Avaliação estocástica de consumo de energia no projeto de sistemas embarcados.

Fernando Ferreira de Carvalho

Dissertação submetida ao Centro de Informática (CIn) da
Universidade Federal de Pernambuco
como requisito parcial para obtenção
do grau de Mestre em Ciências da Computação.
Orientador: Professor Doutor Paulo Romero Martins Maciel

Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn
Recife – Brasil
Março, 2004

Carvalho, Fernando Ferreira de
Avaliação estocástica de consumo de energia no projeto de sistemas embarcados / Fernando Ferreira de Carvalho – Recife : O Autor, 2005.
vii, 122 folhas : il., tab., fig.

Dissertação (mestrado) – Universidade Federal de Pernambuco. CIn. Ciências da Computação, 2005.

Inclui bibliografia.

1. Ciências da computação – Sistemas embarcados. 2. Consumo de energia – Avaliação estocástica. 3. Estimativa de energia e potência – Redes de Petri – Modelo Power-GSPN – Cadeia de Markov – Análise estacionária. I. Título.

004.03	CDU (2.ed.)	UFPE
003.3	CDD (22.ed.)	BC2005-552

Dissertação de Mestrado apresentada por **Fernando Ferreira de Carvalho** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Avaliação Estocástica de Consumo de Energia no Projeto de Sistemas Embarcados"**, orientada pelo **Prof. Paulo Romero Martins Maciel** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Djamel Sadek Hadj Sadoc
Departamento de Sistemas de Computação - CIn / UFPE

Prof. Ricardo Massa Ferreira Lima
Escola Politécnica / UPE

Prof. Paulo Romero Martins Maciel
Departamento de Sistemas de Computação – CIn / UFPE

Visto e permitida a impressão.
Recife, 31 de março de 2005.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedicado à:

Nandinho e **Felipe**, meus filhos,

Luciana, minha esposa,

Antônia, minha mãe,

Esmeralda, minha tia,

Paulinha, minha irmã e

Prof. Dr. Paulo Maciel, orientador.

Avaliação estocástica de consumo de energia no projeto de sistemas embarcados.

Autor: Fernando Ferreira de Carvalho
Orientador: Prof. Dr. Paulo Romero Martins Maciel

Resumo

Em metodologias de projeto de sistemas embarcados, ambientes de suporte à tomada de decisões são de grande importância, principalmente quando o projeto de tais sistemas deve considerar restrições de desempenho, consumo de energia etc. Este trabalho trata sobre estimativa de potência no desenvolvimento de sistemas digitais embarcado. A metodologia proposta baseia-se em uma extensão de modelo das redes de Petri estocástica generalizadas (*Generalized Stochastic Petri Net* - GSPN), denominada Power-GSPN. Este trabalho apresenta um conjunto de métricas que permitem avaliar o consumo de energia, potência, autonomia e o desempenho dos sistemas. A utilização das redes de Petri é de grande importância dada sua fundamentação matemática, a possibilidade de representar sistemas paralelos, síncronos e concorrentes, além de suportar, em um único modelo, especificação, validação, análise de desempenho, estimativa e documentação.

Palavras chave: Estimativas, redes de Petri, processos estocásticos, análises, modelo, avaliação, consumo de energia, potência, desempenho e medidas.

Agradecimentos

Obrigado meu Deus por mais esta conquista!

Gostaria de registrar meus sinceros agradecimentos como reconhecimento da dedicação de muitas pessoas que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Primeiramente, ao meu orientador professor Paulo R. M. Maciel, pelo enorme trabalho e tempo dedicado ao desenvolvimento desta pesquisa de forma objetiva e consistente.

A minha Família, minha esposa Luciana, meus filhos nandinho e Felipe, pela paciência e compreensão. A minha mãe, titia e minha irmã pelo incentivo, ajuda e apoio moral.

Aos professores Djamel F. H. Sadok, PhD, e Ricardo Massa Ferreira de Lima, PhD, pela participação da banca examinadora, contribuindo substancialmente para a melhoria da qualidade deste trabalho.

Aos colegas e professores do ALUPA (Alunos do Paulo), em especial a Raimundo Barreto, prof. Meuse Oliveira, Sérgio Galdino, Vanessa Bittencourt, Roberta Fagundes, Ismênia e todos os outros, companheiros de mestrado e doutorado que tanto deram apoio nos momentos em que parecia não haver luz no fim do túnel.

Aos demais professores e funcionários da UFPE.

Ao CNPQ pelo apoio financeiro.

Sumário

Capítulo 1 - Introdução 1
1.1 Motivação 1
1.2 Proposta de Tese 3
1.3 Estrutura da Dissertação 4
1.4 Resumo 5

Capítulo 2 - Sistemas Embarcados 6
2.1 Introdução 6
2.2 Arquitetura 10
2.2.1 Microprocessadores e seu espaço de projeto 11
2.2.2 Arquiteturas especializadas para sistemas embarcados 12
2.3 Modelagem e projeto de alto nível de sistemas embarcados 12
2.4 Metodologia de projeto 13
2.5 Níveis de abstração 16
2.6 Linguagens de especificação e projeto 17
2.7 Particionamento entre software e hardware 19
2.8 Estimadores 21
2.9 Trabalhos relacionados 23
2.10 Resumo 25

Capítulo 3 - Fundamentação Teórica 27
3.1 Introdução 27
3.2 Processos Estocásticos 29
3.3 Processos Markovianos 30
3.4 Cadeias de Markov 30
3.4.1 Cadeias de Markov de Tempo Contínuo (CTMC) 31
3.4.2 Métodos de soluções para probabilidade em estado estacionário. 33
3.4.3 Métodos de solução para probabilidades em estado transiente. 36
3.5 Redes de Petri 38
3.5.1 Introdução 38
3.5.2 Componentes de uma rede de Petri. 40
3.5.3 Redes place-transition 41
3.5.4 Estrutura definda em matriz 42
3.5.5 Rede de Petri marcada 42
3.5.6 Regras de Execução 43
3.5.7 Matriz de Incidência 44
3.5.8 Grafo de Marcações Acessíveis 44
3.5.9 Redes de petri e suas classes 45
3.5.10 Redes Free-Choice 46
3.5.11 Propriedades das redes de Petri 47
3.5.12 Propriedades Comportamentais 47
3.5.13 Propriedades estruturais 50

3.5.14 Análise das redes de Petri 51
3.5.15 Árvore de Cobertura 51
3.6 O Modelo GSPN 53
3.7 Resumo 56

Capítulo 4 - Modelagem e estimativa de potência 57
4.1 Introdução 57
4.2 Power-GSPN 59
4.3 Modelagem de ações determinísticas através de aproximação por fases. 60
4.4 Geração do Modelo 62
4.4.1 Representação de sistemas com mais de uma tarefa 66
4.4.2 Interrupções e compartilhamento de recursos 66
4.5 Temporizadores como geradores de interrupção 67
4.6 Definição das Métricas 68
4.7 Metodologia de medição do consumo 72
4.8 Um pequeno exemplo didático 75
4.9 Resumo 81

Capítulo 5 - Estudo de Caso 82
5.1 Introdução 82
5.2 Sistema de verificação de Código de Redundância Cíclica de 8 bits (CRC-8). 83
5.2.1 O modelo Power-GSPN do CRC-8 84
5.2.2 Comparação da potência medida com a estimada 86
5.2.3 Estimativas de potência e desempenho 87
5.2.4 Experimentos com variações de parâmetros 89
5.3 Código da rotina de ordenação Bubble Sort. 92
5.3.1 A modelo Power-GSPN do *Bubble Sort* 94
5.3.2 Estimativas de potência e desempenho 96
5.3.3 Experimentos com variação de parâmetros 98
5.4 Oxímetro de pulso 101
5.4.1 O modelo Power-GSPN do Oxímetro de pulso 103
5.4.2 Estimativas de potência e desempenho 109
5.4.3 Experimentos com variações de parâmetros 111
5.5 Resumo 114

Capítulo 6 - Conclusão e Trabalhos Futuros 115
6.1 Conclusão 115
6.2 Trabalhos Futuros 117

Referências Bibliográficas 118

Lista de Figuras

Figura 1.1: Metodologia proposta para estimativas de potência e desempenho em projeto de sistemas embarcados. 3
Figura 2.1: Retorno financeiro e janelas de tempo. 8
Figura 2.2: Estilos de projeto. 9
Figura 2.3:Metodologia de projeto. 13
Figura 2.4:Relação entre particionamento e definição da macro-arquitetura. 20
Figura 3.1:Processo de modelagem simples. 28
Figura 3.2:Principais classificações de Sistemas. 28
Figura 3.3:Amostras de um processo estocástico. 30
Figura 3.4:Elementos básicos de uma rede de Petri. 40
Figura 3.5:Períodos do dia representado com redes de Petri. 41
Figura 3.6:Abstração X Interpretação. 45
Figura 3.7:Rede de Petri Marcada. 48
Figura 3.8:Exemplo de rede não limitada. 48
Figura 3.9:Rede Segura depende da marcação inicial. 49
Figura 3.10:Vivacidade. 49
Figura 3.11:Geração do gráfico de alcançabilidade GSPN. 55
Figura 4.1:Diagrama da metodologia proposta. 58
Figura 4.2:Sub-rede em série, chamado de aproximação por fases. 60
Figura 4.3:Modelo (Erlang) que aproxima um tempo determinístico. 61
Figura 4.4:(estado) antes do disparo da transição Ação. 64
Figura 4.5:Marcação (estado) após o disparo da transição Ação. 64
Figura 4.6:Processador compartilhado com quatro tarefas. 67
Figura 4.7:Modelo do Temporizador. 68
Figura 4.8: a -Gráfico da variação de tensão na resistência ; b - pulsos de sincronismo. 74
Figura 4.9:Representação em Power-GSPN do sistema exemplo. 77
Figura 4.10:Precisão da estimativa X N° de Fases da aproximação determinística. 78
Figura 4.11:Variação das probabilidades da Tarefa 3 X Potência (PS). 80
Figura 4.12:Variação das probabilidades Tarefa 3 X Taxa de execução (Tx). 80
Figura 5.1:Código da rotina CRC-8 implementada em Assembly. 84
Figura 5.2:Modelo do CRC-8 em Power-GSPN. 85
Figura 5.3:Variação da probabilidade do bit ser um (1) X potência dissipada. 90
Figura 5.4:Variação da probabilidade do bit ser um X Taxa de Execução do sistema. 91
Figura 5.5:Variação do tamanho da palavra de entrada X Potência dissipada pelo sistema PS. 91
Figura 5.6:Variação do tamanho da palavra de entrada X Taxa de execução (Tx). 92
Figura 5.7:Código em Assembly da rotina de ordenação Bubble sort. 93
Figura 5.8:Modelo do Bubble sort em Power-GSPN. 95
Figura 5.9:Variação da probabilidade dos elementos estarem ordenados X potência dissipada pelo sistema (PS) (*bubble sort*). 99

Figura 5.10:Variação da probabilidade dos elementos estarem ordenados X Taxa de execução do sistema. (Tx) (*bubble sort*). 99
Figura 5.11:Variação do número de elementos do conjunto de entrada X Potência dissipada pelo sistema (PS) - (bubble sort). 100
Figura 5.12:Variação do tamanho da palavra de entrada X Taxa de execução do sistema (Tx) - (bubble sort). 101
Figura 5.13:Estrutura básica de um oxímetro de pulso. 102
Figura 5.14:Sinais na sonda de um oxímetro de pulso. 102
Figura 5.15:Plano de excitação interrompendo o plano de Amostragem e controle. 105
Figura 5.16:Modelo Power-GSPN do Oxímetro de Pulso. 108
Figura 5.17:Variação da probabilidade da tarefa carregar vetores X Potência dissipada pelo sistema (PS)- (Oxímetro). 112
Figura 5.18:Variação da probabilidade da tarefa controle X Potência dissipada pelo sistema (PS)- (Oxímetro). 113
Figura 5.19:Variação da probabilidade da tarefa controle X Potência dissipada pelo sistema (PS) – (Oxímetro). 113

Lista de Tabelas

Tabela 3.1: Algumas interpretações típicas para lugares e transações 40
Tabela 4.1:Atributos das tarefas do exemplo. 76
Tabela 4.2:Atributos e estimativas das tarefas do exemplo 78
Tabela 5.1:Informações de Tempo, prioridade e Consumo de energia das tarefas do CRC. 86
Tabela 5.2:Métricas de estimativas de potência e desempenho das tarefa do CRC8. 87
Tabela 5.3:Métricas de estimativas de potência e desempenho por tarefa do CRC8. 88
Tabela 5.4:Informações de tempo, consumo de energia e prioridade das tarefas do bubble sort 96
Tabela 5.5:Métricas de estimativas de potência e desempenho das tarefa do *bubble sort*. 97
Tabela 5.6:Informações de tempo, prioridade, consumo de energia das tarefas Oxímetro de pulso. 108
Tabela 5.7:Métricas de estimativas de potência e desempenho das tarefa do Oxímetro de pulso 110

Capítulo 1 - Introdução

Neste capítulo faz-se uma explanação sobre a proposta do trabalho. Inicialmente, mostra-se o contexto onde este trabalho esta inserido. Posteriormente, apresenta-se alguma das necessidades do projeto de sistemas embarcados, particularmente onde se tem restrições quanto à potência e desempenho. Em seguida, é apresentada a proposta do trabalho, e os caminhos para se atingir os objetivos previstos. Finalmente, mostra-se a estrutura desta dissertação.

1.1 Motivação

Os últimos cem anos da história humana foram palcos de um surpreendente avanço tecnológico impulsionado pelo conhecimento científico e pelo complexo jogo dos sistemas econômicos. A tecnologia de sistemas eletro-eletrônicos, em especial, foi constantemente revolucionada pelo progresso contínuo dos processos de fabricação de dispositivos semicondutores devido à demanda crescente dos mercados de bens de consumo. Nos dias atuais em um novo milênio, os sistemas eletrônicos permeiam a vida cotidiana desde os grandes centros urbanos aos mais distantes pontos do planeta, alterando profundamente as relações sociais e a vida humana. O aumento da capacidade de integração e o desenvolvimento de novas tecnologias têm possibilitado o desenvolvimento de sistemas de alto desempenho e funcionalidade, os custos são mantidos em níveis atrativos para o mercado consumidor. Na prática, verifica-se um alto nível de incorporação destes sistemas em produtos como automóveis, eletrodomésticos, equipamentos de comunicação pessoal e médico-hospitalares. Entre as maiores barreiras técnicas para uma maior evolução deste cenário, pode-se

mencionar a disponibilidade de metodologias e ferramentas de suporte ao projeto dos sistemas eletrônicos, uma vez que nenhuma destas acompanhou o avanço tecnológico dos processos de fabricação em silício. De forma resumida, os projetos de sistemas atuais carecem de ferramentas e técnicas que forneçam informações e estimativas precisas sobre restrições de desempenho, funcionalidade e consumo, ainda em tempo de projeto. O futuro dos sistemas eletrônicos e o uso pleno do potencial tecnológico dos dispositivos atuais dependem fortemente de ferramentas que aprimore a forma de se projetar tais sistemas.

A crescente complexidade dos sistemas digitais, associado ao desempenho, autonomia e a redução de seu volume e custo, têm conduzido a novas pesquisas em metodologia de desenvolvimento que satisfaçam as restrições impostas pelo projeto. Um dos aspectos que vem sendo muito estudados atualmente são as técnicas para projeto de sistemas de baixo consumo de energia, assim como métodos de estimativa deste consumo. Por esta razão, este trabalho concentra em projeto de sistemas embarcados que tenham o consumo de energia como aspectos físicos. Atualmente existem poucos ambientes que integram ferramentas para o desenvolvimento de sistemas embarcados e suas restrições. Da mesma forma, tais ambientes não abrangem todas as fases de projeto.

Rede de Petri é um formalismo matemático adequado para modelagem de sistemas, os quais podem ser concorrentes, assíncronos, paralelos, distribuídos. O uso de redes de Petri não está restrito a informações estruturais, sendo possível avaliar o comportamento e a dinâmica dos sistemas.

Um modelo adequado pode possibilitar a obtenção de estimativas em fases iniciais de projeto, auxiliando, portanto o processo de escolha de alternativa de projeto que satisfaçam as restrições impostas. Modelos são convenientes pois podem refletir várias características do sistema modelado, dentre elas, aspectos comportamentais, temporais, de consumo de energia e potência. Poucas soluções estão fortemente embasadas em modelos formais.

1.2 Proposta de Tese

Este trabalho se insere no contexto do projeto de sistemas digitais embarcados, em particular, os que têm restrições quanto ao consumo de energia. Tem como objetivo a formalização de uma metodologia baseada em rede de Petri Estocástica Generalizada (*Generalized Stochastic Petri Net* - GSPN), onde se criou uma extensão denominada Power-GSPN, utilizada para estimar consumo de energia e desempenho. Estudos de casos foram implementados para validar o método proposto.

A idéia geral do trabalho proposto pode ser visualizada na Figura 1.1. Consiste em criar uma metodologia para servir como ferramenta de auxílio no projeto de sistemas embarcados, fornecendo estimativas de potência dissipada, desempenho e autonomia que sirvam de instrumento ao projetista para tomada de decisões.

