PAPER

An Effective GA-Based Scheduling Algorithm for FlexRay Systems*

Shan DING[†a)], ***Nonmember***, **Hiroyuki TOMIYAMA**[††], ***and*** **Hiroaki TAKADA**[††], ***Members***

SUMMARY An advanced communication system, the FlexRay system, has been developed for future automotive applications. It consists of time-triggered clusters, such as drive-by-wire in cars, in order to meet different requirements and constraints between various sensors, processors, and actuators. In this paper, an approach to static scheduling for FlexRay systems is proposed. Our experimental results show that the proposed scheduling method significantly reduces up to 36.3% of the network traffic compared with a past approach.

key words: *real-time systems, distributed embedded systems, FlexRay, genetic algorithm*

1. Introduction

Microprocessor-controlled electro-mechanical systems have been used to replace mechanically linked hydraulic steering and braking of cars from over a decade [1]. Some other computerized vehicle-control applications such as adaptive cruise control, collision avoidance, and autonomous driving are also being developed. These applications will be realized as real-time distributed systems requiring dependable interaction among sensors, processors and actuators.

The FlexRay system [2] is a communication system developed for the next generations of automobiles by a consortium founded in 2000 by BMW, DaimlerChrysler, Motorola, and Philips Semiconductors. The core of the FlexRay system is the FlexRay communication protocol. It has been designed for the high data transmission rates required by advanced automotive control systems. It consists of time-triggered clusters, such as drive-by-wire in cars, in order to meet different requirements and constraints between various sensors, processors, and actuators.

The FlexRay system is a time-triggered architecture providing a computing infrastructure for the design and implementation of dependable distributed embedded systems. The communication in this architecture is based on a fault-tolerant time-triggered protocol (TTP) [3]. Pop et al have been developed scheduling strategies using TTP as a communication protocol for distributed real-time systems [4], [5]. There are two basic approaches for handling tasks in real-time applications [6]. In the event-triggered approach, activities are initiated whenever a particular event is noted. In the time-triggered approach, activities are initiated at predetermined points in time. Two approaches can be used together inside a few certain applications. Pop et al [7] have proposed an approach for multicluster distributed embedded systems consisting of time-triggered and event-triggered clusters, interconnected via gateways by using the worst-case response time analysis of the application for the controller area network (CAN).

In [7], the schedulability analysis is outlined in Fig. 1 (a). In first step, the application is partitioned on the time-triggered cluster (TTC) and event-triggered cluster (ETC), and processes are mapped to the nodes of the architecture (i.e., mapping). In second step, the mapping of messages are combined into a frame in order to be transmitted to the bus(i.e., frame packing). In the last step, the released time of tasks and the timing of sending messages are determined (i.e., time scheduling). For each step, a given set of parameters is leading to find out if a system is schedulable, that is, all the time constraints are met. If application is unschedulable in one of these steps, the set of parameters must be changed until obtaining the approximate optimal solution. If after these steps the application is unschedulable, the approach conclude that no satisfactory implementation could be found with the available amount of resources.

In general, based the mechanism of the above approach, it tends to find only locally optimal solutions. In

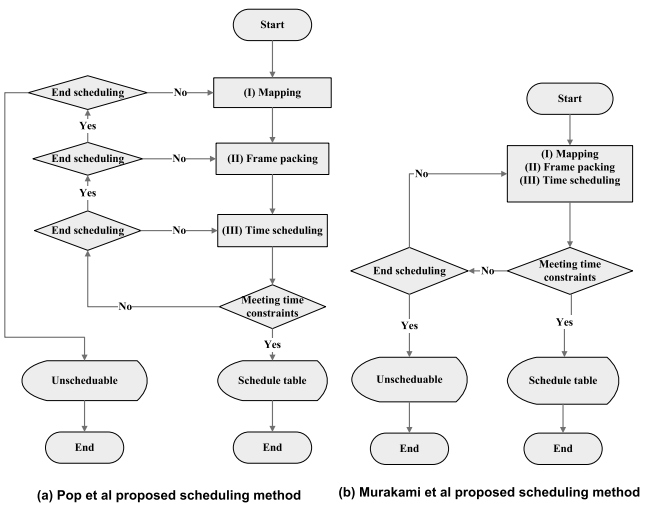

Fig. 1 Comparison of the scheduling methods.

Manuscript received September 4, 2007.
Manuscript revised February 4, 2008.

[†]The author is with College of Information Science and Engineering, Northeastern University, Shenyang, P.R. China.
[††]The authors are with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan.

*This paper was presented at EMSoft 2005.

a) E-mail: dingshan@ise.neu.edu.cn

DOI: 10.1093/ietisy/e91–d.8.2115

addition to the characteristics of objects, more time is necessary to be carried out searching. Pop et al have supposed that all the processes belonging a process graph have the same period. Moreover, such an optimization problem is *NP* complete, thus obtaining the optimal solution is no feasible. Pop et al have proposed two frame-packing optimization strategies, one based on a simulated annealing (SA) approach, the other is based on a greedy heuristic to explore the design space. The main limitation of SA-based methods is the difficulty in tuning the control parameters. Since the complexity of a system on which implements several difference kinds of applications, this model is hard to apply in the real world.

