An AUTOSAR ECU Mapping Algorithm

Chunmao Jiang, Meiyu Xu, Qian Shen
School of Computer Science Technology and Information Engineering, Harbin Normal University, Harbin, Heilongjiang 150025, China

***Abstract*—ECU (Electronic Control Unit) plays an important role in automotive electronics development as the core component of the vehicle. There are problems such as large overhead, unbalanced resource allocation in the current AUTOSAR ECU mapping algorithm that's widely adopted. To solve the problems, this paper made improvements on the current algorithm. The algorithm is proved to ensure to keep the balance of the storage and time resource allocation under the constraints of AUTOSAR.**

Keywords- AUTOSAR;reusability; Software Component To ECU Mapping algorithm;resource consumption;

I. INTRODUCTION

AUTOSAR standard defines a set of support distributed, feature-driven automotive electronics software development methodologies and software on electronic control units standardized solutions with applications in different cars and platforms to improve software reuse, reduce development costs.

Up to now, AUTOSAR systems research and development, China is still in the pre-feasibility study stage, have not yet entered the actual development phase. Zhejiang University in the field of operating systems in compliance with OSEK/VDX standard OSEK organization certified by the Smart0SEKOS [3]; communications systems in the field of research and reference AUTOSARCOM standard design and implementation of a communication system; driver field reference AUTOSAR standard design smartBSP [4].

In addition, the United States cooperate together to provide Freescale with Elektrobit AUTOSAR-compliant vehicle control software development, implementation of the program. The MathWorks and Vector Informatik companies cooperate to achieve mutual AUTOSAR application tool interoperability [5-7].

These studies have a common problem that the relevant constraints in compliance AUTOSAR case, the software components to the ECU map, have adopted the method [8], but the algorithm is stored in the storage time and the uneven distribution of resources and algorithms overhead and other issues [9], so as to improve overall system performance and functionality improvement brought greater bottleneck [9]. This paper intends to design and implementation of this study and the corresponding algorithm.

II. BASIC IDEAS AND DEFINITIONS

ECU like a car with dedicated computer, there is more than ECU in the car is running, the software components take up storage space and execution time is not the same, both from the point of view alone, there will be a problem of load balancing. At this point if the storage capacity used for each ECU is not balanced, there will be severe load imbalances. Therefore, good load balancing system, while considering the balance calculation, the balance of the storage capacity should be considered, so that the practical application in order to obtain higher performance.

If the ECU as a backpack, and software components as goods, storage consumption and execution time as a multidimensional attribute mapping software components to become a multi-dimensional multi-choice knapsack problem [10].Multi-choice knapsack problem is NP-complete problem, and is a multi-dimensional multi-choice knapsack problem child problem, so multi-dimensional knapsack problem is NP-complete problems. Therefore, at present, most of the heuristic algorithm, which were famous LPT algorithm, MULTIF IT algorithm, LPT [11] algorithm and the algorithm combines MULTIF IT BoundFit algorithm. However, too much overhead on these algorithms, and we only consider the execution time, memory space two-dimensional problem, so the introduction of a relatively simple and limited to one-dimensional analysis of the resource algorithms 3-1. For its shortcomings, an improved algorithm to achieve load balancing 3-2 software component mappings.

Assuming that all software components to achieve the desired total capacity of k bytes, using a total of n ECU for processing, the average storage capacity of each ECU is used for byte, called the average storage capacity. In this case, the ECU memory constraints can be expressed using the maximum storage capacity than the memory capacity of a known multiple of the average, said the multiple of the storage constraint factor. Therefore, to meet the constraint factor of the storage case, the execution time of each ECU to achieve a balance between. Given a reasonable case can certainly achieve the execution time and storage resources overall balance.