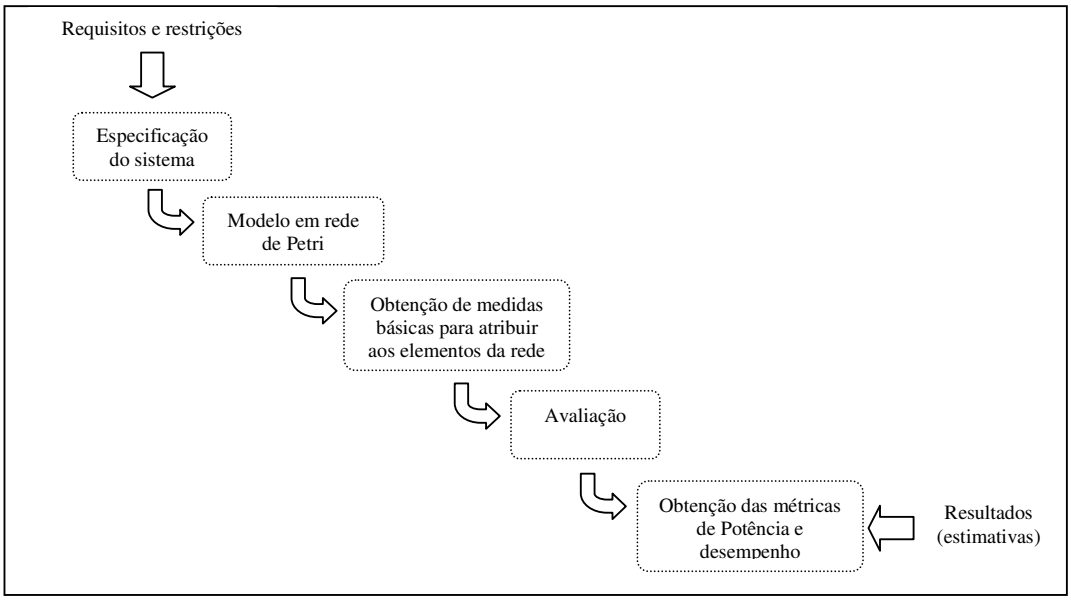


Figura 1.1: Metodologia proposta para estimativas de potência e desempenho em projeto de sistemas embarcados.

A proposta deste trabalho considera que, dada uma especificação de um sistema, que é considerada a informação de entrada da proposta, através de técnicas de modelagem, mostrada neste trabalho, é factível a representação da especificação em

modelo Power-GSPN. Para o modelo Power-GSPN gerado, deve-se atribuir aos elementos informações temporais e de consumo de energia, possibilitando ao modelo uma representação fidedigna do sistema. Estas informações temporais e de consumo de energia obtêm-se através de medição, consulta, *data-sheets* etc. Um processo de medição de consumo de energia para instruções de um processador é mostrado. Um conjunto de métricas é definido para obter-se as estimativas de consumo de energia, potência, desempenho e autonomia. Na fase seguinte, avalia-se o modelo gerado em estado estacionário, isto é, independente do tempo. Quando necessário, o modelo também será avaliado em estado transiente, isto é, dependente do tempo. Calculam-se as métricas definidas através da avaliação. Quando a avaliação analítica não for possível ou conveniente, pode-se recorrer à simulação para a obtenção das métricas, que são consideradas o resultado da metodologia.

Este trabalho possibilita a análise do consumo de energia do sistema ainda em tempo de projeto sem a necessidade da implementação real para obtenção de medidas. Em projetos de sistemas embarcados, faz-se necessário quantificar o consumo ainda em fase de projeto possibilitando a tomada de decisões rápidas, a fim de encontrar uma das possíveis soluções que atenda as restrições de projeto, para a implementação do sistema

1.3 Estrutura da Dissertação

Esta dissertação esta organizada em seis capítulos principais. O Capítulo 2 descreve os principais itens abordados no projeto de sistemas digitais; as metodologias utilizadas e os sistemas com restrição de consumo de energia. São mostrados os principais trabalhos na área de estimativas de consumo e/ou potência de sistemas digitais nos últimos anos.

O Capítulo 3 inicia relacionando os sistemas aos modelos. Em seguida, apresenta-se os processos estocásticos e suas definições. Posteriormente, introduz-se as cadeias de Markov de tempo contínuo (*Contínuos Time Markov Chain* - CMTC) e as técnicas de avaliação. Por fim, apresenta-se as redes de Petri, suas propriedades, técnicas de avaliação e a extensão GSPN.

O Capítulo 4 define o modelo Power-GSPN, definindo os conceitos e as métricas a serem utilizados para obtenção das estimativas de potência, desempenho e autonomia. Um exemplo didático foi utilizado para demonstrar uma aplicação do modelo Power-GSPN.

Tendo em vista a validação do modelo, o Capítulo 5 descreve três estudos de caso que utilizam a metodologia proposta para estimar a potência, desempenho e autonomia. No final os resultados são comentados.

Por fim, o Capítulo 6 discute de forma crítica os resultados obtidos, retirando conclusões e estabelecendo trabalhos.

1.4 Resumo

Neste capítulo, introduzimos ao leitor os problemas modernos do projeto de sistemas digitais. Em especial, a falta de ferramentas que auxiliem o projetista a validar, ainda em tempo de projeto, as restrições especificadas. Argumentamos que a solução destes problemas reside na adoção de ferramentas e técnicas de projeto, apoiadas em formalismo matemático que suporte a criação de modelos, para que ainda em tempo de projeto estimativas sejam obtidas.A proposta de análise e estimativa de potência de sistemas digitais através das Power-GSPN , visa auxiliar, validar e diminuir o tempo de projeto de sistemas digitais onde o consumo de energia é um fator critico. Delineamos a estrutura geral a ser apresentada ao longo deste trabalho.

Capítulo 2 - Sistemas Embarcados

Este capítulo discute a modelagem e o projeto de sistemas computacionais embarcados, as metodologias aplicadas e dá uma visão geral de suas principais atividades. Por envolver restrições mais complexas que um sistema computacional tradicional, como limite de potência, custo e baixo tempo de projeto, estes sistemas são constante objeto de estudo. É também propósito deste capítulo mostrar os principais trabalhos relacionados a estimativas de potência nos últimos anos.

2.1 Introdução

Os sistemas embarcados apresentam características em comum com sistemas computacionais de propósito gerai, mas não possuem a uniformidade desses Cada aplicação pode apresentar requisitos diferentes de desempenho, consumo de potência e área ocupada, o que vai acarretar em uma combinação distinta de módulos de hardware e software para atender estes requisitos. Em muitas aplicações é adequada a integração do sistema em uma única pastilha (*SoC – System-on-a-Chip*). A arquitetura de hardware de um SoC embarcado pode conter um ou mais processadores, memórias, interfaces para periféricos e blocos dedicados. Os componentes são interligados por uma estrutura de comunicação que pode variar de um barramento a uma rede complexa (NoC - *Network-on-a-Chip*). Os processadores podem ser de tipos diversos (RISC, CISC, ARM, VLIW, DSP, até ASIPs – processadores integrados para aplicações específicas), conforme a aplicação. O software de aplicação pode ser composto por múltiplos processos, distribuídos entre diversos processadores e comunicando-se através de diferentes mecanismos. Em muitos casos, são gerenciados através de um sistema

operacional de tempo real (RTOS), que fornece, pelo menos, serviços de comunicação e escalonamento de processos. Esta exploração consiste em avaliar as alternativas de arquitetura de hardware e de software, verificando o impacto sobre desempenho, potência e área do sistema. Para que esta avaliação prévia corresponda ao sistema real é necessária a utilização de bons estimadores.

O projeto deste tipo de sistema computacional é extremamente complexo, por envolver conceitos até agora pouco analisados pela computação de propósitos gerais. Por exemplo, questões da portabilidade, limite de consumo de energia, redução de desempenho, baixa disponibilidade de memória, necessidade de segurança e confiabilidade, possibilidade de funcionamento em uma rede maior, e o curto tempo de projeto tornam o desenvolvimento de sistemas computacionais embarcados uma área de pesquisa de muito interesse.

O projeto de sistemas eletrônicos embarcados enfrenta grandes desafios, pois o espaço de projeto arquitetural a ser explorado é muito vasto. A arquitetura de hardware de um SoC embarcado pode conter um ou mais processadores, memórias, interfaces para periféricos e blocos dedicados. Os processadores podem ser de tipos diversos (RISC, CISC, ARM , VLIW, DSP, até ASIPs – processadores integrados para aplicações específicas), conforme a aplicação. No caso de sistemas contendo componentes programáveis, o software de aplicação pode ser composto por múltiplos processos, distribuídos entre diferentes processadores e comunicando-se através de mecanismos variados. Um sistema operacional de tempo real (RTOS), oferecendo serviços como comunicação e escalonamento de processos, pode ser necessário. Além do grande tempo que pode ser gasto com uma exploração sistemática deste espaço de projeto, deve-se considerar ainda o tempo necessário para o projeto e validação individual de todos os componentes dedicados do sistema – processadores, blocos de hardware, rotinas de software, RTOS – assim como o tempo de validação de sua agregação dentro de um mesmo sistema.

Por outro lado, a grande pressão num mercado mundial globalizado, somada à contínua evolução tecnológica, impõe às empresas a necessidade de projetarem novos sistemas embarcados dentro de janelas de tempo cada vez mais estreitas, de poucos

meses. Além disto, novos produtos têm uma vida cada vez mais curta, de modo que o retorno financeiro de seu projeto deve ser obtido também em poucos meses. Conforme mostrado na Figura 2.1[55], atrasos de poucas semanas no lançamento de um produto podem comprometer seriamente os ganhos esperados de um novo produto no mercado.

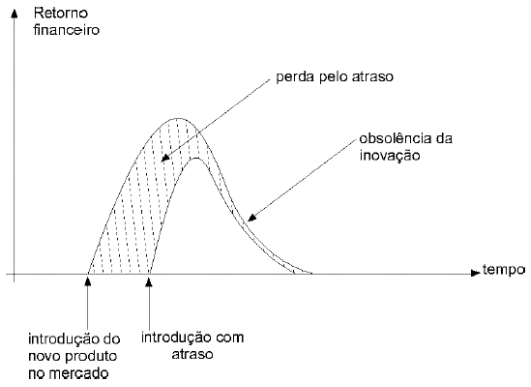


Figura 2.1: Retorno financeiro e janelas de tempo.

Um terceiro problema diz respeito aos custos de engenharia não-recorrentes (NRE, em inglês). O projeto de um sistema embarcado de grande complexidade é bastante caro para uma empresa, envolvendo equipes multidisciplinares (hardware digital, hardware analógico, software, teste) e a utilização de ferramentas computacionais de custo elevado. São especialmente elevados os custos de fabricação de sistemas integrados numa pastilha. Nos dias atuais, um conjunto de máscaras de fabricação está alcançando o custo de um milhão de dólares, o que obriga as empresas ao projeto de componentes que tenham garantidamente grande volume de produção, de forma a amortizar os custos de fabricação.

Estilo de projeto diz respeito à classe de estrutura física de circuitos usada na confecção final de um determinado circuito integrado (*chip* ou CI). As aplicações em geral podem ser implementadas usando circuitos integrados ASICs (*Aplication Specific Integrated Circuits*) *full-custom* ou dispositivos *semi-custom* programáveis e reconfiguráveis. A Figura 2.2 mostra possíveis estilos para implementação de projetos [10].

Embora tecnologias customizadas apresentem um custo mais alto de produção que outros estilos, devido à complexidade de desenvolvimento e fabricação, quando se deseja um grande volume de produção (alta escala de produção), o custo fixo é amortizado, o que permite torná-lo competitivo e mais barato que os FPGAs. Assim, o usuário deve escolher o estilo que melhor lhe convier, satisfazendo suas necessidades de desempenho e preço.

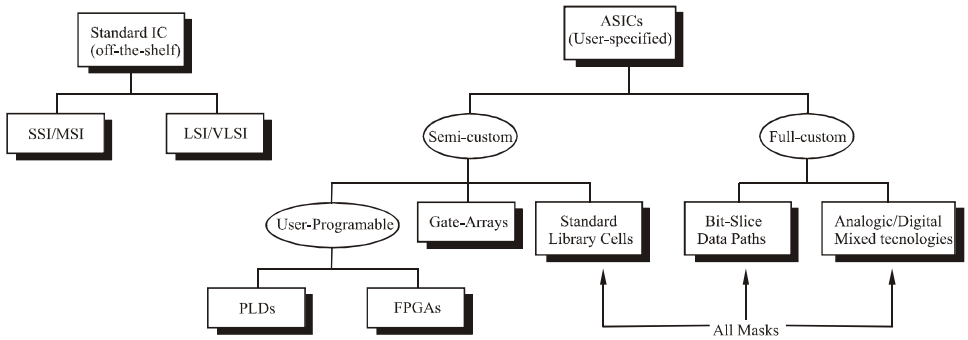


Figura 2.2: Estilos de projeto.

Em muitas aplicações, é adequada a integração do sistema em uma única pastilha (SoC –*system-on-a-chip*). Em situações onde requisitos de área, potência e desempenho sejam críticos, o projeto do SoC na forma de um ASIC (circuito integrado para aplicação específica) pode ser mandatório, elevando bastante os custos de projeto e fabricação. Em muitas outras situações, no entanto, é mais indicada a implementação do sistema em FPGA, alternativa de customização mais econômica para baixos volumes, ou através de sistemas baseados em famílias de microprocessadores, componentes que são fabricados em grandes volumes e integram milhões de transistores.

Na atual situação de competitividade industrial, seguindo-se a lei de Moore, tem-se à disposição o dobro de transistores a cada 18 meses. Consequentemente, sistemas dedicados com milhões de transistores devem ser projetados em poucos meses. Para isto, tem sido adotado o paradigma de projeto baseado em plataformas. Uma plataforma é uma arquitetura de hardware e software específica para um domínio de aplicação [28], mas altamente parametrizável (no número de componentes de cada tipo, na estrutura de comunicação, no tamanho da memória, nos tipos de dispositivos de

E/S, etc.). Esta estratégia viabiliza o reuso de componentes (ou núcleos) [23] previamente desenvolvidos e testados, o que reduz o tempo de projeto. O reuso pode ser ainda reforçado pela adoção de padrões [28] na arquitetura e projeto dos sistemas.

O projeto de um SoC embarcado consiste então em se encontrar um derivativo da plataforma que atenda aos requisitos da aplicação, como desempenho e consumo de potência. Partindo-se de uma especificação de alto nível da aplicação, é feita uma exploração das soluções arquiteturais possíveis, estimando-se o impacto de diferentes particionamento de funções entre hardware e software. Feita a configuração da arquitetura, é necessária a síntese da estrutura de comunicação que integrará os componentes de hardware [44]. Também é vital a existência de uma metodologia adequada ao teste de sistemas complexos integrados numa única pastilha.

Neste estilo de projeto, cada vez mais a inovação de uma aplicação depende do software. Embora a plataforma de hardware de um celular possa ser similar à de um controle de freios ABS, definitivamente o software não é o mesmo. Com a automação do projeto de hardware encaminhada pelo reuso de plataformas e componentes, a automação do projeto do software se torna o principal objetivo a ser alcançado para a diminuição do tempo de projeto, sem sacrifício na qualidade da solução. Esta automação deve idealmente cobrir o software aplicativo, o RTOS, as interfaces entre os processos e os acionadores dos periféricos.Também aqui é essencial o reuso de componentes de software previamente desenvolvidos, de modo que o projeto do sistema embarcado concentre-se apenas na configuração e integração dos mesmos.

2.2 Arquitetura

Nesta seção são revisadas as arquiteturas clássicas de processadores, discutindo-se a adequação de cada estilo de projeto para um dado sistema alvo; discute-se também o impacto das memórias em sistemas embarcados, assim como as estruturas de comunicação hoje disponíveis, à luz de seu impacto em futuros sistemas constituídos de milhões de componentes heterogêneos.

2.2.1 Microprocessadores e seu espaço de projeto

O projeto de sistemas embarcados toma sempre como base um ou mais processadores. Embora esta solução pareça extremamente conservadora do ponto de vista de inovação, ela traz enormes vantagens do ponto de vista operacional. Primeiro, o fator de escala. Como os microprocessadores são encontrados em milhares de projetos, seu custo dilui-se entre muitos clientes, às vezes até competidores entre si. Mais ainda, uma vez que uma plataforma baseada em processador esteja disponível dentro de uma empresa, novas versões de produtos podem ser feitas pela alteração do software da plataforma. A personalização do sistema dá-se através do software de aplicação, que toma atualmente a maior parte do tempo de projeto. Além destas vantagens competitivas, há ainda o fator treinamento de engenheiros, já que estes geralmente se formam com conhecimentos de programação de microprocessadores.

É claro que o projeto usando microprocessadores não é vantajoso em todo aspectos. A questão da potência, cada vez mais valorizada nos tempos atuais, é crítica. Como são projetados para executar qualquer programa, existem estruturas de hardware dentro dos processadores que consomem muitos recursos, mas que são muitas vezes subutilizadas.

Para uma aplicação extremamente específica, um projeto usando lógica programável como FPGAs ou ASICs pode ter um desempenho muito melhor que usando um processador, ou mesmo uma potência mais baixa. O problema é que sistemas reais possuem diversos comportamentos (modelos de computação) e o atendimento simultâneo dos mesmos tende a diminuir o desempenho do hardware dedicado. Além disto, é preciso considerar que se encontram processadores nas mais diversas combinações de preço, desempenho e potência. Os processadores também contam com grupos de projeto imensos, que chegam às centenas de projetistas, e com tecnologias do estado-da-arte para sua fabricação. Tudo isto torna o uso de processadores extremamente interessante para o projeto de sistemas embarcados.

Há, contudo, uma série de questões a serem respondidas, mesmo que o projetista decida usar um microprocessador. Por exemplo, se o projeto tem limitações

de potência, famílias de processadores que trabalham com freqüências mais baixas podem não possuir desempenho suficiente. Nestas situações, é preciso escolher processadores com controle de potência embutido, onde partes do processador possam ser desligadas, ou utilizar técnicas como processadores e multiprocessadores de múltipla voltagem [13]. Uma alternativa mais interessante é escolher a arquitetura adequada para o projeto em questão, pois os ganhos em potência e desempenho podem ser maiores, conforme será discutido a seguir.

