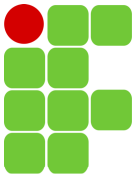


INSTITUTO FEDERAL DE EDUCAÇÃO,
CIÊNCIA E TECNOLOGIA DO CEARÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENERGIAS RENOVÁVEIS

João Victor Castelo Martins

Contribuições para a simulação de modelos de sistemas embarcados com captura de energia

Maracanaú, Ceará

2019

João Victor Castelo Martins

Contribuições para a simulação de modelos de sistemas embarcados com captura de energia

Dissertação apresentada ao curso de Mestrado Acadêmico em Energias Renováveis do Programa de Pós-Graduação em Energias Renováveis do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, campus Maracanaú, como requisito parcial para a obtenção do Título de Mestre em Energias Renováveis. Área de concentração: Energias Renováveis.

Orientador: Prof. Dr. Corneli Gomes Furtado Júnior
Coorientador: Prof. Dr. Otávio Alcântara de Lima Júnior

Maracanaú, Ceará

2019

Dados Internacionais de Catalogação na Publicação (CIP)
Ficha catalográfica elaborada pelo Bibliotecário
Gláucio Barreto de Lima - CRB-3/995

M386 Martins, João Victor Castelo.
Contribuições para a simulação de modelos de sistemas embarcados com captura de energia / João Victor Castelo Martins -- 2019.
82 f.

Dissertação (Mestrado em Energias Renováveis) -- Instituto Federal de Educação, Ciência e Tecnologia do Ceará, *Campus* Maracanaú, 2019.
Orientador: Prof. Dr. Corneli Gomes Furtado Júnior.
Coorientador: Prof. Dr. Otávio Alcântara de Lima Júnior.

1. ENERGIA RENOVÁVEL. 2. ENERGIA ELÉTRICA. 3. SISTEMAS EMBARCADOS. I. Título.
CDD 333.793 2

Sistema AutoCata(Desenvolvido por Cledson Oliveira)
SIBI/PROEN – Biblioteca Rachel de Queiroz – IFCE-Maracanaú

JOÃO VICTOR CASTELO MARTINS

CONTRIBUIÇÕES PARA A SIMULAÇÃO DE MODELOS DE SISTEMAS EMBARCADOS COM CAPTURA DE ENERGIA

Dissertação submetida à Coordenação do Curso de Pós-graduação em Energias Renováveis do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para a obtenção do título de Mestre em Energias Renováveis, área de concentração Energias Renováveis.

Aprovada em 28 / 08 / 2019.

BANCA EXAMINADORA

Prof. Dr. Corneli Gomes Furtado Júnior (Orientador)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Otávio Alcântara de Lima Júnior (Coorientador)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. José Renato de Brito Sousa
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Auzuir Ripardo de Alexandria
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Giovanni Cordeiro Barroso
Universidade Federal do Ceará - UFC

Agradecimentos

Agradeço primeiramente a Deus, o grande criado de tudo e todos.

A minha mãe, Lindalva, por acreditar em meu potencial e apoiar todas as minhas escolhas.

Ao meu pai, Epitácio, por me ajudar com seus conselhos engrandecedores.

A minha esposa, Katarina, seu amor, carinho e compreensão me transmitiram paz.

Ao Instituto Federal do Ceará (IFCE), por me propiciar um ambiente saudável, servindo como uma segunda casa, na qual pude crescer e aprender desde a graduação.

Ao meu orientador Prof. Corneli Gomes Furtado Júnior, por ter acreditado na minha capacidade e ter apostado nos meus interesses acadêmicos. Ao meu coorientador Prof. Otávio Alcântara de Lima Júnior, por todo a ajuda durante a escolha do tema dessa dissertação e por sempre almejar o sucesso deste trabalho.

Resumo

Os sistemas embarcados com captura de energia (SECE) são sistemas microprocessados capazes de converter energia do ambiente em energia elétrica. O projeto desses sistemas é complexo e envolve vários desafios, como por exemplo, o dimensionamento dos seus componentes de hardware, da sua bateria, do sistema de captura de energia, dentre outros. Como muitos desses sistemas ainda estão no âmbito hipotético, a simulação é a principal ferramenta utilizada para análise e dimensionamento de SECEs, entretanto, os simuladores atuais possuem limitações que levam a modelagens imprecisas desses sistemas. Outro problema está no nível de detalhes dado ao cenário de simulação. Cenários muito complexos podem acarretar em um gasto desnecessário de tempo e recursos. Enquanto que, cenários simplificados podem inserir erros de medição capazes de comprometer o projeto do sistema. Neste contexto, é necessário que o projetista de SECE saiba quais e quando os detalhes da simulação podem ser ignorados. O foco desta pesquisa é investigar o impacto energético em SECE causado por 3 simplificações comumente assumidas no cenário de teste (o uso de baterias ideais, troca de contexto do escalonador e o consumo ocioso do processador), além de identificar cenários onde essas simplificações podem ser aceitáveis. Ao longo da pesquisa foi implementado, testado e validado um modelo de simulação descrito em Redes de Petri Coloridas, chamado de SECE-CPN. Para demonstrar os problemas causados pelas simplificações no cenário de teste, foram comparados vários modelos de simulação com diferentes níveis de detalhes e conjuntos de tarefas. Os resultados mostram que a utilização de modelos de simulação com baterias ideais, assim como, desconsiderar o consumo ocioso do processador, podem levar a erros graves nos projetos destes sistemas. Nos casos testados, esses erros podem chegar a 28,08 % da capacidade máxima da bateria utilizada.

Palavras-Chaves: Sistemas embarcados, Sistemas embarcados com captura de energia, Captura de energia, Fontes renováveis de energia.

Abstract

The embedded systems with energy harvesting capabilities (ESEH) are microprocessed systems capable of converting environmental energy into electrical energy. The design of these systems is complex and involves many challenges, such as, sizing your hardware components, battery, the power capture system, among others. Since many of these systems are still hypothetical, simulation is the primary tool used for ESEH analysis. However, current simulators have limitations that lead to inaccurate modeling of these systems. Another problem is the level of detail given to the simulation scenario. Complex scenarios can result in unnecessary time and resource waste. While simplified scenarios can introduce measurement errors that can compromise system design. In this context, the ESEH designer needs to know which and when the details of the simulation can be ignored. The focus of this research is to investigate the energy impact on ESEH caused by 3 commonly assumed simplifications in the test scenario (the use of an ideal battery, the overflow caused by the context switch and the idle consumption of the processor), besides identify scenarios where such simplifications may be acceptable. Throughout the research was implemented, tested and validated a simulation model described in Colored Petri Nets, called SECE-CPN. To demonstrate the problems caused by simplifications in the test scenario, we compared various simulation models with different levels of detail and task sets. The results show that using ideal battery simulation models as well as disregarding idle processor consumption can lead to serious design errors in these systems. In the cases tested, these errors can reach 28.08 % of the maximum battery capacity used.

Key words: Embedded systems, Embedded systems with energy harvesting capabilities, Energy harvesting, Renewable energy source.

Lista de ilustrações

Figura 1 – Taxa de crescimento médio anual do fornecimento de energia renovável de 1990 a 2016, fonte International Energy Agency (2018) adaptado. . 14
Figura 2 – Comparativo da energia disponível em um sistema embarcado operando nos 3 estados de energia. 20
Figura 3 – Arquitetura tipica para SECEs 22
Figura 4 – Ganho energético alcançado com o uso de DVFS. 27
Figura 5 – Comparativo entre escalonamento de tarefas 28
Figura 6 – Escalonamento em conjunto com DVFS 29
Figura 7 – SECE operando em diferentes estados de energia 30
Figura 8 – Consumo energético de um mesmo sistema sob arquiteturas diferentes. 32
Figura 9 – Escalonamento segundo o algoritmo EDF 33
Figura 10 – Comparativo EDF X ALAP 34
Figura 11 – Estrutura Típica de uma RPC 36
Figura 12 – Arquitetura do CPN Tools 4. 38
Figura 13 – Arquitetura de Integração entre um modelo RPC e Interface de Programação Multilinguagem. 39
Figura 14 – Hierarquia do SECE-CPN 43
Figura 15 – Pagina SECE-CPN 44
Figura 16 – Subpágina Configuration 45
Figura 17 – Subpágina Task Generator 47
Figura 18 – Subpágina Scheduler 48
Figura 19 – Subpágina Processor 50
Figura 20 – Subpágina Energy Harvesting 51
Figura 21 – Comparativo entre o SECE-CPN e o TrueTime. 52
Figura 22 – Energia disponível utilizando o conjunto de tarefas 1 54
Figura 23 – Energia disponível utilizando o conjunto de tarefas 2 55
Figura 24 – Comparativo entre o modelo simplificado e o modelo 2. 63
Figura 25 – Erro de medição por segundo (Modelo 2). 64
Figura 26 – Consumo energético total do sistema (modelo 2). 65
Figura 27 – Comparativo entre o modelo simplificado e o modelo 3. 67
Figura 28 – Erro de medição por segundo (Modelo 3) 68
Figura 29 – Consumo energético total do sistema (modelo 3) 68
Figura 30 – Comparativo entre o modelo simplificado e o modelo 4. 71
Figura 31 – Comparativo entre o modelo simplificado e o modelo 4 utilizando os novos conjuntos de tarefas. 72
Figura 32 – Erro percentual por segundo (modelo 4) 73

Figura 33 – Consumo energético do sistema (modelo 4) 74
Figura 34 – Comparativo entre o modelo simplificado e o modelo 5. 75
Figura 35 – Erro percentual por segundo (modelo complexo) 75
Figura 36 – Consumo energético total do sistema (modelo 5) 76
Figura 37 – Energia negligenciada pelas simplificações em relação ao consumo total. 77

Lista de tabelas

Tabela 1 – Geração típica de captura de energia de várias fontes renováveis (Fonte Beeby e White (2010) adaptado). 22
Tabela 2 – Arquiteturas propostas 31
Tabela 3 – Características do conjunto de tarefas 33
Tabela 4 – Comparativo entre os simuladores 42
Tabela 5 – Parâmetros XScale. (Fonte Abbas et al. (2014) Modificado). 51
Tabela 6 – Energia negligenciada pelas simplificações do conjunto de tarefas 1 . . 54
Tabela 7 – Energia negligenciada pelas simplificações do conjunto de tarefas 2 . . 56
Tabela 8 – Trabalhos que utilizam simplificações no cenário de teste. 58
Tabela 9 – Modelos propostos e simplificações utilizadas 59
Tabela 10 – Configurações base do sistema 59
Tabela 11 – Conjuntos de tarefas sintéticos 62
Tabela 12 – Número de tarefas executadas por frequência de operação (Modelo 2) . 65
Tabela 13 – Número de tarefas executadas por frequência de operação (Modelo 3) . 69
Tabela 14 – Novos conjuntos de tarefas sintéticos 70
Tabela 15 – Número médio de tarefas executadas em 50 segundos 72

Lista de abreviaturas, siglas e símbolos

Abreviaturas

nó sensores Pequenos computadores sensores que compõem uma RSSF

id Identificador

SEMB Sistemas embarcados

task-set Conjunto de Tarefas

Simbolos

η Fator de eficiência da bateria

ηs nanosegundos

Γ Conjunto de tarefas

μJ microjoules

μW microwatts

ρ_i Tempo de processamento de uma tarefa na CPU

τ_i Tarefa de identificador i

$E(t)$ Nível da bateria do sistema no tempo t

$E_{\mathrm{EH}}(t_1, t_2)$ Energia capturada efetiva do sistema

$E_{\max}$ Capacidade máxima da bateria

$E_H(t_1, t_2)$ Energia capturada

$E_i(t_1, t_2)$ Consumo ocioso do processador no intervalo de tempo $[t_1, t_2]$

mw miliwatts

ms milissegundos

$P(t)$ Energia consumida pelo sistema no tempo t

P_s(t) Energia gerada pela fonte de energia renovável no tempo t

$R(t)$ Distribuição uniforme randômica no intervalo [0,1].

$U_{\max}$ Utilização máxima da CPU

$U_{inst}(t)$ Utilização instantânea da CPU

$WCET_i$ Pior caso de execução da tarefa de identificador i

D_i Deadline relativo de uma tarefa de identificador i

GW gigawatts

MHz Megahertz

r_i Tempo de chegada de uma tarefa no sistema

S_m Fator de velocidade do processador

T_i Período da tarefa de identificador i

Siglas

CMOS Complementary Metal Oxide Semiconductor

CPN Coloured Petri Nets

CPU Central Processing Unit

DPM Dynamic Power Management

DVFS Dynamic Voltage Frequency Scaling

EADVFS Energy Aware - Dynamic Voltage Frequency Scaling

EDF Earliest Deadline First

FS-EH FeedBack Scheduler with Energy Harvesting

GUI Sigla em inglês para Interface Gráfica de Usuário

IoT Sigla em inglês para Internet das Coisas

IPML Interface de Programação Multilinguagem

PC *Personal Computer*

PID Controlador Proporcional, Integrativo e Derivativo

PN Petri Nets

RF Radio Frequência

RMS Rate Monotonic Schedulling

RSSF Redes de Sensores Sem Fio

RTS Sistemas de tempo real

SCE Sistema de Captura de Energia

SECE Sistema Embarcado com Captura de Energia

SED Sistema a Eventos Discretos

TCP Transmission Control Protocol

TGE Técnica de Gerenciamento de Energia

WCET Worst Case Execution Time

Sumário

Lista de ilustrações **7**

Lista de tabelas **8**

1 INTRODUÇÃO **14**
1.1 Contribuições **15**
1.2 Justificativa **16**
1.3 Objetivos **16**
1.3.1 Objetivos Específicos 17
1.4 Produção Científica **17**
1.5 Organização do Texto **17**

2 FUNDAMENTAÇÃO TEÓRICA **18**
2.1 Princípio da Captura de Energia **18**
2.1.1 Estados de Energia 19
2.2 Sistemas Embarcados com Captura de Energia **21**
2.2.1 Arquitetura Típica de Sistemas Embarcados com Captura de Energia . 22
2.2.2 Sistema de Captura de Energia 23
2.2.3 Sistemas de tempo real 23
2.2.4 Tarefas 24
2.2.5 Processador 25
2.2.6 Técnicas de Gerenciamento de Energia 25
2.2.6.1 *Dynamic power management* 25
2.2.6.2 *Dynamic Voltage and Frequency Selection* 26
2.2.7 Escalonador de Tarefas 27

2.3 Algoritmos de escalonamento . . . 32
2.3.1 Políticas clássicas de escalonamento para RTS . . . 32
2.3.2 *Feedback Scheduler Energy Harvesting* . . . 34
2.4 Modelagem de SECE . . . 35
2.4.1 Redes de Petri Coloridas . . . 35
2.4.1.1 Definição formal das RPC . . . 36
2.4.2 CPN Tools . . . 37
2.4.2.1 Extensibilidade nativa ao CPN Tools . . . 38
2.4.2.2 Interface de Programação Multilinguagem . . . 38
2.5 Conclusão . . . 39

3 SECE-CPN . . . 40
3.1 Revisão Bibliográfica . . . 40
3.2 Arquitetura da rede . . . 42
3.2.1 Hierarquia do SECE-CPN . . . 43
3.2.2 Subpágina Configuration . . . 44
3.2.3 Subpágina Task Generator . . . 46
3.2.4 Subpágina Scheduler . . . 47
3.2.5 Subpágina Processador . . . 49
3.2.6 Subpágina Energy Harvesting . . . 49
3.3 Validação do SECE-CPN . . . 50
3.4 Resultados e Discussões . . . 52
3.4.1 Utilização instantânea máxima de 45 % . . . 53
3.4.2 Utilização instantânea máxima de 58 % . . . 54
3.4.3 Conclusão . . . 56

4 IMPACTO DAS SIMPLIFICAÇÕES NO RESULTADO DAS SIMULAÇÕES DE SECE . . . 57
4.1 Metodologia . . . 57
4.1.1 Detalhes da Simulação . . . 58
4.2 Modelo Simplificado . . . 59
4.2.1 Modelo Computacional . . . 59
4.2.2 Modelo de Consumo de Energia . . . 60
4.2.3 Modelo de Captura de Energia . . . 61
4.3 Cenário de Testes . . . 61
4.4 Resultados e Discussões . . . 62
4.4.1 Análise do modelo de bateria não ideal . . . 62
4.4.2 Análise do modelo de consumo ocioso . . . 66
4.4.3 Análise do modelo com troca de contexto . . . 69
4.4.4 Análise combinada das simplificações . . . 74

4.5 Conclusão 77

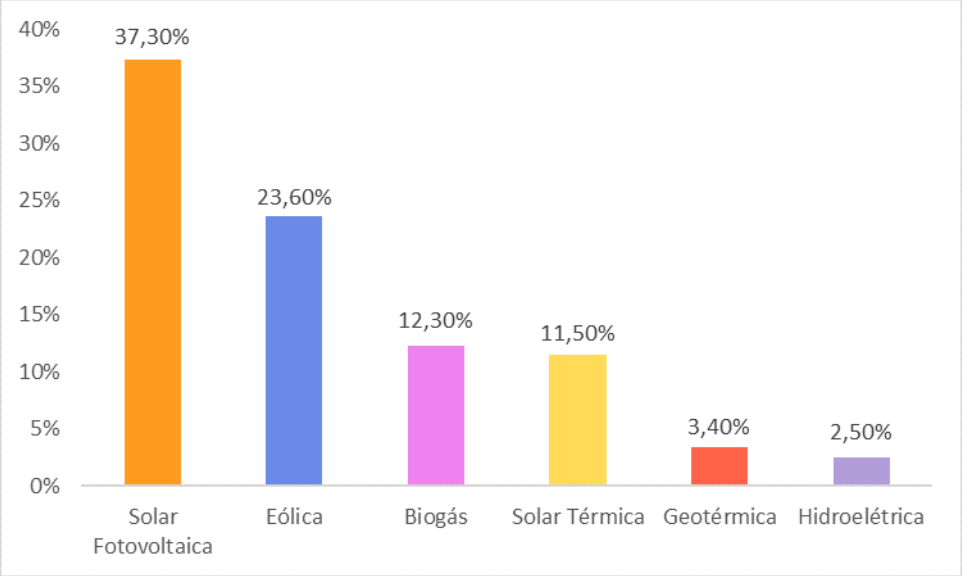
5 CONCLUSÕES E PERSPECTIVAS 79

REFERÊNCIAS 81

1 Introdução

As energias renováveis são obtidas por meio de fontes de energia que são regeneradas espontaneamente pela natureza. A geração de energia por meio de fontes renováveis está em ascensão. Enquanto que em 1990, 19,4 % da produção mundial de energia foram fornecidos através de fontes renováveis, em 2016 essa porcentagem subiu para 23,8 % (International Energy Agency, 2018). Na Figura 1 é apresentado a taxa de crescimento anual das fontes de energias renováveis no mundo de 1990 a 2016.

Figura 1 – Taxa de crescimento médio anual do fornecimento de energia renovável de 1990 a 2016, fonte International Energy Agency (2018) adaptado.



(fonte International Energy Agency (2018) adaptado.)

As energias fotovoltaica e eólica apresentam um crescimento acelerado com taxas de 37,3 % e 23,6 % respectivamente, em seguida aparece o biogás com 12,3 % e a energia térmica proveniente do Sol com 11,5 %. Por fim, temos a energia geotérmica e hidroelétrica com taxa de crescimento de 3,4 % e 2,5 % respectivamente. Esse crescimento emergente da geração de energia renovável denota uma mobilização global para geração de energia sustentável.

Existem tecnologias capazes de criar sistemas de grande porte aptos a gerar energia em grande quantidade utilizando fontes renováveis, como por exemplo, as usinas solares e os parques eólicos. Estes sistemas tem capacidade de geração na escala de gigawatts (GW) (IEA, 2018) (REN21, 2018). Contudo, usando os princípios da captura de energia renovável, é possível criar pequenos sistemas capazes de gerar energia em uma escala reduzida, a energia gerada por estes sistemas pode não ser suficiente para atender totalmente a

demanda energética de uma residência ou de sistemas computacionais de grande porte. Entretanto, pode-se utilizar esta energia na alimentação de sistemas computacionais microprocessados também conhecidos como sistemas embarcados (SEMB).

Os SEMB podem ser desenvolvidos para atuar em áreas diversas, como por exemplo, no mapeamento de florestas, parques, usinas, dentre outras, muitas vezes sendo impossível a alimentação desses tipos de sistemas por meio da rede elétrica. Portanto, o fornecimento de energia para os SEMB acontece geralmente por meio de baterias. Assim, a utilização da captura de energia surge como uma alternativa de alimentação do sistema, contudo, esta arquitetura traz grandes desafios aos projetistas de SEMB como por exemplo, dimensionar o sistema de captura e bateria, escolher e alinhar os componentes de *hardware* e *software* para trabalharem em conjunto gastando o mínimo de energia possível, dentre outros.

Os Sistemas Embarcados com Captura de Energia (SECE) devem ser projetados para utilizar fontes renováveis de energia como alimentação principal. Porém, esses sistemas precisam ser bem dimensionados para trabalhar com essas fontes. Por exemplo, um sistema que utilize energia solar deve possuir uma bateria suficiente para alimentá-lo durante o período noturno, ou ainda, um sistema que exija alto desempenho deve utilizar processador e sistema de captura capazes de atender as restrições temporais e energéticas. Como alguns SECEs ainda estão no âmbito hipotético, ou, apresentam elevados custos de projeto, a simulação torna-se a principal ferramenta para fazer este dimensionamento. Contudo, é vital a utilização de simuladores capazes de representar os componentes de *hardware* e *software* desses sistemas.

A criação do cenário de simulação é outro fator crucial para a análise do SECE. Caso a simulação superestime uma característica do sistema, como por exemplo, a troca de contexto entre as tarefas, tempo e recursos preciosos podem ser gastos em investigações futuras até descobrir que esta característica apresenta uma sobrecarga temporal e energética desprezível para a análise. Por outro lado, subestimar uma característica, como por exemplo o consumo ocioso do processador, pode introduzir erros na simulação que podem levar a conclusões equivocadas tanto do sistema quanto do algoritmo de escalonamento. Para evitar gastos desnecessários de tempo e recursos de projeto, assim como, evitar erros impostos por cenários de simulação simplificados, é vital para o projetista de SECE identificar quais fatores do ambiente e do sistema devem ser considerados no âmbito de simulação. Neste trabalho, são apresentados a metodologia e os resultados de um estudo sistemático que investiga como e em que grau adicionar ou remover detalhes do cenário de simulação impacta na análise do sistema.

1.1 Contribuições

As contribuições deste trabalho são listadas a seguir:

- Criação de um modelo de simulação paramétrico capaz de representar os componentes de *hardware* e *software* de SECEs.
- Avaliação do impacto das simplificações no resultado das simulações de SECEs.
- Identificação de cenários onde as simplificações são aceitáveis.

1.2 Justificativa

A eficiência energética é uma das principais preocupações dos projetistas de sistemas embarcados. As energias renováveis juntamente com a técnica de captura de energia surgem como alternativas de alimentação permitindo uma maior autonomia ao sistema. Contudo, são necessários modelos e ferramentas capazes de representar o comportamento desses sistemas de maneira mais próxima da realidade.

O conjunto de tarefas, assim como, a criação do cenário de simulação impacta na análise dos sistemas. Simplificações, abstrações e considerações muito otimistas podem levar a uma modelagem imprecisa podendo acarretar erros de medição, esses erros podem comprometer a análise da performance do SECE. O conjunto de tarefas representa os processos que serão executados pelo processador, esse conjunto é outra fonte de variação nos resultados da simulação. Conclusões obtidas com um determinado conjunto de tarefas 'A' podem não ser verdadeiras para um conjunto de tarefas 'B' e vice-versa. Enquanto a importância do conjunto de tarefas é geralmente entendida e cuidadosamente considerada na criação dos experimentos, os erros de medição provenientes da montagem do cenário de simulação recebe menos atenção. Na literatura, muitos autores, como por exemplo, (MOSER et al., 2007), (GHOR; CHETTO; CHEHADE, 2011), (LIU et al., 2012), (ABBAS et al., 2013), (ABBAS et al., 2014), (GHADAKSAZ; SAFARI, 2018) e (XU et al., 2018), utilizam cenários simplificados para análise de seus experimentos. Contudo, ao menos que os impactos desses cenários de testes simplificados sejam bem entendidos, os pesquisadores correm o risco de chegar a conclusões incorretas.

Embora alguns trabalhos, tais como, (LIU et al., 2012), (ABBAS et al., 2013) e (ABBAS et al., 2014), alertem o leitor sobre os possíveis erros que podem ser inseridos devido a simplificações no cenário de teste, não foram encontrados trabalhos que busquem medir o impacto desses erros, ou que busquem identificar cenários nos quais essas simplificações são aceitáveis.

1.3 Objetivos

O objetivo geral deste trabalho é analisar os impactos causados pelas simplificações no cenário de teste das simulações de SECE e orientar quando essas simplificações são

justificáveis.

1.3.1 Objetivos Específicos

Os objetivos específicos estão listados abaixo:

- Propor um modelo de simulação paramétrico capaz de representar um SECE.
- Identificar o impacto de uma simplificação específica em SECE.
- Identificar as simplificações que mais impactam no resultado das simulações.
- Identificar cenários onde as simplificações são justificáveis para a análise do sistema.
- Correlacionar parâmetros do conjunto de tarefas com o impacto de uma dada simplificação.

1.4 Produção Científica

Durante o desenvolvimento dessa pesquisa o seguinte trabalho científico foi produzido:

Artigo aceito:

J.V. CASTELO MARTINS, C.G. FURTADO JÚNIOR, O. A. LIMA JÚNIOR. SECE-CPN: UM MODELO PARA A SIMULAÇÃO E ANÁLISE DE SISTEMAS COM CAPTURA DE ENERGIA. **Congresso Brasileiro de Automática - CBA**, (2018).

Vale salientar que existem três trabalhos correlacionados com essa pesquisa que estão em processo de produção.

1.5 Organização do Texto

O texto dessa dissertação está organizado da seguinte forma: no Capítulo 2 são exibidas as fundamentações teóricas necessárias para o desenvolvimento da pesquisa; No Capítulo 3 é apresentado o modelo de simulação utilizado para criação do cenário de teste e análise dos resultados. No Capítulo 4 é exibido um estudo parcial sobre o impacto das simplificações no cenário de testes no resultado da simulação de SECEs. Por fim, as conclusões e perspectivas são exibidas no Capítulo 5.

2 Fundamentação Teórica

Neste capítulo é apresentado a fundamentação teórica necessária para entendimento da pesquisa.

