

Energy Minimization with Soft Real-time and DVS for Uniprocessor and Multiprocessor Embedded Systems *

Meikang Qiu¹ Chun Xue¹ Zili Shao² Edwin H.-M. Sha¹

¹Department of Computer Science

University of Texas at Dallas

Richardson, Texas 75083, USA

{mxq012100, cxx016000, edsha}@utdallas.edu

²Department of Computing

Hong Kong Polytechnic University

Hung Hom, Kowloon, Hong Kong

cszshao@comp.polyu.edu.hk

Abstract

Energy-saving is extremely important in real-time embedded systems. Dynamic Voltage Scaling (DVS) is one of the prime techniques used to achieve energy-saving. Due to the uncertainties in execution times of some tasks of systems, this paper models each varied execution time as a random variable. By using probabilistic approach, we propose two optimal algorithms, one for uniprocessor and one for multiprocessor to explore soft real-time embedded systems and avoid over-designing them. Our goal is to minimize the expected total energy consumption while satisfying the timing constraint with a guaranteed confidence probability. The solutions can be applied to both hard and soft real-time systems. The experimental results show that our approach achieves significant energy-saving than previous work.

1 Introduction

Power and energy reductions are critical for real-time embedded systems. In practice, many of these systems can tolerate occasional deadline misses and some tasks in them may not have fixed execution time. Such tasks usually contain conditional instructions and/or operations that could have different execution times for different inputs [2, 8]. It is possible to obtain the execution time distribution for each task by sampling and knowing detailed timing information about the system or by profiling the target hardware. Also some multimedia applications, such as image, audio, and video data streams, often tolerate occasional deadline misses without being noticed by human visual and auditory systems. For example, in packet audio applications, loss rates between 1% - 10% can be tolerated.

Prior approaches for hardware/software codesign of embedded systems guarantee no deadline missing by considering worst-case execution time of each task [3, 6]. These

approaches are pessimistic and only suitable for hard real-time systems, where any deadline miss will be catastrophic. They are not suitable for soft real-time systems, which can tolerate occasional violations of timing constraints. Using probabilistic approach [2, 8], we can take advantage of this feature and deliver higher performance with less energy consumption. Probabilistic approach can be applied to both hard real-time systems and soft real-time systems. Hard real-time is a special case of soft real-time when the probability equals to 1.

Dynamic voltage scaling (DVS) is one of the most effective techniques to reduce energy consumption [1, 4, 5, 7]. In many embedded systems, the supply voltage can be changed by mode-set instructions according to the workload. With the trend of multiple processors being widely used in embedded systems, it is important to study DVS techniques for multiprocessor systems.

In this paper, we use probabilistic approach [2, 8] and DVS to avoid over-designing systems. We propose two novel optimal algorithms, one for uniprocessor and one for multiprocessor embedded systems, to minimize expected value of total energy consumption while satisfying timing constraints with guaranteed probabilities for real-time applications. Hua et al. [2] proposed a heuristic algorithm for uniprocessor. Their data flow graph (DFG) is a simple path. We call the offline part of it as *HUA* algorithm for convenience. In this paper, we also apply the greedy method of *HUA* algorithm to multiprocessor and call the new algorithm as *Heu*.

Our contributions are listed as the following:

1) For uniprocessor, our algorithm *VAP_S* gives the optimal solution and achieves significant energy saving than *HUA* algorithm.

2) For the general problem, that is, multiprocessor, our algorithm, *VAP_M*, gives the optimal solution and achieves significant average energy reductions than the *Heu* algorithm.

3) Our algorithms are not only optimal, but also provide more choices of smaller *expected* total energy consumption with guaranteed confidence probabilities satisfying timing constraints. In many situations, algorithms *HUA* and *Heu*

*This work is partially supported by TI University Program, NSF EIA-0103709, Texas ARP 009741-0028-2001, NSF CCR-0309461, NSF IIS-0513669, Microsoft, USA, HK POLYU A-PH13, and A-PASX, HK.

cannot find a solution, while ours can find satisfied results.

4) Our algorithms are practical and quick. In practice, when the number of multi-parent nodes and multi-child nodes in the given *Probabilistic Data Flow Graph* (PDFG) graph is small, and the timing constraint is polynomial to the size of PDFG, the running times of these algorithms are very small and our experiments are always finished in very short time.

