

Failure Rate Minimization with Multiple Function Unit Scheduling for Heterogeneous WSNs

Meikang Qiu[†] Jing Deng[‡] Edwin H.-M. Sha^{*}

[†] Dept. of Electrical Engineering, University of New Orleans, New Orleans, LA 70148, USA

[‡] Dept. of Computer Science, University of North Carolina at Greensboro, Greensboro, NC 27402, USA

^{*} Dept. of Computer Science, University of Texas at Dallas, TX 75083, USA

Abstract—Failure-Rate Minimization is becoming one of the major design issues in *wireless sensor network* (WSN) architecture due to multiple available *Functional-units* (FUs). There is a tradeoff between reliability and performance, such as timing constraint. This paper studies how to minimize the total failure rate while satisfying performance requirement for WSN applications. Two novel algorithms are proposed to solve the FRMFS (*Failure Rate Minimization with FU Scheduling*) problem. We use these FU scheduling algorithms to minimize system failure rate without sacrificing performance. Our results show that the average improvement on failure-rate reduction is significant with the use of our algorithms.

Index Terms—Heterogeneous, scheduling, failure rate, multi-FU, minimization

I. INTRODUCTION

Wireless sensor networks (WSNs) have a wide variety of applications, for example: microclimate monitoring [1], [2], plant physiology [3], animal behavior [4]–[6], structural monitoring [2], [7], [8] and condition-based maintenance. Reliability of distributed micro sensor nodes is an important issue in the design of sensor networks.

Heterogeneity is the trend of today’s WSNs. With more and more sensor types or *Functional-units* (FUs) available, it is challenging to take full advantage of these FUs while satisfying the timing constraints in WSN design. The same function can be executed by different sensor type or FU types. Therefore, the same type of operations can be processed by heterogeneous sensors or FUs with different costs or failure rates. This paper focuses on how to reduce the total system failure rate of a WSN composed of heterogeneous sensors (FUs).

According to the *failure rate* model proposed by Srinivasan and Jha [9], the *failure rate of a system* is defined as the rate of failure of a system or component during during the time of executing a DFG (Data Flow Graph). A DFG is a node-weighted and edge-weighted directed graph, where each node represents a task, and each edge defines the precedence relations among nodes. In order to minimize the system failure rate, we need to minimize the sum of failure rates of all the nodes in the system. In other words, we need

to find a schedule such that the timing constraint is satisfied and the sum of failure rates of all nodes is minimized.

Combining the consideration of failure rate [?], [9], [10] and performance, we design in this paper two algorithms to solve the FRMFS (*Failure Rate Minimization with FU Scheduling*) problem, i.e., how to minimize total failure rate while satisfying timing constraint for WSN applications. Our experimental results show that these algorithms achieve significant reductions on an average in total failure rate compared with the previous works. For example, with 3 FUs, compared with the TBFT algorithm in [9], our algorithm *FU_Sch2* shows an average 21.7% reduction in total failure rate.

The paper is organized as follows: we describe necessary background in Section II. Our proposed algorithms are presented in Section III. In Section IV, experimental results are shown and discussed. Concluding remarks are provided in Section V.

II. BASIC CONCEPTS AND MODELS

In this section, we introduce some basic concepts that will be used throughout this paper. First we present the system failure rate model. Then, we describe the FUs scheduling problem with the help of an example.

A. The Failure Rate Model

We use the *failure rate* model proposed by Srinivasan and Jha [9]. Consider a heterogeneous system with M FUs available, $\{F_1, F_2, \dots, F_M\}$, and a DFG containing N nodes, $\{u_1, u_2, \dots, u_N\}$. Let $t_j(i)$ be the execution time of node u_i for FU F_j . Let f_j be the *failure rate* of FU F_j . Then the *failure rate* of node u_i for FU F_j is defined as $t_j(i) \cdot f_j$. Let x_{ij} be a binary number that denotes whether FU F_j is scheduled to node u_i or not (it equals 1 if FU F_j is assigned to u_i ; otherwise it equals 0).

