International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:4, 2007

Soft Real-Time Fuzzy Task Scheduling for Multiprocessor Systems

Mahdi Hamzeh, Sied Mehdi Fakhraie, and Caro Lucas

***Abstract*—**All practical real-time scheduling algorithms in multiprocessor systems present a trade-off between their computational complexity and performance. In real-time systems, tasks have to be performed correctly and timely. Finding minimal schedule in multiprocessor systems with real-time constraints is shown to be NP-hard. Although some optimal algorithms have been employed in uni-processor systems, they fail when they are applied in multiprocessor systems. The practical scheduling algorithms in real-time systems have not deterministic response time. Deterministic timing behavior is an important parameter for system robustness analysis. The intrinsic uncertainty in dynamic real-time systems increases the difficulties of scheduling problem. To alleviate these difficulties, we have proposed a fuzzy scheduling approach to arrange real-time periodic and non-periodic tasks in multiprocessor systems. Static and dynamic optimal scheduling algorithms fail with non-critical overload. In contrast, our approach balances task loads of the processors successfully while consider starvation prevention and fairness which cause higher priority tasks have higher running probability. A simulation is conducted to evaluate the performance of the proposed approach. Experimental results have shown that the proposed fuzzy scheduler creates feasible schedules for homogeneous and heterogeneous tasks. It also and considers tasks priorities which cause higher system utilization and lowers deadline miss time. According to the results, it performs very close to optimal schedule of uni-processor systems.

***Keywords*—**Computational complexity, Deadline, Feasible scheduling, Fuzzy scheduling, Priority, Real-time multiprocessor systems, Robustness, System utilization.

I. Introduction

MANY applications namely avionics, traffic control, automated factory, and military systems require real-time communication and computation. In real-time systems, all tasks have specific parameters such as deadline, priority, etc. Modern embedded computing systems are becoming increasingly complex [1]. Meanwhile, the traditional notions of best-effort and real-time processing have fractured into a spectrum of processing classes with different timeliness requirements including desktop multimedia, soft real-time, firm real-time, adaptive soft real-time, rate-based, and traditional hard real-time [2-5]. Many real-time systems are hard and missing deadline is catastrophic [5-8], whereas in soft real-time system occasional violation of deadline constraints may not result in a useless execution of the application or calamitous consequences, but decreases utilization[9]. A schedule which is executing all real-time tasks within their deadlines and all the other constraints are met, is called a feasible schedule [10]. Real-time scheduling can be classified in two categories, static [7] and dynamic [11] scheduling. A static real-time scheduling algorithm such as Rate Monotonic schedules all real-time tasks off-line using static parameters and requires complete knowledge about tasks and system parameters [12], while dynamic task scheduler calculates the feasible schedule on-line and allows tasks to be invoked dynamically. These algorithms use dynamic parameters such as deadline and laxity[2, 3, 10, 11, 13-16]. Scheduling in real-time system involves allocation of CPU and other resources to run corresponding tasks to meet certain timing constraints [13]. Nonetheless, scheduling is more significant in real-time systems than non-real-time systems[1, 9, 13, 15-20]. In real-time systems, tasks have to be performed correctly and in a timely fashion as well [21]. Tasks are classified as periodic and non-periodic [22, 23]. The execution requests of a periodic task repeatedly occur at regular intervals. On the contrary, execution requests of a non-periodic task are unpredictable.

Nowadays, using of real-time multiprocessor systems is dramatically increasing. Unfortunately, less is known about how to schedule multiprocessor-based real-time systems than that for uni-processors [14]. Optimal scheduling of real-time tasks on multiprocessor systems is known to be computationally intractable for large task sets [15]. Any practical scheduling algorithm in multiprocessor systems presents a trade-off between performance and computational complexity. Having more computational complexity in practical algorithm cause wide range of algorithm's response time hence, deterministic timing behavior is the most important parameter for system's robustness especially in hard real-time system[2, 3, 24-26]. This behavior cause decrease in utilization of the system when unpredictable conditions happened. In heterogeneous systems which tasks have different time constraints algorithm have to avoid starvation[26].

The performance of a scheduling algorithm is measured in terms of additional processor required to be added at a schedule without deadline violations as compared to optimal algorithm [15]. In [18] it has been proved that finding a minimal schedule for a set of real-time tasks in multiprocessor system is NP-hard.