2.1 Princípio da Captura de Energia

A captura de energia é uma técnica que consiste na conversão de energia presente no ambiente em energia elétrica. Esta técnica utiliza o mesmo princípio da geração de energia renovável em larga escala, porém, sua capacidade de geração tem uma escala reduzida (KAźMIERSKI; BEEBY, 2011). Enquanto na geração de energia renovável em larga escala é possível obter um potencial energético na casa de GW, a geração de energia produzida pela captura de energia está na escala de micro-milli watts (μW - mW).

A captura de energia é utilizada principalmente em dispositivos que não podem ser ligados a rede elétrica diretamente, para estes tipos de sistemas a alimentação é fornecida geralmente utilizando baterias. Contudo, baterias possuem uma capacidade finita de energia. Todo sistema computacional alimentado apenas por baterias eventualmente irá interromper sua operação devido ao descarregamento do sistema. Utilizando a técnica de captura de energia é possível converter energia do ambiente em energia elétrica e utilizá-la para recarregamento das baterias. Deste modo, o sistema computacional pode operar por longos períodos sem interrupção ou manutenção.

Os sistemas com capacidade de captura de energia apresentam várias novas restrições de projetos. Estes tipos de sistema precisam ser projetados e adaptados para trabalhar com fontes renováveis de energia. Diferentes fontes de energia podem implicar em diferentes restrições de projeto. Segundo Kansal et al. (2007) as fontes de energia renováveis podem ser classificadas nas seguintes categorias:

1. Incontrolável, porém previsível: A capacidade de geração de energia é impactada por fatores aleatórios externos ao sistema, contudo, o comportamento destes fatores podem ser modelados para prever a disponibilidade da fonte de energia dentro de uma margem de erro. Por exemplo, a energia solar não pode ser controlada, mas existem modelos capazes de prever o comportamento diurno e sazonal da fonte de energia.

2. Incontrolável e imprevisível: A capacidade de geração de energia é impactada por fatores aleatórios externos ao sistema, e a aleatoriedade desses fatores não é simples de prever utilizando modelos conhecidos, ou então, os modelos capazes de prever

o comportamento destes fatores são muito complexos para serem implementados em um sistema embarcado. Por exemplo, um sistema alimentado por fonte solar colocado em um ambiente *indoor* com tráfego de pessoas. Prever o comportamento da fonte de energia pode ser possível, porém, prever os padrões de sombreamento no sistema causado pelas pessoas que transitam a área pode ser impraticável.

3. Totalmente controlável: A fonte de energia pode ser controlada para gerar energia sempre que necessário. Por exemplo, um conjunto de lâmpadas, em que seja possível controlar a intensidade da luminosidade, pode ser utilizado para gerar energia a um sistema sempre que precisar.

4. Parcialmente controlável: A geração de energia pode ser influenciada pelos usuários, contudo, o comportamento da fonte não é totalmente determinístico. Por exemplo, uma fonte de energia por Radio Frequência (RF) pode ser instalada em um ambiente com vários nós sensores que se alimentam desta fonte. Contudo, a quantidade de energia gerada em cada nó depende da propagação do RF dentro do ambiente e não pode ser controlada.

É importante salientar que, uma mesma fonte de energia pode se enquadrar em mais de uma destas categorias dependendo do ambiente em que o sistema está submetido. Como por exemplo, nas fontes incontroláveis porém previsíveis e incontroláveis e imprevisíveis, a fonte de energia solar é utilizada para exemplificar as duas categorias, porém, o ambiente em que o sistema está inserido é crucial para a classificação.

2.1.1 Estados de Energia

O grande objetivo da captura de energia é levar o sistema a um estado neutro de energia (do inglês *Energy Neutral*). O estado neutro de energia é alcançado quando a energia capturada é igual à energia necessária para manter o sistema operando. Desse modo, o sistema se torna autossuficiente podendo operar de modo ininterrupto. A Equação 2.1 representa a condição neutra de energia em que $P_s(t)$ representa a energia capturada no tempo t e $P(t)$ representa a energia consumida pelo sistema.

$$\int_{t_1}^{t_2} P_s(t)dt = \int_{t_1}^{t_2} P(t)dt \tag{2.1}$$

Quando a energia consumida pelo sistema é maior que a energia capturada, tem-se um processo de descarregamento do sistema. Condições ambientais podem impactar negativamente na geração de energia levando a um estado de escassez energética. A submissão prolongada ao estado de escassez energética pode levar o sistema ao descarregamento total. A Equação 2.2 representa o estado de escassez.

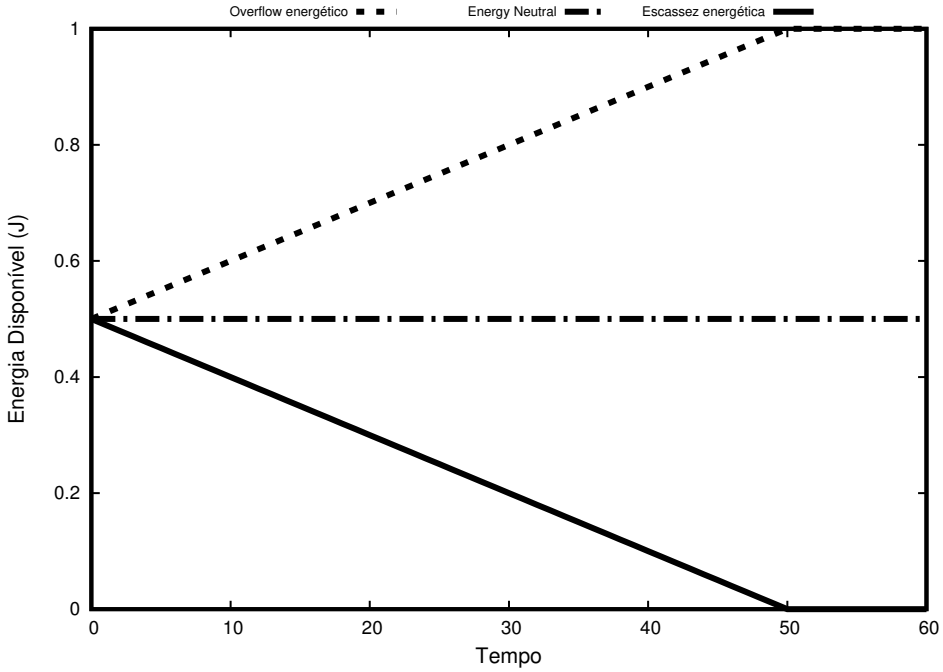
$$\int_{t_1}^{t_2} P_s(t)dt < \int_{t_1}^{t_2} P(t)dt \tag{2.2}$$

Quando a energia consumida pelo sistema é menor do que a energia capturada temos um processo de carregamento do sistema de armazenamento de energia. É importante um dispositivo de armazenamento de energia capaz de armazenar a energia excedente $\int_{t_1}^{t_2} P_s(t)dt - \int_{t_1}^{t_2} P(t)dt \mid P_s(t) > P(t) \ \forall \ t$. Sabendo que qualquer bateria possui uma capacidade finita de armazenamento, submeter o sistema a longos períodos de carregamento pode levar a um estado de *overflow* de energia. A Equação 2.3 representa a condição para o estado de *overflow*, $E(t)$ representa o nível de energia da bateria do sistema no tempo t e E_{max} representa a capacidade máxima da bateria.

$$E(t) + (\int_{t_1}^{t_2} P_s(t)dt - \int_{t_1}^{t_2} P(t)dt) > E_{max} \tag{2.3}$$

Vale salientar que um sistema computacional com captura de energia pode transitar por todos estes estados apresentados anteriormente. Na Figura 2 é mostrado a energia disponível em um sistema embarcado, durante 60 segundos operando sobre os 3 estados de energia apresentados anteriormente. O sistema possui uma bateria com $E_{max} = 1,0$ J e $E(0) = 0,5$ J.

Figura 2 – Comparativo da energia disponível em um sistema embarcado operando nos 3 estados de energia.



No estado neutro de energia a captura de energia consegue manter o nível da bateria constante durante os 60 segundos. Já no estado de escassez energética o consumo do sistema é sempre maior que a energia capturada no ambiente, deste modo, o nível da

bateria decresce levando ao descarregamento total do sistema aos 50 segundos. No estado de *Overflow* energético a energia capturada é maior do que o sistema pode utilizar, a energia excedente é armazenada levando a bateria a atingir sua capacidade máxima aos 50 segundos. Após a bateria atingir sua capacidade máxima, toda a energia capturada não utilizada pelo sistema é desperdiçada devido a incapacidade de armazenamento.

2.2 Sistemas Embarcados com Captura de Energia

Os sistemas embarcados (SEMB) estão presentes de forma ativa no cotidiano da sociedade. Sistemas embarcados são sistemas microprocessados em que o microprocessador está totalmente dedicado ao sistema, divergindo dos computadores pessoais (do inglês *Personal Computers (PCs)*), que possuem propósito geral podendo ser utilizados para resolver diversos problemas, os SEMB são sistemas usados para solucionar um problema específico. Este tipo de sistema está presente em vários equipamentos, tais como, relógios, forno micro-ondas, computador de bordo de automóveis e aeronaves, dentre outros.

Uma série de restrições é levantada ao se projetar um SEMB. O projeto deste tipo de sistema pode envolver conceitos pouco explorados pela computação de propósito geral, tais como, portabilidade, baixa disponibilidade de memória, necessidade de segurança e confiabilidade, consumo energético reduzido sem perda de desempenho e tempo curto de projeto. Estes conceitos tornam o desenvolvimento de SEMBs uma área de conhecimento específica (WOLF, 2008).

O projeto de Sistemas Embarcados com Captura de Energia (SECE) é mais complexo do que o projeto de SEMB convencionais. Os SECEs são sistemas embarcados com capacidade de utilizar a técnica de captura de energia para converter energia do ambiente em energia elétrica, esta energia convertida é utilizada na alimentação do próprio sistema. O projeto de SECEs traz consigo todos os desafios do projeto de SEMBs convencionais, contudo, estes sistemas precisam ser pensados e adaptados para trabalhar com fontes de energia renováveis implicando em um grande leque de novos desafios como, por exemplo, projetar e dimensionar o sistema de captura de energia, dimensionar a bateria, adaptar o comportamento do sistema aos diferentes estados de energia, dentre outras.

Os principais problemas em utilizar a captura de energia para alimentação de sistemas são: a disponibilidade da fonte de energia e a baixa capacidade de geração. Fontes de energia renováveis possuem comportamento intermitente, variando a sua intensidade ao longo do dia. Portanto, a geração de energia por meio destes tipos de fontes não é constante. Além disso, a maioria dos SEMBs apresentam restrições de peso e tamanho, o que impossibilita a criação de sistemas de captura com alta capacidade de geração. Na Tabela 1 é fornecida uma indicação de potências típicas geradas na captura de energia, juntamente com algumas condições assumidas.

Tabela 1 – Geração típica de captura de energia de várias fontes renováveis (Fonte Beeby e White (2010) adaptado).

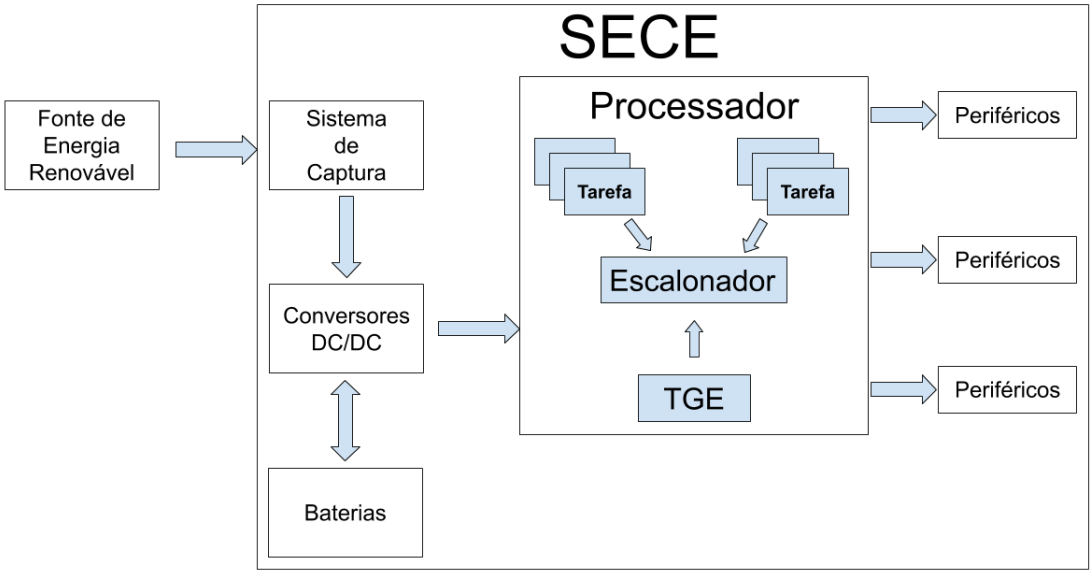
Tipo de Energia	Condições	Energia Gerada	Área ou Volume	Energia/Dia
Vibração	1 m/s^2	100 μW/cm^3	1 cm^3	8,64 J (Assumindo vibração contínua)
Solar	Ambiente externo	7500 μW/cm^2	1 cm^2	324 J (Assumindo luz 50 % do tempo)
Solar	Ambiente interno	100 μW/cm^2	1 cm^2	4,32 J (Assumindo luz 50 % do tempo)
Térmica	ΔT = 5 °C	60 μW/cm^2	1 cm^2	2,59 J (Assumindo calor 50 % do tempo)

Como pode-se observar na Tabela 1, a energia solar em condição externa possui a maior capacidade de geração dentre as fontes de energia apresentadas. Contudo, quando o sistema é inserido em um ambiente interno, a geração de energia se compara às demais fontes.

2.2.1 Arquitetura Típica de Sistemas Embarcados com Captura de Energia

Nesta seção é apresentado a arquitetura tipica de SECEs, assim como, seus componentes de *hardware* e *software*. Na Figura 3 é apresentado uma possível arquitetura de SECEs.

Figura 3 – Arquitetura tipica para SECEs



O sistema de captura de energia (SCE) é responsável por converter energia do ambiente em energia elétrica para alimentar o SEMB. Essa energia convertida é regulada através de conversores DC/DC para ser utilizada pelo processador e os demais periféricos. Caso a energia convertida não seja suficiente, o SEMB pode retirar energia da bateria para manter seu funcionamento normal, caso o SEMB não consiga utilizar toda energia

convertida, o excesso de energia pode ser armazenado na bateria para utilização futura. Os SECEs podem ainda fazer uso de Técnicas de Gerenciamento de Energia (TGE) para uma maior adaptação do escalonador no processo de escalonamento de tarefas.

2.2.2 Sistema de Captura de Energia

O sistema de captura de energia é um componente crucial de um SECE. O SCE é uma das diferenças entre os SEMBs convencionais e os SECEs, e tem como objetivo transformar a energia de uma fonte renovável em energia elétrica para ser utilizada pelo sistema. O projeto de um SCE depende da fonte de energia renovável que será utilizada.

Escolher a fonte de energia mais adequada para o SECE é um fator crucial no projeto desses tipos de sistemas. Existem vários ambientes submetidos a mais de uma fonte renovável. Por exemplo, considere um SECE implantado em uma floresta tropical pluvial para monitorar os níveis de poluentes de um rio. Devido à densidade da vegetação presente neste tipo de floresta, a energia solar pode não ser a melhor escolha de fonte renovável para alimentação do sistema. Neste caso, pode ser mais interessante utilizar a energia cinética produzida pelo próprio rio. A escolha da fonte de energia vai impactar no projeto do sistema de captura.

O projeto do SCE está relacionado ao tipo e intensidade da fonte de energia que será utilizada. No caso de uma fonte de energia piezoelétrica, a energia gerada é proporcional à pressão inserida no sistema de captura de energia, e dependendo da aplicação, a quantidade de energia que pode ser gerada e o projeto do sistema de captura podem divergir bastante. Por exemplo, um sistema de captura projetado para converter energia a partir da pressão imposta por um corpo humano, tem uma solução e uma capacidade de geração totalmente diferente de um sistema projetado para gerar energia por meio da pressão imposta por uma máquina.

2.2.3 Sistemas de tempo real

Sistemas de tempo real (do inglês, *Real Time Systems* (RTS)) são sistemas computacionais nos quais o seu correto funcionamento depende não somente do resultado lógico da computação, como também da obtenção desses resultados no momento certo. Os RTS são divididos em duas categorias: Não tolerantes e tolerantes a falha. Nos RTS não tolerantes a falhas, o descumprimento de alguma restrição temporal leva o sistema a um estado de erro irreparável. Já nos RTS tolerantes a falhas, descumprir restrições temporais leva o sistema a um estado degradado de performance. Contudo, é possível para o sistema se recuperar desse estado posteriormente desde que não descumpra restrições sucessivamente. Este trabalho tem foco em RTS tolerantes a falhas.

2.2.4 Tarefas

Tarefas são partes de um programa em alguma fase de execução. É possível classificar as tarefas em duas categorias: Dependentes e independentes. Tarefas dependentes necessitam de informações ou recursos provenientes da computação de outras tarefas (tarefas de pré-requisito), enquanto que, as tarefas independentes podem ser executadas assim que estejam prontas no sistema.

Algumas tarefas possuem restrição de tempo para serem computadas, estes tipos de tarefas são chamadas de tarefas de tempo real. Perder o prazo das tarefas (*deadlines*), tem um impacto negativo no desempenho, além de forçar o sistema a trabalhar com informações defasadas podendo levar a erros. Alguns sistemas são capazes de se recuperar destes erros desde que as tarefas não percam *deadlines* sucessivamente. Contudo, a perda excessiva de *deadlines* pode levar o sistema a um estado de erro irreversível.

As tarefas de tempo real são classicamente representadas por um conjunto de parâmetros de temporização (LIU; LAYLAND, 1973). É vasto o número de trabalhos na literatura que utilizam conjuntos de tarefas sintéticos para simulação ((MOSER et al., 2007), (LIU; QIU; WU, 2008), (Jun Lu et al., 2010), (LIU et al., 2012), (ABBAS et al., 2013), (ABBAS et al., 2014), (Dang et al., 2016), (GHADAKSAZ; SAFARI, 2018) e etc). Neste trabalho é assumido um conjunto de tarefas independentes, esta consideração visa a reduzir a complexidade da simulação. As tarefas são definidas pelos parâmetros a seguir:

- r_{i} representa o tempo de lançamento da tarefa τ_{i} no sistema.
- T_{i} representa o período da tarefa
- $WCET_{\mathrm{i}}$ é o pior tempo de execução da tarefa na velocidade total do processador.
- $WCET_{\mathrm{i},\alpha}$ é o pior tempo de execução da tarefa na velocidade atual do processador.
- c_{i} é o tempo de execução da tarefa para concluir seu processamento, quando a mesma é executada na frequência máxima do processador.
- $c_{\mathrm{i},\alpha}$ é o tempo de execução da tarefa para concluir seu processamento na velocidade atual do processador.
- D_{i} é o *deadline* absoluto de uma tarefa dada por $r_{\mathrm{i}} + T_{\mathrm{i}}$.

Note que neste modelo, o *deadline* absoluto é calculado a partir do tempo de lançamento e o período da tarefa. Esta definição vem da suposição de que todas as tarefas devem concluir seu processamento antes que uma nova instância da tarefa reapareça no sistema.

2.2.5 Processador

O processador é um circuito integrado cuja finalidade é efetuar o processamento de dados através de um conjunto de instruções pré-definidas. Utilizar processadores no projeto de SECEs traz inúmeras vantagens como:

- Diminuição do custo do projeto: A popularidade deste tipo de componente permite que seu custo seja dividido entre vários clientes.
- Facilidade de atualização do sistema: Novas versões de um sistema podem ser feitas por meio da atualização do software da aplicação.
- Mão de obra: Geralmente os engenheiros possuem conhecimentos de programação para microprocessadores.

Contudo, os processadores também trazem desvantagens aos SECEs. Dependendo da aplicação, algumas estruturas de *hardware* presentes no processador podem ser sub-utilizadas, mas mesmo assim, estas estruturas podem consumir muitos recursos (CARRO, 2003). Como a eficiência energética é um problema crucial enfrentado pelos SEMBs em geral, existe a necessidade de utilizar processadores de baixa potência (*Low-Power*).

Processadores de baixa potência trabalham em uma faixa de frequência reduzida. A frequência de operação está diretamente ligada ao desempenho do processador, frequências maiores implicam em uma maior velocidade do processador, consequentemente um maior consumo energético, enquanto que frequências menores implicam em uma menor velocidade e menor consumo. Portanto, dependendo da aplicação, a família de processadores que trabalhe com frequências mais baixas pode não ter desempenho suficiente para atender aos *deadlines* das tarefas.

2.2.6 Técnicas de Gerenciamento de Energia

A captura de energia por si só, não garante o funcionamento do sistema de maneira ininterrupta. A técnica de captura de energia apresenta uma baixa geração de energia, portanto, essa energia precisa ser utilizada de maneira eficiente pelo SECE. Existem técnicas de gerenciamento de energia que tem como objetivo reduzir a dissipação energética do sistema. Segundo Chetto (2014) *Dynamic Power Management* (DPM) e *Dynamic Voltage and Frequency Selection* (DVFS) são as técnicas convencionais utilizadas para reduzir a dissipação energética.

2.2.6.1 *Dynamic power management*

O DPM consiste em desligar, ou colocar em um modo de baixo consumo, as estruturas de hardware que estão ociosas no sistema. Por exemplo, a troca de informação

de uma Rede de Sensores Sem Fio (RSSF) é a operação que mais consome energia do nó sensor, os rádios utilizados para tal operação têm um consumo significativo mesmo sem estar recebendo ou enviando informação. Portanto, enquanto não existe a necessidade de uso do rádio, é interessante para o sistema mantê-lo desligado, porém, alguns componentes devem se manter operantes mesmo que estejam ociosos. O processador guarda algumas informações em memórias voláteis, as informações salvas neste tipo de memória são completamente apagadas quando o processador é desligado. Portanto, desligar o processador é uma operação inviável para o sistema. Contudo, alguns processadores possuem controle de potência embutidos permitindo o desligamento de partes do processador levando a um estado de consumo reduzido chamado de *Sleep mode*. Neste estado o processador consome energia apenas para se manter funcional.

2.2.6.2 *Dynamic Voltage and Frequency Selection*

O DVFS é uma técnica de gerenciamento de energia que visa a diminuir a dissipação energética do sistema por meio da mudança dinâmica da frequência e tensão do processador durante a execução das tarefas. Existem processadores capazes de utilizar diferentes frequências e em casos nos quais a concorrência pelo processador é baixa e o mesmo se encontra muito tempo ocioso, diminuir a frequência de operação é uma forma de garantir um consumo energético eficiente e aumentar a utilização do processador. Por outro lado, quando o desempenho do sistema se torna crucial, aumentar a frequência de operação força o processador a trabalhar mais rápido aumentando o desempenho. Deste modo, é possível utilizar famílias de processadores que possuam desempenho elevado e ainda assim atender as restrições energéticas do sistema.

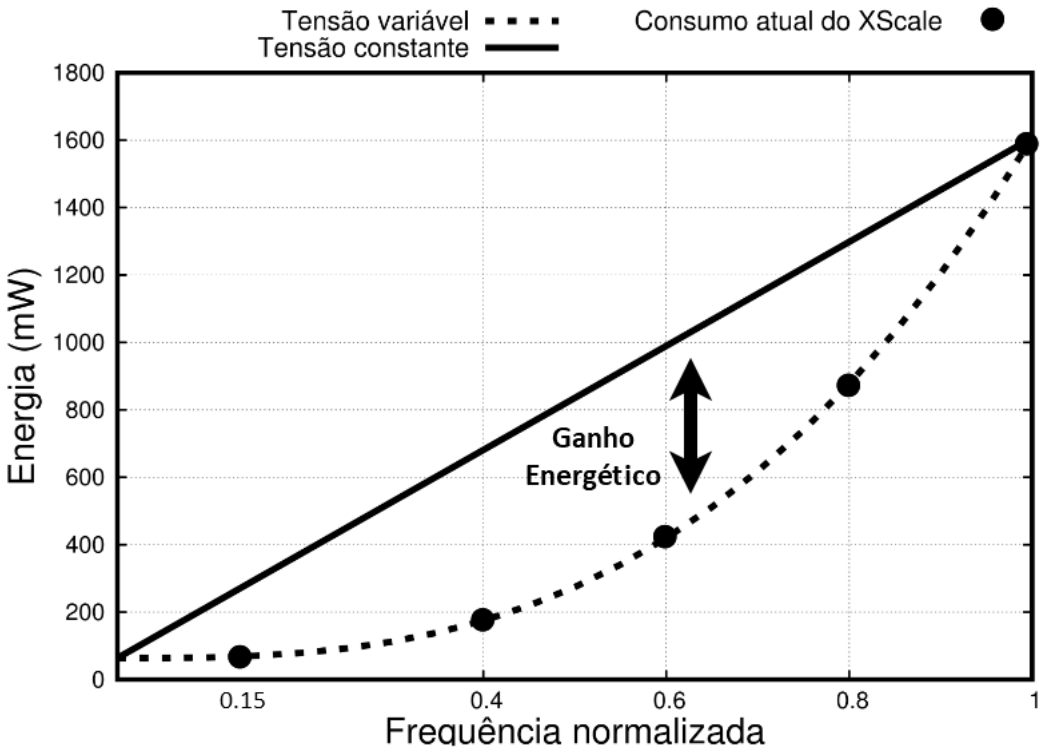
Como o processador é um circuito integrado da família *Complementary Metal Oxide Semiconductor* (CMOS), a dissipação energética nesses tipos de circuitos é dada pela soma da energia dinâmica, estática e de curto circuito. Na Equação 2.4 é representado o consumo energético de dispositivos CMOS:

$$P_{\text{CMOS}} = P_{\text{dinâmico}} + P_{\text{estático}} + P_{\text{curtocircuito}} \tag{2.4}$$

No qual $P_{\text{dinâmico}}$ representa a energia dissipada pela carga e descarga dos capacitores, $P_{\text{estático}}$ é a energia dissipada pelos diodos inversamente polarizados e $P_{\text{curtocircuito}}$ representa a energia dissipada pelo caminho direto entre o Vdd e o GND que acontece por curtos períodos de tempo durante o período de comutação. O $P_{\text{dinâmico}}$ é o principal componente do consumo de dispositivos CMOS (SAINI, 2015). O $P_{\text{dinâmico}}$ é representado como:

$$P_{\text{dinâmico}} = C \times v^2 \times f \tag{2.5}$$

Figura 4 – Ganho energético alcançado com o uso de DVFS.