An optimization procedure based on a SA has been proposed by Murakami et al [8]. For various sensors, processors, and actuators with different execution period, it is necessary to develop an efficient scheduling method for static segment of communication cycle in the FlexRay system. In this paper, a more practical and flexible approach to static scheduling method for the distributed automotive control system which integrate FlexRay and CAN systems has been proposed. They supposed that all the processes belonging a process graph can have difference periods. Moreover, they also took into account the influence of system load on the scheduling method.

In this paper, we propose an effective GA-based scheduling algorithm for FlexRay Systems that can include CAN system which be treated as a FlexRay node. The advantages of a GA-based approach depends heavily on how well the various components of GA incorporate the salient features of the problem under consideration [9]. To evaluate the effectiveness of this approach, we have chosen a representative safety-critical application to simulate as case study. Our experiments show the GA can be applied to such kinds of application to find a schedule better than SA approach.

There has been much attention surrounding hardware-software co-design techniques. In [10], they studied the configuration problem which consists of two sections: specifying the hardware capacities of the processing elements and statically allocating the software tasks to them. This technology can be applied in allocating clusters of tasks to any processor for optimization of scheduling subtasks and communication.

The remainder of this paper is organized as follows: in Sect. 2 the application model and system architecture are presented. In Sect. 3 the problem is defined. Section 4 discusses in detail our GA approach. In Sect. 5, experimental tests are carried out. Some concluding remarks follow in Sect. 6.

2. System Architecture

2.1 FlexRay Systems

The FlexRay system architectures consisting of nodes are connected by broadcast communication channels. The FlexRay protocol is a dual channel protocol. FlexRay systems can be configured as single-channel or dual channel bus network, a single-channel or dual-channel star network, or in various hybrid combinations of bus and star topologies. Figure 2 shows one kind of topology configuration of the communication network as a dual bus.

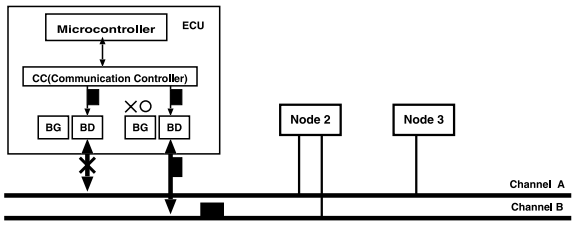

Fig. 2 The node architecture and the dual bus topology configuration.

Each node consists of a microcontroller (host), a communication controller (CC) a bus driver (BD), and bus guardian (BG). The microcontroller is the part of an ECU where an application is executed, and can be separated by the Controller Host Interface (CHI) from the FlexRay protocol engine. The bus guardian is an electronic component that protects a channel from interference caused by communication that is not temporally aligned with the cluster communication schedule by limiting the times that the attached communication controller can transmit to those times allowed by that schedule.

In the FlexRay protocol, media access control is based on a recurring communication cycle. Within one communication cycle FlexRay offers the choice of two media access schemes. One is a static time division multiple access (TDMA) scheme, and the other is a dynamic mini-slotting based scheme. TDMA can partition the bandwidth of the channel in the time domain - [11] provides a "holistic" schedulability analysis in the context of distributed real-time systems. The portion of the communication cycle where the media access is controlled via a TDMA scheme called static segment, which is described as follows.

Objectives of the static segment [12]:

- Deterministic communication behavior in the time domain.(e.g., the released time of tasks)
- Global time implemented by a fault tolerant clock synchronization algorithm.
- Immunity against accepting error-free sub-sequences of a message as valid messages (i.e. short message rejection).

In FlexRay systems, a structure used by the communication system to exchange information within the system is called a frame. A node is a logical entity connected to the network that is capable of sending and/or receiving frames. An interval of time within the static segment of the communication cycle is called static communication slot. Within the during of this slot, access to a communication channel is granted exclusively to a specific node for transmission of a frame. Frame is sent when the slot number corresponds to frame ID. In the static segment, a periodic transmission of the fixed length data is guaranteed. Static slot messages can

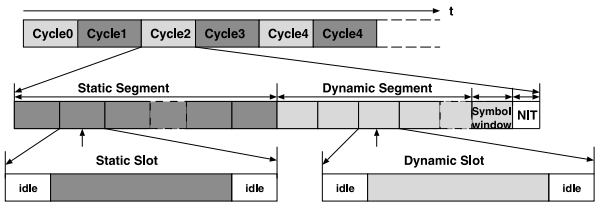

Fig. 3 The communication cycle.

be protected by BG. From the above characteristics, real-time application messages should be allocated to static segments in order to ensure that it can meet its time constraints. In this paper, we focus on the scheduling method for static segments of FlexRay systems.

Because the communication channel is a broadcast channel, a message sent by a node is received by all the other nodes. Node can transmit only during a predetermined time interval, so-called TDMA. In a slot, a node can send several messages packaged in a frame. The sequence and length of the slots are the same for all TDMA rounds. However, the length and contents of the frames may change. Whether the frame can be saved into a node or not is decided by frame ID.

In the FlexRay protocol, media access control is based on a recurring communication cycle. As mentioned above, FlexRay offers a static TDMA scheme and a dynamic mini-slotting base scheme within a communication cycle. Figure 3 shows one complete instance of the communication structure of a communication cycle. Cycle counter, the number of the current communication cycle, ranges from zero to 63.