In this paper, we focus on a real-time multiprocessor system with heterogeneous periodic and non-periodic tasks and compare performance and complexity of our proposed

Mahdi Hamzeh is with the Silicon Intelligence and VLSI Signal Processing Lab, School of Electrical and computer engineering, University of Tehran, Tehran, Iran (phone: +98-21-8800-6064; e-mail: mhamzeh@ece.ut.ac.ir).
Prof. Sied Mehdi Fakhraie is with the School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran (e-mail: fakhraie@ut.ac.ir).
Prof. Caro Lucas is with the Center of Excellence for Control and Intelligent Processing, University of Tehran, and School of Cognitive Science, IPM, Iran (e-mail: lucas@ipm.ir).

fuzzy scheduler with other algorithms using computer simulation.

The rest of this paper is organized as follows. Section II describes scheduling algorithms and task model. Section III describes fuzzy inference engine. Section IV introduces the proposed fuzzy real-time scheduler. Experimental results are presented in section V.

II. Scheduling Algorithms and Task Model

A. Task Model

A task is a complete sequence of instructions. Task execution starts when a task is selected by task dispatcher and one of the system's processors starts to run task's instructions. Tasks are classified according to their deadline, priority, arrival characteristic, and computation cycles requests.

B. Scheduling Algorithms

First-Come-First-Served (FCFS) algorithm [20] selects the task with the earliest arrival time. If system contains periodic tasks, their release time will be considered. This algorithm makes no effort to consider a task's deadline.

Earliest Deadline First (EDF) algorithm [15, 20] always chooses the task with the earliest deadline. It has been proved that this algorithm is optimal in a uni-processor system. Since it cannot consider priority and therefore cannot analyze it, this algorithm fails under overloading conditions.

Least Laxity First (LLF) algorithm [13] selects the task that has the lowest laxity among all the ready ones whenever a processor becomes idle, and executes it to completion. This algorithm is non-preemptive and avoids the well-known problem of its preemptive counterpart that sometimes degenerates to a processor-sharing policy.

Robust Earliest Deadline (RED) algorithm proposed in [3, 14, 15] calculates residual time and workload of tasks as their schedulability. It has some task rejection mechanism to handle system load when there is no feasible schedule [19].

Lee et al. [21] present a fuzzy scheduling algorithm. Their proposed algorithm uses task laxity and task criticality as system parameters. Their simulation model contains small number of tasks on a uni-processor system and they did not consider system overloads. All the tasks in a system are seen as real-time and fairness is not considered.

Thai [15] proposed a real-time scheduling algorithm for multi-processor distributed systems. In their approach, the task with higher computation time is assigned to bottleneck processor and system's worst case processing time is computed. However it is not clear how this task is detected. Their algorithm needs communication time between processors and assume tasks processing times are different but real-time. They do not consider heterogeneous tasks and fairness. The proposed algorithm has acceptable resistance to system overload especially when number of processors is increased.

The model described in [27] uses fuzzy inference for scheduling non-preemptive periodic tasks in soft real-time multiprocessing systems. They use priority and deadline as tasks' parameters and use a fuzzy inference engine to compute each task's priority and select the task with maximum priority to process. Although they wish to use TSK inference engine in their model, their rules are Mamdani. They assume all task are periodic and it is not clear that their processor on system is homogeneous or heterogeneous. The proposed model does not consider task's processing time. Therefore results are more similar to EDF and not suitable for multiprocessing systems.

Chen et al. [28] proposed a scheduling model and a related algorithm that is suitable for both uni-processor and multiprocessor systems. They provide a method to detect work overloading and try to balance load with task dispatching.

Dynamic integrated scheduling of hard real-time, soft real-time, and none real-time tasks are discussed in [29]. They can generate feasible schedules but their model is restricted to periodic tasks and change the tasks' periods dynamically when overloading occurs.

III. Fuzzy Inference Engine

Fuzzy logic [30, 31] is a superset of conventional Boolean logic and extends it to deal with new aspects such as partial truth and uncertainty.

