

# Voltage Assignment with Guaranteed Probability Satisfying Timing Constraint for Real-time Multiprocesor DSP

Meikang Qiu, Zhiping Jia, Chun Xue, Zili Shao, Edwin H.-M. Sha

## Abstract

Dynamic Voltage Scaling (DVS) is one of the techniques used to obtain energy-saving in real-time DSP systems. In many DSP systems, some tasks contain conditional instructions that have different execution times for different inputs. Due to the uncertainties in execution time of these tasks, this paper models each varied execution time as a probabilistic random variable and solves the *Voltage Assignment with Probability* (VAP) Problem. VAP problem involves finding a voltage level to be used for each node of an data flow graph (DFG) in uniprocessor and multiprocessor DSP systems. This paper proposes two optimal algorithms, one for uniprocessor and one for multiprocessor DSP systems, to minimize the expected 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 for multiprocessor achieves an average improvement of 56.1% on total energy-saving with 0.80 probability satisfying timing constraint.

## Index Terms

Probability, assignment, DVS, real-time, DSP

## I. INTRODUCTION

The increasingly ubiquitous DSP systems pose great challenges different from those faced by general-purpose computers. DSP systems are more application specific and more constrained in terms of power, timing, and other resources. Energy-saving is a critical issue and performance metric in DSP systems design due to wide use of portable devices, especially those powered by batteries [1]–[6]. The systems become more and more complicate and some tasks may not have fixed execution time. Such tasks usually contain conditional instructions and/or operations that could have different execution times for different

M. Qiu, C. Xue, and E. H.-M. Sha are with the Department of Computer Science, University of Texas at Dallas, Richardson, Texas 75083, USA. Email:{mxq012100, cxx016000, edsha}@utdallas.edu. Z. Jia is with School of Computer Science and Technology, Shangdong University, Jinan, Shangdong, China. Email:zhipingj@sdu.edu.cn. Z. Shao is with the Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hongkong. Email:cszshao@comp.polyu.edu.hk.

inputs [7]–[11]. 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 [12]. 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 [13].

Prior design space exploration methods for hardware/software codesign of DSP systems [14]–[17] guarantee no deadline missing by considering worst-case execution time of each task. Many design methods have been developed based on worst-case execution time to meet the timing constraints without any deadline misses. These methods are pessimistic and are suitable for developing systems in a hard real-time environment, where any deadline miss will be catastrophic. However, there are also many soft real-time systems, such as heterogeneous systems, which can tolerate occasional violations of timing constraints. The above pessimistic design methods can't take advantage of this feature and will often lead to over-designed systems that deliver higher performance than necessary at the cost of expensive hardware, higher energy consumption, and other system resources.

There are several papers on the probabilistic timing performance estimation for soft real-time systems design [7]–[12], [18], [19]. The general assumption is that each task's execution time can be described by a discrete probability density function that can be obtained by applying path analysis and system utilization analysis techniques. Hu et al. [8] propose a state-based probability metric to evaluate the overall probabilistic timing performance of the entire task set. However, their evaluation method becomes very time consuming when task has many different execution time variations. Hua et al. [9], [10] propose the concept of *probabilistic design* where they design the system to meet the timing constraints of periodic applications statistically. But their algorithm is not optimal and only suitable to uniprocessor executing tasks according to a fixed order, that is, a simple path. In this paper, we will propose an optimal algorithm for the uniprocessor situation. Also, we will give an optimal algorithm for multiprocessor executing tasks according to an executing order in DAG (Direct Acyclic Graph).

Low power and low energy consumptions are extremely important for real-time DSP systems. Dynamic voltage scaling (DVS) is one of the most effective techniques to reduce energy consumption [20]–[23]. In many microprocessor systems, the supply voltage can be changed by mode-set instructions according to the workload at run-time. With the trend of multiple cores being widely used in DSP systems, it is important to study DVS techniques for multiprocessor DSP systems. This paper focuses on minimizing expected energy consumption with guaranteed probability satisfying timing constraints via DVS for real-time multiprocessor DSP systems.

In this paper, we use probabilistic design space exploration and DVS to avoid over-design systems.

We propose two novel optimal algorithms, one for uniprocessor and one for multiprocessor DSP systems, to minimize the expected value of total energy consumption while satisfying timing constraints with guaranteed probabilities for real-time applications. Our work is related to the work in [9], [10]. In [9], [10], Hua et al. proposed an heuristic algorithm for uniprocessor and the *Data Flow Graph* (DFG) is a simple path. We call the offline part of it as *HUA* algorithm for convenience. 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:

- When there is a uniprocessor, the results of our algorithm, *VAP\_S*, gives the optimal solution and achieves significant energy saving than *HUA* algorithm.
- For the general problem, that is, when there are multiple processors and the DFG is a DAG, our algorithm, *VAP\_M*, gives the optimal solution and achieves significant average energy reduction than *Heu* algorithm.
- Our algorithms not only are optimal, but also provide more choices of smaller expected value of total energy consumption with guaranteed confidence probabilities satisfying timing constraints. In many situations, algorithms *HUA* and *Heu* cannot find a solution, yet ours can find satisfied results.
- Our algorithms are practical and quick. In practice, when the number of multi-parent nodes and multi-child nodes in the given DFG graph is small, and the timing constraint is polynomial to the size of DFG, the running times of these algorithms are very small and our experiments always finished in very short time.

We conduct experiments on a set of benchmarks, and compare our algorithms with *HUA* and *Heu* algorithms. Experiments show that our algorithm for uniprocessor, *VAP\_S*, has an average 58.0% energy-saving improvement with probability 0.8 satisfying timing constraint compared with the greedy algorithm *HUA*. Our algorithm for multiprocessor and DAG, *VAP\_M*, has an average 56.1% energy-saving improvement compared with the results of the heuristic algorithm *Heu*.

The rest of the paper is organized as following: The models and basic concepts are introduced in Section II. In Section III, we give motivational examples. In Section IV, we propose our algorithms. The experimental results are shown in Section V, and the conclusion is shown in Section VI.

## II. MODELS AND CONCEPTS

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

### **System Model:**

We focus on real time applications on single-processor and multiprocessor DSP systems. *Probabilistic Data-Flow Graph* (PDFG) is used to model an embedded systems application. A **PDFG**  $G =$

$\langle V, E, T, R \rangle$  is a *directed acyclic graph* (DAG), where  $\mathbf{V} = \langle v_1, \dots, v_i, \dots, v_N \rangle$  is the set of nodes;  $\mathbf{R} = \langle R_1, \dots, R_j, \dots, R_M \rangle$  is a voltage set; the execution time  $\mathbf{T}_{\mathbf{R}_j}(\mathbf{v})$  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 the multiprocessor system, 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  $N \times C_{ap} \times V_{dd}^2$ , where  $N$  represent the number of computation cycles for a node,  $C_{ap}$  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 [21]–[23]. Let  $T$  represent the execution time of a node and  $C$  stand for energy consumption (since  $E$  has been used to represent edge). 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  $v$ , the supply voltage  $V_{dd}$  and the threshold voltage  $V_{th}$ , its computation time  $T(v)$  and the energy  $C(v)$  for node  $v$  are calculated as follows:

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

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

$$C(v) = N \times C_{ap} \times V_{dd}^2 \quad (3)$$

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. In this paper, we assume that there is no energy or delay penalty associated with voltage switching and the energy leakage is very small.

