Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 2, Number 10; April-June, 2015 pp. 808-812
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

128 Points Low Area and Highly Pipelined FFT(Fast Fourier Transform) Processor

Monica Vats[1] and Nidhi Singh[2]

[1]M.tech VLSI Design Department of ECE SRM University Delhi-Ncr
[2]Department of ECE SRM University Delhi-Ncr
E-mail: [1]moni.27m91@gmail.com

Abstract—*This paper represents low power and high speed 128-point pipelined Fast Fourier Transform (FFT) processor. The 128-point architecture consists of an optimized pipeline implementation processor . In the processor 128 -point DFT is divided into two smaller 8- and 16-point DFTs. The 8- and 16 -point DFTs are implemented by the Winograd small point FFT algorithms. The FFT128 processor has the minimum multiplier number which is equal to 4. These facts makes this FFT Processor attractive to implement in ASIC for used in OFDM modems, software defined radio, multichannel coding, and wideband Spectrum analysis.*

1. INTRODUCTION

The FFT (Fast Fourier Transform) and its inverse (IFFT) are the key components of OFDM (Orthogonal Frequency Division Multiplexing) systems. Recently, the demand for long length, high-speed and low-power FFT has increased in the OFDM applications.

There are three kinds of main design architectures for implementing a FFT processor. One is the single-memory architecture. It has one processing element and one main memory. Hence, it occupies a small area. The second is the dual memory architecture, which has two memories. This architecture has a higher throughput than the single-memory architecture because it can store butterfly outputs and read butterfly inputs at the same time. The fast Fourier transform plays an important role in many digital signal processing (DSP) systems.

Recent advances in semiconductor processing technology have enabled the deployment of dedicated FFT processors in applications such as telecommunications, speech and image processing. Specifically, in the OFDM communication systems, FFT and inverse FFT (IFFT) play a very important role. The OFDM technique, due to its effectiveness in overcoming adverse channel effects [1, 2] as well as spectrum utilization, has become widely adopted in wire line and wireless communication standards.

The OFDM technique has been adopted in several standards like digital audio broadcasting (DAB) [3], digital video broadcasting terrestrial (DVB-T) [4], asymmetrical digital subscriber line (ADSL) [5] and very-high-speed digital subscriber line (VDSL) [6]. Therefore, efficient and low-power VLSI implementation of FFT processors is essential for successful deployment of these OFDM-based systems. According to the standards of DAB, DVB-T, ADSL and VDSL, various FFT sizes are required.

In the proposed processor 128 -point DFT is divided into two smaller 8- and16-point DFTs. The 8- and 16 -point DFTs are implemented by the Winograd small point FFT algorithms. This algorithm performs the convolution with minimum number of multiplications and additions and thus the computational complexity of the process is greatly reduced. As a result, The FFT128 processor has the minimum multiplier number which is equal to 4.

FFT8 and FFT16 calculations are implements in highly pipelined architecture.The 8-point and 16-point DFT algorithm is divided into several stages which enables pipeline implementation in eachstage. Therefore in each clock cycle one complex number is read from the input data buffer RAM and the complex result is written in the output buffer RAM.

These facts makes this FFT Processor attractive to implement in ASIC for used in OFDM modems, software defined radio, multichannel coding, and wideband Spectrum analysis.

The discrete Fourier transform (DFT) is an important algorithm in the field of digital signal processing. It transforms a signal from the time domain into the frequency domain, providing information about the spectrum of the signal. DFT is the decomposition of a sampled signal in terms of sinusoidal (complex exponential) components. The symmetry and periodicity properties of the DFT are exploited to significantly lower its computational requirements

The direct computation of an N-point DFT requires to calculate a number of operations proportional to N^2. In order to reduce the number of arithmetic operations, many fast algorithms have been proposed. These algorithms are based on decomposing an N-point DFT recursively into smaller DFTs, leading to a reduction of the computational complexity which lead to lesser hardware. The resulting algorithms are known as Fast Fourier Transforms (FFTs).

An 128-point DFT computes a sequence x(n) of 128 complex-valued numbers given another sequence of data X(k) of length 128 according to the formula

$$X(k) = \sum_{n=0}^{127} x(n) e^{-j2\pi nk/128} \quad ; \quad k = 0 \text{ to } 127.$$

To simplify the notation, the complex-valued phase factor e –j2 nk/128 is usually defined as W128n where: W128 = cos(2π /128) – j sin(2π /128). The FFT algorithms take advantage of the symmetry and periodicity properties of W128n to greatly reduce the number of calculations that the DFT requires. In an FFT implementation the real and imaginary components of WnN are called twiddle factors.

