SECE-CPN: UM MODELO PARA A SIMULAÇÃO E ANÁLISE DE SISTEMAS COM CAPTURA DE ENERGIA

J.V. CASTELO MARTINS[1], C.G. FURTADO JÚNIOR[1], O. A. LIMA JÚNIOR[1]

[1]Laboratório de Sistemas Digitais – Instituto Federal de Educação, Ciência e Tecnologia do Ceará. Av. Parque Central, 1315 - Distrito Industrial I, Maracanaú - CE, 61939-140, Brasil.

E-mails: vitorcastelo.m@gmail.com, cjunior@ifce.edu.br, otavio.junior@gmail.com

Abstract – There is a need to create embedded system with greater energy efficiency, to achieve this goal these systems incorporated power management and energy harvesting techniques. In order to correctly size these kinds of systems, it is necessary models that allows the analyses of their hardware and software components. However, the current simulators have limitations leading to the creation of simplified scenarios that do not fit the reality. In this context, this article presents by Colored Petri Nets the SECE-CPN, a parametric model for representation and analysis of-embedded systems with energy harvesting. Two simulation scenarios were proposed and analyzed on different task sets. The scenarios seek determine whether simplifications can cause project errors. The results suggest that depending on the task set, the differences between the scenarios can cause mistaken analyses of the systems and the scheduling algorithms.

Keywords – Embedded systems with energy harvesting, Coloured Petri Nets, Scheduling algorithm

Resumo – Existe a necessidade de criação de sistemas embarcados com uma maior eficiência energética, para isso são incorporadas técnicas de gerenciamento e captura de energia. Para dimensionar corretamente estes tipos de sistemas é necessário modelos que permitam a análise de seus componentes de hardware e software, entretanto, os simuladores atuais possuem limitações que levam a criação de cenários simplificados que não condizem com a realidade. Neste contexto, este trabalho apresenta por Redes de Petri Coloridas o SECE-CPN, um modelo paramétrico para representação e análise de sistemas embarcados com captura de energia. Dois cenários de simulação foram propostos e analisados sobre diferentes conjuntos de tarefas. Os cenários buscam determinar se as simplificações podem causar erros de projetos. Os resultados sugerem que dependendo do conjunto de tarefas utilizados as divergências entre os cenários podem provocar análises equivocadas do sistema e do algoritmo de escalonamento.

Palavras-chave – Sistemas embarcados com captura de energia, Redes de Petri Coloridas, Algoritmo de escalonamento.

1 Introdução

O consumo de energia é um dos principais problemas enfrentados pelos sistemas embarcados. Com o surgimento das redes de sensores sem fio (RSSF) e o crescimento emergente da internet das coisas (do inglês IoT), surge uma necessidade de criar sistemas com maior eficiência energética.

Apesar do uso de técnicas avançadas de gerenciamento do consumo de energia como DPM (Dynamic Power Management) (Benini et al., 2000 e Chen et al., 2013) e DVFS (Dynamic Voltage Frequency Scaling) (Chen et al., 2013, Liu et al., 2012, Abbas et al., 2013 e Abbas et al. 2016), qualquer sistema embarcado irá eventualmente interromper sua operação devido à falta de energia. Para minimizar esse problema, técnicas de captura de energia (Energy Harvesting) podem ser utilizadas.

A captura de energia aparece como alternativa para fornecer à alimentação necessária para o nós sensores, principalmente quando a troca de baterias é custosa ou até mesmo inviável (Liu et al., 2012). Essa visa coletar a energia do ambiente em suas mais diversas formas (solar; eólica; cinética, dentre outras), e convertê-la em energia elétrica para alimentar um sistema.

Sistemas embarcados com captura de energia (SECE) possuem baixa escala de geração, microwatts à miliwatts, sendo desenvolvidos, portanto, como um método de substituição ou diminuição do tamanho das baterias (Kazmiersk e Beeby, 2011). Um SECE bem dimensionado pode operar sem interrupções e sem manutenção por longos períodos.

Para dimensionar corretamente os requisitos energéticos de um SECE são necessários modelos que permitam a modelagem e análise de seus componentes de hardware e software, tais como processador, DVFS, DPM, periféricos, escalonador de tarefas, dentre outros. A partir daí, é possível, dimensionar corretamente o sistema de captura de energia.

Existem vários simuladores de sistemas a evento discreto capazes de representar um SECE. Contudo estes simuladores apresentam limitações, como por exemplo, a incapacidade de mensurar o tempo e energia gastos na troca de contexto, podendo levar a simplificações no cenário de testes.

Este trabalho apresenta o SECE-CPN, um modelo paramétrico, modular e descrito em Rede de Petri Coloridas (RPC), cujo o objetivo é permitir a representação e análise de cenários de simulação de SECE. Além disso, o modelo proposto descreve a sobrecarga energética e temporal causada pelo uso das técnicas de gerenciamento de energia e escalonamento de tarefas.

Para a validação do SECE-CPN, foi reproduzido o trabalho apresentado em (Abbas et al. 2016), foi feito também um estudo comparativo com o objetivo de demonstrar possíveis falhas causadas por simplificações no cenário de teste. Os resultados são apresentados na seção V.

Este documento está estruturado da seguinte maneira: Após a introdução realizada nesta seção, os

trabalhos relacionados são apresentados na seção II. A seção III apresenta o modelo proposto. O cenário de teste é apresentado na seção IV. A seção V apresenta os resultados obtidos. Por fim, as considerações finais são apresentadas na seção VI.

