Power Minimization using System-Level Partitioning of Applications with Quality of Service Requirements

Gang Qu and Miodrag Potkonjak
Computer Science Department, University of California, Los Angeles, CA 90095
{gangqu,miodrag}@cs.ucla.edu

Abstract

Design systems to provide various quality of service (QoS) guarantees has received a lot of attentions due to the increasing popularity of real-time multimedia and wireless communication applications. Meanwhile, low power consumption is always one of the goals for system design, especially for battery-operated systems. With the design trend of integrating multiple processor cores and memory on a single chip, we address the problem of how to partition a set of applications among processors, such that all the individual QoS requirements are met and the total energy consumption is minimized. We exploit the advantages provided by the variable voltage design methodology to choose the voltage for each application on the same processor optimally for this purpose. We also discuss how to partition applications among the processors to achieve the same goal. We formulate the problem on an abstract QoS model and present how to allocate resources (e.g., CPU time) and determine the voltage profile for every single processor. Experiments on media benchmarks have also been studied.

1 Introduction

Providing the clients their required quality of service (QoS) is crucial for the server in any client-server model. Failure to meet the QoS requirement leads to the unsatisfactory of the clients and makes the server's effort unprofitable. Modern applications (like real-time multimedia, E-commerce, distributed simulation, etc.) have various types of requirements on the QoS. It has already been a challenge to design system that is capable of meeting (a majority of, if not all) these QoS requirements. Meanwhile, low power consumption is considered one of the most important criteria for the design of application specific integrated circuits (ASIC) and other mobile computing devices, the core of systems that carry out these applications. Current semiconductor technology allows the integration of multiple programmable processors and memory structures on a single die, which enables the implementation of systems on a single chip. Idealy, for a given set of applications, we want to build systems that can provide guaranteed QoS and consume as less power as possible. In this paper, we address the following problem:

Given a set of applications with individual QoS requirements, and k processors, how to assign applications to processors such that all the QoS requirements are met and the total energy consumption of the k processors is minimized.

Although there have been plenty of literatures on measuring and pricing for QoS in the networking society [6, 7, 19], it is hard to find an explicit one-fit-all definition for the quality of the real-time, distributed multimedia services, because these services should be application specific and user dependent. In our model, we treat QoS as a function of the required resources such as bandwidth, CPU time, buffer space. No specific assumptions are made so both our approaches and results are applicable to most system designs. The key technique to lower energy consumption is using low supply voltage. We exploit the advantages provided by the variable voltage design methodology [9] to choose the processor and the supply voltage for each application to minimize the total energy consumption with guaranteed amount of QoS.

2 Related Work

There have been several proposals and prototype implementations of end-to-end transport protocols for delivering QoS guarantees, such as ST-II [20], RSVP [21], and the Tenet Real-Time Protocol Suite [4]. [3] gives a survey on the QoS architectures. The QoS of the multimedia applications, at the highest level, can be interpreted as the quality of images or sound, and at lower levels, it can be measured by bits per second or transit delay. In [2], QoS is defined as a combination of the basic quality metrics for the network layer: delay, jitter, bandwidth, and reliability. Lawrence [11] discusses the metrics based on the QoS attributes of timeliness, precision, and accuracy that can be used for system specification, instrumentation, and evaluation. Rajkumar et al. [18] present an analytical approach for satisfying multiple QoS dimensions in a resource-constraint environment (see below for details) and provide optimal and near-optimal resource allocation schemes for two special cases.

The dominant source of power dissipation in CMOS circuits is the dynamic power which is proportional to the square of the supply voltage. Therefore, reducing the supply voltage is quite effective to lower power[17]. Many CMOS circuits have always been capable of operating over a range of supply voltages and recent advances in power supply technology make it possible to create processor cores with supply voltage that can be varied at run time according to application timing constraints. Macken et al. [10] first proposed the idea of dynamically adapting voltage to operate at the point of lowest power consumption for given temperature and process parameters. Namgoong et al. [13] developed efficient DC-DC converters that allow the output voltage to be rapidly changed under external control. Researchers at MIT [5, 8] have applied the idea of voltage adaptation based on data dependent computation time from [14] to synchronously clocked circuits. Finally [9] describes the design methodology of variable voltage core-based systems.