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 Models and Concepts

In this section, we introduce the system model, the energy model, and VAP problem for multiprocessor systems that will be used in the later sections.

System Model: We focus on real-time applications on uniprocessor and multiprocessor embedded systems. *Probabilistic Data-Flow Graph* (PDFG) is used to model an embedded systems application. A **PDFG** $G = \langle U, ED, T, V \rangle$ is a *directed acyclic graph* (DAG), where $U = \langle u_1, \dots, u_i, \dots, u_N \rangle$ is a set of nodes representing tasks; $V = \langle V_1, \dots, V_j, \dots, V_M \rangle$ is a voltage set; the execution time $T_{V_j}(u)$ is a random variable; $ED \subseteq U \times U$ is the edge set that defines the precedence relations among nodes in U . There is a timing constraint L and it must be satisfied for executing the whole PDFG. In multiprocessor systems, each processor has multiple discrete levels of voltages and its voltage level can be changed independently by voltage-level-setting instructions without the influence for other processors.

Energy Model: Dynamic power, which is the dominant source of power dissipation in CMOS circuit, is proportional to $N_c(u) \times C_s \times V_{dd}^2$, where $N_c(u)$ represent the number of computation cycles for node u , C_s is the effective switched capacitance, and V_{dd} is the supply voltage. Reducing the supply voltage can result in substantial power and energy saving. Roughly speaking, system's power dissipation is halved if we reduce V_{dd} by 30% without changing any other system parameters. However, this saving comes at the cost of reduced throughput, slower system clock frequency, or higher cycle period time (gate delay).

We use the similar energy model as in [4, 5, 7]. Let T represent the execution time of a node and E stand for energy consumption. The cycle period time T_c is proportional to $\frac{V_{dd}}{(V_{dd}-V_{th})^\alpha}$, where V_{th} is the threshold voltage and $\alpha \in (1.0, 2.0]$ is a technology dependent constant. Given the number of cycles N of node u , the supply voltage V_{dd} and the threshold voltage V_{th} , its computation time $T(u)$ and the energy $E(u)$ for node u are calculated as follows:

$$T_c = \frac{k \times V_{dd}}{(V_{dd} - V_{th})^\alpha} \quad (1)$$

$$T(u) = N_c(u) \times T_c = N_c(u) \times \frac{k \times V_{dd}}{(V_{dd} - V_{th})^\alpha} \quad (2)$$

$$E(u) = N_c(u) \times C_s \times V_{dd}^2 \quad (3)$$

In Equation (1), k is a device related parameter. From Equations (2) and (3), we can see that the lower voltage will prolong the execution time of a node but reduces its energy consumption.

VAP problem: Assume there are maximum M different voltages in a voltage set $V = \langle V_1, V_2, \dots, V_M \rangle$. For each voltage, there are maximum K execution time variations, although each node may have different execution time variations. An assignment for a PDFG G is to assign a voltage level to each node. In a PDFG G , each varied execution time is modeled as a probabilistic random variable, $T_{V_j}(u)$, $1 \leq j \leq M$, representing the execution times of each node $u \in U$ when running at voltage level V_j . $P_{V_j}(u)$ and $E_{V_j}(u)$ is corresponding probability and *expected* value of energy consumption.

We define the *voltage assignment with probability* (VAP) problem as follows: Given M different voltage levels: V_1, V_2, \dots, V_M , a PDFG $G = \langle U, ED \rangle$ with $T_{V_j}(u)$, $P_{V_j}(u)$, and $C_{V_j}(u)$ for each node $u \in U$ executed on each voltage V_j , a timing constraint L and a confidence probability P , find an assignment of voltage level for G that gives the *minimum expected total energy consumption E with confidence probability P under timing constraint L* .

3 Motivational Examples

First we give an example for multiprocessor embedded systems, which is shown in Figure 1. Figure 1(a) shows the input PDFG, and (b) shows the times, expected energy consumption, and probabilities of each node. Each node has two different voltage levels to choose from, and is executed on them with probabilistic execution times.