Definition 1. The number of software components is n, be used as a one-dimensional vector C = (c_1 , c_2 ,..., c_n).Software components corresponding execution time of T =($t_1, t_2,\ldots, t_n$),which corresponds to storage consumption denoted M=($m_1, m_2,\ldots, m_n$).At this time a total of ECU is number u, ECU instance number is also arranged according to a vector, denoted E=($e_1, e_2,\ldots, e_u$).

Definition 2. Set Partitioning I=($i_1, i_2,\ldots, i_u$) satisfy i_1=1, i_{u+1}= n+1,and $\forall$ r$\in$ 1,...,u are $i_r < i_{r+1}$.We use the u+1 th component of I , The n th component of u into disjoint regions {{$c_j \mid i_r \le$j$<i_{r+1}$}|r=1,...u}, Resulting in a division of C.

We assign each of the divided regions into one ECU, to obtain a software component mapping scheme. Here, I called the partition vectors, I components are called the split point. Split vectors and software components mapping is one to one. Thus, this is a problem of seeking the optimal partition.

Balance efficiency into execution time (ETBE) and storage balance efficiency (MBE) two concepts, to discuss the execution time of the balance and the balance of the storage capacity. Available, the execution time of each region is:

$$w_r = \sum_{j=i_r}^{i_{r+1}-1} t_j, r = 1,...u \quad (1)$$

Denoted $W = (w_1, w_2, ..., w_u)$

Definition 3. Calculate the efficiency of the execution time is：

$$ETBE = \frac{\sum_{j=1}^{u} w_j}{u \times \max(w_k)}, k = 1,...u \quad (2)$$

Each region division total storage requirements:

$$s_r = \sum_{j=i_r}^{i_{r+1}-1} m_j, r = 1,2...u \quad (3)$$

Denoted $S = (s_1, s_2, ..., s_u)$.

Definition 4. Storage efficiency is defined as the equilibrium：

$$MBE = \frac{\sum_{j=1}^{u} s_j}{u \times \max(s_k)} \quad (4)$$

Can be seen, ETBE is defined as the average execution time within the sub-region and the execution time ratio of the longest sub-area. Similarly, MBE sub-region is defined as the average consumption is stored and storage consumption ratio of the largest sub-region. Obviously, a good load balancing algorithm should be balanced so that the execution time and storage efficiency while balancing efficiency is relatively high.

The storage constraint factor previously given once, a threshold value can be obtained

$$M_a = \alpha \times \frac{\sum_{j=1}^{u} s_j}{u} \quad (5)$$

Storage constraint which requires the division to meet $\sum_{j=1}^{u} s_j \le M_a$ .An equivalent form of storage constraint is $MBE \ge \frac{1}{\alpha}$ .This can be transformed to meet the constraints of the memory while maximizing the efficiency of execution time balance, to get an optimal partition.

III. ANALYSIS OF EXISTING ALGORITHMS

Introducing execution time threshold W_{con} ,Sequential segmentation algorithm binary search W_{con} Split finally get optimum partitioning.Existing algorithms 3-1[5]are described below:

A. *algorithms 3-1:*

Function: I=1 Dpart(T,M,u, M_α)

//I is split vector, T is the execution time, M is the storage consumption, u the number of the ECU, the threshold for the storage constraint

// Set the execution time threshold search interval bounds

$$W_{lb} = \frac{\sum_{j=1}^{u} t_j}{u}; W_{ub} = \sum_{j=1}^{u} t_j; \ for(W_{con} = \frac{W_{lb} + W_{ub}}{2}$$

$$W_{ub} - W_{lb} \ge \varepsilon; W_{con} = \frac{W_{lb} + W_{ub}}{2}) \quad \{$$

$$for(i_1 = 1, i_u + 1 = n + 1, r = 1; r < u; r++)$$

$$i_{r+1} = \max(i_{r+1} \mid \sum_{j=i_r}^{i_{r+1}-1} t_j \le W_{con}, \sum_{j=i_r}^{i_{r+1}-1} m_j \le M_a\};$$

$$if(\sum_{j=i_u}^{n} m_j > M_\alpha \quad \text{or} \quad \sum_{j=i_u}^{n} t_j > W_{con}) W_{lb} = W_{con};$$

$$else$$

$$\{if(\sum_{j=i_u}^{n} t_j < W_{con}) return \quad I;\}\}$$

B. *algorithm analysis*

In this algorithm, α is a pre-selected fixed value and fit some preconditions, such as all the components must be less than M_α , In addition, it has the mapping constraints on the software components, etc. Therefore, this algorithm must be modified in using.