Partitioning has been well-studied as a combinatorial problem, especially in many subfield of VLSI CAD, where any top-down hierarchical approach to system design must rely on some underlying partitioning techniques. Alpert and Kahng [1] survey the major variant formulations and approaches of, but not limited to the netlist partitioning problem.

This is the first attempt to balance the requirements of QoS and energy on multiple processors. In particular, we consider how to minimizes energy consumption with given QoS guarantees. We

0-7803-5832-5/99/ $10.00 © 1999 IEEE

adopt an abstract QoS model and use partition techniques as well as dynamically adapting supply voltages to achieve this goal.

3 Problem Formulation

Rajkumar et al. [18] proposed the QoS-based Resource Allocation Model (Q-RAM) to analyze two problems: (i) Satisfying simultaneous requirements along multiple QoS dimensions such as timeliness, cryptography, data quality and reliable packet delivery, and (ii) Allowing applications have access to multiple resources such as CPU, disk bandwidth, network bandwidth, memory simultaneously. Q-RAM considers a system of n applications and m resources each with a finite capacity, as well as a set of QoS requirements. Each application has a minimal resource requirements for each QoS dimension, and it achieves a certain utility with the allocated resources. The objective is to make resource allocations to each application such that every application satisfies its QoS requirements on all dimensions and the total system utility (which is a weighted sum of each application's utility) is maximized.

Following is a description of our system based on Q-RAM, but we focus on the system's total energy consumption:

- **Processor core:** The variable voltage processor core is capable of running at a range of supply voltages. Suppose the supply voltage is $v_{dd}(t)$ at time t, then the power consumption is $P(t) = \alpha C v_{dd}^2 f$ and the energy dissipation over the period $[0, T]$ is $E = \int_0^T P(t)dt$. The circuit delay is $\frac{kv_{dd}}{(v_{dd}-v_t)^2}$, where v_t is the threshold voltage [17].

- **Resources:** $m \geq 1$ resources $\{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_m\}$ are available, each resource has a finite capacity. We abuse the notation a little bit by denoting $\mathbf{R}_i$ the capacity for resource $\mathbf{R}_i$.

- **Applications:** There are $n \geq 1$ applications $\{\tau_1, \tau_2, \cdots, \tau_n\}$ to be executed on k processors. The execution time of an application varies as the processor's speed, which is determined by the supply voltage, changes. (In this paper, we discuss the simplest case where the applications are independent, have the same arrival time and no deadline constraints.)

- **Utilities:** The utility U_i of the application τ_i is the value that is accrued by the system when τ_i is allocated $\mathbf{R}^i = (R_{i,1}, \cdots, R_{i,m})$. The total system utility with a resource allocation scheme $(\mathbf{R}^1, \mathbf{R}^2, \cdots, \mathbf{R}^n)$ is $\sum_{i=1}^n \omega_i U_i(\mathbf{R}^i)$, where ω_i is the relative importance of application τ_i. U_i also depends on the supply voltage. Denote $U_{ref}^i(R^i)$ as the utility at reference voltage v_{ref}.

Problem: For each application τ_i in the above system, determine which processor will execute τ_i, find the resource allocation $R^i = \{R_1^i, \cdots, R_m^i\}$ and supply voltage schemes $v_i = v_i(t)$ such that:

1. $\sum_{i=1}^n R_j^i \leq R_j$: each resource is within its capacity.
2. $U_{v_i}(R^i) \geq U^i$ and/or
2'. $\sum_{i=1}^n \omega_i U_{v_i}(R^i) \geq U^0$: guaranteed QoS.
3. The total energy consumption is minimized.

4 Voltage Profile on a Single Processor

We first discuss how to allocate resource (in particular, the CPU time) and determine the voltage profiles for applications on a single processor. We assume the QoS function is monotone and non-decreasing with respect to the amount of allocated resource. Thus the energy consumption can be minimized when the applications use all the resources. Moreover, if we assume that the QoS function is proportional to the amount of computation, from the convexity of the power *vs.* speed (and thus voltage) curve [15], we have:

Lemma 4.1 To finish the application with a guaranteed QoS, the energy consumption is minimized only if the processor operates at a constant supply voltage.