2.2.2 Arquiteturas especializadas para sistemas embarcados

Das tendências modernas em arquiteturas de computadores, somente algumas das idéias discutidas acima são efetivamente usadas no projeto de sistemas embarcados. O problema de potência dissipada parece definir que, com os recursos tecnológicos atuais, chegam a não ser viáveis. Máquinas VLIW parecem oferecer a combinação correta de desempenho e potência, com paralelismo descoberto em tempo de compilação, já que, a maioria das aplicações embarcadas são estáticas, isto é, não são alteradas pelo usuário final. A reconfiguração de hardware estará cada vez mais presente nos sistemas embarcados. Embora os FPGAs clássicos tenham alto consumo de potência, arquiteturas de mais baixa potência estão sendo estudadas para serem embarcadas em plataformas. A enorme vantagem de se incluir a reconfiguração de hardware é a possibilidade extra de personalização de um chip, e a atenuação do enorme custo de máscaras que as tecnologias nano-métricas apresentam.

2.3 Modelagem e projeto de alto nível de sistemas embarcados

Devido à possível complexidade da arquitetura de um sistema embarcado, contendo múltiplos componentes de hardware e software em torno de uma estrutura de comunicação, e à grande variedade de soluções possíveis visando o atendimento de requisitos de projeto, como desempenho, consumo de potência e área ocupada, é essencial o projeto do sistema em níveis de abstração elevados utilizando ferramentas que automatizem ao máximo as diversas etapas de uma metodologia consistente com os desafios existentes.

2.4 Metodologia de projeto

A Figura 2.3 [55] mostra uma metodologia completa de projeto de um sistema eletrônico embarcado. Esta metodologia é ideal, segundo a perspectiva do estado-da-arte da pesquisa na área, embora na prática ainda não existam ambientes comerciais de software de projeto que a implementem inteiramente.

O projeto de um sistema embarcado é iniciado usualmente por uma especificação da funcionalidade desejada, feita através de uma linguagem ou formalismo adequado. Idealmente, esta especificação deve ter um alto nível de abstração, no qual ainda não tenham sido tomadas decisões em relação à implementação desta funcionalidade em termos da arquitetura-alvo a ser adotada, nem sobre os componentes de hardware ou software a serem selecionados. Esta especificação deve ser preferencialmente executável, para fins de validação.

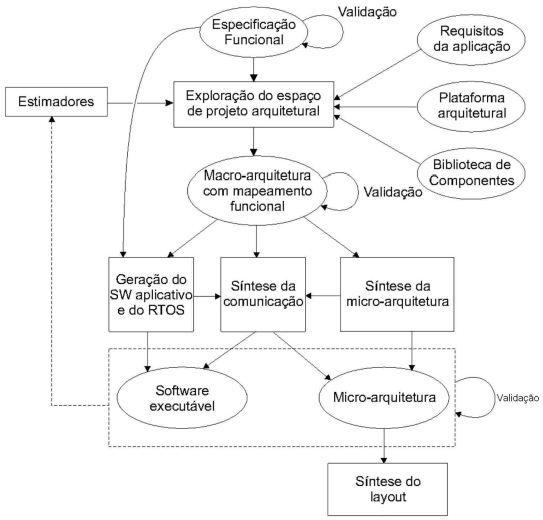


Figura 2.3:Metodologia de projeto.

A seguir, deve-se explorar o espaço de projeto arquitetural, de modo a se encontrar uma arquitetura que implemente as funções contidas na especificação inicial e que atenda aos requisitos de projeto, em termos de custo, desempenho, potência, área, etc. O resultado final desta etapa é uma *macro-arquitetura* (ou *arquitetura abstrata),* contendo um ou mais processadores de determinados tipos (DSP, microcontroladores) e outros componentes necessários (memórias, interfaces, blocos dedicados de hardware),

todos interconectados através de uma infra-estrutura de comunicação (um ou mais barramentos ou uma *Network-on-Chip*, NoC). Entre a especificação funcional e a macro-arquitetura, estabelece-se um *mapeamento,* através do qual cada função do sistema é atribuída a um processador ou a um bloco dedicado de hardware. Este mapeamento estabelece um determinado particionamento de funções entre hardware (blocos dedicados) e software (funções implementadas por um processador de instruções).

A exploração do espaço de projeto deve encontrar uma solução ótima para três questões básicas: 1) Quantos e quais são os processadores e blocos dedicados de hardware necessários? 2) Qual é o mapeamento ideal entre funções e componentes de hardware? 3) Qual é a estrutura de comunicação ideal para conectar os componentes entre si, tendo em vista as trocas de informações que devem ser realizadas entre as funções mapeadas para os componentes? Para que esta exploração seja efetuada rapidamente, é fundamental a existência de estimadores que, a partir da especificação funcional do sistema, sejam capazes de informar, com um grau de precisão adequado, os valores de métricas importantes de projeto (desempenho, potência, área) que irão resultar de alternativa arquitetural (uma macro-arquitetura e um mapeamento de funções).

Tendo em vista um espaço quase infindável de soluções arquiteturais possíveis, com uma correspondente complexidade computacional para exploração do mesmo em busca de uma solução ótima ou mesmo sub-ótima, esta etapa é usualmente simplificada pela escolha prévia de uma plataforma arquitetural conhecida e adequada ao domínio da aplicação, contendo um ou mais processadores de tipos conhecidos, além de outros componentes necessários, todos interconectados através de uma estrutura de comunicação também pré-definida

Usualmente, diversas funções serão mapeadas para um mesmo processador, sendo então implementadas como tarefas concorrentes que precisarão ser escalonadas e gerenciadas por um sistema operacional, e caso a aplicação assim o requisite, este deverá ser um sistema operacional de tempo real (RTOS).

Uma vez definida a macro-arquitetura, é necessária a geração do software para a mesma, a partir da especificação funcional do sistema. Idealmente, seria desejável uma síntese automática do software, incluindo tanto o software aplicativo como o RTOS. Esta geração do software é bastante facilitada se a especificação funcional inicial tiver sido feita sobre uma interface de programação da aplicação (API — *application programming interface)* padronizada, que ofereça recursos para comunicação entre as tarefas e para a qual exista uma implementação sobre a plataforma arquitetural (processadores e RTOS) selecionada. E também necessário um compilador, que traduza a especificação funcional para uma linguagem de programação adequada a cada processador adotado (a menos que a especificação funcional já tenha sido feita em uma tal linguagem).

Uma vez definidos os componentes de hardware da macro-arquitetura, incluindo a infra- estrutura de comunicação e os eventuais adaptadores, pode ser feita a síntese do hardware. Numa primeira etapa, a macro-arquitetura pode ser expandida para uma *micro-arquitetura* (ou arquitetura RT), contendo o detalhamento de todos os componentes e suas interconexões, pino-a-pino e considerando o funcionamento do circuito com precisão de ciclo de relógio. Numa segunda etapa, podem ser usadas ferramentas convencionais de síntese de hardware, que a partir da micro-arquitetura irão gerar o layout final do circuito. Para tanto, é necessário que a micro- arquitetura esteja descrita numa linguagem apropriada para estas ferramentas, como VHDL ou Verilog. A existência prévia de layouts para os componentes de hardware selecionados facilita bastante esta síntese, que se limita então ao posicionamento e roteamento de células.

Em todas as etapas da metodologia de projeto, é necessária uma validação das descrições funcionais e arquiteturais geradas. Normalmente, esta validação se dá por simulação, sendo portanto necessária a existência de simuladores adequados para o tratamento das linguagens utilizadas no processo de projeto. Embora ferramentas de verificação formal, que dispensam simulações exaustivas, sejam bastante atraentes, tais recursos ainda são incipientes no que se refere a sua utilização extensiva em determinados níveis de abstração.

2.5 Níveis de abstração

Como introduzido na seção anterior, o projeto de um sistema eletrônico embarcado passa por uma seqüência de níveis de abstração. Como não existe uma padronização destes níveis, a definição dos mesmos depende de metodologias e ferramentas particulares de projeto. Cada nível permite a validação de determinadas propriedades de projeto e serve de partida para o processo de síntese para um nível inferior subseqüente. Um esforço de padronização recente está sendo patrocinado por diversas empresas, divergindo parcialmente da apresentação a seguir.

A especificação inicial de um sistema é usualmente feita de uma forma puramente funcional, na qual não há nenhuma informação estrutural ou dependente da arquitetura-alvo sobre a qual o sistema será implementado. Esta descrição deve ser neutra em relação a possíveis implementações das funções em software ou em hardware, não necessitando conter informações detalhadas de como implementar os requisitos temporais. O sistema é descrito como um conjunto de funções (tarefas ou objetos, dependendo da linguagem adotada), que se comunicam através de primitivas de comunicação de alto nível, por exemplo na forma de *mensagens* ou de requisições de *serviços* e respostas aos mesmos. Cada transferência pode transportar diversos itens de dados simultaneamente. Este nível de abstração permite a validação da especificação funcional do sistema e serve como entrada para o processo de exploração arquitetural. Uma vez que são tomadas decisões em relação à arquitetura-alvo do sistema, este é descrito no nível de macro-arquitetura (ou arquitetura abstrata). Apesar da ausência de muitos detalhes, uma descrição neste nível de abstração já permite a obtenção de estimativas (de desempenho, potência, área) com um grau de precisão suficiente para que sejam tomadas decisões no processo de exploração arquitetural. Esta descrição já contém os componentes principais da arquitetura (processadores, blocos dedicados de hardware, memória, interfaces, estrutura de comunicação), assim como o mapeamento entre as funções do sistema e os processadores e blocos dedicados de hardware. No entanto, ainda não estão incluídos diversos componentes acessórios que serão necessários, tais como adaptadores de protocolos *(wrappers),* decodificadores de endereço, gerenciadores de interrupção e temporizadores, que dependem de decisões de

projeto tomadas posteriormente. A descrição já é temporizada, mas não tem a precisão de ciclos de relógio.

Uma descrição no nível de macro-arquitetura pode ser usada como entrada para os processos de síntese de comunicação, de software e de hardware. O resultado deste conjunto de sínteses será a descrição do sistema no nível de micro-arquitetura (ou nível RT), no qual o hardware estará inteiramente detalhado em termos de todos os blocos funcionais necessários, com informação a respeito de todos os seus pinos e das interconexões entre os mesmos. O software é descrito em linguagem de máquina de cada processador contido na arquitetura, sendo simulado com precisão de ciclo de relógio, o que exige uma descrição detalhada dos processadores, em termos de pipelines e memória cache, por exemplo. A comunicação estará descrita através de sinais específicos de determinados protocolos, e cada transferência transportará um único item de dados. A descrição serve de entrada para a síntese do layout do circuito integrado, através do uso de ferramentas comercialmente disponíveis.

2.6 Linguagens de especificação e projeto

O projeto de um sistema embarcado passa por diversos níveis de abstração e é orientado a um determinado domínio de aplicação (aplicações orientadas a dados ou controle, por exemplo). Para a representação de cada domínio, existe um modelo de computação mais adequado. Diferentes linguagens de especificação e de projeto têm sido adotadas para tratamento destes níveis de abstração e modelos de computação.

Para a especificação funcional inicial, é importante que a linguagem seja executável, para fins de validação. Seria também desejável que a linguagem permitisse a descrição de funções de forma neutra em relação a sua implementação em software ou hardware e que pudesse ser utilizada como entrada para um processo de síntese automática do software sobre a plataforma arquitetural adotada. Por outro lado, poderia ser também interessante a adoção de uma linguagem neutra em relação aos domínios de aplicação, requisito que no entanto parece conflitante com a possibilidade de síntese automática do software, já que este processo é altamente dependente da plataforma arquitetural e do domínio de aplicação (modelo de computação).

As descrições da macro-arquitetura e da micro-arquitetura devem ser feitas em linguagens que possam ser utilizadas como entrada para processos de síntese automática de hardware, além de serem evidentemente simuláveis.

Linguagens de programação têm sido bastante utilizadas, aproveitando sua popularidade. Este é o caso evidente de C e C++. Se, por um lado, C não é uma linguagem ideal do ponto de vista do conjunto de requisitos anteriormente estabelecidos para linguagens de especificação, especialmente em relação ao grau de abstração e à generalidade dos domínios de aplicação, ela tem a vantagem de permitir a geração de software para um grande número de processadores utilizados no contexto de sistemas embarcados. A adoção da orientação a objetos, como em C++, se por um lado é vantajosa do ponto de vista de especificações de alto nível onde reuso e especialização de componentes são características muito interessantes, por outro lado causa problemas para a síntese de hardware ou tamanho do software.

Para a descrição de hardware, as linguagens mais populares são VHDL e Verilog. Sua grande vantagem é a possibilidade de serem utilizadas como entrada para simulação e síntese automática de circuitos descritos no nível da micro-arquitetura, através da utilização de ferramentas comerciais bastante difundidas. VHDL, no entanto, é uma linguagem mais orientada para simulação, de tal modo que algumas de suas construções não são sintetizáveis, o que força ferramentas de síntese a aceitarem apenas um determinado sub-conjunto da linguagem e/ou estilo de descrição. VHDL e Verilog têm uma semântica orientada apenas para a descrição de hardware, não sendo apropriadas para descrições de software nem para especificações funcionais de alto nível.

C e C++ apresentam um evidente prejuízo quando consideradas como linguagens de projeto no nível da macro ou da micro-arquitetura, tendo em vista sua inadequação semântica para a descrição de aspectos de hardware. A introdução de SystemC visa justamente sanar esta necessidade, combinando as vantagens da popularidade de C++ e sua adequação ao processo de geração de software com uma semântica adicional (na forma de uma biblioteca de funções) apropriada para a descrição de hardware, através de construções como portas, sinais, relógios (que

sincronizam processos), etc. SystemC 1.0 apresentava uma semântica de hardware muito próxima de VHDL, e portanto mais adequada para descrições no nível RT, exigindo por exemplo a descrição da comunicação entre processos através de sinais de um protocolo específico, o que já reflete uma determinada implementação do hardware. SystemC 2.0, por outro lado, introduziu construções de comunicação e sincronização de mais alto nível *(canais, interfaces* e *eventos),* permitindo a modelagem de mecanismos mais abstratos de comunicação e, portanto, o uso da linguagem em níveis de abstração superiores ao nível RT.

Na tentativa de aumentar as possibilidades de modelagem e abstração, Java também tem sido utilizada como ferramenta de descrição e simulação de sistemas. Contudo, o nível de abstração oferecido é equivalente àquele utilizado em C++.

Finalmente, deve ser salientado que as linguagens aqui consideradas são essencialmente destinadas à descrição do comportamento e/ou estrutura de um sistema, seja em software e/ou em hardware, e não à especficaçào funcional e/ou de requisitos, o que poderia ser feito com uma linguagem como UML. Esforços recentes têm procurado investigar a utilização de UML na especificação funcional e de requisitos de sistemas embarcados [55].

2.7 Particionamento entre software e hardware

O particionamento de funções entre software e hardware é realizado como parte da exploração do espaço de projeto arquitetural. A entrada para este processo é a especificação funcional do sistema, idealmente desenvolvida numa linguagem e num estilo de descrição tais que não privilegiem determinados particionamentos. A saída do processo é um mapeamento entre cada função da especificação e um componente da macro-arquitetura (um processador ou um bloco dedicado de hardware).

O processo de particionamento é evidentemente dependente da macro-arquitetura selecionada. Idealmente, um espaço de soluções muito mais amplo seria explorado se a própria definição da macro-arquitetura (número e tipos de componentes) fosse resultante de um processo automático de particionamento, conforme sugerido na

Figura 2.4(a). Esta abordagem, no entanto, teria uma complexidade computacional maior, motivo pelo qual usualmente uma macro-arquitetura é definida e então particionamentos diversos são explorados sobre a mesma, como mostrado na Figura 2.4(b). Esta abordagem é aceitável num grande número de situações, onde o projeto é na verdade apenas uma variação de um projeto anterior para o qual já foi encontrada uma macro-arquitetura aceitável.

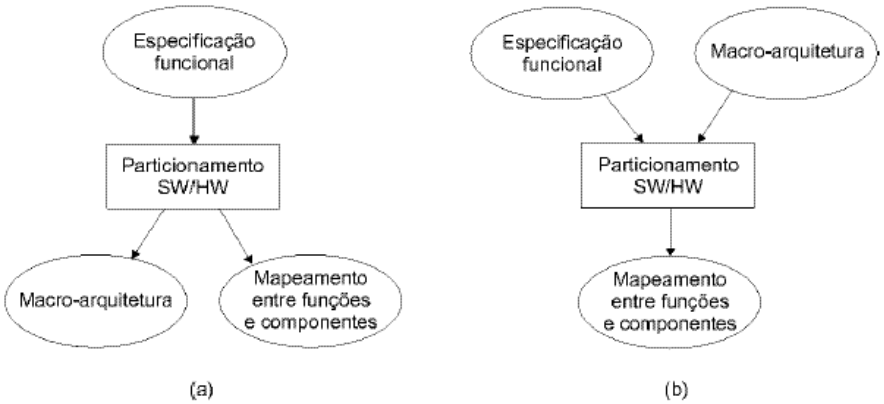


Figura 2.4:Relação entre particionamento e definição da macro-arquitetura.

Há literatura a respeito do particionamento automático entre hardware e software [40]. Uma abordagem clássica supõe inicialmente que o sistema será inteiramente desenvolvido em software, sobre um processador conhecido. Uma avaliação de desempenho, através de um estimador adequado ou de um simulador, permite identificar partes críticas da aplicação, que comprometem restrições temporais. Estas partes são movidas para blocos dedicados de hardware. O processo é repetido até que uma solução aceitável seja encontrada.

Um problema destas abordagens de particionamento automático, no atual contexto de sistemas embarcados, é que elas são geralmente direcionadas ao atendimento de restrições de desempenho, desconsiderando a potência. A maioria delas é restrita à exploração de plataformas arquiteturais contendo um único processador, de tipo previamente conhecido.