The basis of the FFT is that a DFT can be divided into smaller DFTs. In the processorFFT128 a mixed radix 8 and 16 FFT algorithm is used. It divides DFT into two smaller DFTs of the length 8 and 16, as it is shown in the formula:

$$X(k) = X(16r+s) = \sum_{m=0}^{15} W_{16}{}^{mr} W_{128}{}^{ms} \sum_{l=0}^{7} x(16l+m) W_8{}^{sl} \quad , r = 0 \text{ to } 15, s = 0 \text{ to } 7,$$

which shows that 128 -point DFT is divided into two smaller 8- and16-point DFTs. The input complex data x(n) are represented by the 2-dimensional array of data x(16l+m). The columns of this array are computed by 8-point DFTs. The results of them are multiplied by the twiddle factors W128ms . And the resulting array of data X(16r+s) is derived by 16-point DFTs of rows of the intermediate result array.

The 8- and 16 -point DFTs are implemented by the Winograd small point FFT algorithms, which provide the minimum additions and multiplications. As a result, the radix-16 FFT algorithm needs only 128 complex multiplications to the twiddle factors W128ms and a set of multiplications to the twiddle factors W16sl except of 32768 complex multiplications in the origin DFT.

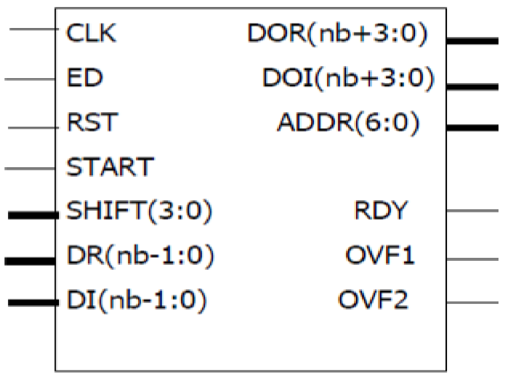

Fig. 1: FFT 128

Table 1: Signal Description

SIGNAL	TYPE	DESCRIPTION
CLK	input	Global clock
RST	input	Global reset
START	input	FFT start
ED	input	Input data and operation enable strobe
DR [nb-1:0]	input	Input data real sample
DI [nb-1:0]	input	Input data imaginary sample
SHIFT	input	Shift left code
RDY	output	Result ready strobe
ADDR [6:0]	output	Result number or address
DOR [nb+3:0]	output	Output data real sample
DOI [nb+3:0]	output	Output data imaginary sample
OVF1	output	Overflow flag
OVF2	output	Overflow flag

2. FFT 128

It performs one dimensional 128 – complex point FFT. The data and coefficient widths are adjustable in the range 8 to 16.

Features

- 128 -point radix-8 FFT.
- Forward and inverse FFT.
- Pipelined mode operation, each result is outputted in one clock cycle, simultaneous loading/downloading supported.
- Input data, output data, and coefficient widths are parametrizable in range 8 to 16 and more.
- Two and three data buffers are selected.
- Overflow detectors of intermediate and resulting data are present.

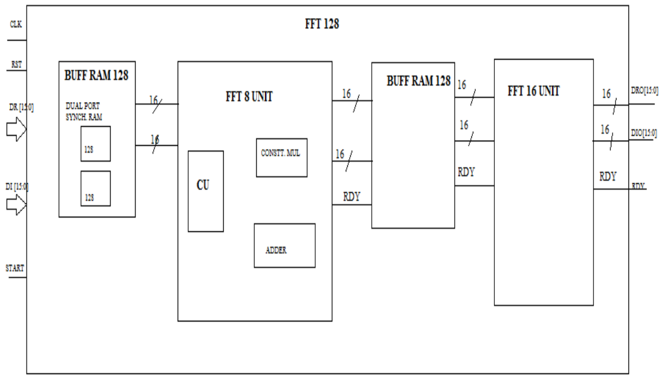

Fig. 2: FFT 128 Top Module

2.1 BUFRAM128

BUFRAM128 is the data buffer, which consists of the two port synchronous RAM of the volume 512 complex data, and the write-read address counter. The real and imaginary partsof the data are stored in the natural ascending order as in the diagram in the Fig. . By the START impulse the address counter is reset and then starts to count (signal addrw). The input data DR and DI are stored to the respective address place by the rising edge of the clock signal.

After writing 128 data beginning at the START signal, the unit outputs the ready signal RDY and starts to write the next 128 data to the second half of the memory. At this period of time it outputs the data stored in the first half of the memory. When this data reading is finished then the reading of the next array is starting. This process is continued until the next START signal or RST signal are entered. The reading address sequence is 8-6-th inverse order, i.e. the order is 0,16,32,...240,1,17,33, ... Really the reading address is derived from the writing address by swapping 4 LSB and 4 MSB address bits.