Any general QoS function can be approximated by piecewise linear functions [16]. Figure 1 shows a given utility function at nominal voltage and its linear approximation at the same voltage level. Given an application which requires a QoS in the amount of U_0 and a finishing time in $[t_1, t_2]$, let v_i be the voltage level such that U_0 is obtained at exactly time t_i and v_{min}, v_{max} be the minimal and maximal physical possible voltages. Then:

Lemma 4.2 The optimal voltage to minimize energy consumption with QoS U_0 is either $\min\{v_{max}, v_1\}$ or $\max\{v_{min}, v_2\}$.

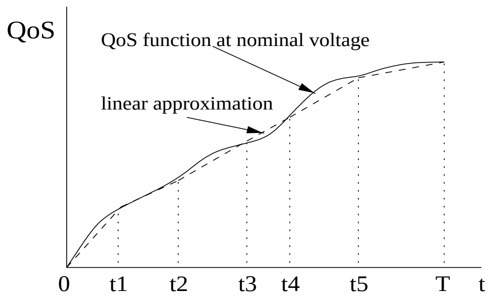

Figure 1: Piece-wise linear approximation for a QoS function.

Based on Lemma 4.2, a heuristic (**P**artition and **L**inear **A**pproximation) is proposed which can determine, for a given amount of QoS U_0 and deadline with the QoS *vs.* execution time at the reference voltage, the voltage scheme and the finish time to accomplish U_0 and consumes the least energy[16]. Moreover,

Theorem 4.3 The PLA heuristic can provide solution that consumes energy arbitrarily close to the optimal strategy with the guaranteed amount of QoS.

We propose an approach to solve the following problem on a single processor: given n applications $\{\tau_1, \cdots, \tau_n\}$ with individual QoS requirements and the QoS *vs.* execution time curves at the reference voltage, determine how much CPU time (and therefore voltage profile) should be assigned to each application such that all application's QoS requirements can be satisfied by a deadline T and the total energy consumption is minimized.

Procedure 4.4

1. evenly partition $[0, T]$: $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T$.
2. for each application τ_i, apply PLA to find the "best" strategy $S_i(t_j)$ to finish τ_i in time $[0, t_j]$ for $j = 1, 2, \cdots, n$.
3. pair-up the τ_i's and for each pair (τ_p, τ_q), determine $S_{(p,q)}(t_j)$ from $S_p(t_l)$ and $S_q(t_j - t_l)$ for $j = 2, \cdots, n, l = 1, 2, \cdots, j-1$, where $S_{(p,q)}(t_j)$ is the "best" strategy to finish the pair (τ_p, τ_q) in $[0, t_j]$.
4. if there are at least two pairs left, goto step 3 and treat these pairs as "new applications" and use $S_{(p,q)}(t_j)$'s as $S_i(t_j)$'s.
5. return $S_{(1,2,\cdots,n)}(T)$ as the "best" strategy.

The essence of Procedure 4.4 is to merge the applications in pairs and use dynamic programming. We partition the interval evenly to avoid repeated calls to the PLA heuristic. By "best" strategy, we mean the solution from PLA which is based on the linear approximation of the QoS *vs.* execution time curve. This gives the minimal energy consumption with the QoS guarantees. PLA heuristic is capable of providing solutions arbitrarily close to the optimal, however this requires refinement of the interval partition[1] and increases the runtime and space complexities.

[1]The accuracy of the solution from PLA depends on how good the linear approximation is, the more tiny the partition is, the better the linear approximation is. Also, it is worth to mention that in PLA, it is not required to have equal partition.

5 Energy Minimization on Multiple Processors

When there are multiple processors available, we can utilize the parallelism by applications partitioning. As a result, each application will share a processor with only part of the applications and get more execution time so as to lower its supply voltage. The total energy consumption can be saved from the relationships among power, energy, gate delay and voltage (in particular the quadratic relation between power and voltage) as shown in Section 3.

We assume no dependency among the applications and use the greedy single-move improvement method combined with Procedure 4.4 to find the partition and determine the voltage.

