2009 International Conference on Advances in Recent Technologies in Communication and Computing

Load Balancing in Multi ECU Configuration

Rajeshwari Hegde,
Dept of Telecommunication Engineering,
BMS College of Engineering,
Bangalore, India,
Email:rajeshwari.hegde@gmail.com.

K S Gurumurthy,
Dept of E & C,
UVCE,
Bangalore, India,
Email:drksgurumurthy@gmail.com

***Abstract*— Electronic Control Units (ECUs) are widely used to improve the comfort and reliability of vehicles. It has become the fundamental building block of any automotive subsystem and is interfaced with electro mechanics counterpart. To meet the system wide requirements, these ECUs are interconnected using the communication infrastructure. Although the communication infrastructure in terms of, predominantly, the CAN based vehicle network took its birth to enable ECUs to work in a coordinated manner in order to support system wide requirements, during the past decade, this infrastructure was also viewed as a potential means to incorporate extensibility in terms of addition of newer ECUs which are built for implementing additional requirements. With this paradigm, the number of ECUs started growing in a steep manner, uncontrolled and as a result, today, it is not hard to see a high segment automotive housing ECUs as large as 75-80. Hence, load balancing mechanisms are needed to ease ECU integration and for efficient utilization of CPU power in ECUs. In this paper, we explain the mathematical approach for load balancing across ECUs on the basis of CPU utilization.**

***Keywords*—AUTOSAR, ECU, Load balancing, OEM.**

I. INTRODUCTION

The importance of electronics in vehicles has greatly increased over the last few years [1]. Modern car is a complex electro-mechanical system whose comfort, safety and performance largely depend on the number of ECUs used and the integration of various functionalities in it. A whole range of electronic functions such as navigation, adaptive control, traffic information, traction control, stabilization control and active safety systems are implemented in today's vehicles. Many of these new functions are not stand-alone in the sense that they need to exchange information and sometimes with stringent time constraints with other functions. [4]. Defects in ECU Software (e.g. driver assistance systems such as steering or braking assistance systems etc.) may have disastrous impacts on the relevant OEM as well as on their suppliers. Defective products may not only cause personal deaths or injuries, but also result in recall actions, producing high costs and causing material damage to the image of all companies involved [7].The automotive supply chain including Automotive OEMs, ECU providers and component providers struggle to cope with an ever increasing functionality implemented on a staggering number of ECUs.[2]. Most of the ECUs currently used are "one box solution for each application" resulting in ECUs of different complexities and capabilities being supplied by different vendors functioning in a single vehicle further adding to the complexity.[5]. To make the situation worse, these vendors also reserve their design philosophy and details as proprietary assets. As a result vehicle manufacturers struggle to cope with an ever increasing functionality implemented on a staggering number of ECUs. Managing the complexity is one of the most important problems to achieve required reliability and performance.[6]. Cost reduction requires integrating functionality from multiple suppliers onto a single ECU, while system integration requires interconnecting several ECU's, sensors, actuators using a network bus (e.g. CAN) and dedicated wiring. Often, the increasing number of ECU's is more a consequence of bad design practice (one ECU per sensor, local redundancy) rather than a real necessity. As a result, buses are needed to replace the otherwise large amount of wiring required to connect the ECU's altogether. In any case, both ECU SW integration and system integration are error prone, so far mostly manual procedures. The real issue in automotives is that no work has been done in design environments to balance the load across ECUs.