Define the *un-failure rate of a system* as the probability that the system will not fail during the time of executing a DFG. The probability of a system not failing during the time of processing a DFG is:

$$\text{Pr} = \prod_{1 \leq j \leq M, 1 \leq i \leq N} (1 - f_j)^{x_{ij} t_j(i)}.$$

This work is partially supported by NSF CCR-0309461, NSF IIS-0513669, HK CERG B-Q60B, NSFC 60728206, Louisiana EPSCoR Pfund LBOR0049PR00C and BoR RCS grant LEQSF (2005-08)-RD-A-43.

When $f_j \ll 1$, we can approximate

$$(1 - f_j)^{x_{ij}t_j(i)} \approx 1 - f_j x_{ij}t_j(i) \approx e^{-f_j x_{ij}t_j(i)}.$$

Therefore, we have

$$\Pr \approx \prod (e^{-f_j x_{ij}t_j(i)}),$$

when $f_j \ll 1$ [11].

Thus, in order to maximize \Pr , we need to minimize $\sum (f_j x_{ij}t_j(i))$, which is called as *failure rate of a system*. In other words, we need to find a schedule such that the timing constraint is satisfied and the sum of failure rates of all nodes is minimized, which is needed to avoid system failure as much as possible.

B. Heterogeneous FU schedule problem

We define the heterogeneous FU scheduling problem as follows: given a heterogeneous system with M FUs, $F = F_1, \dots, F_i, \dots, F_M$, a DAG $G = \langle V, E_d \rangle$ where $V = \langle u_1, \dots, u_i, \dots, u_N \rangle$ is a set of nodes, with each node representing a task, $E_d \subseteq V \cdot V$ is a set of edges representing dependencies relations among nodes in V . $T(u_k) = t_1(k), \dots, t_i(k), \dots, t_M(k)$, where $t_i(k)$ denoted the computation time of u_k on F_i , and a time constraint L , find a task schedule for G such that the failure rate is minimized within L .

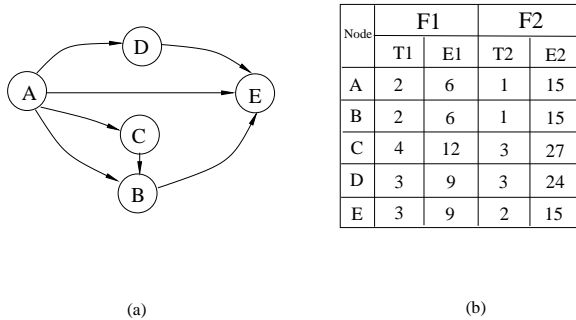


Fig. 1. (a) A DAG. (b) Execution times and failure rate of FUs.

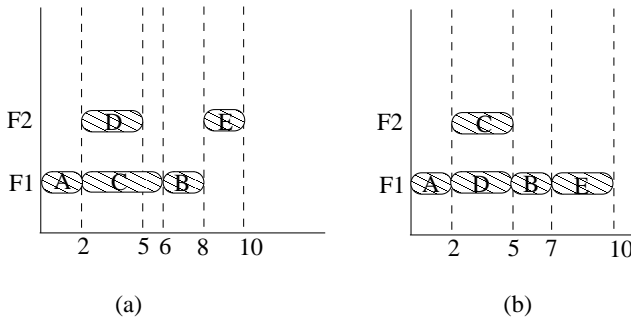


Fig. 2. (a) Schedule 1. (b) Schedule 2.