The number of computation cycles (N_c) for a task is proportional to the execution time. The energy consumption (E) depends on not only the voltage level V , but also the number of computation cycles N_c . We use the expected value of energy consumption ($Exp(E)$) as the energy consumption E under a certain voltage level V . Under different voltage levels, a task has different expected energy consumptions. The higher the voltage level is, the faster the execution time is, and the more expected energy is consumed. According to the energy model of DVS [7], the computation time is proportional to $V_{dd}/(V_{dd} - V_{th})^2$, where V_{dd} is the supply voltage, V_{th} is the threshold voltage; the energy consumption is proportional to V_{dd}^2 . So here we assume the computation time of a node under the low voltage (V_2) is twice as much as it is under the high voltage (V_1); the energy consumption of a node under the high voltage (V_1) is four times as much as it is under the low voltage (V_2).

We use a schedule to preprocess the input PDFG. After preprocessing, we get the scheduling graph in Figure 1(c), which is a DAG. There are two processors, PR_1 and PR_2 . Node 5 is a multi-child node, which has two children: 3 and

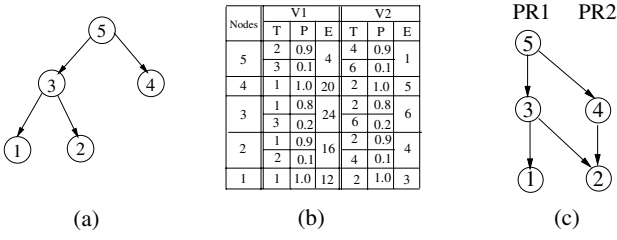


Figure 1. (a) A given PDFG. (b) The parameters of its nodes for different V . (c) The schedule graph.

4. Node 2 is a multi-parent node, and has two parents: 3 and 4. From Figure 1(b) we can compute the time *cumulative distribution functions* (CDFs) of each node at different voltage levels.

T	(P, E)	(P, E)	(P, E)	(P, E)
4	0.65, 76			
5	0.65, 43	0.72, 61		
6	0.72, 34	0.81, 61		
7	0.80, 34	0.90, 52		
8	0.72, 22	0.80, 34	1.00, 52	
9	0.80, 22	0.90, 40	1.00, 52	
10	0.72, 19	0.80, 22	0.90, 34	1.00, 40
11	0.72, 19	0.80, 22	1.00, 34	
12	0.80, 19	0.90, 22	1.00, 34	
13	0.80, 19	1.00, 22		
14	0.90, 19	1.00, 22		
15	0.90, 19	1.00, 22		
16	1.00, 19			

Table 1. Minimum expected total E with computed confidence P under various T for a DAG.

For Figure 1, the minimum total energy consumptions with computed confidence probabilities under the timing constraint are shown in Table 1. For each row of the table, the E in each (P, E) pair gives the minimum total energy consumption with confidence probability P under timing constraint T . Using our algorithm, VAP_M , at timing constraint 11, we can get (0.80, 22) pair. Table 2 shows the assignments of our algorithm. Assignment $A(u)$ represents the voltage selection of each node u . Using our algorithm, we achieve minimum total energy consumption 22 with probability 0.80 satisfying timing constraint 11. While using the heuristic algorithm HUA , the total energy consumption obtained is 61, because HUA still need to use voltage level V_1 and cannot change all node's voltage level to V_2 under timing constraint 11. The energy saving improvement of our algorithm is 59.1% in this case. Also, this case shows that in many situations, the solutions obtained by our algorithms have significant improvement compared with the results gotten by Heu .

		Node id	Time	V Level	Prob.	Energy
OuVs	$A(v)$	5	3	V1	1.00	4
		4	2	V2	1.00	5
		3	4	V2	0.80	6
		2	2	V2	1.00	4
		1	4	V2	1.00	3
		Total		11		0.80
HUA	$A(v)$	5	3	V1	1.00	4
		4	2	V1	1.00	5
		3	1	V1	0.80	24
		2	1	V1	1.00	12
		1	2	V1	1.00	16
		Total		11		0.80

Table 2. Under timing constraint 11, the different assignments between VAP_M and Heu .

4 The Algorithms

4.1 Definitions and Lemma

To solve the VAP problem, we use dynamic programming method traveling the graph in bottom up or top down fashion. For the ease of explanation, we will index the nodes based on bottom up sequence. For example, Figure 1 (a) shows a tree indexed by bottom up sequence. The sequence is: $u_1 \rightarrow u_2 \rightarrow \dots \rightarrow u_5$. Define a *root* node to be a node without any parent and a *leaf* node to be a node without any child. A multi-child node is a node with more than one child. For example, in Figure 1 (a), node 3 and 5 are multi-child nodes. Similarly, a multi-parent node is a node with more than one parent.

