Economia de Energia em Dispositivos Móveis

José M. Urriza*, Bruno A. Novelli†, J.C.B. Leite†, Javier D. Orozco*
*Dep. Ingenería Eléctrica y Computadoras, Universidad Nacional del Sur - CONICET, Bahía Blanca, Argentina
†Instituto de Computação, Universidade Federal Fluminense, Niterói, Brasil
*Email: {$jurriza$, $jorozco$}@criba.edu.ar
†Email: {$bnovelli$, $julius$}@ic.uff.br

***Resumo*— A redução do consumo de energia em dispositivos móveis (e.g., *notebooks*, *palms*, telefones celulares), por diversos fatores, é hoje um problema de importância capital. Dentre esses fatores pode-se citar a crescente necessidade de mais capacidade de processamento exigida pelos novos programas aplicativos e sistemas operacionais. Infelizmente, o avanço da tecnologia de baterias tem sido lento em relação à capacidade de fornecimento de energia e mesmo em relação ao grau de miniaturização exigido pelos dispositivos móveis. A tecnologia CMOS é hoje comumente utilizada no processo de fabricação de processadores. Para essa tecnologia verifica-se que o consumo de energia é aproximadamente proporcional ao quadrado da voltagem de alimentação. Assim, uma redução do nível de voltagem implica em uma diminuição de ordem quadrática no consumo de energia e na dissipação de calor. Vários processadores comerciais exploram essa característica e implementam um mecanismo denominado Regulagem Dinâmica de Voltagem (*Dynamic Voltage Scaling*). Essa é uma técnica efetiva na redução do consumo de energia, aplicável em várias situações. Particularmente, em sistemas móveis de tempo real, o desafio é minimizar o consumo de energia e garantir as restrições temporais desses sistemas. Nesse trabalho é proposto um algoritmo que opera em conjunto com um escalonador baseado em prioridades fixas para minimização do consumo de energia em dispositivos móveis. Resultados obtidos através de simulação indicam que esse algoritmo permite atingir índices de economia de energia próximos ao limite teórico.**

***Abstract*— Power savings in mobile devices (e.g., notebooks, palms, cell phones), for several factors, is becoming a paramount problem. Among those factors, one can indicate the ever increasing processing capacity needed by new application programs and operating systems. Unfortunately, the advances in battery technology have been slow, not only in the capacity of delivering increasing power levels, but also in the dimension aspect. CMOS is the most common technology employed for processor implementation nowadays. For this technology, power consumption is approximately proportional to the square power of the voltage supply. Thus, a reduction in the voltage supply level implies in a quadratic reduction in the power consumption and heat dissipation. Several commercial processors use this characteristic and employ a technique called Dynamic Voltage Scaling. This technique can be applied in several situations. In particular, in real-time mobile systems, the objective is to minimize power consumption and yet guarantee the temporal restrictions. In this paper, an algorithm is proposed, to be used together with a fixed priority scheduler, for minimizing power consumption in mobile systems. Simulation results show that this algorithm can attain power savings close to the theoretical limit.**

*José M. Urriza desenvolveu parcialmente este trabalho durante sua estada no IC-UFF como bolsista do Prosul/CNPq.

I. INTRODUÇÃO

Nos dispositivos móveis atuais (e.g. computadores portáteis, telefones celulares, PDAs, etc.), ano a ano, são incorporadas novas características que implicam na necessidade de uma maior capacidade de processamento, bem como novos componentes (e.g., interfaces de comunicação sem fio, maior quantidade de memória). Este aumento no processamento, contudo, acarreta um aumento no consumo de energia. O problema é agravado pois o avanço da tecnologia de baterias tem sido lento em acompanhar essas necessidades de consumo e ainda atender às limitações de tamanho dos dispositivos móveis. Na realidade, não se espera um substancial aumento da capacidade de fornecimento de energia das baterias do tipo ion de lítio (Li-ion), híbridas de metal níquel (NiMH) ou de níquel cádmio (NiCd) hoje utilizadas, já que se trata de tecnologia amadurecida. Adicionalmente, ainda faltam muitos anos para que as novas tecnologias, como células de energia, estejam comercialmente disponíveis. Assim, restam duas opções: o uso de grandes baterias (que nem sempre podem ser utilizadas) ou a realização de um gerenciamento de energia eficiente. Além disso, como um sub-produto, deve ser notado que com um menor consumo a vida útil da bateria e dos dispositivos por ela alimentados são normalmente estendidas. A redução da potência tem ainda como vantagem a simplificação dos mecanismos de dissipação nos dispositivos móveis, o aumento da confiabilidade dos componentes e uma redução em seu custo.

Dentre os diversos tipos de componentes de *hardware* que constituem um dispositivo móvel destacam-se o processador, o vídeo, o disco e o modem e/ou interface de rede como sendo os principais consumidores de energia. Para cada um deles existem técnicas de *software* que têm por objetivo minimizar o consumo de energia. O estudo aqui apresentado é focado no processador, pois é normalmente o componente que mais consome energia no sistema. Isto é verdadeiro não somente para pequenos dispositivos de mão como PDAs [1], que tem poucos componentes, mas também para computadores portáteis [2] que possuem muitos componentes, incluindo grandes telas de vídeo. Neste trabalho é utilizada a técnica conhecida como RDV - Regulagem Dinâmica de Voltagem (DVS - *Dynamic Voltage Scaling* [3], [4]) e é apresentado um algoritmo para ser usado em um sistema de prioridades fixas para o escalonamento de tarefas com o objetivo de economia de energia. Para motivar essa apresentação, é mostrado na

tabela 1 um exemplo do consumo de energia medido em um computador móvel [5]. Nessa tabela, a coluna *Processador* indica não somente o consumo desse componente mas também o da memória. É fácil notar nesse exemplo que, quando o vídeo está desligado e o disco está em *standby*, o fato de o processador passar do estado de carga máxima para ocioso leva a um substancial decréscimo de consumo. Obviamente, outros dispositivos, como memória, também são desativados, mas o processador pode ser responsável por até cerca de 60% desse decréscimo.