A metodologia pragmática atualmente adotada prevê uma exploração manual, como em ferramentas comerciais como CoCentric System Studio, da Synopsys, onde o

próprio projetista faz uma alocação de funções a um ou mais processadores e a blocos dedicados de hardware, todos previamente definidos, interconectados através de uma dada infra-estrutura de comunicação, sendo o resultado deste mapeamento avaliado através de estimadores (de desempenho e de potência, por exemplo). Assim, a busca de uma solução aceitável é feita por tentativas sucessivas, processo que evidentemente não garante que uma solução ótima será encontrada. No entanto, caso os estimadores tenham suficiente precisão e sejam executados com bastante rapidez, o projetista pode encontrar uma solução sub-ótima num curto espaço de tempo. Caso a exploração não alcance nenhuma solução aceitável, uma nova macro-arquitetura deve ser definida e um novo processo de exploração manual deve ser iniciado.

O particionamento de funções entre hardware e software também pode ser aplicado na geração do RTOS, onde uma função crítica como o escalonamento de tarefas pode ser implementada em hardware em um co-processador dedicado.

2.8 Estimadores

A exploração do espaço de projeto num alto nível de abstração depende da existência de ferramentas de estimativas suficientemente precisas dos resultados finais a serem obtidos na implementação do sistema para cada alternativa de projeto considerada. São necessárias estimativas para métricas relevantes de cada projeto, como desempenho, potência e área no silício. A obtenção destas estimativas deve atender três requisitos principais: precisão, fidelidade e velocidade.

A precisão [4] de uma estimativa é uma medida da proximidade entre a estimativa e o valor real da métrica mensurada após a implantação do projeto. A precisão depende do grau de detalhes do modelo do projeto. Estimadores baseados em modelos simples executam rapidamente e são fáceis de serem desenvolvidos, dado que somente algumas características são computadas no cálculo das métricas. Todavia, modelos simples não têm a precisão que possa ser requerida para guiar o projetista na seleção da alternativa de projeto apropriada. Por outro lado, um modelo de projeto pode incorporar diversos aspectos do sistema. Estimadores baseados em modelos detalhados

requerem maior tempo de computação, contudo produzem estimativas mais precisas, e, portanto, possibilitam uma melhor seleção das alternativas de projeto.

A fidelidade [4] de um método de estimativa é definida como o percentual de comparações de previsões corretas entre implementações de projeto. Se os valores estimados de uma métrica de projeto para duas diferentes implementações comparadas com seus respectivos valores medidos suportam o mesmo relacionamento comparativo, então a estimativa compara corretamente as duas implementações. Em outras palavras, a métrica estimada pode ser usada para selecionar a melhor implementação dentre um conjunto de possíveis implementações diversas. Quanto mais alta a fidelidade de um estimador, mais provável que decisões corretas de projeto sejam tomadas baseadas na comparação das estimativas das duas implementações. Durante a seleção de uma dentre diversas implementações de projeto, previsões de qualidade de projeto baseadas em estimativas com alto grau de fidelidade resultarão, em média, em melhores projetos. Fidelidade depende diretamente do modelo usado para estimar o parâmetro do projeto (como por exemplo: custo e desempenho de implementações de hardware e software). Em geral, quanto mais preciso o modelo, maior a fidelidade do estimador.

Se por um lado deseja-se alta precisão e fidelidade, por outro lado, está só pode ser obtida com estimativas feitas em níveis de abstração mais baixos, que usualmente dependem de uma síntese de software e hardware que pode ser demorada e exigir diversas outras decisões de projeto mais detalhadas. Compromete-se assim uma rápida exploração dos potenciais impactos causados por decisões arquiteturais de alto nível. E evidente que, em qualquer caso, as estimativas devem considerar a plataforma arquitetural sobre a qual o sistema embarcado será implementado.

Estimativas podem ser obtidas através da avaliação ou simulação do modelo, num nível de abstração que considere aspectos temporais. O nível de abstração a ser escolhido, no entanto, depende da precisão desejada para a estimativa. Se a estimativa for feita no nível da micro-arquitetura, isto exigirá que o software aplicativo seja compilado para o processador adotado na plataforma e que exista uma descrição detalhada do hardware no nível RT. Isto exigirá uma síntese de software e hardware bastante demorada, o que inviabiliza a idéia da rápida exploração do espaço de projeto.

Como alternativa num nível de abstração mais alto, pode-se anotar o código fonte da aplicação (por exemplo escrito em C) com atrasos estimados para cada comando ou grupo de comandos. Isto exige obviamente uma caracterização prévia do tempo consumido por estes comandos quando executados sobre o processador desejado. Nesta análise o efeito do compilador sobre a qualidade do código gerado pode ser levado em consideração. Também o efeito da implementação do sistema operacional pode ser considerado, desde que existam estimativas razoáveis sobre o custo de funções importantes como o escalonamento de tarefas e a comunicação entre tarefas.

Estimativas de métricas no projeto de sistemas são essenciais por duas razões; primeiro, por permitir ao projetista avaliar a qualidade do projeto, comparando as estimativas de qualquer métrica do projeto com as restrições especificadas por aquela métrica. Por exemplo, se a potência estimada do projeto for maior que a permitida, o projeto terá que ser revisto. Segundo, estimativas permitem ao projetista de sistemas explorar alternativas de projeto, provendo um rápida retro-alimentação para qualquer decisão de projeto. Deste modo, o projetista pode explorar um grande número de alternativas antes de fazer a síntese de sua implementação final [4]. Pode ser necessário definir um *modelo* para estimar cada métrica do sistema. A utilidade da estimativa depende da precisão e/ou fidelidade que o modelo representa o sistema.

2.9 Trabalhos relacionados

Algumas abordagens propõem análises de consumo de potência utilizando estimadores baseados nas instruções do processador. Em [16] a estimativa é obtida através de experimentos. Outras abordagens também baseadas em simulação ao nível de instruções [11], [12], [19], consideram modelos específicos de consumo de potência para os componentes da arquitetura: unidades funcionais, barramentos, registradores, memórias, unidades de controle, que são afetados por cada instrução, de acordo com a atividade de chaveamento das entradas dos componentes e de suas propriedades físicas. Existem ainda abordagens [66] que combinam ambos os modelos de estimativa: baseado em componentes e baseado em instruções.

Outras abordagens [14] propõem uma estimativa de potência mais precisa a partir da simulação no nível dos ciclos de relógio, também considerando a potência dissipada por cada bloco funcional. A implementação de simuladores de código compilado permite compensar o maior tempo de processamento exigido por este nível de abstração.

Embora simuladores de potência no nível de instrução tenham a velocidade como maior vantagem, eles produzem resultados menos acurados e eventualmente até incorretos. A desvantagem dos simuladores no nível de ciclo de relógio, além do maior tempo de processamento, é a necessidade de se dispor de uma descrição detalhada da arquitetura, usualmente no nível RT. Em caso de mudança de arquitetura significa grande mudança dos parâmetros do simulador. Enquanto a maioria dos simuladores deste tipo está restrito à arquitetura de um processador em particular, [15] apresentam um simulador que aceita como entrada a descrição de uma arquitetura qualquer, a partir da interconexão de componentes arquiteturais quaisquer, para os quais, no entanto, devem ser providos modelos adequados de potência.

Uma metodologia e algoritmos para automatizar a análise de potência de *SoCs* foi proposta [68]. A análise prévia da potência do *Systems-on-Chip* (SoC) é crucial para determinar o invólucro apropriado e seu custo. A análise geralmente baseia-se em fórmulas para avaliação da potência para todos os *Cores*, para diversas configurações de tensão, freqüência, tecnologia e aplicação. Na metodologia proposta, fornecida as potências das máquinas de estado para seus *Cores* individuais, é definida a potência resultante da máquina do estado para o *SoC* inteiro. Usando algoritmos de simulação simbólicos formais para percorrer os estados e computar, a potência mínima e a máxima dissipada pelo conjunto de estados no *SoC*. Apesar de ser um método interessante, ele não calcula a potência média dissipada ou mesmo a potência para um contexto conhecido. Ao calcular a potência através de simulação, não se garante que foi percorrido o pior e o melhor caso, dado que a simulação não explora, necessariamente, todas as possíveis condições.

A maioria dos trabalhos na área de redução de potência tem focado em nível de componente, não a nível de sistema. Luca Benini, Robin Hodgson e Polly Siegel

propuseram uma estrutura para estimar e otimizar potência nos projetos, a nível de sistema. O modelo consiste em um conjunto de recursos em um ambiente de especificação de carga de trabalho, e de uma política de gerenciamento da potência, que serve como coração do modelo proposto. Este modelo adota uma estrutura baseada em simulação para obtenção da estimativa de potência dissipada pelo sistema. Foi implementado um algoritmo para aperfeiçoar a política de gerenciamento da potência. O algoritmo de otimização pode ser usado em um pequeno laço com o mecanismo de estimativa, e derivar novos algoritmos de política de gerenciamento de potência para uma dada descrição em nível de sistema. Esta metodologia, por fazer uso da simulação para obtenção das estimativas de potência, não se garante a precisão dos resultados.

A proposta de Greg Stitt and Frank Vahid [28] é a identificar os laços críticos, aqueles que contribuem com o maior tempo de execução em uma aplicação. Ao invés de executar em software tais laços, os laços são reimplementados em hardware, usando a lógica configurável. O software original é modificado, de forma que quando o laço estiver sendo executado, em hardware, o microprocessador entrará no modo de baixo consumo. Quando a execução do laço acabar, ele sinaliza uma interrupção que faça com que o microprocessador recomece a execução normal. Esta modificação melhora o tempo de execução da aplicação, significando que o microprocessador ficará em estado inativo, de baixo consumo, por períodos de tempo mais longos. Alternativamente, pode-se executar o laço na mesma quantidade de tempo com o mesmo tempo de execução, pela diminuição da freqüência de *Clock* e reduzindo a tensão de alimentação do microprocessador. Baixar tensão resulta em menos consumo de energia. Este metodologia trabalha com sistemas já implementados, com objetivo de redução do consumo. Desta forma, tal metodologia não é muito útil na etapa do projeto de sistemas.

2.10 Resumo

Neste capítulo, apresentou-se as arquiteturas, metodologias, etapas do projeto de sistemas digitais modernos. Inicialmente, mostrou-se a importância do menor *Time-to-make,* quanto menor o tempo de projeto, maior o retorno financeiro com o produto.

Mostrou-se também que, umas da formas da redução do *Time-to-make é* a utilização de ferramentas que auxiliem o projetista de sistemas embarcado, para que, ainda em tempo de projeto, possa-se avaliar as diversas restrições. Outra forma de redução do *Time-to-make* está na reutilização de código.

Dando continuidade, diversas arquiteturas possíveis para implementação de um sistemas embarcado foram apresentadas. Dentre estas arquiteturas, comentou-se das arquiteturas especializadas para sistemas embarcados. Foi mostrado que para uma redução de custo do projeto, uma solução muito utilizada é a adoção de um microprocessador como hardware básico do sistema. Como os microprocessadores são encontrados em milhares de projeto, seu custo é diluído. Na arquitetura com microprocessadores, os sistemas é desenvolvido praticamente em *software*. Uma Outra vantagem desta arquitetura, é a possibilidade da criação de varias versões do sistema sem alterações de h*ardware.* Em seguida, falou-se nas etapas de projeto, dentre estas, o particionamento entre hardware e software. Comentou-se sobre os níveis de abstração e linguagens de especificação de projeto. Definiu-se dois importantes conceitos: precisão e fidelidade. Ao final da seção de projeto de sistemas embarcado, detalhou-se a etapa de estimativas e sua importante contextualização no projeto. Por fim, apresentou-se os principais trabalhos de estimas de energia para sistemas embarcados, proposto nos últimos anos.

Capítulo 3 - Fundamentação Teórica

Este capítulo apresenta os principais conceitos sobre redes de Petri estocástica generalizada (GSPN), assim como, características, propriedades e sua aplicabilidade. Primeiro, são apresentados os conceitos básicos sobre processos estocásticos. São introduzidas as cadeias de Markov de tempo contínuo onde são demonstrados os métodos de análise transiente e estacionária. Faz-se também uma revisão bibliográfica sobre redes de Petri.

3.1 Introdução

O estudo de sistemas envolve uma série de parâmetros de entrada e saída. A estes parâmetros, podem ser associados variáveis. Estas variáveis podem ser classificadas como Contínuas ou Discretas. As variáveis contínuas podem assumir qualquer valor no conjunto dos números reais. Enquanto que as variáveis discretas assumem determinado subconjunto de valores. Suas variações são sempre em quantidades pré-definidas, como por exemplo, no conjunto de números inteiros.

Um modelo é uma representação de um ou mais pontos de vistas de um sistema em um determinado nível de abstração. Em um modelo formal, um sistema é representado através de um conjunto de variáveis, funções e relações matemáticas que representam uma ou mais perspectivas do sistema.

Enquanto um sistema é “algo real” o modelo é uma “abstração”, como por exemplo, um conjunto de equações matemáticas. A Figura 3.1 mostra a representação de um sistema através de um modelo.

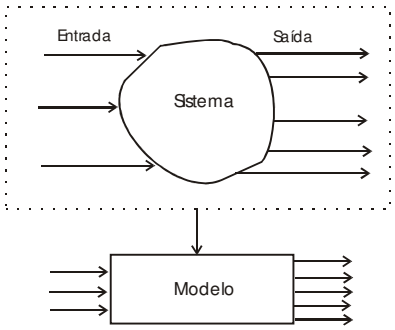


Figura 3.1:Processo de modelagem simples.

A classificação apresentada não é exclusiva e tem a finalidade de descrever o escopo dos diferentes aspectos da teoria de sistemas, assim como ajuda identificar as principais características dos sistemas. A Figura 3.2 fornece uma referência rápida para as classificações de sistemas [22].

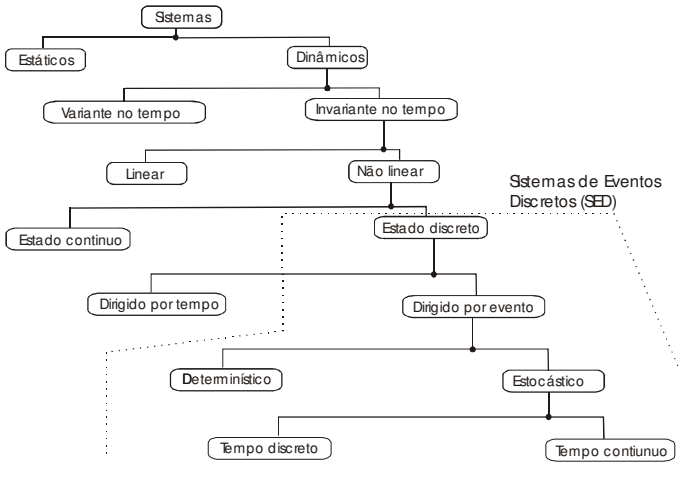


Figura 3.2:Principais classificações de Sistemas.

Diversos modelos de representação têm sido utilizados para representar sistemas computacionais, dentre eles, redes de filas, álgebras de processos, autômatos e redes *de Petri* são os mais utilizados. Descrições detalhadas desses modelos são encontradas em [22].

A não linearidade dos Sistemas de Eventos Discretos (SED) é inerente nas descontinuidades (saltos) das transições entre os estados resultante da ocorrência de um evento. Dentro da classe de SED podem-se considerar modelos de sistemas determinísticos ou estocásticos, de estados discretos ou contínuo, e de tempo discreto ou tempo contínuo.

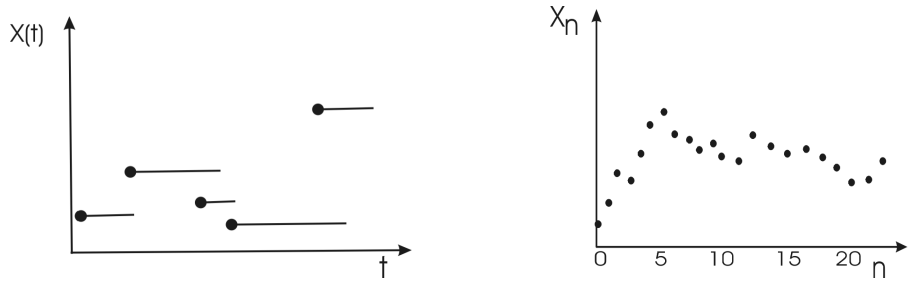
3.2 Processos Estocásticos

A avaliação de desempenho e confiabilidade de sistemas através de métodos analíticos tem sido objeto de numerosos estudos nas áreas de ciência da computação e matemática aplicada. Processos estocásticos são modelos matemáticos úteis para a descrição de um fenômeno de uma natureza probabilística, como uma função de um parâmetro, que, usualmente, tem o significado de tempo.

Um processo estocástico $\{X(t),\ t \in T\}$ é uma família de variáveis aleatórias definidas sobre o mesmo espaço de probabilidades, indexadas por um parâmetro t e assumindo valores no espaço de estado S. Geralmente, t tem o significado de tempo, e T intervalo de tempo, e o espaço de estado S compreende o conjunto dos possíveis valores de número de $X(t)$ [52].

Classificam-se os processos estocásticos de acordo com o tipo da variável aleatória associada ao processo. O processo estocástico de tempo contínuo tem associado uma variável aleatória de tempo contínuo, e espaço de estados pode ser discreto ou contínuo. Enquanto que o processo estocástico de tempo discreto tem associado uma variável aleatória de tempo discreto, e, da mesma maneira, espaço de estado poder ser discreto ou contínuo.

A Figura 3.3 mostra a representação gráfica de dois processos estocásticos, (a) de tempo contínuo,(b) de tempo discreto. Ambos têm espaço de estado discreto.



(a) tempo continuo (b) tempo discreto

Figura 3.3:Amostras de um processo estocástico.

Em geral, a descrição probabilística completa de um processo aleatório genérico não é factível. Processos Markovianos são uma classe especial de processo estocástico, para o qual a descrição probabilística é simples e de relevância particular.

