Implementation of broadband radar Doppler signal processing algorithms based on special computing unit platform

Arsenii Vodolazov[1], *Dmitrii* Koroteev[1], *Sergei* Rastvorov[1], *Pavel* Shatov[1,*], and *Dmitrii* Slyusarenko[1]

[1]NII RET, Bauman Moscow State Technical University, 5 2-ya Baumanskaya Str., 105005, Moscow, Russia

Abstract. An approach to hardware-oriented software optimization during the implementation of real-time digital radar signal processing is described. A typical moving target indication algorithm is considered. The main principles of the optimization approach are given. The speed of the optimized algorithm in comparison to the implementation performed by standard software means is shown. The possibility of potential increase in signal processing speed with hardware-oriented optimization is indicated.

1 Introduction

Modern broadband radar design is characterized by constant increase in complexity and cost of analog-to-digital and digital-to-analog conversion paths, as well as those of specialized computing equipment that provides both digital signal generation and subsequent processing in real time. The complexity of signal processing algorithms also tends to increase.

The largest contribution to the cost of the computing unit is typically made by the parts that implement primary signal processing. Despite the relatively low complexity of algorithms used at this stage, they are required to process large volumes of data in real time, which leads to the need to use quite expensive hardware and software solutions, for example field-programmable gate arrays (FPGA) or even application-specific integrated circuits (ASIC).

Implementation of the primary signal processing algorithms on more general purpose electronic components, such as digital signal processors (DSP) allows on one hand to simplify the structure of the computing unit, on the other hand reduces its cost and simplifies the process of embedded software development.

This article describes an example implementation of the moving target indication (MTI) algorithm on a digital signal processor. The optimization of the algorithm is carried out taking the selected hardware platform into account. Performance of programs written in assembly language and C language with and without compiler tool optimization is compared and results are presented.

* Corresponding author: meisterpaul1@yandex.ru

2 Description of the algorithm and the special computing unit

MTI is a typical primary signal processing algorithm, it can be implemented using iterative compensation with different repetition factors or notch filters [1, 2]. One of the widely used methods is the application of the fast Fourier transform (FFT). Figure 1 shows the general scheme of the Doppler pulse processing using the FFT [3].

Fig. 1. FFT-based MTI algorithm.

The received signal is first passed through a compression filter, where it its convolution with an impulse response of the matched filter is calculated. Then the samples corresponding to the same range are fed into the weighting block, which is necessary to eliminate the spectrum spreading effect:

$$y_{n,k}^{w} = y_{n,k} \cdot w_n, \tag{1}$$

where w is the weighting window of length N and y^w is the compressed signal with applied weighting window. Parameter N is determined by such radar parameters as the width of the main lobe of the antenna, its rotation speed and pulse repetition period.

The weighted signal then undergoes the FFT:

$$S_{m,k}^{y} = \sum_{n=0}^{N-1} y_{n,k}^{w} \cdot e^{-j\frac{2\pi nm}{N}}, \tag{2}$$

where S^y is the spectrum of the signal y^w and m is the number of the frequency channel.

The squares of the frequency response amplitudes $\left|S_{m,k}^{y}\right|^2$ can then be fed into threshold processing. Instead of the traditional absolute value computation step after the FFT, the square of the complex value modulus is calculated. This modification helps avoid the burden of computing the square root, which is typically implemented using iterative methods and requires significant resources to achieve acceptable accuracy.

Further the implementation of the previously described MTI algorithm with $N = 16$ is considered. The hardware platform of the special computing unit is based on a TMS320C6678 DSP produced by Texas Instruments Incorporated. The processor has eight cores, each core is capable of up to 16 single-precision floating-point operations (FLOP) per cycle with clock frequency of 1,0 or 1,25 GHz. The peak core performance is thus 20 GFLOPs, while the total peak processor performance is 160 GFLOPs. The processor allows convenient processing of complex samples due to the vast set of single-instruction-multiple-data (SIMD) operations, including operations on complex numbers. For instance, adding two complex numbers takes only one clock cycle, while multiplication takes just two.

Each core contains sixty four 32-bit general-purpose registers split into two files A and B. Computations are performed by eight dedicated blocks (D1, D2, M1, M2, S1, S2, L1 and L2). Blocks M1 and M2 are for multiplication, D1 and D2 are for memory transfers in different addressing modes, blocks S1, S2, L1 и L2 are for logic and arithmetic operations. The register file A is directly linked to the inputs and outputs of the function blocks D1, M1, S1 and L1 and to the inputs of the blocks D2, M2, S2 and L2 via a 64-bit bus. Similar connections exist for the register file B. Figure 2 shows the simplified structure of one processor core.