Tabela 1
CONSUMO DE ENERGIA MEDIDO EM UM LAPTOP HP N3350

Vídeo	Processador	Disco	Energia
Ligado	Ocioso	Girando	13,5W
Ligado	Ocioso	*Standby*	13,0W
Desligado	Ocioso	*Standby*	7,1W
Desligado	Carga Máxima	*Standby*	27,3W

A principal área de interesse desse trabalho são os sistemas móveis de tempo real. Por definição, sistemas de tempo real são aqueles submetidos a requisitos de natureza temporal. Nestes sistemas, os resultados devem estar corretos não somente do ponto de vista lógico-aritmético, mas também devem ser gerados no momento correto. Esses sistemas podem ser classificados como críticos (*hard real-time*) e não críticos (*soft real-time*). Nos sistemas críticos um atraso ou resposta incorreta é considerado um erro inaceitável, que pode resultar em um desastre natural ou envolver perda de vidas (e.g., sistemas de controle de tráfego aéreo, de controle de linhas ferroviárias e de usina nucleares). Já nos sistemas não críticos pode-se aceitar ocasionalmente uma resposta atrasada (e.g., sistemas de telefonia digital ou de vídeo conferência). Quando um sistema cumpre todas as restrições de tempo diz-se que ele é escalonável.

Seguindo o modelo multiprocessos-monoprocessador do trabalho seminal de Liu e Layland [6], as tarefas de tempo real a serem consideradas nesse trabalho são críticas, periódicas, independentes e preemptáveis. O conjunto de tarefas é especificado como S(n) = {(C_1, T_1, D_1), (C_2, T_2, D_2), ... , (C_n, T_n, D_n)}, onde C_i, T_i e D_i denotam, respectivamente, o tempo de execução de pior caso (WCET - *Worst Case Execution Time*), o período de ativação e o prazo para execução relativamente à ativação para cada tarefa *i*. Uma suposição comum é que, $\forall$i, $D_i = T_i$. Quando as tarefas competem pelo processador o conflito é resolvido de acordo com as regras definidas por uma política de prioridades. Embora o presente trabalho seja voltado para sistemas críticos, a mesma técnica pode ser utilizada, com pequenas modificações, para sistemas não críticos, como os empregados em aplicações multimídia (e.g., [7]).

O hiperperíodo de um conjunto de tarefas é definido como o mínimo múltiplo comum dos seus períodos. Claramente, devido à periodicidade das tarefas, os sucessivos hiperperíodos reproduzem exatamente o mesmo esquema de ativação de tarefas do primeiro hiperperíodo. Durante um hiperperíodo há momentos em que o processador estará ocupado executando tarefas e outros nos quais o processador se encontrará ocioso. Estes espaços vazios (ou seja, onde há ociosidade do processador) são chamados de folga (*slack*). Esses períodos de ociosidade são tradicionalmente aproveitados de diversas maneiras, como, por exemplo, para a execução de tarefas aperiódicas não críticas ([8], [9], [10]) ou ainda para a execução de partes opcionais associadas às tarefas críticas cujo processamento pode implicar em uma recompensa ([11], [12], [13], [14]). No trabalho aqui apresentado é utilizado um método de roubo de folga (*slack stealing*) para adiantar partes do tempo livre e, assim, poder diminuir a voltagem e a freqüência das tarefas, garantindo ainda o atendimento aos prazos definidos pelas aplicações. Outras técnicas de economia de energia combinam a redução da velocidade das tarefas e a redução da voltagem através de cálculos estáticos e de modificações dinâmicas na grade de execução. Esse último caso é denominado *gain time*, e consiste em aproveitar o tempo não utilizado por uma tarefa para diminuir a velocidade de execução de tarefas subseqüentes. Deve ser enfatizado que essas técnicas não tiram vantagem da possibilidade de avançar folgas. Contudo, elas geralmente atingem bons resultados pois a escala de execução estática é determinada pelos tempos de execução de pior caso. Em muitas situações, o tempo de pior caso pode chegar a ser até 10 vezes maior que o tempo médio de execução. No algoritmo aqui apresentado, além do potencial aproveitamento do *gain time*, uma economia de energia adicional pode ser obtida através do mecanismo de avanço de folgas.

Esse artigo é organizado como se segue. Na seção II é feita uma breve introdução ao funcionamento do mecanismo de RDV. Na seção III são apresentados trabalhos relacionados. Na seção IV é explicado o algoritmo de economia de energia. Na seção V são apresentados alguns resultados, na seção VI são indicados os trabalhos futuros e, finalmente, na seção VII é feita a conclusão.

II. REGULAGEM DINÂMICA DE VOLTAGEM

A tecnologia CMOS é hoje comumente utilizada em diversos processadores. Em circuitos integrados implementados com essa tecnologia o consumo de energia é aproximadamente proporcional ao quadrado da voltagem e varia linearmente com a freqüência ($\frac{E}{clock} \propto V^2$). A redução da freqüência operacional do processador, embora implique em diminuição do consumo, acarreta no aumento do tempo de execução das tarefas. Assim, o uso dessa técnica isoladamente não traz nenhum benefício. Uma redução no nível de voltagem, porém, tem um impacto de ordem quadrática no consumo de energia e na dissipação de calor. Contudo, para circuitos integrados reais, a redução na tensão de alimentação exige uma conseqüente redução na freqüência de operação. Vários processadores comerciais já exploram essas características (e.g., AMD *Mobile*, Intel *Centrino Mobile Technology* e *Xscale Technology*, Transmeta *Efficeon* e *Crusoe*, e National Semiconductors com a empresa ARM) e implementam o mecanismo

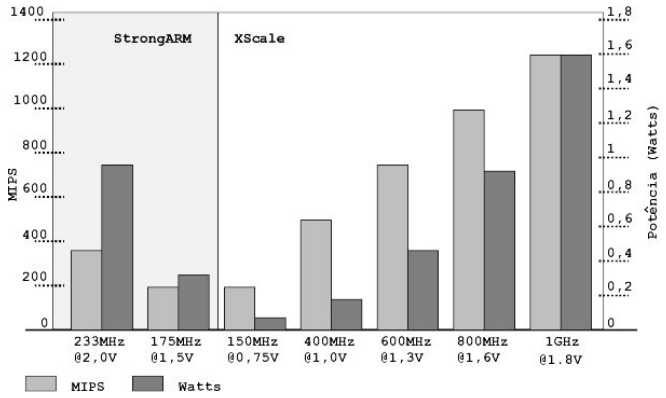

Fig. 1. Comparação do consumo de energia em Watts e do desempenho em MIPS para as arquiteturas StrongARM e XScale.

de regulagem dinâmica de voltagem (RDV). Para todos os processadores citados há um número limitado de níveis de tensão que podem ser utilizados. Ou seja, a tecnologia atual ainda não permite o oferecimento de circuitos com variação contínua de seu ponto de operação. Com base no mecanismo de RDV diversos algoritmos têm sido desenvolvidos, especialmente para sistemas de tempo real ([3], [5], [15], [16], [17], [18], [19], [20]).

