Practical Time Optimization of DSP Program for High-frequency Applications

Zhuyu Xun¹, Hongfa Ding¹,* and Zhou He¹
¹ Wuhan National High Magnetic Field Center, School of Electrical and Electronic Engineering, Huazhong University of Science and Technology, Wuhan, China

*Corresponding author email: dhf@hust.edu.cn

Abstract. The rapid development of the high frequency power conversion techniques makes great demands on the methods that can reduce the execution time of the program effectively. This paper is aiming at reducing the execution time of the program in several aspects such as sampling, complex expressions, and so on. As one of the most widely applied methods, reducing the execution time of the program at the cost of the memory space is adopted in this paper. Furthermore, in order to confirm the feasibility and superiority of programs that are proposed in this paper, they are compared with other programs that can realize the same function in terms of the execution time.

Keywords: Execution time; High frequency; Complex expressions.

1. Introduction
As one of the fastest growing areas in electrical engineering, the power electronic technology attract more and more considerable attention. Meeting the requirement of high-efficiency and high-quality electrical energy, the power electronic technology has been widely applied in transportation, power system, chemical industry, machinery, household electric appliance, spacecraft, and so on. The share of the electrical energy that has been processed by the power converters is an important index of a country's technological advancement. This important index will reach 100% in the not too distant future. Furthermore, the problem of environmental protection has become more and more serious. The power electronic technology can not only reduce the demand for copper and steel but also save the electrical energy. Therefore, the power electronic technology can effectively improve the environment that we live in [1-3].

The rapid development of the communication industry and consumer electronics makes new demands on the power electronic technology, such as high power density and high efficiency. Therefore, for higher power density and higher efficiency, the power electronic technology has developed from low frequency power supplies predominated into high frequency power supplies predominated [4,5]. The power supplies that use high frequency technology have become a research topic of interest [6,7]. With the maturity of the silicon technology, the performance of power converters that use silicon-based semiconductor devices reaches the theoretical peak [8,9]. Therefore, to meet the market demand for high voltage, low power dissipation, miniaturization, high current, low electromagnetic susceptibility, high power density and so on, the significant breakthrough of new wide band gap semiconductors such as SiC and GaN has been made, making it possible to develop a new generation of power devices [10,11].

The band gap of GaN is 3.40 eV and the band gap of SiC is 3.25 eV. The wide band gap characteristic can improve the withstand voltage performance of power devices. Possessing the characteristic of
higher thermal conductivity, the new wide band gap semiconductors can be applied in high-temperature power devices. Furthermore, the new wide band gap semiconductors have larger electron saturation velocity. This advantage makes them more suitable for high-frequency power devices [12,13].

On the one hand, the goal of realizing higher power density and higher efficiency makes great demands on the power supplies that use high frequency technology [14]. On the other hand, the rapid development of new wide band gap semiconductor devices makes it possible to produce the power supplies with much higher frequency [15]. Therefore, these two aspects provide the high frequency power conversion techniques such as zero voltage switching and zero current switching technology with great development opportunities.

Although the high frequency power conversion techniques have ushered in the great development opportunity, there are few researches about how to reduce the execution time of the program. The rapid development of the high frequency power conversion techniques makes great demands on the methods that can reduce the execution time of the program effectively. In fact, the execution time of different programs that can realize the same function or the same technology varies widely. Therefore, searching for new approaches that can reduce the execution time of the program effectively is urgent.

This paper is aiming at reducing the execution time of the program in the following four aspects.

- Four fundamental rules.
- Complementation.
- Sampling.
- Complex expressions.

The execution time of the program in this paper is tested in the DSP TMS320F28379D without further optimization.

2. Reducing the Execution Time

2.1. Four Fundamental Rules

Within the four fundamental rules, the DSP has a good ability in addition, subtraction and multiplication. However, it’s difficult for the DSP to do division. Different kinds of programs that can realize division and the corresponding execution times are listed in Table 1.

| Program | Execution time |
|---------|----------------|
| //The following program is executed at the beginning. Vdc=30; | 1245 ns |
| //The following program is executed when needed. k1=uc1/Vdc; | |
| //The following program is executed at the beginning. Vdc=30; VdcReciprocal=1.0/Vdc; | 50 ns |
| //The following program is executed when needed. k1=uc1 * VdcReciprocal; | |

According to table 1, using multiplication to realize division in another different way can effectively decrease the execution time.
2.2. Complementation

Complementation is usually used to judge whether the number is odd or not. Different kinds of programs that can realize complementation and the corresponding execution times are listed in Table 2.