In comparison with previous generations of signal processes, such architecture of the processor core significantly expands the capabilities of an embedded software developer to take advantage of parallel execution of computing operations. The VLIW architecture allows up to eight commands (one for each of the function blocks) to be executed per one clock cycle. Taking in account the aforementioned features, as well as additional functional limitations arising from the organization of data buses and command system, the previously mentioned MTI algorithm was adapted for the architecture of the described processor.

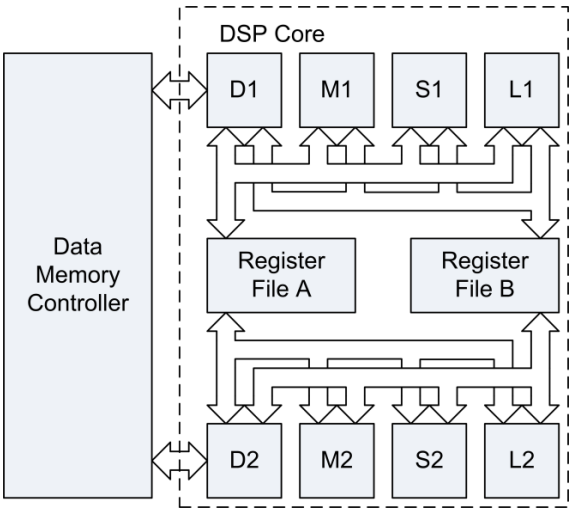

Fig. 2. Structure of one TMS320C6678 processor core.

One of the factors limiting the speed of the algorithm is the presence of only two functional units capable of loading data from memory per core. Taking the width of the data bus into account, each unit is able to load only two floating point numbers per cycle that constitute one complex sample. Another trait of the algorithm is the small number of pulses per packet, thus it seems reasonable to store the weighting window coefficients directly in registers, since they are the most frequently used data. The same approach can be applied for storing intermediate results of calculations. In that sense the weighting operation is fused into the MTI algorithm instead of being a separate processing step.

Another feature of the processor core architecture is support for floating-point operations. For example, adding two floating point numbers requires only one clock cycle of the functional blocks L or S, but the resulting value will only be available after two clock cycles. Thus, the speed of pipelined algorithms operating on floating-point numbers can be compared with similar algorithms running on integer numbers. However, floating-point operations typically do not need dedicated monitoring of overflows and block processing, that complicate calculations.

Figure 3 shows the functional scheme of the Doppler processing algorithm for a pack of 16 pulses. The FFT procedure is implemented using blocks of four-point FFT, where multiplications by rotating coefficients are reduced to swapping of the real and imaginary parts of the complex number and changing the sign of one of them [4]. Input samples are read from memory in binary-inverse order and after multiplication by the corresponding coefficients of the weighting window arrive at the first stage of four-point FFT. Then multiplication by the rotating coefficients (some of them are degenerate, i.e. equal to ± 1, $\pm j$) and interleaving of intermediate samples using buffer memory is performed. This operation is followed by the second stage of four-point FFT and squaring of the modules of complex samples, which are finally stored in memory in direct order.

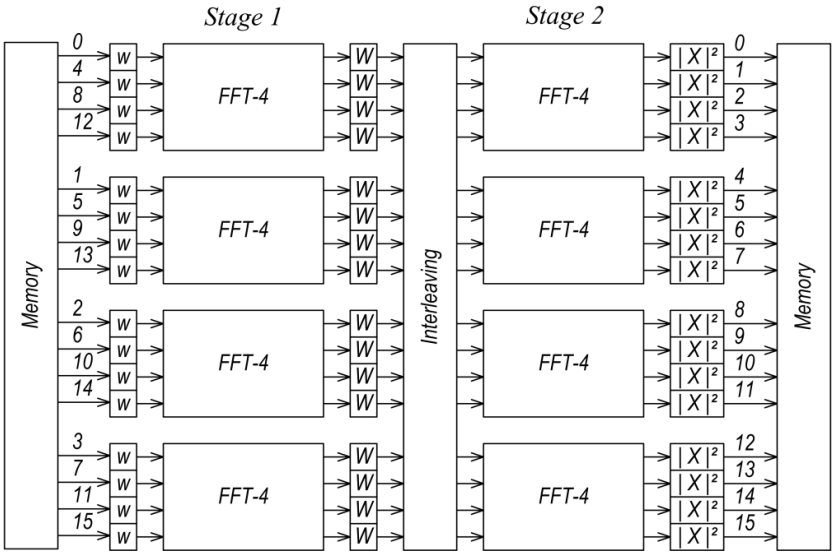

Fig. 3. Functional scheme of FFT-based MTI algorithm implementation, *w* denotes multiplication by the weighting coefficient, *W* denotes multiplication by the rotation coefficient.