Given the timing constraint L , a PDFG G , and an assignment A , we first give several definitions as follows:

1) 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, in Figure 1 (a), G^3 is the tree containing nodes 1, 2, and 3.

2) $E_A(G^i)$ and $T_A(G^i)$: The total energy consumption and total execution time of G^i under the assignment A . In our algorithm, each step will achieve the minimum total energy consumption of G^i with computed confidence probabilities under various timing constraints.

3) Define the **(Probability, Energy)** pair $(P_{i,j}, E_{i,j})$ as follows: $E_{i,j}$ is the minimum energy consumption of $E_A(G^i)$ computed by all assignments A satisfying $T_A(G^i) \leq j$ with probability $\geq P_{i,j}$.

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

In every step in our algorithm, one more node will be included for consideration. The information of this node is stored in local table $B_{i,j}$. A local table store only data of probabilities and energy of a node itself. Table $B_{i,j}$ is the

local table storing only the data of node v_i . In more detail, $B_{i,j}$ is a local table of linked lists that store pair $(P_{i,j}, E_{i,j})$ sorted by $P_{i,j}$ in an ascending order; $E_{i,j}$ is the energy consumption only for node u_i at time j , and $P_{i,j}$ is the corresponding probability. The building procedures of $B_{i,j}$ are as follows. First, sort the execution time variations in an ascending order. Then, accumulate the probabilities of same type. Finally, let $L_{i,j}$ be the linked list in each entry of $B_{i,j}$, insert $L_{i,j}$ into $L_{i,j+1}$ while redundant pairs canceled out based on Lemma 4.1.

In our algorithm, table $D_{i,j}$ will be built. Each entry of table $D_{i,j}$ will store a linked list of (Probability, Energy) pairs sorted by probability in ascending order. In our algorithm, $D_{i,j}$ is the table in which each entry has a linked list that store pair $(P_{i,j}, E_{i,j})$ sorted by $P_{i,j}$ in an ascending order. Here, i represents a node number, and j represents time. For example, a linked list can be $(0.1, 2) \rightarrow (0.3, 3) \rightarrow (0.8, 6) \rightarrow (1.0, 12)$. Usually, there are redundant pairs in a linked list. We can use the following Lemma to cancel redundant pairs.

Lemma 4.1 Given $(P_{i,j}^1, E_{i,j}^1)$ and $(P_{i,j}^2, E_{i,j}^2)$ in the same list:

1. If $P_{i,j}^1 = P_{i,j}^2$, then the pair with minimum $E_{i,j}$ is selected to be kept.
2. If $P_{i,j}^1 < P_{i,j}^2$ and $E_{i,j}^1 \geq E_{i,j}^2$, then $E_{i,j}^2$ is selected to be kept.

4.2 The VAP_S Algorithm

The Algorithm for uniprocessor system is shown in VAP_S. Using dynamic programming, it can give the optimal solution for the VAP problem when there is only one processor.

The VAP_S Algorithm

In algorithm VAP_S, first build a local table $B_{i,j}$ for each node. Next, in step 2 of the algorithm, when $i = 1$, there is only one node. We set the initial value, and let $D_{1,j} = B_{1,j}$. Then using dynamic programming method, build the table $D_{i,j}$. For each node u_i under each time j , we try all the times k ($1 \leq k \leq j$) in table $B_{i,j}$. We use “ \oplus ” on the two tables $B_{i,k}$ and $D_{i-1,j-k}$. Since $k + (j - k) = j$, the total time of nodes from u_1 to u_i is j . The “ \oplus ” operation add the energy consumptions of two tables together and multiply the probabilities of two tables with each other. Finally, we use Lemma 4.1 to cancel the conflicting (Probability, Energy) pairs. The new energy consumption in each pair obtained in table $D_{i,j}$ is the energy consumption of current node u_i at time k plus the energy consumption in each pair obtained in $D_{i-1,j-k}$. Since we have used Lemma 4.1 canceling redundant pairs, the energy consumption of each pair in $D_{i,j}$ is the minimum total energy consumption for graph G^i with confidence probability $P_{i,j}$ under timing constraint j .

Algorithm 4.1 Optimal algorithm for the VAP problem when there is a uniprocessor (VAP_S)

Input: M different voltage levels, a DAG, and the timing constraint L .