First, to determine the α initial value, from its definition, α is the ratio of the maximum memory consumption and the mean,

And M_α is the maximum value of the area occupied, but this threshold value is set in advance, the software component does not know at this time is the maximum number.

If there is a software component storage consumption is greater than this threshold, then no matter how, this component can not be placed in a zone.

So, first get the storage volume (M_{ETotal}) that all the ECU has and all the components of total memory consumption (M_{CTotal} , then get a upper value of α from the formula C($\alpha_{\max}$).

Then, taking up storage consumes the largest software component ($M_{c\max}$) in all software components, and then

compare with the average ratio of the (M_{avg}) with 1, take the larger one as the lower limit.

$$a_{\min} = Max(1, \frac{M_{c\max}}{M_{avg}}) \quad (6)$$

$$a_{\max} = \frac{M_{ETotal}}{M_{CTotal}} \quad (7)$$

Similarly, when the minimum is 1, the efficiency is optimal on storage balancing. This time set it as the initial value. Once set, it can be directly used for sequential subdivision iterative algorithm , then the optimal partition can be derived.

However, we found when a get a value, it may not take the division that we want.

For example, there are two ECU, now only consider the storage consumption, for each ECU storage capacity are 20, there are three storage consumption are 10 software components. According to the previous algorithm, calculate the value is 1.167, that is, each software component in the ECU memory consumption is less than M_α = 17.5. Obviously, the third software component will not put any one in the ECU, because no matter how its value is bound greater than M_α.

This is because there is a problem in the initial setting, and therefore, the value must be amplified, this time will surely have find a solution to meet the requirements. On the contrary, if successful, to remember the current division, further reduced the value to see if you can find the solution of a smaller storage consumption.

Also, consider the mapping constraints from the component to the ECU, when the selected component into an ECU, to achieve these constraints.

For some software components must be mapped in the same ECU constraints, because the table when creating it as the same software components involved in the mapping, so the constraint has been met.

Of course, this may affect the execution time of a certain degree of balance efficiency and storage capacity balance efficiency.

But you can certainly be able to find in certain circumstances, the execution time consuming optimal partition.

IV. IMPROVED ALGORITHM

Over the above analysis can be improved algorithm 4-1 is described as follows:

A. *Algorithm 4-1:*

Function: I=1 DPart(T,M,u, M_α)

//I is split vector, T is the execution time, M is the storage consumption, u the number of the ECU, the threshold for the storage constraint

$$M_\alpha = \alpha_{\min} \times \frac{\sum_{j=1}^{u} s_j}{u};$$

$$for(\alpha_{\min} = Max(1, \frac{M_{c\max}}{M_{avg}}); a_{\min} > a_{\max}; a_{\min} = a_{\min} + \delta)\{$$

$$W_{lb} = \frac{\sum_{j=1}^{u} t_j}{u}; W_{ub} = \sum_{j=1}^{u} t_j;$$

$$for(W_{con} = \frac{W_{lb} + W_{ub}}{2}; W_{ub} - W_{lb} \geq \varepsilon; W_{con} = \frac{W_{lb} + W_{ub}}{2})\{$$

//Before u-1 regions in execution time and memory constraints to meet the same time, as much as possible the distribution of components.