Fuzzy inference is the process of formulating the mapping from a given input set to an output using fuzzy logic. The basic elements of fuzzy logic are linguistic variables, fuzzy sets, and fuzzy rules [32]. The linguistic variables' values are words, specifically adjectives like "small," "little," "medium," "high," and so on. A fuzzy set is a collection of couples of elements. It generalizes the concept of a classical set, allowing its elements to have a partial membership. The degree to which the generic element "x" belongs to the fuzzy set A (expressed by the linguistic statement x is A) is characterized by a membership function (MF), $f_A(x)$. The membership function of a fuzzy set corresponds to the indicator function of the classical sets. It can be expressed in the form of a curve that defines how each point in the input space is mapped to a membership value or a degree of truth between 0 and 1. The most common shape of a membership function is triangular, although trapezoidal and bell curves are also used. This operation normalizes all inputs to the same range and has a direct effect on system performance and accuracy.

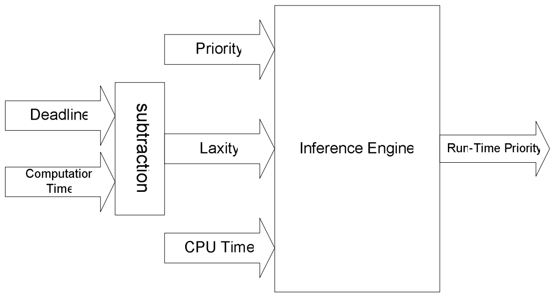

Fig. 1 Proposed inference model

A fuzzy set A is defined within a finite interval called universe of discourse U as follows:

$$A = \{(x, f_A(x)), f_A(x) : U \to [0,1]\}$$

U is the whole input range allowed for a given fuzzy linguistic variable. All fuzzy sets related to a given variable make up the term set, the set of labels within the linguistic variable described or, more properly, granulated. Fuzzy rules form the basis of fuzzy reasoning. They describe relationships among imprecise, qualitative, linguistic expressions of the system's input and output. Generally, these rules are natural language representations of human or expert knowledge and provide an easily understood knowledge representation scheme. A typical conditional fuzzy rule assumes a form such as

IF Speed is "Low" AND Race is "Dry" THEN Braking is "Soft".

Speed is Low AND Race is Dry is the rule's premise; while Braking is Soft is the consequent. The premise predicate might not be completely true or false, and its degree of truth ranges from 0 to 1. We compute this value by applying the membership functions of the fuzzy sets labeled "Low" and "Dry" to the actual value of the input variables Speed and Race. After that, fuzzification is applied to the conclusion; the way in which this happens depends on the inference model.

There are two types of fuzzy inference models:

1. Mamdani [33],
2. TSK or Sugeno [34].

Interpreting an if-then rule involves two distinct parts: first evaluating the antecedent and then applying results to the consequent (known as implication) [35, 36]. In the case of two-valued or binary logic, if-then rules do not present much difficulty. If the premise is true, then the conclusion is true, whereas with fuzzy approach, if the antecedent is true to some degree of membership, then the consequent is also true to that same degree.

Mamdani-type [33] inference expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It is possible, and in many cases much more efficient, to use a single spike as the output's membership function rather than a distributed fuzzy set. This is sometimes known as a singleton output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, Sugeno-type systems use weighted sum of a few data points. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.

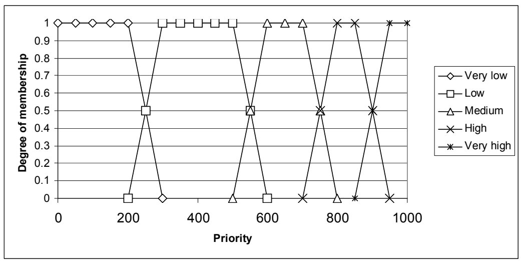

Fig. 2 Fuzzy sets corresponding to priority

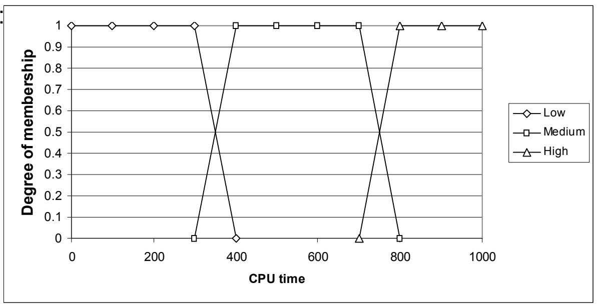

Fig. 3 Fuzzy sets corresponding to CPU time

IV. Proposed Model

As shown in Fig. 1, the major factors considered in our approach to determine the scheduling are task priority, deadline, required computation time, and used CPU time. The notion of laxity is used in the proposed approach to facilitate the computation. Laxity is the maximum time that a task can wait before being executed (i.e., laxity = deadline - computation time).

