Voltage Assignment and Loop Scheduling for Energy Minimization while Satisfying Timing Constrtaint with Guaranteed Probability *

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

Abstract

Low energy consumption is an important problem in real-time embedded systems and loop is the most energy consuming part in most cases. Due to the uncertainties in execution time of some tasks, this paper models each varied execution time as a probabilistic random variable. We use rotation scheduling and DVS (Dynamic Voltage Scaling) to minimize the expected total energy consumption while satisfying the timing constraint with a guaranteed confidence probability. Our approach can handle loops efficiently. In addition, it is suitable to both soft and hard real-time systems. And even for hard real-time, we have good results.

1 Introduction

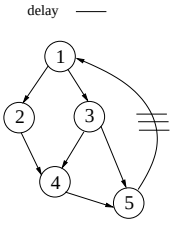
Energy consumption has become a primary concern in today's real-time embedded systems. In DSP systems, some tasks may not have fixed execution time. Such tasks usually contain conditional instructions and/or operations that could have different execution times for different inputs. It is possible to obtain the execution time distribution for each task by sampling or profiling. Prior design space exploration methods for hardware/software codesign of embedded systems [3] guarantee no deadline missing by considering worst-case execution time of each task. These methods are pessimistic and will often lead to over-designed systems with high cost. In this paper, we use probabilistic approach and loop scheduling to avoid over-design systems. We propose a novel optimal algorithm to minimize expected value of total energy consumption while satisfying timing constraints with guaranteed probabilities for real-time applications.

Dynamic voltage scaling (DVS) is one of the most effective techniques to reduce energy consumption. Many researches have been done on DVS for real-time applications in recent years [4]. Zhang et. al. [4] proposed an ILP (Integer Linear Programming) model to solve DVS on multiple processor systems. We design new rotation scheduling algorithms for real-time applications that produce schedules consuming minimal energy. In our algorithms, we use rotation scheduling [1] to get schedules for loop applications. The schedule length will be reduced after rotation. Then, we use DVS to assign voltages to computations individually in order to decrease the voltages of processors as much as possible within the timing constraint. The experimental data show that our algorithms can get better results on energy saving than the previous work.

*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.

†M. Qiu, C. Xue, Q. Zhuge, M. Liu, and E. H. M. Sha are with the Dept. of CS, Univ. of Texas at Dallas, Richardson, Texas 75083, USA. Email:{mxq012100, cxx016000, qfzhuge, mxl024100, edsha}@utdallas.edu. Z. Shao is with the Dept. of Computing, Hong Kong Polytechnic Univ., Hung Hom, Kowloon, Hongkong. Email:cszshao@comp.polyu.edu.hk.

2 Motivational Examples



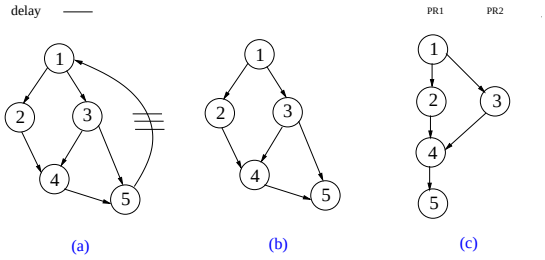
(a)

Nodes	R1			R2			R3		
	T	P	C	T	P	C	T	P	C
1	1	0.8	9	3	0.9	3	5	0.7	1
	2	0.2		4	0.1		6	0.3	
2	1	1.0	8	2	1.0	6	3	1.0	2
3	1	1.0	8	2	1.0	6	3	1.0	2
4	1	0.7	8	2	0.9	4	5	0.9	2
	3	0.3		4	0.1		6	0.1	
5	1	0.9	10	3	0.8	3	5	0.8	1
	2	0.1		4	0.2		6	0.2	

(b)

Figure 1. (a) A PDFG. (b) Parameter table

Assume an input PDFG (*Probability Data Flow Graph*) shown in Figure 1(a). Each node can select one of the three different voltages: R_1, R_2, and R_3. The execution times (T), corresponding probabilities (P), and expected energy consumption (C) of each node under different voltage levels are shown in Figure 1(b). The input PDFG has five nodes. Node 1 is a multi-child node, which has two children: 2 and 3. Node 5 is a multi-parent node, and has two parents: 3 and 4. The execution time T of each node is modeled as a random variable. For example, When choosing R_1, node 1 will be finished in 1 time unit with probability 0.8 and will be finished in 2 time units with probability 0.2.