Table 2. Different kinds of programs that can realize complementation and the corresponding execution times.

| Program                                      | Execution time |
|----------------------------------------------|----------------|
| if(count%2) Goal=1; else Goal=0;             | 280 ns         |
| if(count&1) Goal=1; else Goal=0;             | 45 ns          |
| Goal=count&1;                                | 15 ns          |

According to table 2, using the bitwise operations to realize complementation in another different way can effectively decrease the execution time. Furthermore, without the judgment process, the direct assignment has an obvious advantage in execution time.

2.3. Sampling

Owing to the fluctuation of the sampling data, filter methods need to be adopted after sampling. Among many filter methods, the moving average filter is an important filter method. The moving average filter doesn’t need to do some complicated operations such as sorting. Furthermore, only one sampling operation is required in the moving average filter. These characteristics make the moving average filter suitable for high-frequency applications.

Table 3. Different kinds of programs that can update data and the corresponding execution times.

| Program                                      | Execution time |
|----------------------------------------------|----------------|
| for(i=0;i<3;i++) filter[i]=filter[i+1];     | 505 ns         |
| filter[3]=AdcResult;                         |                |
| filter[0]=filter[1]; filter[1]=filter[2];   | 65 ns          |
| filter[2]=filter[3];                         |                |
| filter[3]=AdcResult;                         |                |

The sum of the stored sampling data is updated by deducting the oldest sampling data and adding the newest sampling data. The length of the stored sampling data N is a key parameter. To reduce the impact on the sensitivity, N is usually in single figures. N = 4 is taken as an example. Different kinds of programs that can update the stored sampling data and the corresponding execution times are listed in Table 3. AdcResult is the sampling result and filter is used to store the sampling data. The newest sampling data is stored in filter[3]. The execution time of the “for loop” is 505 ns while the execution time of the direct assignment is merely 65 ns. Both of them can update the stored sampling data. The cause of this phenomenon is that too much time is spent on judging whether i < 3. Therefore, “for loop”
is simple and readable, but poor at decreasing the execution time. On the contrary, the direct assignment has an obvious advantage in execution time. Without additional procedures, the direct assignment can finish the task quickly. The condition that $N$ is in single figures eliminates the disadvantage of the direct assignment in simplicity. Therefore, the direct assignment is more suitable for high-frequency applications. With no complicated operations such as sorting and only one sampling operation, the moving average filter that adopts the direct assignment is attractive and competitive in high-frequency applications.

2.4. Complex Expressions

DSP has a remarkable ability of calculation and has been widely used in many applications. However, some complex math expressions sharply increase the execution time of programs. To solve this problem, the method that can reduce the execution time of the program at the cost of the memory space need to be adopted. This basic and powerful method is suitable for not only reducing the time complexity in the heuristic algorithm but also solving the problem of complex math expressions in the DSP. For example, $t_a$ can be calculated as follows:

\[
 t_{a} = \sqrt{\frac{(k_1 + p_2)^2 - (k_1 + p_2)^2}{1 - p_1 - k_1}} \tag{1}
\]

where $k_1$ is the only independent variable. Others are constants. Different kinds of programs that can finish the calculation of $t_a$ and the corresponding execution times are listed in Table 4.