DVS (Dynamic voltage scaling) is a technique that varies system's operating voltages and clock frequencies based on the computation load to provide desired performance with the minimum energy consumption. It has been demonstrated as one of the most effective low power system design techniques and has been supported by many modern microprocessors. Examples include Transmeta's Crusoe, AMD's K-6, Intel's XScale and Pentium III and IV, and some DSPs developed in Bell Labs [9].

Low power design is of particular interest for the soft real time multimedia systems and we assume that our system has multiple voltages available on the chip such that the system can switch from one level to another. For the energy and delay overhead associated with multiple-voltage systems, we assume that voltage scaling occurs simultaneously without any energy and delay overhead.

**VAP problem:**

For multiple-voltage systems, assume there are maximum  $M$  different voltages in a voltage set  $R = \langle R_1, R_2, \dots, R_M \rangle$ . 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 processor to each node. Define an *assignment*  $A$  to be a function from domain  $V$  to range  $R$ , where  $V$  is the node set and  $R$  is the voltage set. For a node  $v \in V$ ,  $A(v)$  gives selected voltage level of node  $v$ . In a PDFG  $G$ , each varied execution time is modeled as a probabilistic random variable,  $\mathbf{T}_{R_j}(\mathbf{v})$ ,  $1 \leq j \leq M$ , represents the execution times of each node  $v \in V$  when running at voltage level  $R_j$ ;  $\mathbf{P}_{R_j}(\mathbf{v})$ ,  $1 \leq j \leq M$ , represents the corresponding probability function. For each voltage  $R_j$  with respect to node  $v$ , there is a set of pairs  $(p_i, c_i)$ ,  $\sum p_i = 1$ , where  $p_i$  is the probability and  $c_i$  is the energy consumption of each node in PDFG.  $\mathbf{C}_{R_j}(\mathbf{v})$ ,  $1 \leq j \leq M$ , is used to represent the **expected value** of energy consumption of each node  $v \in V$  at voltage  $R_j$ ,  $C_{R_j}(v) = \sum p_i c_i$ , which is a fixed value. Because of the linearity of expected values, we can define the total energy consumption for a system. Given an assignment  $A$  of a PDFG  $G$ , we define the *system expected total energy consumption under assignment  $A$* , denoted as  $\mathbf{C}_A(G)$ , to be the summation of energy consumptions,  $C_{A(v)}(v)$ ,  $v \in V$ , of all nodes, that is,  $C_A(G) = \sum_{v \in V} C_{A(v)}(v)$ . In this paper we call  $C_A(G)$  *total energy consumption* in brief.

For the input PDFG  $G$ , given an assignment  $A$ , assume that  $\mathbf{T}_A(G)$  stands for the *execution time of graph  $G$  under assignment  $A$* .  $\mathbf{T}_A(G)$  can be obtained from the longest path  $p$  in  $G$ . The new variable  $T_A(G) = \max_{v \in p} T_{A(v)}(p)$ , where  $T_{A(v)}(p) = \sum_{v \in p} T_{A(v)}(v)$ , is also a random variable. The *minimum expected total energy consumption  $C$  with confidence probability  $P$  under timing constraint  $L$*  is defined as  $C = \min_A C_A(G)$ , where probability of  $(T_A(G) \leq L) \geq P$ . For each timing constraint  $L$ , our algorithm will output a serial of (Probability, Consumption) pairs  $(P, C)$ .

$T_{R_j}(v)$  is either a discrete random variable or a continuous random variable. We define  $\mathbf{F}(t)$  to be the *cumulative distribution function* (abbreviated as **CDF**) of the random variable  $T_{R_j}(v)$ , where  $F(t) = P(T_{R_j}(v) \leq t)$ . When  $T_{R_j}(v)$  is a discrete random variable, the CDF  $F(t)$  is the sum of all the probabilities associating with the execution times that are less than or equal to  $t$ . If  $T_{R_j}(v)$  is a continuous random variable, then it has a *probability density function (PDF)*. If assume the PDF is  $f$ , then  $F(t) = \int_0^t f(s) ds$ . Function  $F(t)$  is nondecreasing, and  $F(-\infty) = 0$ ,  $F(\infty) = 1$ .

We define the *VAP (voltage assignment with probability) problem* as follows: Given  $M$  different

voltage levels:  $R_1, R_2, \dots, R_M$ , a PDFG  $G = \langle V, E \rangle$  with  $T_{R_j}(v)$ ,  $P_{R_j}(v)$ , and  $C_{R_j}(v)$  for each node  $v \in V$  executed on each voltage  $R_j$ , a timing constraint  $L$  and a confidence probability  $P$ , find the voltage for each node in assignment  $A$  that gives the *minimum expected total energy consumption  $C$  with confidence probability  $P$  under timing constraint  $L$* .

### III. MOTIVATIONAL EXAMPLE

#### A. Multiprocessor Systems

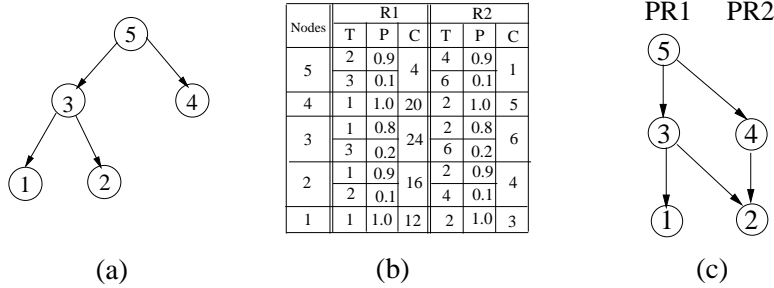


Fig. 1. (a) A given PDFG. (b) The times, probabilities, and energy consumption of its nodes for different voltage levels. (c) The schedule graph of two processors.

First we give an example for multiprocessor embedded systems, which is shown in Figure 2. In this paper, we assume the tasks have already been preprocessed. We already know which task will be processed by which processor, and the scheduling graph is given. For example, the task graph is shown in Figure 1(a). After preprocessing, we get the scheduling graph in Figure 1(c). This is a same graph to the input DFG that is shown in Figure 2(b).

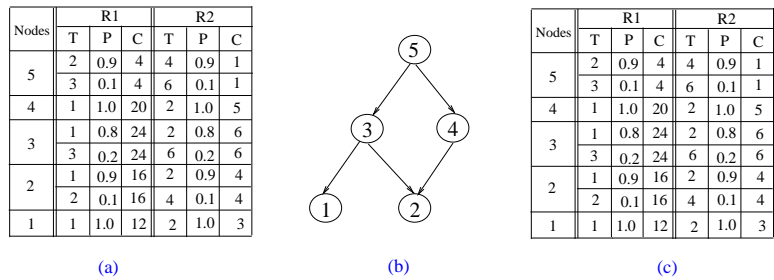


Fig. 2. (a) The times, probabilities, and energy consumption of nodes for different voltage levels. (b) A DAG. (c) The time cumulative distribution functions (CDFs) and energy consumption of its nodes for different voltage levels

In Figure 2, each node has two different voltage levels to choose from, and is executed on them with probabilistic execution times. The input DFG has five nodes. The times, energy consumption, and probabilities of each node is shown in Figure 2(a). Node 5 is a multi-child node, which has two children: 3 and 4. Node 2 is a multi-parent node, and has two parents: 3 and 4. Figure 2(c) shows the time cumulative

distribution functions (CDFs) and energy consumption of each node for different voltage levels. In DSP applications, 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.