(a) (b) (c)

Figure 2. (a) Original PDFG. (b) The static schedule. (c) The schedule graph

For initial schedule graph in Figure 2 (c), the minimum total energy consumptions with computed confidence prob-

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

Table 1. Minimum C satisfying T with P.

		Node id	T	M	Prob.	Consum.
Ass_1	$A(v)$	1	2	R_1	1.00	9
		2	3	R_3	1.00	2
		3	3	R_3	1.00	2
		4	2	R_2	0.90	4
		5	4	R_2	1.00	3
	Total		11		0.90	20
Ass_2	$A(v)$	1	2	R_1	1.00	9
		2	1	R_1	1.00	8
		3	1	R_1	1.00	8
		4	4	R_2	1.00	4
		5	4	R_2	1.00	3
	Total		11		1.00	32

Table 2. The assignments with $T = 11$.

abilities under the timing constraints are shown in Table 1. The results are generated by our algorithm, *VAP_M*, which is a sub-algorithm of our *VASP_RS* algorithm. The entries with probability that is equal to 1 (see the entries in boldface) actually give the results to the hard real-time problem which shows the worst-case scenario. For each row of the table, the C in each (P, C) pair gives the minimum total energy consumption with confidence probability P under timing constraint j. For example, using our algorithm, at timing constraint 11, we can get (0.90, 20) pair. The assignments are shown as "Ass_1" in Table 2. Assignment $A(v)$ represents the voltage selection of each node v. Hence, we find the way to achieve minimum total energy consumption 20 with probability 0.90 satisfying timing constraint 11. While using the ILP and heuristic algorithm in [3], the total energy consumption obtained is 32. The assignments are shown as "Ass_2" in Table 2.

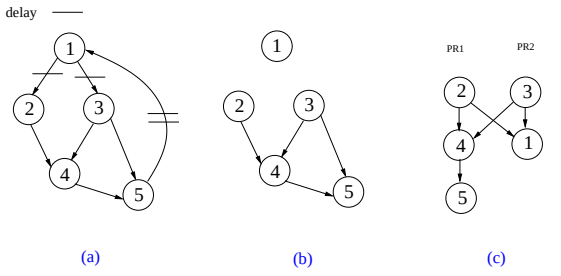


Figure 3. (a) The PDFG after retiming (b) The static schedule. (c) New schedule

For the new schedule graph shown in Figure 3(c), we get (0.90, 10) and (1.00, 12) pairs at timing constraint 11. For (0.90, 10) pair, node 5's type was changed to R_3, then the T was changed from 4 to 6, and energy consumption change from 3 to 1. Hence the total execution time is 11, and the total energy consumption is 10. So the improvement of energy saving is 50.0% while the probability is still 90%. For (1.00. 12) pair, the execution times of nodes 2, 3 were changed from 1 to be 3 and node 1's was changed from 2 to 6. The total energy consumptions were changed to 12, and the total execution time is still 11. Hence compared with original 32, the improvement of total energy saving is 62.5%. If we consider the energy consumptions of switch activity, we can get more practical results. For example, after rotation once, node 1 has changed from processor P1 to P2. Assume the energy consumption of this switch is 1, then the final total energy consumption is 13. Then the energy saving is 59.4% compared with previous scheduling and assignment.

3 Models and Concepts

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, $\mathbf{C_{ap}}$ is the effective switched capacitance, and $\mathbf{V_{dd}}$ 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). The cycle period time $\mathbf{T_c}$ 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. We use the energy model in [4]. Given the number of cycles N of node v, the supply voltage V_{dd} and the threshold voltage V_{th}, the execution time $T(v)$ is $N \times T_c$. The energy consumption $C(v)$ is $N \times C_{ap} \times V_{dd}^2$.