An example is shown in Fig. 1 and Fig. 2. Assume there is a heterogeneous system that consists of two heterogeneous

FUs, F1 and F2. An exemplary DAG is shown in Fig. 1(a). The given time constraint for the DAG to be executed is 10 time units. The execution time and failure rate of each node for different FUs are shown in Fig. 1(b). For the convenience of computation and presentation, we use an integer (enlarged 1000 times) to represent the failure rate of each node. In Fig. 1(b), T_i denotes the execution time and E_i denotes the first part of the failure rate E_{ij} , which is the failure rate for u_i to be scheduled on F_j . Two schedules for the DAG are shown in Fig. 2. In Schedule 1, the schedule length is 10 time units and its system failure rate is 63. In Schedule 2, the schedule length is 10 time units and its system failure rate is 57. Both schedules satisfy the time constraint while the latter has a lower failure rate. This example shows that different task schedules will produce different system failure rate.

III. THE ALGORITHMS

In this section, two algorithms, *FU_Sch1* and *FU_Sch2* are designed to solve the FRMFS (*Failure Rate Minimization with FU Scheduling*) problem, i.e., how to obtain the minimum total system failure rate without sacrificing performance, based on multi-FU scheduling.

Scheduling problems with time and resource constraints are well-known to be NP complete. We are going to solve a scheduling problem with time and resource constraints in a heterogeneous system and minimize the failure rate of the system at the same time. Therefore, our problem is also NP complete. In this section, two novel algorithms, *FU_Sch1* and *FU_Sch2*, are developed to solve this problem. *FU_Sch1* uses a bipartite matching strategy based on ALAP (As Late As Possible) scheduling and *FU_Sch2* uses a progressive relaxation strategy based on ALAP scheduling. Some symbols used in our algorithms are listed in Table I.

Symbol	Meaning
Min	Minimum failure rate for the current node
EST_i	Earliest starting time for node i
LST_i	Latest starting time for node i
FT_i	Finish time of node i
SL_j	Schedule length for FU_j
X_i	The FU that node i is scheduled on
E_{total}	Overall failure rate for the system

TABLE I
SYMBOLS IN THE FU SCHEDULING ALGORITHMS

A. FU Scheduling with Bipartite Matching

FU_Sch1 is designed to use the bipartite matching strategy to schedule tasks. The idea is: first, use the ALAP scheduling, which minimizes the schedule length, to get the latest starting time for every node. Then construct a bipartite matching graph with the nodes in the ready list in one set and all FUs in the other set. Then reschedule nodes based on the minimum failure-rate bipartite matching. This algorithm is shown in Algorithm III.1 V_1 and V_2 in the algorithm represent the two sets used in the bipartite matching.

This algorithm first schedules all nodes using the ALAP scheduling. Based on the schedule, we use the bipartite

Algorithm III.1 FU_Sch1 Algorithm**Input:** a DAG $G = \langle V, E_d \rangle$, a set of FUs, a timing constraint L **Output:** A task scheduling with failure-rate minimization

```

1: for all  $u_i \in V$ ; do
2:    $EST_i \leftarrow$  starting time of node  $i$  in ASAP;
3:    $LST_i \leftarrow$  starting time of node  $i$  in ALAP;
4:    $FT_i \leftarrow$  Finish time of node  $i$  in ALAP;
5: end for
6: for all  $F_j \in F$  do
7:    $SL_j \leftarrow 0$ ;
8: end for
9:  $E_{total} \leftarrow 0$ ;
10: while  $\exists$  nodes not marked do
11:   for all  $F_j \in F$  do
12:     if the first node in  $F_j$  has no dependency constraint then
13:       put it into  $V_1$ ;
14:     end if
15:     put  $F_j$  into  $V_2$ ;
16:   end for
17:   construct a weighted bipartite matching graph  $G_{BM} = \langle V_{BM}, E_{BM} \rangle$ ;  $V_{BM} = V_1 \cup V_2$ ;
18:   for all  $u_i \in V_1$  do
19:     for all  $F_j \in V_2$  do
20:       compute  $E_{ij}$ ;
21:       add an edge  $e_{ij}$  between  $u_i$  and  $F_j$  into  $E_{BM}$ ;
22:       if  $SL_j + t_j(i) + \text{data migration delay} \leq FT_i$  then
23:         set  $E_{ij}$  as the edge weight;
24:       else
25:         set the edge weight as infinity;
26:       end if
27:     end for
28:   end for
29:    $M \leftarrow$  minimum-cost-bipartite-matching for nodes in  $G_{BM}$ ;
30:   for all  $e_{ij}$  in the matching  $M$  do
31:     mark  $u_{ij}$  as scheduled;  $X_i \leftarrow$  the matching  $F_j$ ;
32:      $E_{total} \leftarrow E_{ij} + E_{total}$ ;
33:      $SL_{X_i} \leftarrow \max(EST_i, SL_j) + t_{X_i}(i)$ ;
34:     Add the data migration delay into  $SL_{X_i}$ ;
35:     for all  $u_k$  which is a dependent of  $u_i$  do
36:       update the dependence information for  $u_k$ ;
37:       if  $EST_k < SL_{X_i}$  then
38:          $EST_k \leftarrow SL_{X_i}$ ;
39:       end if
40:     end for
41:   end for
42: end while