### Table 4. Different kinds of programs that can finish the calculation of (1) and the corresponding execution times.

| Program                                                                 | Execution time |
|------------------------------------------------------------------------|----------------|
| //The following program is executed at the beginning.                  |                |
| Con1=sqrt(Lr*C);                                                       |                |
| Con2=(k3+p2)*(k3+p2);                                                  | 1700 ns        |
| Con3=1-p1;                                                             |                |
| //The following program is executed when needed.                       |                |
| ta=Con1*sqrt((Con2-(k1+p2)*(k1+p2))/(Con3-k1));                       |                |
| //The following program is executed at the beginning.                  |                |
| Con1=sqrt(Lr*C);                                                       |                |
| Con2=(k3+p2)*(k3+p2);                                                  |                |
| Con3=1-p1;                                                             |                |
| for(i=0;i<501;i++)                                                     | 105 ns         |
| {                                                                     |                |
|   k1=i/1000.0;                                                         |                |
|   Result[i]=Con1*sqrt((Con2-(k1+p2)*(k1+p2))/(Con3-k1));              |                |
| }                                                                     |                |
| //The following program is executed when needed.                       |                |
| index=k1*1000.0+0.5;                                                   |                |
| ta=Result[index];                                                      |                |

In Table 4 index is an integer variable. $k_1$ varies from 0 to 0.5. Divide the range of variation into 500 even pieces and the error is neglectable. Furthermore, the reason why index = $k_1*1000.0+0.5$ is that the nearest $t_a$ of the real value can be calculated. Result[500] is generated for the $k_1$ that is larger than
0.4995. Obviously, the execution time of the direct calculation is much longer than reading the value from the array. For another example, $u_d$, $u_q$, and $u_0$ can be calculated as follows:

$$
\begin{bmatrix}
  u_d \\
  u_q \\
  u_0
\end{bmatrix} =
\begin{bmatrix}
  \cos \omega t & \cos(\omega t - \frac{2\pi}{3}) & \cos(\omega t + \frac{2\pi}{3}) \\
  -\sin \omega t & -\sin(\omega t - \frac{2\pi}{3}) & -\sin(\omega t + \frac{2\pi}{3}) \\
  \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix}
\begin{bmatrix}
  u_x \\
  u_y \\
  u_z
\end{bmatrix}
$$

(2)

Different kinds of programs that can finish the calculation of $u_d$, $u_q$, and $u_0$ and the corresponding execution times are listed in Table 5. In Table 5 index is an integer variable. $\omega t$ varies from 0 to $2\pi$. Divide the range of variation into 360 even pieces and the error is negligible. Similarly, the execution time of the direct calculation is much longer than reading the value from the array.

**Table 5.** Different kinds of programs that can finish the calculation of (2) and the corresponding execution times.

| Program                                                                 | Execution time |
|------------------------------------------------------------------------|----------------|
| /*The following program is executed at the beginning.                  |                |
| Con1_sq2=1/sqrt(2);                                                    |                |
| Consq3_2=sqrt(3)/2;                                                    |                |
| /*The following program is executed when needed.                       |                |
| CsRel=cos(w*t);                                                        | 1365 ns        |
| SnRel=sin(w*t);                                                        |                |
| ud=ua*CsRel+ub*(-0.5*CsRel+Consq3_2*SnRel)+uc*(-0.5*CsRel-Consq3_2*SnRel); |                |
| uq=-ua*SnRel-ub*(-0.5*SnRel-Consq3_2*CsRel)-uc*(-0.5*SnRel+Consq3_2*CsRel); |                |
| u0=Con1_sq2*(ua+ub+uc);                                                |                |
| /*The following program is executed at the beginning.                  |                |
| pi=3.1415926;                                                          |                |
| piReciprocal=1/pi;                                                     |                |
| Con1_sq2=1/sqrt(2);                                                    |                |
| Consq3_2=sqrt(3)/2;                                                    |                |
| for(i=0;i<361;i++)                                                     |                |
| {                                                                      |                |
|   CosOrResult[i]=cos(i*2*pi/360);                                      | 570 ns         |
|   SinOrResult[i]=sin(i*2*pi/360);                                      |                |
| }                                                                      |                |
| /*The following program is executed when needed.                       |                |
| index=w*t*180*piReciprocal+0.5;                                       |                |
| CsRel=CosOrResult[index];                                             |                |
| SnRel=SinOrResult[index];                                             |                |
| ud=ua*CsRel+ub*(-0.5*CsRel+Consq3_2*SnRel)+uc*(-0.5*CsRel-Consq3_2*SnRel); |                |
| uq=ua*SnRel-ub*(-0.5*SnRel-Consq3_2*CsRel)-uc*(-0.5*SnRel+Consq3_2*CsRel); |                |
| u0=Con1_sq2*(ua+ub+uc);                                               |                |
Reducing the execution time of the program at the cost of the memory space is a basic and powerful method and has a good ability in solving the problem of complex math expressions.