2 Trabalhos Relacionado

OMNeT++ (OMNeT++, 2015) é um simulador de sistemas a eventos discretos, modular e com arquitetura genérica, podendo ser utilizado, por exemplo, na modelagem de protocolos diversos e na avaliação de desempenho de sistemas complexos. Os módulos são conectados entre si através de *gates* e podem ser combinados para compor novos componentes parametrizáveis. Embora o OMNeT++ disponha de uma interface de programação que permite ao usuário descrever o funcionamento de cada módulo, não é possível recuperar informações sobre demanda energética do sistema simulado. Dessa forma, o OMNeT++ não permite analisar o desempenho de algoritmos de escalonamento ou modelar técnicas de gerenciamento de energia (DVFS e DPM) por exemplo.

O *Jitterbug* (Lincoln e Cervin, 2002) é uma ferramenta baseada em Matlab que permite a computação de sistemas de controle lineares sobre diferentes condições de tempo. A ferramenta tem como objetivo identificar a sensibilidade de um sistema de controle por *delays*, *jitters*, perda de amostras e etc, sem recorrer a técnicas de simulações usando apenas análise estocásticas. A ferramenta é bastante genérica podendo ser usada também para análise de controles aperiódicos. Contudo, o custo dos sistemas de controle é calculado através de uma função quadrática que representa apenas o custo medido no caso-médio da performance do controle, sendo desprezadas o pior e melhor caso. Semelhante ao OMNeT++, o Jitterbug não permite analisar o desempenho de algoritmos de escalonamento.

O *TrueTime* (Cervin et al., 2013) é um simulador baseado em Matlab/Simulink para análise de sistemas em tempo real. Assim como no Jitterbug, no TrueTime os sistemas são representados através de dois blocos, o bloco *kernel* e o bloco *network*. O primeiro representa um computador com *kernel* de tempo real, enquanto o segundo simula as trocas de mensagens do sistema. O TrueTime possui diversas políticas de escalonamento (*earliest dead line first, fixed-priority, deadline monotonic)*, permite a criação de cenários reais com tarefas periódicas e aperiódicas, simula sistemas não lineares, representa técnicas de gerenciamento de energia e, de forma análoga ao OMNeT++, permite a definição de novas políticas de escalonamento e escalonadores. O TrueTime tem foco em sistemas de controle, podendo oferecer as seguintes métricas: energia disponível, *deadlines* perdidos e qualidade do controle, entretanto, as métricas percentual de ociosidade, tempo e energia perdidos em tarefas que não atingiram suas restrições temporais, tempo e energia gastos na troca de contexto do escalonador, tempo e energia gastos na troca de frequência do processador e energia gasta pelo processador em modo ocioso não estão disponíveis. As faltas destas métricas tornam difíceis a análise de alguns eventos do sistema (troca de contexto, troca de frequência, ociosidade da CPU) levando os usuários a produzirem cenários de testes simplificados, desconsiderando o tempo e energia gastos nesses eventos, podendo comprometer a análise do sistema.

Embora todos os trabalhos relacionados nesta seção possam ser utilizados para simulação de sistemas embarcados, apenas o TrueTime aborda os sistemas embarcados com captura de energia, porém, utilizando um conjunto de métricas que podem comprometer a análise do problema.

O SECE-CPN, simulador apresentado neste trabalho, busca expandir as características dos trabalhos apresentados nesta seção. Com o SECE-CPN é possível, assim como o TrueTime, representar sistemas não lineares, modelar sistemas de captura de energia e técnicas de gerenciamento (DPM e DVFS). Adicional aos trabalhos relacionados é possível mensurar tempo e energia gastos com troca de contexto, troca de frequência de operação e ociosidade da CPU, além de permitir a reconfiguração do sistema apenas modificando parâmetros de entrada, não sendo necessário a modificação estrutural da rede .

A semelhança do OMNeT++ e do TrueTime, o SECE-CPN, dispõe de uma interface para a extensão de suas funcionalidades. Através desta é possível, por exemplo, programar a função de consumo para o processador, o sistema de captura de energia, o escalonador, a bateria, dentre outras.

3 SECE-CPN

O SECE-CPN é uma arquitetura para modelagem e simulação de sistemas embarcados com captura de energia. A modelagem dos cenários de simulação do SECE-CPN é baseada no uso de Redes de Petri Coloridas (Jensen e Kristensen, 2009) , uma extensão das Redes de Petri (Petri Nets – PN). Uma PN permite a representação de aspectos importantes de sistemas computacionais, tais como: concorrência, controle, conflitos, sincronização e compartilhamento de recursos. Com uma PN, modelam-se aspectos estruturais de sistemas reais, utilizando formalismo matemático, definindo, de forma não ambígua, as entidades relacionadas e as suas interações (Murata, 1989). Adicionalmente, é possível executar simulações e verificar as propriedades do sistema antes mesmo de sua implementação (Cardoso e Valette, 1997).

Estendendo a capacidade das PN ordinárias, as Redes de Petri Coloridas (*Coloured Petri Nets* – CPN) podem representar tipos e estruturas de dados mais complexos, possuindo linguagem de programação associada: a CPN ML. Com isso, as CPN permitem a redução do tamanho do modelo, permitindo com que fichas com estruturas mais complexas (fichas coloridas) representem diferentes processos ou recursos em uma mesma subrede.