A figura 1 apresenta um comparativo do consumo de energia e do desempenho para as arquiteturas StrongARM[1] e XScale[2]. Observa-se que um processador XScale operando com relógio de 1GHz, alimentado com 1,8V e consumindo 1,6W tem um desempenho de 1.200MIPS. O mesmo processador, operando a 800MHz e 1,6V, consome 0,9W com desempenho de 1.000MIPS. Verifica-se neste exemplo que, embora o desempenho tenha caído apenas 17%, a potência foi reduzida a quase metade (56%). Ou seja, a perda de desempenho quando se reduz a tensão e a freqüência é pequena comparada com o ganho na economia de energia.

No que se segue, quando houver menção a redução de freqüência estará subentendido que haverá uma correspondente diminuição na tensão de alimentação.

III. Trabalhos Relacionados

Diversos trabalhos têm tratado da aplicação de RDV e de algoritmos de escalonamento de processos para obtenção de economia de energia. Em sua grande maioria, esses trabalhos assumem a hipótese de que o consumo de energia é diminuído quando há redução da tensão de alimentação. Contudo, isso nem sempre é verdadeiro. Em [21] é demonstrado que, para alguns processadores comerciais, algumas das freqüências de operação são energeticamente ineficientes, ou seja, a freqüência imediatamente superior produz melhores resultados. Para que a hipótese anteriormente indicada seja verdadeira, o processador deve ter uma relação potência-freqüência convexa não decrescente. O trabalho desenvolvido por [22] indica como obter uma grade ótima de freqüências de operação de forma a minimizar os erros de quantização causados pelo uso de freqüências discretas. Esse é um resultado importante, não só para a construção e avaliação de modelos teóricos, mas também para a construção de processadores reais.

Em [23] Ishihara *et al.* apresentam alguns resultados teóricos para sistemas que usam RDV. Dentre os mais importantes resultados estão os teoremas a seguir:

- *Teorema 1*: Se um processador somente pode utilizar um número discreto de voltagens, o escalonamento com o uso de no máximo dois níveis de tensão, para qualquer restrição temporal, minimiza o consumo de energia;
- *Teorema 2*: Se um processador somente pode utilizar um número discreto de voltagens, as duas voltagens que minimizam o consumo de energia, para uma dada restrição temporal, são aquelas imediatamente vizinhas à voltagem ideal (aquela que implicaria em um único nível para todas as tarefas).

Um trabalho baseado em *gain time* e aproveitamento de folga é discutido em [5], onde Pillai e Shin apresentam abordagens estáticas (*off-line*) e dinâmicas (*on-line*) para sistemas de tempo real críticos. Na abordagem estática seleciona-se a menor freqüência operacional possível que ainda permita ao escalonador, no caso EDF (*Earliest Deadline First*) [6] ou RM (*Rate Monotonic*) [6], garantir todos os prazos do conjunto de tarefas. A redução é feita com base nos testes de escalonabilidade conhecidos para esses dois algoritmos, pela introdução de um fator de escala (relação entre a freqüência de operação pretendida e a freqüência máxima permitida pelo processador) para os tempos de execução máximos das tarefas. Na abordagem dinâmica, ou seja, para o caso em que as tarefas executam por menos tempo que o seu WCET, são apresentados dois algoritmos: o *Cycle Conserving*, desenvolvido para escalonadores RM e EDF, e o *Look Ahead*, desenvolvido somente para EDF. Ambos os algoritmos se baseiam no cálculo da utilização do processador ao término da execução de cada tarefa, de forma a reclamar quaisquer ciclos (previstos no WCET) não utilizados. A diferença entre esses algoritmos é que no *Look Ahead* é assumida uma abordagem otimista, mais agressiva, onde, no início da execução da tarefa ela é processada a uma velocidade inferior, podendo essa velocidade (feqüência-voltagem) ser aumentada para garantir o atendimento à restrição temporal.

O trabalho discutido em [11] também apresenta um algoritmo baseado em RDV para o escalonamento de tarefas periódicas em sistemas de tempo real críticos. A solução apresentada é feita em duas fases. A primeira, *off-line*, onde é obtida uma freqüência ótima para cada tarefa, assumindo que elas gastam seu tempo máximo de processamento em cada ativação. Na segunda fase, *on-line*, o tempo não gasto pelas tarefas (em relação ao seu WCET) é aproveitado durante a execução. É reportado no artigo que uma economia de energia de até 50% é obtida nessa fase em relação à fase estática. É ainda apresentado um mecanismo adaptativo de ajuste de

[1] http://www.arm.com

[2] http://developer.intel.com/design/intelxscale/

velocidade que utiliza a informação da carga de trabalho média (medida) para prever términos antecipados de execuções futuras e, assim, reduzir especulativamente velocidades. Segundo os autores, essa fase consegue atingir uma economia suplementar de energia de até 20% sobre o esquema dinâmico anteriormente citado.

Em [24], Kim *et al.* apresentam um algoritmo baseado em *gain time* para um conjunto de tarefas periódicas, em sistemas de tempo real críticos, utilizando um escalonador EDF. A principal característica deste algoritmo está na determinação do *gain time*, que é feita de duas formas. A primeira determina o *gain time* para as tarefas de maior prioridade que completam sua execução antes do seu WCET. Já na segunda forma, o aproveitamento provém das tarefas de menor prioridade que durante sua execução utilizaram menos que o seu WCET. Ao serem preemptadas, elas permitem que tarefas de maior prioridade possam aproveitar esse *gain time*. Este método acrescenta um pequeno *overhead* para a determinação do *gain time*, porém não há nenhuma fase *off-line*.