The periods of tasks and messages are assumed to be the value 2^n(n = 0,1,2,. . .) times at FlexRay communication cycle. The main reason for this assumption is to avoid the time violation between messages. For example, if m_1 and m_2 have periods of 2, 5 respectively, it can not avoid a time violation at communication cycle. In an actual design, it is necessary to adjust the communication cycle and the period of the application in FlexRay systems.

2.2 Application Model

In this paper, we model an application to be described as directed acyclic graphs, where a node is a task and the directed arcs are dependencies between tasks. The message represents dataflow between tasks. N_i represents a node in FlexRay or CAN. Each task N_i is mapped on a processor, and has the maximum execution time C_i and a period T_i on that processor. The designer can provide manually such maximum execution time, or tools can be used in order to determine the maximum execution time of a piece of code on a given processor.

For each message, we know its size and its period, which is identical with that of the sender process. The task that processes message from a sensor is called input task N_{in}. The task that processes message in a actuator is called output task N_{out}. The task in the middle layer of graph that achieves either computing or relay is called task N_{mid}. Since messages between tasks in the same processor can be transmitted by the buffer of the processor, after sent by source task, the message can be used by destination task immediately. Such kind of message is called messages in the node.

3. Problem Description

This paper addresses the same problem as proposed in [10]. In order to indicate our algorithm, we give a brief description of the problem as follows. An application consists of a set of task graphs $G = (N, E)$, where $N = \{n_1, \ldots, n_m\}$ is a finite set of task vertices (i.e., nodes) , E is a finite set of message edges representing connections between these nodes. Each node n_i is allocated to a uncertain processor, and has a known worst case execution time C_i, a period T_i, a deadline D_i. Output is a schedule comprising identical bus. A schedule includes following items need to be determined.

1. Periods of tasks and messages.
2. Nodes that processed by each middle tasks.
3. Sending message node of FlexRay slot.
4. Frame packing for FlexRay message.
5. Receiving message node of FlexRay slot.
6. The released time of each task.

As the primary objective, we construct a schedule meeting all deadlines of task groups and performance goals of the embedded application. The secondary objective is that the schedule can optimize communication buses to minimize hardware cost. Since bin-packing problem is known as $NP - complete$[13], in this paper, we propose a GA for static scheduling based on the following advantages.

1. GAs have broad applicability and are intrinsically suitable for parallel implementation.
2. In certain cases, GAs can find the global optimum of a problem with very high probability.
3. Their performance is robust in many settings.

Constraints

As mentioned above, an application should meet its time constraints in static segment of communication cycle. Before the time constraints are described, some preliminaries are defined.

Response Time (RT): To the output task processes, messages are sent by the input task processes through the communication routes. RT is the time from beginning of all input task processes that inflected by the output task processes, until end of the output task processes.

Freshness Time (FT): Messages are sent by the input task processes through the communication routes. FT is the time from beginning of input task processes until the end of all output task proceesses that be inflected by these messages.

It should be noted that RT and FT may difference in case of multirate systems. Figure 4 shows the difference

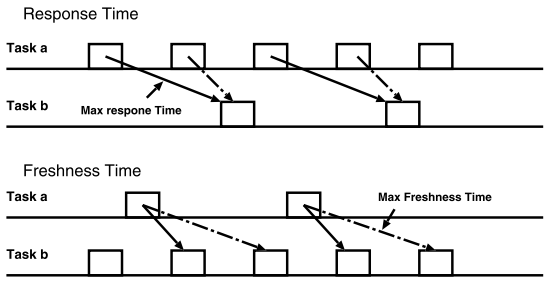

Fig. 4 The difference between respone time and freshness time.

between respone time and freshness time.

Maximum Response Time: The maximum response time of all input tasks processes.

Maximum Freshness Time: The maximum Freshness time of all output tasks processes.

Based on above preliminaries, four time constraints are defined as follows;

Response Constraint: For certain route, the maximum response time must be less than a given time.

Freshness Constraint: For certain route, the maximum freshness time must be less than a given time.

Synchronous Input Constraint: To these routes which have the same output task, the maximum difference of feshness time of these routes must be smaller than a given time.

Synchronous Output Constraint: To these routes which have the same input task, the maximum difference of response time of these routes must be smaller than a given time.

In this paper, we assume that all the constraints are equal to deadline of task graph. Besides above time constraints, we define another constraint by considering character of static segment.

Slot Redundancy: The number of the slot not used at the end of the communication cycle continuously expresses the degree of empty slots in the schedule. The larger number unused slots is, the higher the slot redundancy is.

4. Genetic Algorithm for Scheduling Problem

GA is an example of the meta-heuristics which have been successfully applied to a variety of problems. Since the search space is large and has a complex structure, finding a solution by genetic search is appropriate. Our GA requires the definition of a set of genetic operations and an evaluation function as follows:

4.1 Coding Scheduling Individual

Figure 5 shows the structure of an individual. A string is used for showing which processor the node belongs to. The length of this string is the number of nodes in the task graph.