Utilizando o SECE-CPN é possível representar os requisitos temporais do sistema. A reconfiguração de cenários é feita apenas modificando parâmetros do simulador, não sendo necessário a modificação estrutural do modelo. Estão disponíveis ainda diferentes fontes de energia renováveis (solar, eólica, cinética, dentre outras), dois modelos de processadores XScale (Intel, 2017) e PXA270 (Marvell, 2007), 4 algoritmos de escalonamento, EA-DVFS (Abbas et al., 2013), FS-EH (Abbas et al., 2016), *Earliest Deadline First* (EDF) e Rate Monotonic Schedulling (RMS), e alguns conjuntos de tarefas (*task-sets*) predefinidos. Por fim, ao final de cada simulação as seguintes métricas são disponibilizadas: energia disponível, *deadlines* perdidos, percentual de ociosidade, tempo e energia perdidos em tarefas que não atingiram suas restrições temporais, tempo e energia gastos na troca de contexto do escalonador, tempo e energia gastos na troca de frequência do processador, energia gasta pelo processador em modo ocioso e deadlines perdidos, tempo médio de processamento e energia consumida por uma tarefa específica. Caso essas métricas não sejam suficientes para análise, novas métricas estatísticas podem ser programadas através da interface de extensões (IE) da ferramenta.

3.1 Hierarquia do SECE-CPN

O modelo CPN SECE-CPN é organizado em uma hierarquia de subredes (ou páginas). Uma subpágina é um modelo CPN, cuja semântica depende de outras páginas e/ou super-páginas (JENSEN; KRISTENSEN, 2009).

O SECE-CPN possui um total de 6 páginas, divididos em 3 níveis hierárquicos. A Figura 1 apresenta a página “SECE-CPN”, que representa o modelo de SECE em maior nível de abstração. A página “Configuration” representa a configuração do SECE que desejamos simular. A página “Task Generator” representa o gerador que injeta tarefas no modelo. A página “Energy Harvesting” representa o sistema de captura de energia. A página “Processor” representa o processador do sistema. Por fim, a página “Scheduler” representa o escalonador de tarefas.

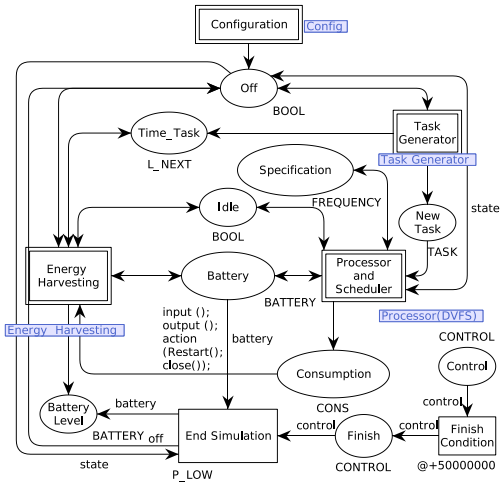


Figura 1. Página SECE-CPN

As subseções seguintes apresentam as páginas do modelo RPC do SECE-CPN.

3.2 Subpágina Configuration

A subpágina “Configuration”, apresentada pela Figura 2, permite a reconfiguração de cenários de simulação pela simples alteração da marcação inicial do modelo. Nela é possível, por exemplo, especificar o processador e definir os parâmetros do escalonador de tarefas. Além disso, é nesta página que ocorre a comunicação entre o SECE-CPN e a IE disponibilizada pelo modelo.

Os lugares *Open Connection* e *Connection Opened* armazenam fichas do tipo UNIT e representam uma conexão disponível e uma conexão realizada entre o modelo CPN e a IE, respectivamente. O disparo da transição *Connect* abre a comunicação entre modelo CPN e a IE.

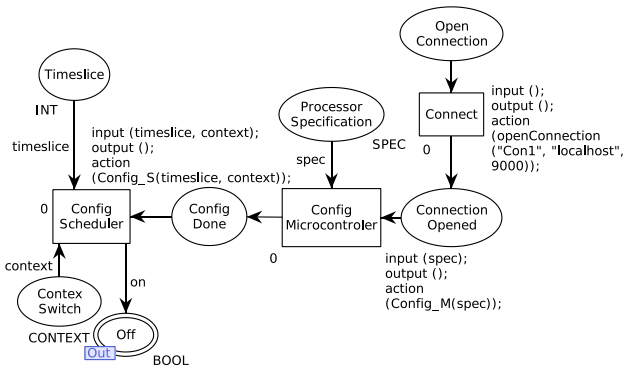


Figura 2. Subpágina Configuration

O lugar *Processor Specification* tem cor SPEC, que é composta pelo produto REAL * REAL * REAL * *list* INT e representam (i) o consumo do processador em modo ocioso, (ii) a *switch capacitancy* do processador, (iii) a relação energia/frequência do processador e uma lista de frequências (em MHZ) de operação disponíveis para o processador. A definição da cor SPEC permite modelar um processador utilizando a função de consumo proposta por Chen et al, 2013. Adicionalmente, é possível programar novas funções de consumo a partir da IE.

O disparo da transição *Config Processor* retira uma ficha do lugar *Processor Specification* e deposita uma ficha no lugar *Config Done* que é do tipo UNIT. Fichas depositadas nesse lugar representam a conclusão da configuração do processador utilizado no SECE.

O lugar *Timeslice* tem cor INT e representa a janela de tempo em nanosegundos (*ns*) que cada tarefa ocupa o processador.

O lugar *Context Switch* tem cor CONTEXT, que é composta pelo produto INT * REAL e representa, respectivamente, o tempo (*ns*) e energia (uJ) gastos na troca de contexto do escalonador. O disparo da transição *Config Scheduler* retira uma ficha dos lugares Config Done, TimeSlice e *Context Switch* e deposita uma ficha (de valor *false*) no lugar *Off*. Essa mudança de estado representa (i) a completa definição das características do processador e escalonador de

tarefas e (ii) a manutenção do SECE-CPN em estado "aguardando", enquanto outras definições são realizadas em outras subpáginas do modelo.

