Efficent Algorithm of Energy Minimization for Heterogeneous Wireless Sensor Network *

Meikang Qiu[1], Chun Xue[1], Zili Shao[2], Qingfeng Zhuge[1], Meilin Liu[1], and Edwin H.-M. Sha[1]

[1] Univ. of Texas at Dallas,
Richardson, Texas 75083, USA
{mxq012100, cxx016000, qfzhuge, mxl024100, edsha}@utdallas.edu
[2] Hong Kong Polytechnic Univ.,
Hung Hom, Kowloon, Hongkong
cszshao@comp.polyu.edu.hk

Abstract. Energy and delay are critical issues for wireless sensor networks since most sensors are equipped with non-rechargeable batteries that have limited lifetime. Due to the uncertainties in execution time of some tasks, this paper models each varied execution time as a probabilistic random variable and incorporating applications' performance requirements to solve the MAP (Mode Assignment with Probability) problem. Using probabilistic design, we propose an optimal algorithm to minimize the total energy consumption while satisfying the timing constraint with a guaranteed confidence probability. The experimental results show that our approach achieves significant energy saving than previous work. For example, our algorithm achieves an average improvement of 32.6% on total energy consumption.

1 Introduction

Recent advances in heterogeneous wireless communications and electronics have enabled the development of low cost, low power, multifunctional sensor nodes that are small in size and communicate in short distances. These tiny sensor nodes have capability to sense, process data, and communicate. Typically they are densely deployed in large numbers, prone to failures, and their topology changes frequently. They have limited power, computational capacity, bandwidth and memory. As a result of its properties traditional protocols cannot be applied in this domain. Sensor networks have wide applications in areas such as health care, military, collecting information in disaster prone areas and surveillance applications [1–4].

Lifetime of distributed micro sensor nodes is a very important issue in the design of sensor networks. The wireless sensor node, being a microelectronic

* This work is partially supported by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, NSF CCR-0309461, NSF IIS-0513669, and Microsoft, USA.

device, can only be equipped with a limited power source ($\leq$ 0.5 Ah, 1.2 V). In some application scenarios, replenishment of power resources might be impossible. Hence, power conservation and power management take on additional importance [5]. Optimal energy consumption, i.e., minimizing energy consumed by sensing and communication to extend the network lifetime, is an important design objective. In the data transmisson, real-time is a critical requirement for many application for wireless sensor network. There are three modes (active, vulnerable, and sleep) for a sensor network. We call it as *Mode Assignment with Probability* (MAP) problem. For example, in a Bio-sensor, we sample the temperature every minutes. The data collected will go through a fixed topology to the destination. Assume we need the data transmisson within 20 seconds. Given this requirement, we need to minimize the total energy consumed in each transmisson. Due to the transmisson line situation and other overheads, the execution time of each transmision is not a fix number. It may tranmit a data in 1 seconds with 0.8 probability and in 3 seconds with 0.2 probability. The mode of a sensor node will affect both the energy and delay of the node.

This paper presents assignment and optimization algorithms which operate in probabilistic environments to solve the MAP problem. In the MAP problem, we model the execution time of a task as a random variable [6]. For heterogeneous systems, each node has different energy consumption rate, which related to area, size, reliability, etc. [7]. Faster one has higher energy consumption while slower one has lower consumption. This paper shows how to assign a proper mode to each node of a *Probability Data Flow Graph* (PDFG) such that the total energy consumption is minimized while the timing constraint is satisfied with a guaranteed confidence probability. With confidence probability P, we can guarantee that the total execution time of the DFG is less than or equal to the timing constraint with a probability that is greater than or equal to P.

Our contributions are listed as the following: 1) our algorithm *MAP_Opt* gives the optimal solution and achieves significant energy saving than *MAP_CP* algorithm. 2) Our algorithm not only is optimal, but also provides more choices of smaller total energy consumption with guaranteed confidence probabilities satisfying timing constraints. In many situations, algorithm *MAP_CP* cannot find a solution, while ours can find satisfied results. 3) Our algorithm is practical and quick.

The rest of the paper is organized as following: The models and basic concepts are introduced in Section 2. In Section 3, we give a motivational example. In Section 4, we propose our algorithms. The experimental results are shown in Section 5, and the conclusion is shown in Section 6.