Fonte: (SAINI, 2015) modificado.

No qual C denota a capacitância de comutação efetiva, v representa a tensão imposta sobre o circuito e f é a frequência de operação. Na Figura 4 é apresentado o possível ganho energético do processador XScale com o uso de DVFS.

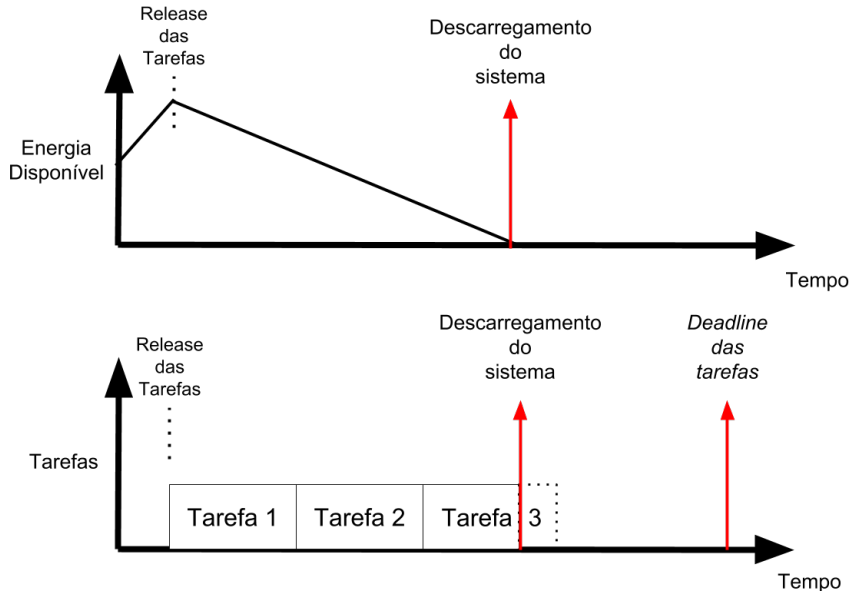
2.2.7 Escalonador de Tarefas

Os SEMBs frequentemente possuem várias tarefas competindo pela unidade central de processamento (CPU) do processador ao mesmo tempo. Caso o processador não possua CPUs suficientes para executar todas as tarefas simultaneamente, cabe ao escalonador de tarefas escolher a ordem de execução das tarefas concorrentes seguindo uma política imposta por um algoritmo de escalonamento. Como os SEMBs geralmente são sistemas limitados, se comparados aos PCs, a escolha de um bom algoritmo de escalonamento pode fazer grande diferença no desempenho e no consumo do sistema.

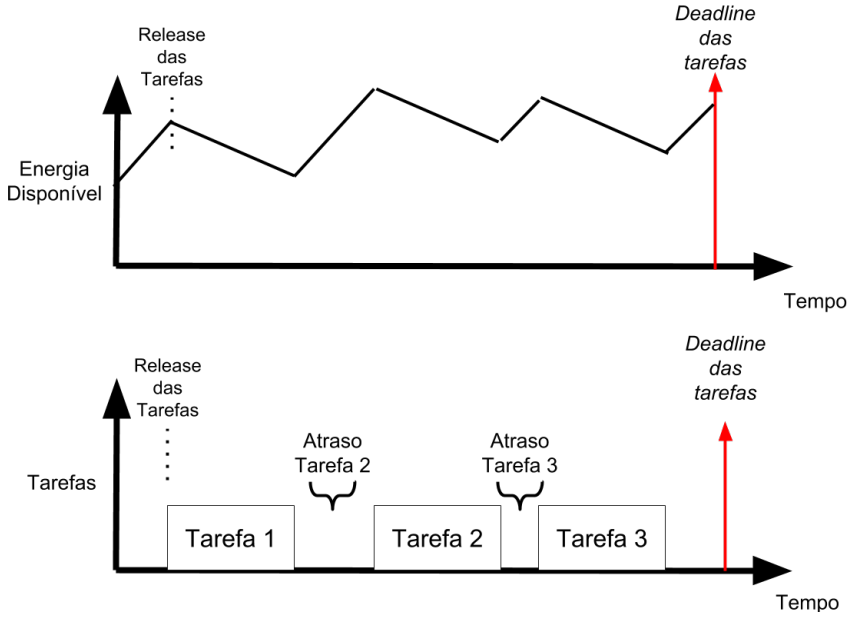
Existem estratégias de escalonamento, que são aliadas com as TGE, para garantir a eficiência energética do sistema. Geralmente a demanda energética de um processador quando está executando uma tarefa é maior do que a energia gerada pelo SCE. Portanto, o sistema de captura só consegue carregar a bateria quando o processador está ocioso. O escalonador pode atrasar as tarefas visando à captura de energia. Na Figura 5 são mostrados os benefícios dessa técnica no escalonamento de tarefas de tempo real.

Como pode-se ver na Figura 5(a), a bateria encontra-se carregando até a Tarefa

Figura 5 – Comparativo entre escalonamento de tarefas



(a) Escalonamento sequencial mais rápido possível



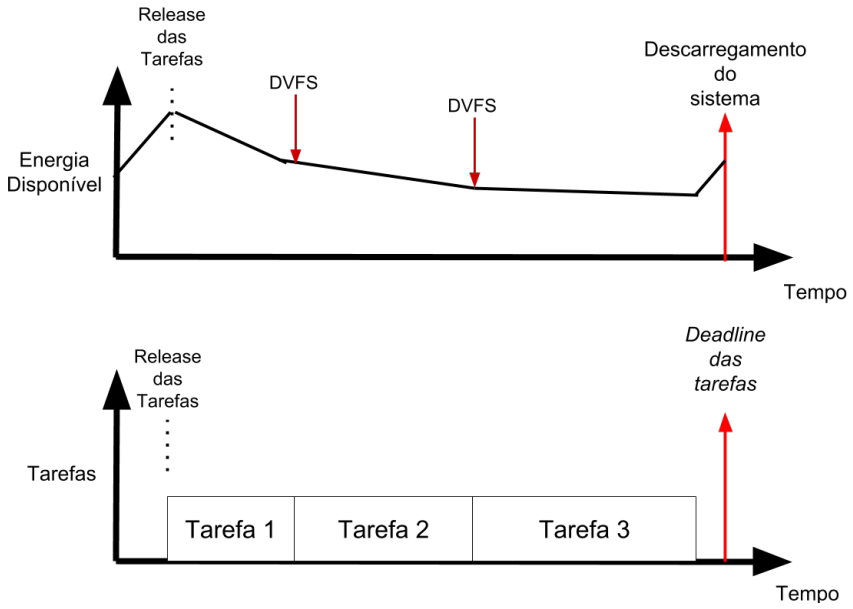
(b) Escalonamento com atraso

1 começar a ser processada. As tarefas são processadas o mais rápido possível, ou seja, imediatamente após uma tarefa terminar sua execução outra é entregue ao processador. O processamento das tarefas causa uma queda na energia do sistema até o descarregamento total. Note que, embora exista tempo até os *deadlines* das tarefas, o sistema descarrega antes da Tarefa 3 terminar sua execução.

Já na Figura 5(b), o escalonador atrasa a execução das tarefas para preservar a energia do sistema. Ao final da Tarefa 1 o escalonador decide atrasar o processamento da Tarefa 2 visando à energia do sistema. Vale salientar que o sistema aproveita os atrasos para capturar energia, desse modo, todas as tarefas são executadas antes dos seus *deadlines* e o nível de energia do sistema permanece elevado. Alinhar o escalonador de tarefas com o DVFS é outra técnica que pode ser utilizada.

Na Figura 6 é mostrado como o escalonador de tarefas pode se utilizar do DVFS para aumentar a eficiência energética do sistema.

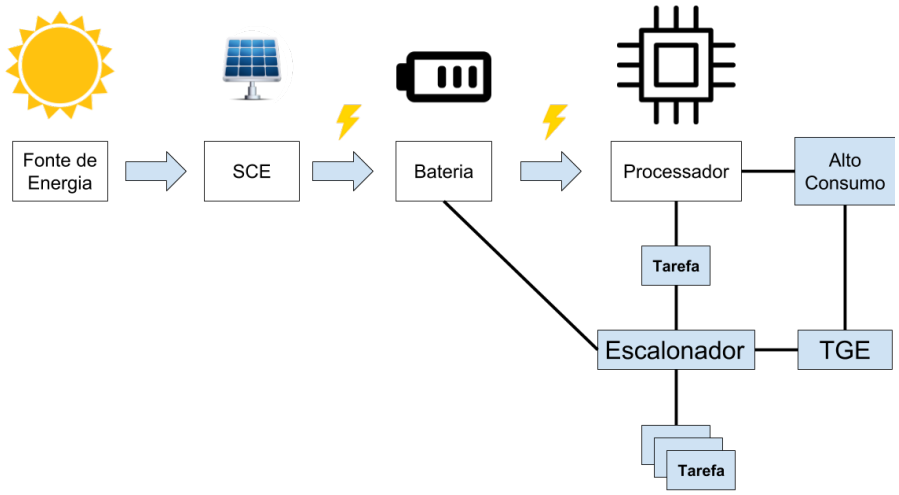
Figura 6 – Escalonamento em conjunto com DVFS



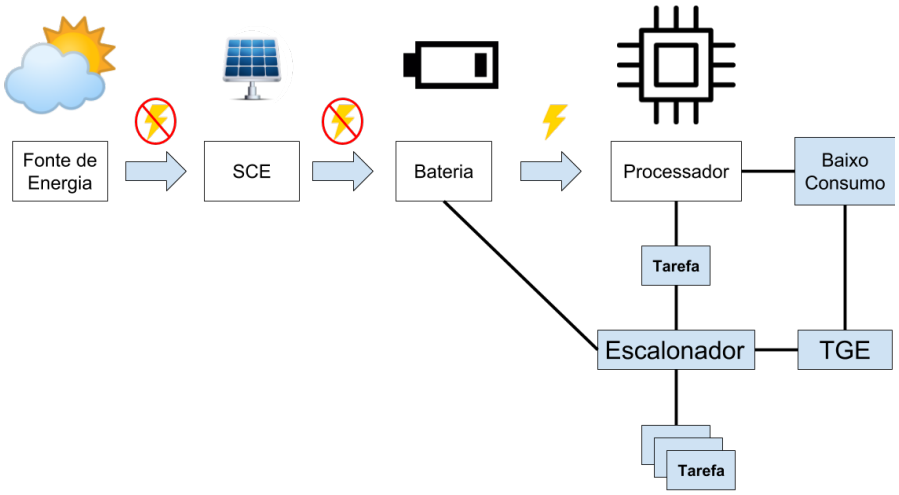
Como se pode analisar, a Tarefa 1 é executada na frequência máxima do processador causando uma queda acelerada no nível de energia do sistema. Ao fim da Tarefa 1, a técnica de DVFS é aplicada, nesse momento o tempo de processamento da tarefa aumenta, porém a energia gasta para execução diminui, causando uma queda menor no nível de energia. Por fim, a frequência do processador é mudada novamente para melhorar a eficiência energética. Nesse estado a Tarefa 3 tem o dobro do tempo de processamento se comparado à Tarefa 1, contudo, a energia utilizada para executar essa tarefa é menor. Desde modo, foi possível manter o sistema ativo para executar as 3 tarefas antes dos seus *deadlines*. Como os SECEs podem trabalhar em vários estados de energia, o escalonador de tarefas precisa ser capaz de mudar o comportamento do sistema dependendo da energia disponível. Na

Figura 7 é apresentado um SECE com um processador com múltiplas frequências, em condições contrárias.

Figura 7 – SECE operando em diferentes estados de energia



(a) Estado de energia abundante



(b) Estado de energia escassa

Na Figura 7(a) a energia convertida pelo SCE é suficiente não somente para atender a demanda energética do processador, como também carregar a bateria do sistema. Neste caso, através do nível da bateria, o escalonador identifica o estado de abundância de energia e utiliza a técnica de DVFS para fazer o processador trabalhar com sua frequência máxima, desse modo evitando um possível *overflow* da bateria.

Já na Figura 7(b), devido a mudanças climáticas, a capacidade de geração da fonte de energia reduz-se drasticamente. Neste estado, o SCE é capaz de fornecer pouquíssima ou nenhuma energia para o sistema, transformando a bateria na principal fonte de alimentação. O consumo do processador vai descarregando a bateria e neste momento cabe ao escalonador identificar a escassez energética enfrentada pelo sistema e utilizar as TGE de modo a

minimizar o consumo, aumentando assim a vida útil do sistema. Deste modo, o sistema pode continuar operante até que o SCE volte a fornecer energia. O processador trabalha de maneira mais lenta quando está em modo de baixo consumo, portanto, as tarefas demoram mais para terminar seus processamentos podendo levar à perda de *deadlines*. O escalonador também tem o papel crucial de minimizar essas perdas. Para os SECEs, o escalonador de tarefas tem um papel crucial para garantir a eficiência energética e o correto funcionamento do sistema (GHOR; CHETTO; CHEHADE, 2011).

O escalonador sozinho não é suficiente para assegurar uma boa eficiência energética. É vital também um bom dimensionamento do sistema, assim como, a escolha de uma arquitetura adequada. Na Tabela 2 são apresentadas 4 arquiteturas diferentes.

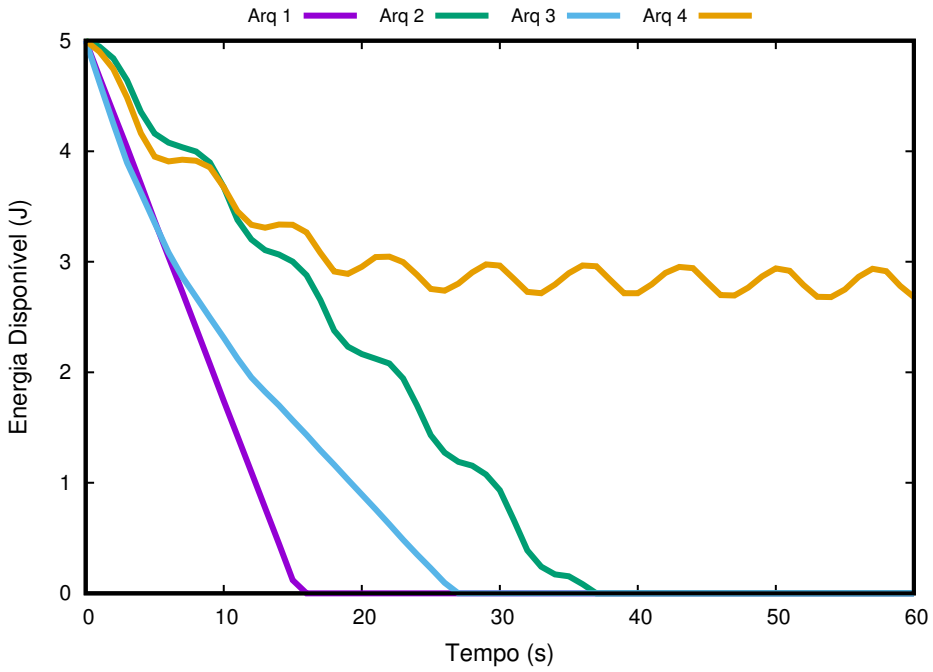
Tabela 2 – Arquiteturas propostas

Arquiteturas	Processador	Baterias	TGE	SCE
Arquitetura 1	✓	✓	x	x
Arquitetura 2	✓	✓	x	✓
Arquitetura 3	✓	✓	✓	x
Arquitetura 4	✓	✓	✓	✓

Todas as arquiteturas utilizam o mesmo processador e baterias. Na primeira, o sistema não utiliza técnicas de captura ou gerenciamento de energia, fazendo da bateria o único meio de alimentação. A segunda arquitetura utiliza um SCE, mas não faz uso de TGE. A terceira arquitetura utiliza TGE, mas não possui SCE. Por fim, a quarta arquitetura possui SCE e faz uso de TGE. O algoritmo de escalonamento utilizado foi o *Earliest Deadline First* (EDF), em que as tarefas que possuem o *deadline* mais próximo tem a maior prioridade. A bateria possui uma capacidade máxima de 5 J e encontra-se totalmente carregada no tempo $t = 0$. Na Figura 8 é apresenta um exemplo de como o escalonador de tarefas alinhando com as TGE podem aumentar a eficiência energética do sistema. Neste exemplo, foi medido o consumo energético durante 60 segundos de um mesmo sistema sob a perspectiva das arquiteturas citadas acima.

Na primeira arquitetura, o consumo constante de energia do processador leva ao descarregamento total da bateria aos 16 segundos de uso do sistema. Já na segunda arquitetura, o SCE consegue gerar energia dando uma sobrevida ao sistema se comparado com a Arquitetura 1. Contudo, esta energia gerada não é suficiente para atender a demanda energética do processador, o que leva a um descarregamento total do sistema aos 37 segundos. A Arquitetura 3 faz uso de TGE, mas como a bateria é a única fonte de energia do sistema, estas técnicas não conseguem evitar o descarregamento total que acontece aos 26 segundos. Finalmente, o uso de SCE aliado com TGE presente na Arquitetura 4, leva o sistema a um estado de equilíbrio evitando o descarregamento total. Caso não aconteçam grandes atenuações na fonte de energia renovável, a Arquitetura 4 pode manter o sistema operante por longos períodos sem a necessidade de interrupções ou manutenções.

Figura 8 – Consumo energético de um mesmo sistema sob arquiteturas diferentes.



2.3 Algoritmos de escalonamento

Nesta seção, são apresentados alguns algoritmos clássicos utilizados no escalonamento de RTS, assim como, algoritmos específicos para SECE de tempo real.

2.3.1 Políticas clássicas de escalonamento para RTS

O *Earliest Deadline First* (EDF) é um algoritmo de escalonamento dinâmico no qual a tarefa que possui o *deadline* mais próximo tem maior prioridade no sistema. A tarefa anuncia sua presença e seu *deadline* sempre que necessita de tempo de CPU, o escalonador possui uma lista de processos executáveis ordenada pelo *deadlines* das tarefas. O algoritmo executa a tarefa que está na cabeça da lista, aquela que está mais próxima de atingir a restrição temporal. Caso uma tarefa mais urgente apareça, o algoritmo interrompe a tarefa que está em execução e a substitui pela tarefa mais prioritária.

Considere 3 tipos de tarefas, as de tipo A possuem tempo de execução = 15ms, período = 25 ms e *deadline* = 25 ms após ela ser lançada no sistema. As tarefas de tipo B possuem tempo de execução = 10 ms, período = 40 ms e *deadline* = 40 ms após seu lançamento no sistema. Finalmente, as tarefas tipo C apresentam tempo de execução = 5 ms, período = 55 ms e *deadline* = 25 ms após ela ser lançada no sistema. Na Tabela 3 é mostrado as características do conjunto de tarefas.

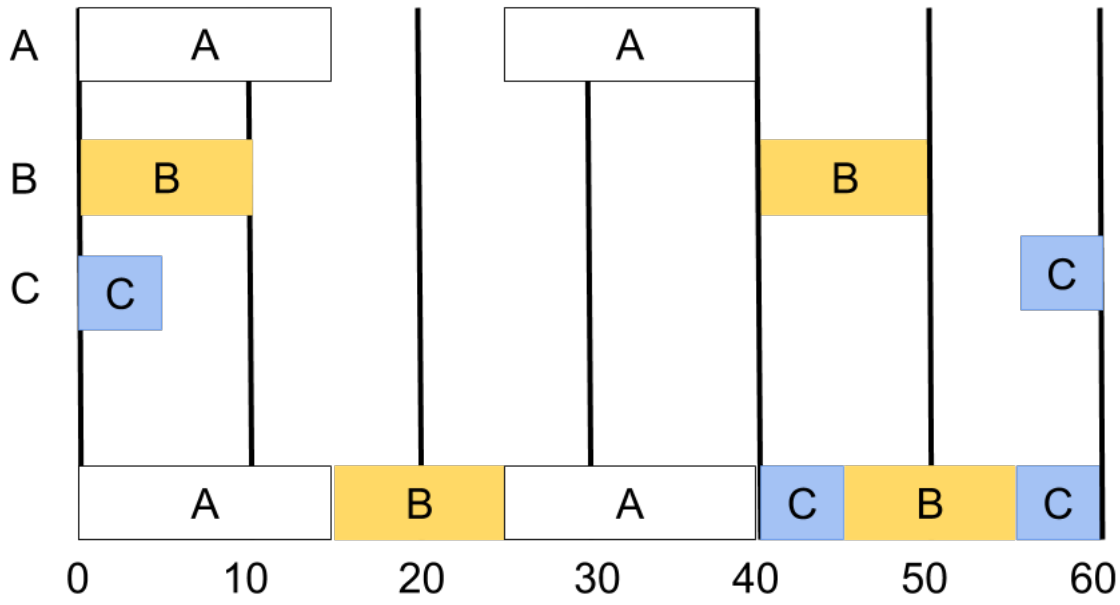
Na Figura 9 é apresentado o escalonamento do conjunto de tarefas acima utilizando o algoritmo EDF. As Tarefas A, B e C aparecem no sistema no tempo $t = 0$, como A possui o prazo mais curto, é escalonada primeiro, seguida pela tarefa B. Aos 25 ms a

Tabela 3 – Características do conjunto de tarefas

Tarefas	Tempo de lançamento (r_i)	Tempo de execução (c_i)	Período (T_i)	Deadline ($D_i = r_i + T_i$)
A	0	15	25	25
B	0	10	40	40
C	0	5	55	55

tarefa A possuindo *deadline* = 50 ms é lançada novamente no sistema, nesse momento, como a tarefa C possui *deadline* = 55 ms a tarefa A é executada. Aos 40 ms a tarefa B reaparece com *deadline* = 80 ms, mas a tarefa C é executada primeiro devido sua prioridade. Após a execução da tarefa C e B, aos 55 ms uma nova instancia da tarefa C reaparece no sistema, como não existe mais tarefas concorrendo pelo processador, essa tarefa é executada imediatamente.

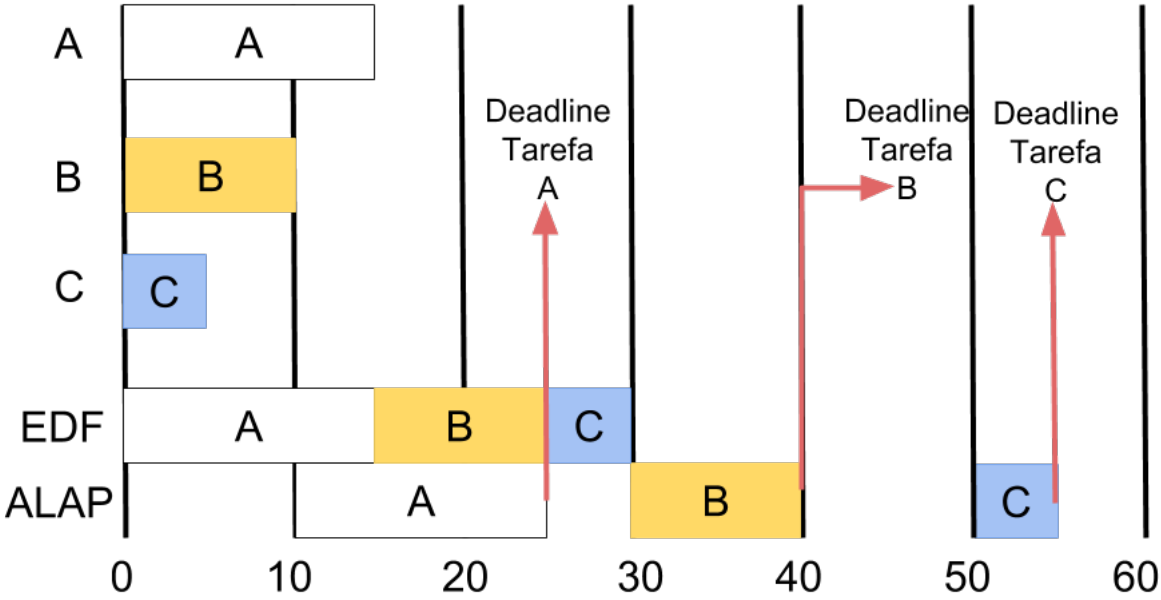
Figura 9 – Escalonamento segundo o algoritmo EDF



O algoritmo *As Late As Possible* (ALAP) é uma variação do EDF que busca atrasar a execução das tarefas o máximo possível. Desse modo, um SECE pode aproveitar a ociosidade gerada para recarregar as baterias. Diferente do EDF que executa as tarefas assim que as mesmas estão prontas, o ALAP busca terminar o processamento das tarefas no momento de seus *deadlines*. Na Figura 10 é mostrado o comparativo do escalonamento do conjunto de tarefas utilizando os algoritmos EDF e ALAP.

Note que os 2 algoritmos de escalonamento seguem a mesma ordem de execução. Contudo, diferente do ALAP, o EDF executa as tarefas assim que as mesmas estão prontas. Utilizando o algoritmo de escalonamento ALAP, o sistema gasta 55 ms para executar todas as tarefas, enquanto que, utilizando o EDF, o tempo necessário é de apenas 30

Figura 10 – Comparativo EDF X ALAP



ms. Apesar dessa diferença de tempo os dois algoritmos conseguiram atender todas as restrições temporais. Vale salientar que em um sistema real o tempo de execução das tarefas é variável, portanto, caso este tempo seja mal estipulado o ALAP pode não conseguir atender os *deadlines* das tarefas.

2.3.2 *Feedback Scheduler Energy Harvesting*

O *Feedback Scheduler Energy Harvesting* (FS-EH) (ABBAS et al., 2014) 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 utiliza uma tarefa de *feedback* que tem como objetivo estipular o tempo de execução de cada tarefa do sistema e utilizar 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 energia da bateria $E(t)$ for maior do que *Threshold* ($E(t)$ está acima do mínimo desejado), ou $U_{inst}(t) \geq 1$, 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 $\{S_m \geq max(U_{inst}(t), \frac{E(t)}{Threshold})\}$.

Considere um processador capaz de trabalhar com os seguintes fatores de velocidade

0,3, 0,6, 0,8 e 1 e $U_{inst}(t) = 0,5$, $Threshold = 2{,}5$ e $E(t) = 1,0$, logo, como $E(t) < Threshold$ e $U_{inst}(t) < 1$ o fator de velocidade utilizado será o maior valor entre $U_{inst}(t) = 0,5$ e $\frac{E(t)}{Threshold} = 0,4$, ou seja 0,5. Contudo, o processador em questão não consegue operar utilizando este S_m, logo, o fator de velocidade escolhido será o menor S_m possível que seja maior que 0,5, ou seja, $S_m = 0,6$. Devido a sua fácil implementação e entendimento o FS-EH é o algoritmo de escalonamento utilizado nessa pesquisa.

