Algorithms

Task assignment and scheduling for multi-processor distributed systems is an NP-complete problem [1]. [2] proposes two methods based on Simulated Annealing. [3] uses Genetic Algorithm to solve the task assignment and scheduling problem. [4] proposes Longest Time First (LTF) method. For all above work, energy reduction is performed by utilizing slack only after task assignment and scheduling.

Two algorithms are proposed for task assignment and voltage scaling with continuous supply voltage. One is based on non-linear programming and the other is a heuristic algorithm based on the computation intensity of tasks. We explicitly consider energy minimization and voltage setup during task assignment and voltage scaling.

1 System Model

We have total M PEs, and N peroidic tasks, each of which is representing by a tuple (a_i, d_i, e_i), where a_i and d_i are arrival time and deadline of task i, respectively, and e_i is the execution time of the task under the maximum V_{dd}, which we call nominal V_{dd}. In this work we assume the one-to-one relationship between frequency $freq$ and supply voltage V_{dd} as:

$$f = \frac{k \cdot (V_{dd} - V_{th})^2}{V_{dd}} \tag{1}$$

where k is a constant and V_{th} is the threshold voltage. We further assume nominal $freq$ under nominal V_{dd} as 1. All other frequencies are normalized to the nominal $freq$.

We sort the arrival time and deadline of all tasks in the non-decreasing order. as the sequence of (t_1, t_2, ..., t_{2N}). Such sequence partition the whole timeline into $2N - 1$ time slots.

We need to solve two problems: (1) which PE each task is assigned to; and (2) what is the V_{dd} for each task. We assume continuous V_{dd} and EDF scheduling with preemption among tasks assign on the same PE. Context switch overhead is incorporated into task's execution time.

2 Non-Linear Programming

We define the following variables:

- $$A_{ij} = \begin{cases} 1 & \text{if task i is scheduled on PE j} \\ 0 & \text{otherwise} \end{cases}$$
- S_{ik}: time of task i to be executed during the time slot $[t_k, t_{k+1}]$.

- f_{ik}: frequency when task i to be executed during the time slot $[t_k, t_{k+1}]$.

We further assume there is a quadratic relationship between total energy and frequency on each PE as $E(freq)$. The problem becomes to minimize:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{2N-1}(A_{ij}\cdot E(f_{ik})) \tag{2}$$

subject to:

$$\sum_{j=1}^{M} A_{ij} = 1 \qquad \text{for i = 1, ..., N} \tag{3}$$

$$\sum_{j=1}^{M}\sum_{k=1}^{2N-1}(A_{ij}\cdot S_{ik}\cdot f_{ik}) = e_i \qquad \text{for i = 1, ..., N} \tag{4}$$

$$\sum_{i=1}^{N}(A_{ij}\cdot S_{ik}) \leq t_{k+1} - t_k \quad \text{for } 1 \leq k \leq 2N-1, 1 \leq j \leq M \tag{5}$$

3 Heuristic Algorithm

The heuristic algorithm is based on the concept of task computation intensity, which is define as $R = \frac{c_i}{d_i - a_i}$. For any interval ($A$, B), the intensity of that interval is $\sum_{i \in S} R_i$, where S is the task set containing $\forall$ task i with [a_i, d_i] $\subseteq$ [A, B]. The algorithm has five stages:

1. Identify the critical interval. The critical interval is the interval (A, B) with largest total task computation intensity.

2. Schedule all task in the critical interval on M hypothetic PEs. Suppose there are k tasks as $1 \leq i \leq k$, and sorted in the non-ascending order of their intensity, i.e., $R_i \geq R_j$ if $i < j$. The algorithm is to allocate task one by one, from that with largest intensity to that with smallest intensity. If one task can be assigned to one hypothetic PE without interfering with other PE, then it is assigned to that PE. Otherwise, the task is assigned to a PE that schedulability of that PE is satisfied.

3. Map the M hypothetic PEs to M realistic PEs. During this mapping we try to minimize the voltage difference between tasks on our hypothetic PEs and neighboring tasks on realistic PEs.

4. Remove all tasks in the critical interval, compress remaining tasks. If there is any task unscheduled, go back to Stage 1.

References

[1] W. H. Wolf, "Hardware-software co-design of embedded systems," *Proc. IEEE*, vol. 82, pp. 967–989, July 1994.

[2] F. Gruian, "System-level design methods for low-energy architectures containing variable voltage processors," in *Workshop on Power-Aware Computer Systems*, 2000.

[3] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins, "Energy-aware runtime scheduling for embedded-multiprocessor socs," *IEEE Design & Test of Computers*, vol. 18, pp. 46–58, Sept 2001.

[4] D. Zhu, R. Melhem, and B. R. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 686–700, July 2003.