simulation is performed. Fig.8 shows the time simulation results. Input x , y , z and get the output x_out , y_out after iterations.

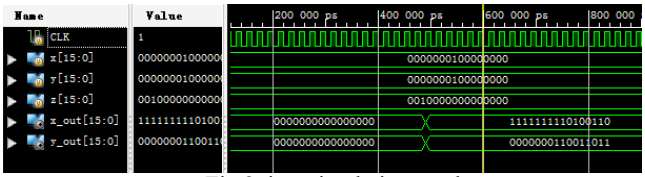

Fig.8 time simulation results

5. CONCLUSION

In this paper, we present a system framework of FFT processer for the MIMO channel emulator and introduce the design of each part in detail. To balance the area complexity and computation time, the parallel pipelined architecture and the radix-2 butterfly computation method of the FFT algorithm targeting FPGA devices is adopted.

The proposed processor will complete the computations of 1024-point complex FFT in average time of 20 μs at operating frequency of 100MHz. This technology is also strongly connected to other research fields and provides foundation.

REFERENCES

[1] G. J. Foschini andM. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," Wireless Personal Commun., pp. 311–335, 1998.

[2] I. E. Telatar, "Capacity ofmulti-antennaGaussian channels,"AT&T Bell Labs., 1995.

[3] Ke Liu,Vasanthan Reghavan and Akbar M.Sayeed,"Capacity Scaling and Spectral Efficiency in Wide-Band Correlated MIMO Channels,"IEEE Trans. Inform. Theory, VOL.49,pp.2504-2526, Oct 2003.

[4] A. F. Molisch, M. Steinbauer, M. Toeltsch, E. Bonek, and R. S. Thomä,"Capacity of MIMO systems based on measured wireless channels,"IEEE J. Select. Areas Commun., vol. 20, pp. 561–569, Apr. 2002.

[5] G. J. Foschini and R. A. Valenzuela, "Initial estimation of communication efficiency of indoor wireless channels," Wireless Networks, vol. 3,no. 2, pp. 141–154, 1997.

[6] A. J. Paulraj and C. B. Papadias, "Space-time processing for wireless communications," IEEE Signal Processing Mag., vol. 14, pp. 49–83,Nov. 1997.

[7] J.J.Olmos, A.Gelonch, F.J.Casadevall, and G.Fermenias, "Design and implementation of a wide-band real-time wireless channel emulator," IEEE Trans. on Vehicular Technology, Vol. 48, no.3, pp.746-764,May 1999.

[8] A.Dassatti, G. Masera, M. Nicola, A. Concil and A. Poloni, "High performance channel model hardware emulator for 802.11n," Proc. Of Int. Conf.on Field-Programmable Technology, pp. 303-304, Dec. 2005.

[9] D. N. Dung, et al., "Implementation and evaluation of 4x4 MIMO fading simulator considering antena characteristics," Int. Conf. on Comm. And Electronics, pp. 472-477, Oct. 2006.

[10] M. Cui, M. Hidekazu and A. Kiyomichi, "FPGA implementation of 4x4 MIMO test-bed for spatial multiplexing systems," Proc. of Personal, Indoor and Mobile Radio Comm., Vol. 4, PP. 3045-3048, Sept. 2004.

[11] H.Eslami, S.V.Tran, and A.M.Elatwil, "Design and implementation of a Scalable Channel Emulator for Wideband MIMO Systems," IEEE Trans. On Vehicular Technology, Vol.58, no.9, pp.4698-4709, Nov.2009

[12] A.A.Saleh and R.A.Valenzuela, "A statistical model for indoor multipath propagation," IEEE Journal on Slected Areas in Comm., vol. SAC-5, no.2, pp. 128-137, Feb. 1987.

[13] L. Schumacher, K. I. Pedersen and P. E. Mogensen, "From antenna spacing to theoretical capacities – guidances for simulation MIMO systems," Proc. Of Personal, Indoor and Mobile Radio Comm., pp. 587-592, Sept. 2002.

[14] V. Erceg, L. Schumacher, et al. "TGn channel models," IEEE 802.11- 03/940r4, May 2004.

[15] A. V. Oppenheim, R. W. Schafer and J. R. Buck, "Discrete-time signal processing," second edition, Prentice Hall, p. 623 1999.

[16] Y. Hu, "Cordic-based VLSI architecture for digital signal processing," IEEE Signal Processing Mag., pp. 16-35, 1992