The increasing number of ECUs warranted refinements to the communication infrastructure so that the ECUs get integrated with ease. The automotive industry established framework to incorporate peer-to-peer communication stack across ECUs. This stack was further based on OSEK operating system, a significant innovation of early nineties. This innovation along with other standardization initiatives streamlined the process of ECU integration by late nineties.
As automotive OEMs gained the capability to build complex ECUs and integrate them within automotives, the number of ECUs started growing in a steep manner, uncontrolled. This has resulted in the following setbacks:
1. Management of complex network of ECUs - a formidable task.
2. Proprietary nature of the ECU owners.
3. Overall cost of the ECUs and the associated infrastructure becoming a non-trivial fraction of the vehicle cost.
These setbacks are getting addressed by industry initiatives and AUTOSAR (Automotive Open System Architecture) is a recent consortium which is responsible for the standardization of subsystem design and implementation for future vehicle generations. AUTOSAR architecture, as a prime objective, inherently, features mechanisms to reduce the number of ECUs by exploiting the CPU power of the ECUs.
This paper is organized as follows. Section 2 explains the AUTOSAR technical concept. Section 3 deals with load

balancing. Section 4 explains the implementation. Paper is concluded in section 5.

II. AUTOSAR TECHNICAL CONCEPT

The main challenge of the automotive industry is to come up with methods and tools to facilitate the integration of different ECUs supplied by various Tier1 suppliers into the vehicle's global electronic architecture to reduce the complexity and cost of the vehicles. Automotive OEMs(Original Equipment Manufacturer) are facing difficulties in integrating subsystems which are designed and implemented by multiple Tier-1 vendors. In the last ten years several industry wide projects have been undertaken in that direction and significant results have already been achieved. The next step is to build an accepted open software architecture, as well as the associated development processes and tools, which should allow for easy integration of different functions and ECUs provided by car makers and third party suppliers. This is ongoing work in the context of AUTOSAR.[4]. AUTOSAR aims at facilitating the re-use of soft- and hardware components between different vehicle platforms, OEMs and suppliers. To achieve this, AUTOSAR defines a methodology that supports a distributed, function-driven development process and standardizes the software-architecture for each ECU in such a system.[8].

A. *Impact of AUTOSAR on E/E Architecture*

There is a tradeoff between standardization and optimization. Introducing this standardized concept is likely to lead to software and runtime overhead. This overhead may make it necessary to increase microcontroller resources, such as RAM, ROM and CPU performance which would lead to an increase in system cost. One solution to the problem could be to alter the vehicle E/E architecture, by moving away from the current one-ECU-one-function concept to a more centralized concept where several functions are bundled into one ECU.Changing this style of E/E architecture with fewer ECUs but more functionality would save a lot of overhead cost such as, Housing, PCBs, voltage regulator, transceivers etc. Introducing AUTOSAR software will ease integration work significantly. In that sense the success of the AUTOSAR standardization could be a decisive factor for further growth in vehicle software functionality [13]. With the increasing distribution of functions over several ECUs in a car, the importance of end-to-end timing (and deadlines) is also increasing. Industrial standardization efforts such as AUTOSAR have already defined models for capturing such "timing chains" composed of communicating "software components", illustrated in Figure 1.[6]

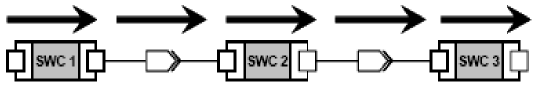

Figure 1: AUTOSAR View on " Timing Chains"

The primary goal of AUTOSAR is not to solve timing problems in particular. AUTOSAR rather defines a software infrastructure for application and basic software, illustrated in Fig. 2 (16).