A task's priority shows the importance of the task. The inputs of these parameters are justified and represented as linguistic variables and fuzzy rules are then applied to those linguistic variables to compute the level value for deciding which task to select to schedule next.

CPU time is another parameter which could guarantee scheduling fairness. We considered 5 trapezoid membership functions for task's priority. "Very high", "High", "Medium", "Low" and "Very low" are these membership functions. This number and naming of membership is same for task's laxity however CPU time membership function considered 3 and also trapezoid. "High", "Medium" and low are the name of these functions. For the $f_a(x)$ as the membership function, a large class of functions can be taken such as triangular, trapezoidal, Gaussian and bell function however we selected trapezoidal for its usability in fuzzy dedicated hardware [35-37].The used membership functions for this model illustrated in Fig. 2 and 3.

In our proposed algorithm as shown in Fig. 4, a newly arrived task will be added to the input queue. This queue consists of the remaining tasks from last cycle that has not yet been assigned.

```
1. For each task of input queue
     a. Feeds task's run-time priority using fuzzy
        inference engine
2. While system has a free processor
     a. assign the task with highest run-time
        priority to the processor
3. Loop forever
     a. If processor event occurs
          i. Go to 2.
     b. If scheduling event occurs
          i. Update tasks parameters.
          ii. Go to 1.
```

Fig. 4 Proposed algorithm

Fuzzy scheduler processes each task separately and computes its run-time priority and sends it to task dispatcher's priority queue. In a multiprocessor system, this queue offers tasks to dispatcher by their run-time priority order. Dispatcher offers a new task whenever one of the processors of the system finishes its task.

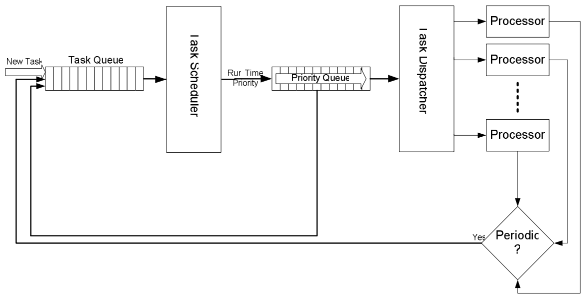

Fig. 5 System view of soft real-time fuzzy scheduler

Periodic tasks which their execution requests occur repeatedly will remain in the system queue while non-periodic tasks will be finished and their next request starts with task initialization. In this model all kinds of tasks are considered (Fig. 5).

In firm real-time systems with more real-time constraints, our model can be adapted with multiple input queues with multiple schedulers where task's priorities differ. With this technique, when the number of system tasks is very high, scheduler can select most important tasks and send them to dispatcher queue while with last model processing all tasks parameters could be time consuming and waste system time.

Scheduler and dispatcher are independent components and they are connected with a queue; consequently our proposed scheduler is extendable.

Due to the model extendibility and adaptability, this model can be used in a variety of systems with multi-criteria constraints.

Satisfactory performance is achieved by using 39 Sugeno rules only. This number is obtained by simplifying 169 rules in different examples. Some of them are mentioned below:

- If (**Laxity is "Very low"**) and (**Priority is "Very high"**) then $R\Pr iority = 100 \times priority - 10 \times laxity$.
- If (**Laxity is "Low"**) and (**Priority is "Very high"**) then $R\Pr iority = 50 \times priority - 20 \times laxity$.
- If (**Laxity is "Medium"**) and (**Priority is "Normal"**) and (**CPU time is "High"**) then $R\Pr iority = 25 \times priority - 40 \times laxity$ $-50 \times CPUtime$

Choosing number of rules and membership functions directly affects system accuracy while performance of the system increases with rule size decrease. There are some techniques for adjusting membership functions however; in this paper we did not consider these approaches.