The number of computation cycles ( $N$ ) for a task is proportional to the execution time. The energy consumption ( $C$ ) depends on not only the voltage level  $R$ , but also the number of computation cycles  $N$ . We use the expected value of energy consumption ( $Exp(C)$ ) as the energy consumption  $C$  under a certain voltage level  $R$ . 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 [21], [22], [24], 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 ( $R_2$ ) is twice as much as it is under the high voltage ( $R_1$ ); the energy consumption of a node under the high voltage ( $R_1$ ) is four times as much as it is under the low voltage ( $R_2$ ).

T	(P, C)	(P, C)	(P, C)	(P, C)
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
<b>11</b>	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 I

MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR A DAG.

### B. Our Solution

For Figure 2, the minimum total energy consumptions with computed confidence probabilities under the timing constraint are shown in Table I. 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  $T$ . Using our algorithm,  $VAP_M$ , at timing constraint 11, we can get (0.80, 22) pair. Table II shows the assignments of our algorithm. Assignment  $A(v)$  represents the voltage selection of each node  $v$ . 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%. 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	T	Type	Prob.	Consumption
<b>Ours</b>	$A(v)$	5	3	R1	1.00	4
		4	2	R2	1.00	5
		3	4	R2	0.80	6
		2	2	R2	1.00	4
		1	4	R2	1.00	3
	Total		11		0.80	22
<b>HUA</b>	$A(v)$	5	3	R1	1.00	4
		4	2	R1	1.00	5
		3	1	R1	0.80	24
		2	1	R1	1.00	12
		1	2	R1	1.00	16
	Total		6		0.80	61

TABLE II

UNDER TIMING CONSTRAINT 11, THE DIFFERENT ASSIGNMENTS BETWEEN  $VAP_M$  AND  $Heu$ .

Given the requirement of probability, we can get a minimum total energy consumption under every timing constraint. For example, if the required probability is 0.80, then the total energy consumption varies from 61 to 19 at different timing constraints. The energy consumptions under different timing constraints are shown in Table III.

T	6	7, 8	9 - 11	12 - 16
(P, C)	(0.80, 61)	(0.80, 34)	(0.80, 22)	(0.80, 19)

TABLE III

GIVEN AN REQUIREMNT OF PROBABILITY, THE (PROBABILITY, CONSUMPTION) PAIRS UNDER DIFFERENT TIMING CONSTRAINTS.

## IV. THE ALGORITHMS

### A. Definitions and Lemma

To solve the VAP problem, we use dynamic programming method traveling the graph in bottom up fashion. For the easiness of explanation, we will index the nodes based on bottom up sequence. For example, Figure 2 (a) shows a tree indexed by bottom up sequence. The sequence is:  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_5$ . 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 2 (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 DFG  $G$ , and an assignment  $A$ , we first give several definitions as follows:

- $G^i$ : 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 2 (a),  $G^i$  is the tree containing nodes 1, 2, and 3.
- $C_A(G^i)$  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.
- In our algorithm, table  $D_{i,j}$  will be built. Each entry of table  $D_{i,j}$  will store a link list of (Probability, Consumption) pairs sorted by probability in ascending order. Here we define the **(Probability, Consumption) pair**  $(C_{i,j}, P_{i,j})$  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  $\geq P_{i,j}$ .

We introduce the *operator* “ $\oplus$ ” in this paper. For two (Probability, Cost) pairs  $H_1$  and  $H_2$ , if  $H_1$  is  $(P_{i,j}^1, C_{i,j}^1)$ , and  $H_2$  is  $(P_{i,j}^2, C_{i,j}^2)$ , then, after the  $\oplus$  operation between  $H_1$  and  $H_2$ , we get pair  $(P', C')$ , where  $P' = P_{i,j}^1 * P_{i,j}^2$  and  $C' = C_{i,j}^1 + C_{i,j}^2$ . We denote this operation as “ $\mathbf{H}_1 \oplus \mathbf{H}_2$ ”.

In our algorithm,  $D_{i,j}$  is the table in which each entry has a link list that store pair  $(P_{i,j}, C_{i,j})$ . Here,  $i$  represents a node number, and  $j$  represents time. For example, a link 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 link list. We give the redundant-pair removal algorithm as following:

For example, we have a list with pairs  $(0.1, 2) \rightarrow (0.3, 3) \rightarrow (0.5, 3) \rightarrow (0.3, 4)$ , we do the redundant-pair removal as following: First, sort the list according  $P_{i,j}$  in an ascending order. This list becomes to  $(0.1, 2) \rightarrow (0.3, 3) \rightarrow (0.3, 4) \rightarrow (0.5, 3)$ . 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

---

**Algorithm IV.1** redundant-pair removal algorithm
 

---

**Require:** A list of  $(P_{i,j}^k, C_{i,j}^k)$

**Ensure:** A redundant-pair free list

```

1: Sort the list by  $P_{i,j}$  in an ascending order such that  $P_{i,j}^k \leq P_{i,j}^{k+1}$ .
2: From the beginning to the end of the list,
3: for each two neighboring pairs  $(P_{i,j}^k, C_{i,j}^k)$  and  $(P_{i,j}^{k+1}, C_{i,j}^{k+1})$  do
4:   if  $P_{i,j}^k = P_{i,j}^{k+1}$  then
5:     if  $C_{i,j}^k \geq C_{i,j}^{k+1}$  then
6:       cancel the pair  $P_{i,j}^k, C_{i,j}^k$ 
7:     else
8:       cancel the pair  $P_{i,j}^{k+1}, C_{i,j}^{k+1}$ 
9:     end if
10:  else
11:    if  $P_{i,j}^k \geq P_{i,j}^{k+1}$  then
12:      cancel the pair  $(P_{i,j}^{k+1}, C_{i,j}^{k+1})$ 
13:    end if
14:  end if
15: end for

```

---

In summary, we can use the following Lemma to cancel redundant pairs.

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

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

Using Lemma 4.1, we can cancel many redundant-pair  $(P_{i,j}, C_{i,j})$  whenever we find conflicting pairs in a list during a computation. After the  $\oplus$  operation and redundant pair removal, the list of  $(P_{i,j}, C_{i,j})$  has the following properties:

*Lemma 4.2:* For any  $(P_{i,j}^1, C_{i,j}^1)$  and  $(P_{i,j}^2, C_{i,j}^2)$  in the same list:

- 1)  $P_{i,j}^1 \neq P_{i,j}^2$  and  $C_{i,j}^1 \neq C_{i,j}^2$ .
- 2)  $P_{i,j}^1 < P_{i,j}^2$  if and only if  $C_{i,j}^1 < C_{i,j}^2$ .