2 System Model

System Model: *Probabilistic Data-Flow Graph* (PDFG) is used to model a DSP application. A ***PDFG G*** $= \langle V, E, T, R \rangle$ is a *directed acyclic graph* (DAG), where $\mathbf{V} = \langle v_1, \cdots, v_i, \cdots, v_N \rangle$ is the set of nodes; $\mathbf{M} = \langle M_1, \cdots, M_j, \cdots, M_R \rangle$ is a mode set; the execution time $\mathbf{T_{R_j}}(\mathbf{v})$ is a random variable; $\mathbf{E} \subseteq V \times V$ is the edge set that defines the precedence relations among nodes in V. There is a timing constraint L and it must be satisfied for executing the whole PDFG.

In sensor netowrk, we know that there are three kinds of mode, ie. active, vulnerable, and sleep modes. We assume under same mode, the energy consumption is same, while the execution time can be a random variable. Also, we assume that from source to destionation there is a fixed steps to go through before a node stop working. The Data Flow Graph is assumed to be a DAG (Directed Acyclic Graph), that is, there is no cycle in it.

Definitions: We define the ***MAP (Mode Assignment with Probability)*** problem as follows: Given R different voltage levels: M_1,M_2,$\cdots$,M_R, a PDFG $G = \langle V, E \rangle$ with $T_{M_j}(v)$, $P_{M_j}(v)$, and $C_{M_j}(v)$ for each node $v \in V$ executed on each mode M_j, a timing constraint L and a confidence probability P, find the mode for each node in assignment A that gives the *minimum total energy consumption C with confidence probability P under timing constraint L.*

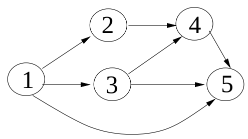
3 Motivational Example

In our model, under the same mode (M), the execution time (T) of a task is a random variable, which is usually due to condition instructions or operations that could have different execution times for different inputs. The energy consumption (C) depends on the mode M. Under different modes, a task has different energy consumptions. The execution time of a node in active mode is less than that of it in vulnerable mode, and they both are less than the execution time of it in sleep mode; The relations of energy consumption are just the reverse. This paper shows how to assign a proper mode to each node of a *Probabilistic Data-Flow Graph* (PDFG) such that the total energy consumption is minimized while satisfying the timing constraint with a guaranteed confidence probability.

An exemplary PDFG is shown in Figure 1(a). Each node can select one of the three different modes: M_1 (active), M_2 (vulnerable) , and M_3 (sleep). The execution times (T), corresponding probabilities (P), and energy consumption (C) of each node under different modes are shown in Figure 1(b). The input DAG (*Directed Acyclic Graph*) has five nodes. Node 1 is a multi-child node, which has three children: 2, 3, and 5. Node 5 is a multi-parent node, and has three parents: 1, 3, and 4. The execution time T of each node is modeled as a random variable. For example, When choosing M_1, node 1 will be finished in 1 time unit with probability 0.8 and will be finished in 2 time units with probability 0.2. Node 1 is the source and node 5 is the destination or the drain.

In sensor network application, a real-time system does not always has hard deadline time. The execution time can be smaller than the hard deadline time with certain probabilities. So the hard deadline time is the worst-case of the varied smaller time cases. If we consider these time variations, we can achieve a better minimum energy consumption with satisfying confidence probabilities under timing constraints.

For Figure 1, the minimum total energy consumptions with computed confidence probabilities under the timing constraint are shown in Table 1. The results



(a)

Nodes	M1 T	M1 P	M1 C	M2 T	M2 P	M2 C	M3 T	M3 P	M3 C
1	1	0.8	9	3	0.9	3	5	0.7	1
	2	0.2		4	0.1		6	0.3	
2	1	1.0	8	2	1.0	6	3	1.0	2
3	1	1.0	8	2	1.0	6	3	1.0	2
4	1	0.7	8	2	0.9	4	5	0.9	2
	3	0.3		4	0.1		6	0.1	
5	1	0.9	10	3	0.8	3	5	0.8	1
	2	0.1		4	0.2		6	0.2	

(b)

Fig. 1. (a) A sensor network topology. (b) The times, probabilities, and energy consumptions of its nodes in different modes.

are generated by our algorithm, *MAP_Opt*. The entries with probability that is equal to 1 (see the entries in boldface) actually give the results to the hard real-time problem which shows the worst-case scenario of the MAP problem. For each row of the table, the C in each (P, C) pair gives the minimum total energy consumption with confidence probability P under timing constraint j. For example, using our algorithm, at timing constraint 12, we can get (0.81, 14) pair. The assignments are shown in Table 2. We change the mode of nodes 2 and 3 to be M_2. Hence, we find the way to achieve minimum total energy consumption 14 with probability 0.81 satisfying timing constraint 12. While using the heuristic algorithm *MAP_CP* [8], the total energy consumption obtained is 28. Assignment $A(v)$ represents the voltage selection of each node v. We will prove that the results obtained by algorithm *MAP_Opt* are always optimal.