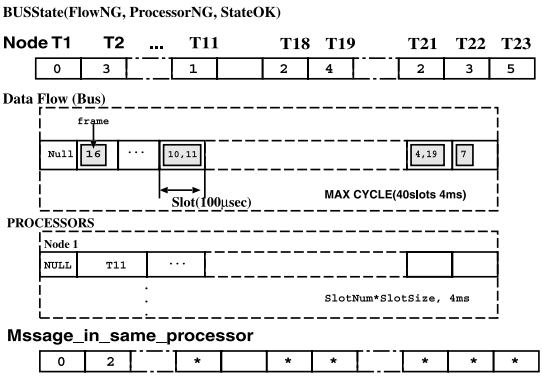

Fig. 5 An individual.

The *i*th number in the string expresses that the *i*th node belongs to processor *j*. (e.g., T11 belongs to processor 1)

Another string is used for showing the messages (i.e., edges) in the communication cycle. Because the communication cycle can be transformed as the product of number of slots and the slot size, the length of these strings is the product of number of slots and the slot size. We assume that the slot size is 32 bytes in this paper. If the communication cycle time is Δ_{cycle} and each transmission slot time is Δ_{slot}, the number of slots can be calculated as $\Delta_{cycle}/\Delta_{slot}$. As mentioned above, a structure called frame can load messages in the dataflow (bus). Several messages can be assembled into one frame when they are sent by the same node and sum of their sizes is not larger than the slot size. Several strings are prepared for scheduling of processor to record when and how long the nodes execute.

Individual states include three states, processorNG, busNG, and stateOK. The processorNG means that the scheduling of processor is failed. The busNG means that there is no enough number of slots for message transmission. If no NG state is shown, then the individual state will turn to stateOK and calculate the fitness value.

The process of generating individual is shown in Fig. 6. It is easy to judge whether dataflow (bus) state is NG or not. Schedulability of processor is judged in the portion of scheduling for each processor. The process of message transmission can be considered two portions, i.e., send and receive. As a synchronous algorithm, the sending node is executed before dataflow (bus) slot number (i.e., time) in the processor which it belong to, and receiving node is executed after dataflow (bus) slot number (i.e., time) in the processor which it belong to. When there is no time to execute in the processor, the individual state will become processorNG state. When the message is transmitted in the same processor, it will not display in the dataflow. It must be recorded in the message_in_same_processor string instead. This is because it will be treated inside of the processor. The execution time of a node will be recorded in the processor string. When a node has more than one message to send, execution time of the node must be moved in front of all messages it send. Finally, based on the node order appeared in the routes, we check the lost nodes in the processor strings. If

```
Generate Individual()
{
    for each edge ei in edge set Ei;
    do {
        if (two vertexes of ei in difference processors){
            Select a slot in dataflow randomly;
            while (unsatisfy contraints condition){
                Slip one slot;
                if (dataflow is enough){
                    BUSState = FlowNG;
                    break; }
            }
            if (satisfy contraints condition){
                Insert the edge into dataflow;
            }
        }else{
            write message_in_same_processor;
        }
    }while(Ei != ϕ)
    scheduling for each processor;
}
```

Fig. 6 The algorithm for generating an individual.

there are lost nodes, it is necessary to schedule the lost nodes in their processors that they belong to. We insert lost node in the front of the node that appears to the individual by the same route.

4.2 Generate Initial Generation

Individuals are generated randomly as many as population size. Individual value is calculated by an evaluation function. In the case the same individual is generated, it is deleted using differentiation of the same individual and a new different individual is obtained using mutation. Then the generation will be formed by sorting individuals in ascending order.

4.3 Evaluation and Optimization Strategy

Our evaluation function captures the "degree of schedule" for a certain individual. The evaluation result is used as an individual fitness value. For a given application, decreasing individual cost value means that optimization result of this individual is good.

To the individual, there are two kinds of NG states as discusses before. When the individual state is processorNG or busNG, it does not need calculate its cost function since it is not schedulability. However, in order to differentiate the "degree of schedulability" of the individual, the general constraints are checked such as the frame size is smaller than the slot size or not, and the messages in frame is sent by the same task or not. When an individual is in stateOK state, we calculate the optimization cost value.

According to definition of RT/FT, the routes which only include same period nodes or from longer period nodes to shorter period nodes must be calculated by response constraint for measuring the synchronous input/output constraints. However, when the routes is from shorter period nodes to longer period nodes, it is necessary to use the freshness constraint for measuring the synchronous input/output constraints.

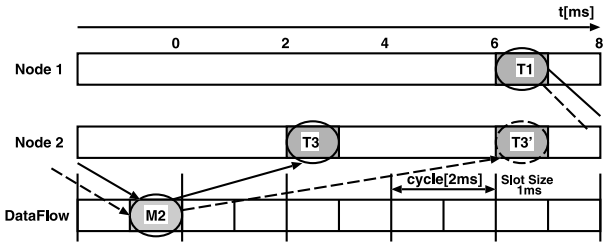

Fig. 7 The example for calculating freshness constraint.

We assume that all of the four constraints described in Sect. 3 are equal to the deadline of the task graph. If the route cost is larger than deadline, we give a penalty as the following formula.

$$P = \sum_{i=1}^{n} f_i = \begin{cases} \dfrac{R_i}{D_i}, & if \quad R_i \le D_i \\ \dfrac{(R_i - D_i)}{D_i} * \alpha + \beta, & if \quad R_i > D_i \end{cases}$$

f_i is the cost value of ith routes, R_i is the execution time of ith route which will be calculated by freshness time constraint or response time constraint. D_i is the deadline of the ith route. α and β are coefficients. This formula demonstrates that, when the execution time of ith route is smaller that its deadline, the P value is smaller than 1, since it is the proportion of its execution time to its deadline. However, when the execution time of ith route is larger that its deadline, the gradient is enlarged by α times, furthermore, the constant value β is added as more penalty. In this paper, we set α and β to 10 and 2, respectively.