Since the link list is in an ascending order by probabilities, if  $P_{i,j}^1 < P_{i,j}^2$ , while  $C_{i,j}^1 \geq C_{i,j}^2$ , based on the definitions, we can guarantee with  $P_{i,j}^1$  to find smaller energy consumption  $C_{i,j}^2$ . Hence  $(P_{i,j}^2, C_{i,j}^2)$  has already covered  $(P_{i,j}^1, C_{i,j}^1)$ . We can cancel the pair  $(P_{i,j}^1, C_{i,j}^1)$ . For example, we have two pairs: (0.1, 3) and (0.6, 2). Since (0.6, 2) is better than (0.1, 3), in other words, (0.6, 2) covers (0.1, 3), we can cancel (0.1, 3) and will not lose useful information. We can prove the vice versa is true similarly. When  $P_{i,j}^1 = P_{i,j}^2$ , smaller  $C$  is selected.

In every step in our algorithm, one more node will be included for consideration. The information of

this node is stored in local table  $E_{i,j}$ , which is similar to table  $D_{i,j}$ . A local table only store information, such as probabilities and costs, 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 cost only for node  $v_i$  at time  $j$ , and  $p_{i,j}$  is the corresponding probability. The building procedures of  $E_{i,j}$  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 link list in each entry of  $E_{i,j}$ , insert  $L_{i,j}$  into  $L_{i,j+1}$  while redundant pairs canceled out based on Lemma 4.1. For example, For example, if a node has the following (T: P, C) pairs: (1: 0.9, 10), (3: 0.1, 10) for type  $R_1$ , and (2: 0.7, 4), (4: 0.3, 4) for type  $R_2$ . After sorting and accumulating, we get (1: 0.9, 10), (2: 0.7, 4), (3: 1.0, 10), and (4: 1.0, 4). We obtain Table IV after the insertion.

Similarly, for two link lists  $L_1$  and  $L_2$ , the operation " $L_1 \oplus L_2$ " is implemented as follows: First, implement  $\oplus$  operation on all possible combinations of two pairs from different link lists. Then insert the new pairs into a new link list and remove redundant-pair using Lemma 4.1.

<i>Time</i>	1	2	3	4
$(P_i, C_i)$	(0.9, 10)	(0.7, 4)	(0.7, 4) (1.0, 10)	(1.0, 4)

TABLE IV

AN EXAMPLE OF LOCAL TABLE,  $E_{0,j}$ 

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  $VAP\_SG$ .

---

**Algorithm IV.2** algorithm to get scheduling graph ( $VAP\_SG$ )

---

**Require:** a task graph PDFG

**Ensure:** 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 4.3.
- 

In the graph built in step 1 of  $VAP\_SG$ , If there is a edge from  $v_i$  to  $v_j$ , this means that  $v_i$  is scheduled before  $v_j$  in the same processor or  $v_j$  depends on  $v_i$  in the original PDFG. The new graph is a DAG that represents the order of nodes and dependencies.

*Lemma 4.3:* For two nodes  $v_i$  and  $v_j$ , there is an edge  $e_{ij}$ , if we can find another separate path  $v_i \rightarrow v_j$ , then the edge  $e_{ij}$  can be deleted.

For example, Figure 3 shows the input PDFG. There are two processors  $PR_1$  and  $PR_2$ . The order of nodes are given. In Figure 3(b), there are paths  $C \rightarrow E$  and  $E \rightarrow G$ . Also, there is a separate path

$C \rightarrow G$ . Then we can cancel the path  $C \rightarrow G$ .

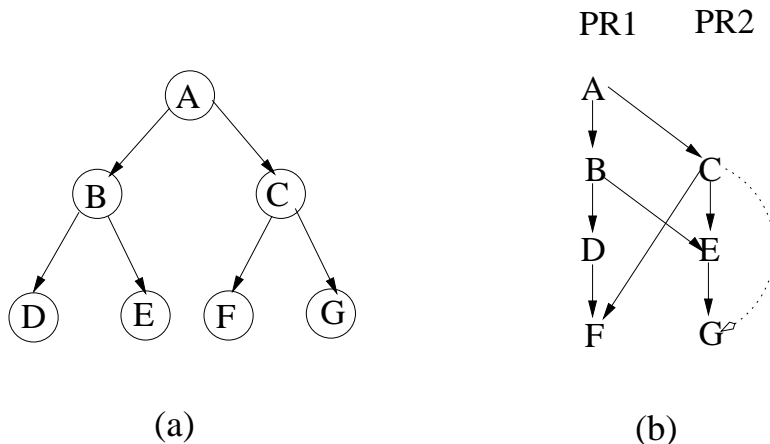


Fig. 3. (a) A task graph. (b) The schedule graph for two processors using list scheduling.

### B. The VAP\_S Algorithm

The Algorithm for uniprocessor system is shown in VAP\_S. 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  $E_{i,j}$  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  $D_{1,j} = E_{1,j}$ . Then using dynamic programming method, build the table  $D_{i,j}$ . For each node  $v_i$  under each time  $j$ , we try all the times  $k$  ( $1 \leq k \leq j$ ) in table  $E_{i,k}$ . We use “ $\oplus$ ” on the two tables  $E_{i,k}$  and  $D_{i-1,j-k}$ . Since  $k + (j - k) = j$ , the total time of nodes from  $v_1$  to  $v_i$  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, Consumption) pairs. The new energy consumption in each pair obtained in table  $D_{i,j}$  is the energy consumption of current node  $v_i$  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  $j$ .

The energy consumption in  $D_{N,j}$  is the minimum total energy consumption with computed confidence probability under timing constraint  $j$ . Given the timing constraint  $L$ , the minimum total energy consumption for the graph  $G$  is the energy consumption in  $D_{N,L}$ . In the following, we show Theorem 4.1 about this.

*Theorem 4.1:* For each pair  $(P_{i,j}, C_{i,j})$  in  $D_{i,j}$  ( $1 \leq i \leq N$ ) obtained by algorithm VAP\_S,  $C_{i,j}$  is the minimum total energy consumption for graph  $G^i$  with confidence probability  $P_{i,j}$  under timing constraint  $j$ .