4 The Algorithms For MAP Problem

4.1 Definitions and Lemma

To solve the MAP problem, we use dynamic programming method traveling the graph in a bottom up fashion. For the easiness of explanation, we will index the nodes based on bottom up sequence. For example, Figure 2 (a) shows nodes indexed in a bottom up sequence After topological sorting the Figure 1 (a), that is, $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5$.

Given the timing constraint L, a PDFG G, and an assignment A, we first give several definitions as follows: 1) $\mathbf{G^i}$: The sub-graph rooted at node v_i, containing all the nodes reached by node v_i. In our algorithm, each step will add one node which becomes the root of its sub-graph. For example, G^3 is the graph containing nodes 1, 2, and 3 in Figure 2 (a). 2) In our algorithm, table $\mathbf{D_{i,j}}$ will be built. Each entry of table $D_{i,j}$ will store a link list of (Probability, Consumption) pairs sorted by probability in an ascending order. Here we define the **(Probability, Consumption) pair** $(\mathbf{P_{i,j}}, \mathbf{C_{i,j}})$ as follows: $C_{i,j}$ is the minimum energy consumption of $C_A(G^i)$ computed by all assignments A satisfying $T_A(G^i) \leq j$ with probability $\geq P_{i,j}$.

T	(P , C)	(P , C)	(P , C)	(P , C)	(P , C)
4	0.50, 43				
5	0.65, 39				
6	0.65, 35	0.81, 39			
7	0.65, 27	0.73, 33	0.81, 35	0.90, 39	
8	0.81, 27	0.90, 35	**1.00, 43**		
9	0.58, 20	0.73, 21	0.81, 27	0.90, 32	**1.00, 39**
10	0.72, 20	0.81, 21	0.90, 28	**1.00, 36**	
11	0.65, 14	0.90, 20	**1.00, 32**		
12	0.81, 14	0.90, 20	**1.00, 28**		
13	0.65, 12	0.90, 14	**1.00, 20**		
14	0.81, 12	0.90, 14	**1.00, 20**		
15	0.50, 10	0.90, 12	**1.00, 14**		
16	0.72, 10	0.90, 12	**1.00, 14**		
17	0.90, 10	**1.00, 12**			
18	0.50, 8	0.90, 10	**1.00, 12**		
19	0.72, 8	**1.00, 10**			
20	0.90, 8	**1.00, 10**			
21	**1.00, 8**				

Table 1. Minimum total energy consumptions with computed confidence probabilities under various timing constraints.

We introduce the ***operator* "⊕"** in this paper. For two (Probability, Consumption) pairs H_1 and H_2, if H_1 is ($P^1_{i,j}$, $C^1_{i,j}$), and H_2 is ($P^2_{i,j}$, $C^2_{i,j}$), then after applying the ⊕ operation between H_1 and H_2, we get pair ($P^{'}$, $C^{'}$), where $P^{'}$ = $P^1_{i,j}$ * $P^2_{i,j}$ and $C^{'}$ = $C^1_{i,j}$ + $C^2_{i,j}$. We denote this operation as "$\mathbf{H_1 \oplus H_2}$".

$D_{i,j}$ is the table in which each entry has a link list that stores pair ($P_{i,j}$, $C_{i,j}$) sorted by $P_{i,j}$ in an ascending order. Here, i represents a node number, and j represents time. For example, a link list can be (0.1, 2)→(0.3, 3)→(0.8, 6)→(1.0, 12). Usually, there are redundant pairs in a link list. We use Lemma 1 to cancel redundant pairs.