Output: An optimal voltage assignment

1. Build a local table $B_{i,j}$ for each node of PDFG. $|V| \leftarrow N$, where $|V|$ is the number of nodes.
 2. let $D_{1,j} = B_{1,j}$
for each node $u_i, i > 1$ do
for each time j do
for each time k in $B_{i,k}$ do
if $D_{i-1,j-k} \neq NULL$ then
 $D_{i,j} = D_{i-1,j-k} \oplus B_{i,k}$
else
continue
end if
end for
insert $D_{i,j-1}$ to $D_{i,j}$ and remove redundant pairs using Lemma 4.1.
end for
end for
 3. return $D_{N,j}$
-

4.3 The Algorithms For Multiprocessor

In this subsection, we first give the algorithm of schedule-graph construction. Next, we give a heuristic algorithm *Heu*, then we propose our novel and optimal algorithm, VAP_M, for multiprocessor embedded systems. We will compare them in the experiments section.

Algorithm 4.2 Algorithm to get scheduling graph (VAP_SG)

Input: a task graph PDFG

Output: a scheduling graph

- 1: build a graph to show the order using list scheduling.
 - 2: show the dependency in the graph.
 - 3: remove all redundant edges according Lemma 4.2.
-

For an input PDFG with multiple processors, given the order of nodes and expected energy consumption of each node, the basic steps of our schedule are shown in algorithm VAP_SG. In the graph built in step 1 of VAP_SG, If there is an edge from u_i to u_j , this means that u_i is scheduled before u_j in the same processor or u_j depends on u_i in the original PDFG. The new graph is a DAG that represents the order of nodes and dependencies.

Lemma 4.2 For two nodes u_i and u_j , there is an edge e_{ij} , if we can find another separate path $u_i \rightarrow u_j$, then the edge e_{ij} can be deleted.

The Heu Algorithm

We first design an heuristic algorithm for multiprocessor systems according to the HUA algorithm in [2], we call this algorithm as *Heu*. The PDFG now is a DAG and no longer

Algorithm 4.3 Heuristic algorithm for the VAP problem when there are multiple processors and the PDFG is DAG (*Heu*)

Input: M different voltage levels, a DAG, and the timing constraint L .

Output: a voltage assignment to minimize energy with a guaranteed probability θ satisfying L

- 1: Scheduling graph construction.
- 2: for each vertex u_i , let $l_i = k_i$; /* assign worst-case time to u_i */
- 3: $P = 1$;
- 4: while ($P > \theta$)
- 5: { pick v_i that has the maximum $(t_{il_i} - t_{i(l_i-1)}) \cdot \frac{P_{il_i}}{P_{i(l_i-1)}}$;
- 6: $P = P \cdot \frac{P_{il_i}}{P_{i(l_i-1)}}$;
- 7: if ($P > \theta$)
- 8: $l_i = l_i - 1$;
- 9: $T = \sum t_{il_i}$; /* calculate the total execution time T */
- 10: if ($T > L$) exit; /* if θ cannot be met */
- 11: for each vertex u_i , let $S_i = t_{il_i} \cdot \theta / T$;

limited to a simple path. The authors of [2] did not give the algorithm for multiple processors situation, and their data flow graph is a simple path. *Heu* is an algorithm that use the idea of the algorithm of [2] and using for multiple processors situation, the data flow graph is a DAG. In *Heu* algorithm, k_i is the largest possible time variation for node u_i , l_i is the time variation of node u_i , S_i is the scaled time slot for node u_i , and t_{ij} is the j time variation of node u_i .

The VAP_M Algorithm

Input: M different voltage levels, a DAG, and the timing constraint L .

Output: An optimal assignment for the DAG

1. Scheduling graph construction.
2. Topological sort all the nodes, and get a sequence A .
3. 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.
4. For bottom up approach, use the following algorithm. For top down approach, just reverse the sequence.
5. 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.
6. For each of the K^t possible fixed assignments, Assume the sequence after topological sorting is $u_1 \rightarrow u_2 \rightarrow \dots \rightarrow u_N$, in bottom up fashion. Let $D_{1,j} = B_{1,j}$. 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 u_i except u_i . Nodes $u_{i_1}, u_{i_2}, \dots, u_{i_w}$ are all child nodes of node u_i and w is the number of child nodes of node u_i , then

$$D'_{i,j} = \begin{cases} (0, 0) & \text{if } w = 0 \\ D_{i_1,j} & \text{if } w = 1 \\ D_{i_1,j} \oplus \dots \oplus D_{i_w,j} & \text{if } w \geq 1 \end{cases} \quad (4)$$

7. For $D_{i_1,j} \oplus D_{i_2,j}$, G' is the union of all nodes in the

graphs rooted at nodes u_{i_1} and u_{i_2} . Travel all the graphs rooted at nodes u_{i_1} and u_{i_2} .