---

**Algorithm IV.3** optimal algorithm for the VAP problem when there is a single processor (VAP\_S)
 

---

**Require:**  $M$  different levels of voltages, a DAG, and the timing constraint  $L$ .

**Ensure:** An optimal voltage level assignment

- 1) Construct schedule graph using algorithm VAP\_SG.
  - 2) Build a local table  $E_{i,j}$  for each node of PDFG.
  - 3) 1: let  $D_{1,j} = E_{1,j}$ 
    - 2: **for** each node  $v_i, i > 1$  **do**
    - 3:   **for** each time  $j$  **do**
    - 4:     **for** each time  $k$  in  $E_{i,k}$  **do**
    - 5:       **if**  $D_{i-1,j-k} \neq NULL$  **then**
    - 6:           $D_{i,j} = D_{i-1,j-k} \oplus E_{i,k}$
    - 7:       **else**
    - 8:          continue
    - 9:       **end if**
    - 10:     **end for**
    - 11:     insert  $D_{i,j-1}$  to  $D_{i,j}$  and remove redundant pairs using Lemma 4.1 and redundant-pair remove algorithm.
    - 12:   **end for**
    - 13: **end for**
  - 4) return  $D_{N,j}$
- 

*Proof:* By induction. **Basic Step:** When  $i = 1$ , there is only one node and  $D_{1,j} = E_{1,j}$ .  $D_{1,j} = \min_{1 \leq k \leq M} \{c_k(1) \mid j \geq t_k(1)\}$ . Thus, when  $i = 1$ , Theorem 4.1 is true. **Induction Step:** We need to show that for  $i \geq 1$ , if for each pair  $(P_{i,j}, C_{i,j})$  in  $D_{i,j}$ ,  $C_{i,j}$  is the minimum total energy consumption for graph  $G^i$  with confidence probability  $P_{i,j}$  under timing constraint  $j$ , then for each pair  $(P_{i+1,j}, C_{i+1,j})$  in  $D_{i+1,j}$ ,  $C_{i+1,j}$  is the minimum total energy consumption for graph  $G^{i+1}$  with confidence probability  $P_{i+1,j}$  under timing constraint  $j$ . In step 2 of the algorithm, since  $j = k + (j - k)$  for each  $k$  in  $E_{i+1,j}$ , we try all the possibilities to obtain  $j$ . Then we use  $\oplus$  operator to add the energy consumptions of two tables and multiply the probabilities of two tables. Finally, we use Lemma 4.1 to cancel the conflicting (Probability, Consumption) pairs. The new energy consumption in each pair obtained in table  $D_{i+1,j}$  is the energy consumption of current node  $i + 1$  at time  $k$  plus the energy consumption in each pair obtained in  $D_{i,j-k}$ . Since we have used Lemma 4.1 to cancel redundant pairs, the energy consumption of each pair in  $D_{i+1,j}$  is the minimum total energy consumption for graph  $G^{i+1}$  with confidence probability  $P_{i+1,j}$  under timing constraint  $j$ . Thus, Theorem 4.1 is true for any  $i$  ( $1 \leq i \leq N$ ). ■

From Theorem 4.1, we know  $D_{N,L}$  records the minimum total energy consumption of the whole path with corresponding confidence probabilities under the timing constraint  $L$ . We can record the corresponding FU type assignment of each node when computing the minimum total energy consumption in step 2 in

the algorithm *VAP\_S*. Using these information, we can get an optimal assignment by tracing how to reach  $D_{N,L}$ .

It takes  $O(M * K)$  to compute one value of  $D_{i,j}$ , where  $M$  is the maximum number of FU types, and  $K$  is the maximum number of execution time variations for each node. Thus, the complexity of the algorithm *VAP\_S* is  $O(|V| * L * M * K)$ , where  $|V|$  is the number of nodes and  $L$  is the given timing constraint. Usually, the execution time of each node is upper bounded by a constant. So  $L$  equals  $O(|V|^c)$  ( $c$  is a constant). In this case, *VAP\_S* is polynomial.

### C. The *VAP\_M* Algorithm

In this subsection, we give a heuristic algorithm *Heu* first, then we propose our novel and optimal algorithm, *VAP\_M*, for multiprocessor DSP systems. We will compare them in the experiments section.

#### The *Heu* Algorithm

We first design an heuristic algorithm for multiprocessor systems according to the *HUA* algorithm in [9], [10], we call this algorithm as *Heu*. The DFG now is a DAG and no longer limited to a simple path. The authors of [9], [10] 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 [9], [10] 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  $v_i$ ,  $l_i$  is the time variation of node  $v_i$ ,  $S_i$  is the scaled time slot for node  $v_i$ , and  $t_{ij}$  is the  $j$  time variation of node  $v_i$ .

Let  $E_{ij}(S_i)$  be the minimum energy to complete the workload  $t_{ij}$  required by vertex  $v_i$  in time  $S_i$ . On an ideal variable voltage processor,  $E_{ij}(S_i)$  is the energy consumed by running at voltage  $R_{ideal}$  throughout the entire assigned slot  $S_i$ , where  $R_{ideal}$  is the voltage level that enables the processor to accumulate the workload  $t_{ij}$  at the end of the slot. On a multiple voltage processor with only a finite set of voltage levels  $R_1 < R_2 < \dots$ ,  $E_{ij}(S_i)$  is the energy consumed by running at  $R_k$  for a certain amount of time and then switching to the next higher level  $R_{k+1}$  to complete  $t_{ij}$ , where  $R_k < R_{ideal} < R_{k+1}$  and the switching point can be conveniently calculated from  $S_i$  and  $t_{ij}$  [9], [10].

We propose our algorithm, *VAP\_M*, for multiprocessor DSP systems, which shown as follows. 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.

#### The *VAP\_M* Algorithm

**Require:**  $M$  different voltage levels, a DAG, and the timing constraint  $L$ .

---

**Algorithm IV.4** Heuristic algorithm for the VAP problem when there are multiple processors and the DFG is DAG (*Heu*)

---

**Require:**  $M$  different levels of voltages, a DAG, and the timing constraint  $L$ .

**Ensure:** a voltage assignment to minimize energy consumption with a guaranteed probability  $\theta$  satisfying  $L$

```

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

```

---

**Ensure:** An optimal voltage assignment for the DAG

- 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  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_N$ , 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  $v_{i_1}, v_{i_2}, \dots, v_{i_R}$  are all child nodes of

node  $v_i$  and  $R$  is the number of child nodes of node  $v_i$ , then

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

6) For  $D_{i_1,j} \oplus D_{i_2,j}$ ,  $G'$  is 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  $v_{i_2}$ . If a node is a common node, then use a selection function to choose the type of a node.

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

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

8) 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.

9) Then use the Lemma 4.1 to remove redundant pairs. Finally get  $D_{N,j}$ .

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 DFG is a DAG. In the following, we give the Theorem 4.2 and Theorem 4.3 about this.

*Theorem 4.2:* In each possible fixed assignment, for each pair  $(P_{i,j}, C_{i,j})$  in  $D_{i,j}$  ( $1 \leq i \leq N$ ) obtained by algorithm  $VAP_M$ ,  $C_{i,j}$  is the minimum total energy consumption for the graph  $G^i$  with confidence probability  $P_{i,j}$  under timing constraint  $j$ .

