FPGA-Based Area-Efficient Specific Radar Signal Processor

He Chen Ming Liu Yueqiu Han

Department of Electronic Engineering, Beijing Institute of Technology

Abstract

This paper outlines the design techniques that have been applied to the development of a flexible processor which is used to perform high speed, real time signal processing applicable to one measuring radar. The core structure of this processor is one modified 512-point FFT which can also perform other signal processing operations by design reuse technology. The design applies the method of pipeline and block floating algorithm. It not only improves the processing speed but also precision.

Keywords: Radar signal processor, FPGA, FFT, hardware sharing

1. Introduction

Recent advances in chip technology have increased field programmable gate array (FPGA) resources and, as result, the computation of complex algorithms, such as the FFT, can be implemented on a programmable device [1].

In this paper we give an implementation method for a high performance specific radar signal processor which can converts three frames of complex data from 6 channel ADCs into their sum of energy spectra. Five kinds of operation are performed in sequence: subtracting direct current component, multiplying window coefficients, 512-point FFT, computing complex modulus and the sum of the three frame complex data's modulus square. The CFAR arithmetic in frequent field is used to detect the working state of receiving channels. This signal processing system is implemented in one FPGA chip.

The core of this system is the FFT processor based on radix-2 butterfly algorithm and recursion structure. Many paper discuss the design of FFT[2,4,5,6]. Compared with those designs, in this paper the hardware sharing technology is used in chip design. Three operations of processor share one FFT's butterfly unit. So area of processor reduces greatly.

Block floating algorithm approves the design result's precision. The pipeline architecture of processor increases the chip's operation speed and makes the chip satisfy radar's requirement of real time operation. This processor can calculate the three frames of 512-point complex signal's sum of power spectra in 224μs.

2. Design of Processor

2.1. System Overview

Fig 1 shows the whole flow of this processor. Firstly processor reads one frame 512-point complex data from external dual-port ram to internal ram1 after processor is enabled. Then subtracting DC component and multiplying window function are operated, and the results are stored in internal ram2 while processor enables to read next frame data from external dual-port ram. After multiplying window function FFT operation begins. The interim results of FFT operation are stored in internal ram3, and final results are stored in internal ram4. Then modulus square unit is enabled to compute modulus square which results are stored in internal ram5. The internal ram6 stores sum of three frames modulus square which wait for read of external DSP. At the same time CFAR arithmetic is used to detect the working state of receiving channels in order to estimate whether phase reference signal of each channel is received correctly or not.

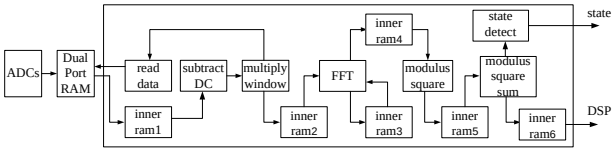

Fig. 1: Work flow of processor

2.2. Processor Design Overview

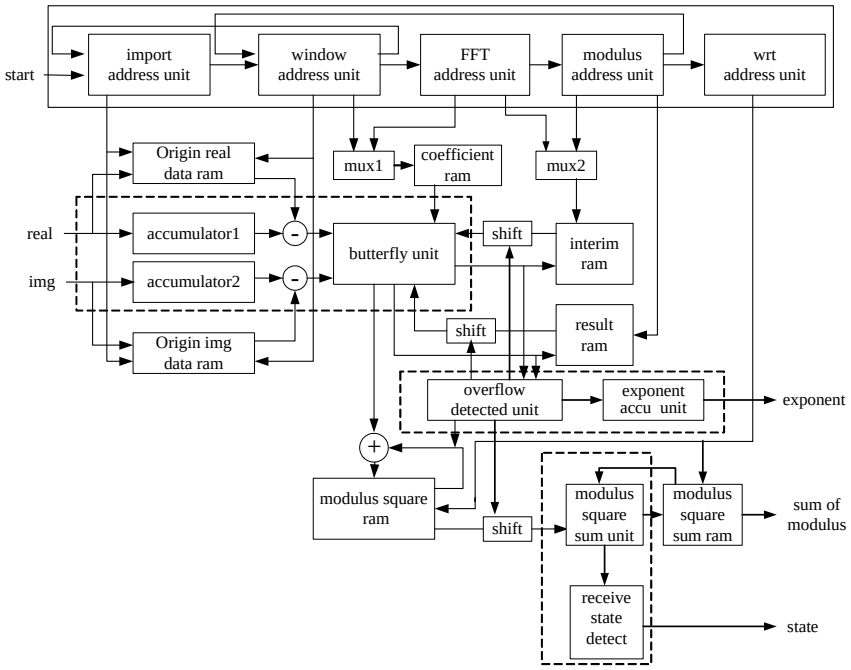

Fig.2: The block diagram of the signal processor

Fig 2 is the whole diagram of this processor. It is made up of five kinds of unit: the operation unit, the block floating point unit, the address unit, the memory unit, and modulus square sum and receive state detect unit.