Como exemplo, fichas com valores (0.0, 1543.28, 2.87, [800,1000]), (1000), (10, 1.0) nos lugares "*Processor Specification"*, "TimeSlice" e "*Context Switch"* representam um processador capaz de operar em duas frequências (1000 e 800 MHz) e com um consumo em modo ocioso desconsiderado (0.0) para caráter de simulação, um escalonador de tarefas com janela de tempo de processamento de 1 ms, capaz de realizar a troca de contexto em 10 ns consumindo 1.0 uJ por troca.

3.3 Subpágina Task Generator

A subpágina *Task Generator* apresentada na Figura 3 é responsável pela definição do conjunto de tarefas realizadas pelo SECE simulado.

O lugar *FB_Task* recebe fichas de cor T_TASK e representa o conjunto de tarefas (do inglês *task-set)* do sistema. Fichas de cor T_TASK são compostas pelo produdo id_i : INT * WCET: INT * D_i: INT * T_i: INT* P_i: INT * flag_dvfs: BOOL e representam o identificador da tarefa, o pior caso de execução em ms, o deadline relativo da tarefa em ms, o período da tarefa em ms, a prioridade da tarefa e um indicador de que as técnicas de DVFS podem ser aplicadas a tarefa, respectivamente.

O disparo da transição *Register Task* retira uma ficha do lugar *FB_Task* e deposita uma ficha nos lugares *Tasks* e *Task_Registered*. Fichas nesses lugares representam uma tarefa pronta para ser injetada no sistema e uma lista de tarefas registradas pelo escalonador de tarefas, respectivamente. Deste modo, o escalonador possui conhecimento sobre o *task-set*.

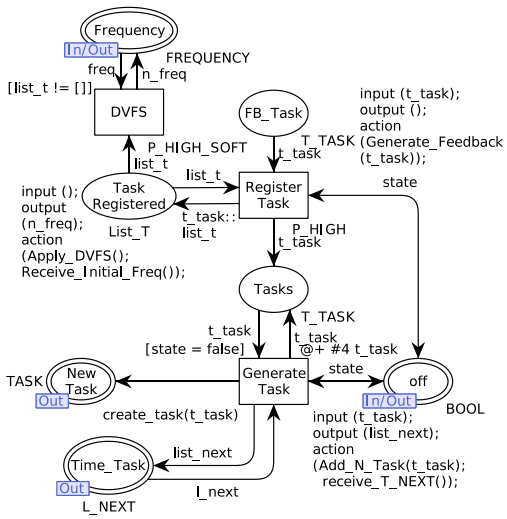


Figura 3. Subpágina Task Generator

Após o registro de todas as tarefas (cada tarefa é registrada pelo disparo da transição *Register Task*), a transição *DVFS* é habilitada. O disparo desta transição retira uma ficha do lugar *Task Registered* e deposita uma ficha do tipo FREQUENCY: INT no lugar "Frequency". Uma ficha nesse lugar representa a menor frequência possível que o sistema pode operar sem perder deadlines.

O lugar *Off* tem cor BOOL e indica o estado do sistema (*1`false* indica simulação em andamento, enquanto *1`true* indica que a simulação foi concluída). Há apenas duas condições de parada para a simulação: (i) foi atingido o tempo predefinido de simulação ou (ii) o nível de bateria ficou abaixo do necessário para o funcionamento do sistema.

Fichas no lugar *Tasks* representam as tarefas que chegaram ao sistema. O disparo da transição *Generate Task* deposita uma ficha de cor Task no lugar *New Task*. Fichas nesse lugar são compostas pelo produto T_Task * INT * INT e representam uma tarefa que pode ser executada pelo processador. Os atributos adicionais representam o tempo de chegada desta tarefa ao sistema (r_i) e o tempo de processamento da tarefa na CPU (p_i).

As fichas depositadas no lugar *Tasks* têm cor T_TASK e são temporizadas. Isso permite que a transição *Generate Task* siga as restrições temporais, podendo ser disparada novamente por uma mesma tarefa apenas no próximo período da tarefa.

Por fim, a ficha do lugar *Time_Task* é da cor L_NEXT. L_NEXT é composta por uma lista de elementos do tipo produto INT * INT. Estes atributos representam o identificador único (id) e o período de uma tarefa, e serve para determinar quando uma tarefa chega ao sistema novamente.

3.4 Subpágina Scheduler

O escalonador de tarefas do SECE-CPN é apresentado no modelo da Figura 4.

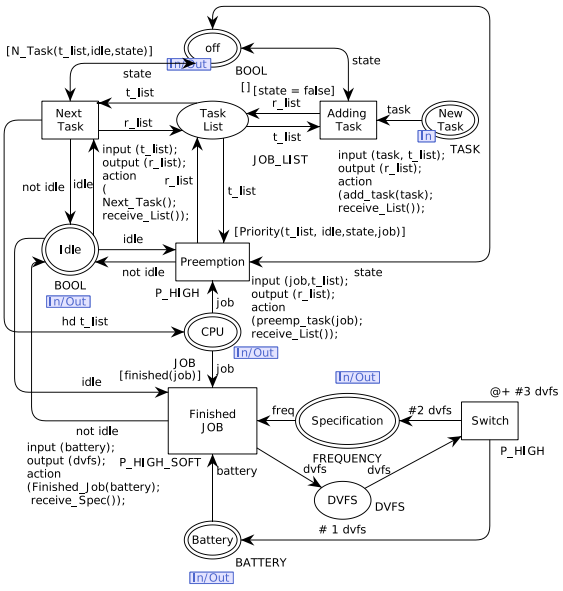


Figura 4. Subpágina Scheduler.