Procedure 5.1

1. randomly assign n applications to the k processors.
2. for each processor $i(= 1, 2, \cdots, k)$, calculate its energy consumption E_i by Procedure 4.4.
3. let processor i be the one that consumes the least energy.
4. for each application τ not assigned to processor i, compute the gain by moving τ from its assigned processor j to i:

$$gain(\tau) = E_i + E_j - E'_i - E'_j$$

   where E'_i and E'_j are the energy consumed by processors i and j after application τ is moved from j to i.
5. if the maximal gain is positive, make the move and goto step 3.
6. return the applications on each processor and their voltage profiles.

We start from a random partition (steps 1 and 2) and iteratively move to the best neighboring solution (steps 4 and 5). By neighbors we mean two partitions that only differ at the assignment of one application. In step 3, we decide to assign one more application to the processor that has the least workload. The one that can save most energy is selected (step 5). This process terminates when the algorithm reaches a local optimal.

When we re-assign an application, we have to run Procedure 4.4 again to compute the energy consumption of both processor which releases and receives this application. Procedure 4.4 is conducted in such a way that the recomputation is minimized. In the worst case, moving (or adding) an application needs to re-calculate at most the solutions for $\log_2 n$ nodes (all the ancestors of the removed node) in the binary merging tree (Figure 2).

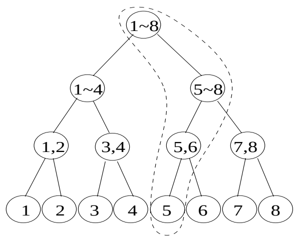

Figure 2: Merge tree for 8 applications. After node 5 is moved, node (56) will be identical to node 6; nodes (5∼8) and (1∼8) have to be recomputed.

There exist many possible improvements to Procedure 5.1. For example, instead of the simple single-move, we can use Kernigan-Lin algorithm, Fiduccia-Mattheyses algorithm, simulated annealing, tabu search, or genetic algorithms. However, since this approach depends on Procedure 4.4 and essentially depends on the PLA and linear approximation of the QoS *vs.* execution time curves at various voltage. By no means the linear approximation can be accurate, and therefore any powerful partitioning heuristic cannot guide us towards the optimal. For the same reason, in step 5 we stop at a local optimal without doing any hill-climbing to get out of such local minima. Also, multiple starting points can be used to replace the random solution in step 1. Further improvement is possible if we take advantage of the applications' QoS *vs.* execution time curves when building the merging tree in Procedure 4.4.

Unfortunately, due to the NP-hardness of the partitioning problem, we are unable to provide any error bound for the total energy consumption, even the PLA heuristic can give solution arbitrarily close to the optimal for any single application. Because of the non-linear functional dependency between power and execution time, it remains a challenge as how to estimate the optimal total energy consumption for multiprocessor.

6 Experimental Result

We used six applications [12] to demonstrate the effectiveness of the approach. JPEG software from the Independent JPEG Group implements the JPEG baseline, extended-sequential, and progressive compression processes. We used integer DCT for decoding a JPEG file to a PPM file. Integer DCT and progressive compression process options are used for compression. MPEG software from MPEG Software Simulation Group decode MPEG-1 and MPEG-2 video bitstreams. We obtained and used an MPEG animation file for our experimentation. As an input to the PGP software, we used an ASCII text file. The data file for GSM and G.721 encoding is obtained from Fine Arts Graphics Lab, Ball State University. The data file is in Sun u-law format. Table 1 gives a brief summary of the applications, while summary of data used is given in Table 2. (also available at http://www.cs.ucla.edu/~leec/mediabench). Table 3 lists the characteristics of the microprocessors we use in the simulation. For each application we defined 8 different levels of services for each application, mainly related to the requested resolution, the rate of service, and the bit error-rate (BER). For MPEG and JPEG, the levels are defined by the standards themselves.

Application	Source	Dynamic instructions
JPEG encoder	Independent JPEG Group	3.9 million
JPEG decoder	Independent JPEG Group	13.8 million
PGP encryption	Philip Zimmermann	68.8 million
MPEG decoder	MPEG Software Simulation Group	1.1 billion
G.721	Sun Microsystems, Inc.	62.9 million
GSM	Technische Universitaet Berlin	148.4 million

Table 1: Applications used in the experimentation.