A. Operation Unit

The operation unit performs arithmetic operation. It includes of subtracting DC component unit, butterfly unit.

Real data and imaginary data are imported into the processor parallel and stored in origin data ram separately. At the same time they are accumulated in accumulators. After a frame of data is read into the processor, the real data and imaginary data from origin rams subtract the average of them separately and entered into the butterfly unit parallel, that is the function of subtracting DC component.

In succession, multiplying window function and 512-point FFT are to be implemented. In this processor radix-2 algorithm and recursion structure are used to implement FFT. The coefficients enter into the butterfly and calculate with the input data to generate windowing results and FFT results. The windowing results and the first eight stages' FFT results are stored into the interim ram. The last stage FFT results are stored into the result ram. Then FFT results are imported into the butterfly unit to generate their modulus squares.

The core of this processor is the butterfly unit. Multiplying window coefficients, 512-point FFT, and computing the complex modulus square are performed in it. Fig3 is its overall block diagram.

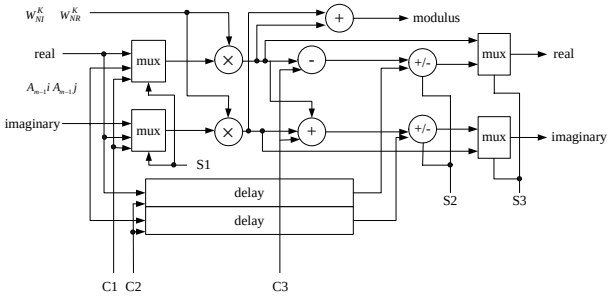

Fig.3 . The structure of butterfly unit

(1) In the control of S1, window coefficients and the data reach multiplier directly. The results export the butterfly unit by the control of S3 to interim ram.

(2) Decimation-in-time radix-2 algorithm is defined as:

$$A_m(i)=A_{m-1}(i)+A_{m-1}(j)W_N^k \qquad (1)$$

$$A_m(j)=A_{m-1}(i)-A_{m-1}(j)W_N^k \qquad (2)$$

The complex multiplier is defined as:

$$(A_{m-1}(j)W_N^k)_R = A_{m-1}(j)_R W_{NR}^k - A_{m-1}(j)_I W_{NI}^k \qquad (3)$$

$$(A_{m-1}(j)W_N^k)_I = A_{m-1}(j)_I W_{NR}^k + A_{m-1}(j)_R W_{NI}^k \qquad (4)$$

In the above formulas **i** and **j** express the sequence number of data, **R** and **I** express the real part and the imaginary part. $\mathbf{W}_N^k$ represent the twiddle coefficient, **m** expresses the stage of FFT. The real part of twiddle coefficient firstly enters butterfly. After one clock, the imaginary part enters it. The complex signal are imported in butterfly parallel in sequence of $A_{m-1}(j)$,$A_{m-1}(i)$. Because $A_{m-1}(i)$ doesn't take part in complex multiplier, it adds or subtracts the result of complex multiplier after the delay controlled by C2. In the control of C3, $A_{m-1}(j)_R W_{NR}^k$ and $A_{m-1}(j)_I W_{NI}^k$ perform subtraction, $A_{m-1}(j)_R W_{NI}^k$ and $A_{m-1}(j)_I W_{NR}^k$ perform addition. S3 control the result to export butterfly unit to interim ram.

(3) Modulus square is defined as $I^2+R^2=M^2$. The real part **I** and the imaginary part **R** enter butterfly unit parallel to multiply and add. To approve the precision, the result of modulus square is 32-bit.

B. Block Floating Point Unit

Block floating algorithm's implementation is easier than floating-point algorithm's, and it's precision is higher than fixing-point algorithm's[3].

The block floating point unit is made up of two parts: the overflow detected unit and the exponent accumulator unit. If the import of butterfly is N-bits and the expansion of one time of addition is one, the export of butterfly is N+2 bits. It passes the overflow detected unit to determine the overflow. If the first three bits are "000" or "111" then overflow is 0. If the first three bits are "001" or "110", then overflow is 1. If they are "01x" or "10x" (x represents don't-care), then overflow is 2. The exponent accumulator implements the accumulator of the overflows of FFT's every stage. The result of every frame's modulus square is made up of a fixed exponent and a 32-bit two's complement binary.

Because the block floating algorithm is use, the complex modulus is made up of two parts: output of the butterfly unit and the exponent that is generated in the butterfly unit. Before the two outputs are accumulated, their exponents are to be uniformed to the bigger one. The times to be shifted are twice of the exponents' difference. The overflow in the procedure of accumulation is detected by the overflow detected unit. The exponent of the last result is the sum of the overflow and the twice of the exponent that is generated in the butterfly unit.

C. Address Unit

Five modules build up the address unit. The import address unit imports exterior data into the processor. The window address unit, the FFT address unit and the modulus address unit generate address and control signals to implement all of the four functions. The wrt address unit makes final result into interior memories to wait for be exported.

The data to be calculated its modulus square will be read twice. Because FFT's results are stored in memory in reverse, the address that the modulus square unit's import is reversed. Based on the in-place algorithm, memory's write address can be attained by delaying the memory's read address.