2.4 Modelagem de SECE

Esta seção apresenta as ferramentas utilizadas nesta trabalho para a modelagem, simulação e análise dos SECE.

2.4.1 Redes de Petri Coloridas

Um Sistema a Eventos Discretos (SEDs) é um sistema dinâmico em que a evolução no tempo é determinada pela ocorrência de eventos físicos, possivelmente em intervalos regulares. É caracterizado pela mudança de estados, valores discretos, à medida que os eventos acontecem. (CASSANDRAS; LAFORTUNE, 2008)

Redes de Petri Coloridas (RPCs) consistem em uma linguagem gráfica para a construção de modelos de SED e análise de suas propriedades. RPC é ainda uma linguagem de modelagem que combina as funções das redes de Petri com linguagens de programação de alto nível (JENSEN; KRISTENSEN, 2009). A RPC adota a linguagem de programação CPN ML (do inglês, *Coloured Petri Net Modelling Language*) como a linguagem padrão que, por sua vez, fundamenta-se na linguagem de programação funcional *Standard ML* (ULLMAN, 1998).

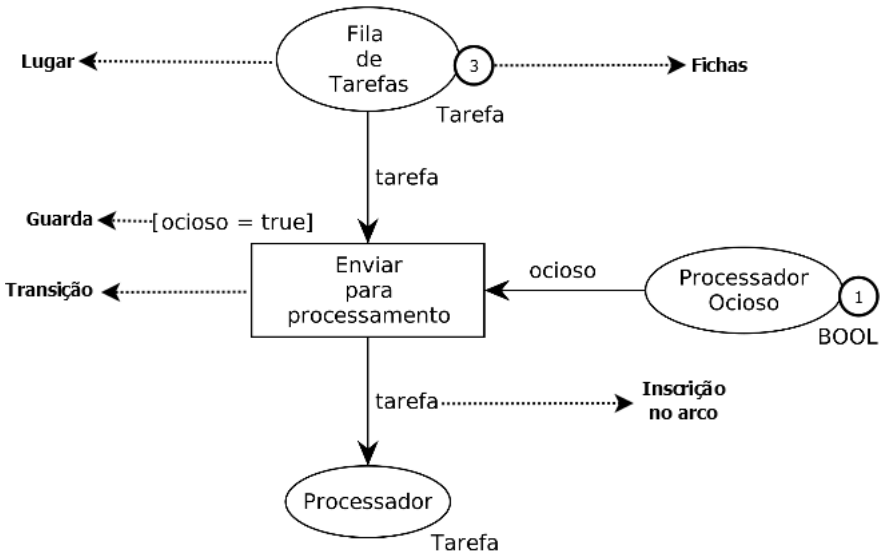
RPCs são destinadas ao uso prático, em particular porque elas permitem a construção de modelos compactos e paramétricos. A vantagem da RPCs sobre as redes de Petri (RPs) é a capacidade de modelar sistemas complexos gerando modelos com alto nível de abstração e melhor representação gráfica (JENSEN; KRISTENSEN, 2009).

Modelos RPC podem conter vários níveis de abstração. Podem ser construídos em um conjunto de módulos ou páginas que interagem entre si através de um conjunto de interfaces bem definidas, assim como as linguagens de programação. O conceito de páginas em RPC fundamenta-se num mecanismo de estrutura hierárquica, que permite uma página possuir sub-páginas reutilizáveis em diversas partes do modelo.

Na Figura 11 é apresentada a estrutura básica de uma RPC que consiste de lugares, transições e arcos. Os lugares representam os estados do sistema modelado. Os nomes dos lugares são escritos dentro de elipses ou círculos e não possuem significado formal, mas possuem um significado prático na legibilidade do modelo. Cada lugar pode ser marcado

com uma ou mais fichas que contém um determinado valor associado. Este valor é chamado de cor. O estado do sistema, ou seja, a marcação do modelo, é representado pela quantidade de fichas, suas cores e em que lugares estas fichas estão contidas.

Figura 11 – Estrutura Típica de uma RPC



As transições representam os eventos que podem ocorrer no sistema. Cada transição é representada por um retângulo com o nome no seu interior. Quando o evento representado pela transição ocorre, diz-se que a transição ocorre ou é disparada. Quando uma transição é disparada, fichas são removidas dos lugares de entrada (lugares de origem de um arco) da transição e fichas são adicionadas aos lugares de saída (lugares de destino de um arco) da transição. A remoção e adição de fichas é determinada pela expressão de arco (inscrições próximas a cada arco). As expressões de arco são escritas utilizando a linguagem CPN ML (ULLMAN, 1998)) e utilizam-se de variáveis, constantes, operadores e funções. Quando todas as variáveis em uma expressão são ligadas ao tipo de dado correto, a expressão pode ser calculada.

Transições possuem guardas, que são expressões booleanas, as guardas devem ter valor verdadeiro para que a transição possa estar habilitada, caso contrário, a transição é desabilitada e não pode ocorrer. Assim, a guarda adiciona uma restrição extra para a habilitação de uma transição.

2.4.1.1 Definição formal das RPC

A definição formal dessa linguagem gráfica foi extraída de Jensen e Kristensen (JENSEN; KRISTENSEN, 2009) e é apresentada a seguir.

Definição: Uma rede de Petri colorida é uma 9-tupla.

$$RPC = (P, T, A, \Sigma, V, C, G, E, I,) \tag{2.6}$$

em que:

- P é um conjunto finito de lugares;
- T é um conjunto finito de transições t tal que P$\cap$T $= \emptyset$;
- A $\subseteq$ PT $\cup$ T P é um conjunto de arcos direcionados;
- Σ é um conjunto finito de um conjuntos de cores não vazios;
- V é um conjunto finito de variáveis tais que $Type[v] \in \Sigma$ para todas as variáveis $v \in V$;
- $C : P \rightarrow \Sigma$ é uma função do conjunto de cores que atribui um conjunto de cores a cada lugar;
- $G : T \rightarrow EXPR_V$ é uma função de guarda que associa uma guarda a cada transição t tal que $Type[G(t)] = Bool$;
- $E : A \rightarrow EXPR_V$ é uma expressão no arco que atribui uma expressão de arco a cada arco, tal que $Type[E(a)] = C(p)_{\text{MS}}$, onde p é o lugar conectado ao arco a;
- $I : P \rightarrow EXPR_\emptyset$ é uma função de inicialização que atribui uma expressão de inicialização para cada lugar p tal que $Type[I(p)] = C(p)_{\text{MS}}$.

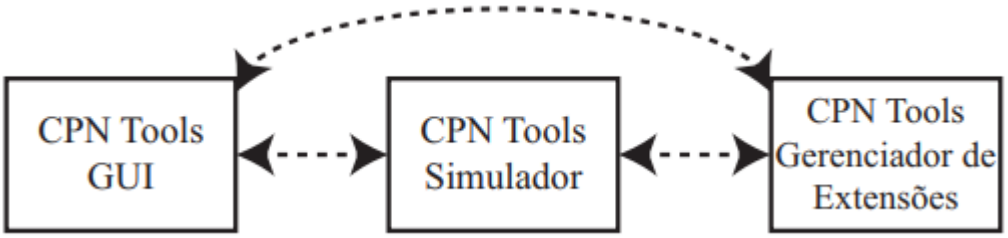
2.4.2 CPN Tools

CPN Tools (CPN Tools, 2018) é uma ferramenta para edição, simulação e análise de RPC. O grande número de usuários associado aos grandes investimentos em seu desenvolvimento resultou em uma ferramenta estável e bastante versátil para usuários e pesquisadores que trabalham com RPC. Entretanto, algumas escolhas de *design* feitas no início do desenvolvimento do CPN Tools tornaram-na de difícil utilização com outros formalismos (WESTERGAARD, 2013).

Na Figura 12 é apresentado a arquitetura do CPN Tools 4. Esta é composta por três processos: uma interface gráfica de usuário (GUI - do inglês, *graphical user interface*), um simulador e um gerenciador de extensões.

O simulador é responsável, dentre outras coisas, pela checagem de sintaxe, compilação e simulação de uma RPC, enquanto que a GUI permite a criação do modelo de forma gráfica e a exibição das interações de simulação.

Figura 12 – Arquitetura do CPN Tools 4.



(Fonte: (JÚNIOR, 2016))

O processo gerenciador de extensões propõe-se a estender as funcionalidades do simulador através da reutilização de códigos. A partir dele é possível, dentre outras coisas, realizar o pré-processamento e modificações no modelo bem como criação de novos métodos e funcionalidades, que podem ser invocados a partir da GUI. Toda a comunicação entre esses três processos ocorre através do *Transmission Control Protocol* (TCP) (WESTERGAARD, 2013).

2.4.2.1 Extensibilidade nativa ao CPN Tools

Comms/CPN é uma biblioteca escrita em CPN ML que implementa a interface *socket* no CPN Tools, habilitando a comunicação entre modelos RPC e processos externos através da pilha TCP/IP (CPN Tools, 2018). Embora a motivação inicial por trás do desenvolvimento de comunicação entre modelos RPC e processos externos venha do desejo de visualização do comportamento do sistema modelado através de retornos gráficos, a Comms/CPN pode ser utilizada, por exemplo, para que o modelo RPC faça interações com o meio físico. Exemplos dessa interação incluem leitura de sensores de temperatura e luminosos, de teclados, etc. Além disso, algoritmos que possuam alto custo computacional podem ser executados externamente durante uma simulação, beneficiando-se da comunicação com processos externos.

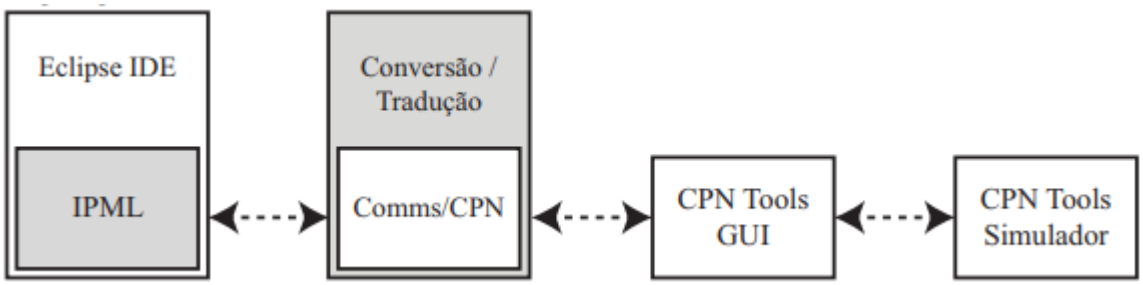
2.4.2.2 Interface de Programação Multilinguagem

No CPN Tools a adição de novas funcionalidades, mesmo as mais simples, normalmente é uma atividade custosa, que envolve conhecimentos aprofundados em RPC e linguagem de programação CPN-ML, e está associada à reprogramação do modelo. A situação é mais grave em eventuais necessidades de adaptações em modelos complexos concebidos por terceiros, pois o custo para entender a programação e a lógica empregada na construção do mesmo, associada ao custo da reprogramação, são fatores que podem comprometer o reuso.

Visando contornar os problemas citados acima, neste trabalho utilizamos a Interface de Programação Multilinguagem (IPML) proposta por JÚNIOR (2016). Através dela é possível adicionar especializações a um modelo RPC sem a reprogramação completa do

modelo. Adicionalmente, essas especializações podem ser realizadas em linguagens de programação que sejam mais familiares ao usuário (Java ou C++, por exemplo). A IPML utiliza a Comms/CPN como mecanismo de comunicação do modelo RPC com processos externos. A Arquitetura de Integração entre um modelo RPC e Interface de Programação Multilinguagem é apresentada na Figura 13.

Figura 13 – Arquitetura de Integração entre um modelo RPC e Interface de Programação Multilinguagem.



(Fonte: (JÚNIOR, 2016))

Para a interface de programação adotamos a *Integrated Development Environment* (IDE) Eclipse (ECLIPSE, 2018). A camada de Conversão/Tradução é responsável por transformar fichas de um modelo em RPC em objetos de uma linguagem Orientada a Objetos (OO), permitindo que tratamentos específicos a essa ficha sejam realizados por métodos externos ao modelo em RPC, o que permite estender simuladores construídos com essa tecnologia sem alterar a modelagem. O retorno do método da linguagem OO, que efetua a modificação de estado nos dados recebidos, é novamente convertido em ficha e passa novamente a reintegrar o modelo.

2.5 Conclusão

No presente capítulo foram apresentados os conceitos aplicados para o desenvolvimento deste trabalho. No próximo capítulo é apresentado o SECE-CPN um modelo em RPC para análise e simulação de SECEs.

3 SECE-CPN: Um modelo para simulação e análise de sistemas embarcados com captura de energia

A simulação é a principal ferramenta para análise e dimensionamento de SECE. Contudo, 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, bateria e etc. Existem vários simuladores de sistemas a eventos discretos capazes de representar o comportamento de 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.

Neste capítulo é apresentado o SECE-CPN, um modelo paramétrico, modular e descrito em Redes de Petri Coloridas (do inglês CPN), cujo 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., 2014). Foi feito também um breve estudo mostrando os riscos que as simplificações podem representar para o projeto desses sistemas.

3.1 Revisão Bibliográfica

Nesta seção são apresentados alguns simuladores utilizados para a análise e simulação de SECE.

O 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 por meio 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; 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 estocástica. 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 o melhor caso. Semelhante ao OMNeT++, o Jitterbug não permite analisar o desempenho de algoritmos de escalonamento.

O TrueTime (CERVIN, A., HENRIKSSON, D. AND OHLIN, M, 2016) é um simulador baseado em Matlab/Simulink para análise de sistemas em tempo real. Assim como no Jitterbug, no TrueTime os sistemas são representados por 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. A falta destas métricas torna difícil a análise de alguns eventos do sistema (troca de contexto, troca de frequência, ociosidade da Unidade Central de Processamento (do inglês CPU)) levando os usuários a produzirem cenários de testes simplificados, desconsiderando o tempo e a 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 capítulo, 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.

À 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. Na Tabela 4 é mostrado um comparativo entre os simuladores apresentados.

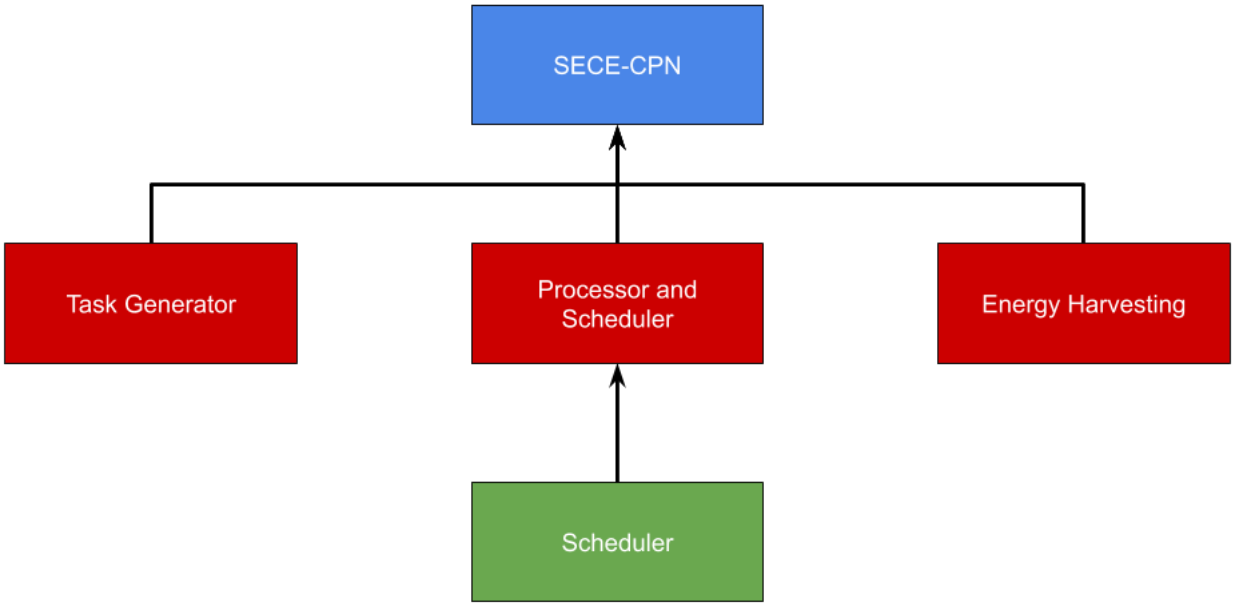
Tabela 4 – Comparativo entre os simuladores

Características	Jitterbug	OMNeT++	Truetime	SECE-CPN
Representação de sistemas não lineares	X	✓	✓	✓
Modelar SCE	X	✓	✓	✓
Modelar TGE	X	X	✓	✓
Medir energia disponível no sistema	✓	X	✓	✓
Medir energia gasta na troca de contexto	X	X	X	✓
Medir energia gasta pelas TGE	X	X	X	✓
Medir energia gasta pela ociosidade da CPU	X	X	X	✓

3.2 Arquitetura da rede

O SECE-CPN é uma arquitetura para modelagem e simulação de sistemas embarcados com captura de energia. 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 modelos de fontes de energia renováveis (solar, eólica, cinética, dentre outras), dois modelos de processadores: XScale (INTEL, 2007) e PXA270 (MARVELL, 2007), 4 algoritmos de escalonamento, *Energy Aware - Dynamic Voltage Frequency Scaling* (EADVFS) (ABBAS et al., 2013), *FeedBack Scheduler with Energy Harvesting* (FS-EH) (ABBAS et al., 2014), *Earliest Deadline First* (EDF) e *As Late As Possible* (ALAP), e alguns conjuntos de tarefas (*task sets*) predefinidos. Por fim, ao final de cada simulação as

Figura 14 – Hierarquia do SECE-CPN



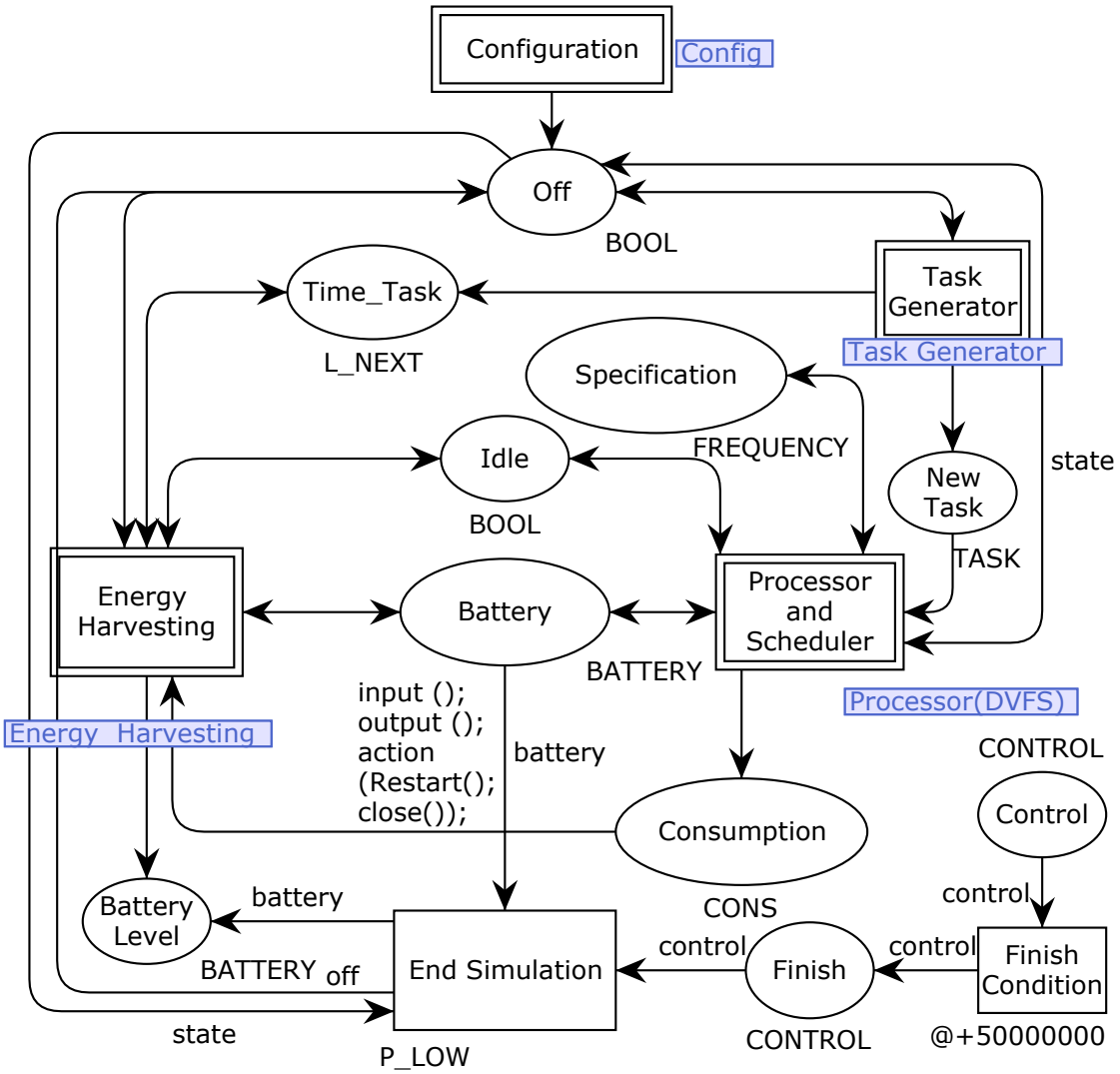
seguintes métricas são disponibilizadas: energia disponível, *deadlines* perdidos, percentual de ociosidade, tempo e energia perdidos em tarefas que não atingiram suas restrições temporais, tempo e energia gastos na troca de contexto do escalonador, tempo e energia gastos na troca de frequência do processador, 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 IPML da ferramenta.

3.2.1 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). Na Figura 14 é apresentado a hierarquia do SECE-CPN.

O SECE-CPN possui um total de 6 páginas, divididos em 3 níveis hierárquicos. Na Figura 15 é apresentada a página "SECE-CPN", que representa o modelo de SECE em maior nível de abstração. A página "Configuration" representa a configuração do SECE que deseja-se simular. A página "Task Generator" modela o gerador que injeta tarefas no modelo. A página "Energy Harvesting" modela o sistema de captura de energia. A página "Processor" modela o processador do sistema. Por fim, a página "Scheduler" modela o escalonador de tarefas. As seções seguintes apresentam as páginas do SECE-CPN em detalhes.

Figura 15 – Pagina SECE-CPN



3.2.2 Subpágina Configuration

A subpágina "*Configuration*", apresentada na Figura 16, 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 IPML 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 o modelo CPN e a IE.

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

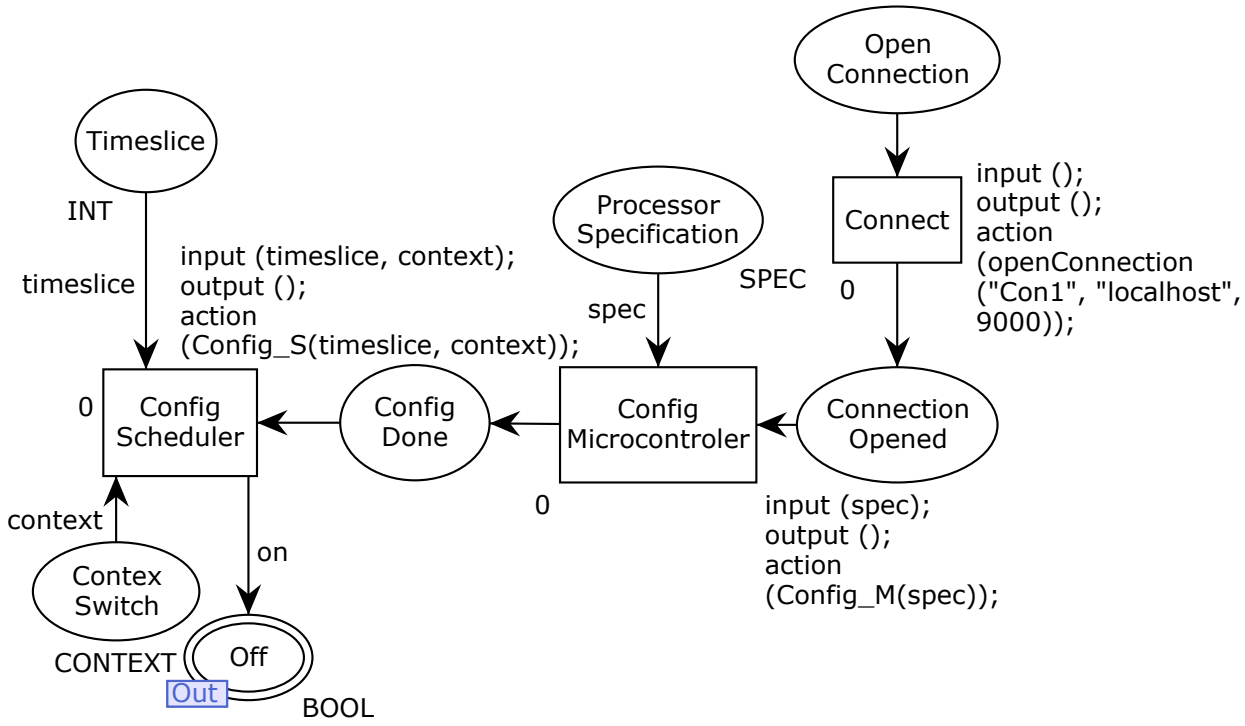


Figura 16 – Subpágina Configuration

e (iv) uma lista de frequências (em Megahertz (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 IPML.

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 (ηs) 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 (ηs) e energia (microJoules (μJ)) 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 milisegundo (ms), capaz de realizar a troca de contexto em 10 ηs consumindo 1.0 μJ por troca.

3.2.3 Subpágina Task Generator

A subpágina *Task Generator* apresentada na Figura 17 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 produto 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 à 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.