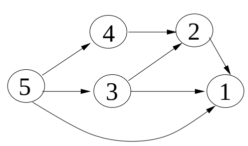
Lemma 1. *Given ($P^1_{i,j}$, $C^1_{i,j}$) and ($P^2_{i,j}$, $C^2_{i,j}$) in the same list:*

1. *If $P^1_{i,j} = P^2_{i,j}$, then the pair with minimum $C_{i,j}$ is selected to be kept.*
2. *If $P^1_{i,j} < P^2_{i,j}$ and $C^1_{i,j} \geq C^2_{i,j}$, then $C^2_{i,j}$ is selected to be kept.*

For example, if we have a list with pairs (0.1, 2) → (0.3, 3) → (0.5, 3) → (0.3, 4), we do the redundant-pair removal as following: First, sort the list according $P_{i,j}$ in an ascending order. This list becomes to (0.1, 2) → (0.3, 3) → (0.3, 4) → (0.5, 3). Second, cancel redundant pairs. Comparing (0.1, 2) and (0.3, 3), we keep both. For the two pairs (0.3, 3) and (0.3, 4), we cancel pair (0.3, 4) since the cost 4 is bigger than 3 in pair (0.3, 3). Comparing (0,3, 3) and (0.5, 3), we cancel (0.3, 3) since $0.3 < 0.5$ while $3 \geq 3$. There is no information lost in redundant-pair removal. Using Lemma 1, we can cancel many redundant-pair ($P_{i,j}$, $C_{i,j}$) whenever we find conflicting pairs in a list during a computation.

		Node id	T	M	Prob.	Consum.
Ours	$A(v)$	1	3	M_2	0.90	3
		2	3	M_3	1.00	2
		3	3	M_3	1.00	2
		4	2	M_2	0.90	4
		5	4	M_2	1.00	3
	Total		12		0.81	14
MAP_CP	$A(v)$	1	2	M_1	1.00	9
		2	2	M_2	1.00	6
		3	2	M_2	1.00	6
		4	4	M_2	1.00	4
		5	4	M_2	1.00	3
	Total		12		1.00	28

Table 2. The assignments of algorithms *MAP_Opt* and *MAP_CP* with timing constraint 12.



(a)

Nodes	M1			M2			M3		
	T	P	C	T	P	C	T	P	C
5	1	0.8	9	3	0.9	3	5	0.7	1
	2	0.2		4	0.1		6	0.3	
4	1	1.0	8	2	1.0	6	3	1.0	2
3	1	1.0	8	2	1.0	6	3	1.0	2
2	1	0.7	8	2	0.9	4	5	0.9	2
	3	0.3		4	0.1		6	0.1	
1	1	0.9	10	3	0.8	3	5	0.8	1
	2	0.1		4	0.2		6	0.2	

(b)

Fig. 2. (a) The resulted DAG after topological sorting of Figure 1 (a). (b) The times, probabilities, and energy consumptions of its nodes in different modes.

In every step of our algorithm, one more node will be included for consideration. The information of this node is stored in local table $\mathbf{E_{i,j}}$, which is similar to table $D_{i,j}$, but with accumulative probabilities only on node v_i. A local table only store information, such as probabilities and consumptions, of a node itself. Table $E_{i,j}$ is the local table only storing the information of node v_i. In more detail, $E_{i,j}$ is a local table of link lists that store pair $(p_{i,j}, c_{i,j})$ sorted by $p_{i,j}$ in an ascending order; $c_{i,j}$ is the energy consumption only for node v_i with timing constraint j, and $p_{i,j}$ is CDF (cumulative distributive function) F(j).

4.2 The MAP_CP Algorithm

In this subsection, we first design an heuristic algorithm for sensor network according to the *DFG_Assign_CP* algorithm in [8], we call this algorithm as *MAP_CP*.

The MAP_CP Algorithm

Algorithm 4.1 Heuristic algorithm for the MAP problem when the PDFG is DAG (*MAP_CP*)

Require: R different mode types, a DAG, and the timing constraint L.
Ensure: a mode assignment to minimize energy while satisfying L.

Assign the best energy type to each node and mark the type as assigned.
Find a CP that has the maximum execution time among all possible paths based on the current assigned types for the DAG.
For every node v_i in CP,
    for every unmarked type p,
        change its type to p,
        calculate $r = cost\_increase/time\_reduce$
        select the minimum r.
        if $(T > L)$
            contiune
        else
            exit /* This is the best assignment */

A critical path (CP) of a DAG is a path from source to its destionation. To be a leagl assignment for a DFG (*Data Flow Graph*), the execution time for any critical path should be less than or equal to the given timing constraint. In algorithm *MAP_CP*, we only consider the hard execution time of each node, that is, the case when the probability of the random variable T equals 1. This is a heuristic solution for hard real-time systems. We find the CP with minimized energy consumption first, then adjust the energy of the nodes in CP until the total execution time is $\leq L$.