D. Memory Unit

The processor use embedded dual-port ram in FPGA to store all of the data and coefficients. In this chip the data is stored with the Triple-Memory-Space mode. At the time of dealing with one group of data, the other group of data can be imported into rams. By this way, the operation unit can deal with the data without waiting for the ram's I/O operations and the whole speed of this system is increased.

E. Modulus Square Sum and Receive State Detect Unit

Three frames data's modulus squares are accumulated in the modulus square sum unit. The results are imported into the modulus square sum ram and wait for being read.

The phase reference signal is real sine, so frequency point of each channel 512-point FFT are symmetrical. CFAR arithmetic in frequent field is used to confirm 18dB SNR threshold. If two in three frequency points exceed threshold, this channel is considered normal state. This unit is composed of four accumulators and one comparators, and outputs final comparison result.

3. Design Implementation

Design usese the top-down methodology and RTL-level VHDL. The complex signal processor is implemented in a

single Xillinx Virtex-II FPGA chip.

The main features of this chip are shown in the table 1.

Table1 The main features of this chip

FEATURE	VALUE
Device	XC2V500-4
Maximal Operating Frequency	100MHz
Gate Count	260000 eq. gates
Peak Power	412mW
Data Output	16-bit Block Floating

To test this chip, some representative data is selected to operate with the chip. The operation speed is analyzed and the result is compared with the result of Matlab simulation.

When the three frames signal are

10*cos(2*pi/512*30*t)-5+randn(1,512)+j*
(10*sin(2*pi/512*30*t)-5+randn(1,512)),

-10*cos(2*pi/512*30*t)+5+j*(-10*sin(2*pi/512*30*t)+5),

-15*cos(2*pi/512*30*t)+5+j*(-15*sin(2*pi/512*30*t)-5),

S/N of the result from this processor compared with the fact value is depicted in the Fig4. We can see that error between result of processor and result of Matlab simulation is very small at the frequency point of strong or moderate signal, but error is bigger at the frequency point of small signal. The reason is part mantissa rounding of multiplication result in signal processing which affect precision of small signal. But as a whole, the block float-point algorithm can improve the dynamic range and data precision of radar system.

The radar system is simulated with this signal processor being a part of system. When the clock frequency is 50MHz, three frames of 512-point complex signal's sum of power spectra in 448μs. It satisfies radar's requirement of real time operation.

4. Conclusion

This complex signal processor is implemented in a single FPGA chip and can performs five kinds of operation in sequence: subtracting DC component, multiplying window coefficients, FFT, computing complex data's modulus and computing the sum of the three frame complex data's modulus square in 224μs. FFT's butterfly unit is shared in the design to make the processor saves a great deal of area. FPGA makes the design easy to be extended function. Block floating point algorithm is used to make the design have very high precision.

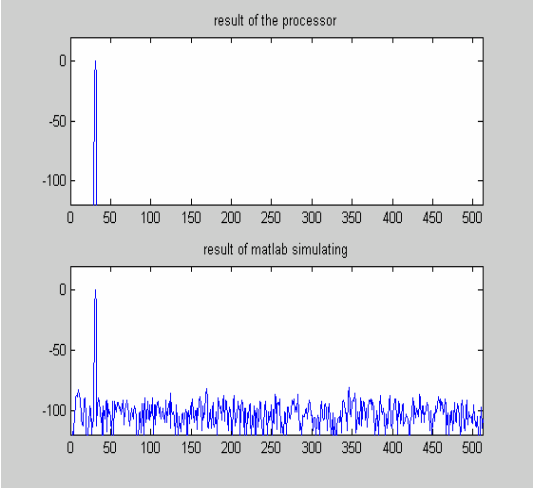

Fig. 4: Comparison of the S/N

References

[1] YANG, Z.-X., HU, Y.-P., PAN, C.-Y., and YANG, L."Design of a 3780-point IFFT processor for TDS-OFDM," *IEEE Transactions Broadcast.*, 2002, 48, (1), pp. 57–61

[2] S.He and M.Torkelson, "A new approach to pipeline FFT processor," *in Proc.IEEE International Parallel Processing Symposium,* pp.760-770,IPPS'96,IEEE,April 1996

[3] A.V.Oppenheim. "Realization of digital filers using block-floating point arithmetic," *IEEE Trans.on Audio Electro*,1970,18(1),pp.130-136

[4] C.D.Thompson, "Fourier transforms in VLSI" *IEEE transactions on computers*, vol.C-32, pp.1047-1057, November 1983

[5] S.Sukhsawas, K.Benkrid, "A high-level implementation of a high performance pipeline FFT on Virtex-E FPGAs," *VLSI, 2004. Proceedings. IEEE Computer society Annual Symposium on* , 19-20 Feb. 2004,pp.229 - 232

[6] MaY.T, "VLSI-oriented parallel FFT algorithm," *IEEE Trans on Signal Processing* , 1996, 44 (2) , pp.445- 448