Taking the aforementioned features of the core architecture and the command set of the processor in to account, the Doppler processing can be conveniently split into two stages:

1) First stage
   - loading of input samples from memory (D blocks)
   - application of weighting window coefficients (M and S blocks)
   - computation of 4-point FFT (L and S blocks)
   - multiplication by rotating coefficients (D, M and L blocks)
   - saving of intermediate results in memory (D blocks)
2) Second stage:
   - loading intermediate results from memory (D blocks)
   - computation of 4-point FFT (L and S blocks)
   - squaring of complex sample modulus (M and L blocks)
   - saving of final results into memory (D blocks)

This way at each stage independent processing of four groups of four complex samples per group is performed. Smart allocation of computational resources allows almost parallel processing of all four sample groups. Figure 4 presents the timing diagram of loading of computational resources of the processor core. The implementation of the algorithm takes into account the possibility to not store certain intermediate results in buffer memory and use registers to do interleaving instead.

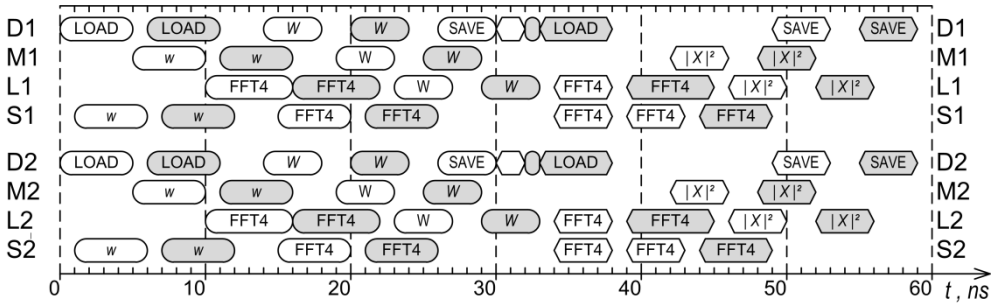

Fig. 4. Timing diagram of core computing blocks load.

Table 1 presents the testing results of the programs implementing the aforementioned MTI algorithm with different levels of optimization. According to the results, it can be seen that the implementation of the algorithm in assembly language, with all the hardware features of the core taken into account, is several times faster than the program after optimization by means of the vendor-supplied C code compiler.

This implementation leads to around 85% utilization of available registers and around 40% load of computational resources. The main part of the algorithm (excluding initial initialization) is performed in 59 processor cycles.

Table 1. Performance with different levels of optimization.

Type of algorithm implementation	**Run time, ns**
Program in C language without optimization	4119
Program in C language with compiler optimization	278
Program in assembly language with manual optimization	60

From Picture 4 it can be seen, that during the execution of the algorithm there are periods of time when certain computing blocks are idle. Thus, there is an opportunity to further increase the efficiency of signal processing, for example, by integrating into the algorithm further processing steps, such as threshold processing.

3 Conclusions

After testing of the program code, that was manually optimized taking into account the features of the hardware platform, it was found that its performance is more than four times higher than that of the program written in C and optimized by standard means of the compiler. Thus, there are obvious savings in computing time and equipment resources, which in turn can substantially reduce the cost of the computing unit itself, as well as reduce its power consumption and heat generation.

The disadvantages of the described low-level manual program code optimization can be attributed to a relatively higher complexity of the development process, as well as potential difficulties if the program needs further modification or upgrade.

References

1. Ia.D. Shirman, V.N. Golikov, I.N. Busygin, G.A. Kostin, V.N. Manzhos, N.N. Minervin, B.V. Naidenov, V.I. Poliakov, A.S. Chelpanov, *Teoreticheskie osnovy radiolokatsii: uchebnoe posobie dlya vuzov* (Sovetskoe radio, Moscow, 1970)
2. V.A. Vasin, I.B. Vlasov, Iu.M. Egorov, V.V. Kalmykov, A.A. Kuznetsov, A.I. Nikolaev, V.B. Pudlovskii, V.A. Rodzivilov, Iu.N. Sebekin, A.I. Senin, G.P. Slukin, I.B. Fedorov, *Informatsionnye tekhnologii v radiotekhnicheskikh sistemakh: uchebnoe posobie* (Bauman Moscow State Technical University Publishing House, Moscow, 2003)
3. L.R. Rabiner, B. Gold, *Theory And Application of Digital Signal Processing* (Prentice-Hall, New Jersey, 1975)
4. A.B. Sergienko, *Tsifrovaya obrabotka signalov: uchebnik dlya vuzov* (Piter, Saint Petersburg, 2003)