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

$$D_{i,j} = D'_{i,j-k} \oplus B_{i,k} \quad (5)$$

9. 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.

10. Then use the Lemma 4.1 to remove redundant pairs. $D_{N,j} \leftarrow$ a table of MIN(E) with $Prob.(T \leq j) \geq P$;
Output $D_{N,L}$.

Now we explain our optimal algorithm *VAP_M* in details. In *VAP_M*, 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 *VAP_M* gives the optimal solution when the given PDFG is a DAG. In the following, we give the Theorem 4.1 and Theorem 4.2 about this. Due to page limit, we will not give proofs in this paper.

Theorem 4.1 In each possible fixed assignment, for each pair $(P_{i,j}, E_{i,j})$ in $D_{i,j}$ ($1 \leq i \leq N$) obtained by algorithm *VAP_M*, $E_{i,j}$ is the minimum total energy consumption for the graph G^i with confidence probability $P_{i,j}$ under timing constraint j .

Theorem 4.2 For each pair $(P_{i,j}, E_{i,j})$ in $D_{N,j}$ ($1 \leq j \leq L$) obtained by algorithm *VAP_M*, $E_{i,j}$ is the minimum total energy consumption for the given DAG G with confidence probability $P_{i,j}$ under timing constraint j .

In algorithm *VAP_M*, there are K^t loops and each loop needs $O(|V|^2 * L * M * K)$ running time. The complexity of Algorithm *VAP_M* is $O(K^{t+1} * |V|^2 * L * M)$. Let t_{mp} be the number of nodes with multi-parent, and t_{mc} be 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, M is the maximum number of voltage levels for each node, and K is the maximum number of execution time variation for each node. The experiments show that algorithm *VAP_M* runs efficiently.

5 Experiments

This section presents the experimental results of our algorithms. We conduct experiments on a set of benchmarks including volterra filter, 4-stage lattice filter, 8-stage lattice filter, differential equation solver RLS-languerre lattice filter, and elliptic filter. Three different voltage levels, V_1 , V_2 , and V_3 , are used in the system, in which a processor under V_1 is the quickest with the highest energy consumption and a processor under V_3 is the slowest with the lowest

energy consumption. There are two processors: *PR1* and *PR2*. We compare our *VAP_M* algorithm with the *Heu* algorithm. The distribution of execution times of each node is Gaussian. For each benchmark, the first timing constraint we use is the minimum execution time. The experiments are performed on a Dell PC with a P4 2.1 G processor and 512 MB memory running Red Hat Linux 7.3.

RLS-Laguerre Filter (19 nodes)										
TC	0.8			0.9			1.0			
	Heu	Ours	%	Heu	Ours	%	Heu	Ours	%	
130	900	892	0.9	×	×		×	×		
136	900	804	10.6	900	892	0.9	×	×		
140	900	756	16.0	900	727	9.2	900	892	0.9	
180	900	408	54.6	900	430	52.2	900	444	50.7	
220	900	268	70.2	900	277	69.2	900	261	70.0	
260	225	218	3.2	900	248	72.4	900	251	72.1	
272	225	202	10.2	225	218	3.2	900	232	74.2	
280	225	186	17.2	225	212	6.0	225	218	3.2	
320	225	152	32.4	225	164	27.2	225	172	23.6	
360	225	94	58.2	225	98	56.4	225	106	52.9	
420	225	58	74.2	225	58	74.2	225	58	74.2	
Ave. Redu.(%)			54.3				57.6			59.2

Table 3. The minimum expected total E with computed confidence P under various T for RLS-Laguerre filter.