3.3 Processos Markovianos

Definem-se Processos Markovianos como os processos estocásticos que satisfazem as propriedades Markovianas:

M_1 – A evolução futura dos estados não depende dos estados anteriores;

M_2 – A evolução futura dos estados não depende do tempo de permanência no estado atual.

As propriedades Markovianas também são conhecidas por ausentes de memória. Denominam-se os processos Markovianos de tempo contínuo e espaço de estado discreto de Cadeias de Markov de Tempo Contínuo (CTMC). O processo Markovianos com espaço de estado discreto e tempo discreto, são denominados de Cadeias de Markov de Tempo Discreto (DTMC).

3.4 Cadeias de Markov

O Modelo Markoviano herdou este nome pelo trabalho pioneiro desenvolvido pelo matemático russo A.A. Markov no início do século XX. De fato, seu trabalho iniciou os

estudos na área dos processos Estocásticos. Nas primeiras duas décadas do século XX, o matemático A.K. Erlang aplicou técnicas Markovianas para resolver problemas planejamento de capacidade para a companhia de Telefone Copenhagen. Seu modelo foi adaptado por outros, entre eles pelo Post Office. O matemático russo A.N. Kolmogorov consolidou as cadeias de Markov nos anos 30 do século passado.

Durante todo o século XX o trabalho destes pioneiros se tornou mais compreendido e difundido. Atualmente, os processos estocásticos e as Cadeias de Markov têm importância fundamental para avaliações de desempenho e confiabilidade de sistema, em muitas áreas da ciência e engenharia. Por exemplo, na biologia tem sido utilizado para modelar o crescimento e a diminuição das populações; na física, para modelar iterações entre partículas elementares; na engenharia química, para modelar as reações (cadeias) entre moléculas ou para modelar processos de mistura; na ciência de gerência para modelar o fluxo dos produtos ou sistemas de produção flexível ou para modelar a disponibilidade de linhas de produção e, principalmente, em computadores e na ciência e na engenharia de comunicação, para modelar o desempenho e dependência do sistema em uma variedade parâmetros.

3.4.1 Cadeias de Markov de Tempo Contínuo (CTMC)

As Cadeias de Markov de Tempo Contínuo é uma classe de processos estocásticos com espaço de estado enumerável onde as variáveis aleatórias $\{X(t): t \in T\}$ são de tempo contínuo e satisfazem as propriedades Markovianas, isto é, ausência de memória.

A representação padrão de uma CTMC é feita usando diagramas de transições entre estados, isto é, com grafos dirigidos rotulados. Os estados da CTMC são representados pelos vértices. Aos arcos são rotulados as taxas de transição correspondentes entre estados (λ).

Em uma CTMC, as probabilidades de transição de estado são calculadas pela equação:

$$p_{i,j}(s,t) = P\{X(t) = j \mid X(s) = i\}$$

O comportamento do processo markoviano é descrito pelas equações de probabilidade de transição de Champman-Kolmogoroff [59]

$$p_{i,j}(s,t) = \sum_{todo\ r} p_{i,r}(s,u) \cdot p_{r,j}(u,t), \quad s \le u \le t$$

Onde *i, r* e *j* são estados; *s*, *u*, e *t* instantes de tempo.

Desenvolvendo-se esta equação (na notação matricial), obtêm-se a equação diferencial:

$$\frac{d\pi(t)}{dt} = \pi(t)Q$$

Onde: $\underline{\pi}(t)$ – é o vetor das probabilidades de estado *i* no tempo *t*;

$\underline{\pi}(0)$ – é o vetor das probabilidades de estado no tempo inicial;

Q – é a matriz das taxas de transição entre os estado;

$\frac{d\underline{\pi}(t)}{dt}$ - é a primeira derivada do vetor probabilidade com relação ao tempo.

A solução transiente, ou dependente do tempo, é importante quando o sistema a avaliar é dependente do tempo. Para modelos ergódicos [5], considerando tempos de execução longos, pode-se mostrar que a probabilidade dos estados converge para valores constantes [59]. O comportamento transiente da cadeia de Markov nos fornece uma informação de desempenho precisa sobre os instantes iniciais do sistema. Assumindo-se que a probabilidade $\pi(t)$ é independente do tempo, isto é, $\pi_i = \lim_{t\to\infty} \pi_i(t)$ (homogeneidade), consequentemente, $\pi'(t) = 0$, portanto:

$$\underline{\pi}Q = \underline{0}, \qquad \sum_{i=1}^{N} \pi_i = 1$$

A equação da direita é a condição de normalização, adicionada para assegurar que a solução obtida é um único vetor de probabilidade. A parte esquerda, sozinha, teria um conjunto de soluções infinitas. Normalizando as soluções, chega-se em um único vetor de probabilidades.

3.4.2 Métodos de soluções para probabilidade em estado estacionário.

Para computar as probabilidades em estado estacionário de uma CTMC finita, com *N* estados, faz-se necessário resolver o sistema de *N* equações lineares.

$$\underline{\pi}Q = \underline{0}, \qquad \sum_{i=1}^{N} \pi_i = 1$$

Assumindo-se que a cadeia de Markov é irredutível, e que $\underline{\pi}$ existe e é independente de $\underline{\pi}(0)$, para se solucionar o sistema de equações, dois tipos de métodos de solução podem ser escolhidos: Métodos Diretos e Métodos Iterativos. Um método é chamado *direto* quando fornece a solução exata desejada após um número finito de passos. Um método é chamado iterativo quando fornece uma seqüência de soluções aproximadas que convergem para o valor exato.

Métodos diretos:

A principal característica dos chamados métodos diretos é que eles focam na reescrita do sistema de equações de tal forma que se obtêm expressões explicitas para o cálculo das probabilidades em estado estacionário.

- **Eliminação Gaussiana**

  Esta técnica de solução direta é a mais conhecida. A eliminação de Gauss [59] consiste em duas fases: uma fase de redução e uma fase de substituição. Tem o objetivo de transformar o sistema linear original num sistema linear equivalente triangular. O sistema equivalente transforma-se numa matriz triangular superior,

ou seja, a diagonal e a parte superior é diferente de zero. Desta forma, obtêm-se a solução desejada.

- **Decomposição LU**

  A vantagem deste método[60] é quando sistemas de equações múltiplas têm que ser resolvidas, todas na forma $A\underline{x} = \underline{b}$, para diferentes valores de $\underline{b}$. O processo de fatoração para resolução do sistema $A\underline{x} = \underline{b}$, consiste em decompor a matriz *A* em produto de duas matrizes *L* e *U*. Tem-se:

$$A\underline{x} = \underline{b} \Rightarrow L\underbrace{U\underline{x}}_{\underline{z}} = \underline{b}$$

  Em seguida, resolve-se $L\underline{z} = \underline{b}$ e depois $U\underline{x} = \underline{z}$. Como os dois sistemas de equações são triangular, encontra-se a solução por substituição simples.

Por razões computacionais e eficiência de memória os métodos diretos não são utilizados quando o número de estados é maior que mil (1000). Nestas condições, utiliza-se dos métodos iterativos.

Métodos iterativos:

Os Métodos Iterativos são métodos numéricos para resolução de um sistema linear, que geram uma seqüência de vetores $\{x^{(k)}\}$, a partir de uma aproximação inicial $x^{(0)}$. Sob certas condições, esta seqüência converge para uma solução, caso ela exista.

Os procedimentos iterativos não resultam em uma solução explícita do sistema de equações. Não sendo possível estimar quantas interações serão necessárias para obter-se a precisão desejada. Os métodos iterativos mais conhecidos são:

- **O Método da Potência**

  O Método da Potência [60] faz sucessivas multiplicações do vetor π de probabilidade de estado estacionário com o vetor de probabilidades *P,* até que a

convergência para uma solução seja alcançada. O método da potência é o mais simples dos métodos, porém na prática ele não é eficiente

- **O Método de Jacobi**

Os métodos iterativos mais conhecidos são Jacobi [60] e Gauss-Seidel. Nestes métodos, reescreve-se a *i*-éssima equação do sistema linear e atribui-se um valor inicial ao vetor $\underline{\pi}^{(0)}$ para iniciar o processo interação.

Há duas formas de estimar os valores iniciais para $\underline{\pi}^{(0)}$: quando se conhece uma solução aproximada, que pode-se usar como valor inicial; ou escolher a distribuição uniforme que é uma escolha razoável, assim: $\underline{\pi}_i^{(0)} = \frac{1}{N}$.

O processo iterativo é repetido até que o vetor $\underline{\pi}^{(k)}$ esteja suficientemente próximo do vetor $\underline{\pi}^{(k-1)}$. É necessário fazer um teste de parada dessas iterações. O teste de parada é obtido através da diferença: máx$|\underline{\pi}^{(k+1)} - \underline{\pi}^{(k)}| < \varepsilon$, ($\varepsilon$ é uma precisão). Quando a diferença for muito pequena, nem sempre implica que o vetor solução foi encontrado, pois, pode ser que a convergência para a solução seja muito lenta.

- **O Método de Gauss-Seidel**

O método de Jacobi [60] requer o armazenamento de ambos $\underline{\pi}^{(k)}$ e $\underline{\pi}^{(k+1)}$ durante as iterações. No método Gauss-Seidel a computação da solução é estruturada de tal forma que estimativas ($k+1$)-éssimas são usadas tão logo elas tenham sido computadas.

Assume-se que a computação é de $\pi_1^{(k+1)}$ para $\pi_N^{(k+1)}$. Assim, necessita-se armazenar apenas um vetor de probabilidade.

- **O Método SOR**

O último método mencionado é o *Successive Over-Relaxation (SOR). SOR* [60] é uma extensão do método de Gauss-Seidel, onde o vetor $\underline{\pi}^{(k+1)}$ é computado como a média ponderada do vetor $\underline{\pi}^{(k)}$ e o vetor $\underline{\pi}^{(k+1)}$, e será utilizado na iteração de Gauss-Seidel.

Tem-se um $w \in (0,2)$ que é o fator relaxação. Quando w=1, este método é reduzido as iteração Gauss-Seidel.

O método da potência converge mais lentamente para o resultado, em seguida vem o método de Jacobi e depois o Gauss-Seidel [60].

Avaliações de modelos em estado estacionário, na maioria dos sistemas, são suficientes para cálculos das métricas. Existe, contudo, exceções, como por exemplo: quando se deseja uma precisão mais apurada das métricas; ou quando o tempo de vida do sistema é tão curto que o estado estacionário não é atingido; ou quando o período para atingir a situação de estado estacionário, é o período de interresse; ou quando não existe solução em estado estacionário. Para os casos citados, devem-se obter as probabilidades de estado dependentes do tempo, isto é, em estado transiente.

3.4.3 Métodos de solução para probabilidades em estado transiente.

Dada uma cadeia de Markov, especificada pelo seu espaço de estados *S*, matriz das taxas de transições *Q* e o vetor de probabilidades inicial $\pi(0)$, deve-se determinar o vetor de probabilidades em estado transiente $\pi(t)$.

As probabilidades de estado transiente de um CTMC são especificadas por um sistema de equações lineares:

$$\underline{\pi}'(t) = \underline{\pi}(t)\, Q, \quad dado\ \underline{\pi}(0) \tag{2}$$

As medidas são especificadas em termos de $\underline{\pi}(t)$ e chamadas de medidas de instantes do tempo. Existem diversos métodos de resolução da equação diferencial (2) para computação das probabilidades transientes.

A solução numérica de sistemas de equações diferenciais do tipo (2) tem, desde muito tempo, sido um tópico importante em matemática numérica. Muitos procedimentos numéricos têm sido desenvolvidos para esta finalidade. Entre eles, os mais utilizados são:

- **Métodos de Runge-Kutta.**

  No métodos Runge-Kutta (R.K.) [61], a função vetorial contínua $\underline{\pi}(t)$ que surge da equação diferencial $\underline{\pi}'(t) = \underline{\pi}(t)Q$, dado $\underline{\pi}(0)$, é aproximada por uma função discreta $\underline{\tilde{\pi}}_i (i \in \aleph)$, onde $\underline{\tilde{\pi}}_i \approx \underline{\pi}(ih)$, onde h é o tamanho do passo na discretização. Quando menor o valor de *h,* mais precisa e onerosa é a solução. Os métodos R.K., também chamados *métodos de passo simples,* pois apenas a última iteração ($\underline{\tilde{\pi}}_i$) é usada para calcular a iteração seguinte ($\underline{\tilde{\pi}}_{i+1}$).

- **Uniformização para medidas transientes.**

  O Método de Uniformização para medidas transientes [61], também conhecido como método de Jensen ou randomização, consiste em calcular as probabilidades (π_i) através da solução para a equação (2) em:

$$\underline{\pi}(t) = \underline{\pi}(0) \cdot e^{Q \cdot t}$$

  Porém, têm-se alguns problemas em se trabalhar com funções exponenciais e a matriz *Q,* entre eles, a existência de coeficientes negativos em *Q*. A solução adotada trabalha com a matriz *P*, que se obtêm a partir da matriz:

$$Q.P = I + \frac{Q}{\lambda} \quad \Rightarrow \quad Q = \lambda(P - I)\mathrm{E}$$

Escolhe-se λ tal que $\lambda \geq max_i\{|q_{i,i}|\}$, então os coeficientes de P são todos entre 0 e 1, enquanto as linhas de P somam 1. Em outras palavras, P é uma matriz estocástica e descreve uma DMTC. O valor de λ é chamado de taxa de uniformização. Para aumentar a eficiência da uniformização em situações especificas, várias variantes, tem sido desenvolvidas. Uma boa avaliação pode ser encontrada em [1],[3],[61].

3.5 Redes de Petri

Antes de avançar no detalhamento da técnica de estimativas de consumo, é importante visitar algumas noções do formalismo, introduzido em 1962 por Carl Adam Petri e, largamente conhecido como Redes de Petri. Todos os procedimentos deste trabalho estão diretamente ligados às redes de Petri. Sendo esta, uma vasta área de pesquisa. Detalha-se neste capítulo noções necessárias para o completo entendimento desta aplicação.

3.5.1 Introdução

Atualmente redes de Petri é um termo genérico utilizado para denotar um conjunto de formalismos com uma interpretação gráfica utilizada para representar o comportamento de sistemas de eventos discreto. É uma ferramenta utilizada para descrever e estudar sistemas de processamento caracterizados como concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e/ou estocásticos. Seu uso em metodologias de projeto de hardware se torna particularmente interessante pois estas aplicações requerem um formalismo rigoroso, mas também de fácil compreensão pelo projetista.

Tradicionalmente, por exemplo, um dos modelos mais utilizados para modelamento de hardware são as máquinas de estado finito (MEF). A principal característica das MEF é que o sistema é definido de uma forma seqüencial. Os melhores esforços para representar aspectos como concorrência e paralelismo, neste tipo de abordagem, levam a *interleavings* e composições de várias máquinas de estado, incorrendo, frequentemente, em problemas de explosão de estados. As redes de Petri,

em contra-partida, oferecem uma abordagem natural à representação dos conceitos de paralelismo e concorrência devido a sua semântica. Adicionalmente, as metodologias para análise, verificação e validação de modelos baseados em rede de Petri são largamente estudadas e estabelecidas na literatura, e em especial interesse deste trabalho, diversas abordagens enfocam a modelagem de sistemas de hardware [28].

Hoje em dia, redes de Petri é considerada uma técnica para especificação de sistemas consolidada. Grupos de pesquisa em todo o mundo têm redes de Petri como tema de pesquisa, desenvolvendo estudos sobre seus aspectos teóricos e suas aplicações.

Diversas técnicas de modelagem matemática de sistemas em diversas áreas da ciência têm sido propostas. Conforme classificação:

Técnicas Baseadas em estados: Fornece uma descrição abstrata explícita sobre estados e operações que transformam os estados, no entanto não oferece meios explícitos para especificar concorrência.

Técnicas Baseadas em Álgebra de Processos: Estas técnicas fornecem meios explícitos para especificar concorrência. O comportamento dos processos é representado através de ações observáveis.

Técnicas Baseadas em Lógica: Uma grande variedade de técnicas baseadas em lógica tem sido postas, onde se analisam as relações causais e aspectos relacionados à temporização.

Técnicas Baseadas em Redes: Estas técnicas modela concorrência, através de mecanismos implícitos de fluxo de *tokens* na rede. Este fluxo é controlado por condições que habilitam a realização de tarefas (eventos). Ex.: Redes de Petri.

3.5.2 Componentes de uma rede de Petri.

O grafo de uma rede de Petri é direcionado e bipartido, possui pesos e dois tipos de vértices, chamados lugares *(places)* e transições *(transitions)* com arcos que vão de um lugar a uma transição ou de uma transição para um lugar.

Graficamente, representa-se lugares com círculos e transição com barras ou retângulos (ver Figura 3.4). Os arcos são rotulados com seus pesos (exceto se o peso for igual a um[1]), e um arco de peso *k* pode ser interpretado como *k* arcos paralelos.

Figura 3.4:Elementos básicos de uma rede de Petri.

Lugares e transições representam respectivamente condições e eventos. Os lugares, portanto, são ditos componentes passivos e as transições são componentes ativos. Uma transição possui um certo número de lugares de entrada e saída, o que representa as pré e póscondições do evento observado. A realização de uma ação, portanto, está associada a algumas pré-condições, ou seja, existe uma relação entre os lugares e as transições, que possibilita a realização de uma ação. De forma semelhante, após a realização de uma ação, alguns lugares terão suas informações alteradas (pós-condições). Algumas interpretações típicas de transições e seus lugares de entrada e saída são mostradas na Tabela 3.1.