```

matching to schedule nodes. For each FU, among those nodes not marked by matching, the node with the earliest start time is considered: if it has no dependency constraint at that time, it's inserted into the ready list. A bipartite matching graph is constructed as follows: all nodes from the ready list are on one side, denoted by a set V_1 , and all FUs are on the other side, denoted by a set V_2 . Each node, u_i , in V_1 has an edge connected with each FU, F_j , in V_2 . If the schedule length of the FU plus the node's computation time is less than the finish time of the node in ALAP, the edge weight is set to the failure rate E_{ij} . Otherwise, the edge weight is set to be infinity. After constructing the graph, call the minimum-cost-bipartite-matching function to get a minimum cost bipartite matching. Since the edge weight is set to the failure rate, the matching produced by the function minimizes the reliability cost in

Algorithm III.2 FU_Sch2 Algorithm**Input:** a DAG $G = \langle V, E_d \rangle$, a set of FUs, a timing constraint L **Output:** A task scheduling with failure-rate minimization

```

1: for all  $u_i \in V$ ; do
2:    $X_i \leftarrow -1$ ;
3:    $EST_i \leftarrow$  starting time of node  $i$  in ASAP;
4:    $LST_i \leftarrow$  starting time of node  $i$  in ALAP;
5:    $FT_i \leftarrow$  Finish time of node  $i$  in ALAP;
6: end for
7: for all  $F_j \in F$  do
8:    $SL_j \leftarrow 0$ ;
9: end for
10:  $E_{total} \leftarrow 0$ ;
11: while  $\exists$  nodes not marked do
12:    $Min \leftarrow \infty$ ;
13:   take the node  $u_i$ , with minimum  $LST_i$  and not marked;
14:   for all  $F_j \in F$  do
15:     compute  $E_{ij}$ ;
16:     if  $SL_j + t_j(i) + \text{data migration delay} \leq FT_i$  then
17:       if  $E_{ij} < Min$  then
18:          $Min \leftarrow E_{ij}$ ;  $X_i \leftarrow F_j$ ;
19:       end if
20:       if  $E_{ij} = Min$  then
21:          $X_i \leftarrow$  the FU with earlier finish time;
22:       end if
23:     end if
24:   end for
25:   mark  $u_{ij}$  as scheduled;
26:    $E_{total} \leftarrow Min + E_{total}$ ;
27:    $SL_{X_i} \leftarrow \max(EST_i, SL_j) + t_{X_i}(i)$ ;
28:   Add the data migration delay into  $SL_{X_i}$ ;
29:   for all  $u_k$  which is a dependent of  $u_i$  do
30:     update the dependence information for  $u_k$ ;
31:     if  $EST_k < SL_{X_i}$  then
32:        $EST_k \leftarrow SL_{X_i}$ ;
33:     end if
34:   end for
35: end while

```

each scheduling step. After a match is found, schedule the tasks on the corresponding FUs, mark the node and update their descendants' information, i.e. dependence constraints and earliest starting time, recursively. Repeat this process until there is no more node to be rescheduled.