In addition to the slot redundancy, the individual fitness value is calculated as follows,

$$Cost = P * w_1 + N_{slot} * w_2$$

N_{slot} is slot redundancy, w_1 and w_2 are coefficients that means how much the attribute affects the individual fitness value. The coefficient of each attribute is decided by the characteristics of the problem and the policy of the scheduling. We set w_1 and w_2 to 1 and 0.5, respectively. This fitness function is also to be applied to the SA [10].

We calculate the freshness time constraint as follows. The route can be expressed by a task sequence and the message sequence alternately like input task → message → task → message $\cdots$ → message → output task. According to the definition of RT/FT, the freshness constraint is calculated depending on the execution timing of the output tasks.

Figure 7 shows a freshness time constraint calculating case. The route can be expressed as T1 → m2 → T3. Periods of Task $T1$ and T3 are 8 ms and 4 ms respectively. There are two routes from input task $T1$ to output task $T3$. They are expressed by solid lines and dotted lines. The cost values (i.e., time) of solid lines and dotted lines are 5 and 9 respectively. As a result, the maximum freshness time of this route is 9. The response time constraint can be calculated similarly, but depending on the execution timing of the input tasks instead.

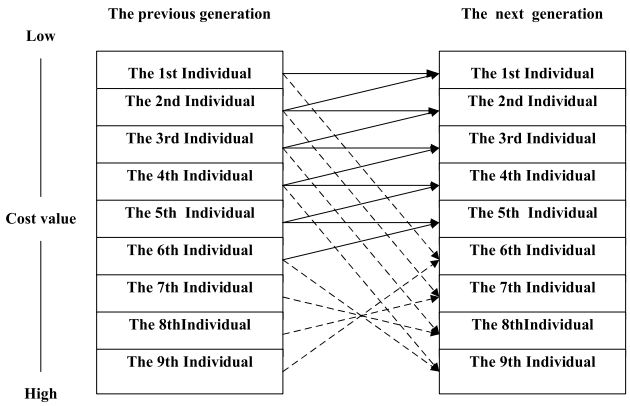

Fig. 8 The select operation.

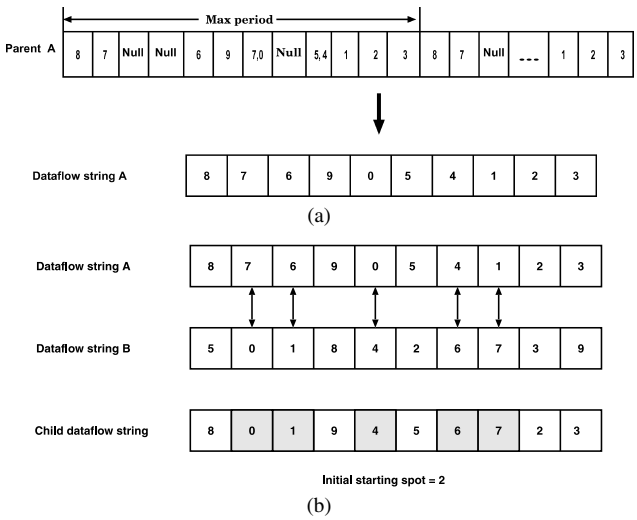

Fig. 9 The crossover operation.

4.4 Selection

The selection operation is performed as shown in Fig. 8. All individuals in the population are sorted out according to their fitness, so the first individual is the best in this generation. The following operation is performed.

In our method, all individuals are selected for mating. We perform crossover to the ith individual of the next generation, where $i = 1, 2, \cdots, [(p+1)/2]$ and p is the population size. Then the individuals with lower fitness could be generated because the fitness of parents are lower. Also, we perform crossover to the jth individual and the $(p+1\text{-}j)$th individual to generate $[p/2]$ individuals of the next generation, where $j = 1, 2, \cdots, [p/2]$, then the minimal fitness of individuals in the next generation could be decreased or equal to the minimal fitness of individual in previous generation. Because the fitness of his parents is the lowest in previous generation and if the minimal fitness cost of individuals in the next generation is larger than its parents, minimal fitness of parent individual will be copied to his offspring.

4.5 Crossover

The crossover operation is held between two parents. If the two strings that show to which the node belongs are the same in two individual parents, we conduct crossover operation by randomly choose one of the following operations.

- crossover 1:
  a. Copy the better fitness parent individual to child individual.
  b. Search one slot with two messages in another parent individuals.
  c. Check the two messages in child individual. If the two messages are in two independent slots, merge the two messages in one slot.
  d. Repeat steps b and c, until the end of communication cycle.

- crossover 2:
  In child's string, insert a message at the slot whose number is equal to the average value of two parents'.

- crossover 3:
  Based on the order of the message number appeared in the two dataflow of parents' genes, two dataflow strings are composed without considering a empty slot and message period as shown in Fig. 9 (a). The exchanged set is found and the child individual is generated by the following algorithm.