Tabela 3.1: Algumas interpretações típicas para lugares e transações

Lugares de entrada	Transição	Lugares de Saída
Pré-condições	Evento	Pós-condições
Dados de entrada	Passo computacional	Dados de saída
Sinais de entrada	Processador de sinais	Sinais de saída
Recursos requeridos	Tarefa	Recursos liberados
Condições	Cláusula lógica	Conclusões
Buffers	Processador	*Buffers*

[1] Dá-se o nome de arco *monovalorado*

Dado os conceitos informalmente apresentados sobre redes de Petri ilustramos a seguir, segundo [5], um pequeno exemplo para entendimento: o ciclo repetitivo dos turnos (períodos) de um dia. Dividamos o dia em três períodos: *manhã, tarde* e *noite,* ou seja, há três condições. A transição de uma dessas condições para uma outra, por exemplo- amanhecer *(noite* $\rightarrow$ *manhã),* são os eventos. O modelo que representa o ciclo operacional desse sistema é formado pelas três condições, representadas por três variáveis de estado (lugares), e por três eventos (transições): *amanhecer, entardecer* e *anoitecer.* Para representar a situação atual, ou seja, em que a condição encontra-se o sistema modelado, usa-se uma marca grafada (um ponto) no lugar que corresponde a essa situação, por exemplo: a condição atual é *manhã.* Na Figura 3.5a tem-se o modelo que representa este sistema e a sua situação atual.

Figura 3.5:Períodos do dia representado com redes de Petri.

Nesse modelo, temos a condição atual representada pela marca no lugar *manhã.* Estando nesta condição o único evento que poderá ocorrer é *entardecer,* que é representado pela transição de mesmo nome. Na ocorrencia desse evento, teremos uma nova situação atual, ou seja, *tarde,* que é representada graficamente na Figura 3.5.b por uma marca no lugar *tarde.*

3.5.3 Redes place-transition

Pode-se definir mais formalmente as redes de Petri de três maneiras distintas. Uma sobre o ponto de vista da teoria *bag*[2] [5], que apresenta mapeamentos das transições para os *bags* de lugares. Um segundo tipo de definição usa a notação matricial. E a terceira, define as Redes de Petri através de relações e pesos associados a estas relações.

[2] Um *bag é* uma generalização do conceito de conjunto que admite a repetição de elementos.

Neste trabalho será apresentado apenas o segundo tipo de definição, a notação matricial.

3.5.4 Estrutura definda em matriz

Devido aos resultados existentes nos estudos da álgebra matricial, as redes de Petri utilizam este ferramental para a formalização de sua teoria, possibilitando a análise de propriedades comportamentais e estruturais, posteriormente apresentadas neste capítulo.

Segundo o ponto de vista matricial, a estrutura das redes de Petri é apresentada por uma quíntupla formada pelo conjunto de lugares, o conjunto de transições, a matriz de entrada das transições, a matriz de saída das transições, a capacidade de cada lugar, formalmente apresentada abaixo:

Definição 3.1 (Estrutura de Redes de Petri em Matrizes) Seja a estrutura de uma rede definida por uma quíntupla *R (P, T, I O, K)*, onde *P* é um conjunto finito de lugares. *T* é um conjunto finito de transições. $I : (P \times T) \rightarrow \aleph$ é a matriz de pré-*condições.* $O : (P \times T) \rightarrow \aleph$ *é a matriz de pós-condições. K é o vetor das capacidades associadas aos lugares(*$K : P \rightarrow \aleph \cup \{\Omega\}$**).**

Se o conjunto de lugares ou o conjunto de transições é vazio, a rede é dita degenerada.

3.5.5 Rede de Petri marcada

Uma marca[3] é um conceito primitivo em redes de Petri , tal qual lugar e transição. As marcas são informações atribuídas aos lugares. Uma marcação, ou estado, associa um inteiro *k,* não-negativo, a cada lugar da rede. Abaixo seguem as seguintes definições: vetor marcação; a definição de rede de Petri marcada.

[3] Pode também ser denominado *ficha* ou *token.*

Definição 3.2 (Marcação) *Seja P o conjunto de lugares de uma rede R. Define-se formalmente marcação como uma função que mapeia o conjunto de lugares P a inteiros não-negativos M: P* $\rightarrow \aleph$.

Definição 3.3 (Vetor Marcação) *Seja P o conjunto de lugares de uma rede R. A marcação pode ser definida como um vetor M (*$M(p_1)$, ..., $M(p_n)$*), onde n= #P, tal que* $M(p_i) \in \aleph$*, para todo* $p_i \in P$.

Definição 3.4 (Rede Marcada) *Define-se uma rede de Petri marcada pela dupla RM (R;* M_0*), onde R é a estrutura da rede e* M_0 *é a marcação inicial.*

O comportamento dos sistemas pode ser descrito em função dos seus estados e suas alterações. Para simular o comportamento dos sistemas, a marcação da rede de Petri é modificada a cada ação realizada (transição disparada), segundo regras de execução apresentadas a seguir.

3.5.6 Regras de Execução

O disparo de transições (execução das ações) é controlado pelo número e distribuição de marcas nos lugares. Uma transição está habilitada se cada um dos seus lugares de entrada possuirem um número de *tokens* pelo menos igual ao peso do arco que os liga. Denota-se a habilitação de uma transição *t* para uma marcação M_k por $M_k[t>$.

Definição 3.5 (Regra de Habilitação) *Seja RM (R;* M_0*) uma rede de Petri marcada,* $t \in T$ *uma transição e* M_k *uma marcação. Se* $M_k[t>$, $M_k(p_i) \geq I(p_i,t)$, $\forall p_i \in P$.

Formalmente, apresentamos as regras de execução das redes de Petri a seguir:

Definição 3.6 (Regra de Disparo) *Seja RM = (R;* M_0*) uma rede de Petri marcada,* $t \in T$ *uma transição e uma marcação M. A transição t pode disparar quando ela está habilitada. Disparando uma transição habilitada, a marcação resultante é* $M = M_0 - I(p_j,t) + O(p_j,t)$, $\forall p \in P$. *Se uma marcação M é alcançada por* M_0 *pelo disparo de uma transição t, ela é denotada por* $M_0[t> M$.

3.5.7 Matriz de Incidência

A *matriz de incidência* representa a estrutura da rede em termos do peso de todos os arcos. Tal definição é de suma importância desde que esta estrutura é a base para uma análise estrutural e para descrição do comportamento da rede em termos da equação de estado.

Uma maneira prática de representar a matriz de uma rede é através de pré e pós-condições de incidência. Estas matrizes representam as interconexões entre lugares e transições e também seus pesos. Quando uma transição t dispara, a diferença entre a nova marcação alcançável e uma anterior é obtida por $O(p_i, t) — I(p_i, t)$, considerando cada lugar p_i.

A matriz $C = O - I$ é chamada matriz de incidência. Esta matriz representa a estrutura da rede do sistema ora modelado. Abaixo, segue uma definição formal de uma matriz de incidência:

Definição 3.7 (Matriz de Incidência) *Seja R = (P, T, 1, O, K) ser uma rede de Petri.*

A matriz de incidência C representa a relaçao $(P \times T) \rightarrow Z$ *definida por* $C(p, t) = O(p,t) - I(p,t), \forall p \in P, \forall t \in T$.

3.5.8 Grafo de Marcações Acessíveis

O grafo das marcações acessíveis (ou alcançáveis) é uma representação gráfica do conjutno das marcações que podem ser alcançadas para uma dada rede de Petri. O disparo de uma transição modifica a marcação, conforme a marcação atual e a estrutura da rede. Essas marcações obtidas após os disparos das transições são marcações acessíveis de uma rede para uma determinada marcação inicial.

Definição 3.8 (Conjunto de Marcações Acessíveis) *Seja uma rede marcada RM (R; M_0); define-se conjunto das marcações acessíveis CA(R; M_0) pelo conjunto demarcações obtidas a partir de uma marcação inicial M_0 e pelo disparo de todas as*

possíveis seqüências de transições habilitadas, ou seja, CA(R; M_0)= { $M_i \in \aleph^m$}, tal que existe uma seqüência de transições sq, que M_0 [sq > M_i.

O grafo das marcações acessíveis pode ser definido por um par *GA*(*R; M_0*)= (*CA*(*R; A*),), onde *CA*(*R; M_0*) são os vértices do grafo e *A* os arcos. Os arcos representam a alteração da marcação pelo disparo das transições. Sempre que o contexto esteja claro, utiliza-se *CA* ao invés de *CA*(*R; M_0*).

Definição 3.9 (Grafo das Marcações Acessíveis) *Seja uma rede marcada RM = (R;M_0); define-se grafo das marcações acessíveis GA, por um par GA(R; M_0)* (*CA, $A_{(i,j)}$*)*, onde $M_i \in$ CA(R; M_0) são os vértices do grafo e A os arcos. Os arcos representam M'[> M" se, e somente se, existe uma transição t tal que M'[t > M" (mudança de marcação).*

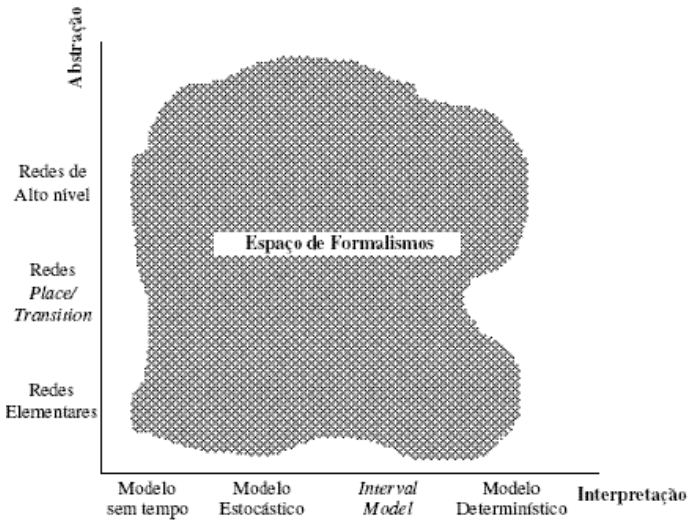


Figura 3.6:Abstração X Interpretação.

3.5.9 Redes de petri e suas classes

Dois aspectos ortogonais associados às redes de Petri que devem ser resaltados são: o nível de abstração e as diferentes interpretações. Estas características especificam um espaço de formalismos apropriados para diferentes propósitos e são apresentados na Figura 3.6. O primeira nível de abstração são as *redes elementares,* onde lugares

representam condições boleanas. No segundo nível estão as redes *Place-Transition,* onde lugares são variáveis onde seu domínio são inteiros não negativos. Adicionalmente algumas extensões tem sido consideradas, tais quais arco inibidores e redes com prioridades. Tais extensões afetam não somente a concisão, mas também o poder de modelagem dos modelos adotados. A interpretação mostrada basicamente no eixo x modela em redes de Petri várias interpretações temporizadas incluindo modelos determinísticos , disparo no tempo representado por intervalos, modelos estocásticos etc, adequadas para análises de desempenho, escalonamento, controle de tempo real, etc.

As subclasses de redes são definidas pela introdução de restrições a estrutura de redes *Place- Transition.* Pelo fato de restringir a generalidade de um determinado modelo, torna-se fácil estudar seus comportamentos [28]. Este trabalho considera a construção de modelos baseado na classe de redes de Petri denominada *Extended Free-Choice Nets* (redes Livre de Escolha estendida).

3.5.10 Redes Free-Choice

O nome desta subclasse implica que este modelo pode modelar conflito, no entanto, o mais importante é ficar claro que nele é permitido um mecanismo para a resolução do conflito. Na verdade, esta subclasse de rede permite modelar concorrência, sincronização e modelos de conflitos, no entanto, de uma maneira mais restrita do que a rede *PlaceTransition.*

Definição 3.10 (Redes *Free-Choice*) *Seja RM = (R; M_0) é uma rede de Petri marcada classificada como uma free-choice* $\forall p \in P$, $I(t_j)= \{p_j\}$ *ou* $O(p_i)= \{t_j\}$.

Esta classe de rede permite ter conflitos controlados, pois quando um lugar é entrada de diversas transições, este lugar é a única entrada destas transições. Desta fotma, todas as transições estarão habilitadas ou nenhuma estará, possibilitando a escolha da realização do evento livremente.

3.5.11 Propriedades das redes de Petri

Em paralelo aos modelos apresentados foi desenvolvido uma série de métodos que permitem a análise de um grande número de propriedades em sistemas [24]. Dividem-se em dois tipos as propriedades das redes de Petri : as propriedades dependentes da marcação e as independentes da marcação, denominadas propriedades comportamentais e estruturais, respectivamente.

3.5.12 Propriedades Comportamentais

Alcançabilidade - Alcançabilidade ou *reachability* é de suma importância para o estudo de propriedades dinâmicas dos sistemas. A alcançabilidade indica a possiblidade de atingirmos uma determinada marcação pelo disparo de um número finito de transições, a partir de uma dada marcação. Será apresentado adiante este conceito no contexto de redes de Petri [24]. Dada um determinada rede de Petri marcada $RM = (R; M_0)$, representada na Figura 3.7, o disparo de uma transição t altera a marcação da rede, conforme as regras descritas a seguir. Uma marcação M' é dita acessível de M_0 se existe uma seqüência de transições que, disparadas, levam a marcação M'. Ou seja, se a marcação M_0 habilita a transição t_0, disparando-se esta transição atinge-se a marcação M_1. A marcação M_1 habilita t_1 a qual sendo disparada atinge-se a marcação M_2 e assim por diante até a obtenção da marcação M'. Mais formalmente temos:

Definição 3.11 (Alcançabilidade) *Seja* $M_i[t_j > M_k$ *e* $M_k[t_h > M_1$ *então* $M_i[t_j t_h > M_1$. *Por recorrência designamos o disparo de uma seqüência* $s \in T^*$ *por* $M[s > M'$. *O conjunto de todas as possíveis marcações obtidas a partir da marcação* M_0 *na rede* $RM = (R; M_0)$ *é denotado por* $CA(R; M_0) = \{M' \in \aleph^m | \exists s , M_o[s > M'\}$, *onde m é a cardinalidade do conjunto de lugares da rede.*

A análise da alcançabilidade de uma marcação consiste em determinarmos se uma dada marcação $M' \in CA(R; M_0)$ da rede marcada *RM*. Em alguns casos, deseja-se observar apenas alguns lugares específicos da rede em estudo. Este problema *é* denominado *sub- marcação alcançável (submarking reachability).*

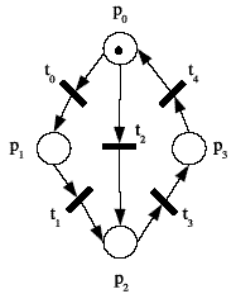


Figura 3.7:Rede de Petri Marcada.

Muitos outros problemas de análise podem ser observados em termos do problema de *alcançabilidade.* Por exemplo, se uma rede fica em *deadlock* em uma determinada marcação, pode-se querer saber se essa marcação é acessível.

Limitação Abaixo segue definição formal do conceito de limitação *(boundedness)* nas redes de Petri e sua importância na verificação de uma especificação de sistemas.

Definição 3.12 (Limitação) *Seja um lugar $p_i \in P$, de uma rede de Petri marcada $RM = (R; M_0)$. Este lugar são ditos k-limitados (k-bounded) ($k \in \mathbb{N}$) ou simplesmente limitados se para toda marcação acessível $M \in CA(R; M_0)$, $M(p_i) \leq k$.*

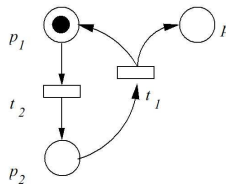


Figura 3.8:Exemplo de rede não limitada.

Caso esta propriedade não seja identificada, o lugar é dito *não-limitado.* Perceba que o limite *k é* o número de marcas que um lugar pode acumular. Uma rede de Petri marcada *RM = (R; M_0) é* dita k-limitada se o número de marcas de cada lugar de *RM* não excede *k* em qualquer marcação acessível de $RM(M(p) \leq k, \forall M \in GA(R; M_0), \forall p \in P)$.

Definição 3.13 (Rede Limitada) *Diz-se que uma rede $RM = (R; M_0)$ é limitada (bounded) se $k(p_i) \leq \infty$, $\forall p \in P$.*

Segurança O conceito de segurança *(safeness) é* uma particularização do conceito de limitação. Um lugar p_i *é* dito *k-limitado* se o número de marcas que este lugar pode acumular estiver limitado ao número *k.* Um lugar que é *1-limitado* pode ser simplesmente chamado de seguro *(safe).*

Definição 3.14 (Lugar Seguro) *Seja um lugar* $p_i \in P$ *de uma rede marcada RM = (R; M₀), onde* p_i *é seguro se para toda marcaçao* $M' \in CA(R; M_0)$ *tivermos* $M(p_i) \leq 1$.

Dizemos que a rede *RM (R; M₀) é* segura se todos os lugares dessa rede são seguros, ou seja, todos os lugares desta rede podem conter uma ou nenhuma marca.

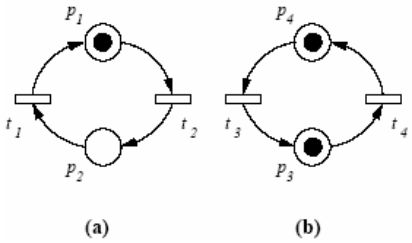


Figura 3.9:Rede Segura depende da marcação inicial.

Definição 3.15 (Rede Segura) *Uma rede RM (R; M₀) é definida como segura se* $M(p_i) \leq \mathbf{1}$, *para todo* $p \in P$.

Vivacidade A ausência de *deadlock* (impasse) em sistemas está fortemente ligada ao conceito de vivacidade(*Líveness).* Deadlock em uma rede de Petri é a impossiblidade do disparo de qualquer transição da rede.

Definição 3.16 (Transição Potencialmente Disparável) *Chamamos uma transição* t_i *de potencialmente disparível em uma marcaçao* M_0, *se existe uma marcaçao* $M' \in CA(R; M_0)$ *tal que* t_i *esteja habilitada para esta marcação.*

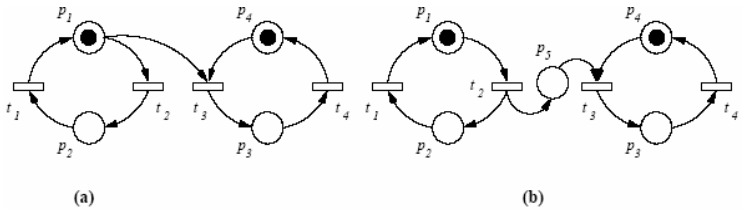


Figura 3.10:Vivacidade.