Because the FU selection for a node is limited by the finish time in the ALAP, the node can at least be scheduled on the same FU as ALAP. Thus, as long as there is an ALAP schedule for the graph, this algorithm won't fail to produce a schedule for all the tasks. As the failure rate is the matching factor in the bipartite matching, the failure-rate minimization is improved over the list scheduling algorithm.

B. FU Scheduling with Progressive Relaxation

FU_Sch2 progressively improves the reliability based on the schedule obtained by the ALAP scheduling. The idea is to reschedule each node to reduce the system failure rate as much as possible. It is shown in Algorithm III.2. After the initialization, this algorithm first obtains an ALAP schedule for all the tasks in the graph. Then repeat the following steps until all nodes are marked: among all nodes that are not marked, take the node with earliest starting time and reschedule it to a FU such that the system failure

Four Methods Comparison with 3 FUs									
Bench.	N.	M1	M2	M3			M4		
		E (μ J)	E (μ J)	E (μ J)	% M1 (%)	% M2 (%)	E (μ J)	% M1 (%)	% M2 (%)
WDF(1)	4	625	550	446	28.6	18.9	431	31.0	21.6
WDF(2)	12	1862	1649	1335	28.3	19.0	1284	31.0	22.1
IIR	16	2456	2135	1751	28.7	18.0	1678	31.7	21.4
DPCM	16	2548	2267	1837	27.9	19.0	1762	30.8	22.3
2D(1)	34	5402	4732	3825	28.6	18.5	3725	31.0	21.3
2D(2)	4	665	591	478	28.1	19.1	461	30.7	22.0
MDFG1	8	1321	1153	942	28.7	18.3	902	31.7	21.8
MDFG2	8	1487	1325	1075	27.7	18.9	1034	30.5	22.0
Floyd	16	2624	2259	1872	28.7	17.1	1801	31.4	20.3
All-pole	29	4475	3946	3211	28.2	18.6	3082	31.1	21.9
Average Reduction (%)					28.4	18.5		31.1	21.7

TABLE II

THE COMPARISON OF TOTAL FAILURE RATE WITH FOUR METHODS ON VARIOUS BENCHMARKS WHEN THE TIME CONSTRAINT IS 1000.

Four Methods Comparison with 4 FUs									
Bench.	N.	M1	M2	M3			M4		
		E (μ J)	E (μ J)	E (μ J)	% M1 (%)	% M2 (%)	E (μ J)	% M1 (%)	% M2 (%)
WDF(1)	4	768	665	534	30.5	19.7	505	34.2	24.1
WDF(2)	12	2298	2006	1603	30.2	20.1	1521	33.8	24.2
IIR	16	3041	2601	2104	30.8	19.1	1979	34.9	23.9
DPCM	16	3102	2669	2168	30.1	18.8	2021	34.8	24.3
2D(1)	34	6523	5645	4521	30.7	19.9	4294	34.2	23.9
2D(2)	4	812	705	569	29.9	19.3	531	34.6	24.7
MDFG1	8	1687	1483	1185	29.8	20.1	1121	33.6	24.4
MDFG2	8	1811	1568	1273	29.7	18.8	1187	34.5	24.3
Floyd	16	3257	2822	2265	30.5	19.7	2145	34.1	24.0
All-pole	29	5503	4790	3831	30.4	20.0	3623	34.2	24.4
Average Reduction (%)					30.3	19.6		34.3	24.2

TABLE III

THE COMPARISON OF TOTAL FAILURE RATE WITH FOUR METHODS ON VARIOUS BENCHMARKS WHEN THE TIME CONSTRAINT IS 1000.

rate is minimized. A node can only be rescheduled to a FU if its finish time is earlier than that in ALAP and the task does not overlap with other tasks remaining in the ALAP schedule. The choice of the node is made this way because the remaining nodes can always be scheduled within the time constraint as long as the ALAP schedule exists for this task graph. After this task is scheduled, mark the node, update the dependence constraint information and the earliest start time for all its descendants recursively, update the system failure rate and the schedule length, then continue this process until all nodes are marked.