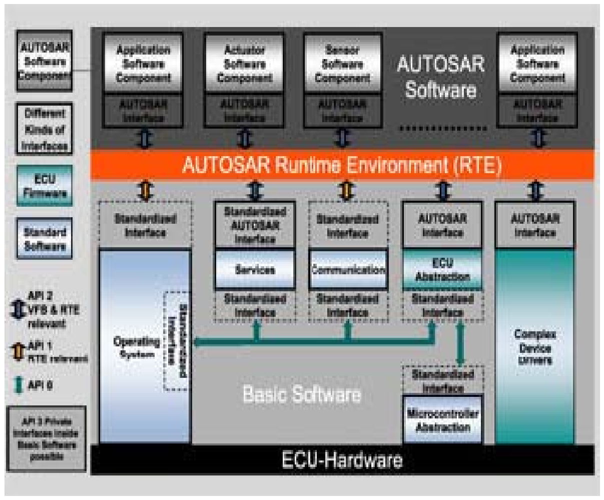

Figure 2: Standardized AUTOSAR Software

III. WHAT IS LOAD BALANCING

Load-balancing, by definition, is dividing the amount of work that a computer has to do between one or more additional computers so that more work gets done in the same amount of time and, in general, all processing get done faster [9]. It is the assignment of work to processors and is critical in parallel simulations. It maximizes application performance by keeping processor idle time and interprocessor communication as low as possible. The problem of load balancing is much more difficult in large distributed systems. Algorithms have to minimize both load imbalance and communication overhead of the application. Additionally they should be efficient themselves and scalable.[14]. In applications with constant workloads, static load balancing can be used as a pre-processor to the computation. Other applications, such as adaptive finite element methods, have workloads that are unpredictable or change during the computation; such applications require dynamic load balancing that adjusts the decomposition as the computation proceeds. Numerous strategies for static and dynamic load balancing have been developed in embedded systems, including recursive bisection (RB) methods, space-filling curve based (SFC) partitioning and graph partitioning. In the migration strategy, each processor works out a schedule for the exact amount of load that it should send to (or receive from) its neighboring processors. Once this schedule is worked out, each processor decides which particular node it should send to or receive from its neighboring processors.[12]. The migration of load then takes place. The scheduling algorithms are mostly iterative and hence there is a startup cost which is usually very high compared with the subsequent cost of transmitting a word.

Consider the load balancing across ECUs. In static load balancing, the load on each ECU is known in advance. Hence the work is equally distributed across ECUs and no extra cost is required for balancing the load [6]. This can be explained using graph theory, where in, vertices represent individual ECU load and the edges represent the amount of load to be transferred from one ECU to another [9].Dynamic Load Balancing across ECUs can improve the utilization of CPUs and the efficiency of parallel computations through migrating workload across CPUs at runtime. Workload migration can be carried out through transferring processes across nearest neighbor ECUs. Iterative strategies have become prominent in recent years because of the increasing popularity of point-to-point interconnection networks. [11]. There are many reasons to institute load balancing across ECUs.
The two most popular are:
1.Response time- With two or more ECUs sharing the load, each of them will be running less of a load than a single ECU alone, there by keeping the response time low.
2. Redundancy-If a load is balanced across 3 ECUs and one of them dies completely, then the other two can keep running and a vehicle will not even notice any downtime.
Any load-balancing solution worth its salt will immediately stop trying to send traffic to the down ECU.
Usually, the load-balancing mechanism aim is to move the running tasks across the CPUs in order to insure that no CPU is idle while some tasks are waiting to be scheduled on other CPUs.

A. Need for Load Balancing

Distributed systems such as automotives can suffer from poor performance due to a bottleneck at overloaded ECUs. To address this performance bottleneck, an adaptive load balancing is used to distribute the load from densely loaded ECUs to scarcely loaded ECUs. Not much research has been done on keeping the load balanced across ECUs. To achieve good performance, it is essential to maintain a balanced work load among all the ECUs. Sometimes the load can be balanced statically. However, in many cases, the load on each ECU cannot be predicted a priory. Dispatching tasks from densely loaded ECUs to scarcely loaded ones to improve the overall performance of the vehicle is both logical and feasible.[3] A schedule of the work load that should be moved between any two ECUs , such that each ECU will have the same load on completion is a challenging task. One way to balance the load is to dispatch the job immediately upon arrival. The best load balancing status occurs when all ECUs are at the point of full utilization, without saturation. Each ECU's work load is proportional to its capacity. Allocating more jobs to a fully utilized ECUs might cause imbalance without improving the overall throughput. Since the data movement between ECUs incurs communication cost, the schedule should give balanced load with minimal data movement. Restricting the data movement to the neighboring ECUs might reduce communication cost. According to dimension Exchange Algorithm, the ECUs can be grouped in pairs and an ECU pair (a, b) with load la and lb will exchange load, after which each will have the load (la+ lb)/2.