2.2 FFT16

The datapath FFT16 implements the 16-point FFT algorithm in the pipelined mode. 16 input complex data are calculated for 46 clock cycles, but each new 16 complex results are outputted each 16 clock cycles. The FFT algorithm of this transform is selected from the book "H.J.Nussbaumer.

$t1 := x(0) + x(8)$; $m4 := x(0) - x(8)$;

$t2 := x(4) + x(12)$; $m12 := -j*(x(4)-x(12))$;

$t3 := x(2) + x(10)$; $t4 := x(2) - x(10)$;

$t5 := x(6) + x(14)$; $t6 := x(6) - x(14)$;

$t7 := x(1) + x(9)$; $t8 := x(1) - x(9)$;

$t9 := x(3) + x(11)$; $t10 := x(3) - x(11)$;

$t11 := x(5) + x(13)$; $t12 := x(5) - x(13)$;

$t13 := x(7) + x(15)$; $t14 := x(7) - x(15)$;

$t15 := t1 + t2$; $m3 := t1 - t2$;

$t16 := t3 + t5$; $m11 := -j*(t3 - t5)$;

$t17 := t15 + t16$; $m2 := t15 - t16$;

$t18 := t7 + t11$; $t19 := t7 - t11$;

$t20 := t9 + t13$; $t21 := t9 - t13$;

$t22 := t18 + t20$; $m10 := -j*(t18 - t20)$;

$t23 := t8 + t14$; $t24 := t8 - t14$;

$t25 := t12 + t10$; $t26 := t12 - t10$;

$m0 := t17 + t22$; $m1 := t17 - t22$;

$m13 := -j* \sin(p/4)*(t19 + t21)$;

$m5 := \cos(p/4)*(t19 - t21)$;

$m6 := \cos(p/4)*(t4 - t6)$;

$m14 := -j* \sin(p/4)*(t4 + t6)$;

$m7 := \cos(3p/8)*(m24+m26)$;

$m15 := -j* \sin(3p/8)*(t23 + t25)$;

$m8 := (\cos(p/8) + \cos(3p/8))*t24$;

$m16 := -j* (\sin(p/8) - \sin(3p/8))*t23$;

$m9 := -(\cos(p/8) - \cos(3p/8))*t26$;

$m17 := -j*(\sin(p/8) + \sin(3p/8))*t25$;

$s7 := m8 - m7$; $s15 := m15 - m16$;

$s8 := m9 - m7$; $s16 := m15 - m17$;

$s1 := m3 + m5$; $s2 := m3 - m5$;

$s3 := m13 + m11$; $s4 := m13 - m11$;

$s5 := m4 + m6$; $s6 := m4 - m6$;

$s9 := s5 + s7$; $s10 := s5 - s7$;

$s11 := s6 + s8$; $s12 := s6 - s8$;

$s13 := m12 + m14$; $s14 := m12 - m14$;

$s17 := s13 + s15$; $s18 := s13 - s15$;

$s19 := s14 + s16$;

$y(0) := m0$;

$y(1) := s9 + s17$;

$y(2) := s1 + s3$;

$y(3) := s12 - s20$;

$y(4) := m2 + m10$;

$y(5) := s11 + s19$;

$y(6) := s2 + s4$;

$y(7) := s10 - s18$;

$s20 := s14 - s16$;

$y(8) := m1$;

$y(15) := s9 - s17$;

$y(14) := s1 - s3$;

$y(13) := s12 + s20$;

$y(12) := m2 - m10$;

$y(11) := s11 - s19$;

$y(10) := s2 - s4$;

$y(9) := s10 + s18$;

where x and y are input and output arrays of the complex data, $t1,\ldots,t26$, $m1,\ldots, m17, s1,\ldots,s20$ are the intermediate complex results, j = v(-1). As we see the algorithm contains only 20 real multiplications to the untrivial coefficients sin(p/4) = 0.7071; sin(3p/8) = 0.9239; cos(3p/8) = 0.3827; (cos(p/8) + cos(3p/8)) =1.3066; (sin(p/8) – sin(3p/8)) = 0.5412; and 156 real additions and subtractions.

The counter ct counts the working clock cycles from 0 to 15. So a single inferred adder adds $x(0) + x(8)$ in one cycle, $x(1) + x(9)$ in the next cycle, $D(1) + D(5)$ in another cycle and so on, and $x(7) + x(15)$ in the final cycle of the sequence of cycles deriving the results $t1, t7, t9, \ldots, t13$ respectively.