4.3 The MAP_Opt Algorithm

We propose our algorithm, *MAP_Opt*, for sensor network, which shown as follows.
The MAP_Opt Algorithm

Require: R different modes, a DAG, and the timing constraint L.
Ensure: An optimal mode assignment

1. Topological sort all the nodes, and get a sequence A.
2. Count the number of multi-parent nodes t_{mp} and the number of multi-child nodes t_{mc}. If $t_{mp} < t_{mc}$, use bottom up approach; Otherwise, use top down approach.
3. For bottom up approach, use the following algorithm. For top down approach, just reverse the sequence.
4. If the total number of nodes with multi-parent is t, and there are maximum K variations for the execution times of all nodes, then we will give each of these t nodes a fixed assignment.
5. For each of the K^t possible fixed assignments, assume the sequence after topological sorting is $v_1 \to v_2 \to \cdots \to v_N$, in bottom up fashion. Let $D_{1,j} = E_{1,j}$. Assume $D^{'}_{i,j}$ is the table that stored minimum total energy

consumption with computed confidence probabilities under the timing constraint j for the sub-graph rooted on v_i except v_i. Nodes $v_{i_1}, v_{i_2}, \cdots, v_{i_R}$ are all child nodes of node v_i and R is the number of child nodes of node v_i, then

$$D^{'}_{i,j} = \begin{cases} (0,0) & \text{if } R = 0 \\ D_{i_1,j} & \text{if } R = 1 \\ D_{i_1,j} \oplus \cdots \oplus D_{i_R,j} & \text{if } R \geq 1 \end{cases} \tag{1}$$

6. Then, for each k in $E_{i,k}$.

$$D_{i,j} = D^{'}_{i,j-k} \oplus E_{i,k} \tag{2}$$

7. For each possible fixed assignment, we get a $D_{N,j}$. Merge the (Probability, Consumption) pairs in all the possible $D_{N,j}$ together, and sort them in ascending sequence according probability.
8. Then use the Lemma 1 to remove redundant pairs. Finally get $D_{N,j}$.

In algorithm *MAP_Opt*, we exhaust all the possible assignments of multi-parent or multi-child nodes. Without loss of generality, assume we using bottom up approach. If the total number of nodes with multi-parent is t, and there are maximum K variations for the execution times of all nodes, then we will give each of these t nodes a fixed assignment. We will exhausted all of the K^t possible fixed assignments. Algorithm *MAP_Opt* gives the optimal solution when the given PDFG is a DAG. In equation (1), $D_{i_1,j} \oplus D_{i_2,j}$ is computed as follows. let $G^{'}$ be the union of all nodes in the graphs rooted at nodes v_{i_1} and v_{i_2}. Travel all the graphs rooted at nodes v_{i_1} and v_{i_2}. For each node a in $G^{'}$, we add the energy consumption of a and multiply the probability of a to $D^{'}_{i,j}$ for only once, because each node can only have one assignment and there is no assignment conflict. The final $D_{N,j}$ we get is the table in which each entry has the minimum energy consumption with a guaranteed confidence probability under the timing constraint j.

In algorithm *MAP_Opt*, there are K^t loops and each loop needs $O(|V|^2 * L * R * K)$ running time. The complexity of *Algorithm MAP_Opt* is $O(K^{t+1} * |V|^2 * L * R)$. Since t_{mp} is the number of nodes with multi-parent, and t_{mc} is the number of nodes with multi-child, then $t = min(t_{mp}, t_{mc})$. $|V|$ is the number of nodes, L is the given timing constraint, R is the maximum number of modes for each node, and K is the maximum number of execution time variation for each node. The experiments show that algorithm *MAP_Opt* runs efficiently.

5 Experiments

This section presents the experimental results of our algorithms. We conduct experiments on a set of DAGs. Three different modes, M_1, M_2, and M_3, are used in the system, in which a node with mode M_1 (active) is the quickest with the highest energy consumption and a node with type M_3 (sleep) is the slowest with the lowest energy consumption. The execution times, probabilities, and energy consumptions for each node are randomly assigned. The experiments are performed on a Dell PC with a P4 2.1 G processor and 512 MB memory running Red Hat Linux 9.