As tarefas geradas pela subpágina *Task Generator* são entregues diretamente para a subpágina *Scheduler* através do lugar *New Task*. O disparo da transição *Adding Task* retira uma ficha do lugar *New Task* e adiciona uma ficha do tipo lista (que representa um

conjunto de tarefas) no lugar *Task List*. O lugar *Idle* recebe fichas de cor BOOL e indica se a unidade central de processamento ou CPU (*Central Processing Unit*) está ocupada processando alguma outra tarefa (1`true indica que a CPU está ociosa, enquanto 1`false indica que alguma tarefa já está ocupando a CPU). Em caso de ociosidade da CPU, a transição *Next_Task* é habilitada. O disparo desta transição retira a tarefa de maior prioridade da lista de tarefas (fichas do lugar *Task List*) e alterna a CPU para o estado ocupado (em processamento).

Caso uma tarefa de maior prioridade chegue ao escalonador, este faz a preempção da tarefa que está na *CPU* pela de maior prioridade. Essa operação é representada pelo disparo da transição *Preemption.*

A transição Finished Job encontra-se habilitada quando a tarefa que está na *CPU* termina seu processamento ou perde seu *deadline*. O disparo da transição *Finished Job* deposita uma ficha de cor DVFS no lugar *Switch Frequency*. Fichas dessa cor são compostas pelo produto REAL * INT * INT, que representam a energia gasta pela troca de frequência do processador, a nova frequência de operação da CPU e o tempo gasto pela técnica de DVFS.

Os lugares *Specification* e *Battery* representam a frequência atual usada pelo processador e o nível de bateria do sistema. Por fim, o disparo da transição *Switch* atualiza o nível de bateria e informa a nova frequência a ser utilizada após o DVFS.

3.5 Subpágina Processador

O processador do SECE-CPN é apresentado no modelo da figura 5.

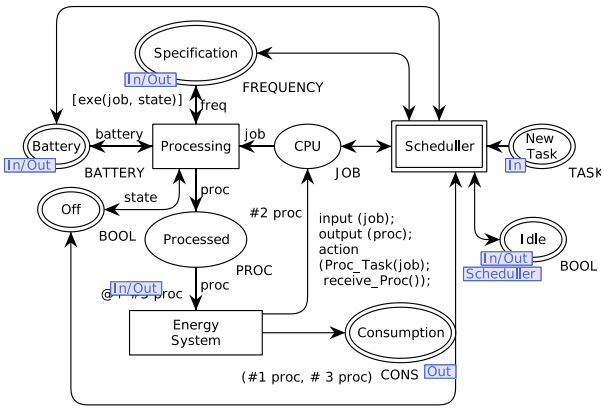


Figura 5. Módulo Processor

As tarefas escolhidas pelo escalonador para execução são entregues diretamente para a subpágina *Processor* através do lugar *CPU*. O disparo da transição *Processing* retira a ficha do lugar *CPU* e adiciona uma ficha de cor *PROC* no lugar *Processed.* Fichas neste lugar são compostas pelo produto REAL * JOB * INT e representam a energia gasta processando a tarefa, a tarefa que está sendo processada, e o tempo que esta tarefa utilizou a CPU em ms. Vale salientar que a energia gasta depende da frequência utilizada pelo processador. Após uma tarefa ser processada pela CPU a transição *Energy System* é habilitada. O disparo desta transição retira a ficha do lugar *Processed* e deposita fichas no lugar *Consumption* e *CPU*. O lugar *Consumption* recebe fichas do tipo *CONS e* representa a energia e o tempo gastos executando uma tarefa. O processador segue executando uma determinada tarefa até que a mesma termine seu processamento ou perca o deadline.

3.6 Módulo Energy Harvesting

O sistema de captura de energia do SECE-CPN é apresentado no modelo da figura 6.

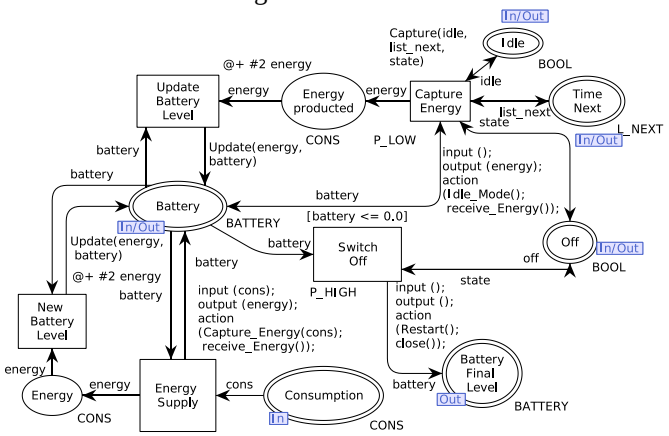


Figura 6. Módulo Energy Harvesting

A energia e tempo gastos ao processar uma tarefa, o estado da CPU (ociosa ou processando) e a lista de próximas tarefas são entregues diretamente para a subpágina *Energy Harvesting* através dos lugares *Consumption*, *Idle e Time Task* respectivamente. O disparo da transição *Energy Supply* retira fichas do lugar *Consumption* e deposita no lugar *Energy*. Fichas neste lugar representam a diferença entre a energia capturada pelo sistema e a energia consumida pelo processador e o tempo de processamento da tarefa. O disparo da transição *New Battery Level* atualiza o nível da bateria do sistema. Caso a energia capturada seja maior que a energia gasta processando a tarefa, a diferença é somada ao nível atual da bateria (Recarga), caso contrário, a diferença é subtraída (Descarga). A transição *Capture Energy* é habilitada caso a CPU esteja ociosa e não exista nenhuma tarefa no sistema. Esta transição deposita uma ficha do tipo *CONS* no lugar *Energy Producted* e representa a energia capturada pelo sistema enquanto a CPU está ociosa e o tempo de ociosidade. A transição *Update Battery Level* retira uma ficha do lugar *Energy producted* e adiciona a energia capturada ao nível atual da bateria recarregando o sistema. Caso a bateria do sistema atinja um limite mínimo para o sistema continuar funcionando a transição *Switch Off* é habilitada. Esta transição tem prioridade máxima e retira uma ficha dos lugares *Battery* e *Off* e deposita uma ficha (de valor *true*) no lugar *Off*. Esta mudança de estado representa o descarregamento total do sistema finalizando a simulação.