$$for(i_1 = 1, i_{u+1} = n+1, r = 1; r < u; r++)$$

$$i_{r+1} = \max(i_{r+1} \mid \sum_{j=i_r}^{i_{r+1}-1} t_j \leq Wcon, \sum_{j=i_r}^{i_{r+1}-1} m_j \leq M_\alpha\}$$

$$if(\sum_{j=i_u}^{n} m_j > M_\alpha \quad or \sum_{j=i_u}^{n} t_j > W_{con}) W_{lb} = W_{con};$$

$$else\{if(\sum_{j=i_u}^{n} t_j < W_{con}) W_{ub} = W_{con};$$

$$else\{M_\alpha = \alpha_{\min} \times \frac{\sum_{j=1}^{u} s_j}{u}; W_{lb} = \frac{\sum_{j=1}^{u} t_j}{u}; W_{ub} = \sum_{j=1}^{u} t_j;$$

$$if(\alpha_{\min} > \alpha_{\max}) return;\}\}\}$$

Record the value of *I*, W_{con} and $\alpha_{\min}$.}

B. *Algorithm Analysis*

Previous iterations of the algorithm, possibly due to the value is too small, the first two layers of the for loop can make W_{con} expanded to a maximum threshold, which exits the loop, record the current value associated.

Then, the algorithm according to certain δ adjustments continue iteration, will be able to find one W_{con} , making it $\alpha_{\min}$ in a certain fixed value, its execution time is less than W_{con} all, the correlation value of the current record.

Thus, until $\alpha_{\min} > \alpha_{\max}$, In this case, α change is obtained, each α corresponds to a value of W_{con} , and can get information about the execution time of a similar storage constraint factor β= $W_{con} \times u / \sum_{j=1}^{u} w_j$.

Mathematically we can prove [12] when α take a certain value, the execution time threshold W_{con} can converge to the optimal linear division. Since the calculation of the amount of the first zone increases monotonically with W_{con}, so the second split point increases monotonically with W_{con}. By mathematical induction to prove that the split point increases monotonically with W_{con}. Given any one subdivision to meet the memory constraints I, remember one of the most overloaded region execution time $\overline{W}$. Taken in

the algorithm $W_{con}=\overline{W}$ the calculated split I meet $W_{\max} \le \overline{W}$.

Execution time threshold $W_{con}=\overline{W}$, so that the amount of calculation of the front p - 1 regions is not more than $\overline{W}$. Also, because the last dividing point I is not less than $\overline{I}$, so the calculation of the last region of not more than $\overline{W}$. This explains to a meet either split memory constraints, the algorithm proposed subdivision methods can not produce quality worse subdivision.

W_{best} represents the optimal solution with the most overloaded region execution time. Second cycle executed once for each, W_{con} is reduced to half of the search interval. According to the last part of the generated mesh meets the memory constraints, the value of the W_{con} range can be divided into two intervals (0, x), and [x, + ∞), where x is a T, M, u , M_{α} and associated constant. Take any $W_{con} \in (0,x)$, executes this step be split to meet $s_u > M_{\alpha}$, while take any $W_{con} \in [\ x,+\infty)$, has $s_u > M_{\alpha}$. Also, according to the last part of the generated split execution time constraints are met, can be in the range of W_{con} into two intervals (0, y) and [y, + ∞), where y is a T, M, u, M_{α} and related constants.

Take any $W_{con} \in$ (0, y), after the implementation of the resulting segmentation satisfy $w_u > W_{con}$, at the same time take any D$W_{con} \in [y,+\infty)$, both $w_u \le W_{con}$. The algorithm, W_{con} is bound to converge to the larger of x and y that, classification discussed below.

(1) x ≥ y, W_{con} converges to x. If W_{best} <x, from the above said, take $W_{con}=W_{best}$ can also generate a feasible solution, which meets the memory constraint and W_{con} is a minimum value of W contradiction. Therefore W_{best} =x. This indicates that W_{con} converges to W_{best}.