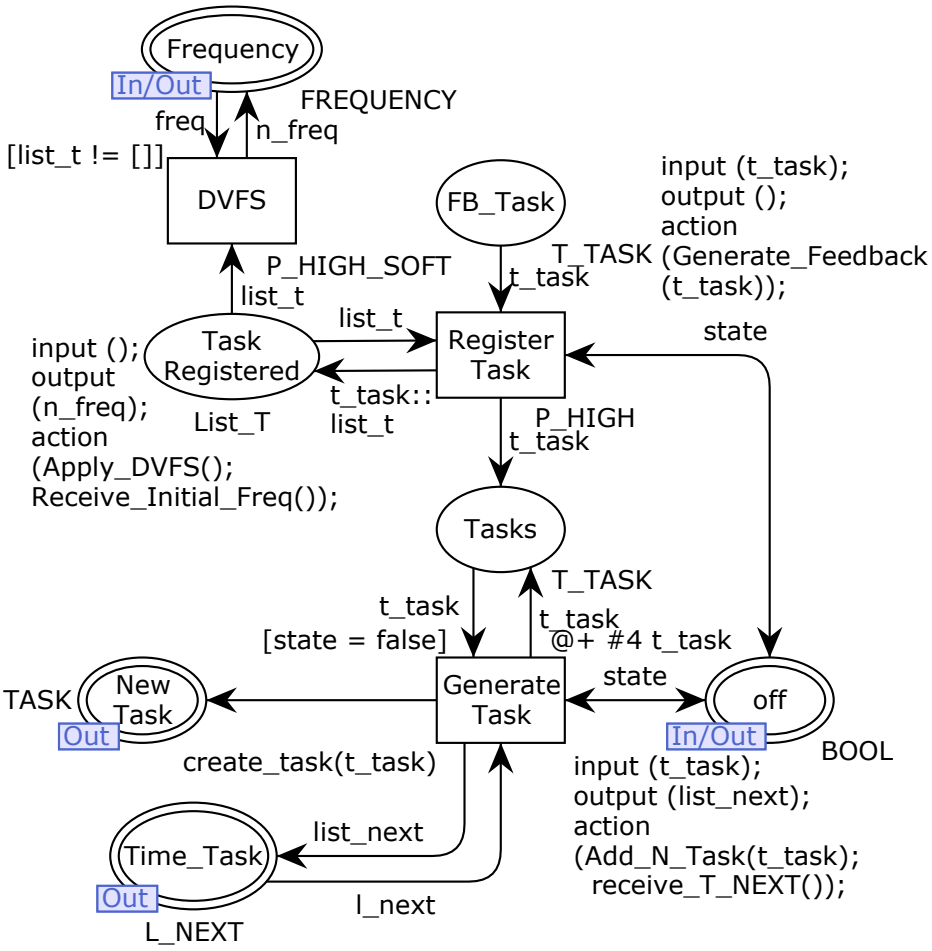
Após o registro de todas as tarefas (cada tarefa é registrada pelo disparo da transição *Register Task*), a transição *DVFS* é habilitada. O disparo desta transição retira uma ficha do lugar *Task Registered* e deposita uma ficha do tipo FREQUENCY: INT no lugar *"Frequency"*. Uma ficha nesse lugar representa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

Figura 17 – Subpágina Task Generator

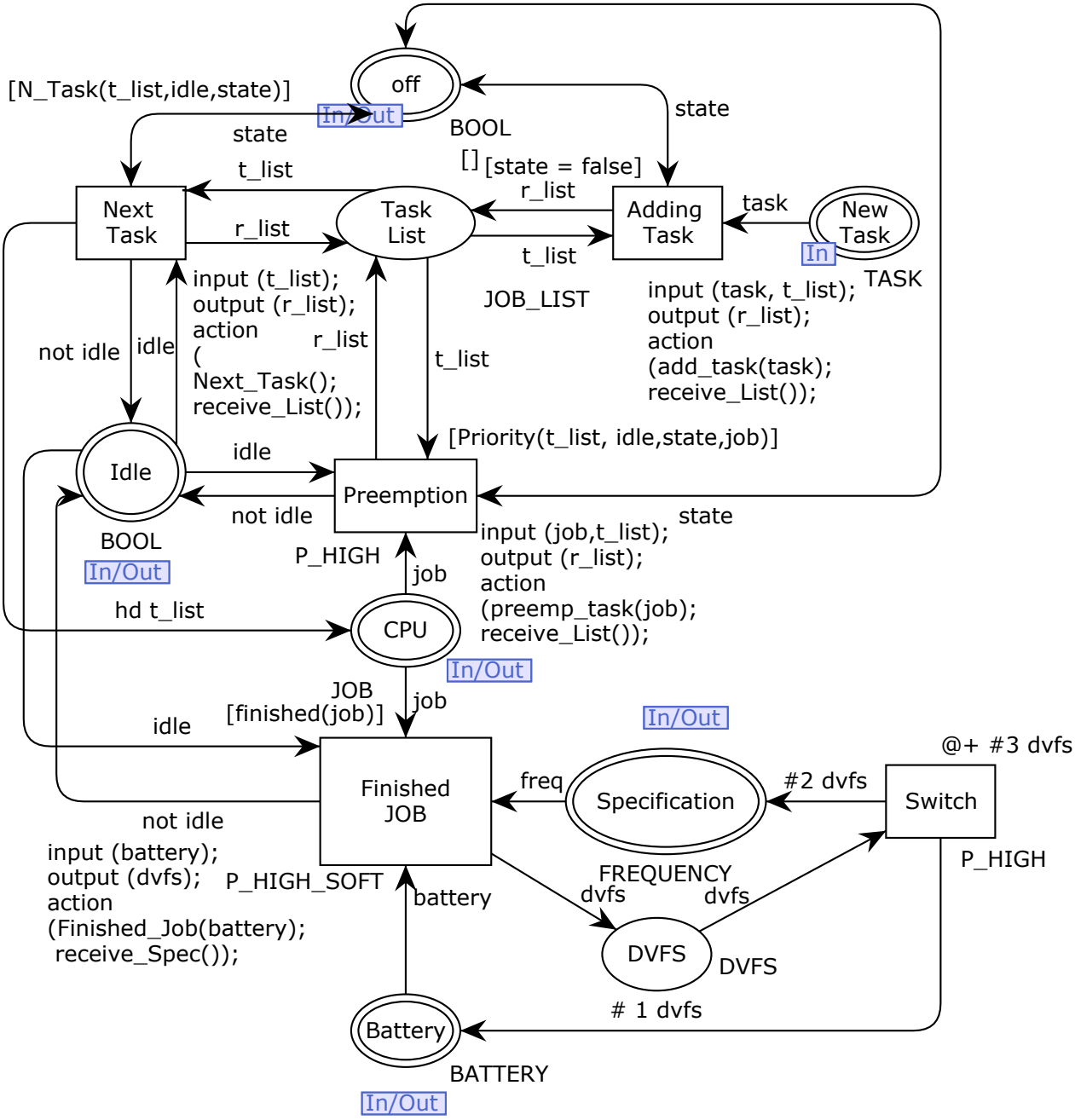


Por fim, a ficha do lugar *Time_Task* é da cor L_NEXT. L_NEXT é composta por uma lista de elementos do tipo produto INT * INT. Estes atributos representam o identificador único (id) e o período de uma tarefa, e serve para determinar quando uma tarefa chega ao sistema novamente.

3.2.4 Subpágina Scheduler

O escalonador de tarefas do SECE-CPN é apresentado no modelo da Figura 18. 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 CPU 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).

Figura 18 – Subpágina Scheduler



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.

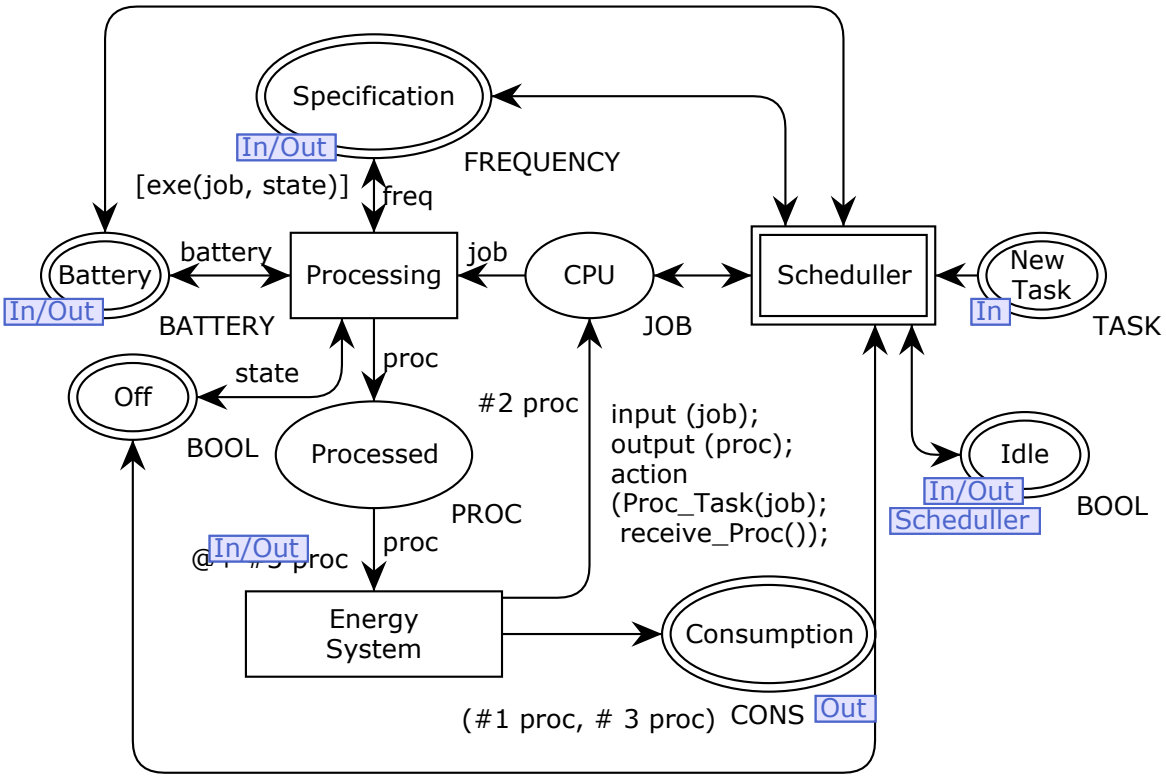
3.2.5 Subpágina Processador

O processador do SECE-CPN é apresentado no modelo da Figura 19. 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.

3.2.6 Subpágina 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).

Figura 19 – Subpágina Processor



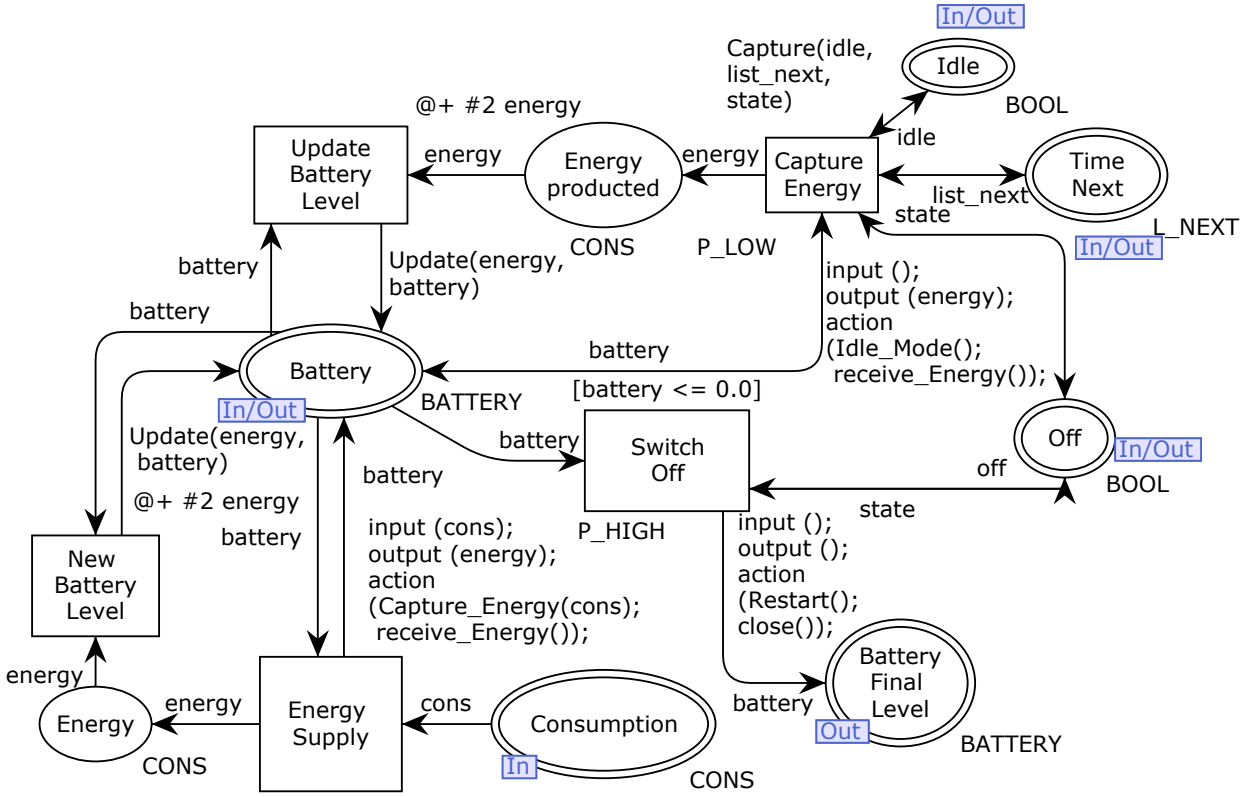
A transição *Capture Energy* é habilitada caso a CPU esteja ociosa e não exista nenhuma tarefa no sistema. Esta transição deposita uma ficha do tipo CONS no lugar *Energy Producted* e representa a energia capturada pelo sistema enquanto a CPU está ociosa e o tempo de ociosidade. A transição *Update Battery 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.

O sistema de captura de energia do SECE-CPN é apresentado no modelo da Figura 20.

3.3 Validação do SECE-CPN

Para a validação do SECE-CPN foram reproduzidos os experimentos feitos por (ABBAS et al., 2014) e seus resultados comparados. Utilizamos o modelo de consumo proposto por (CHEN et al., 2013) de um processador XScale, cujas especificações estão apresentadas na Tabela 5, a função de consumo é dada pela Equação 3.1.

Figura 20 – Subpágina Energy Harvesting



$$f(S) = 1543,28 \times S^{2,28} + 63,28 \tag{3.1}$$

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

Tabela 5 – Parâmetros XScale. (Fonte Abbas et al. (2014) Modificado).

Voltage	0,75	1	1,3	1,6	1,8
Frequência (MHz)	150	400	600	800	1000
Velocidade do Processador (Sm)	0,15	0,4	0,6	0,8	1
Consumo ocioso (Watts)	63,58				
Relacionamento Energia/Frequência	2,58				
Switching Capacitancy	1543,28				
Troca de Frequência (μJ)	1,2				
Tempo para Troca de Frequência (μs)	12				

A energia necessária para alimentar o sistema é fornecida através de um conjunto de baterias recarregáveis que podem armazenar no máximo 2,5 Joules. Foi utilizada a

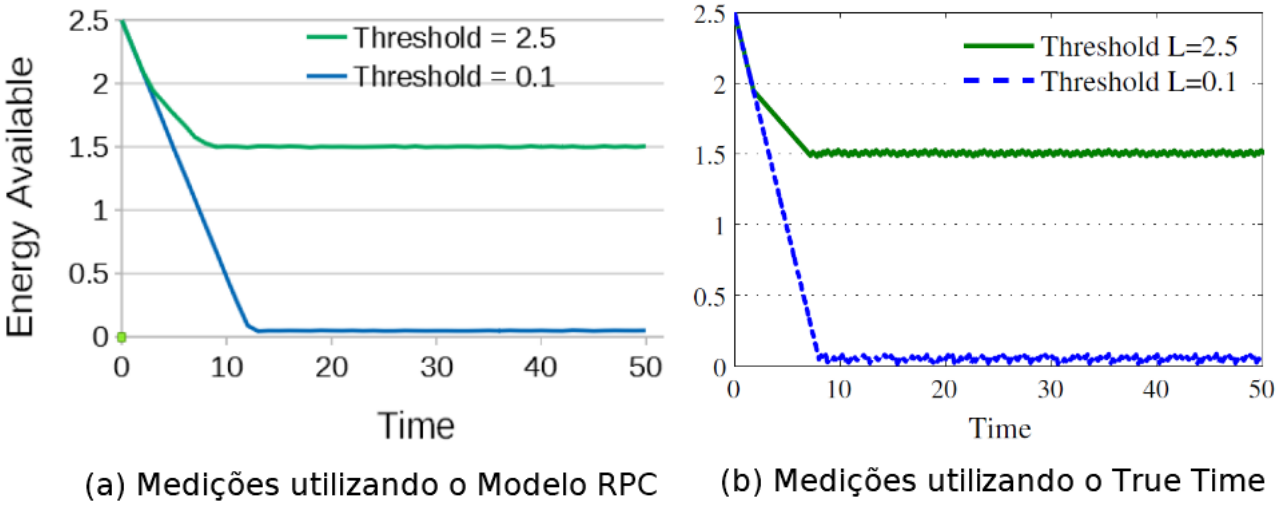
captura de uma fonte de energia solar para fornecer energia extra as baterias. Para simular o comportamento da fonte de energia utilizamos o modelo apresentado na Equação 3.2.

$$Ps(t) = |0.9 \times R(t) \times \cos(\frac{t}{0.7 \times \pi}) \times \cos(\frac{t}{0.1 \times \pi})| \tag{3.2}$$

Em que $R(t)$ é uma distribuição uniforme randômica no intervalo [0,1]. Segundo Abbas et al. (2014), a função $Ps(t)$ é similar à 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*". Esta 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 $T_0 = 200ms$. Os períodos das demais tarefas do conjunto Γ são $T_1 = 19$ ms, $T_2 = 20$ ms e $T_3 = 21$ ms, o pior caso de execução para toda tarefa τ_i ($WCET_i$) é igual a 3 ms e o *deadline* relativo (D_i) é igual ao período da tarefa ($D_i = T_i$), 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 $WCET_i$. Utilizamos o algoritmo de escalonamento FS-EH descrito na seção 2.3.2. Na Figura 21 é apresentado o comparativo da energia disponível no sistema pelo tempo entre o SECE-CPN e a ferramenta de simulação TrueTime.

Figura 21 – Comparativo entre o SECE-CPN e o TrueTime.



(a) Medições utilizando o Modelo RPC (b) Medições utilizando o True Time

3.4 Resultados e Discussões

Visando demonstrar os problemas causados pelas simplificações nos cenários de testes, foi feito 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 representada pela Equação 3.2. Foi simulado dois conjuntos de 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 *deadline* absoluto igual ao período da tarefa, o tempo de computação de cada tarefa varia entre 1 ms e o WCET. O primeiro conjunto de tarefas 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 conjunto de tarefas é composto pelo primeiro mais uma tarefa de período igual à 22 ms, aumentando a $U_{inst}(t)$ para 58 % da CPU.

Todos os conjuntos de tarefas foram simulados em dois cenários. O primeiro cenário foi simplificado para desconsiderar 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 foi retirado as simplificações. Segundo as especificações do XScale (ver Tabela 5), o processador gasta 63,58 milliwatts 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.

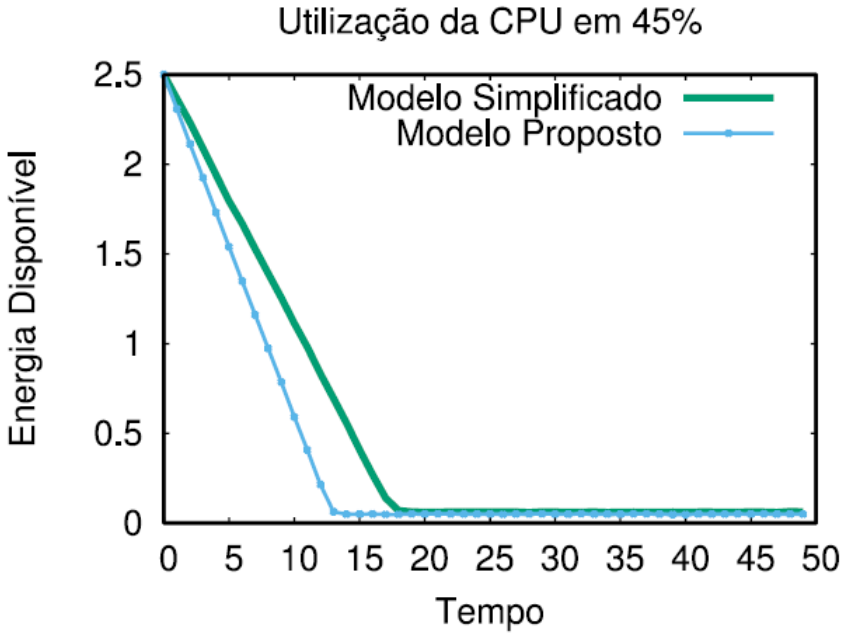
3.4.1 Utilização instantânea máxima de 45 %

Utilizando um *threshold* 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 22 apresenta a energia disponível para o primeiro conjunto de tarefas.

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 energia da 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 diminui 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.

Na Tabela 6 é apresentado o consumo do sistema negligenciado pelas simplificações.

Figura 22 – Energia disponível utilizando o conjunto de tarefas 1



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.

Tabela 6 – Energia negligenciada pelas simplificações do conjunto de tarefas 1

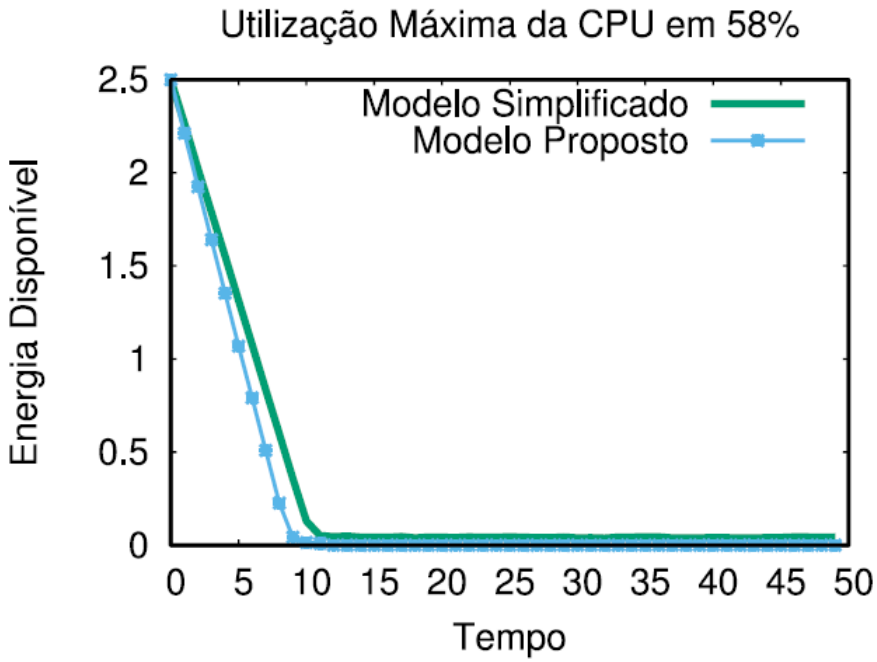
Consumo do sistema	Tempo gasto (s)	Energia Gasta (J)	Porcentagem de energia gasta em relação a E_{max}
Ociosidade	34,62000	2,20000	88,00000 %
DVFS	0,00694	0,00833	0,33000 %
Troca de Contexto	0,00060	0,00060	0,02000%

3.4.2 Utilização instantânea máxima de 58 %

Na Figura 23 é apresentada a energia disponível para o segundo conjunto de tarefas analisado.

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

Figura 23 – Energia disponível utilizando o conjunto de tarefas 2



de *Energy Neutral*, 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. Na Tabela 7 é apresentado o consumo negligenciado no conjunto de tarefas 2. Vale salientar que os testes apresentados anteriormente são estendidos no próximo capítulo.

Tabela 7 – Energia negligenciada pelas simplificações do conjunto de tarefas 2

Consumo do sistema	Tempo gasto (s)	Energia Gasta (J)	Porcentagem de energia gasta em relação a Emax
Ociosidade	7,22000	0,45000	18,00000%
DVFS	0,00018	0,00220	0,08000%
Troca de Contexto	0,00120	0,00120	0,00480%

3.4.3 Conclusão

Neste capítulo 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. (2014), 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 conjunto de tarefas utilizado as simplificações não são justificáveis podendo levar a uma análise equivocada tanto do sistema quanto do algoritmo de escalonamento.

4 Impacto das Simplificações no Resultado das Simulações de SECE

Como já foi visto nos capítulos anteriores, a simulação é a principal ferramenta utilizada para analisar e dimensionar os SECEs. Contudo, muitos sistemas são analisados por meio de cenários simplificados que não refletem a realidade que o sistema irá vivenciar. Estas simplificações podem levar a erros graves no projeto destes sistemas. Desse modo, existe uma necessidade de representar na simulação cenários cada vez mais próximos da realidade.

Por outro lado, cenários muito complexos podem exigir recursos e tempo de projeto na análise de fatores que não tem impacto no funcionamento do SECE. Por exemplo, um sistema de irrigação inteligente deve medir a umidade do solo antes da próxima irrigação, como não existe a necessidade de medição o tempo todo e, considerando que a irrigação aconteça duas vezes ao dia, a sobrecarga energética imposta pela utilização do sensor de umidade é baixíssima, podendo ser desprezado. Contudo, para um sistema onde a umidade do ambiente precise ser medida no intervalo de poucos segundos, o somatório da energia gasta para fazer estas medições pode impactar no resultado da simulação.

É necessário que o projetista de SECE saiba quais e quando os detalhes da simulação possam ser desconsiderados. O foco desta pesquisa é investigar o impacto energético em SECE causado por 3 fatores comumente negligenciados no cenário de teste, estes fatores são: o uso de baterias ideais, o consumo ocioso do processador e a troca de contexto do escalonador. Esta pesquisa também busca identificar cenários nos quais essas negligencias podem ser aceitáveis.

4.1 Metodologia

A metodologia proposta neste trabalho consiste em comparar vários modelos de simulação com diferentes níveis de detalhes sob a perspectiva de diferentes conjuntos de tarefas. A diferença entre os modelos indica o nível de sensibilidade do SECE a uma dada simplificação. Por exemplo, usando as mesmas configurações e conjunto de tarefas para comparar um modelo simplificado, que não considera o consumo ocioso do processador, com um modelo complexo, a diferença do consumo energético entre os dois modelos indica o impacto desta simplificação no resultado final da simulação. A mudança do conjunto de tarefas pode causar um aumento ou uma redução do consumo energético indicando quais parâmetros são associados à simplificação. Deste modo, é possível identificar cenários onde esta simplificação é aceitável.