4 Validação do SECE-CPN

Para a validação do SECE-CPN foi reproduzido os experimentos feitos por Abbas et al., 2016 e seus resultados comparados.

Utilizamos o modelo de consumo proposto por Chen et al., 2013 de um processador XScale, cujo as especificações estão apresentadas na Tabela 2, a função de consumo é dada pela Equação (1).

$$1543{,}28\ x\ S^{2{,}28} + 63{,}28 \qquad (1)$$

Onde S corresponde a razão entre a frequência de operação atual e máxima.

Tabela 2. Parâmetros XScale. (Fonte Abbas et al. (2016) Modificado).

Voltage	0.75	1	1.3	1.6	1.8
Frequency (MHZ)	150	400	600	800	1000
Processor Speed (S_m)	0.15	0.4	0.6	0.8	1
Idle Consumption	63.58				
Power/Frequency Relationship	2.28				
Switching Capacitancy	1543.28				
Energy Switch (uJ)	1.2				
Time Switch (us)	12				

A energia necessária para alimentar o sistema é fornecida através de um conjunto de baterias recarregáveis que podem armazenar uma capacidade máxima de 2.5 Joules, utilizamos a captura de uma fonte de energia solar para fornecer energia extra as baterias. Para simular o comportamento da fonte de energia utilizamos o modelo apresentado na Equação 2.

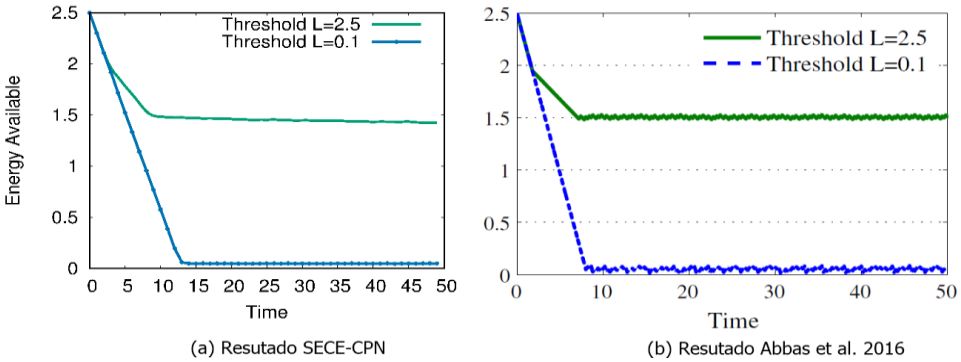
$$P_s(t) = |0{,}9xR(t)x\cos(t/0{,}7\pi)\ x\cos(t/0{,}7\pi)| \quad (2)$$

Onde $R(t)$ é uma distribuição uniforme randômica no intervalo [0,1]. Segundo Abbas et al. (2016), a função $P_s(t)$ é similar a coleta de energia feita por uma célula solar real.

Utilizamos um conjunto de quatro tarefas tal que, $\Gamma = \{\tau_i | 0 \leq i \leq 3\}$. A tarefa de identificador 0 é chamada de "*Feedback Task*", está tarefa é responsável por decidir e modificar a frequência de operação do processador e possui a maior prioridade entre o conjunto de tarefas Γ, devendo seguir um período $T_0 = 200\ ms$. Os períodos das demais tarefas do conjunto Γ são $T_1 = 19\ ms$, $T_2 = 20\ ms$ e $T_3 = 21\ ms$. O $WCET_i$ para toda tarefa τ_i é igual a 3 ms e o deadline relativo $D_i = T_i$, portanto, a tarefa deve terminar sua execução até a próxima instância da mesma ser lançada no sistema. O tempo de execução é variado e segue uma distribuição de Weibull que limita o tempo de execução da tarefa ao $WCET_i$.

O FS-EH é um algoritmo heurístico que visa a execução das tarefas com a máxima velocidade possível caso o nível de energia seja maior que um limiar (*Threshold*) estipulado. O algoritmo estipula o tempo de execução de cada tarefa do sistema, e utiliza esta estipulação para fazer um teste de escalonamento do sistema através da utilização instantânea da CPU ($U_{inst}(t)$). Caso $U_{inst}(t) \leq 1$ o conjunto de tarefas Γ pode ser escalonado, caso contrário, não existe tempo suficiente para executar todas as tarefas acarretando perda de deadlines. Se o nível de bateria $E(t)$ for maior do que *Threshold* (nível de bateria está acima do mínimo desejado), ou a utilização instantânea da *CPU* $U_{inst}(t) \geq 1$, o processador trabalha com frequência máxima visando diminuir o desperdício de energia causado pela capacidade finita de armazenamento da bateria e a perda de deadlines devido a impossibilidade de escalonar todas as tarefas. Caso $E(t) \leq Threshold$ e $U_{inst}(t) \leq 1$ o processador irá trabalhar com um fator de velocidade (S_m) mínimo tal que $\{S_m \geq \max(U_{inst}(t), E(t)/Threshold)\}$.