The experimental results of RLS-Laguerre filter are shown in Table 3. In the table, column “TC” represents the given timing constraint, “Heu” represents the heuristic algorithm *Heu*, and “Ours” represents our optimal algorithm *VAP_M*. The minimum total energy consumption obtained from different algorithms, *VAP_M* and *Heu*, are presented in each entry. Columns “1.0”, “0.9”, and “0.8”, represent that the confidence probability is 1.0, 0.9, and 0.8, respectively. Column “%” shows the percentage of reduction on the total energy consumption, compared the results of algorithm *Heu*. The average percentage reduction is shown in the last row “Ave. Redu(%)” of Table 3, which is computed by averaging energy reductions at all different timing constraints. The entry with “×” means no solution available.

We found in many situations, algorithm *VAP_M* has significant energy saving than algorithm *Heu*. For example, in Table 3, under the timing constraint 220, for probability 0.8, the entry under “Heu” is 900, which is the minimum total energy consumption using algorithm *Heu*. The entry under “Ours” is 268, which means by using *VAP_M* algorithm, we can achieve minimum total energy consumption 268 with confidence probability 0.8 under timing constraint 220, and the energy reduction is 70.2%.

The experimental results of energy-saving improvement of *VAP_M* over *Heu* for different DSP filters are shown in Table 4. “Node #” stands for the number of nodes of a PDFG. The experimental results show that our algorithm can greatly reduce the total energy consumption while have a guaranteed confidence probability. On average, algorithm *VAP_M* gives an energy-saving of 56.1% with confidence probability 0.8 satisfying timing constraints, and energy-savings of 59.3% and 61.7% with confidence probabilities

Filter	Node #	type	0.8	0.9	1.0
Volterra	27	Tree	58.6%	61.7%	64.8%
4_stage Lattice	26	Tree	55.9%	59.7%	62.4%
8_stage Lattice	42	Tree	59.5%	62.8%	65.2%
Diff. Equ. Solver	11	DAG	52.6%	55.3%	57.4%
RLS_Laguerre	19	DAG	54.3%	57.6%	59.2%
Elliptic	34	DAG	55.8%	58.7%	61.3%
Ave. Saving			56.1%	59.3%	61.7%

Table 4. Experimental results of energy-saving improvement of *VAP_M* over *Heu* for different DSP filters.

0.9 and 1.0 satisfying timing constraints, respectively. The experiments using *VAP_M* on these benchmarks are finished within several minutes using low-end PC.

6 Conclusion

This paper used the probabilistic approach to explore design space for real-time embedded systems. By taking advantage of the uncertainties in execution time of tasks, our approach relaxes the rigid hardware requirements for software implementation and eventually avoids over-designing the system. For the *voltage assignment with probability* (VAP) problem, by using *Dynamic Voltage Scaling* (DVS), we proposed two optimal algorithms, *VAP_S* and *VAP_M*, to give the optimal solutions for uniprocessor and multiprocessor embedded systems. Experimental results demonstrated the effectiveness of our approach.

References

- [1] Y. Chen, Z. Shao, Q. Zhuge, C. Xue, B. Xiao, and E. H.-M. Sha. Minimizing energy via loop scheduling and dvs for multi-core embedded systems. In *ICPADS, Volume II*, pages 2 – 6, 2005.
- [2] S. Hua, G. Qu, and S. S. Bhattacharyya. Energy reduction techniques for multimedia applications with tolerance to deadline misses. In *DAC*, pages 131–136, 2003.
- [3] K. Ito, L. Lucke, and K. Parhi. Ilp-based cost-optimal dsp synthesis with module selection and data format conversion. *IEEE Trans. on VLSI Systems*, 6:582–594, Dec. 1998.
- [4] T. Sakurai and A. R. Newton. Alpha-power law mosfet model and its application to cmos inverter delay and other formulas. *IEEE J. Solid-State Circuits*, SC-25(2):584–589, 1990.
- [5] H. Saputra, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, J. S. Hu, C.-H. Hsu, and U. Kremer. Energy-conscious compilation based on voltage scaling. In *LCTES’02*, June 2002.
- [6] Z. Shao, Q. Zhuge, C. Xue, and E. H.-M. Sha. Efficient assignment and scheduling for heterogeneous dsp systems. *IEEE Trans. on Parallel and Distributed Systems*, 16:516–525, Jun. 2005.
- [7] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *DAC*, pages 183–188, 2002.
- [8] T. Zhou, X. Hu, and E. H.-M. Sha. Estimating probabilistic timing performance for real-time embedded systems. *IEEE Trans. on VLSI Systems*, 9(6):833–844, Dec. 2001.