Em [25], Kim *et alli* apresentam uma avaliação de vários algoritmos com base em RDV ([5], [15], [24], [26]) para conjuntos de tarefas periódicas e premptivas em sistemas de tempo real críticos. Na avaliação é utilizado um simulador, *SimDVS*, que incorpora as características desses diferentes esquemas. Como resultado, eles apresentam uma comparação da eficiência na conservação de energia entre os algoritmos baseados em EDF ([5], [15], [24], [26]) e aqueles que usam RM ([5], [26]) com o limite ótimo teórico. Os autores concluem o trabalho mostrando que alguns dos algoritmos que utilizam EDF atingem resultados próximos ao ótimo (cerca de 9 a 12% piores). Para os algoritmos baseados em RM eles reportam resultados que ficam a uma distância significativa do ótimo.

O artigo [7] apresenta um esquema de escalonamento, GRACE-OS, para sistemas móveis multimídia. GRACE-OS utiliza o mecanismo de RDV em um escalonador voltado para aplicações não críticas de tempo real (*soft real-time*) para decidir qual a velocidade de processamento a ser empregada e também para determinar por quanto tempo uma tarefa será executada. As decisões de escalonamento são tomadas com base na distribuição de probabilidades das demandas das aplicações, que são obtidas através da estimação *on-line* do perfil dessas aplicações. Os autores indicam economias de energia variando entre 7% e 72%, oferecendo garantias probabilísticas de desempenho.

Um método bem distinto dos anteriores é descrito em [27], onde um compilador *power-aware* é proposto. Esse método é voltado para aplicações executando em sistemas embutidos, onde o WCET e o tempo médio de execução de seções de código de uma tarefa (e.g., *loops*, *procedures*) sejam conhecidos. O compilador insere pontos de gerenciamento de energia no início e no fim das seções para monitorar o tempo de execução, verificar a velocidade do processador e, eventualmente, alterá-la. A comparação entre o tempo de execução e o WCET permite ou incrementar ou diminuir a velocidade do processador. Outros métodos baseados nessa mesma técnica procuram, principalmente, otimizar a ocupação de memória (um problema importante em sistemas embutidos) e, assim, economizar energia [28].

IV. O Algoritmo Proposto

A. Motivação

O escalonamento com prioridades fixas é o método mais utilizado atualmente para a implementação de aplicações sujeitas a restrições de tempo real. No entanto, nos diversos métodos de economia de energia encontrados na literatura o escalonamento é majoritariamente realizado usando-se prioridades dinâmicas, mais especificamente, o algoritmo EDF. Algoritmos baseados em prioridades dinâmicas, como EDF e LLF [29], são mais eficientes no aproveitamento do processador que aqueles baseados em prioridades fixas, como o RM, sendo o seu limite teórico de utilização igual a 100%. Esses algoritmos são capazes de tratar tanto tarefas aperiódicas quanto periódicas, podendo ser usados em aplicações de tempo real críticas e não críticas. Contudo, se as aplicações necessitam somente de tarefas periódicas, o algoritmo RM, por sua implementação eficiente, simples e intuitiva, mostra-se uma opção interessante. Adicionalmente, RM é o algoritmo mais utilizado em sistemas industriais e é indicado pelo DoD dos EUA para aplicações de tempo real críticas [30]. Uma razão para isso é o fato do EDF ter um comportamento instável em circunstâncias especiais, como, por exemplo, em condições de sobrecarga [31]. A grande aceitação de RM como base para os sistemas operacionais de tempo real atuais foi um importante motivador para a sua utilização nesse trabalho.

B. Escala Estática de Freqüências

Dado um conjunto de tarefas, definido para uma determinada fase de operação do sistema, a pior situação de carga que pode ser apresentada para o escalonador é quando ocorre o instante crítico e quando todas as tarefas executam o WCET. O instante crítico é definido como aquele no qual todas as tarefas do sistema estão simultâneamente prontas para serem executadas. Nessa situação, o objetivo é escolher a menor freqüência de operação possível para o conjunto de tarefas, de tal forma que o escalonador RM ainda garanta todas as restrições de tempo. Nessa fase, o algoritmo funciona como uma variante exata ao proposto para RM em [5], procurando utilizar a folga disponível que ocorre para o sistema operando em freqüência máxima.

Essa primeira fase já oferece uma substancial melhoria na utilização da energia. Contudo, a economia é limitada pela quantidade de níveis de voltagem e freqüência do processador e pelo fator de utilização do sistema. Como os modelos de processadores reais ainda não trabalham com variações contínuas no mecanismo RDV, nem sempre é possível utilizar uma freqüência ótima e é, portanto, selecionado o nível discreto imediatamente acima ao que seria o ótimo para o RM. Em relação ao fator de utilização, esse esquema não aproveita

de forma ótima os tempos ociosos do processador, como explicado no exemplo a seguir.

Seja o conjunto de tarefas definidas na tabela 2. Admitindo-se uma situação de instante crítico, a figura 2(a) apresenta a escala gerada pelo algoritmo RM para o hiperperíodo desse conjunto de tarefas, assumindo-se freqüência máxima de operação. Nessa figura, os números nos retângulos indicam as tarefas e *e* representa os períodos ociosos (folga) do processador. É possível notar que, sem a utilização de RDV, as tarefas executam à maxima freqüência e por 75% do tempo do hiperperíodo. Existirá, então, 25% de folga cada vez que o hiperperíodo for executado. Em 2(c) está representada a escala gerada pela aplicação da fase estática do algoritmo, onde todas as tarefas executam à mesma freqüência (menor que a máxima). Como pode ser observado, mesmo após essa fase, ainda restam períodos de folga. Em 2(b) está representada a escala ótima em termos de ocupação de períodos ociosos, ou seja, desde que seja possível ao processador assumir qualquer freqüência e voltagem. Comparando-se as figuras 2(b) e 2(c) vê-se que há possibilidade de otimização do mecanismo, o que será feito em sua fase dinâmica. Graficamente, dado que a energia é calculada como o produto da potência pelo tempo, pode-se visualizar o ganho obtido como a área limitada entre a linha horizontal de f_{max} e o topo dos retângulos indicativos de execução das tarefas.