System Model: PDFG:: *Probabilistic Data-Flow Graph* (PDFG) is used to model a embedded systems application. A ***PDFG G*** $= \langle V, E, T, R \rangle$ is a *directed acyclic graph* (DAG), where $\mathbf{V} = \langle v_1, \cdots, v_i, \cdots, v_N \rangle$ is the set of nodes; $\mathbf{R} = \langle R_1, \cdots, R_j, \cdots, R_M \rangle$ is a voltage set; the execution time $\mathbf{T_{R_j}}(\mathbf{v})$ is a random variable; $\mathbf{E} \subseteq V \times V$ is the edge set that defines the precedence relations among nodes in V. **Cyclic PDFG:** A cyclic DFG G=⟨V, E, d, T, R ⟩is a node-weighted and edge-weighted directed graph, where d(e) is a function to represent the number of delays for any edge e ∈ E, The edge without delay represents the intra-iteration data dependency; the edge with delays represents the inter-iteration data dependency and the number of delays represents the number of iterations involved. **Static Schedules:** From the PDFG of an application, we can obtain a static schedule. A static schedule of a cyclic PDFG is a repeated pattern of an execution of the corresponding loop. The DAG is obtained by removing all edges with delays in the PDFG.

Retiming: Retiming [2] is an optimal scheduling technique for cyclic DFGs considering inter-iteration dependencies. It can be used to optimize the cycle period of a cyclic DFG by evenly distributing the delays. Retiming generates the optimal schedule for a cyclic DFG when there is no resource constraint. Given a cyclic DFG G=⟨V,

E, d, T, R⟩, retiming r of G is a function from V to integers. For a node $u \in V$, the value of r(u) is the number of delays drawn from each of incoming edges of node u and pushed to all of the outgoing edges. Let G_r=⟨V, E, d_r, T, R⟩ denote the retimed graph of G with retiming r, then $d_r(e) = d(e) + r(u) - r(v)$ for every edge $e(u \rightarrow v) \in E$. **Rotation Scheduling:** Rotation Scheduling [1] is a scheduling technique used to optimize a loop schedule with resource constraints. It transforms a schedule to a more compact one iteratively in a PDFG. In most cases, the minimal schedule length can be obtained in polynomial time by rotation scheduling.

Definitions: Define the ***VASP (voltage assignment and scheduling with probability)*** problem as follows: Given M different voltage levels: R_1,R_2,$\cdots$,R_M, a PDFG $G = \langle V, E, T, R\rangle$ 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.

4 The Algorithms

To solve the VASP problem, we use dynamic programming method traveling the graph in a bottom up fashion. For the easiness of explanation, we will index the nodes based on bottom up sequence. Here we define the **(Probability, Consumption) pair ($P_{i,j}$, $C_{i,j}$)** as follows: $C_{i,j}$ is the minimum energy consumption of $C_A(G^i)$ computed by all assignments A satisfying $T_A(G^i) \leq j$ with probability $\geq P_{i,j}$. We introduce the ***operator "⊕"*** in this paper. For two (Probability, Consumption) pairs H_1 and H_2, if H_1 is ($P^1_{i,j}$, $C^1_{i,j}$), and H_2 is ($P^2_{i,j}$, $C^2_{i,j}$), then after applying the ⊕ operation between H_1 and H_2, we get pair (P', C'), where $P' = P^1_{i,j} * P^2_{i,j}$ and $C' = C^1_{i,j} + C^2_{i,j}$. We denote this operation as "$\mathbf{H_1} \oplus \mathbf{H_2}$".

$D_{i,j}$ is the table in which each entry has a link list that stores pair ($P_{i,j}$, $C_{i,j}$) sorted by $P_{i,j}$ in an ascending order. Here, i represents a node number, and j represents time. For example, a link list can be (0.1, 2)→(0.3, 3)→(0.8, 6)→(1.0, 12). Usually, there are redundant pairs in a link list. We use Lemma 4.1 to cancel redundant pairs.