IV. EXPERIMENTS

In this section, we present our experiments with the our algorithms on a set of well-known DSP benchmarks including Wave Digital filter (WDF), Infinite Impulse filter (IIR), Differential Pulse-Code Modulation device (DPCM), Two dimensional filter (2D), Floyd-Steinberg algorithm (Floyd), and All-pole filter. We build a simulation framework to evaluate the effectiveness of our approach. K different FU types, F_1, \dots, F_K , are used in the system, in which a FU with type F_1 is the quickest with the highest failure rate and

a FU with type F_K is the slowest with the lowest failure rate. Each task node has different failure rates under different FU types. Due to the page limit, we don't list the detail table in this paper.

We conducted experiments on four methods: Method 1: list scheduling; Method 2: the TBFT algorithm in [9]; Method 3: our algorithm FU_Sch1; Method 4: our algorithm FU_Sch2. In the list scheduling, the priority of a node is set as the longest path from this node to a leaf node [12]. The experiments are performed on a Dell PC with a P4 2.1 G processor and 512 MB memory running Red Hat Linux 9.0.

The experimental results for the four methods are shown in Table II to Table III when the number of FUs is 3 and 4, respectively. Column "Bench." stands for the benchmarks we used in the experiments. Column "N." represents the number of nodes of each filter benchmark. Column "M1" to "M4" represents the four methods we used in the experiments. Column "E" represents the minimum total system failure rate obtained from four different algorithms. Column "% M1", "% M2" under "M3" and "M4" represents the percentage of reduction in total failure rate, compared to Method 1 and Method 2, respectively. The average reduction

Four Methods Comparison on IIR with 3 FUs								
Time	M1	M2	M3			M4		
	E (μ J)	E (μ J)	E (μ J)	% M1 (%)	% M2 (%)	E (μ J)	% M1 (%)	% M2 (%)
300	2702	2630	2329	13.8	11.4	2256	16.5	14.2
400	2650	2578	2221	16.2	13.9	2176	17.9	15.6
500	2627	2473	2136	18.7	13.6	2067	21.3	16.4
600	2603	2356	2017	22.5	14.4	1937	25.6	17.8
700	2578	2302	1939	24.8	15.8	1872	27.4	18.7
800	2527	2222	1867	26.1	16.0	1787	29.3	19.6
900	2478	2148	1802	27.3	16.1	1712	30.9	20.3
1000	2456	2135	1751	28.7	18.0	1678	31.7	21.4
1100	2431	2106	1707	29.8	19.0	1641	32.5	22.1
1200	2381	2046	1655	30.5	19.1	1568	33.5	22.5

TABLE IV

THE COMPARISON OF TOTAL FAILURE RATE WITH FOUR METHODS ON IIR FILTER UNDER DIFFERENT TIME CONSTRAINTS.

is shown in the last row of the table.

The results show that our algorithms for the FRMFS problem can significantly improve the performance of applications on WSNs. All of our algorithms (Method 3 and Method 4) improve the failure-rate reduction over the traditional list scheduling algorithm. Among them, FU_Sch2 (Method 4) gives the best performance. We can see that with more FUs selections, the reduction ratio for the total failure rate has increased. For example, with 3 FUs, compared with Method 1, FU_Sch2 (Method 4) shows an average 31.1% reduction in total failure rate. While using 4 FUs, the reduction rate changed to be 34.3% for total failure rate.