*Proof:* By induction. **Basic Step:** When  $i = 1$ , There is only one node and  $D_{1,j} = E_{1,j}$ . Thus, when  $i = 1$ , Theorem 4.2 is true. **Induction Step:** We need to show that for  $i \geq 1$ , if for each pair  $(P_{i,j}, C_{i,j})$  in  $D_{i,j}$ ,  $C_{i,j}$  is the minimum total energy consumption of the graph  $G^i$ , then for each pair  $(P_{i+1,j}, C_{i+1,j})$  in  $D_{i+1,j}$ ,  $C_{i+1,j}$  is the total energy consumption of the graph  $G^{i+1}$  with confidence probability  $P_{i+1,j}$  under timing constraint  $j$ . According to the bottom up approach (for top down approach, just reverse the sequence), the execution of  $D_{i,j}$  for each child node of  $v_{i+1}$  has been finished before executing  $D_{i+1,j}$ . From equation (4),  $D'_{i+1,j}$  gets the summation of the minimum total energy consumption of all child nodes of  $v_{i+1}$  because they can be executed simultaneously within time  $j$ . We avoid the repeat counting of the common nodes. Hence, each nodes in the graph rooted by node  $v_{i+1}$  was counted only once. From equation (5), the minimum total energy consumption is selected from all possible energy consumptions caused by adding  $v_{i+1}$  into the sub-graph rooted on  $v_{i+1}$ . So for each pair  $(P_{i+1,j}, C_{i+1,j})$  in  $D_{i+1,j}$ ,

$C_{i+1,j}$  is the total energy consumption of the graph  $G^{i+1}$  with confidence probability  $P_{i+1,j}$  under timing constraint  $j$ . Therefore, Theorem 4.2 is true for any  $i$  ( $1 \leq i \leq N$ ). ■

*Theorem 4.3:* For each pair  $(P_{i,j}, C_{i,j})$  in  $D_{N,j}$  ( $1 \leq j \leq L$ ) obtained by algorithm *VAP\_M*,  $C_{i,j}$  is the minimum total energy consumption for the given DAG  $G$  with confidence probability  $P_{i,j}$  under timing constraint  $j$ .

*Proof:* According to Theorem 4.2, in each possible fixed assignment, for each pair  $(P_{i,j}, C_{i,j})$  in  $D_{i,j}$  we obtained,  $C_{i+1,j}$  is the total energy consumption of the graph  $G^{i+1}$  with confidence probability  $P_{i+1,j}$  under timing constraint  $j$ . In algorithm *VAP\_M*, we try all the possible fixed assignments, combine them together into a new row  $D_{N,j}$  in dynamic table, and remove redundant pairs using the Lemma 4.1. Hence, for each pair  $(P_{i,j}, C_{i,j})$  in  $D_{N,j}$  ( $1 \leq j \leq L$ ) obtained by algorithm *VAP\_M*,  $C_{i,j}$  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  $O(K^{t+1} * |V|^2 * L * M)$ , where  $t$  is the total number of nodes with multi-parent (or multi-child) in bottom up approach (or top down approach),  $|V|$  is the number of nodes,  $L$  is the given timing constraint,  $M$  is the maximum number of FU types for each node, and  $K$  is the maximum number of execution time variation for each node. Algorithm *VAP\_M* is exponential, hence it can not be applied to a graph with large amounts of multi-parent and multi-child nodes.

## V. EXPERIMENTS

This section presents the experimental results of our algorithms. We conduct experiments on a set of benchmarks including 4-stage lattice filter, 8-stage lattice filter, volterra filter, differential equation solver, RLS-languerre lattice filter, and elliptic filter. Among them, the PDFG for first three filters are trees and those for the others are DAGs.

Three different voltage levels,  $R_1$ ,  $R_2$ , and  $R_3$ , are used in the system, in which a processor under  $R_1$  is the quickest with the highest energy consumption and a processor under  $R_3$  is the slowest with the lowest energy consumption. The execution times, probabilities, and energy consumptions for each node are randomly assigned. 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.

We compare our uniprocessor algorithm with the heuristic algorithm *HUA* in [9], [10] on all of the six benchmarks. There is only one processor: *PR1*. These experiments are finished in less than one second. The experimental results on volterra filter, 4-stage lattice filter, and 8-stage lattice filter are shown in Table V-VII. In each table, column “TC” represents the given timing constraint, “HUA” represents the

Volterra Filter (27 nodes)									
TC	0.8			0.9			1.0		
	HUA	Ours	%	HUA	Ours	%	HUA	Ours	%
103	708	701	1.0	×	×		×	×	
108	708	647	8.6	708	701	1.0	×	×	
110	708	623	12.1	708	671	5.2	708	701	1.0
150	708	310	56.2	708	338	52.3	708	347	51.0
200	708	186	73.7	708	204	71.2	708	206	70.9
206	177	175	1.2	708	196	72.3	708	198	72.0
216	177	162	8.5	177	175	1.2	708	182	74.3
220	177	156	11.9	177	172	2.8	177	175	1.2
250	177	118	33.4	177	130	26.6	177	132	25.6
300	177	73	59.8	177	76	57.1	177	82	54.7
350	177	45	74.6	177	45	74.6	177	45	74.6
Ave. Redu.(%)			58.6			61.7			64.8

TABLE V

THE MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR VOLTERA FILTER.

heuristic algorithm *HUA* in [9], [10], and “Ours” represents our optimal algorithm *VAP\_S*. The minimum total energy consumption obtained from different algorithms: *VAP\_S* and the optimal *HUA* [9], [10], 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 *HUA* [9], [10]. The average percentage reduction is shown in the last row “Ave. Redu(%)” of all Tables V-VII, which is computed by averaging energy-savings at all different timing constraints. The entry with “×” means no solution available. For the timing constraint below certain value, we can not find a solution using *HUA* algorithm, while we can find solutions using our algorithm. For example, in Table V, under timing constraint 103, we can not find solution for probability 0.9 using *HUA* algorithm, but we can find solution 705 using our optimal algorithm.

From the Table V-VII, we found in many situations, algorithm *VAP\_S* has significant energy consumption reduction than algorithm *HUA* [9], [10]. For example, in Table V, under the timing constraint 200, for probability 0.8, the entry under “HUA” is 708, which is the minimum total energy consumption using algorithm *HUA*. The entry under “Ours” is 186, which means by using *VAP\_S* algorithm, we can achieve minimum total energy consumption 186 with confidence probability 0.8 under timing constraint 200. The energy reduction is 73.7%.

The experimental results show that our algorithm can greatly reduce the total energy consumption while

4-stage Lattice IIR Filter (26 nodes)									
TC	0.8			0.9			1.0		
	HUA	Ours	%	HUA	Ours	%	HUA	Ours	%
96	684	676	1.0	×	×		×	×	
101	684	616	9.8	684	676	1.0	×	×	
107	684	592	13.5	684	671	6.7	684	676	1.0
150	684	309	54.8	684	355	48.2	684	369	45.8
180	684	212	69.0	684	227	66.8	684	238	65.2
192	171	168	1.8	684	186	62.9	684	190	62.2
202	171	154	10.0	171	168	1.8	684	174	64.6
214	171	150	12.1	171	152	11.8	171	168	1.8
240	171	108	26.9	171	120	29.9	171	124	27.5
280	171	64	62.6	171	68	60.2	171	74	56.7
320	171	41	76.0	171	41	76.0	171	41	76.0
Ave. Redu.(%)			55.9			59.7			62.4

TABLE VI

THE MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR 4-STAGE LATTICE FILTER.

have a guaranteed confidence probability. On average, algorithm *VAP\_S* gives an energy consumption reduction of 58.0% with 0.8 confidence probability satisfying timing constraints, and energy consumption reductions of 61.4% and 64.1% with 0.9 and 1.0 confidence probabilities satisfying timing constraints, respectively. The experiments using *VAP\_S* on these benchmarks are finished within several minutes.