Lemma 4.1 *Given ($P^1_{i,j}$, $C^1_{i,j}$) and ($P^2_{i,j}$, $C^2_{i,j}$) in the same list: 1) If $P^1_{i,j} = P^2_{i,j}$, then the pair with minimum $C_{i,j}$ is selected to be kept. 2) If $P^1_{i,j} < P^2_{i,j}$ and $C^1_{i,j} \geq C^2_{i,j}$, then $C^2_{i,j}$ is selected to be kept.*

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

Algorithm 4.1 ***VASP_RS*** Algorithm

Require: M different voltage levels, a PDFG, the timing constraint L, and the rotation times Q.
Ensure: An optimal voltage assignment to minimize energy while satisfying L

```
1: rotation ← 0;
2: while (rotation < Q)
3:     Get the static schedule (a DAG) from input PDFG.
4:     Get scheduling graph from the DAG.
5:     Using algorithm VAP_M to get the near optimal assignment of voltage levels for the schedule graph.
6:     Using Rotation scheduling RS to retime the original PDFG and rotate down the first row.
7:     rotation++;
```

Algorithm 4.2 ***VAP_SG*** Algorithm

Require: M different voltage levels, a DAG, and the timing constraint L.
Ensure: An efficient voltage assignment

1) Topological sort all the nodes, and get a sequence A. 2) Count the number of multi-parent nodes t_{mp} and the number of multi-child nodes t_{mc}. If $t_{mp} < t_{mc}$, use bottom up approach; Otherwise, use top down approach. 3) For bottom up approach, use the following algorithm. For top down approach, just reverse the sequence. 4) Assume the sequence after topological sorting is $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_N$, 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 $v_{i_1}, v_{i_2}, \cdots, v_{i_W}$ are all child nodes of node v_i and *W* is the number of child nodes of node v_i, then

$$D'_{i,j} = \begin{cases} (0,0) & \text{if } W = 0 \\ D_{i_1,j} & \text{if } W = 1 \\ D_{i_1,j} \oplus \cdots \oplus D_{i_W,j} & \text{if } W > 1 \end{cases} \quad (1)$$

5) 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. 6) For each k in $E_{i,k}$, $D_{i,j} = D'_{i,j-k} \oplus E_{i,k}$ 7) Use the Lemma 4.1 to remove redundant pairs. Finally get $D_{N,j}$.

In algorithm *VAP_M*, without loss of generality, assume using bottom up approach. Algorithm *VAP_M* gives the near-optimal solution when the given PDFG is a DAG. In equation (1), $D_{i_1,j} \oplus D_{i_2,j}$ is computed as follows. let G' be the union of all nodes in the graphs rooted at nodes v_{i_1} and v_{i_2}. Travel all the graphs rooted at nodes v_{i_1} and v_{i_2}. If a node q in G' appears for the first time, we add the energy consumption of q and multiply the probability of q to $D'_{i,j}$. If q appears more than once, that is, q is a common node, then use a selection function to choose the type of q. For instance, the selection function can be defined as selecting the type that has a smaller execution time. The final $D_{N,j}$ we get is the table in which each entry has the minimum energy consumption with a guaranteed confidence probability under the timing constraint j.

The *RS* algorithm is described in detail as following: First input a DFG G and the environment parameters such as the timing constraint T, rotation times Q, the levels of

Algorithm 4.3 ***RS*** Algorithm

Require: the PDFG G and environment parameter.
Ensure: a new graph

1: Create a schedule S using list scheduling.
2: For i = 1 to Q,
3: Get a set of nodes U to be rotated;
4: Delete nodes in U from S;
5: For each $v \in S$, Retime v, and get new graph G_r;
6: Compact S according to G_r;
7: Insert nodes in U into S according to G_r;

voltages M, and voltages values R. Then create a schedule table S in step 2 by using list scheduling. From step 3 to 9, there is a loop. In each loop body, we rotate the schedule and retime the graph. In step 4, get a set of nodes U to be rotated, which are the nodes in the first row of S, and remove them from S. In step 6, modify the retiming function r of G when retime the nodes in U, and get a new DFG G_r. In step 7, compact S, which means find an earliest position for each node in S. Then in step 8 put the nodes in U back to S according to the principle of As Early As Possible (AEAP),

Algorithm Complexity In the worst-case, there are $O(|V|)$ steps to solve the common node problem, where $|V|$ is the number of nodes. Therefore, the complexity of *Algorithm VAP_M* is $O(|V|^2 * L * R * K)$. L is the given timing constraint, R is the maximum number of voltage levels for each node, and K is the maximum number of execution time variation for each node. Since there are Q rotations, the complexity of *VASP_RS* is $O(Q * |V|^2 * L * R * K)$.