Denominamos uma transição t_i viva (*live*) em uma marcação M se t_i é potencialmente disparável para todas as marcações $M \in A(R; M_0)$.

Definição 3.17 (Rede viva) *Uma rede $RM = (R; M_0)$ é dita viva (live) se para toda $M \in CA(R; M_0)$ é possível disparar-se qualquer transição de RM através do disparo de alguma seqüência de transições.*

Reversibilidade Uma rede de Petri $RM = (R; M_0)$ é definida como reversível se for sempre possível retornar à marcação inicial M_0, isto é, M_0 é alcançável de qualquer outra marcação acessível M.

Definição 3.18 (Reversibilidade) *Uma rede $RM = (R; M_0)$ é dita reversível se para toda e qualquer $M \in CA(R; M_0)$ é possível retornar a marcação inicial da rede (M_0) através do disparo de alguma seqüência de transições.*

3.5.13 Propriedades estruturais

Tais propriedades possibilitam a análise do comportamento em função da estrutura do modelo. As propriedades estruturais são aquelas que refletem características independentes da marcação.

Desde que as redes sejam puras, a estrutura da rede pode ser representada pela matriz de incidência. Serão visto os conceitos de limitação estrutural, conservação e consistência.

Limitação Estrutural Uma rede de Petri $R = (P, T, I, O, K)$ *é* classificada como estruturalmente limitada *(structural bounded)* se é limitada para qualquer marcação inicial.

Conservação - A conservação é uma importante propriedade das redes de Petri permitindo, por exemplo, a verificação da não-destruição de recursos através da simples conservação de marcas [5]. Uma rede é dita *conservativa* se esta é conservativa para algum vetor de peso de inteiros positivos.

Definição 3.19 (Rede Conservativa) *Uma rede marcada RM = (R; M_0) é dita conservativa se existe um vetor de pesos W = (w_i, w_2, ..., w_n), tal que $\sum w_i.M_k(p_i) = \sum w_i.M_0(p_i)$, onde n = #P e w_i é um inteiro positivo, $\forall p_i \in P$ e $\forall M_k \in CA(R; M_0)$.*

Consistência Uma rede de Petri tem a propriedade de consistência se existe uma seqüência de transições habilitadas a partir de uma marcação onde todas as transições da rede são disparadas pelo menos uma vez.

Definição 3.20 (Consistência) *Seja RM = (R; M_0) uma rede marcada. RM é dita consistente se M_i [$s > M_j$ e pra toda transição t_i dispara ao menos uma vez em s* ($s(t_i)>0$ – número de disparos de t_i em *s*).

3.5.14 Análise das redes de Petri

O estudo dos métodos de análise [24] das redes de Petri são fundamentais para verificação e análise das propriedades de uma rede. Os métodos de análise das redes de Petri podem ser classificados em três grupos: análise baseada na árvore de cobertura (análise por enumeração), os métodos baseados na equação fundamental das redes de Petri (técnicas de análise estrutural) e as técnicas de redução (análises por transformação). Neste trabalho, apenas será apresentado o primeiro grupo de análise: análise baseada na árvore de cobertura.

3.5.15 Árvore de Cobertura

O método de análise ora denominado *Árvore de Cobertnra* baseia-se na construção de uma árvore que possibilite a representação de todas as possíveis marcações de uma rede.

Para uma dada rede de Petri , com uma marcação inicial, é possível obtermos diversas marcações para um grande número de transições potencialmente habilitadas. Essas marcações podem ser representadas por uma árvore, onde os nós são as marcações e os arcos as transições disparadas. Algumas propriedades, tais como limitação *(boundedness),* segurança *(safeness),* transição morta *(dead)* e alcançabilidade

de marcações, podem ser analisadas através da árvore de cobertura. A árvore de cobertura é um gráfico utilizado para representar finitamente [4]um número infinito de marcações.

Definição 3.21 (Árvore de Cobertura) *Seja RM = (R; M_0) uma rede de Petri marcada. Define-se árvore de cobertura pelo par AC = (S, A), onde S representa as marcações e A os arcos rotulados por $t_j \in T$.*

Para uma rede de Petri limitada, a árvore de cobertura é denominada *árvore de alcançabilidade,* dado que esta contém todas as possíveis marcações da rede. Neste caso, *S = CA*. Uma outra possível representação finita das marcações acessíveis pode ser obtida do grafo de cobertura [5].

Definição 3.22 (Grafo de Cobertura) *Seja RM = (R; M_0) uma rede de Petri marcada. Define-se árvore de cobertura pelo par GC = (S, A), onde S representa o conjunto de todos os nós na árvore de cobertura e A os arcos rotulados por $t_j \in T$ representando todas os possíveis disparos de transição tal que $M_i[t_k > M_j$, onde $M_i, M_j \in S$.*

Se uma rede é limitada o grafo de cobertura é também denominado de *grafo de alcançabilidade.* Embora, através desse método de análise, pode-se analisar as propriedades de limitação *(boundedness),* segurança *(safeness),* conservação e problemas de cobertura de marcações *(coverability),* não possibilita a resolução de *liveness,* alcançabilidade *(reachability)* e reversibilidade em redes não-limitadas. Este problema está relacionado à introdução do símbolo Ω que é uma informação da qual não sabemos o seu real valor.

Propriedades comportamentais e estruturais serão verificadas para as redes que forem obtidas no decorrer do trabalho. A definição utilizada a partir de agora para redes de Petri será a definição segundo um estrutura definida em matriz.

Desde a sua definição inicial as redes de Petri têm sido usada para modelagem de sistemas, como os sistemas de computação; protocolos de comunicação; sistemas

[4] Para possibilitar a representação finita das marcações, é utilizado o símbolo .

embarcados; etc. A abordagem deste trabalho utilizará em especial a classe de rede de Petri estocástica generalizada denominada *GSPN.*

3.6 O Modelo GSPN

Modelos GSPN compreendem dois tipos básicos de transições: as transições temporizadas, as quais têm retardos (*delays*) exponencialmente distribuídos associados; e as transições imediatas, que disparam com tempo zero, e têm prioridade superior as transições temporizadas. Níveis diferentes de prioridade podem ser atribuídos às transições. As prioridades podem servir para solucionar situações de confusão. Associam-se pesos às transições imediatas, a fim de solucionar situações de conflito. Outras distribuições, além da distribuição exponencial, podem ser modeladas utilizando-se o recurso de aproximação por fases de exponenciais, que será mostrada no capítulo seguinte (Capítulo 4).

Definição 3.23 (GSPN) é uma óctupla: $GSPN = (P, T, \pi, I , O, H, M_0, W)$, onde $(P, T, \pi, I , O, H, M_0)$ é a rede não temporizada PN subjacente, que compreende:

- um conjunto de locais P ;
- um conjunto de transições T;
- as funções de entrada e saída $I, O: T \rightarrow \aleph$;
- a função de arco inibidor $H: T \rightarrow \aleph$;
- uma marcação inicial M_0;
- Adicionalmente, a definição GSPN compreende a função de prioridade $\pi: T \rightarrow \aleph$; a qual associa a menor prioridade (≥ 1) às transições temporizadas e prioridades mais alta (0) para transições imediatas:

$$\pi(\mathrm{t}) = \begin{cases} \geq 1 \text{ se t é temporizada} \\ 0 \text{ se t é imediata} \end{cases}$$

Finalmente, a função *W*: T$\rightarrow \Re \cup \{0\}$, que associa um valor real não negativo com transições. *w(t)* é:

- se *t* for uma transição temporizada, então *w* será o valor do parâmetro da f.d.p. exponencial;
- se *t* for uma transição imediata, então *w* será um peso, que é usado para o cálculo das probabilidades de disparo das transições imediatas em conflitos.

Nos modelos GSPN há dois tipos de estados, chamados de estados tangíveis (*tangible*) e os estados voláteis (*vanish*). Os estados voláteis são assim denominados, porque o seu tempo de vida é igual a zero. O estado volátil é criado em decorrência da marcação dos lugares que são pré-condição de uma transição imediata. Desta forma, quando as marcas chegam a estes lugares, são instantaneamente consumidas. O tempo de permanência das marcas nestes lugares é zero. Esta é a razão de chamá-los de estados voláteis, pois são criados e instantaneamente destruídos.

O modelo GSPN faz uso da semântica *interleaving* de ações. Assume-se que as transições são disparadas uma a uma, mesmo se o estado compreenda transições imediatas não conflitantes. A análise de um modelo GSPN requer a solução de um sistema de equações igual ao número de marcações tangíveis. O gerador infinitesimal *Q* da CTMC associado ao modelo GSPN, é derivado de uma redução de um grafo de alcançabilidade, rotulado com as taxas ou pesos das transições imediatas.

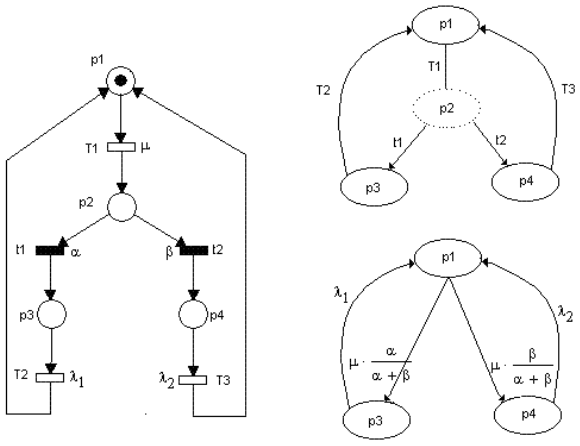


Figura 3.11:Geração do gráfico de alcançabilidade GSPN.

A Figura 3.11 apresenta um exemplo de geração do grafo de alcançabilidade de uma GSPN. Na GSPN da Figura 3.11 a esquerda existe um conflito entre as duas transições imediatas. Mostra-se o grafo de alcançabilidade na parte superior direita. As linhas pontilhadas do estado p_2 indicam que o estado é volátil. De fato, quando a transição T_1 dispara o sistema entra estado p_2, habilitando as duas transições imediatas, t_1 e t_2, gerando o estado p_2. Há uma mudança imediata (tempo zero) para o estado p_3 ou p_4, através do disparo da transição t_1 ou t_2, com probabilidades $\frac{\alpha}{\alpha+\beta}$ e $\frac{\beta}{\alpha+\beta}$ respectivamente. O grafo de alcançabilidade tangível na parte direita inferior da Figura 3.11 é obtido eliminando-se o estado volátil p_2. A taxa na qual o sistema se move do estado p_1 para p_3 ou p_4 é obtida pelo produto da taxa λ da transição do estado p_1 para o estado volátil p_2, com a probabilidade de ir do estado p_2 para o estado p_3 ou p_4.

Desta forma, [24] chegou ao seguinte teorema: "qualquer rede de Petri estocástica marcada, com um número finito de lugares e transições, é isomórfica a uma cadeia de Markov".

De acordo com [43] e [45], obtêm-se o isomorfismo da GSPN com uma cadeia de Markov a partir do grafo de alcançabilidade reduzido, que é dado através da eliminação dos estados voláteis (não tangíveis) e com as taxas de transição entre os estados rotulando os arcos . A avaliação se dá através da cadeia de Markov embutida na GSPN. Podem ser realizadas avaliações em estado estacionário, estado transiente ou através de simulação.

A solução em estado estacionário e transiente da GSPN é obtida pela solução da CTMC subjacente, o qual foi mostrada na seção de cadeias de Markov de tempo contínuo.

As redes de Petri, portanto, serão o método formal utilizado para modelar, nos próximos capítulos, sistemas embarcados com algumas particularidades. Os modelos obtidos em redes de Petri serão passíveis de verificações, análises e sumulações.

3.7 Resumo

Iniciou-se o capítulo apresentando-se os conceitos de sistemas e modelos, bem como a representação de sistemas através de modelos. Mostrou-se que um modelo pode representar um sistema sob diversos ângulos. Em seguida, foi mostrado os processos estocásticos como modelos matemáticos úteis na descrição de fenômenos probabilísticos. Dando continuidade, introduziu-se as cadeias de Markov, mas especificamente, as cadeias de tempo contínuo (CTMC). Revisou-se o processo de avaliação em estado transiente e estacionário de uma CTMC. Por último, apresentou-se rede de Petri, sua classificação, propriedades comportamentais e estruturais. Encerrou-se o capítulo descrevendo o modelo GSPN que é uma extensão do modelo de redes de Petri.

Capítulo 4 - Modelagem e estimativa de potência

Neste capítulo é apresentada a abordagem para o cálculo da estimativa de potência dissipada em sistemas embarcados. O método proposto adota rede de Petri estocástica generalizada (GSPN) e utiliza o recurso da aproximação por fases de exponencial para contemplar ações com tempos determinísticos. Inicialmente é definido o modelo chamado de Power-GSPN. Em seguida, é mostrada a técnica de aproximação por fase de exponenciais e técnicas de medições de consumo de energia. Será mostrada a análise das redes de Petri em estado transiente e estacionário, e a métricas pelas quais serão obtidas as estimativas de consumo de energia, potência e desempenho do sistema embarcado.

4.1 Introdução

Em projetos de sistemas embarcados, é comum, após a especificação do sistema, a criação de um modelo. Estes modelos podem representar diversas perspectivas do sistema. Utilizando alguns destes modelos, estimativas de área, potência dissipada, custo e desempenho podem ser obtidos. Essas estimativas servem como parâmetros para análise, de forma a auxiliar as implementações que satisfaçam às restrições do projeto.

Para o cálculo das métricas é necessária a definição de um conjunto de métodos e a elaboração de ferramentas que implementem estes métodos. Os referidos algoritmos devem considerar o modelo que representa as especificações do sistema, as restrições

de projeto e a arquitetura de implementação. O resultado da avaliação obtido nesta fase é de essencial importância, pois possibilita uma exploração rápida do espaço de projeto pela avaliação de diferentes alternativas de implementação e verificação de violações de restrições, sem a necessidade da implementação dos sistemas, tornando mais ágil todo o processo de desenvolvimento.

Neste capítulo será apresentada uma metodologia para modelar aspectos temporais e de consumo de energia em sistemas embarcados, através de uma classe de redes de Petri denominada *Rede de Petri Estocástica Generalizada (GSPN).*

A Figura 4.1 descreve os passos principais da metodologia proposta. O requisito inicial é uma especificação de um sistema, a partir da qual cria-se um modelo GSPN. Após a construção do modelo GSPN, deve-se atribuir aos seus elementos as informações de consumo de energia e tempo. Avaliando-se o modelo GSPN para o estado estacionário e/ou transiente, estimativas de potência e desempenho são obtidas.

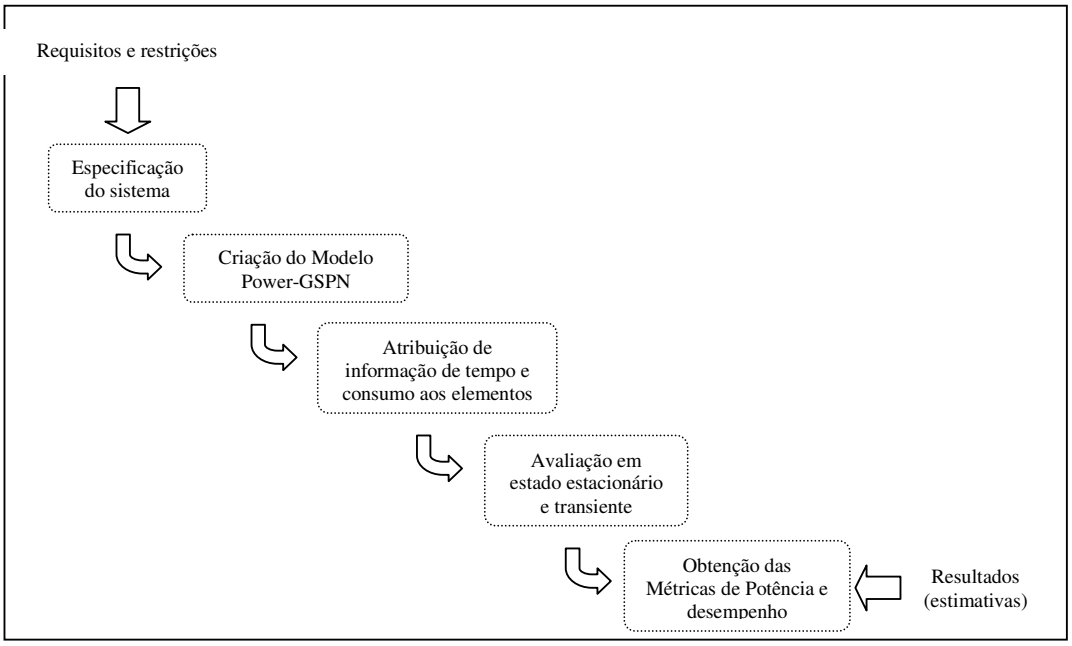


Figura 4.1:Diagrama da metodologia proposta.

Estimativas no projeto de sistemas são essenciais por duas razões: primeiro, por permitir ao projetista avaliar a qualidade do projeto, comparando as métricas obtidas

com as restrições especificadas para aquela métrica. Por exemplo, se a potência dissipada pelo sistema for maior que a prevista ou permitida, pode ser necessário alterar o projeto do sistema para satisfazer tal restrição. Segundo, estimativas permitem ao projetista de sistemas explorar alternativas de projeto, provendo uma rápida retro-alimentação para qualquer decisão de projeto. Deste modo, o projetista pode explorar alternativas antes de fazer a sua implementação final.