  Crossover 3
  Input: Dataflow string A=(A_i | 1 ≤ i ≤ n);
  Dataflow string B=(B_i | 1 ≤ i ≤ n);
  Output: Child dataflow string=(C_i | 1 ≤ i ≤ n);
  Procedure:
  Initialize transitive closure sets;
  $\alpha_0=\emptyset$, $\alpha_1=\emptyset$
  select A_i from Dataflow string A randomly;
  do {
  set $\alpha_0=\alpha_0 \cup A_i$
  set $\alpha_1=\alpha_1 \cup B_i$
  if ($\alpha_0=\alpha_1$) then break;
  A_i = index(B_i)
  } while($\alpha_0!=\alpha_1$)
  $C_i=A_i \oplus \alpha_1$
  return Child dataflow string

  The $\oplus$ operator means insertion of B_i into the corresponding positions of B_i in C_i. How the replacement is taken place is shown in Fig. 9 (b). Exchanging genes is done by exchanged set in the child individual.

When the processors that a node belongs to are different in its two parents, the processor which the node belong to will be decided once again randomly. When an edge appears in one of its parents, we insert the same slot number as the number of its parents. When an edge do not appear in either

of its two parents, we insert the edge into the child individual gene randomly.

4.6 Mutation

The exchange mutations are defined as follows,

- Search for the slots unsatisfying constraints. Suppose that there are two messages unsatisfying constraints messagee_a and messagee_b.
- If the period of messagee_a is longer than that of messagee_b, then messagee_b occupies the slot of messagee_a. Messagee_a must find its new slot by scanning the static segment.

  If the period of messagee_a is shorter than that of messagee_b, then messagee_b must find its new position by scanning the static segment.
- If there is no a slot unsatisfying constraints, randomly select two slots in static segment, exchange them in each cycle.

4.7 Differentiation of the Same Individual

After crossover or mutation, we sort the individuals. If the same individual is found, then we operate mutation to it in order to generate a new different individual.

5. Case Study and Experiments

To demonstrate the effectiveness of the GA-base scheduling algorithm, we conducted a set of experiments. We assume the FlexRay communication protocol having a bandwidth of 250 kb/s and a bandwidth of 100 μsec for the transmission slots. In this paper, we assume that the deadline of the node and edge is equal to the period of the node and edge. The developed GA is implemented using C language on the Linux.

A safety critical application with hard real-time constraints, to be implemented on a FlexRay based architecture, includes a vehicle adaptive cruise controller (ACC), electric power steering (EPS), and traction control (TC) as detailed in [14].

5.1 A Realistic Application

The ACC application automatically maintains a safe following distance between two cars, while EPS uses an electric motor to provide necessary steering assistance to the driver. The TC application actively stabilizes the vehicle to maintain its intended path even under slippery road conditions. Considering the synchronous input/output constraints, we model the application as Fig. 10.

Task group a, b and c have periods of 2 ms, 4 ms and 8 ms, respectively. The edge F4 and F23 have periods of 4 ms, 8 ms respectively. There are six processors available to allocate the nodes in this application. In Fig. 10, the nodes

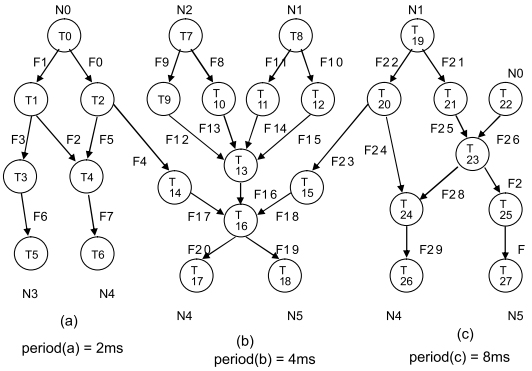

Fig. 10 A safety critical application model.

Table 1 The execution time of nodes and the message sizes.

task	e(μsec)	task	e(μsec)
T0	150	T14	300
T1	350	T15	200
T2	200	T16	400
T3	250	T17	350
T4	300	T18	400
T5	200	T19	400
T6	400	T20	300
T7	300	T21	600
T8	350	T22	350
T9	400	T23	800
T10	250	T24	300
T11	400	T25	400
T12	300	T26	300
T13	400	T27	400

Edge	Size
E0	12
E1	12
E2	20
E3	12
E4	20
E5	12
E6	12
E7	10
E8	12
E9	10
E10	10
E11	12
E12	20
E13	12
E14	12

Edge	Size
E15	10
E16	12
E17	12
E18	12
E19	10
E20	12
E21	20
E22	20
E23	12
E24	20
E25	12
E26	20
E27	10
E28	22
E29	20
E30	12

Table 2 The network traffic of the optimum schedule.

Slot	Edge	Slot	Edge	Slot	Edge	Slot	Edge
0	4	20	20	40	4	60	20
1	9, 13	21	24	41	9, 13	61	null
2	3, 26	22	3	42	3	62	3
3	11, 15	23	28, 30	43	11, 15	63	null
4	17	24	null	44	17	64	null
5	7	25	7	45	7	65	7
6	23	26	null	46	null	66	null

labeled Ni, means that this node is allocated to ith processor. The other nodes can be allocated to any processor freely.

Table 1 summarizes the execution time of nodes and the various message attributes affecting network topology generation. The names of nodes and edges are in column "task" and column "Edge", respectively. The column "e(μsec)" and column "Size" show the execution time (μsec) of nodes and message size (in byte) of edge, respectively.