Benchmark	File size	Format	Description
JPEG encoding	101,484 bytes	PPM	bit map
JPEG decoding	5,770 bytes	JPG	a JPEG compressed
PGP encryption	194,723 bytes	ASCII	a plain text
MPEG decoding	1,567,055 bytes	MPEG	14.98 s., 476 frames
GSM encoding	26,372 bytes	Sun .au u-law	one sentence
G.721 encoding	26,372 bytes	Sun .au u-law	one sentence

Table 2: Characteristics of the data used as application input parameter in the experimentation.

Microprocessor core	Clock (MHz)	MIPS	Power diss. (mW) (Voltage)
StrongARM	233	266	300 (1.65)
ARM, 7	40	36	200 (5)
ARM, 7 Low-Power	27	24	45 (3.3)
LSI Logic, TR4101	81	30	81 (3.3)
LSI Logic, CW4001	60	53	120 (3.3)
LSI Logic, CW4011	80	120	280 (3.3)
DSP Group, Oak	80	80	190 (5)
NEC, R4100	40	40	120 (3.3)
Toshiba, R3900	50	50	400 (3.3)
Motorola, 68000	33	16	35 (3.3)

Table 3: Characteristics of the microprocessor cores.

For the simplicity of demonstration of the effectiveness of the proposed approaches, we use partitioning to maximize the cumulative benefit (QoS) of all tasks on a set of processor for a given amount of power. Our result is compared to the following upper bound: schedule (optimally) the tasks on one processor that has processing speed and power equal to the sum of the processing speeds and powers of each of the individual processors.

We tested our results on several mixes of different number of tasks and processors. Tables 4 and 5 contain information for two different levels of total power consumption as a function of the total power consumption of both processors at the nominal voltage (3.3V). Table 4 has information when the total consumption is equal to the sum of the maximal power consumptions of the individual processors, Table 5 shows the case when at most 75% of that amount can be used.

Rows and columns in the tables indicate the number of tasks and processors in the experiment respectively. The entries in both table indicate what percentage of the upper bound benefit obtained for a given level of power and processing speed. In almost all cases exceptionally close match between the obtained partitioning results and the upper bound is achieved, clearly indicating that all potential of partitioning is essentially realized. Only in few cases the discrepancy is somewhat higher. In these cases, since the instances are small, the complete solution enumeration indicates that the optimal solutions were obtained.

Number of Tasks	Number of Processors			
	2	3	4	8
8	99.9	99.7	99.3	99.1
10	100	99.7	99.6	99.4
20	100	100	99.8	99.6
25	100	100	100	99.6
50	100	100	100	99.7
75	100	100	100	99.9
100	100	100	100	100

Table 4: The effectiveness of the partitioning vs. the upper bound on QoS. each processor uses 100% of its maximal power.

Number of Tasks	Number of Processors			
	2	3	4	8
8	96.8	93.2	93.0	88.9
10	96.9	93.5	93.2	90.6
20	98.3	95.5	95.3	93.7
25	98.7	96.2	94.9	93.8
50	99.4	99.1	98.7	97.4
75	99.6	99.3	99.1	99.0
100	99.9	99.8	99.4	99.3

Table 5: The effectiveness of the partitioning vs. the upper bound on QoS, each processor uses at most 75% of its maximal power

7 Conclusion

Quality of service is intrinsically connected to many popular applications such as multimedia. Minimizing power/energy consumption is an important issue for modern system design, especially for the battery-operated systems that support the QoS-sensitive applications. We propose the problem of how to allocation resource (e.g., CPU time) and determine the supply voltage to satisfy the various QoS requirements of a set of applications on a multiprocessor system with minimal energy. We develop a dynamic programming-based approach for single processor and use iteratively improvement algorithm to find the partition.

The non-precise linear approximation for the Qos *vs.* execution time curve at different supply voltage prevents the usage of other powerful partitioning algorithms. It remains a challenge as how to model and measure the QoS, not to mention the variable supply voltages. In this paper, we only consider the case that there is no function dependency among applications. It is of great interest when real-time applications require interaction and timing constraints (e.g., individual arrival time, latency, synchronization). In our approach, we use an abstract QoS model, on which many explicit QoS requirements can be built. This makes our approach and results applicable to many real life systems.