Tabela 2
CONJUNTO DE TAREFAS I

Tarefa	WCET	Período
1	1	3
2	1	4
3	1	6

C. Escala Dinâmica de Freqüências

O método aqui apresentado é baseado no mecanismo de determinação de folga desenvolvido por Urriza e Orozco [32]. Toda vez que uma tarefa for ser executada, o algoritmo verifica, em sua etapa dinâmica, qual a quantidade de folga para ela disponível e, de maneira gulosa, a atribui a essa tarefa. Com essa folga, calcula-se então se é possível baixar a freqüência de operação dessa tarefa. Caso seja possível, a tarefa executará nessa nova freqüência e a folga utilizada será subtraída da folga disponível. Caso essa folga disponível não seja suficiente para se baixar ao menos um nível a tarefa executará normalmente na freqüência em que se encontra. Os detalhes do método de determinação de folgas estão fora do escopo dessa apresentação e podem ser encontrados no relatório citado nesse parágrafo.

Embora os conjuntos de tarefas de tempo real crítico sejam especificados com o seu WCET, essas tarefas, durante sua execução, utilizam, em média, um tempo bem menor. Cada vez que uma tarefa está para ser executada não há como saber quanto tempo ela gastará e, para manter a escalonabilidade do conjunto, deve-se considerar que ela utilizará o seu tempo

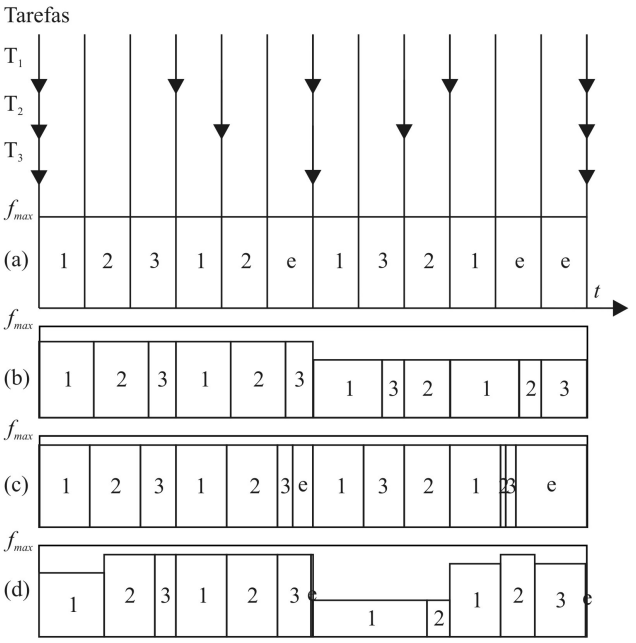

Fig. 2. Escalonamento do conjunto de tarefas I.

máximo. Uma vez que a tarefa termine sua execução pode-se observar quanto tempo ela gastou. Este *gain time* pode ser aproveitado pelo método de determinação folga. No momento, encontra-se em fase de implementação essa etapa do projeto, onde é considerada a variação do tempo de execução das tarefas e o conseqüente aproveitamento do *gain time*.

D. Exemplo de Aplicação do Método

Nessa seção será apresentado um exemplo de funcionamento do esquema de RDV com o uso do roubo de folga aqui proposto. No exemplo será assumido o mesmo conjunto de tarefas indicado na tabela 2. O processador aqui considerado é teórico e está definido na tabela 3. Nessa tabela, *Nível* indica os níveis que o processador possui, *F(MHz)* significa as freqüências de operação, *Volt* representa os níveis de voltagem correspondentes, *I(A)* é a corrente drenada, *Potência* é a potência consumida ($P = V * I$), *R.P.* é a potência relativa à potência máxima normalizada e *R.Ci* é o fator de extensão do tempo de execução relativo ao tempo à freqüência máxima. Esta tabela é utilizada para calcular a quantidade de energia gasta pelos diversos métodos.

Na figura 2(d) está indicado o escalonamento produzido na fase dinâmica, ou seja, com o uso da heurística gulosa. Na fase estática determina-se que a menor freqüência na qual se pode executar todo o conjunto de tarefas, garantindo todos os prazos, é a correspondente ao nível 9. Na fase dinâmica uma economia suplementar é obtida pois, após tentar baixar o nível de todo o sistema, ela trabalha com cada tarefa individualmente. Por isso, conforme indicado na figura, observa-se que a folga disponível é melhor utilizada, aumentando assim a economia de energia. A título de ilustração, na tabela 4, são mostrados os

Tabela 3
PROCESSADOR TEÓRICO

Nível	F(MHz)	Volt	I(A)	Potência	R.P.	R.Ci
1	100	0,1	0,1	0,01	0,01	10
2	200	0,2	0,2	0,04	0,04	5
3	300	0,3	0,3	0,09	0,09	3,333
4	400	0,4	0,4	0,16	0,16	2,5
5	500	0,5	0,5	0,25	0,25	2
6	600	0,6	0,6	0,36	0,36	1,667
7	700	0,7	0,7	0,49	0,49	1,429
8	800	0,8	0,8	0,64	0,64	1,25
9	900	0,9	0,9	0,81	0,81	1,111
10	1000	1	1	1	1	1

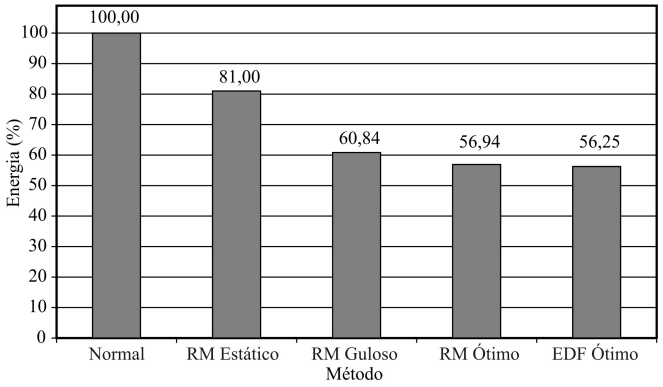

Fig. 3. Energia gasta pelo conjunto de tarefas I.

instantes de chaveamento, os tempos de execução e os níveis de freqüência atingidos (dentre aqueles da tabela 3) para cada uma das tarefas (de acordo com o indicado na figura2(d)). Na figura 3 é apresentada a quantidade normalizada de energia consumida por cada método (é assumido que o esquema sem RDV consome 100%).