Este trabalho busca medir o impacto energético de 3 simplificações comumente utilizadas na criação de cenários de simulação de SECE. Para cada simplificação, um modelo complexo é criado e comparado com um modelo simplificado. As simplificações observadas são listadas abaixo:

- Bateria ideal.
- Consumo ocioso nulo
- Troca de contexto nula.

Em (MOSER et al., 2007), (LIU; QIU; WU, 2008), (ABBAS et al., 2013), (ABBAS et al., 2014) e (GHADAKSAZ; SAFARI, 2018), um dispositivo de armazenamento ideal é utilizado na simulação de um SECE. Uma bateria ideal é capaz de armazenar e fornecer energia sem qualquer perda. Entretanto, As operações de carga e descarga de uma bateria real sempre possuem um custo energético associado. Mesmo quando um sistema está ocioso, o processador consome energia para se manter operante. Diversos trabalhos na literatura [(MOSER et al., 2007),(LIU; QIU; WU, 2008), (LIU et al., 2012), (LIN et al., 2016), (GHADAKSAZ; SAFARI, 2018)] não consideram este consumo, podendo levar a conclusões equivocadas do sistema simulado. A troca de contexto é uma operação que salva e carrega as informações de uma tarefa nos registradores do sistema. Geralmente esta operação é extremamente rápida, contudo, existe um custo temporal e energético associado. Embora este custo esteja presente no sistema, vários trabalhos [(MOSER et al., 2007), (LIU; QIU; WU, 2008), (LIU et al., 2012), (ABBAS et al., 2013), (ABBAS et al., 2014), (LIN et al., 2016)] consideram uma troca de contexto instantânea sem custo adicional.

Tabela 8 – Trabalhos que utilizam simplificações no cenário de teste.

Simplificações	Trabalhos
Bateria ideal	[(MOSER et al., 2007), (LIU; QIU; WU, 2008), (ABBAS et al., 2013), (ABBAS et al., 2014) e (GHADAKSAZ; SAFARI, 2018)]
Consumo ocioso	[(MOSER et al., 2007),(LIU; QIU; WU, 2008), (LIU et al., 2012), (LIN et al., 2016), (GHADAKSAZ; SAFARI, 2018)]
Troca de contexto	[(MOSER et al., 2007), (LIU; QIU; WU, 2008), (LIU et al., 2012), (ABBAS et al., 2013), (ABBAS et al., 2014), (LIN et al., 2016)]

Todas as simplificações listadas anteriormente tem um pequeno consumo energético sendo completamente justificáveis. Contudo, este trabalho levanta a hipótese de que, dependendo do conjunto de tarefas, a soma da sobrecarga energética negligenciado pelas simplificações pode resultar em uma quantidade considerável de energia.

4.1.1 Detalhes da Simulação

O impacto energético causado pelas simplificações é medido por meio da comparação entre diferentes modelos de simulação. Na Tabela 9 são apresentados os modelos utilizados, assim como as simplificações inseridas.

Tabela 9 – Modelos propostos e simplificações utilizadas

Modelos	Simplificações		
	Bateria ideal	Consumo ocioso	Troca de contexto
Modelo 1	✓	✓	✓
Modelo 2	x	✓	✓
Modelo 3	✓	x	✓
Modelo 4	✓	✓	x
Modelo 5	x	x	x

O primeiro modelo ignora o *overhead* causado pela troca de contexto, o consumo ocioso e a técnica de DVFS, além de considerar uma bateria ideal capaz de armazenar e fornecer energia sem nenhum custo. O segundo modelo substitui a bateria ideal presente no modelo simplificado, por um modelo de bateria eficiente. O terceiro modelo considera o *overhead* causado pelo consumo ocioso do processador. O quarto modelo considera o custo energético da troca de contexto. Finalmente, o quinto modelo é composto pelas considerações dos modelos 2, 3 e 4 com o propósito de medir o efeito combinado destas simplificações.

Os modelos 2-5 são comparados com o modelo 1 para determinar os impactos no consumo energético do sistema. Diferentes conjuntos de tarefas são utilizados para identificar o relacionamento entre as tarefas e as simplificações. Na Tabela 10 é apresentada as configurações base do sistema.

Tabela 10 – Configurações base do sistema

Parâmetros	valores
Processador	Intel XScale
Frequência do clock da CPU	[150,400,600,800,1000] MHz
Fonte de energia renovável	Solar
Algoritmo de escalonamento	FS-EH
Capacidade máxima da bateria	5,0 Joules

4.2 Modelo Simplificado

Nesta seção é apresento o modelo simplificado (modelo 1) usado como base ao longo desta pesquisa.

4.2.1 Modelo Computacional

Considerando um processador que pode operar em diversas frequências, a velocidade deste processador (S) é definida pela relação entre a frequência atual e a frequência de operação máxima, isto é:

$$S = f_{\text{current}}/f_{\text{max}} \tag{4.1}$$

Neste trabalho é assumido um conjunto de tarefas independentes, em outras palavras, as tarefas não se comunicam ou trocam informações entre si. Esta consideração visa reduzir a complexidade da simulação. As tarefas são definidas pelos parâmetros apresentados na Seção 2.2.4.

Um gerador de tempo de execução randômico, proposto por (ABBAS et al., 2014), foi utilizado para gerar diferentes tempos de execução para as tarefas, assim, uma instância j de um tarefa i ($\tau_{\text{i,j}}$) pode ter um tempo de execução diferente de uma instancia k da mesma tarefa i ($\tau_{\text{i,k}}$). O gerador de tempo de execução randômico é apresentado na Equação 4.2.

$$c_{\text{i}} = -log(1-R)^{1/\text{k}} \times \beta \tag{4.2}$$

Em que R representa uma distribuição randômica e uniforme entre 0 e 1 e k e β são constantes. Assumindo que todas as tarefas possuam um $WCET = 3$ ms, os parâmetros $k = 3$, $\beta = 0{,}0015$ garantem um c_{i} menor ou igual ao $WCET$.

As tarefas são escalonadas segundo a política do algoritmo de escalonamento FS-EH apresentado na seção 2.3.2. O FS-EH utiliza uma tarefa de *feedback*, esta tarefa é periódica e apresenta prioridade máxima no sistema e tem a responsabilidade de modificar a frequência de operação do processador através da estipulação do tempo de execução das tarefas do sistema.

O DVFS muda a frequência de operação do processador para reduzir o consumo de energia. Esta técnica pode aumentar ou reduzir o tempo de execução e o $WCET$ de uma tarefa. Neste trabalho consideramos um relacionamento proporcional entre a frequência de operação e a velocidade do processador. Considerando uma tarefa τ_{i} executando com uma frequência f_{m} com $WCET_{\text{i}} = 3$ ms e $c_{\text{i}} = 1$ ms, o $WCET_{\text{i},\alpha}$ e o $c_{\text{i},\alpha}$ desta mesma tarefa executando a uma frequência $f_\alpha = f_{\text{m}}/2$, será respectivamente 2 vezes o valor do pior caso de execução, $2 \times WCET_i$, e 2 vezes o valor do tempo de processamento da tarefa i, $2 \times c_i$.

4.2.2 Modelo de Consumo de Energia

Um processador similar ao Xscale apresentado na Tabela 5 foi utilizado. Xu, Mossé e Melhem (2007) propôs uma função polinomial para descrever o consumo energético de um processador em função da velocidade S. Esta equação é descrita como: $P = a * S^{\gamma} + p_{\text{idle}}$, no qual $\gamma(\gamma > 1)$ representa a relação energia/frequência, p_{idle} denota o consumo ocioso do processador e a reflete a capacitância de comutação efetiva. Utilizando os parâmetros do processador usado nas simulações, a função de consumo é descrita como:

$$P = 1543,28 * S^{2,83} + 63,58 \tag{4.3}$$

A energia consumida no intervalo $[t_1,t_2]$ é dada por: $E_{\mathrm{P}}(\mathrm{t}_1,\mathrm{t}_2) = \int_{t_1}^{t_2} P(S(t))dt$, onde $S(t)$ representa a velocidade do processador no tempo t. Neste modelo a energia consumida pelo sistema quando ocioso é nula.

4.2.3 Modelo de Captura de Energia

Para simular o comportamento da fonte de energia foi utilizado s o modelo já apresentado na Equação 3.2. Excluindo qualquer perda energética provocada por circuitos auxiliares, a energia capturada no intervalo de tempo $[t_1, t_2]$ é descrita como:

$$E_{\mathrm{H}}(t_1, t_2) = \int_{t_1}^{t_2} Ps(t)dt \tag{4.4}$$

Usando um dispositivo de armazenamento ideal, a energia capturada $E_{\mathrm{H}}(\mathrm{t}_1,\mathrm{t}_2)$ pode ser armazenada integralmente. O dispositivo de armazenamento usado neste trabalho tem capacidade nominal denotada por E_{C} e deve permanecer entre dois limiares: $E_{\max}$ e $E_{\min}$. Caso $E_{\mathrm{C}} < E_{\min}$ o sistema desliga devido a escassez de energia. Por outro lado, se $E_{\mathrm{C}} = E_{\max}$ a bateria é incapaz de armazenar mais energia. esta pesquisa utiliza os limiares $E_{\max} = 5,0$ e $E_{\min} = 0$ *Joules*.

4.3 Cenário de Testes

Um sistema embarcado com 3 tarefas de controle foi utilizado na simulação. A energia para alimentação do sistema é fornecida por uma bateria recarregável com capacidade máxima igual a $5,0$ *Joules*, no inicio da simulação, o nível de energia da bateria é igual a $\frac{E_{\max}}{2}$. As tarefas são escalonadas pelo algoritmo FS-EH com $Threshold = \frac{E_{\max}}{2} = 2,5$ *Joules*.

A maioria dos trabalhos nesta área usa *task-sets* sintéticos para conduzir seus experimentos (LIU et al., 2012). Este trabalho utiliza diversos *task-sets* sintéticos com 4 tarefas cada (uma tarefa de *feedback* e 3 tarefas de tempo real periódicas). O $WCET$ e T são definidos para garantir uma utilização máxima da CPU especifica ($U_{\max}$). A Equação 4.5 define a utilização máxima da CPU.

$$U_{\max} = \sum_{i=1}^{n} \frac{WCET_{\mathrm{i}}}{T_{\mathrm{i}}} \tag{4.5}$$

O $U_{\max}$ representa a utilização da CPU quando todas as tarefas executam no $WCET$. As tarefas utilizadas representam tarefas de controle e são idênticas ($\tau_1 = \tau_2 = \tau_3$).

Portanto, as tarefas apresentam o mesmo $WCET$, período, *deadlines* e prioridades. Na Tabela 11 é mostrado os conjuntos de tarefas utilizados.

Tabela 11 – Conjuntos de tarefas sintéticos

Conjunto de tarefas	Número de Tarefas	WCET (ms)	Período (ms)	Deadline (ms)	$U_{\max}$	Período tarefa *feedback* (ms)
1	3	3	30	30	0,1	200
2	3	3	20	20	0,15	200
3	3	3	15	15	0,2	200

Vale salientar que a bateria possui uma capacidade pequena de armazenamento, esta consideração é feita para representar um sistema em estado de descarga eminente. Devido a baixa capacidade da bateria, os conjuntos de tarefas possuem uma baixa utilização da CPU. A simulação termina após 50 segundos de operação do sistema. Por motivos de confiança, são feitas 33 simulações para cada conjunto de tarefas. As simulações são feitas através do SECE-CPN apresentado no capítulo 3.

4.4 Resultados e Discussões

Esta seção apresenta os resultados e discussões da pesquisa.

4.4.1 Análise do modelo de bateria não ideal

O modelo 2 utiliza um modelo de bateria eficiente similar ao usado por (LIU et al., 2012). Neste modelo, a sobrecarga causada pelas operações de carga/descarga são calculadas através de um fator de eficiência (η). Visando reduzir o custo energético do modelo de bateria proposto, o sistema é alimentado usando a energia fornecida pela fonte de energia renovável. Caso a fonte de energia consiga gerar mais energia do que o necessário para o sistema operar, a energia extra será armazenada na bateria. Por outro lado, se a fonte de energia não puder fornecer energia suficiente, a parcela de energia que não pode ser fornecida pela fonte, será fornecida pela bateria.

Considerando $E_{\mathrm{S}}(t_1, t_2)$ como a mudança da capacidade restante da bateria a partir da instância de tempo t_1 à t_2. Se $E_{\mathrm{P}}(t_1, t_2) > E_{\mathrm{H}}(t_1, t_2)$, a diferença $E_{\mathrm{P}}(t_1, t_2) - E_{\mathrm{H}}(t_1, t_2)$ deve ser retirada da bateria (descarga). Neste caso $E_{\mathrm{S}}(t_1, t_2)$ possui um valor negativo e pode ser calculado como:

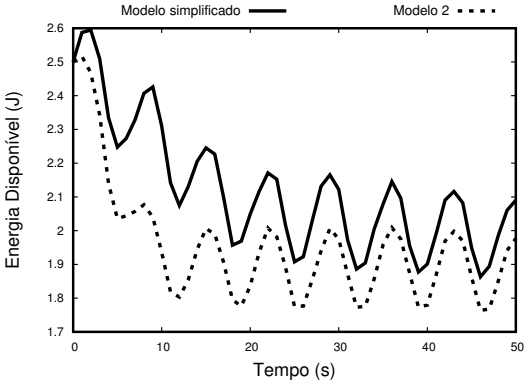
$$E_{\mathrm{S}}(t_1, t_2) = \frac{(E_{\mathrm{H}}(t_1, t_2) - E_{\mathrm{P}}(t_1, t_2))}{\eta} \tag{4.6}$$

Por outro lado, se $E_{\mathrm{P}}(t_1, t_2) < E_{\mathrm{H}}(t_1, t_2)$, a energia remanescente deve ser armazenada na bateria (carga). Neste caso, $E_{\mathrm{S}}(t_1, t_2)$ possui um valor positivo dado como:

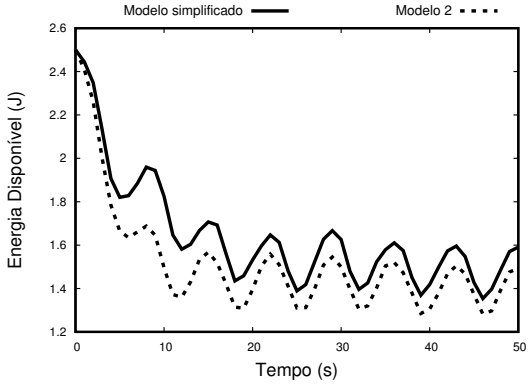
$$E_{\mathrm{S}}(t_1, t_2) = \eta(E_{\mathrm{H}}(t_1, t_2) - E_{\mathrm{P}}(t_1, t_2)) \tag{4.7}$$

Assim como o trabalho proposto por Liu et al. (2012), para os testes foi utilizado um $\eta = 90\ \%$, ou seja, a bateria consome 10 % da energia para cada operação.

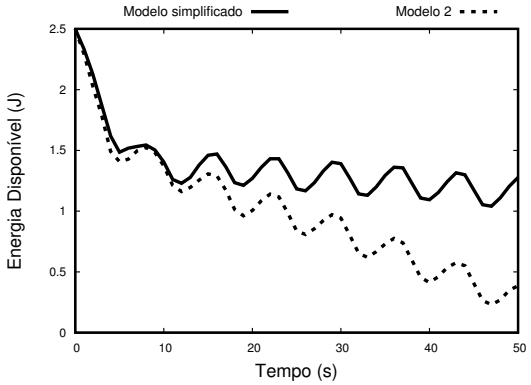
Figura 24 – Comparativo entre o modelo simplificado e o modelo 2.



(a) *task-set* 1 ($U_{\max} = 0,1$)



(b) *task-set* 2 ($U_{\max} = 0,15$)



(c) *task-set* 3 ($U_{\max} = 0,2$))

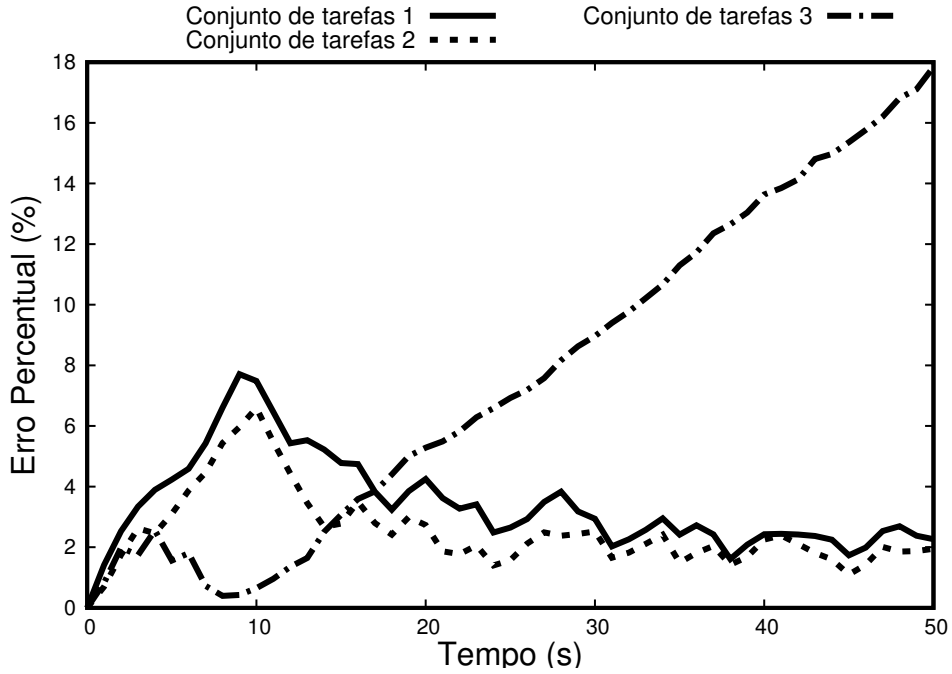
Na Figura 24 é apresenta o gráfico da energia disponível no sistema por segundo. Um grande nível de ociosidade é gerado pelo conjunto de tarefas 1, permitindo que o SCE tenha picos de carregamento. Contudo, o modelo simplificado consegue manter um estado de estabilidade no qual o nível de bateria varia entre $[1,88 - 2,17]$ Joules, enquanto que no modelo 2 o nível da bateria varia no intervalo de $[1,77 - 2,04]$ Joules.

O conjunto de tarefas 2 reflete um cenário no qual devido o montante de tarefas executadas, a energia capturada não consegue garantir altos picos de carregamento, mas é suficiente para garantir um estado de estabilidade. Se comparado com o conjunto de tarefas anterior, o conjunto 2 apresenta discrepâncias menores no nível de bateria, mais ainda visíveis. Nesse caso, após o modelo 2 atingir um estado estável de energia, o nível da bateria transita entre $[1,28 - 1,55]$ Joules, enquanto que no modelo simplificado este intervalo está entre os valores $[1,39 - 1,66]$ Joules.

Por fim, o último conjunto de tarefas apresenta estados de energia diferente para os dois modelos analisados. Enquanto o modelo simplificado apresenta uma estabilidade energética, o modelo 2 mostra um estado de escassez de energia. Nesse caso, a energia gerada pelo SCE não consegue atender a demanda energética do sistema levando ao descarregamento gradativo da bateria.

Na Figura 25 é mostrado o erro de medição por segundo. Os erros são medidos através da diferença entre o modelo simplificado e o modelo 2. Enquanto o conjunto de tarefas 1 apresenta um erro médio de 3,39 % da $E_{\max}$, o erro médio nos conjuntos 2 e 3 são de 2,5 % e 7,59 % da E_{max} respectivamente. Vale salientar que os erros apresentados no conjunto 3 tem grande variância em torno da média. Nesse conjunto de tarefas o erro entre os dois modelos pode ser de até 17,84 % da E_{max}.

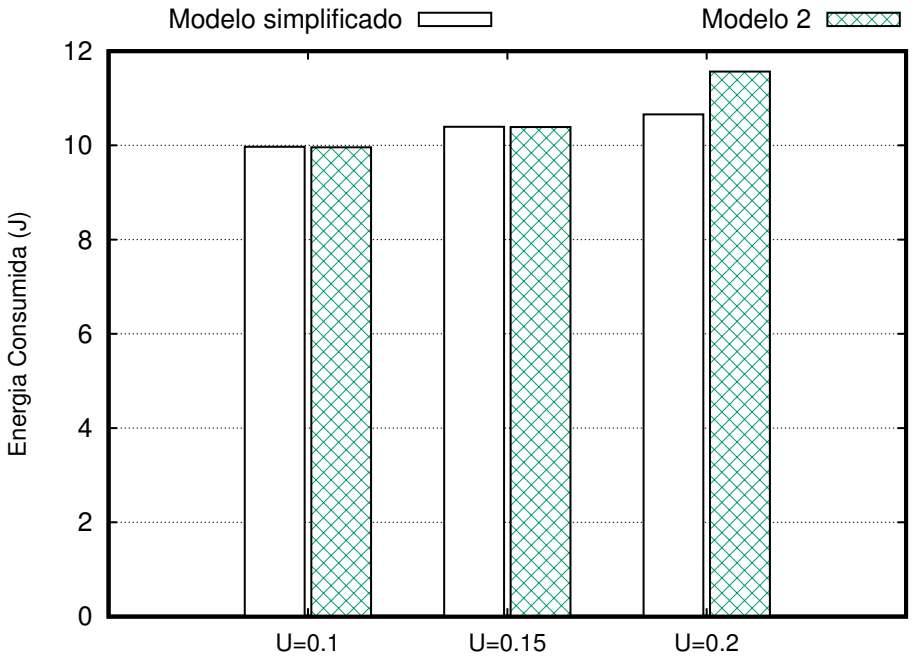
Figura 25 – Erro de medição por segundo (Modelo 2).



O uso de uma bateria não ideal tem impacto significativo na energia consumida pelo sistema. Na Figura 26 é apresentado um comparativo do consumo total do sistema durante os 50 segundos de simulação para os diferentes conjuntos de tarefas analisados.

O montante de energia consumida pelas operações de carga e descarga da bateria foram de 15,65 %, 13,31 % e 0,97 % da energia total consumida pelo sistema, para os conjuntos de tarefas 1, 2 e 3 respectivamente. Para o conjunto de tarefas 3, o modelo 2 consumiu 8,53 % mais energia que o modelo simplificado. Embora nos conjuntos 1 e 2 a energia consumida pelas operações da bateria seja significativa no consumo total do sistema, o modelo simplificado teve um consumo total compatível com o modelo 2. Isso acontece devido ao DVFS. Como no modelo 2 existe perdas referentes à bateria, a energia do sistema diminui de modo que o algoritmo de escalonamento utiliza a técnica de DVFS

Figura 26 – Consumo energético total do sistema (modelo 2).



mais vezes. No modelo simplificado, existe um maior número de tarefas sendo executado em frequências altas do que no modelo 2. Na Tabela 12 é apresentado a porcentagem de tarefas executadas por frequência de operação para os dois modelos.

Tabela 12 – Número de tarefas executadas por frequência de operação (Modelo 2)

Modelos		Tarefas Executadas			
		1000 MHz	800 MHz	600 MHz	400 MHz
Modelo simplificado	Conjunto de tarefas 1	70,17 %	29,83 %	0,0 %	0,0 %
	Conjunto de tarefas 2	7,06 %	62,93 %	30,01 %	0,0 %
	Conjunto de tarefas 3	5,11 %	8,77 %	86,02 %	0,10 %
Modelo 2	Conjunto de tarefas 1	26,95 %	73,05 %	0,0 %	0,0 %
	Conjunto de tarefas 2	6,12 %	31,35 %	63,53 %	0,0 %
	Conjunto de tarefas 3	4,39 %	4,85 %	88,35 %	2,41 %

Para o conjunto de tarefas 1, o modelo simplificado executou 70,17 % das tarefas com a máxima frequência do processador, enquanto que, no modelo 2 apenas 26,95 % das tarefas são executadas nessa frequência. Por causa disso, mesmo com a sobrecarga imposta pelas operações da bateria, o modelo 2 teve um consumo total semelhante ao modelo simplificado.

Já para o conjunto de tarefas 2, o modelo simplificado executou 62,93 % da tarefas a uma frequência de 800 MHz, enquanto que o modelo 2 executou 63,53 % das tarefas a 600MHz. Devido essa diferença de frequências, mesmo com a sobrecarga imposta pela bateria, o modelo 2 possui um consumo equivalente ao modelo simplificado.

Por fim, para o conjunto de tarefas 3, o modelo 2 consumiu 8,53 % mais energia que o modelo simplificado. Nesse caso em especifico, a técnica de DVFS foi aplicada de maneira semelhante aos dois modelos, portanto, a diferença entre o consumo total foi causada principalmente pela sobrecarga energética inserida pelas operações da bateria.

Os erros de medições inseridos pela bateria ideal pode levar os desenvolvedores a superestimar a bateria do sistema. Embora que algumas vezes a sobrecarga energética causada pelas operações de carga/descarga da bateria não modifique o consumo total do sistema, essa sobrecarga ainda interfere no nível da bateria em todos os conjuntos de tarefas analisados (ver Figura 24) podendo levar a erros de projetos referentes ao dimensionamento da bateria. Vale salientar que baterias com uma η menor podem inserir erros ainda maiores.