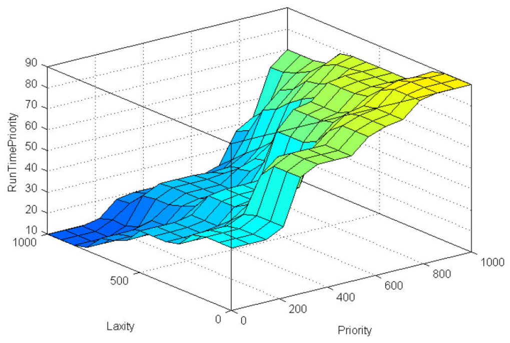

Fig. 6 The decision surface corresponding to inference rules

The corresponding decision surface to these rules and membership functions is illustrates in Fig. 6.

V. Experimental Results

We are simulated our algorithm using our custom-designed simulator implemented using Java. In our simulator we have 100 tasks, among which 10 has very high priority, 30 has high prioritym 20 has medium and 20 has very low priority. We considered priority in 0-1000.

Each task's deadline and required computation cycles considered in 0-1000 which means maximum allowed laxity is 1000. These parameters are generated or updated randomly when a new arrived task generated of its computation finished.

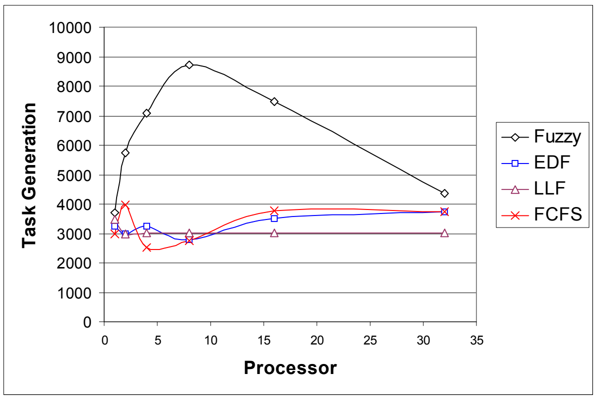

Fig. 7 High priority task generation

Simulation results show that by increasing the number of system's processors, generation of high priority tasks increases until high priority task's waiting times is reduced to an acceptable range (Fig. 7). By increasing processor, high priority tasks have higher probability of execution while their laxity would not decrease to critical region. This behavior results in more execution for low priority tasks in medium load cycles. Next, low priority tasks generation increases to handle low priority task's waiting time.

Simulation results show that the model can feasibly schedule tasks when system load increases and keep system processors loads close to one even at crowded times. However, other algorithms like LLF and EDF break down when the system is overloaded. In this model, we did not consider scheduler processing time and this process is independent of the number of system's processors. We note that processor's load remain always below one because of dispatcher's processing time. By analysis of task scheduler, periodic task's period increases automatically by scheduler with consideration of their priority and CPU time. This behavior of the system is similar elastic scheduling proposed by [19].

While number of the system's processors increases, our model balances the load between processors. This well balancing will causes efficient processing time in symmetric systems. The proposed scheduler's average waiting time is close to LLF and EDF algorithms. Simulations demonstrate the algorithm is capable of task balancing when the number of processors increases. However in comparison to other algorithms high priority tasks have smaller waiting times. This implies a better response time for the system and it selects high priority tasks with higher probability.

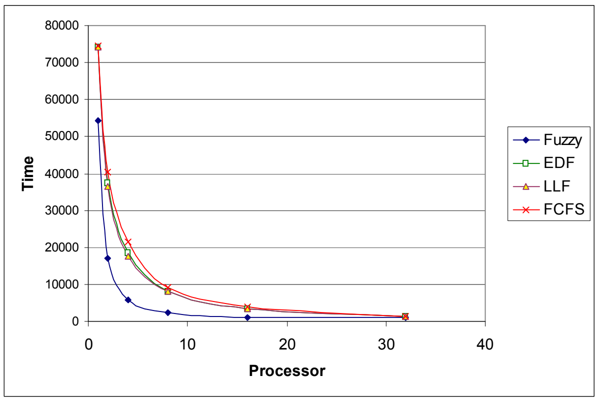

Fig. 8 High priority tasks miss time

Since the total system computation power is constant, so when some high priority tasks get a higher portion of system's computation power, the other ones will receive lower attention. This causes a reduction in selection of low priority tasks (Fig. 8).