(2) x <y, where W_{con} converges to y. If W_{best} <y, with $W_{con}=W_{best}$ substitution for the first three cycles, from the above to get $W_{\max} \le W_{best}$. Thus $w_u \le W_{con}$, but when W_{con} <y, $w_u > W_{con}$, contradictory, so $W_{con} \ge$y. To W_{con}=y into the first three for loop, $W_{\max} \le W_{con}$, so W_{best} =y, indicates that W_{con} converge to F.

In summary, W_{con} converges to W_{best}. It can be introduced algorithm converges to the optimal split.

We consider the system configuration, there is a consumption of the ECU external communication between the software components. There is a the communications matrix that is between software component, when the software components are not in the same ECU, as shown in Table 1.

TABLE 1 SWC Communication Matrix

	SWC1	**SWC2**	**SWC3**	**SWCn**
SWC1	—	—	—	—
SWC2	—	—	—	—
SWCn	—	—	—	—

At this point, we have been a group of the above A and B in the case where the relative optimum division I, each A is a B and a corresponding software component to the ECU mapping. According to Table 1 for each partition I can calculate the partition communication software components under total consumption. At this point we can compare each group resulting communication cost, whichever is the smallest division of the total communication cost as the mapping division. In this case, the storage space, and a communication amount of execution time, this maps to achieve an optimal partition.

V. CONCLUSION

For the current AUTOSAR system design, ECU mapping algorithm commonly used existing problems, the paper to be a detailed analysis, and gives the improved algorithm. Through the analysis, it can be seen,

Whether or storage overhead in the algorithm and timing of balance in the distribution of resources, etc. The improved algorithm than the original algorithm has been greatly improved.

ACKNOWLEDGMENT

This work was supported by Scientific research projects of Heilongjiang province education department (No.12531189), Harbin reserve Talent project (2014RFQXJ073).

REFERENCES

[1] AUTOSAR Development Partnership. AUTOSAR – Current results and preparations for exploitation. 7th EUROFORUM conference 'Software in the vehicle',Stuttgart 2006

[2] H.Heinecke. Automotive system design - challenges and potential. Design, Automation and Test in Europe. 2005:656~657

[3] K Klobedanz, C Kuznik, A Thuy, et al. Timing modeling and analysis for AUTOSAR-based software development: a case study[C]. 2010: 642-645.

[4] K Lakshmanan, G Bhatia, R Rajkumar. Integrated end-to-end timing analysis of networked autosar-compliant systems[C]. 2010: 331-334.

[5] Senthilkumar K, Ramadoss R. Designing multicore ECU architecture in vehicle networks using AUTOSAR[C]. 2011, (270-275).Advanced Computing (ICoAC), 2011 Third International Conference on.

[6] Caliebe P, Lauer C, German R. Flexible integration testing of automotive ECUs by combining AUTOSAR and XCP[C]. 0004-07-20, 2011, (67-72).Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on.

[7] Vanholme B, Lusetti B, Gruyer D, et al. Highly automated driving on highways: System implementation on PC and automotive ECUs[C]. 0005-07-20, 2011, (1465-1470).Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on.

[8] S Furst. AUTOSAR Newsletter Q2/2010[Z]. AUTOSAR, 2010.

[9] A E Hergenhan, G Heiser. Operating systems technology for converged ECUs[C]. 2008

[10] P Hladik, A Deplanche, S E B Faucou, et al. Adequacy between AUTOSAR OS specification and real-time scheduling theory[C]. 2007: 225-233.

[11] Chen Wei-dong, Yang Jian-jun. Two mathematical models andalgorithm s of internet communications chinese [J]. Joural Computers 1999, 22 (1) : 51-55.

[12] T Scharnhorst, H Heinecke, K Schnelle, et al. Application to engine control at Renault AUTOSAR-Challenges and Achievements 2005[J]. VDI BERICHTE, 2005, 1907: 395.