4.4.2 Análise do modelo de consumo ocioso

O processador precisa de energia mesmo quando não está executando nenhuma tarefa, o modelo 3 considera o consumo ocioso do processador. O XScale, processador utilizado neste estudo, tem consumo ocioso de $63,58\ mW$ (ver Tabela 5)

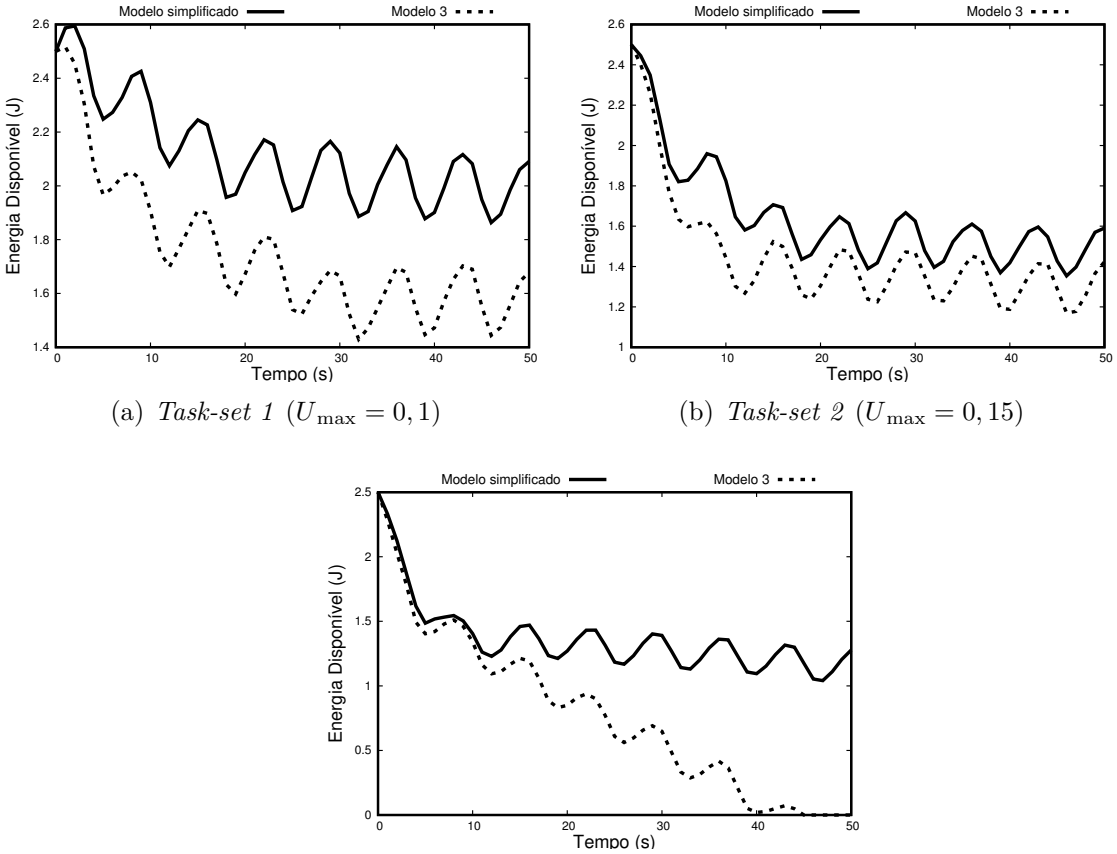
O consumo ocioso será descontado da energia capturada pelo SCE. Considere um sistema que esteja ocioso no intervalo de tempo $[t_1, t_2]$, o consumo ocioso é representado por $E_i(t_1, t_2)$. A energia capturada efetiva do sistema é dada por: $E_{\text{EH}}(t_1, t_2) = E_H(t_1, t_2) - E_i(t_1, t_2)$. Caso $E_H(t_1, t_2) > E_i(t_1, t_2)$, a $E_{\text{EH}}(t_1, t_2)$ tem valor positivo e deve ser armazenada na bateria. Se $E_H(t_1, t_2) < E_i(t_1, t_2)$, a $E_{\text{EH}}(t_1, t_2)$ é negativa e deve ser retirada da bateria. Finalmente, se $E_H(t_1, t_2) = E_i(t_1, t_2)$, o sistema está em um estado de *energy neutral*. Vale salientar que $E_H(t_1, t_2)$, assim como $E_i(t_1, t_2)$ é sempre maior ou igual a 0.

Na Figura 27 é feito um comparativo entre a energia disponível por segundo apresentada pelo modelo simplificado e o modelo 3. Devido o grande nível de ociosidade apresentado pelo conjunto de tarefas 1, o consumo ocioso do processador causa diferenças significativas no nível da bateria. Embora os dois modelos consigam alcançar um estado de estabilidade do sistema, no final da simulação a diferença entre os níveis de bateria dos dois modelos é de aproximadamente 8,24% da $E_{\max}$.

Já no conjunto de tarefas 2, devido a uma maior utilização da CPU, o consumo ocioso teve um impacto menor no resultado da simulação se comparado com o cenário acima. Contudo ainda houve discrepâncias entre o nível de bateria nos dois modelos. Enquanto o modelo 3 consegue manter um estado de estabilidade no qual o nível da bateria varia entre $[1,42 - 1,80]$ Joules, no modelo simplificado a bateria varia entre $[1,86 - 2,17]$ Joules.

Finalmente, o conjunto de tarefas 3 é o mais afetado por esta simplificação. A

Figura 27 – Comparativo entre o modelo simplificado e o modelo 3.



(a) *Task-set 1* ($U_{\max} = 0,1$)

(b) *Task-set 2* ($U_{\max} = 0,15$)

(c) *Task-set 3* ($U_{\max} = 0,2$)

utilização da CPU e a ociosidade do processador possuem um relacionamento inversamente proporcional, isto é, quanto maior a utilização da CPU, menor a ociosidade do processador. Contudo, neste caso a energia consumida pelo processador quando ocioso impossibilita o sistema de alcançar um estado estável de energia levando ao descarregamento total da bateria aos 45 segundos de simulação.

Na Figura 28 é mostrado o erro percentual entre o modelo simplificado e o modelo 3. Devido a baixa utilização da CPU, o conjunto de tarefas 3 teve erros de medição de até 25,55% da E_{max}, enquanto que os conjuntos 1 e 2 apresentam erros de até 9,75% e 17% da E_{max} respectivamente. Embora o conjunto de tarefas 3 tenha a maior utilização da CPU dentre os conjuntos analisados, o consumo ocioso produziu um impacto menor utilizando o conjunto de tarefas 2. Esse caso pode ser explicado devido a estabilidade do sistema alcançada ao utilizar esse conjunto. Enquanto que o conjunto de tarefas 3 leva o sistema a um descarregamento total, o conjunto 2 fornece um estado de estabilidade levando a um erro quase constante durante toda simulação.

A energia gasta pelo processador quando ocioso também causa impactos significativos no consumo total do sistema. Na Figura 29 é mostrado um comparativo do consumo

Figura 28 – Erro de medição por segundo (Modelo 3)

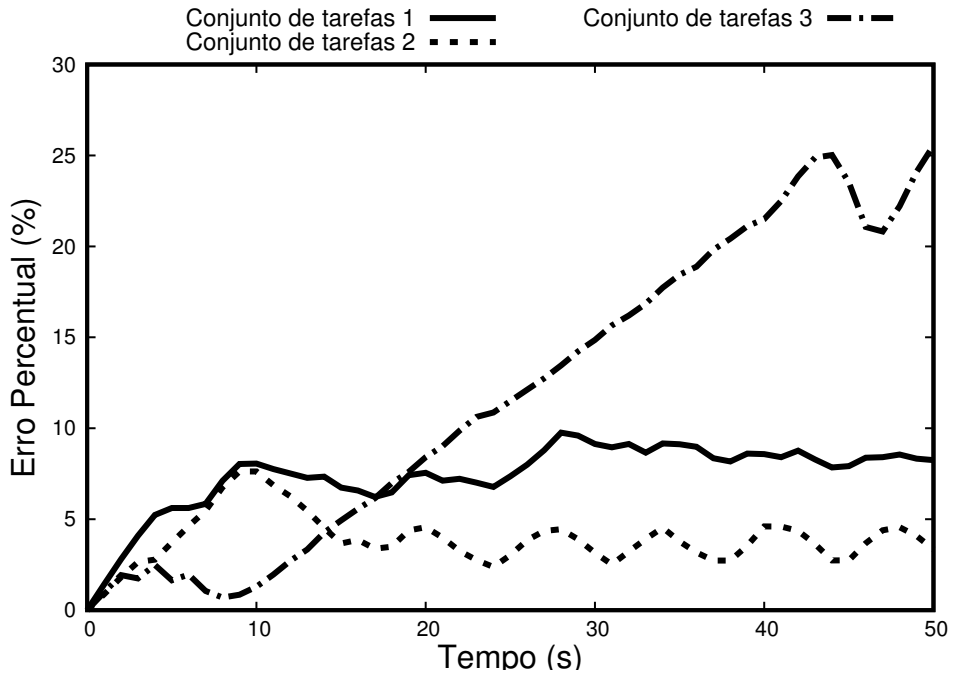
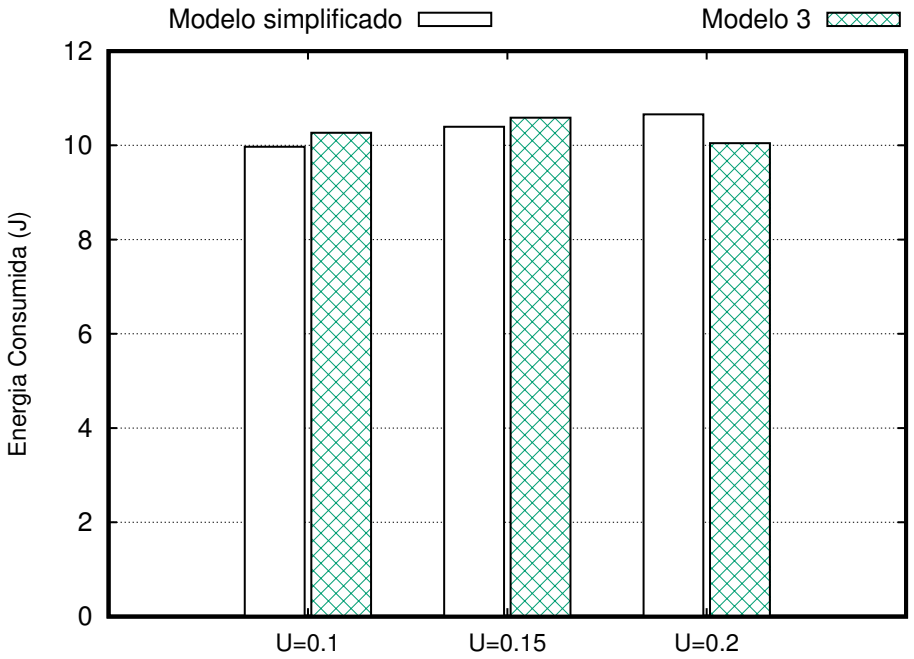


Figura 29 – Consumo energético total do sistema (modelo 3)



de energia nos dois modelos analisados.

O modelo 3 utilizando o conjunto de tarefas 1 passa cerca de 84,00% do tempo de simulação ocioso, consumindo um montante de 2,69J para se manter operante. Esse valor representa aproximadamente 53,89% da E_{max}. Já para o conjunto de tarefas 2, o modelo 3 fica aproximadamente 68,00% do tempo ocioso, a quantidade de energia consumida pelo processador quando ocioso foi de 44,45% da E_{max}. Por fim, para o conjunto de tarefas 3, o

modelo 3 passa aproximadamente 56,09% do tempo operante ocioso, durante esse tempo o processador consumiu cerca de 30,02% da E_{max}. Vale salientar que para o conjunto de tarefas 3, o modelo 3 se mantém operante durante 41 segundos.

Embora o processador consuma um montante de energia significativo quando ocioso, o consumo total do sistema é semelhante nos dois modelos utilizados. Novamente o DVFS é responsável por essa pequena diferença no consumo total. Devido o consumo do processador quando ocioso, a energia disponível no modelo 3 diminui de maneira mais acelerada do que no modelo simplificado, o escalonador, por sua vez, tenta aumentar a eficiência energética do sistema através da técnica de DVFS. Utilizando o conjunto de tarefas 1, o modelo 3 executa 13,59% das tarefas na frequência máxima, 72,77% a 800 MHz e 13,64% a 600 MHz. Já para o conjunto de tarefas 2, 6,04% das tarefas são executadas a 1000 MHz, enquanto que 15,13% e 78,83% das tarefas são executadas a 800 e 600 MHz respectivamente. Por fim no conjunto de tarefas 3, o modelo 3 executa 5,37%, 5,44%, 87,29% e 1,90% das tarefas utilizando as frequências de 1000, 800, 600 e 400 MHz respectivamente. A Tabela 13 mostra a porcentagem de tarefas executadas por frequência de operação.

Tabela 13 – Número de tarefas executadas por frequência de operação (Modelo 3)

Modelos		Tarefas Executadas			
		1000 MHz	800 MHz	600 MHz	400 MHz
Modelo simplificado	Conjunto de tarefas 1	70,17 %	29,83 %	0,00 %	0,00 %
	Conjunto de tarefas 2	7,06 %	62,93 %	30,01 %	0,00 %
	Conjunto de tarefas 3	5,11 %	8,77 %	86,02 %	0,10 %
Modelo 3	Conjunto de tarefas 1	13,59 %	72,77 %	13,64 %	0,00 %
	Conjunto de tarefas 2	6,04 %	15,13 %	78,83 %	0,00 %
	Conjunto de tarefas 3	5,37 %	5,44 %	87,29 %	1,90 %

O modelo 3 teve um consumo 2,98 % e 1,83 % maior do que o modelo simplificado, para os conjuntos 1 e 2 respectivamente. Já para o conjunto de tarefas 3, o modelo simplificado consumiu 6,08 % mais energia. Contudo, para esse conjunto de tarefas, o modelo 3 ficou operante apenas por 44 segundos.

Vale salientar que o modelo simplificado e o modelo 3 apresentam o mesmo comportamento no inicio da simulação. Contudo, após alguns segundos, o somatório da energia gasta pelo processador quando ocioso causa erros de medição significativos. Também é importante notar que conforme U_{max} aumenta, a ociosidade do sistema diminui. Nesse caso podemos usar a utilização instantânea da CPU como indicativo para negligenciar ou não o consumo ocioso do processador.

4.4.3 Análise do modelo com troca de contexto

As tarefas possuem um estado interno bem definido que representa sua situação atual: parte do código que está sendo executado, variáveis e arquivos utilizados, dentre

outras. O estado de uma tarefa em um determinado instante é chamado de contexto. Sempre que uma tarefa começa sua execução seu contexto deve ser carregado no sistema, quando uma tarefa é interrompida por outra de maior prioridade, o estado interno da tarefa deve ser salvo e o contexto da tarefa prioritária precisa ser carregado, essa operação é chamada de troca de contexto.

Existem duas formas para que ocorra a troca de contexto no modelo em questão: Uma tarefa é interrompida por outra de maior prioridade. Nesse caso as informações da tarefa interrompida devem ser salvas e as da nova tarefa devem ser carregadas, ou, uma tarefa vai começar sua execução em um processador ocioso. Nesse caso o contexto da nova tarefa deve ser carregado.

Para os testes, foi considerado que o sistema demora 20 μs e consome 20 μJ de energia para realizar a troca de contexto das tarefas. Na Figura 30 é feito um comparativo entre a energia disponível por segundo apresentada pelo modelo simplificado e o modelo 4. A troca de contexto não teve impacto no resultado da simulação em nenhum dos 3 conjuntos de tarefas, podendo ser completamente negligenciada do cenário de simulação. O total de energia consumida pela troca de contexto foi de 0,21 %, 0,32 % e 0,42 % da E_{max} respectivamente para os conjunto de tarefas 1, 2 e 3. Podemos observar que conforme a U_{max} aumenta, a energia consumida pela troca de contexto também cresce, isso acontece devido o aumento de instâncias das tarefas no sistema. Enquanto no conjunto de tarefas 1 foram executadas em média 5108 tarefas em 50 segundos, para o conjunto de tarefas 2 e 3 o montante de tarefas executadas foram de 7654 e 10207 respectivamente.

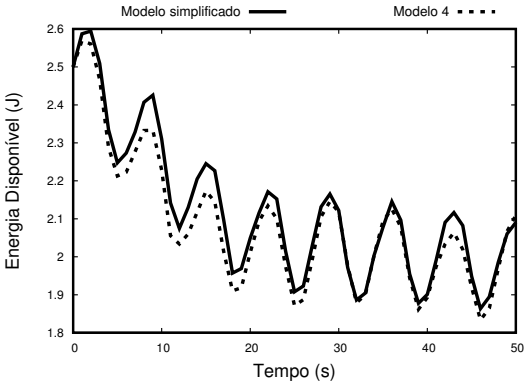
Tabela 14 – Novos conjuntos de tarefas sintéticos

Conjunto de tarefas	Número de Tarefas	WCET (ms)	Período (ms)	Deadline (ms)	$U_{\max}$	Período tarefa *feedback* (ms)
4	3	0,3	3,0	3,0	0,1	200
5	3	0,3	2,0	2,0	0,15	200
6	3	0,3	1,5	1,5	0,2	200

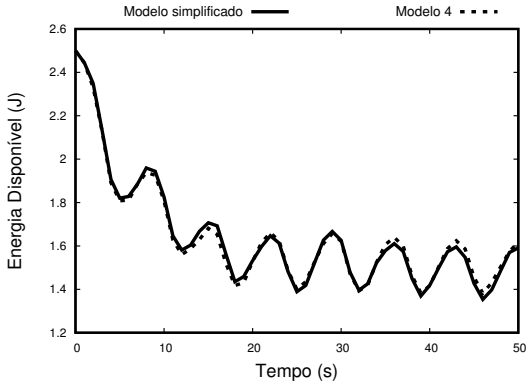
Diferente do consumo ocioso, a utilização máxima da CPU não tem influência na energia gasta pela troca de contexto. Podemos ter conjuntos de tarefas compostos por tarefas relativamente longas e períodos apertados. Por exemplo, considere um conjunto de tarefas composto por 1 única tarefa τ_i com $WCET_i = 24$ segundos e $T_i = D_i = 25$ segundos. Para esse conjunto de tarefas $U_{max} = 0,96$, contudo, caso essa tarefa execute no pior caso, durante os 50 segundos será realizada apenas duas trocas de contexto.

Para demostrar cenários onde a troca de contexto pode ser impactante dividimos o $WCET$ e período dos conjuntos de tarefas 1, 2 e 3 por um fator de 10 criando assim novos conjuntos de tarefas. Na Tabela 14 é apresentado os novos conjuntos criados. Note que a U_{max} continua a mesma que o grupo de tarefas anterior, contudo, as tarefas possuem

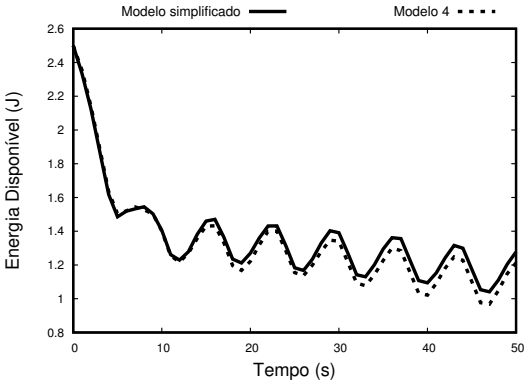
Figura 30 – Comparativo entre o modelo simplificado e o modelo 4.



(a) *Task-set 1* ($U_{\max} = 0,1$)



(b) *Task-set 2* ($U_{\max} = 0,15$)



(c) *Task-set 3* ($U_{\max} = 0,2$)

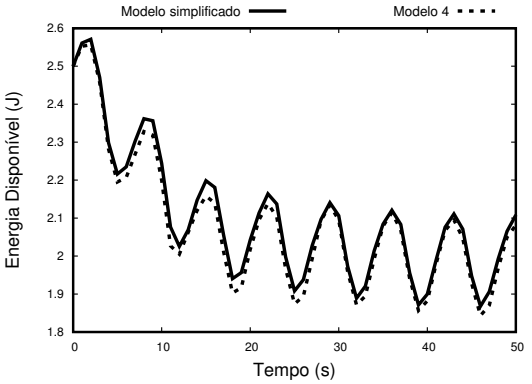
um menor tempo de processamento e período, provocando um aumento no número de instâncias de tarefas que será executado.

Foram refeitos os testes utilizando os conjuntos de tarefas 4, 5 e 6, na Figura 31 é apresentado um comparativo entre o modelo simplificado e o modelo 4 utilizando os novos conjuntos.

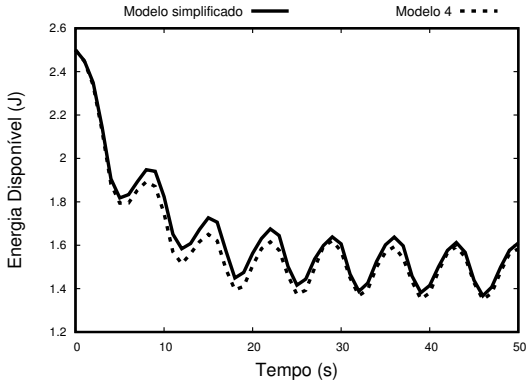
Não houve diferenças significativas no consumo do modelo simplificado em relação aos conjuntos de tarefas antigos. Embora exista um aumento de tarefas executadas nos novos conjuntos, essas tarefas são menores e consomem menos energia. Contudo, utilizando os novos conjuntos de tarefas, o modelo 4 apresentou diferenças em relação ao modelo simplificado. Na Tabela 15 é apresentado o número médio de tarefas executadas ao longo de 50 segundos de simulação para todos os conjuntos de tarefas analisados.

O aumento de tarefas processadas leva a uma maior troca de contexto. Enquanto o conjunto de tarefas 4 executou cerca de 51004 tarefas ao longo de 50 segundos, o conjunto de tarefas 5 processou cerca de 76503 tarefas. Embora o número de tarefas executadas seja maior se comparado com os conjuntos 1 e 2, a troca de contexto para os conjuntos 4 e 5 não apresentou um *overhead* energético significante.

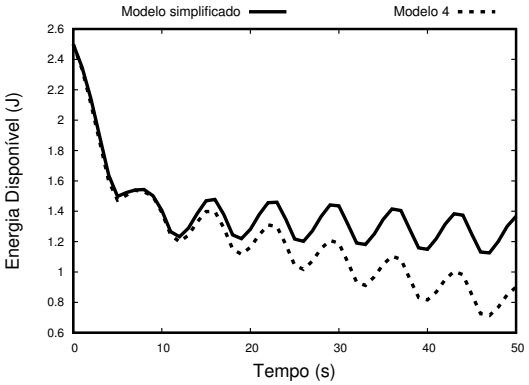
Figura 31 – Comparativo entre o modelo simplificado e o modelo 4 utilizando os novos conjuntos de tarefas.



(a) *Task-set 4* ($U_{\max} = 0,1$)



(b) *Task-set 5* ($U_{\max} = 0,15$)



(c) *Task-set 6* ($U_{\max} = 0,2$)

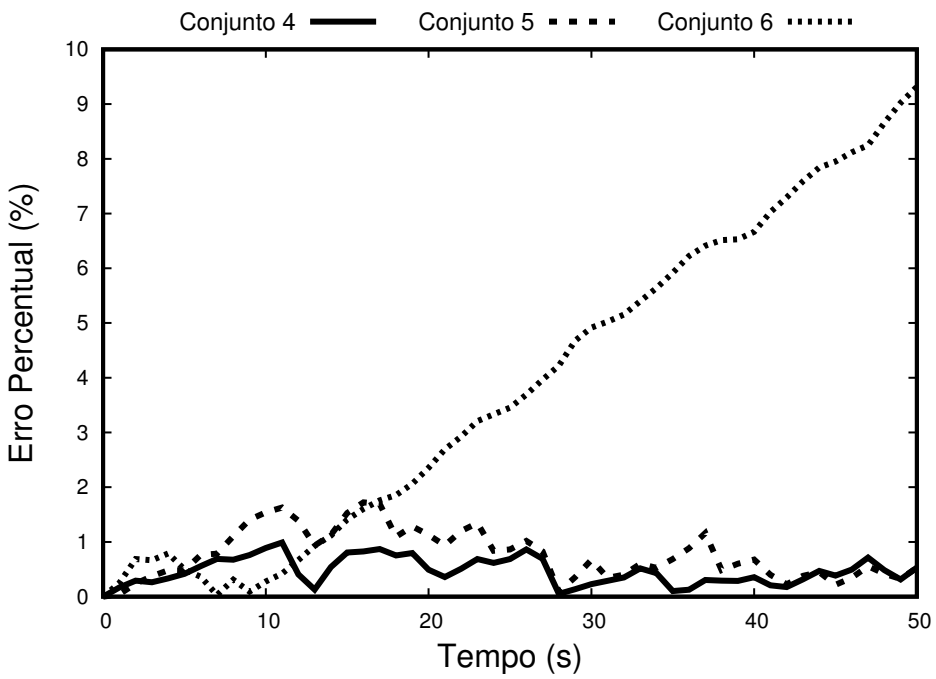
Tabela 15 – Número médio de tarefas executadas em 50 segundos

Conjunto de tarefas	U_{max}	Número de tarefas executadas
1	0,1	5108
2	0,15	7654
3	0,2	10167
4	0,1	51004
5	0,15	76503
6	0,2	102004

Foram executadas em média 102004 tarefas utilizando o conjunto 6, neste conjunto o modelo simplificado apresentou um estado energético estável, enquanto que no modelo 4, a sobrecarga energética produzida pela troca de contexto levou o sistema a um estado de descarregamento gradativo.

Na Figura 32 é apresentado o erro percentual médio em relação a E_{max} por segundo.

Figura 32 – Erro percentual por segundo (modelo 4)

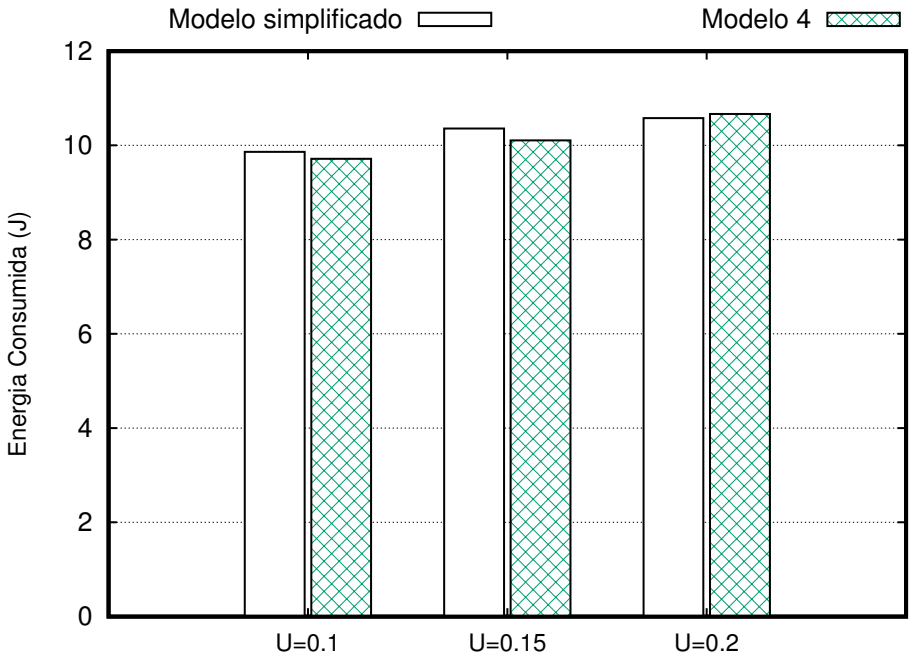


A energia consumida pela troca de contexto apresentou maior impacto no conjunto de tarefas 6, nesse caso o modelo 4 produziu uma diferença de até 9,31 % da E_{max}, enquanto que a diferença máxima mostrada pela utilização do conjunto de tarefas 4 e 5 foi de 0,99 % e 1,72 % respectivamente. Embora o conjunto 5 apresente mais trocas de contexto do que o conjunto 4, a troca de contexto teve um menor impacto nesse conjunto. O conjunto de tarefas 5 consegue alcançar um estado estável de energia levando a variações de erro menores.