IV. IMPLEMENTATION

Much of the load balancing problems can be described using terminology from graph theory. [15]. A graph G has two key components. The vertex set N and the edge set E. Let N be the number of ECUs. Let the ECU graph be represented by a graph(V,E), where V=(1, 2 ,3,…,N) is the set of nodes each representing an ECU and E is the set of edges connecting the nodes. Two nodes i and j form an edge if they share a load. Associated with each ECU i is a scalar l_i representing the load on the ECU. The average load per ECU is

$$L_{avg} = \frac{\sum_{\iota=1}^{N} \lambda_\iota}{N} \qquad (1)$$

The amount of load to be transferred from node i to node j is given by δ_{ij}. The load balancing schedule should make the load on each processor equal to the average load L_{avg}. The conjugate Gradient Algorithm is used to calculate the average load on each ECU. Consider the graph of five ECUs as shown in figure 1.The numbers in bracket indicate the load on each ECU. Let the average degree of the graph be 2. Each ECU has a work load l_i associated with it.

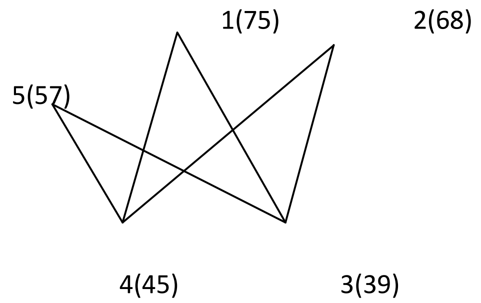

Figure3. Graph of five ECUs with average degree=2.4.

The Laplacian matrix of the graph is given by

$$\begin{pmatrix} 2 & 0 & -1 & -1 & 0 \\ 0 & 2 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{pmatrix} \begin{pmatrix} 18.2 \\ 11.2 \\ -17.8 \\ -11.8 \\ 0.2 \end{pmatrix}$$

$\lambda_1=9+\lambda_5.$ $\lambda_2=5.5+\lambda_5$
$\lambda_3=-1.1+\lambda_5$ $\lambda_4=0.9+\lambda_5$ $\lambda_5=\lambda_5$

The amount of load to be transferred from one ECU to another is given by

$\delta_{14}=\lambda_1-\lambda_4$, $\delta_{13}=\lambda_1-\lambda_3$, $\delta_{24}=\lambda_2-\lambda_4$, $\delta_{23}=\lambda_2-\lambda_3$,
$\delta_{31}=\lambda_3-\lambda_1$, $\delta_{32}=\lambda_3-\lambda_2$, $\delta_{35}=\lambda_3-\lambda_5$, $\delta_{42}=\lambda_4-\lambda_2$,
$\delta_{45}=\lambda_4-\lambda_5$, $\delta_{45}=\lambda_4-\lambda_5$, $\delta_{54}= - \delta_{45}$

The fig.3 shows the load on each ECU before and after load balancing. The average degree of the graph is 2.4. The load balancing algorithm converges after four iterations.

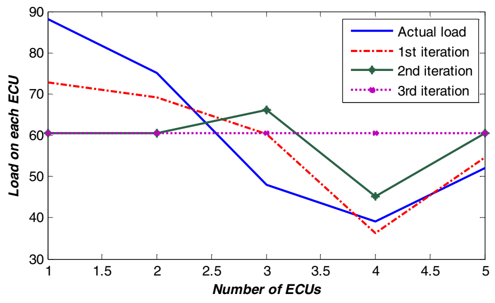

Figure 3:Average load vs number of ECUs (Average degree of the graph=2.4)

It is found from the simulation result that, as the average degree of the graph increases, the number of iterations required to converge also increases. When the average degree of each node is two, the number of iterations required to converge is three.