It is worthwhile to pointing out that we obtain this improvement ratio without sacrificing performance. For example, Table IV shows the failure rate vs. time constraints by benchmark IIR filter. FU_Sch2 (Method 4) improves the failure-rate reduction significantly when the time constraint is large. If the time constraint is small, it still improves the failure-rate reduction while meeting the constraint. FU_Sch1 (Method 3) can always improve failure-rate reduction as long as there exists a schedule by the ALAP scheduling. The improvement is not as significant as FU_Sch2 (Method 4) when the time constraint is large. This is because the algorithm always tries to use all available FUs in each step, while FU_Sch2 (Method 4) schedules one node in each step and avoid FUs with high failure rate if possible. Both FU_Sch1 and FU_Sch2 are novel algorithms. FU_Sch1 is more straight forward and has lower complexity. FU_Sch2 can achieve better results but is more suitable for smaller systems. Both these two algorithms can give solutions that are very close to the optimal solutions (less than 15% to 5% difference in average) in most cases.

V. CONCLUSION

In this paper, we studied the scheduling problem that minimizes the total failure rate of a system without sacrificing performance on WSNs. We proposed two highly efficient algorithms to solve the FRMFS (*Failure Rate Minimization with FU Scheduling*) problem for WSN applications. By using multi-FU scheduling, our algorithms improve both the

performance and failure-rate-minimization of WSN applications. A wide range of benchmarks have been tested on the experiments. The experimental results showed that our algorithms significantly improved the failure-rate-minimization while satisfying performance requirements for applications on WSNs.

REFERENCES

- [1] G. Toll, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, and et al., "A macroscope in the redwoods," in *ACM SenSys'05, San Diego, CA, USA*, Nov. 2-4, 2005, pp. 51–63.
- [2] G. Werner, P. Swieskowski, and M. Welsh, "Demonstration: real-time volcanic earthquake localization," in *ACM SenSys'06, Boulder, CO, USA*, Nov. 1-3, 2006, pp. 357–358.
- [3] L. Laffea, R. Monson, R. Han, R. Manning, A. Glasser, S. Oncley, and et al., "Comprehensive monitoring of CO₂ sequestration in subalpine forest ecosystems and its relation to global warming," in *ACM SenSys'06, Boulder, CO, USA*, Nov. 1-3, 2006, pp. 423–424.
- [4] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrinet, "Habitat monitoring with sensor networks," *Communications of ACM*, vol. 47, no. 6, pp. 34–40, 2004.
- [5] P. Zhang, C. M. Sadler, S. Lyon, and M. Martonosi, "Hardware design experiences in zebrant," in *ACM SenSys'04, Baltimore, Maryland, USA*, Nov. 3-5, 2004.
- [6] P. Sikka, P. Corke, P. Valencia, C. Crossman, D. Swain, and G. B.-Hurlley, "Wireless adhoc sensor and actuator networks on the farm," in *ACM SenSys'06, Boulder, CO, USA*, Nov. 1-3, 2006, pp. 492–499.
- [7] D. S. Glaser, "Some real-world applications of wireless sensor nodes," in *SPIE Symposium on smart structures and materials, San Diego, CA, USA*, Mar. 2004.
- [8] S. B. Eisenman and A. T. Campbell, "Poster abstract: skiscape sensing," in *ACM SenSys'06, Boulder, CO, USA*, Nov. 1-3, 2006, pp. 401–402.
- [9] S. Srinivasan and N. K. Jha, "Safety and reliability driven task allocation in distributed systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, pp. 238–251, Mar. 1999.
- [10] S. Borkar, N. Jouppi, and P. Stenstrom, "Microprocessors in the era of terascale integration," in *IEEE/ACM DATE*, April 2007, pp. 1–6.
- [11] S. M. Shatz, J.-P. Wang, and M. Goto, "Task allocation for maximizing reliability of distributed computer systems," *IEEE Trans. on Computers*, vol. 41, pp. 1156–1168, Sep. 1992.
- [12] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.