Parameters for GA were set as follows. Population size was 28, generation number was 3000, optimum value was 17.09 (taking 1888 seconds of optimization time). Table 2 shows the network traffic of an optimum schedule. In Table 2, column "Slot" and column "Edge" show the slot number and edge name. Null means there is no edge in the slot. Table 2 shows the minimum number of slots can be used for

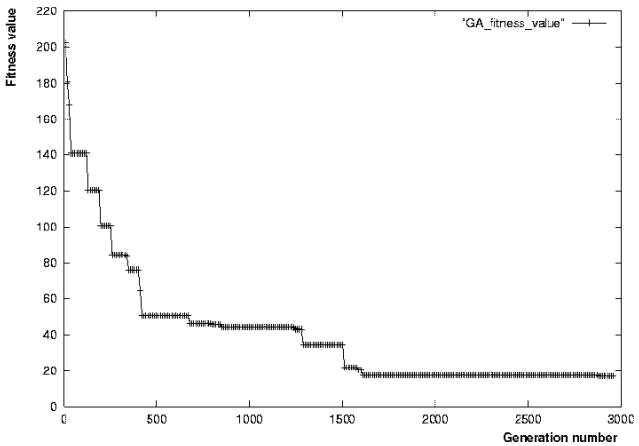

Fig. 11 The relation between the best fitness value and the generation number.

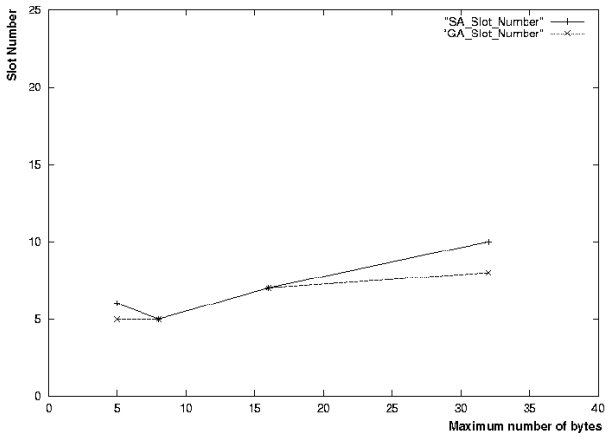

Fig. 12 The relation between message size and number of slots.

schedule is 7. In [8], the minimum number of slot can be used for schedule was 11. This experimental result shows that the proposed scheduling method significantly reduces up to 36.3% of the network traffic compared with the SA approach. This schedule also can meet all deadlines of the task group a, b and c.

The Relationship between the best fitness value of generation and generation number is shown Fig. 11.

Next, we tested our algorithm with respect to the maximum message size allowed. For the results depicted in Fig. 12 we have assumed the maximum message size as 5, 8, 16 and 32 bytes. Figure 12 shows that the number of slots used for schedule decreases with the decrease of the maximum number of bytes in a message.

We changed the graph for general experimental purpose. We considered four graph architectures consisting of 15, 20, 25, and 30 nodes. Execution time, periods and message size were assigned randomly within certain intervals. Figure 13 shows the number of slots used for schedule satisfying the constraints and deadlines. Good results were obtained by both algorithms. However, GA used a fewer number of slots than SA.

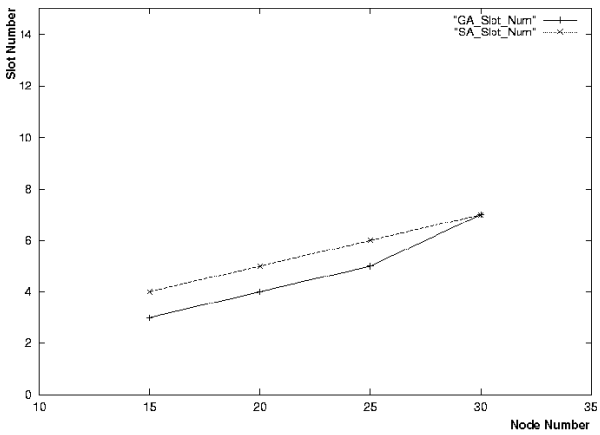

Fig. 13 Maximum number of nodes.

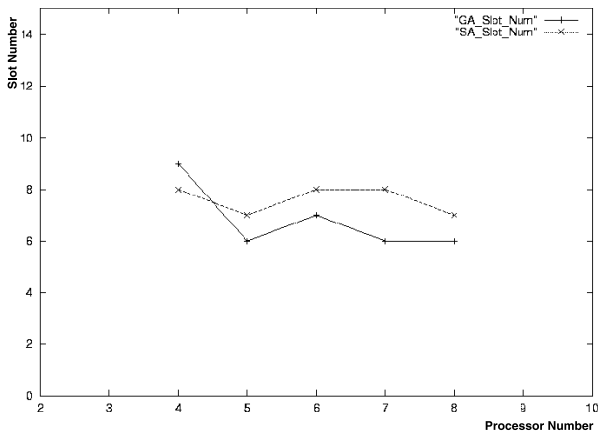

Fig. 14 Maximum number of processors.

Moreover, we have assumed that 28 nodes were allocated to 4, 5, 6, 7 and 8 processors in a graph respectively. Figure 14 shows that GA is better than SA [8] at almost all kind of conditions. Experimental results show that our developed method is able to efficiently produce good quality results.