References

[1] C.J. Alpert and A.B. Kahng, *Recent Directions in Netlist Partitioning: A Survey.* Integration: The VLSI Journal. Vol. 19, pp. 1-81, 1995.

[2] J. Altmann and P. Varaiya. *INDEX project: user support for buying QoS with regard to user's preferences.* 1998 Sixth International Workshop on Quality of Service (IWQoS'98), pp. 101-104, 1998.

[3] C. Aurrecoechea, A.T. Campbell, and L. Hauw. *A Survey of QoS Architectures.* Multimedia Systems, Vol.6, No.3, pp. 138-151, 1998.

[4] A. Banerjea, D. Ferrari, B. Mah, M. Moran, D. Verma, and H. Zhang. *The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences.* IEEE/ACM Transactions on Networking, Vol.4, No.1, pp. 1-11, February 1996.

[5] A. Chandrakasan, V. Gutnik, T. Xanthopoulos. *Data driven signal processing: an approach for energy efficient computing.* International Symposium on Low Power Electronics and Design, pp. 374-352, 1996.

[6] D. Ferrari and L. Delgrossi. *Charging for QoS.* 1998 Sixth International Workshop on Quality of Service (IWQoS'98), pp. VII-VIII, 1998.

[7] E.W. Fulp, M. Ott, D. Reininger and D.S. Reeves. *Paying for QoS: an optimal distributed algorithm for pricing network resources.* 1998 Sixth International Workshop on Quality of Service (IWQoS'98), pp. 75-84, 1998.

[8] J. Goodman, A.P. Dancy, and A.P. Chandrakasan. *An Energy/Security Scalable Encryption Processor Using an Embedded Variable Voltage DC/DC Converter.* IEEE Journal of Solid-state Circuits, pp. 1799-1809, 1998.

[9] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M.B. Srivastava. *Power Optimization of Variable Voltage Core-Based Systems.* Proceedings 1998 Design and Automation Conference (35th DAC), pp. 176-181, 1998.

[10] V. Von Kaenel, P. Macken, M. G. R. Degrauwe. *A voltage reduction technique for battery-operated systems.* IEEE Journal of Solid-State Circuits, Vol. 25, No. 5, pp. 1136-1140, 1990.

[11] T.F. Lawrence. *The quality of service model and high assurance.* Proceedings of High-Assurance Engineering Workshop, pp. 38-39, 1997.

[12] C. Lee, M. Potkonjak, and W.H. Mangione-Smith. MediaBench: a tool for evaluating and synthesizing multimedia and communications systems. International Symposium on Microarchitecture, pp.330-335, 1997.

[13] W. Namgoong, M. Yu, T. Meng. *A high-efficiency variable-voltage CMOS dynamic dc-dc switching regulator.* 1997 IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers, pp. 380-381, 1997.

[14] L. S. Nielsen, C. Niessen, J. Sparso, K. van Berkel. *Low-power operation using self-timed circuits and adaptive scaling of the supply voltage.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.2, No.4, pp. 391-397, 1994.

[15] G. Qu. *Scheduling Problems for Reduced Energy on Variable Voltage Systems.* Master thesis. Spring 1998.

[16] G. Qu and M. Potkonjak. *Energy Minimization and Utility Maximization with Guaranteed Quality of Service.* Technical Report, Computer Science Department, UCLA. 1999.

[17] J.M. Rabaey and M. Pedram (Ed.). *Low Power Design Methodologies (Chapter 1).* Kluwer Academic Publishers, 1996.

[18] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. *A resource allocation model for QoS management.* Proceedings. The 18th IEEE Real-Time Systems Symposium, pp. 298-307, 1997.

[19] M. Siler and J. Walrand. *On-line measurement of QoS for call admission control.* 1998 Sixth International Workshop on Quality of Service (IWQoS'98), pp. 39-48, 1998.

[20] C. Topolcic. *Experimental Internet Stream Protocol, Version 2 (ST-II).* Internet RFC 1190, October 1990.

[21] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. *RSVP: A New Resource ReSerVation Protocol.* IEEE Network Magazine, Vol.7, No.5, pp.8-18, September 1993.

[22] *Microprocessor Report.* Issues in 1997.