For the multiprocessor systems, we compare our *VAP\_M* algorithm with the *Heu* algorithm. We also conduct experiments on all of the six benchmarks. There are two processors: *PR1* and *PR2*. The experimental results for different equation solver, RLS-Laguerre filter, and elliptic filter are shown in Table VIII-X. In each table, column “TC” represents the given timing constraint, “Heu” represents the heuristic algorithm *Heu* in, and “Ours” represents our optimal algorithm *VAP\_M*. The minimum total energy consumption obtained from different algorithms: *VAP\_M* and the optimal *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 all Tables VIII-X. The entry with “×” means no solution available. Under timing constraint 63 in Table VIII, there is no solution for probability 0.9 using algorithm *Heu*. However, we can find solution 426 with 0.9 probability that guarantees the total execution time of the DFG are less than or equal to the timing constraint 63.

8-stage Lattice IIR Filter (42 nodes)									
TC	0.8			0.9			1.0		
	HUA	Ours	%	HUA	Ours	%	HUA	Ours	%
166	1100	1090	0.9	×	×		×	×	
172	1100	1003	8.2	1100	1090	0.9	×	×	
175	1100	960	12.8	1100	1032	6.2	1100	1090	0.9
200	1100	548	50.2	1100	597	45.7	1100	626	43.1
180	1100	404	63.2	1100	421	61.7	1100	442	59.8
332	275	272	1.1	1100	305	72.3	1100	308	72.0
344	275	250	9.0	275	272	1.1	1100	282	74.4
350	275	239	13.2	275	242	12.1	275	272	1.1
400	275	183	33.5	275	200	27.3	275	205	25.5
420	275	112	59.3	275	118	57.1	275	127	53.9
450	275	68	75.3	275	68	75.3	275	68	75.3
Ave. Redu.(%)			59.5			62.8			65.2

TABLE VII

THE MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR 8-STAGE LATTICE FILTER.

From the Table VIII-X, we found in many situations, algorithm *VAP\_M* has significant energy consumption reduction than algorithm *Heu*. For example, in Table VIII, under the timing constraint 80, for probability 0.8, the entry under “Heu” is 432, which is the minimum total energy consumption using algorithm *Heu*. The entry under “Ours” is 162, which means by using *VAP\_M* algorithm, we can achieve minimum total energy consumption 186 with confidence probability 0.8 under timing constraint 80. The energy reduction is 72.6%, which is very impressive.

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 consumption reduction of 56.1% with 0.8 confidence probability satisfying timing constraints, and energy consumption reductions of 59.3% and 61.7% with 0.9 and 1.0 confidence probabilities satisfying timing constraints, respectively. The experiments using *VAP\_M* on these benchmarks are finished within several minutes.

We also give the impact of guaranteed probability on energy consumption in Figure 4. The experiment was implemented on Volterra Filter. The timing constraints are fixed. We selected three timing constraints: 150, 206, and 250. Figure 4 shows the energy consumption values from algorithm *HUA* and *VAP\_S*. The three lines with square box stands for the results from algorithm *HUA*, and the three lines with “+” stands for the results from algorithm *VAP\_S*. We can find that the energy consumption from algorithm *VAP\_S* is significant lower than the results from *HUA*. In curve *HUA206*, there is a jump or energy consumption from 708 to 177 at probability 0.9. The reason is: at this time, we can switch the voltage level from high

Different Equation Solver (11 nodes)									
TC	0.8			0.9			1.0		
	Heu	Ours	%	Heu	Ours	%	Heu	Ours	%
63	432	426	1.5	×	×		×	×	
66	432	394	8.8	432	426	1.5	×	×	
68	432	380	12.1	432	397	8.2	432	426	1.5
60	432	196	54.6	432	208	51.8	432	214	50.4
80	432	162	72.6	432	126	70.8	432	133	69.2
126	108	104	3.7	432	118	62.7	432	123	71.6
132	108	98	9.2	108	104	3.7	432	112	74.1
138	108	95	12.5	108	103	4.2	108	104	3.7
150	108	72	33.4	108	80	26.0	108	84	22.3
180	108	46	57.4	108	48	55.6	108	58	46.3
200	108	32	70.4	108	32	70.4	108	32	70.4
Ave. Redu.(%)			52.6			55.3			57.4

TABLE VIII

THE MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR DIFFERENT EQUATION SOLVER.

level  $R1$  to  $R2$ . While at the probability below 0.9, we cannot switch according to algorithm  $HUA$ . The results from algorithm  $VAP\_S$  can freely choose voltage level for each node and hence achieve impressive energy saving.

Figure 5 depicts timing constraints' impact on energy consumption. We fixed the guaranteed probability as 0.80. Figure 4 shows the energy consumption values from algorithm  $HUA$  and  $VAP\_S$ . The solid line with "+" stands for the results from algorithm  $VAP\_S$ . It decreases quickly from 701 to 45 as the timing constraint increases from 103 to 350. The dashed line with square box stands for the results from algorithm  $HUA$ . It keeps 708 unchanged from timing constraint 103 to 205; Then at 206, it drops sharply to 177; After that it keeps 177 from timing constraint 206 to 350. The reason of the drop from 708 to 177 is because the voltage level switched from high voltage  $R1$  to low voltage  $R2$  for all the nodes. Before timing constraint 206, we can switch the voltage from  $R1$  to low voltage  $R2$  for all nodes since we cannot find way to guarantee the probability of satisfying timing constraint to reach 0.80 under low voltage  $R2$ . The dot dashed line with "." is the result curve of energy saving percentage of algorithm  $VAP\_S$  to  $HUA$ . The percentage first increases gradually; Then at the middle, it dropped to nearly 0; After that, it increases gradually again. We can find that the energy consumption from algorithm  $VAP\_S$  is significant lower than the results from  $HUA$ . The larger the timing constraints, the lower the energy needed. This matches our expectation.

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 IX

THE MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR RLS-LAGUERRE FILTER.

## VI. CONCLUSION

This paper proposed two optimal algorithms to minimize energy in real-time DSP 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 algorithms, *VAP-S* and *VAP-M*, to give the optimal solutions for uniprocessor or multiprocessor DSP systems, respectively. Experimental results show that our proposed algorithms achieved significant improvement on energy consumption savings than previous work. Our approach has great potential for further exploration.

## VII. ACKNOWLEDGEMENT

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.

## REFERENCES

- [1] I. Foster, *Designing and Building Parallel Program: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley, 1994.
- [2] B. P. Lester, *The Art of Parallel Programming*, Englewood Cliffs, N.J.: Prentice Hall, 1993.
- [3] M. E. Wolfe, *High Performance Compilers for Parallel Computing*, Redwood City, Calif.: Addison-Wesley, 1996.