6. Concluding Remarks

An advanced communication system, the FlexRay system, has been developed for future automotive applications. In this paper, an approach to static scheduling for FlexRay systems is proposed. Some new concepts are proposed for measuring the "degree of schedulability". Optimization is performed on reducing the network traffic while meeting deadlines and satisfying the constraints which have been identified. Some genetic operations are proposed in the GA. A safety critical application that includes ACC, EPS and TC, has been studied in this paper. Our experimental results show that the proposed scheduling method significantly reduces up to 36.3% of the network traffic compared with a past approach. The effectiveness of the developed GA is confirmed by the experiments.

In the future, we will apply the GA-based scheduling

algorithm to real-world applications.

References

[1] E.A. Bretz, "By-wire cars turn the corner," IEEE Spectr., vol.38, no.4, pp.68–73, April 2001.
[2] R. Mores, G. Hay, R. Belschner, J. Berwanger, C. Ebner, S. Fluhrer, E. Fuchs, B. Hedenetz, W. Kuffner, A. Kruger, P. Lohrmann, D. Millinger, M. Peller, J. Ruh, A. Schedl, and M. Sprachmann, "FlexRay — The communication system for advanced automotive control systems," Proc. SAE World Congress, 2001–01–0676, 2001.
[3] H. Kopetz.k.Real-Time Systems, Design Principles for Distributed Embedded Applications, Kluwer Academic Publishers, Boston, 1997.
[4] P. Pop, P. Eles, and Z. Peng, "Schedulability-driven communication synthesis for time triggered embedded system," Proc. 6th International conference on Real-time Computing Systems and Applications (RTCSA'99), pp.287–294, 1999.
[5] P. Eles, A. Doboli, P. Pop, and Z. Peng, "Scheduling with bus access optimization for distributed embedded systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.8, no.5, pp.472–491, 2000.
[6] H. Kopetz, Real-time systems-Design principles for distributed Embedded Applications, Kluwer Academic Publishers, Norwell, MA, 1997.
[7] P. Pop, P. Eles, and Z. Peng, "Schedulability-driven frame packing for multi-cluster distributed embedded systems," ACM Trans. Embedded Computing Systems, vol.4, no.1, pp.112–140, 2005.
[8] N. Murakami, S. Iiyama, H. Takada, M. Kido, and I. Hosotani, "A static scheduling method for distributed automotive control systems," Trans. IPSJ Advanced Computing Systems, vol.48, no.SIG8 (ACS18), pp.203–215, May 2007.
[9] Y. Leung, G. Li, and Z. Xu, "A genetic algorithm for the multiple destination routing problems," IEEE Trans. Evol. Comput., vol.2, no.4, pp.150–161, Nov. 1998.
[10] J.E. Beck and D.P. Siewiorek, "Automatic configuration of embedded multicomputer systems," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.17, no.2, pp.84–95, 1998.
[11] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," Microprocessing and Microprogramming, vol.40, no.2-3, pp.117–134, April 1994.
[12] FlexRay Requirements Specification. Version 2.0.2, April 2002, www.flexray-group.com.
[13] D.S. Johnson, "Fast algorithm for bin packing," J. Comput. Syst. Sci., vol.3, no.2, pp.272–314, 1974.
[14] N. Kandasamy, J.P. Hayes, and B.T. Murray, "Dependable communication synthesis for distributed embedded systems," Int'l Conf. on SAFECOMP, LNCS 2788, pp.275–288, Sept. 2003.

Shan Ding received the B. E. degree in Department of Computer Science and Application in Northeastern University, Shenyang, China, in 1989, and the M. E. degree in Department of Electrical and Computer Engineering from Nagoya Institute of Technology of Japan, in 1999. He received his Ph.D. degree in computer science from Nagoya Institute of Technology in 2003. From Sept. 2004 to Aug. 2006, he was a visiting postdoctoral researcher in the Department of Information Engineering, the Graduate School of Information Science, Nagoya University, where he was financially supported by JSPS Postdoctoral Fellowship for Foreign Researchers. He is currently an Associate Professor in College of Information Science and Engineering, Northeastern University, P. R. China. His current research interests include algorithm design and analysis, real-time scheduling theory, and embedded system design.

Hiroyuki Tomiyama received his Ph.D. degree in computer science from Kyushu University in 1999. From 1999 to 2001, he was a visiting postdoctoral researcher with the Center of Embedded Computer Systems, University of California, Irvine, where he was financially supported by JSPS Postdoctoral Fellowship for Research Abroad. From 2001 to 2003, he was a researcher at the Institute of Systems & Information Technologies/KYUSHU. He is currently an Associate Professor with the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. His research interests include system-level design methodologies for embedded systems, computer-aided designed of VLSI systems, and compilers for embedded systems. He has served on the organizing and program committees of several premier conferences including ASP-DAC and ICCAD. He is a member of ACM, IEEE, and IPSJ.

Hiroaki Takada is a Professor at the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. He received his Ph.D. degree in Information Science from University of Tokyo in 1996. He was a Research Associate at University of Tokyo from 1989 to 1997, and was an Assistant Professor and then an Associate Professor at Toyohashi University of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IEEE, IPSJ, and JSSST.