Four constant multipliers are used to derive the multiplication to 5 different coefficients. So the unit in MPUC707.v implements the multiplication to the coefficient 0.7071 in the pipelined manner. Note that the unit MPUC924_383.v implements the multiplication both to 0.9239 and to 0.3827. The multipliers use the adder tree, which adds the multiplicand shifted to different bit numbers. For example, for short input bit width the coefficient , for long input bit width it is approximated as 0.10110101000000101 2 0.7071 is approximated as 0.10110101 2 . The long coefficient bit width is set by the parameter FFT128bitwidth_coef_high. The first kind of the constant multiplier occupies 3 adders, and the second one occupies 4 adders.

The importance of the long coefficient selection is seen from the following fact. When the input bit width is 16 and higher, the selection of the long coefficient bit width decreases the FFT128 result error in two times.

The FFT16 unit implements both FFT and inverse FFT depending on the parameter FFT128paramifft. Practically the inverse FFT is implemented on the base of the direct FFT by the inversion of operations in the final stage of computations for all the results except $y(0)$, $y(8)$. For example, $y(1):=s9 + s17$; is substituted to $y(1):=s9 – s17$;

The FFT16 unit starts its operation by the START impulse. The first result is preceded by the RDY impulse which is delayed from the START impulse to 30 clock impulses. The output results have the bit width which is in 4 higher than the input data bit width. That means that all the calculations except multiplication by coefficients like 0.7071 are implemented without truncations, and therefore, the FFT128 results have the minimized errors comparing to other FFT processors.

2.3 FFT8

The datapath FFT8 implements the 8-point FFT algorithm in the pipelined mode. 8 input complex data are calculated for 22 clock cycles, but each new 8 complex results are outputted each 8 clock cycles. The FFT algorithm of this transform is selected from the book "H.J.Nussbaumer. FFT and convolution algorithms". Due to this algorithm the calculations are:

$t1=D(0) + D(4); m3=D(0) - D(4);$

$t2=D(6) + D(2); m6=j*(D(6)-D(2));$

$t3=D(1) + D(5); t4=D(1) - D(5);$

$t5=D(3) + D(7); t6=D(3) - D(7);$

$t8=t5 + t3; m5=j*(t5-t3);$

$t7=t1 + t2; m2=t1 - t2;$

$m0=t7 + t8; m1=t7 - t8;$

$m4=\sin(p/4)*(t4 - t6);$

$m7=-j* \sin(p/4)*(t4 + t6);$

$s1=m3 + m4; s2=m3 - m4;$

$s3=m6 + m7; s4=m6 - m7;$

$DO(0)=m0; DO(4)=m1;$

$DO(1)=s1 + s3; DO(7)=s1 - s3;$

$DO(2)=m2 + m5; DO(6)=m2 - m5;$

$DO(5)=s2 + s4; DO(3)=s2 - s4;$

where *DI* and *DO* are input and output arrays of the complex data, $j = v(-1)$, $t1,\ldots,t8$, $m1,\ldots, m7$, $s1,\ldots,s4$ are the intermediate complex results. As we see the algorithm contains only 4 multiplications to the untrivial coefficient sin(p/4) = 0.7071, and 26*2 real additions and subtractions. The multiplication to a coefficient *j* means the negation the imaginary part and swapping real and imaginary parts.

The FFT8 unit starts its operation by the START impulse. The first result is preceded by the RDY impulse which is delayed from the START impulse to 17 clock impulses.

2.4 CNORM

During computations in FFT8 and FFT16 the data magnitude increases up to 8 and 16 times, respectively, and the FFT128 result can increase up to 128 times depending on the spectrum properties of the input signal. Therefore, to prevent the signal dynamic bandwidth loose, the output signal bit width must be at least in 8 bits higher than the input signal bit width. To prevent this bit width increase, to provide the proper signal dynamic bandwidth, and to ease the next computation of the derived spectrum, the CNORM units are attached to the outputs of the FFT16 units.

CNORM unit provides the data shift left to 0,1,2, and 3 bits depending on the code SHIFT. The input data width is nb+3 and the output data width is nb+2, where nb is the given processor input bit width. The overflow occurs in CNORM unit when the SHIFT code is given too high. The SHIFT code must be set by the customer to prevent the data overflow and to provide the proper dynamic bandwidth. The CNORM unit contains the overflow detector with the output OVF. When FFT128 core in operation, a 1 at the output OVF signals that for some input data an overflow occurred. OVF flag is resetted by the RST or START signal.