3. Conclusion
The goal of realizing higher power density and higher efficiency and the rapid development of new wide band gap semiconductor devices provide the high frequency power conversion techniques with great development opportunities. This paper is aiming at reducing the execution time of the program in several aspects such as sampling, complex expressions, and so on. As one of the most widely applied methods, reducing the execution time of the program at the cost of the memory space is adopted in this paper. Programs that are proposed in this paper are compared with other programs that can realize the same function in terms of the execution time. The comparison results confirm the feasibility and superiority of programs that are proposed in this paper.

References
[1] D Das, V M Hrishikesan, C Kumar and M Liserre 2021 Smart transformer-enabled meshed hybrid distribution grid *IEEE Trans. Ind. Electron.* 68(1) pp 282-292
[2] Y Yan et al. 2021 High-power magnetoelectric voltage tunable inductors *IEEE Trans. Ind. Electron.* 68(6) pp 5355-5365
[3] S Oh et al. 2021 High-efficiency multilevel multimode dynamic supply switching modulator for LTE power amplifier *IEEE Trans. Power Electron.* 36(6) pp 6967-6977
[4] H Chalangar, T Ould-Bachir, K Sheshyekani and J Mahsereedian 2021 A direct mapped method for accurate modeling and real-time simulation of high switching frequency resonant converters *IEEE Trans. Ind. Electron.* 68(7) pp 6348-6357
[5] K. Li et al. 2021 Design and implementation of four-port megawatt-level high-frequency-bus based power electronic transformer *IEEE Trans. Power Electron.* 36(6) pp 6429-6442
[6] B Liu, W Chen, J Wang and Q Chen 2021 A practical inductor loss testing scheme and device with high frequency pulsewidth modulation excitations *IEEE Trans. Ind. Electron.* 68(5) pp 4457-4467
[7] D Lan, P Das, C Ye, S K Sahoo and D Srinivasan 2021 High-frequency link matrix rectifier in discontinuous conduction mode with reduced input current distortion *IEEE Trans. Ind. Electron.* 68(7) pp 5517-5526
[8] A Chafi, N Idir, A Videt and H Maher 2021 Design method of PCB inductors for high-frequency GaN converters *IEEE Trans. Power Electron.* 36(1) pp 805-814
[9] J Chen, D Sha and J Zhang 2021 Current ripple prediction and DPWM-based variable switching frequency control for full ZVS range three-phase inverter *IEEE Trans. Ind. Electron.* 68(2) pp 1412-1422
[10] Z Qi et al. 2021 An accurate datasheet-based full-characteristics analytical model of GaN HEMTs for deadtime optimization *IEEE Trans. Power Electron.* 36(7) pp 7942-7955
[11] R Hou, J Lu, Z Quan and Y W Li 2021 A simple desaturation-based protection circuit for GaN HEMT with ultrafast response *IEEE Trans. Power Electron.* 36(6) pp 6978-6987
[12] K -I Do, J -I Won and Y -S Koo 2021 A 4H-SiC MOSFET-based ESD protection with improved snapback characteristics for high-voltage applications *IEEE Trans. Power Electron.* 36(5) pp 4921-4926
[13] H Li, Y Jiang, Z Qiu, Y Wang and Y Ding 2021 A predictive algorithm for crosstalk peaks of SiC MOSFET by considering the nonlinearity of gate-drain capacitance *IEEE Trans. Power Electron.* 36(3) pp 2823-2834
[14] Q Chen, J Xu, L Wang, R Huang and H Ma 2021 Analysis and improvement of the effect of distributed parasitic capacitance on high-frequency high-density three-phase buck rectifier *IEEE Trans. Power Electron.* 36(6) pp 6415-6428
[15] T Liu, C Chen, K Xu, Y Zhang and Y Kang 2021 GaN-based megahertz single-phase inverter with a hybrid TCM control method for high efficiency and high-power density *IEEE Trans. Power Electron.* 36(6) pp 6797-6813