TABLE I
REQUIRED PROCESSOR FOR FEASIBLE SCHEDULING IN DIFFERENT ALGORITHM WITH DIFFERENT HIGH PERFORMANCE TASKS RATIO

HP to all ratio	Proposed	Priority based	EDF	LLF	FCFS
20%	1	1	2	2	4
40%	2	2	4	4	5
80%	4	4	7	7	8

TABLE II
REQUIRED PROCESSOR FOR FEASIBLE SCHEDULING IN DIFFERENT ALGORITHM WITH DIFFERENT LOW PERFORMANCE TASKS RATIO

LP to all ratio	Proposed	Priority based	EDF	LLF	FCFS
20%	6	12	4	4	4
40%	4	8	2	2	2
80%	1	2	1	1	1

As shown in Table I, by considering ratio of high priority task to all, our model select high priority tasks with higher probability which makes acceptable system's waiting time and miss ratio for high priority tasks by use the lower number of processors. For applications which average waiting time for all tasks is an important parameter and designer have to care about low priority task to restrict average waiting time. Number of processor required for acceptable miss ratio is listed in Table II which its first column is the ratio of low priority tasks to all tasks. Our algorithm provides an average utilization similar to other algorithm. However, Fig. 8 demonstrate, our algorithm significantly performs better for high priority tasks in a real-time environment.

VI. CONCLUSION AND FUTURE WORK

The proposed scheduler which proposed in this paper has low complexity due to the simplicity of fuzzy inference engine. As a consequence, its computation complexity and response time is constant and by increasing the number of processors will not increase. This model is efficient when system has heterogeneous tasks with different constraints.

Our future work is to map this algorithm on our real-time fuzzy processor.

REFERENCES

[1] F. Gruian, "Energy-centric scheduling for real-time systems," in Department of Computer Science. Ph.D dissertation: Lund University, 2002, p. 164.
[2] Z. Deng, J. W. Liu, and S. Sun, "Dynamic scheduling of hard real-time applications in open system environment," Tech. Rep., University of Illinois at Urbana-Champaign 1996.
[3] G. Buttazzo and J. A. Stankovic, "RED: robust earliest deadline scheduling," in Proc. 3rd Intl. Workshop Responsive Computing Systems, Lincoln, NH, 1993, pp. 100-111.
[4] S. M. Petters, "Bounding the execution time of real-time tasks on modern processors," in Proc. 7th Intl. Conf. Real-Time Computing Systems and Applications, Cheju Island, 2000, pp. 498-502.
[5] J. Zhu, T. G. Lewis, W. Jackson, and R. L. Wilson, "Scheduling in hard real-time applications," IEEE Softw., vol. 12, pp. 54-63, 1995.
[6] K. Taewoong, S. Heonshik, and C. Naehyuck, "Scheduling algorithm for hard real-time communication in demand priority network," in Proc. 10th Euromicro Workshop Real-Time Systems, Berlin, Germany, 1998, pp. 45-52.
[7] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," J. ACM, vol. 20 pp. 46-61, 1973.
[8] D. Babbar and P. Krueger, "On-line hard real-time scheduling of parallel tasks on partitionable multiprocessors," in Proc. Intl. Conf. Parallel Processing, 1994, pp. 29-38.
[9] W. Lifeng and Y. Haibin, "Research on a soft real-time scheduling algorithm based on hybrid adaptive control architecture," in Proc. American Control Conf, Lisbon, Portugal, 2003, pp. 4022-4027 vol.5.
[10] T. F. Abdelzaher and K. G. Shin, "Comment on a pre-run-time scheduling algorithm for hard real-time systems," IEEE Trans Software Engineering, vol. 23, pp. 599-600, Sep 1997.
[11] K. Ramamritham and J. A. Stankovic, "Dynamic task scheduling in hard real-time distributed systems," IEEE Softw., vol. 1, pp. 65-75, July 1984.
[12] P. A. Laplante, "The certainty of uncertainty in real-time systems," IEEE Instrum. Meas. Mag., vol. 7, pp. 44-50, Dec 2004.
[13] K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems," Proc. IEEE, vol. 82, pp. 55-67, Jan 1994.
[14] J. Kreuzinger, A. Schulz, M. Pfeffer, T. Ungerer, U. Brinkschulte, and C. Krakowski, "Real-time scheduling on multithreaded processors," in Proc. 7th Intl. Conf. Real-Time Computing Systems and Applications, Cheju Island, South Korea, 2000, pp. 155-159.