Considere $U_{inst}(t) = 0{,}5$, $Threshold = 2{,}5$ e $E(t) = 1$, logo, como $E(t) \leq Threshold$ e $U_{inst}(t) \leq 1$, o fator de velocidade utilizado será o maior valor entre $U_{inst}(t) = 0{,}5$ *e* $E(t)/Threshold = 0{,}4$, ou seja, 0,5. Contudo, o processador em questão não consegue operar utilizando este S_m (ver Tabela. 2), logo, o fator de velocidade escolhido será o menor S_m possível que seja maior que 0,5, ou seja, $S_m = 0{,}6$. A Figura 7 mostra o comparativo dos resultados apresentados por Abbas et al., 2016 e o SECE-CPN.



(a) Resutado SECE-CPN (b) Resutado Abbas et al. 2016

Figura 7. Validação do SECE-CPN

5 Resultados e Discussões

Visando demonstrar os problemas causados pelas simplificações nos cenários de testes, fizemos um estudo comparativo entre cenários simplificados e cenários mais próximos da realidade.

O ambiente de simulação utiliza o processador Intel XScale, uma bateria de 2.5 Joules, o algoritmo de escalonamento FS-EH com *Threshold* de 0.1 e uma fonte de energia solar (ver seção 4). Simulamos duas *task-sets* com tarefas semelhantes, porém, com quantidades diferentes, com o objetivo de aumentar a utilização instantânea da CPU ($U_{inst}(t)$).

Todas as tarefas possuem WCET = 3 ms e o deadline absoluto igual ao período da tarefa, o tempo de computação de cada tarefa varia entre 1 ms e o WCET. O primeiro *task-set* possui três tarefas com períodos iguais à 19 *ms*, 20 *ms* e 21 ms respectivamente, gerando uma utilização instantânea máxima de aproximadamente 45% da CPU, isto significa que, mesmo que as tarefas sempre operem no WCET, a CPU ficará ocupada apenas 45% do tempo. O segundo *task-set* é composto pelo primeiro mais uma tarefa de período igual à 22 *ms*, aumentando o $U_{inst}(t)$ para 58% da CPU.

Todos os *task-sets* foram simulados em dois cenários. Simplificamos o primeiro cenário desconsiderando o tempo e energia gastos com a troca de contexto e frequência do processador, assim como, a energia consumida pelo processador quando ocioso. No cenário 2 retiramos as simplificações. Segundo as especificações do XScale (ver tabela 2), o processador gasta 63.58 milliwatts/segundo quando ocioso, 1.2 micro Joules e 12 microssegundos para fazer a troca de frequência de operação. Consideramos a troca de contexto como uma sub-rotina que consome 1 micro Joule e demora 1 microssegundo para terminar. Por questões de confiança cada cenário foi simulado 33 vezes e os resultados são apresentados em forma de média, simulamos o referente a 50 segundos de processamento do sistema, não houve perda de deadline em nenhuma situação.

5.1 Utilização Instantânea máxima de 45%

Utilizando um limiar de 0.1, o FS-EH funciona com a potência máxima até o nível de bateria atingir o limiar, após isso, o algoritmo tenta escalonar as tarefas na menor frequência possível que não resulte em perda de deadlines. A Figura 8 apresenta a energia disponível para o primeiro *task-set*.

Os gráficos começam sobrepostos, até os 3 primeiros segundos da simulação, após este tempo é perceptível uma diferença entre os dois cenários, esta diferença é causada pelo alto nível de ociosidade da CPU. No modelo proposto (não simplificado) o nível de bateria atinge o limiar aos 12 segundos de simulação, neste momento o FS-EH utiliza a técnica de DVFS passando a consumir menos energia e diminuir a ociosidade da CPU. No modelo simplificado o nível de bateria atinge o limiar apenas aos 18 segundos de simulação. Como podemos ver no gráfico, após a utilização do DVFS o consumo do processador em modo ocioso volta a ser desprezível.

O somatório do tempo em que o processador ficou ocioso é de aproximadamente 34,62 segundos gastando uma energia total de 2,20 Joules representando 88% da capacidade máxima da bateria. A energia total gasta na troca de contexto e de frequência foi de 8,33 e 0,6 mJ respectivamente, estes valores representam aproximadamente 0,33% e 0,02% da capacidade máxima da bateria. Para este *task-set* é justificável não considerar a energia gasta pela troca de contexto e a troca de frequência de operação do processador. Entretanto, o consumo do processador quando ocioso não pode ser desprezado.

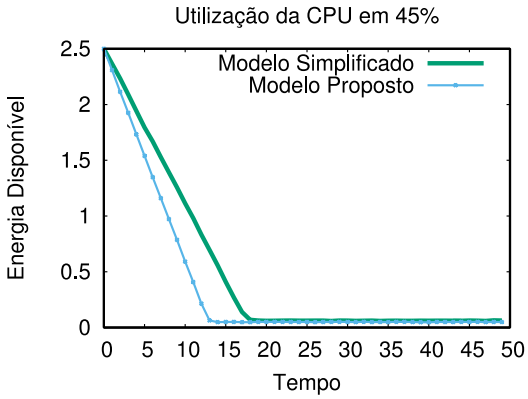


Figura 8. Energia disponível utilizando o *task-set* 1

5.3 Utilização Instantânea máxima de 58%

A Figura 9 apresenta a energia disponível para o último *task-set* analisado.

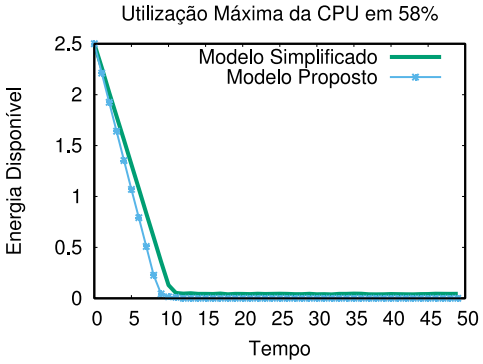


Figura 9. Energia disponível utilizando o *task-set* 2

Com o aumento da ocupação da CPU, o cenário de simulação 2 apresenta um erro menor se comparado ao cenário 1. Assim como no cenário anterior o erro causado pelo consumo em modo ocioso é desprezível no início da simulação, porém, aos 7 segundos, este erro começa a ser perceptível.