The SHIFT inputs of two CNORM stages are concatenated to the 4-bit input SHIFT of the FFT128 core, 2 LSB bits control the first stage, and 2 MSB bits do the second stage.

The selection of the proper SHIFT code depends on the spectrum property of the input signal. When the input signal is the sinusoidal one or contains a few of sinusoids, and the noise level is small then SHIFT =0000, or 0001, or 0010. When the input signal is a noisy signal then SHIFT can be 1100 and higher. When the input signal has the stable statistic properties then the code SHIFT can be set as a constant. Then the OVF outputs can be not in use, and the CNORM units will be removed from the project by the hardware optimization when the core is synthesized.

2.5 Rotator 128

The unit ROTATOR implements the complex vector rotating to the angles *W* 128 *ms* . The complex twiddle factors are stored in the unit WROM128. Here the ROM contains the following table of coefficients

(w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w0, w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w0, w3, w6, w9, w12,w15,w18,w21, w24, w27, w30, w33, w36, w39, w42,w45,…w0,w7,w15,w23,w31,w39,w47,w55,w63,w71,w79,w97,w103,w111,w119,w127), where wi = W 128 I .

Here the row and column indexes are m and s respectively. These coefficients are read in the natural order addressing by the 7-bit counter addrw. The complex vector rotating is implemented by the usual schema of the complex number multiplier which contains 4 multiply units and 2 adders.

3. INPUT AND OUTPUT WAVEFORM

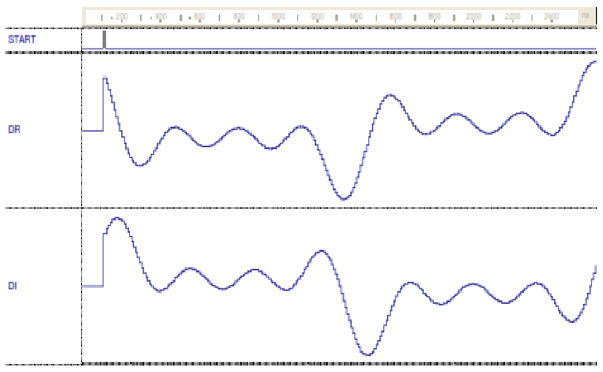

Fig 3: Input Waveform

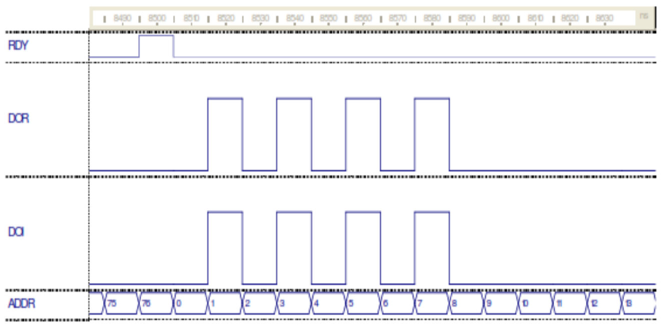

Fig. 4: Output Waveform

4. CONCLUSION AND FUTURE WORK:

The FFT Processor converts the time domain signal into frequency domain signal. This 128 point FFT Processor is made up of FFT 8 and FFT 16 using Winograd algorithm. Here we have designed the FFT processor such that the total number of multiplier used are 4 (complex multiplier). Further improvement can be made using different algorithm for FFT.

REFERENCES

[1] W.Li, L.Wanhammar, "Complex multiplication reduction in FFT processor," SSoCC'02, Falkenberg, Sweden, Mar. 2002.

[2] Weidong Li, "Studies on implementation of lower power FFT processors," Linköping Studies in Science and Technology, Thesis No.1030, ISBN 91-7373-692-9, Linköping, Sweden, Jun. 2003

[3] P.Verma, H. Kaur, M.Singh, M, B.Singh, " VHDL Implementation of FFT/IFFT Blocks for OFDM," In Proc. of Intern. Conf. on Advances in Recent Technologies in Communication and Computing, pp. 186-188, PI. 978-1-4244-5104-3, Kerala, 2009.

[4] S. Minhyeok, L. Hanho, A High-Speed FourParallel Radix-24 FFT/IFFT Processor for UWB Applications, in Proc. IEEE Int. Symp. Circuits and Systems, 2008, pp. 960-963.

[5] M.Arioua, S.Belkouch, M.M.Hassani, "Complex multiplication reduction in pipeline FFT architecture," In Proc. of 20th Intern. Conf. on Computer Theory and Applications (ICCTA), Alexandria, Egypt, Oct. 2010.