5 Experiments

This section presents the experimental results of our algorithms. We compare our algorithm with list scheduling and the ILP techniques in [4] that can give near-optimal solution for the DAG optimization. We conduct experiments on a set of benchmarks including 4-stage lattice filter, 8-stage lattice filter, voltera filter, differential equation solver, RLS-languerre lattice filter, and elliptic filter. Three different voltage levels, R_1 (2.5), R_2 (1.8), and R_3 (1.2), 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 in Gaussian distribution. Each application has timing constraints. This assumption is good enough to serve our purpose to compare the relative improvement among different algorithms. For each benchmark, we conduct the experiments based on 3 processor cores. The experiments are performed on a Dell PC with a P4 2.1 G processor and 512 MB memory running Red Hat Linux 9.0.

The experimental results with 3 processors are shown in Table 3. In the table, columns "List", "ILP" , "VASP_RS" represent the results obtained by list scheduling, the ILP in [4], our *VASP_RS* algorithm, respectively. Columns "Ene." represents the energy consumption and columns "%I" represents the improvement of our algorithm over the ILP in [4] with different probability. The total average improvement of *VASP_RS* is shown in the last row.

From the experimental results, we can see that our algorithms achieve significant energy saving compared with the ILP in [4]. On average, *VASP_RS* shows a 32.6% reduction in hard real-time, and reductions of 46.8%, 55.6% with probability 0.9 and 0.8, respectively, for soft real-time DSP systems. The reason of such big improvement is because we use loop scheduling and DVS to shrink the schedule length and assign the best possible voltage level to minimize energy while satisfying time constraint with guaranteed probabilities.

TC	List	ILP	VASP_SR					
			0.8		0.9		1.0	
	Ene. (μJ)	Ene. (μJ)	Ene. (μJ)	%I (%)	Ene. (μJ)	%I (%)	Ene. (μJ)	%I (%)
RLS-laguerre Filter, 3 processors								
60	1518	1265	541	57.2	688	45.6	841	33.5
80	1325	993	461	53.6	565	43.1	677	31.8
100	1231	876	448	48.8	532	39.2	624	28.8
110	1128	733	314	57.1	394	46.3	461	37.1
120	1052	710	313	55.9	378	46.7	451	36.5
8-stage Lattice Filter, 3 processors								
160	3208	2309	1106	52.1	1312	43.2	1584	31.4
180	2501	1625	736	54.7	902	44.5	1074	33.9
200	2106	1495	728	51.3	855	42.8	1009	32.5
220	1892	1381	695	49.7	831	39.8	941	31.2
240	1587	1095	541	50.6	624	43.1	733	33.1
Elliptic Filter, 3 processors								
100	2507	1830	626	54.6	963	47.4	1245	32.1
120	2210	1658	746	55.1	857	48.3	1071	35.4
140	1992	1395	653	53.2	765	45.2	927	33.5
160	1880	1341	551	58.9	685	48.9	896	33.2
170	1677	1209	510	57.8	640	47.1	785	35.1
Average Imp. over ILP			55.6	–	46.8	–	32.6	

Table 3. The energy comparison for the schedules generated by list scheduling, the ILP in [4], and the VASP_SR algorithm.

6 Conclusion

This paper combined loop scheduling and DVS to solve the VASP problem. By taking advantage of the uncertainties in execution time of tasks, our approach give out voltage assignments and scheduling to minimize the expected total energy consumption while satisfying timing constraint with guaranteed probabilities. We repeatedly regroup a loop based on rotation scheduling and decrease the energy by voltage selection as much as possible within a timing constraint. We proposed an algorithm *VASP_RS*, to give the efficient solutions. Our approach can handle loops efficiently.

References

[1] L.-F. Chao, A. LaPaugh, and E. H.-M. Sha. Rotation scheduling: A loop pipelining algorithm. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16:229–239, Mar. 1997.

[2] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.

[3] 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.

[4] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *DAC*, pages 183–188, 2002.