[15] N. D. Thai, "Real-time scheduling in distributed systems," in Proc. Intl. Conf. Parallel Computing in Electrical Engineering, Warsaw, Poland, 2002, pp. 165- 170.
[16] C. Lin and S. A. Brandt, "Efficient soft real-time processing in an integrated system," in Proc. 25th IEEE Real-Time Systems Symp., 2004.
[17] I. E. W. Giering and T. P. Baker, "A tool for the deterministic scheduling of real-time programs implemented as periodic Ada tasks," Ada Lett., vol. XIV, pp. 54-73, 1994.
[18] L. Hluchý, M. Dobrucký, and J. Astalos, "Hybrid approach to task allocation in distributed systems," in Proc. 4th Intl. Conf. Parallel Computing Technologies, Yaroslavl, Russia, 1997 pp. 210-215.
[19] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic scheduling for flexible workload management," IEEE Trans. Comput., vol. 51, pp. 289-302, Mar 2002.
[20] A. S. Tanenbaum, Distributed operating systems: Prentice Hall, 1994.
[21] J. Lee, A. Tiao, and J. Yen, "A fuzzy rule-based approach to real-time scheduling," in Proc. 3rd IEEE Conf. Fuzzy Systems, IEEE World Congress Computational Intelligence, FL, 1994, pp. 1394-1399 vol.2.
[22] M. Silly-Chetto, "Dynamic acceptance of aperiodic tasks with periodic tasks under resource sharing constraints," IEE Proc. Software, vol. 146, pp. 120-127, Apr 1999.
[23] M. Caccamo and G. Buttazzo, "Exploiting skips in periodic tasks for enhancing aperiodic responsiveness," in Proc. 18th IEEE Real-Time Systems Symp., San Francisco, CA, 1997, p. 330.
[24] J. Mario J. Gonzalez, "Deterministic processor scheduling," ACM Comput. Surv., vol. 9, pp. 173-204, 1977.
[25] R. Jiminez-Peris, M. Patino-Martinez, and S. Arevalo, "Deterministic scheduling for transactional multithreaded replicas," in Proc. 19th IEEE Symp. Reliable Distributed Systems, Nurnberg, Germany, 2000, pp. 164-173.
[26] S. Zhiao, E. Jeannot, and J. J. Dongarra, "Robust task scheduling in non-deterministic heterogeneous computing systems," in Proc. IEEE Intl. Conf. Cluster Computing, Barcelona, Spain, 2006, pp. 1-10.
[27] M. Sabeghi, M. Naghibzadeh, and T. Taghavi, "Scheduling non-preemptive periodic tasks in soft real-time systems using fuzzy inference," in Proc. 9th IEEE Intl. Symp. Object and Component-Oriented Real-Time Distributed Computing, Gyeongju, KOREA, 2006, pp. 27-32.
[28] G. Chen, G. Chen, O. Ozturk, and M. Kandemir, "An adaptive locality-conscious process scheduler for embedded systems," in Proc. 11th IEEE Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, 2005 pp. 354-364.
[29] S. A. Brandt, S. Banachowski, L. Caixue, and T. Bisson, "Dynamic integrated scheduling of hard real-time, soft real-time, and non-real-time processes," in Proc. 24th IEEE Intl. Real-Time Systems Symposium, Cancun, Mexico, 2003, pp. 396-407.
[30] L. A. Zadeh, "Fuzzy sets versus probability," Proc. IEEE, vol. 68, pp. 421-421, March 1980.
[31] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," Commun. ACM, vol. 37, pp. 77-84, March 1994.
[32] W. Pedrycz and F. Gomide, An introduction to fuzzy sets: analysis and design: The MIT Press, 1998.
[33] E. H. Mamdani, "Application of fuzzy algorithms for the control of a dynamic plant," Proc. IEE, vol. 121, pp. 1585-1588, Dec 1974.
[34] T. Takagi and M.Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Trans. Syst., Man, Cybern., vol. 15, pp. 116-132, 1985.
[35] H. Surmann and A. P. Ungering, "Fuzzy rule-based systems on general-purpose processors," IEEE Micro, vol. 15, pp. 40-48, Aug 1995.
[36] G. Ascia and V. Catania, "A general purpose processor oriented to fuzzy reasoning," in Proc. 10th IEEE International Conf. Fuzzy Systems, Melbourne, Australia, 2001, pp. 352-355.
[37] K. Youngdal and L.-K. Hyung, "An architecture of fuzzy logic controller with parallel defuzzification," in Proc. Biennial Conf. of the North American Fuzzy Information Processing Society, Berkeley, CA, 1996, pp. 497-501.