Tabela 4
PROCESSADOR TEÓRICO: HIPERPERÍODO DO ESCALONAMENTO GULOSO PARA AS TAREFAS DO CONJUNTO I

Tempo	Tarefa	Tempo de Execução	Nível
0	1	1,429	7
1,429	2	1,111	9
2,54	3	0,46	9
3	1	1,111	9
4,111	2	1,111	9
5,222	3	0,732	8
5,954	vazio	0,046	8
6	1	2,5	4
8,5	2	0,5	8
9	1	1,111	9
10,111	2	0,75	8
10,861	3	1,111	9
11,972	vazio	0,028	9

Conforme pode ser notado, a fase dinâmica do método proporcionou uma melhoria substancial, de 20,16%, em relação à fase estática. Adicionalmente, esse resultado está somente a 3,9% do ótimo para o método RM. Isso é encorajador e aponta para a necessidade de experimento com outras heurísticas.

V. RESULTADOS PARA UM PROCESSADOR REAL

Nas simulações realizadas foram comparados os resultados dos algoritmos com o limite teórico obtido analiticamente. Para processadores teóricos ou reais, o cálculo do valor ótimo é feito admitindo-se que o processador possa assumir qualquer freqüência, e calcula-se então qual o valor de freqüência que leva a 100% de utilização do processador. Esta consideração é equivalente ao escalonamento com EDF utilizando um esquema contínuo de freqüências. Essa será a freqüência ótima para efeito de economia de energia.

Os conjuntos de tarefas aqui considerados serão aqueles definidos nas tabelas 2 e 5. O modelo de processador adotado é um modelo real, Intel Pentium M de 2,0GHz, com as características especificadas na tabela 6. Como no *datasheet* do processador não são disponibilizados os valores de corrente para todos os níveis de tensão, foi feita uma interpolação linear a partir dos valores mínimo e máximo fornecidos. Com estes valores é então calculada a potência e, conseqüentemente, a energia consumida.

Tabela 5
CONJUNTO DE TAREFAS II

Tarefa	WCET	Período
1	1	3
2	1	5
3	1	15

Tabela 6
PROCESSADOR PENTIUM M 2,0GHZ

Nível	F(MHz)	Volt	I(A)	Potência	R.P.	R.Ci
1	600	0,988	8,1	8,003	0,284	3,333
2	800	1,052	10,445	10,989	0,39	2,5
3	1000	1,1	12,205	13,425	0,477	2
4	1200	1,148	13,964	16,03	0,57	1,667
5	1400	1,196	15,723	18,804	0,668	1,429
6	1600	1,244	17,482	21,747	0,773	1,25
7	1800	1,292	19,241	24,859	0,883	1,111
8	2000	1,34	21	28,14	1	1

Na figura 4(b) é ilustrada a utilização da escala estática de freqüências para o conjunto de tarefas indicados na tabela 2. Neste exemplo, o escalonamento estático consegue baixar um único nível de freqüência e, portanto, o sistema é escalonado à freqüência de 1.800MHz. Operando nessa freqüência o conjunto de tarefas economizará 11,66% de energia. Na figura 4(c) é apresentada a mesma simulação empregando-se o método dinâmico guloso, que consegue economizar 26,32% de energia. Comparando esse resultado com o limite téorico para o RM verifica-se que são consumidos somente 1,49% a mais de energia. Na tabela 7 são mostrados os instantes de chaveamento, os tempos de execução e os níveis de freqüência atingidos (dentre aqueles da tabela 6) para cada uma das tarefas (de acordo com o indicado na 4(c)). Na figura 5 é indicada a quantidade normalizada de energia utilizada por cada método (o esquema sem RDV representa 100%).

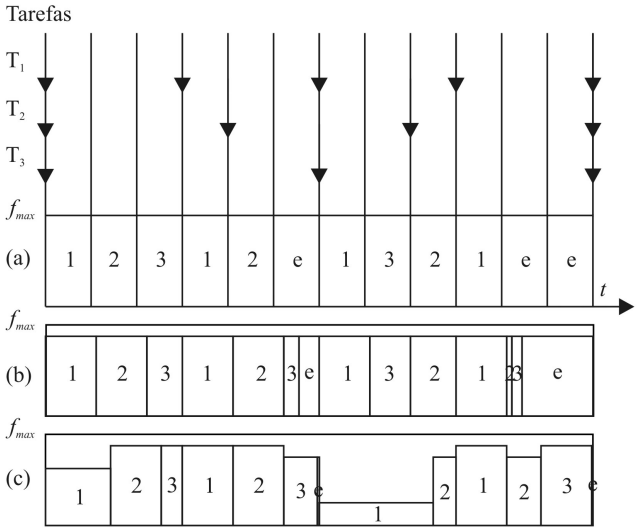

Fig. 4. Pentium M 2,0GHz: escalonamento do conjunto de tarefas I.

Tabela 7
PENTIUM M 2,0GHZ: HIPERPERÍODO DO ESCALONAMENTO GULOSO PARA AS TAREFAS DO CONJUNTO I

Tempo	Tarefa	Tempo de Execução	Nível
0	1	1,429	5
1,429	2	1,111	7
2,54	3	0,46	7
3	1	1,111	7
4,111	2	1,111	7
5,222	3	0,732	6
5,954	vazio	0,046	6
6	1	2,5	2
8,5	2	0,5	6
9	1	1,111	7
10,111	2	0,75	6
10,861	3	1,111	7
11,972	vazio	0,028	7

No segundo exemplo dessa seção foi utilizado o conjunto de tarefas da tabela 5. A figura 6 mostra o diagrama temporal de execução para um hiperperíodo. Na figura 6(b) é mostrada a solução para o escalonamento estático. Como há uma maior folga no sistema, consegue-se diminuir em dois níveis a freqüência de operação e, só nessa etapa, consegue-se uma economia de energia de 33,16%. Novamente, aplicando-se a heurística gulosa durante a fase de operação, conforme mostrado na figura 6(c), consegue-se refinar a solução e é atingido o valor de 43,35% de economia, que é somente 0,68% acima do limite teórico. Na tabela 8 são mostrados os instantes de chaveamento, os tempos de execução e os níveis de freqüência atingidos (dentre aqueles da tabela 6) para cada uma das tarefas (de acordo com o indicado na 6(c)). Na figura 7 é indicada a quantidade de energia utilizada por cada método.