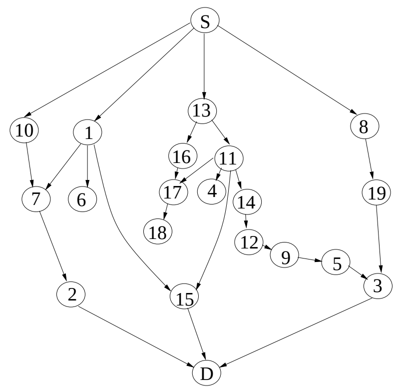


Fig. 3. The Data Flow Graph of exp1.

TC	*MAP_CP*	*MAP_Opt*					
		0.7		0.8		0.9	
	Energy	Energy	Saving	Energy	Saving	Energy	Saving
30	×	5202		×		×	
40	×	5190		5191		5192	
50	×	4721		4725		5188	
60	**5186**	3602	30.5%	3994	23.0%	3995	23.0%
70	**5180**	2646	48.9%	3112	39.9%	3586	30.8%
80	**2395**	1042	59.8%	1512	58.0%	2072	13.6%
100	**3111**	1042	66.4%	1509	49.3%	1509	38.6%
120	**3109**	1042	66.5%	1042	66.5%	1509	51.5%
136	**1509**	1042	30.9%	1042	30.9%	1042	30.9%
137	**1042**	1042		1042		1042	
Average Saving			51.5%		43.8 %		32.6 %

Table 3. Experimental results of algorithms *MAP_CP* and *MAP_Opt* for exp1.

Figure 3 shows a DAG with 21 nodes. We assume this is the topology of a sensor network. S is the source and D is the destination. Each node has three modes with different execution times and energy consumptions. The collected data need to go through the topology to the destination within a timing constaint. Exclude the source and destination node, this DAG has 3 multi-child nodes and 4 multi-parent nodes. Using top-down approach, we implemented all $3^3 = 27$ possibilities. The experimental results for exp1 is shown in Table 3.

Column "TC" stands for the timing constraint of the DAG. Column "Saving" shows the percentage of reduction on system energy consumptions, compared the results for soft real-time with those for hard real-time. The average percentage reduction is shown in the last row "Average Saving" of the table. The entry with "×" means no solution available. Under timing constraint 30 in Table 3, there is no solution for hard real-time using MAP_CP algorithm. However, we can find solution 5202 with probability 0.9 that guarantees the total execution time of the DFG is less than or equal to the timing constraint 30.

6 Conclusion

This paper proposed a probability approach for real-time sensor network applications to assign and optimize sensor systems using heterogeneous functional units with probabilistic execution time. The systems become very complicate when considering probability in execution time. For the *Mode assignment with probability* (MAP) problem, One optimal algorithms was proposed to solve it. Experiments showed that our algorithm provides more design choices to achieve minimum total cost while the timing constraint is satisfied with a guaranteed confidence probability.

References

1. Akyildiz I.F., Su W., Sankarasubramaniam Y., Cayirci E.: A Survey on Sensor Networks. IEEE Communications Magazine, Vol. 40, No. 8, (Aug. 2002) 102–116
2. Tan H., Lu I.: Power efficient data gathering and aggregation in wireless sensor networks. ACM SIGMOD Record, SPECIAL ISSUE: Special section on sensor network technology and sensor data management, Vol. 4, No. 3, (2003) 66–71
3. Law Y. W., Hoesel L. , Doumen J. , Havinga P.: Sensor networks: Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks SASN '05, Alexandria, VA, USA (Nov. 2005) 76–88
4. Chatzigiannakis I., Kinalis A., Nikoletseas S.: Power Conservation Schemes for Energy Efficient Data Propagation in Heterogeneous Wireless Sensor Networks. Proceedings of the 38th annual Symposium on Simulation, (Apr. 2005) 60–71
5. Kumar S., Lai T. H., Balogh J.: On k-coverage in a mostly sleeping sensor network. Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (Mobicom '04) (2004) 144–158
6. Tongsima S., Sha Edwin H.-M., Chantrapornchai C., Surma D., Passose N.: Probabilistic Loop Scheduling for Applications with Uncertain Execution Time. IEEE Trans. on Computers, Vol. 49, (Jan. 2000) 65-80
7. Li W. N., Lim A., Agarwal P., Sahni S.: On the Circuit Implementation Problem. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, (Aug. 1993) 1147–1156
8. Shao Z., Zhuge Q. , Xue C., Sha Edwin H.-M.: Efficient Assignment and Scheduling for Heterogeneous DSP Systems. IEEE Trans. on Parallel and Distributed Systems, Vol. 16, (Jun. 2005) 516–525