Enquanto no cenário anterior, o FS-EH encontrou o ponto de energia neutra (energia capturada >= energia consumida), O modelo proposto no cenário 2 não consegue estabilizar a energia do sistema levando a um descarregamento aos 12 segundos de simulação. Por outro lado, o modelo simplificado, consegue manter o sistema funcional durante os 50 segundos. Para este *task-set* o sistema opera em um estado crítico no qual qualquer gasto de energia deve ser considerado. O modelo simplificado induz a um erro de análise que pode levar a um dimensionamento equivocado da bateria do SECE. Além de uma análise equivocada do algoritmo de escalonamento.

O processador passou aproximadamente 7,22 segundos ocioso produzindo um gasto de energia de 0.45 Joules, este gasto equivale a 18% da capacidade total da bateria. A energia total gasta na troca de contexto e frequência é de aproximadamente 2.2 e 0.12 mJ respectivamente, estes valores representam 0,08% e 0,0048% da capacidade máxima da bateria. Vale salientar, que diferente do cenário 1, o sistema ficou operante por apenas 12 segundos. A troca de contexto não apresentou mudanças significativas. Porém, é importante enfatizar que os conjuntos de tarefas utilizadas possuem tempo de computação relativamente longo, portanto, simulando cenários nos quais o número de tarefas seja elevado e o tempo de computação curto a troca de contexto pode apresentar um gasto considerável.

6 Conclusão

Neste trabalho foi apresentado o SECE-CPN, um modelo paramétrico, descrito em Redes de Petri Coloridas para representação e análise de SECE reais.

Diferente dos outros trabalhos apresentados, utilizando o SECE-CPN é possível mensurar tempo e energia gastos com troca de contexto, troca de frequência de operação e ociosidade da CPU. Os usuários podem reconfigurar o SECE-CPN apenas modificando parâmetros de entrada, não sendo necessário a modificação estrutural da rede. O SECE-CPN possui recursos já implementados e prontos para utilização para auxiliar a criação de cenários mais condizentes com a realidade. Para validação do SECE-CPN reproduzimos o modelo proposto por Abbas et al, 2016, os resultados mostraram que o modelo RPC apresenta resultados compatíveis com a ferramenta de simulação TrueTime.

Para demonstrar os problemas causados pelas simplificações, foram descritos 2 cenários de teste. Foi detectado que dependendo do *task-set* utilizado as simplificações não são justificáveis podendo levar a uma análise equivocada tanto do sistema quanto do algoritmo de escalonamento.

Referências Bibliográficas

[1] Abbas, A., Grolleau, E., Loudini, M., and Mehdi, D (2013). A Real-Time Feedback Scheduler for Environmental Energy Harvesting. Proceedings of the 3rd IEEE International Conference on Systems and Control (ICSC), pp. 989–995.

[2] Abbas, A, Loudini, M, Grolleau, E, Mehdi, D and Hidouci, W (2016). A Real-Time Feedback Scheduler for Environmental Energy with Discrete Voltage/Frequency Modes. Computer Standards and Interfaces, pp. 264-273.

[3] Benini, L., Bogliolo, A. and Micheli, G (2000). "A Survey of Design Techniques for System-Level Dynamic Power Management", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 8, No. 3.

[4] Cervin, A (2003). Integrated Control and Real-Time Scheduling Ph.D. thesis Lund University.

[5] Cervin, A., Henriksson, D. and Ohlin, M. TrueTime: Reference Manual, URL http://www.control.lth.se/truetime. Acessado: novembro/2017.

[6] Chen, G., Huang, K., Huang, J., Buckl, C., and Knoll, A. (2013). Effective Online Power Management with Adaptive Interplay of DVS and DPM for Embedded Real-Time System. Proceedings of the 2013 Euromicro Conference on Digital System Design (DSD), pp. 881–889.

[7] Furtado Júnior, C.G., Soares, J.M, Barroso G.C. (2015). "CacheSIM: A Web Cache Simulator Tool Based on Coloured Petri Nets and Java Programming". Submitted to IEEE Latin America Transaction, Vol. 13, No. 5.

[8] Intel XScale Microarchitecture URL: http://developer.intel.com/design/intelxscale/2007. Acessado em: 08/02/2018.

[9] Kazmierski, T. Z. and Beeby, S. (2011). Energy Harvesting Systems: Principles, Modelling and Aplications, Spring Science + Business Media.

[10] Lincoln, B. and Cervin, A. (2002). Jitterbug: A tool for analysis of real-time control performance. In Proceedings of the 41st IEEE Conference on Decision and Control.

[11] Liu, S., Lu, J., Wu, Q. and Qiu, Q. (2012). Harvesting-Aware Power Management for Real-Time Systems with Renewable Energy, IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 20, No. 8, August 2012.

[12] Marvell PXA270 Processor. URL: http://www.marvell.com.cn/application-processors/pxa-family/assets/pxa_27x_emts.pdf. Acessado em: 08/02/2017.

[13] OMNeT++, OMNeT++ Simulation Manual version 5.2, URL https://omnetpp.org/doc/omnetpp/manual/#sec:simple-modules:signal-based-statistics. Acessado: novembro/2017.

[14] K. Jensen and L. M. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, 2009.

T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, vol. 77, no. 4, pp. 541–580, April 1989.

[15] J. Cardoso and R. Valette, Redes de Petri. Ed. Da UFSC, 1997.