Mahdi Hamzeh was born in Isfahan, Iran, in 1982. He received the B.S. degree in computer engineering from Azad University of Qazvin in 2005. He is currently working toward the M.S. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran.
His research interests include hardware implementation of intelligent systems, High performance intelligent computing, Reconfigurable computing, Multiprocessor SoC and real-time systems and applications. .Since 2005, he is member of Silicon Intelligence and VLSI Signal Processing Laboratory (SILab).

Sied Mehdi Fakhraie was born in Dezfoul, Iran, in 1960. He received his M.Sc. degree in electronics from the University of Tehran, Tehran, Iran, in 1989 an the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada in 1995. Since 1995, he has been with the School of Electrical and Computer Engineering, University of Tehran, where he is now an Associate Professor. He has been the founder of the VLSI Circuits and Systems Laboratory and is now Director of Silicon Intelligence and VLSI Signal Processing Laboratory. From September 2000 to April 2003, he was with Valence Semiconductor Inc. and has worked in Dubai, UAE, and Markham, Canada offices of Valence as Director of ASIC/SoC Design and also technical lead of Integrated Broadband Gateway and Family Radio System baseband processors.
During the summers of 1998, 1999, and 2000, he was a visiting professor at the University of Toronto, where he continued his work on efficient implementation of artificial neural networks. He is coauthor of the book VLSI-Compatible Implementation of Artificial Neural Networks (Boston, MA: Kluwer, 1997). He has also published more than 70 reviewed conference and journal papers. He has worked on many industrial IC design projects including design of network processors and home gateway access devices, DSL modems, pagers, and one- and two-way wireless messaging systems, and digital signal processors for personal and mobile communication devices. His research interests include system design and ASIC implementation of integrated systems, novel techniques for high-speed digital circuit design, and system-integration and efficient VLSI implementation of intelligent systems.

Caro Lucas received the Ms. degree from the University of Tehran, Iran, in 1973, and the Ph.D degree from the university of California, Berkeley, in 1976. He is a Professor of Center of Excellence for Control and Intelligent Processing at the School of Electrical and Computer Engineering, University of Tehran, Iran, as well as a researcher at the School of Intelligent Systems (SIS), Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran. He has served as the Director of SIS (1993-1997), Chairman of the ECE Department at the University of Tehran (1986-1988), Managing Editor of the Memories of the Engineering Faculty, University of Tehran (1979-1991), Reviewer of Mathematical Reviewers (since 1987), Associate Editor of Journal of Intelligent and Fuzzy systems (1992-1999) , and Chairman of the IEEE, Iran section (1990-1992). He was also a Visiting Associate Professor at the University of Toronto (summer, 1989-1990), University of California, Berkeley (1988-1989), an Assistant Professor at Garyounis University (1984-1985), University of California at Los Angeles (1975-1976), a Senior Researcher at the International Center for Theoretical Physics and the International Center for Genetic Engineering and Biotechnology, both in Trieste, Italy, the Institute of Applied Mathematics Chinese Academy of Sciences, Harbin Institute of Electrical Technology, a Research Associate at the Manufacturing Research Corporation of Ontario, and a Research Assistant at the Electronic Research Laboratory, University of California, Berkeley. He is the holder of Patent on “Speaker Independent Farsi Isolated Word Neurorecognizer”. His research interests include biological computing, computational intelligence, uncertain systems, intelligent control, neural networks, multi-agent systems, data mining, business intelligence, financial modelling and knowledge management. Professor Lucas has served as the chairman of several International Conferences. He was the founder of the SIS and has assisted in founding several new research organizations and engineering disciplines in Iran. He is the recipient of several research grants at the University of Tehran and SIS.