Na Figura 33 é mostrado um comparativo do consumo total do sistema nos 3 novos conjuntos de tarefas analisados.

Para os conjuntos de tarefas 4 e 5, o modelo 4 consumiu cerca de 2,05 % e 3,07 % da E_{max} respectivamente, somente com a troca de contexto. Contudo, esta energia consumida não representa um impacto significante no consumo geral do sistema. Já para o conjunto de tarefas 6, o modelo 4 gastou aproximadamente 4,09 % da E_{max} com a troca de contexto. Neste caso em específico, a sobrecarga energética causada pela troca de contexto é suficiente para retirar o sistema do estado estável de energia apresentado pelo modelo simplificado. Contudo, o consumo total do sistema em ambos os modelos continuaram similar.

Figura 33 – Consumo energético do sistema (modelo 4)



Note que para tarefas com pouco tempo de execução e período curto, a troca de contexto não deve ser negligenciada, pois a sobrecarga energética gerada pelo alto números de instâncias de tarefas pode ser considerável. Contudo, para tarefas com longos tempos de execução e períodos longos, a troca de contexto pode não impactar no resultado da simulação.

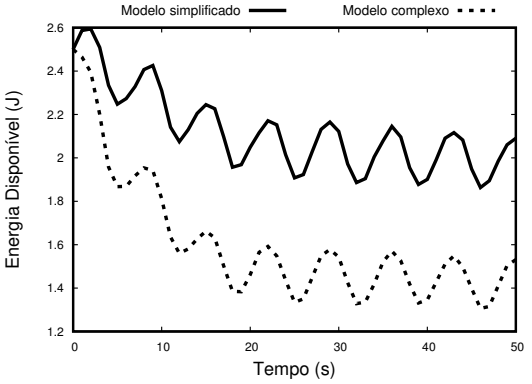
4.4.4 Análise combinada das simplificações

Para analisarmos o efeito combinado das simplificações no cenário de teste utilizamos um modelo complexo que leva em consideração a energia gasta pelo processador quando ocioso, a troca de contexto das tarefas, além de utilizar um modelo de bateria não ideal. Foi feito um comparativo entre esse modelo e o modelo simplificado utilizando o primeiro grupo de conjunto de tarefas. Na Figura 34 é apresentado o consumo energético ao longo do tempo nos dois modelos.

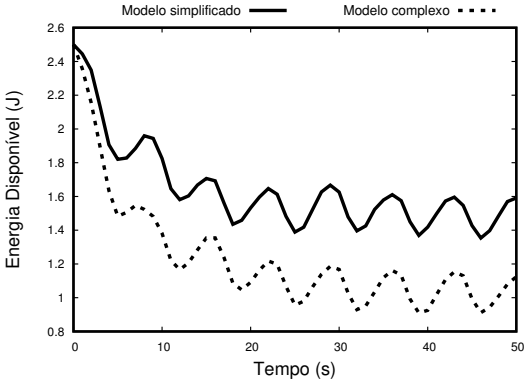
As simplificações estudadas causam uma diferença energética considerável entre os dois modelos. O modelo simplificado consegue estabilizar a energia do sistema com qualquer um dos 3 conjuntos utilizados. Contudo, o modelo complexo não consegue garantir a eficiência energética do sistema para o conjunto de tarefas 3. Nesse caso em específico, o consumo negligenciado pelo modelo simplificado leva o sistema a um estado de descarregamento total aos 25 segundos de simulação. A Figura 35 mostra o erro de medição ao longo do tempo para os 3 conjuntos analisados.

Devido ao descarregamento do sistema no modelo complexo, o conjunto de tarefas 3 apresenta os maiores erros de medição com diferenças de até 28,05 % da E_{max}. Assim como

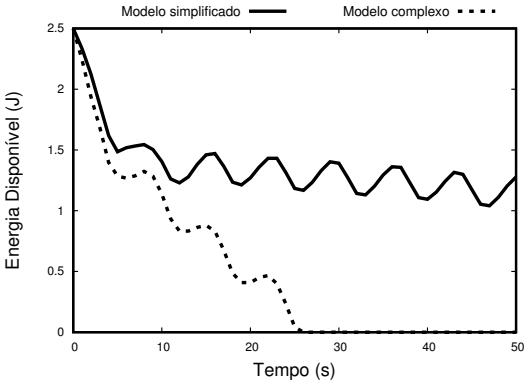
Figura 34 – Comparativo entre o modelo simplificado e o modelo 5.



(a) Conjunto de tarefas 1 ($U_{\max} = 0,1$)

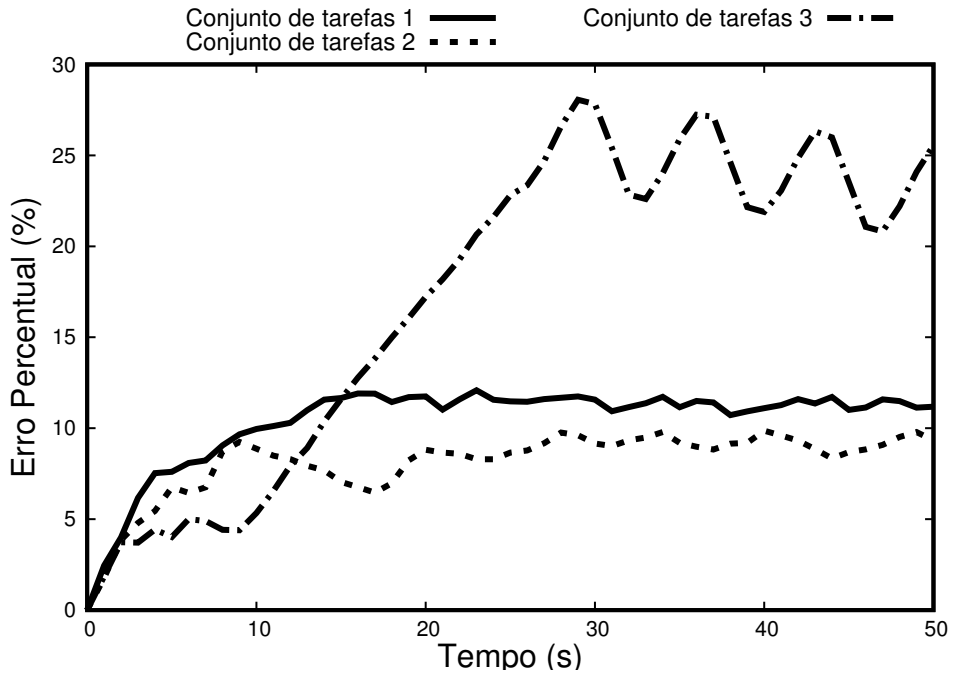


(b) Conjunto de tarefas 2 ($U_{\max} = 0,15$)



(c) Conjunto de tarefas 3 ($U_{\max} = 0,2$)

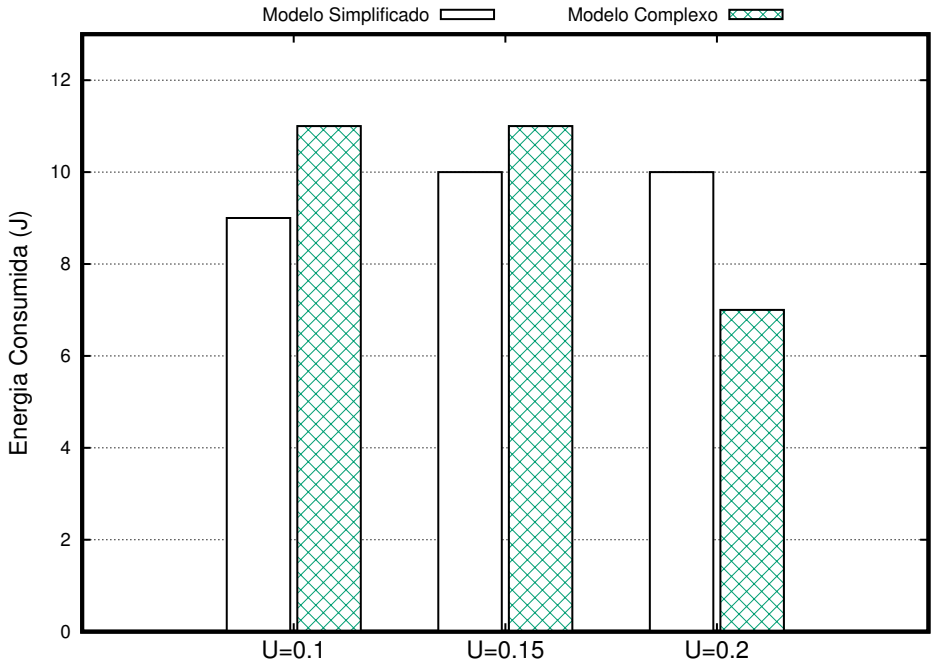
Figura 35 – Erro percentual por segundo (modelo complexo)



o modelo simplificado, o modelo complexo alcança um estado de estabilidade energética para os conjuntos de tarefas 1 e 2, criando um erro quase constante ao longo do tempo. Contudo, devido a alta ociosidade presente no conjunto de tarefas 1, as perdas energéticas da bateria e o consumo do processador quando ocioso causam um erro maior entre os modelos. Para esses dois conjuntos de tarefas o erro máximo encontrado foi de 12,08 % e 9,96 % da E_{max} respectivamente.

As simplificações tem impacto importante no consumo energético do sistema. Na Figura 36 é apresentado um comparativo do consumo total durante os 50 segundos de simulação para os diferentes conjuntos de tarefas analisados.

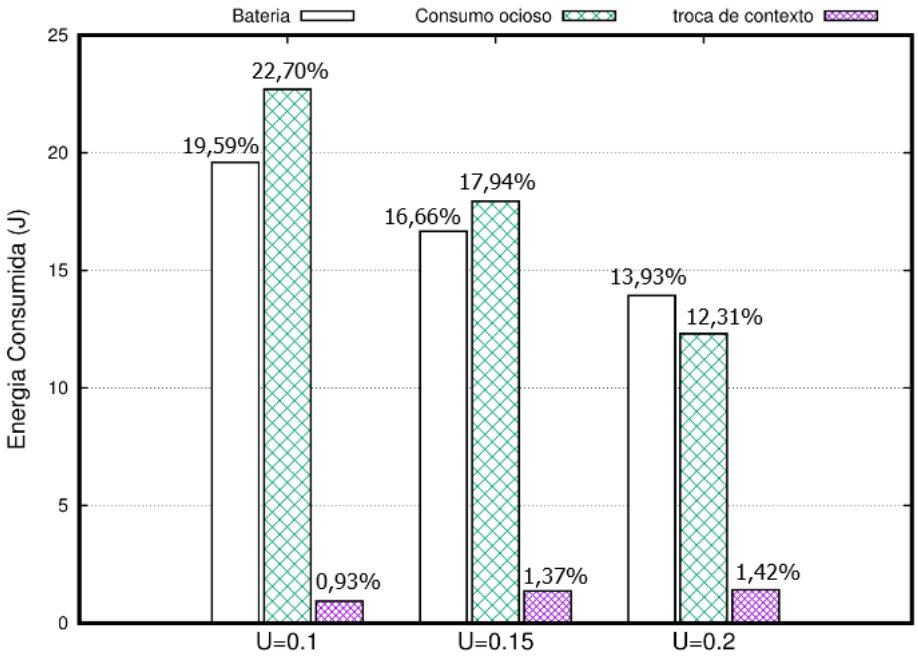
Figura 36 – Consumo energético total do sistema (modelo 5)



Para o conjunto de tarefas 1, comparado com o modelo simplificado, o modelo complexo teve um aumento de 15,92 % no consumo energético. Já no conjunto de tarefas 2, o modelo complexo consumiu 12,86 % mais energia do que o simplificado. Por fim, no conjunto de tarefas 3 o modelo simplificado consumiu 44,33 % mais energia que o modelo complexo. Contudo, nesse caso em especifico, o modelo simplificado ficou operante durante toda a simulação, enquanto que o modelo complexo descarregou aos 25 segundos de simulação. Para o conjunto de tarefas 1, 2 e 3, a energia gasta pelo sistema sem está executando tarefas, ou seja, apenas com troca de contexto, perdas na bateria e consumo ocioso, foi respectivamente 43,22 %, 35,97 % e 27,66 % da energia total consumida. Na Figura 37 é apresentado o gasto energético negligenciado pelo uso das simplificações estudadas

Para os casos testados nessa seção, negligenciar a troca de contexto é a única simplificação justificável, todas as outras tem impactos significativos no consumo total do

Figura 37 – Energia negligenciada pelas simplificações em relação ao consumo total.



sistema levando a erros de medições consideráveis entre os dois modelos.

4.5 Conclusão

Neste capítulo foi apresentado um estudo dos impactos energéticos causados por simplificações no cenário de teste de sistemas embarcados com captura de energia. Este estudo abordou 3 simplificações comumente usadas nas simulações desses tipos de sistemas: Uso de uma bateria ideal, consumo neutro quando o processador está ocioso e troca de contexto sem sobrecargas energéticas. A magnitude dos erros causados por estas simplificações é medida através de um comparativo de vários modelos de simulação com diferentes níveis de detalhes. Os resultados mostram que caso combinadas, as simplificações podem acarretar em erros de medição de até 28,08 % da capacidade máxima da bateria utilizada, além de induzir os desenvolvedores a conclusões otimistas que não refletem a realidade.

Desconsiderar a troca de contexto foi a simplificação que teve o menor impacto energético entre as simplificações estudadas neste trabalho. Contudo, em casos no qual o tempo de computação e o período das tarefas são curtos, esta simplificação pode produzir um *overhead* considerável. Por outro lado, o consumo neutro do processador quando este está ocioso foi a simplificação que introduziu os maiores erros de medição com diferenças de até 25,55 % da capacidade máxima da bateria. Também foi identificada uma relação inversamente proporcional entre o uso máximo da CPU e o consumo do processador quando o sistema estava ocioso. No entanto, a técnica DVFS pode diminuir o consumo do sistema, além disso, pode aumentar o tempo de execução das tarefas reduzindo a

ociosidade do sistema. O uso de uma bateria ideal também teve um impacto significante em todos os conjuntos de tarefas analisados, em casos específicos, 15,65 % de toda a energia consumida pelo sistema foi gasta pelas operações de carga e descarga da bateria. Também foi identificado um relacionamento proporcional entre a energia gasta com as operações da bateria e a frequência operacional do processador. Frequências menores implicam em menor consumo e ociosidade, portanto, as perdas da bateria são menores do que quando o sistema opera em altas frequências.

Embora mesmo que a sobrecarga energética introduzida pela retirada das simplificações seja significativa, o consumo total do sistema permanece semelhante se comparado com o modelo simplificado. Este caso pode ser explicado devido à diminuição acelerada do nível da bateria causada pela remoção das simplificações. O escalonador de tarefas tenta aumentar a eficiência energética por meio da técnica de DVFS. Assim, a energia consumida pela retirada das simplificações é compensada pela técnica DVFS, trazendo os dois modelos para um consumo similar.

Os resultados apresentados neste estudo levam a 3 conclusões principais. Primeiro, a quantidade de energia negligenciada pelo uso de cenários de simulação simplificados pode levar os desenvolvedores a conclusões equivocadas. Segundo, sistemas embarcados com captura de energia são adaptativos e aplicam a técnica DVFS para aumentar sua eficiência energética, portanto, comparar o consumo total do sistema entre modelos complexos e simplificados pode não ser uma abordagem eficiente para medir os impactos das simplificações. Finalmente, a relação entre as simplificações e os erros de medição não é uniforme ou estável entre os experimentos, portanto, não é possível aplicar um fator corretivo constante para compensar erros decorrentes de simplificações.

5 Conclusões e Perspectivas

Com o objetivo de permitir a representação e análise de sistemas embarcados com captura de energia, além de descrever a sobrecarga energética e temporal causada pelo uso das técnicas de gerenciamento de energia e escalonamento de tarefas, neste trabalho apresentamos a ferramenta de simulação baseada em redes de Petri coloridas SECE-CPN.

O SECE-CPN disponibiliza para seus usuários diferentes modelos de fontes de energia renováveis (solar, eólica, cinética, dentre outras), dois modelos de processadores, 4 algoritmos de escalonamento e alguns conjuntos de tarefas predefinidos. Por fim, ao final de cada simulação a ferramenta disponibiliza uma série de métricas para analise dos sistemas simulados. O SECE-CPN pode ser estendido através de uma Interface de Programação Multilinguagem (IPML) proposta por JÚNIOR (2016), Com a IPML é possível adicionar outros processadores, algoritmos de escalonamento e fontes de energia, além de, criar outras métricas de análise sem a necessidade de modificação na estrutura do simulador.

Para validação do SECE-CPN foi reproduzido o modelo proposto por Abbas et al. (2014), os resultados mostraram que o SECE-CPN apresenta resultados compatíveis com o trabalho em questão.

Utilizando o SECE-CPN foi feito um estudo de caso do impacto que cenários de simulação simplificados podem causar na análise de sistemas com captura de energia. Foram analisados 3 fatores comumente negligenciados no cenário de teste, estes fatores são: a energia dissipada nas operações de carga e descarga da bateria, o consumo ocioso do processador e a troca de contexto entre as tarefas que devem ser processadas pelo sistema. Os resultados mostram que combinadas as simplificações acarretaram em erros de medição de até 28,08 % da capacidade máxima da bateria utilizada.

Negligenciar a sobrecarga energética causada pela troca de contexto foi a simplificação que teve o menor impacto energético entre as simplificações estudadas neste trabalho sendo completamente justificável em alguns casos. Por outro lado, o uso de uma bateria ideal, assim como, desconsiderar o consumo do processador quando ocioso tiveram impactos significantes em todos os casos testados. Contudo, embora as simplificações tenham introduzido erros significativos, devido a técnica de DVFS, em alguns casos, o consumo total do sistema permaneceu semelhante ao modelo simplificado. portanto, comparar o consumo total do sistema entre modelos complexos e simplificados pode não ser uma abordagem eficiente para medir os impactos das simplificações. Por fim, o estudo apresentado levou a conclusão de que, embora a sobrecarga energética adicionada pelo uso de cenários complexos seja pequena, o somatório desta sobrecarga pode causar diferenças significantes ao longo da simulação podendo levar a conclusões equivocadas dos projetistas

de sistemas com captura de energia.

Como trabalhos futuros, sugere-se os seguintes temas:

- Estudar a viabilidade de novas técnicas de gerenciamento de energia para esses tipos de sistemas.
- Estudar os diversos escalonadores de tarefas para esses tipos de sistemas sob a perspectiva de diferentes fontes de energia.
- Comparar um modelo no SECE-CPN com um sistema embarcado com captura de energia real.

Referências

ABBAS, A. et al. A real-time feedback scheduler for environmental energy harvesting. In: **3rd International Conference on Systems and Control**. [S.l.: s.n.], 2013. p. 1013–1019. ISSN 2379-0059.

ABBAS, A. et al. A real-time feedback scheduler for environmental energy with discrete voltage/frequency modes. In: **2014 International Conference on Future Internet of Things and Cloud**. [S.l.: s.n.], 2014. p. 458–463.

BEEBY, S.; WHITE, N. **Energy Harvesting for Autonomous System**. Norwood: Artech House, 2010.

CARRO, F. R. W. L. Jornadas de atualização em informática, sbc. In: **Sistemas Computacionais Embarcados**. [S.l.: s.n.], 2003.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to Discrete Event Systems**. 2. ed. [S.l.]: Springer, 2008. ISBN 9780387686127.

CERVIN, A., HENRIKSSON, D. AND OHLIN, M. **TrueTime 2.0 - Reference Manual**. 2016. Disponível em: <http://www.control.lth.se/truetime.> Acesso em: nov. 2017.

CHEN, G. et al. Effective online power management with adaptive interplay of dvs and dpm for embedded real-time system. In: **2013 Euromicro Conference on Digital System Design**. [S.l.: s.n.], 2013. p. 881–889.

Chetto, M. Optimal scheduling for real-time jobs in energy harvesting computing systems. **IEEE Transactions on Emerging Topics in Computing**, v. 2, n. 2, p. 122–133, June 2014. ISSN 2168-6750.

CPN Tools. **CPN Tools Homepage**. 2018. Disponível em: <http://cpntools.org/>. Acesso em: junho de 2019.

Dang, N. et al. Harvesting-aware adaptive energy management in solar-powered embedded systems. In: **2016 17th International Symposium on Quality Electronic Design (ISQED)**. [S.l.: s.n.], 2016. p. 331–337. ISSN 1948-3295.

ECLIPSE. **Eclipse Foundation: The Platform for Open Innovation and Collaboration**. 2018. Disponível em: <http://eclipse.org/>. Acesso em: junho de 2019.

GHADAKSAZ, E.; SAFARI, S. Storage capacity for edf–asap algorithm in energy-harvesting systems with periodic implicit deadline hard real-time tasks. **Journal of Systems Architecture**, v. 89, p. 10 – 17, 2018. ISSN 1383-7621. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1383762117303739>.

GHOR, H. E.; CHETTO, M.; CHEHADE, R. H. A real-time scheduling framework for embedded systems with environmental energy harvesting. **Comput. Electr. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 37, n. 4, jul 2011. ISSN 0045-7906.

IEA. **Key World Energy Statistics**. [S.l.], 2018. Disponível em: <https://webstore.iea.org/download/direct/2291?fileName=Key_World_2018.pdf>.

INTEL. **Intel XScale Microarchitecture**. 2007. Disponível em: <http://developer.intel.com/design/intelxscale/2007.> Acesso em: fev. 2018.

International Energy Agency. **Renewables Information**. [S.l.], 2018. Disponível em: <https://webstore.iea.org/download/direct/2291?fileName=Key_World_2018.pdf>.

JENSEN, K.; KRISTENSEN, L. M. **Coloured Petri Nets: Modelling and Validation of Concurrent Systems.** 1th. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2009.

Jun Lu et al. Accurate modeling and prediction of energy availability in energy harvesting real-time embedded systems. In: **International Conference on Green Computing**. [S.l.: s.n.], 2010. p. 469–476.

JÚNIOR, C. G. F. **CONTRIBUIÇÕES À MODELAGEM E À SIMULAÇÃO DE SISTEMAS DISTRIBUÍDOS, USANDO REDES DE PETRI COLORIDAS – ALOCAÇÃO DE WEBCACHES E PARTICIONAMENTO DE MODELOS EM PROCESSOS INDEPENDENTES**. Tese (Doutorado) — Universidade Federal do Ceará, 2016. Disponível em: <http://www.repositorio.ufc.br/handle/riufc/24936>.

KANSAL, A. et al. Power management in energy harvesting sensor networks. **ACM Trans. Embed. Comput. Syst.**, ACM, New York, NY, USA, v. 6, n. 4, set. 2007. ISSN 1539-9087. Disponível em: <http://doi.acm.org/10.1145/1274858.1274870>.

KAźMIERSKI, T.; BEEBY, S. **Energy Harvesting Systems: Principles, Modeling and Applications**. New York: Springer Science+Business Media, 2011.

LIN, X. et al. Concurrent task scheduling and dynamic voltage and frequency scaling in a real-time embedded system with energy harvesting. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 35, n. 11, p. 1890–1902, Nov 2016. ISSN 0278-0070.

LINCOLN, B.; CERVIN, A. Jitterbug: a tool for analysis of real-time control performance. In: **Proceedings of the 41st IEEE Conference on Decision and Control, 2002.** [S.l.: s.n.], 2002. v. 2, p. 1319–1324 vol.2. ISSN 0191-2216.

LIU, C. L.; LAYLAND, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. **J. ACM**, ACM, New York, NY, USA, v. 20, n. 1, p. 46–61, jan. 1973. ISSN 0004-5411. Disponível em: <http://doi.acm.org/10.1145/321738.321743>.

LIU, S. et al. Harvesting-aware power management for real-time systems with renewable energy. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 20, n. 8, p. 1473–1486, Aug 2012. ISSN 1063-8210.

LIU, S.; QIU, Q.; WU, Q. Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting. In: **2008 Design, Automation and Test in Europe**. [S.l.: s.n.], 2008. p. 236–241. ISSN 1530-1591.

MARVELL. **Marvell PXA270 Processor**. 2007. Disponível em: <http://www.marvell.com.cn/applicationprocessors/pxa-family/assets/pxa_27x_emts.pdf.> Acesso em: fev. 2018.

MOSER, C. et al. Real-time scheduling for energy harvesting sensor nodes. **Real-Time Systems**, v. 37, n. 3, p. 233–260, Dec 2007. ISSN 1573-1383. Disponível em: <https://doi.org/10.1007/s11241-007-9027-0>.

OMNET++. **OMNeT++ Simulation Manual version 5.2**. 2015. Disponível em: <https://omnetpp.org/doc/omnetpp/manual/#sec:simple-modules:signal-based-statistics.> Acesso em: nov. 2017.

REN21. **RENEWABLES 2018 GLOBAL STATUS REPORT**. [S.l.], 2018. Disponível em: <http://www.ren21.net/wp-content/uploads/2018/06/17-8652_GSR2018_FullReport_web_-1.pdf>.

SAINI, S. **Low Power Interconnect Design**. 1. ed. [S.l.]: Springer-Verlag New York, 2015.

ULLMAN, J. D. **Elements of ML Programming**. 1. ed. [S.l.]: Prentice Hall, 1998. ISBN 9780131848542.

WESTERGAARD, M. Cpn tools 4: Multi–formalism and extensibility. In: **Application and Theory of Petri Nets and Concurrency**. [S.l.]: Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38696-1.

WOLF, W. **Computers as Components: Principles of Embedded Computing System Design.** 2th. ed. [S.l.]: Elsevier, 2008.

XU, H. et al. Energy-efficient scheduling with reliability guarantee in embedded real-time systems. **Sustainable Computing: Informatics and Systems**, v. 18, p. 137 – 148, 2018. ISSN 2210-5379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210537916300075>.

XU, R.; MOSSÉ, D.; MELHEM, R. Minimizing expected energy consumption in real-time systems through dynamic voltage scaling. **ACM Trans. Comput. Syst.**, ACM, New York, NY, USA, v. 25, n. 4, dez. 2007. ISSN 0734-2071. Disponível em: <http://doi.acm.org/10.1145/1314299.1314300>.