Elliptic Filter (34 nodes)									
TC	0.8			0.9			1.0		
	Heu	Ours	%	Heu	Ours	%	Heu	Ours	%
208	1424	1406	2.3	×	×		×	×	
218	1424	1284	9.8	1424	1406	2.3	×	×	
220	1424	1236	13.2	1424	1307	8.2	1424	1406	2.3
300	1424	595	58.2	1424	618	56.6	1424	678	52.4
350	1424	435	69.4	1424	452	68.2	1424	481	66.2
416	356	348	2.3	1424	430	69.0	1424	438	69.2
436	356	316	11.2	356	348	2.3	1424	424	70.2
440	356	303	14.8	356	335	6.8	356	348	2.3
500	356	236	33.7	356	262	26.4	356	266	25.3
550	356	148	58.4	356	154	56.7	356	164	54.0
600	356	98	72.5	356	98	72.5	356	98	72.5
Ave. Redu.(%)			55.5			58.7			61.3

TABLE X

THE MINIMUM EXPECTED TOTAL ENERGY CONSUMPTION WITH COMPUTED CONFIDENCE PROBABILITIES UNDER VARIOUS TIMING CONSTRAINTS FOR ELLIPTIC FILTER.

- [4] P. Gl Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of asic's," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, pp. 661–679, Jun. 1989.
- [5] L.-F. Chao and Edwin H.-M. Sha, "Static scheduling for synthesis of dsp algorithms on various models," *Journal of VLSI Signal Processing Systems*, vol. 10, pp. 207–223, 1995.
- [6] L.-F. Chao and Edwin H.-M. Sha, "Scheduling data-fbw graphs via retiming and unfolding," *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, pp. 1259–1267, Dec. 1997.
- [7] S. Tongsima, Edwin H.-M. Sha, C. Chantrapornchai, D. Surma, and N. Passose, "Probabilistic loop scheduling for applications with uncertain execution time," *IEEE Trans. on Computers*, vol. 49, pp. 65–80, Jan. 2000.
- [8] T. Zhou, X. Hu, and Edwin H.-M. Sha, "Estimating probabilistic timing performance for real-time embedded systems," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol. 9, no. 6, pp. 833–844, Dec. 2001.
- [9] Shaoxiong Hua, Gang Qu, and Shuvra S. Bhattacharyya, "Exploring the probabilistic design space of multimedia systems," in *IEEE International Workshop on Rapid System Prototyping*, 2003, pp. 233–240.
- [10] Shaoxiong Hua, Gang Qu, and Shuvra S. Bhattacharyya, "Energy reduction techniques for multimedia applications with tolerance to deadline misses," in *ACM/IEEE Design Automation Conference (DAC)*, 2003, pp. 131–136.
- [11] Shaoxiong Hua and Gang Qu, "Approaching the maximum energy saving on embedded systems with multiple voltages," in *International Conference on Computer Aid Design (ICCAD)*, 2003, pp. 26–29.
- [12] T. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. Wu, and J. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *Proceedings of Real-Time Technology and Applications Symposium*, 1995, pp. 164 – 173.
- [13] J. Bolot and A. Vega-Garcia, "Control mechanisms for packet audio in the internet," in *Proceedings of IEEE Infocom*, 1996.
- [14] C.-Y. Wang and K. K. Parhi, "High-level synthesis using concurrent transformations, scheduling, and allocation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 274–295, Mar. 1995.
- [15] K. Ito, L. Lucke, and K. Parhi, "Ihp-based cost-optimal dsp synthesis with module selection and data format conversion," *IEEE Trans. on VLSI Systems*, vol. 6, pp. 582–594, Dec. 1998.

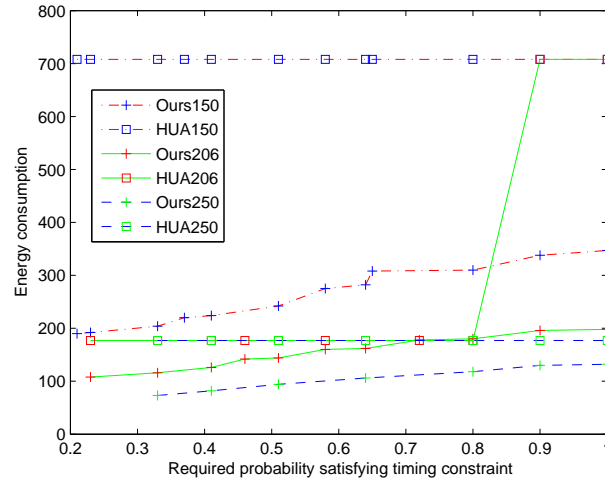


Fig. 4. Guaranteed probability's impact on energy consumption.

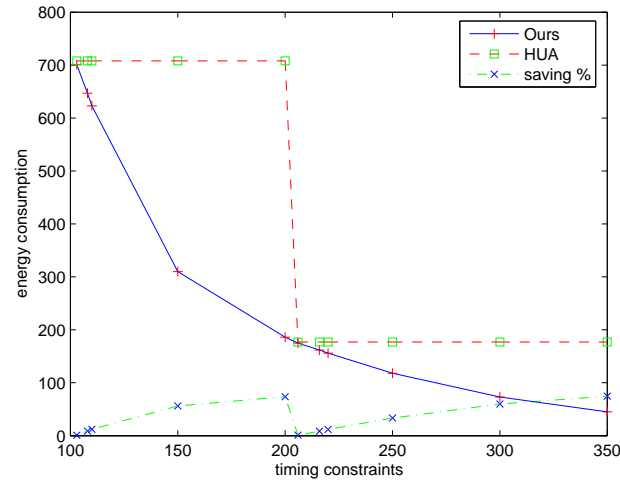


Fig. 5. Timing constraints' impact on energy consumption.

- [16] K. Ito and K. Parhi, "Register minimization in cost-optimal synthesis of dsp architecture," in *Proc. of the IEEE VLSI Signal Processing Workshop*, Oct. 1995.
- [17] Z. Shao, Q. Zhuge, C. Xue, and Edwin H.-M. Sha, "Efficient assignment and scheduling for heterogeneous dsp systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, pp. 516–525, Jun. 2005.
- [18] M. Qiu, M. Liu, C. Xue, Z. Shao, Q. Zhuge, and E. H.-M. Sha, "Optimal assignment with guaranteed confidence probability for trees on heterogeneous dsp systems," in *Proceedings The 17th IASTED International Conference on Parallel and Distributed Computing Systems (PDCS 2005)*, Phoenix, Arizona, 14-16 Nov. 2005.
- [19] A. Kalavade and P. Moghe, "A tool for performance estimation of networked embedded end-systems," in *Proceedings of Design Automation Conference*, Jun. 1998, pp. 257 – 262.
- [20] 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 *Proceedings the 11th International Conference on Parallel and Distributed Systems (ICPADS'05) Volume II, The IEEE/IFIP International Workshop on Parallel and Distributed Embedded Systems (PDES 2005)*, Fukuoka, Japan, 20-22 Jul. 2005, pp. 2 – 6.

- [21] D. Shin, J. Kim, and S. Lee, "Low-energy intra-task voltage scheduling using static timing analysis," in *DAC*, 2001, pp. 438–443.
- [22] Y. Zhang, X. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *DAC*, 2002, pp. 183–188.
- [23] 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.
- [24] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processor," in *ISLPED*, 1998, pp. 197–202.