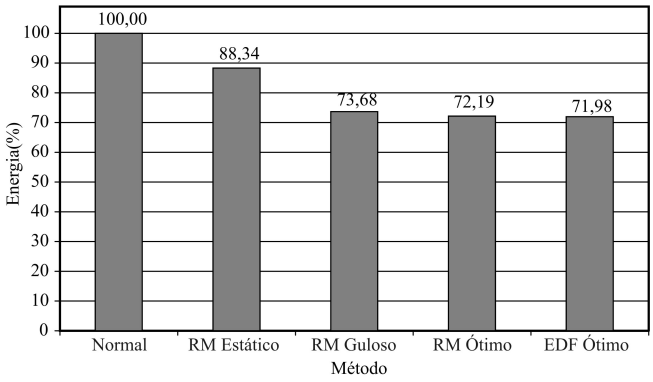

Fig. 5. Pentium M 2,0GHz: energia gasta pelo conjunto de tarefas I.

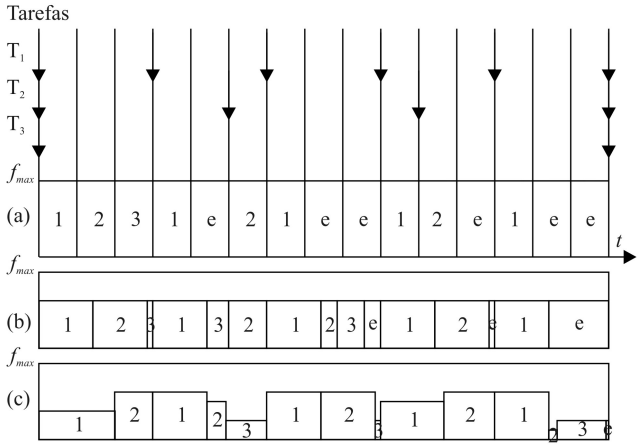

Fig. 6. Pentium M 2,0GHz: escalonamento do conjunto de tarefas II.

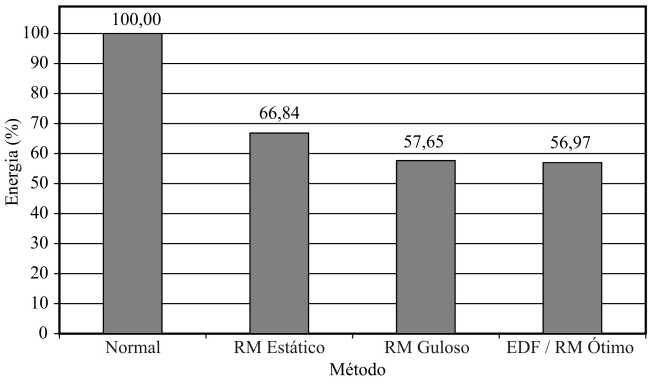

Fig. 7. Pentium M 2,0GHz: energia gasta pelo conjunto de tarefas II.

Tabela 8
PENTIUM M 2,0GHZ: HIPERPERÍODO DO ESCALONAMENTO GULOSO PARA AS TAREFAS DO CONJUNTO II

Tempo	Tarefa	Tempo de Execução	Nível
0	1	2	3
2	2	1	5
3	1	1,429	5
4,429	2	0,5	4
4,929	3	1,071	2
6	1	1,429	5
7,429	2	1,429	5
8,857	3	0,143	2
9	1	1,667	4
10,667	2	1,333	5
12	1	1,429	5
13,429	2	0,222	1
13,651	3	1,286	2
14,937	vazio	0,063	2

VI. TRABALHOS FUTUROS

O método aqui proposto representa uma melhoria em relação a outros trabalhos que usam a técnica de RDV e o algoritmo RM para sistemas móveis de tempo real. Foi mostrado que a técnica de roubo de ciclos (*slack-stealing*) em um sistema escalonado com RM pode atingir resultados próximos ao ótimo no que tange à economia de energia. Métodos baseados em EDF não podem utilizar esse tipo de técnica de forma *on-line* devido ao alto custo computacional. Em trabalhos futuros serão exploradas novas heurísticas para aproveitamento de folga que, em resultados preliminares já obtidos, demonstraram potencial para uma economia suplementar, atingindo valores próximos ao ótimo. Adicionalmente, está sendo também investigada a extensão da técnica aqui empregada para sistemas de tempo real não críticos.

VII. CONCLUSÃO

Dispositivos portáteis têm se tornado cada vez mais populares. Uma de suas principais características é a de depender de alimentação por bateria para a sua operação. Adicionalmente, em várias situações, o tipo de carga a ser processada apresenta requisitos de tempo real. Dessa forma, técnicas que permitem economia de energia vêm sendo atualmente pesquisadas. Um dos maiores consumidores de energia em sistemas como os aqui citados é o processador. Assim, nesse trabalho foi apresentado um algoritmo de escalonamento de tarefas de tempo real que procura ocupar os tempos livres da UCP, colocando-a a operar em um nível de voltagem (e também de freqüência de relógio) mais baixo, de forma a diminuir o consumo de energia. Os resultados preliminares apresentadas são promissores e situam-se próximos aos níveis teóricos ótimos.

AGRADECIMENTOS

Os autores agradecem ao CNPq e ao CONICET (Argentina) pelo financiamento parcial deste trabalho.

REFERÊNCIAS

[1] C. S. Ellis, “The Case for Higher-Level Power Management,” in *7th IEEE Workshop on Hot Topics in Operating Systems*, Rio Rico, Arizona, EUA, Março 1999, pp. 162–167.

[2] J. R. Lorch e A. J. Smith, “Apple Macintosh Energy Consumption,” *IEEE Micro*, vol. 18, no. 6, pp. 54–63, 1998.

[3] M. Weiser, B. Welch, A. Demers, e S. Shenker, “Scheduling for Reduced CPU Energy,” in *1st Symposium on Operating Systems Design and Implementation*, Monterey, California, EUA, Novembro 1994, pp. 13–23.

[4] L. Bertini e J. C. B. Leite, “Um Breve Survey: Escalonamento em Sistemas de Tempo Real com Otimização do Consumo de Energia,” in *VI Workshop de Tempo Real*, Gramado, RS, Brasil, Maio 2004, pp. 288–297.

[5] P. Pillai e K. G. Shin, “Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems,” in *18th Symposium on Operating Systems Principles*, Banff, Alberta, Canadá, Dezembro 2001, pp. 89–102.