V. CONCLUSION

The present work is taken up by the authors to have a formal look at Load balancing in Multi ECU Configuration, on the basis of CPU utilization. The load balancing approach reduces the complexity of the automotive system by equally distributing the load across different ECUs. This mechanism in automotives eases the ECU integration by reducing the total number of ECUs. Reduction in number of ECUs provides huge opportunity towards saving cost, reducing complexity and possibility of adding new features using existing computing resources available across ECUs in the vehicle. Advances in the hardware technology like advent of multi-core processors makes it possible to provide enough computing resources on a single ECU to integrate multiple functionalities.

REFERENCES

[1] Christian Wewetzer, Klaus Lamberg and Rainer Otterbach, "Creating Test Patterns for Model-based Development of Automotive Software", SAE International 2006.

[2]. Paolo Giusto , Jean-Yves Brunel, Alberto Ferrari, Eliane Fourgeau, Luciano Lavagno, Alberto Sangiovanni-Vincentelli, "Automotive Virtual Integration Platforms: Why's, What's, and How's", IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02).

[3]. Karen D. Devine 1, Erik G. Boman, Robert T. Heaphy, Bruce A. Hendrickson, "New Challenges in Dynamic Load Balancing", Preprint submitted to Elsevier Science.

[4]. Nicolas Navet, Francoise Simonot-Lion, " Automotive Embedded Systems Handbook", CRC Press.

[5]. Rajeshwari Hegde, K S Gurumurthy, " Model Based Approach for the Integration of ECUs", ICCSE, World Congress on Engineering 2008, U.K.

[6]. Daehyun Kum, Gwang-Min Park, Seonghun Lee, Wooyoung Jung, "AUTOSAR Migration from Existing Automotive Software", International Conference on Control, Automation and Systems 2008, Korea.

[7]. Meinhard Erben, Wolf Günther,Tobias Sedlmeier, "The Impact of Automotive Standardization to Liability Risks Arising from Defective Software, Especially under European Law", SAE International Journal of Passenger Cars- Electronic and Electrical Systems **April 2009** vol. 1 no. 1 **38-44.**

[8]. Helmut Fennel, Stefan Bunzel, Harald Heinecke, Jürgen Bielefeld, Simon Fürst, Klaus-Peter Schnelle, Walter Grote, Nico Maldener, Thomas Weber, Florian Wohlgemuth, Jens Ruh, Lennart Lundh, Tomas Sandén, Peter Heitkämper, Robert Rimkus, Jean Leflour, Alain Gilberg, Ulrich Virnich, Stefan Voget, Kenji Nishikawa, Kazuhiro Kajio, Klaus Lange, Thomas Scharnhorst, Bernd Kunkel, "Achievements and Exploitation of the AUTOSAR Development Partnership", CTEA 2006.

[9].http://webhosting.devshed.com/c/a/Web-Hosting-Articles/What-is-Load-balancing-and-Do-I-Need-It/

[10] Can Static Load Balancing Algorithms Be Appropriate in a Dynamic Setting? http://www.dl.ac.uk/TCSC/Staff/Hu_Y_F/MEETING/TALKS/hendrickson.ps.gz.

[11]. Cheng-Zhong XU amd Francis C.M. Lau, "Iterative Dynamic load balancing in multicomputers", Journal of Operation Research Society, Vol. 45, No. 7, July 1994, pp,786-796.

[12]. Y. F. Hu, R. J. Blake and D. R. Emerson, An Optimal Migration Algorithm for Dynamic Load Balancing, Concurrency-Practice and Experience, Journal Article, 1998.

[13]. AUTOSAR, www.eu.necel.com.

[14]. Ralf Diekmann, Burkhard Monien, Robert Preis, " Load Balancing Strategies for Distributed Memory Machines", In:F Karsch, H.Satz(ed.): Multi-Scale Phenomena and their Simulation, World Scientific, 1997 (to appear).

[15]. Y F Hu, R J Blake, "An optimal dynamic load balancing Algorithm", citeseer.ist.psu.edu/121199.html, 1995.

[16]. www.autosar.org