[6] C. L. Liu e J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[7] W. Yuan e K. Nahrstedt, “Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems,” in *20th Symposium on Operating Systems Principles*, Bolton Landing, Nova York, EUA, Outubro 2003, pp. 149–163.

[8] J. P. Lehoczky e S. Ramos-Thuel, “An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems,” in *IEEE Real-Time Systems Symposium*, Phoenix, Arizona, EUA, Dezembro 1992, pp. 110–123.

[9] B. Sprunt, L. Sha, e J. P. Lehoczky, “Aperiodic Task Scheduling for Hard Real-Time Systems,” *The Journal of Real-Time Systems*, vol. 1, no. 1, pp. 27–60, 1989.

[10] J. K. Strosnider, J. P. Lehoczky, e L. Sha, “The Deferrable Server Algorithm for Enhanced Aperiodic Responsiveness in Hard Real-Time Environments,” *IEEE Trans. on Computers*, vol. 44, no. 1, pp. 73–91, 1995.

[11] H. Aydin, R. Melhem, D. Mossé, e P. Mejía-Alvarez, “Power-Aware Scheduling for Periodic Real-Time Tasks,” *IEEE Trans. on Computers*, vol. 53, no. 5, pp. 584–600, 2004.

[12] J. K. Dey, J. Kurose, e D. Towsley, “On-line Scheduling Policies for a Class of IRIS (Increasing Reward with Increasing Service) Real-Time Tasks,” *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 802–813, 1996.

[13] H. Aydin, R. G. Melhem, D. Mossé, e P. Mejía-Alvarez, “Optimal Reward-Based Scheduling for Periodic Real-Time Tasks,” *IEEE Transactions on Computers*, vol. 50, no. 1, pp. 111–130, 2001.

[14] R. M. Santos, J. Urriza, J. Santos, e J. Orozco, “Heuristic use of Singularities for On-Line Scheduling of Real-Time Mandatory/Reward-Based Optional Systems,” in *14th Euromicro Conference on Real-Time Systems*, Vienna, Áustria, Junho 2002, pp. 103–110.

[15] H. Aydin, P. Mejía-Alvarez, D. Mossé, e R. Melhem, “Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems,” in *22nd IEEE Real-Time Systems Symposium*, Londres, Reino Unido, Dezembro 2001, pp. 95–105.

[16] T. Pering e R. Brodersen, “Energy Efficient Voltage Scheduling for Real-Time Operating Systems,” in *4th IEEE Real-Time Technology and Applications Symposium*, Denver, Colorado, EUA, Junho 1998, work-in-progress session.

[17] J. Pouwelse, K. Langendoen, e H. Sips, “Energy Priority Scheduling for Variable Voltage Processors,” in *International Symposium on Low-Power Electronics and Design*, Huntington Beach, California, EUA, Agosto 2001, pp. 28–33.

[18] A. Sinha e A. Chandrakasan, “Energy Efficient Real-Time Scheduling,” in *International Conference on Computer-Aided Design*, San Jose, California, EUA, Novembro 2001, pp. 458–470.

[19] ——, “Dynamic Voltage Scheduling Using Adaptive Filtering of Workload Traces,” in *14th International Conference on VLSI Design*, Bangalore, India, Janeiro 2001, pp. 221–226.

[20] J. Luo e N. K. Jha, “Static and Dynamic Variable Voltage Scheduling Algorithms for Real-Time Heterogeneous Distributed Embedded Systems,” in *ASP-DAC/VLSI Design*, Washington, DC, EUA, Janeiro 2002, pp. 719–726.

[21] A. Miyoshi, C. Lefurgy, E. V. Hensbergen, R. Rajamony, e R. Rajkumar, “Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling,” in *16th ACM International Conference on Supercomputing*, Nova York, Nova York, EUA, Junho 2002, pp. 35–44.

[22] S. Saewong e R. Rajkumar, “Practical Voltage-Scaling for Fixed-Priority RT-Systems,” in *9th IEEE Real-Time and Embedded Technology and Applications Symposium*, Toronto, Canadá, Maio 2003, pp. 106–115.

[23] T. Ishihara e H. Yasuura, “Voltage Scheduling Problem for Dynamically Variable Voltage Processors,” in *International Symposium on Low-Power Electronics and Desing*, Monterey, California, EUA, Janeiro 1998, pp. 197–201.

[24] W. Kim, J. Kim, e S. Min, “A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis,” in *Conference on Design, Automation and Test in Europe*, Washington, DC, EUA, Março 2002, pp. 788–794.

[25] W. Kim, D. Shin, H. Yun, J. Kim, e S. Min, “Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems,” in *8th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Jose, California, EUA, Setembro 2002, pp. 219–228.

[26] Y. Shin, K. Choi, e T. Sakurai, “Power Optimization of Real-Time Embedded Systems on Variable Speed Processors,” in *International Conference on Computer-Aided Design*, San Jose, California, EUA, Novembro 2000, pp. 365–368.

[27] D. Mossé, H. Aydin, B. Childers, e R. Melhem, “Compiler-Assisted Dynamic Power-Aware Scheduling for Real-Time Applications,” in *Workshop on Compiler and OS for Low Power*, Filadélfia, EUA, Outubro 2000.

[28] O. S. Unsal e I. Koren, “System-Level Power-Aware Design Techniques in Real-Time Systems,” *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1055–1069, 2003.

[29] A. K.-L. Mok, “Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment,” Ph.D. dissertation, MIT, 1983.

[30] R. Obenza, “Rate Monotonic Analysis for Real-Time Systems,” *IEEE Computer*, vol. 26, no. 3, pp. 73–74, 1993.

[31] G. C. Buttazzo, “Rate Monotonic vs. EDF: Judgment Day,” in *3rd International Conference on Embedded Software*, Filadélfia, PA, EUA, Outubro 2003, pp. 67–83.

[32] J. M. Urriza e J. D. Orozco, “Métodos Rápidos para el Cálculo del Slack Stealing Exacto y Aproximado para Aplicaciones en Sistemas Embebidos,” Dep. de Ing. Eléctrica y Computadoras, Universidad Nacional del Sur, Bahía Blanca, Argentina,” Relatório Técnico, Agosto 2004. [Online]. Available: http://www.ingelec.uns.edu.ar