4.2 Power-GSPN

A metodologia proposta para análise e estimativa de potência dissipada em sistemas embarcados é baseada em uma extensão da GSPN denominada POWER-GSPN. Neste modelo serão adicionadas informações referentes ao consumo de energia às ações, que são elementos que promovem mudança de estado do sistema embarcado, além das informações já contempladas pelo modelo GSPN. À medida que se consegua descrever ou caracterizar bem as ações do sistema, serão obtidos resultados com maior precisão.

A fim de utilizar o formalismo estocástico intrínseco às GSPNs, foram adicionadas ao modelo GSPN funções que atribuem aos elementos, informações relativas ao consumo de energia.

Definição 4.3.1. Power-GSPN: Seja $N = (P, T, \pi, I , O, H, M_0, W)$ uma GSPN. **Power-GSPN =**(N, E_d, E_e) é uma GSPN estendida, com duas funções que atribuem consumo de energia aos elementos da rede.

- $E_d : T \rightarrow \Re^+ \cup \{0\}$ atribui o consumo de energia associada a uma execução de uma ação a uma transição - chamada de consumo de energia dinâmico;
- $E_e : P \rightarrow \Re^+ \cup \{0\}$ representa o consumo de energia pela permanência nos estados (representado pela marcação nos lugares) - chamada de consumo de energia estático.

As métricas de estimativas de potência que serão computadas neste trabalho consideram apenas a dissipação de potências dinâmicas, uma vez que a tecnologia atual

utilizada nos sistemas embarcados tem dissipação de potência estática extremamente baixa quando comparada à potência dinâmica, e pode ser desprezada.

Dado que o modelo se baseia em GSPN, a distribuição de probabilidade associada às transições temporizadas é a distribuição exponencial. Para representar outras distribuições, como por exemplo, distribuições determinísticas, será utilizada a técnica de aproximação por fases de exponenciais. Na próxima seção será mostrada uma aproximação por fases de exponenciais para modelar uma distribuição determinística.

4.3 Modelagem de ações determinísticas através de aproximação por fases de exponenciais.

Em GSPN as transições representam apenas atividades cujas informações de desempenho sejam exponencialmente distribuídas ou ações imediatas. Contudo, é possível representar atividades com outras distribuições através de aproximação por fases de exponenciais [18], [59].

Esta técnica flexibiliza o conjunto de distribuições que são suportados pelo método proposto. Uma das distribuições não-exponenciais muito comumente associadas às atividades realizadas por sistemas embarcados é a distribuição com tempo determinístico.

Para modelar as distribuições com tempos determinísticos é utilizada uma sub-rede que associa um conjunto *N* de transições com tempos (fases) iguais em série. Esta sub-rede representa uma distribuição de Erlang com *N* fases. A Figura 4.2 apresenta uma sub-rede com duas fases.

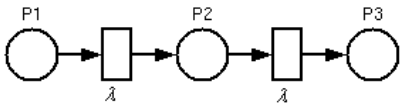


Figura 4.2:Sub-rede em série, chamado de aproximação por fases.

Onde: *N* – é o numero de fases da aproximação e λ taxa de disparo de cada fase.

Dado que estas fases serão representadas por transições em série, tem-se *N* lugares ligados a *N* transições exponenciais. Na Figura 4.2, *N,* é igual a dois (*N*=2). Quanto maior for o número de fases *N,* maior será a precisão da aproximação. Por outro lado, quanto maior for o número de fases, maior será o número de estados do sistema e mais complexa será sua avaliação.

Aldous e Shepp [2] provaram que a distribuição de Erlang é a distribuição menos variável do tipo-fase para um dado número de fases, sendo portanto a aproximação por fases de exponenciais mais apropriada para a aproximar distribuição determinística.

Dada uma distribuição determinística com média μ_D em uma distribuição de Erlang com um pré-especificado número *N* de fases, deve-se casar o primeiro momento e obtém-se o valor da taxa de disparo da transição λ.

Para aproximar-se uma distribuição determinística de tempo igual a μ_D (no contexto estudado: tempo) através de uma distribuição de Erlang com *N* fases, dividi-se o valor μ_D pelo número de fases (*N).* Desta forma, tem-se que o valor médio de cada fase igual a $\frac{\mu_d}{N}$. Sabe-se, contudo, que esta aproximação tende para valores mais próximos à medida que o número de fases aumenta.

Neste trabalho, considera-se uma distribuição de Erlang com 10 fases (N = 10). Este número de fases, normalmente, é suficiente para uma boa aproximação. A rede da Figura 4.3 representa esta aproximação.

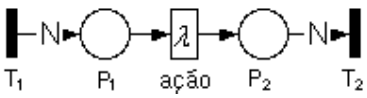


Figura 4.3:Modelo (Erlang) que aproxima um tempo determinístico.

O retardo (*Delay*) da transição exponencial associada será calculado pela fórmula:

$$D = \frac{1}{\lambda}$$

Onde:

$$\lambda = \frac{N}{\mu_D}$$

Substituindo-se, tem-se:

$$D = \frac{\mu_D}{N}$$

Onde: D é o retardo associado a ação determinística e N é o número de fases.

Note que quando a transição imediata t_1 for disparada, será colocado N marcas no lugar P_1, habilitando a transição exponencial *'ação'*. Em torno do tempo $\frac{1}{\lambda}$, a transição *'ação'* irá disparar, retirando uma marca do lugar P_1 e colocando no lugar P_2. Este processo se repetirá até que todas as marcas do lugar P_1 sejam consumidas, isto é, N vezes. Depois que todas as marcas forem transferidas de P_1 para P_2, é habilitada a transição t_2, finalizando o processo.

Desta forma, para representar uma ação de tempo determinístico, será utilizado uma *aproximação por fase* de ordem N. Esta distribuição determinística de valor μ_D é dividida por N. Quando a transição tiver disparado as N vezes, o retardo resultante é aproximadamente μ_D, o que nos garante uma aproximação bem razoável. Lembrando que quanto maior for o número de fases N, melhor será a aproximação, porém maior será o número de estados do modelo.

4.4 Geração do Modelo

Esta seção apresenta um conjunto de etapas e passos adotados para que, a partir de uma especificação de um sistema, possa ser criado um modelo Power-GSPN. Este modelo é utilizado para obtenção de estimativas de potência e desempenho.

Posteriormente, analisam-se os recursos e condições afetadas pelas execuções das ações (representados por lugares). Relação de pré e pós-condições das transições (ações e eventos) devem, portanto, ser estabelecidas. Estas relações associam lugares às transições.

Um conjunto de atributos ou características é mapeado para cada ação do sistema. Tais atributos são informações necessárias para a representação temporal e de consumo de energia do modelo. As informações devem ser rotuladas às transições. A seguir, descrevem-se as propriedades das transições.

Considerando uma especificação de um sistema para o qual estimativas devem ser computadas, um modelo Power-GSPN deve ser criado seguindo as seguintes associações:

- para cada ação ou operação do sistema (dependendo da granularidade adotada), tem-se uma transição que irá representá-la no modelo;
- os lugares representam recursos e estados locais associados (que habilitam ou são afetados) ao disparo das transições (que representam as execuções das ações);
- arcos valorados estabelecem as relações das pré-condições e pós-condição entre transições e lugares;
- as marcas representam o estado atual do sistema.

Desta forma, associadas a uma ação do sistema (representada por uma transição) tem-se um ou mais lugares de entrada, representando as pré-condições e um ou mais lugares de saída, representando as pós-condições. Os lugares de entradas serão ligados às transições através dos arcos conforme Figura 4.4 e Figura 4.5:

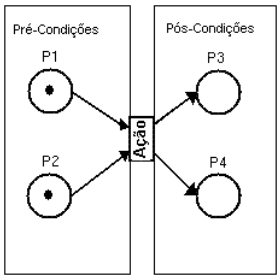


Figura 4.4:(estado) antes do disparo da transição Ação.

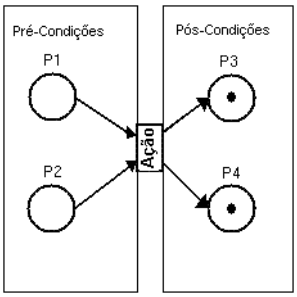


Figura 4.5:Marcação (estado) após o disparo da transição Ação.

Cada ação do sistema terá um conjunto de atributos, comentadas anteriormente, mas nem todas serão obrigatoriamente informadas. São elas:

- Nome
- Taxa de execução (ou o tempo médio)
- Prioridade
- Peso
- Consumo de energia

Nome – toda ação deve ter um nome único que a identifique no modelo, de preferência um nome que tenha uma associação semântica a fim de facilitar o entendimento.

Tempo de execução – para toda ação, deve-se especificar seu tempo de execução, que pode inclusive ser zero, caso em que serão utilizadas transições imediatas ao invés de transições temporizadas.

Prioridade – deverá ser atribuída uma prioridade às transições. A princípio, as transições imediatas têm prioridade zero (maior prioridade), enquanto as transições temporizadas têm prioridade um. Contudo, situações de confusão devem ser solucionadas atribuindo-se diferentes níveis de prioridades às transições pertencentes à confusão (*Extended conflict set).*

Peso das transições – ao conjunto de transições imediatas em conflito (mais de uma transição imediata habilitada para uma mesma marcação da rede), deve-se atribuir um peso a cada uma destas transições, onde o valor destes pesos serão normalizados e convertidos em probabilidade de execução de transição.

Consumo de energia – associa-se a cada transição temporizada um valor de energia que representa o respectivo consumo de energia associado a uma execução da ação representada pela transição.

Definido-se as transições que representarão as ações, com seus respectivos atributos, o próximo passo para criação do modelo é o mapeamento entre as transições e os lugares através dos arcos. O mapeamento define as pré-condições das transições, isto é, como cada transição estará habilitada, assim como, as pós-condições, isto é, os lugares que serão marcados após o disparo da transição. Concluída esta etapa, deve-se encontrar todas as transições imediatas em conflito (mais de uma transição imediata habilitada numa mesma marcação com a mesma prioridade) e atribuir um peso a cada transição que faça parte do conflito. Estes pesos definirão as probabilidades de disparo das transições imediatas conflitantes.

No modelo Power-GSPN, assim como nas GSPN, quando temos dois conflitos locais habilitados em uma mesma marcação, a normalização dos pesos das transições imediatas conflitantes será realizada globalmente e não localmente. Isto significa dizer que se dois conjuntos de transições conflitantes estão habilitados em uma mesma marcação (por exemplo, cada um com duas transições) a normalização dos pesos das transições se dará entre todas as transições participantes dos conjuntos de transições conflitantes.

Situações de confusão podem ser resolvidas alterando os valores das prioridades das transições, de forma que não influenciem na fidelidade do modelo. A resolução de confusões deve ser feita de maneira cuidadosa, analisando-se caso a caso. Ressalta-se que, para que a avaliação seja realizada através dos métodos de análise (estacionário e transiente) é fundamental que os modelos sejam livres de confusão. Este não é o caso quando se avalia através de simulação

4.4.1 Representação de sistemas com mais de uma tarefa

A maioria das aplicações embarcadas requerem dispositivos periféricos, tais como conversores A/D e D/A, portas de comunicação serial, temporizadores, entre outros. Estes dispositivos executam tarefas específicas e em determinados momentos precisam da atenção do processador para efetuar transferências de dados. Por exemplo, numa porta serial os dados chegam bit-a-bit e são transformados em bytes pelo controlador serial. Só após receber todos os bits, da palavra serial, o controlador requer a atenção do processador para a transferência do byte. A maneira pela qual os periféricos solicitam a atenção do processador denomina-se ***Interrupção.***

Mais genericamente, a necessidade de interrupção da CPU cai numa das seguintes categorias:

- necessidade de transferências de dados entre os periféricos e o processador;
- sinais de entradas usados com propósito de marcar tempo;
- situações de emergências.

Em todas as categorias acima é necessária uma sinalização para o processador.

4.4.2 Interrupções e compartilhamento de recursos

É comum em sistemas embarcados a utilização de interrupções do fluxo de execução de uma tarefa para realizar outra tarefa de maior prioridade. Todas as solicitações de interrupções têm um nível de prioridade, assim como as tarefas que estão sendo

executadas. Desta forma, quando uma tarefa está sendo executada e o processador recebe uma solicitação de interrupção, a solicitação será aceita se o nível de prioridade da interrupção solicitada for maior que a prioridade da tarefa que está em execução.

Para representar o processo de interrupção em modelo Power-GSPN, foi necessário modelar o processador como um recurso, as tarefas que irão solicitar este recurso para sua execução com suas respectivas prioridades.

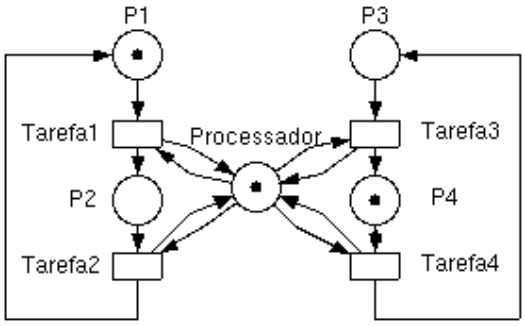


Figura 4.6:Processador compartilhado com quatro tarefas.

A Figura 4.6 mostra quatro tarefas distintas que compartilham o mesmo recurso, no caso, o processador. Note que quando uma ou mais tarefas solicitam o processador, as tarefas um e quatro, o modelo possibilita a execução primeiro da tarefa de maior prioridade. Após o término da execução, disponibiliza-se o recurso novamente para que a tarefa com menor prioridade seja executada. Ressalta-se, contudo que no modelo GSPN os tempos são representados por retardos (*delay*) e não duração. Portanto, as marcas permanecem nos lugares até que ocorra o disparo da transição. O disparo das transição em si não consome tempo algum.

4.5 Temporizadores como geradores de interrupção

Temporizadores são circuitos autônomos, periféricos ao processador, que podem ter função de gerar solicitações de interrupção em intervalos de tempos definidos.

Para representar um temporizador em um modelo Power-GSPN, faz-se necessário definir sua base de tempo. As temporizações serão sempre múltiplas desta base.

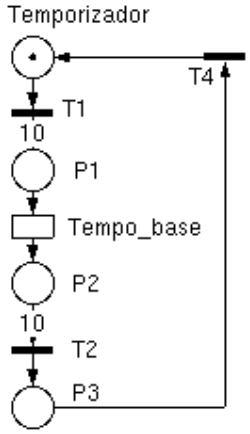


Figura 4.7:Modelo do Temporizador.

A Figura 4.7. apresenta o modelo do temporizador. Dado que o tempo associado ao temporizador é determinístico, adotou-se uma distribuição de Erlang com dez (10) fases para representar este valor. Com uma marca no lugar *Temporizador* obtêm-se a menor temporização possível, denominada de *tempo base* (Δt_{base}). Para outras temporizações deve-se calcular o número de marcas "*m*" a ser inseridas no lugar *Temporizador* de acordo com a fórmula seguinte:

$$m(temporizador) = \frac{\Delta t}{\Delta t_{base}}$$

Onde: *m(Temporizador)* - é o número de marcas a ser inserida no lugar *Temporizador*;

Δt - é o intervalo de tempo que se deseja temporizar;

Δt_{base} - é o tempo base do temporizador, ou seja, menor temporização possível.

4.6 Definição das Métricas

Esta seção define o conjunto de métricas utilizadas para avaliar o consumo de energia e o desempenho dos sistemas representados através do modelo proposto. Contudo, convém definir formalmente alguns conceitos necessários para o entendimento e definições das respectivas métricas.

O *Subconjunto de estados que habilita uma transição* ($S(t_i)$) representa o conjunto de estados onde a transição $t_i \in T$ está habilitada.

Definição 4.6.1. *Subconjunto de estados de transição habilitada - $S(t_i)$* - Seja $N=(P, T, \pi, I, O, H, M_0, W, E_d, E_e)$ uma Power-GSPN limitada, viva e reversível e S o seu conjunto de estados alcançáveis. Define-se $S(t_i) = \{S_k \mid m_k(p_j) \geq I(p_j, t_i), \forall p_j \in P\}$ como todos os estados onde a transição $t_i \in T$ está habilitada.

A *Probabilidade de Habilitação de Transição* ($PHT(t_i)$) representa a probabilidade de uma transição $t_i \in T$ estar habilitada, considerando o conjunto de estados S.

Definição 4.6.2. *Probabilidade de Habilitação de Transição* - $PHT(t_i): T \rightarrow \Re^+ \cup \{0\}$ - Seja t_i uma transição pertencente a T, s_k um estado pertencente ao subconjunto de estado $S(t_i)$ e π_k a probabilidade estacionária do estado s_k. Define-se $PHT(t_i) = \sum_{s_k \in S(t_i)} \pi_k$ como a probabilidade da transição $t_i \in T$ estar habilitada em um instante qualquer.

O *Taxa de execução de uma transição* ($Tx(t_i)$) representa o número de execução da transição t_i por intervalo de tempo (segundo). É obtida pelo produto da probabilidade de habilitação da transição (PHT), pela taxa de disparo da transição (λ).

Definição 4.6.3. *Taxa de execução de uma transição - $Tx(t_i)$*: $T \rightarrow \Re^+ \cup \{0\}$ - Seja t_i uma transição pertencente a T, *e* $PHT(t_i)$ a probabilidade de habilitação da transição t_i. Define-se $Tx(t_i) = PHT(t_i) \times \lambda(t_i)$ como a métrica de desempenho que está associado ao número de execução da transição $t_i \in T$ por unidade de tempo (segundo), dada em execuções por segundo (s^{-1}).

A *Potência dissipada de uma transição* ($P(t_i)$) representa a potência dissipada pelas execuções da ação associada à transição t_i. É obtida pelo produto da taxa de execução ($Tx(t_i)$), pelo consumo de energia da transição ($E(t_i)$).

Definição 4.6.4. *Potência dissipada de uma transição - $P(t_i)$*: $T \rightarrow \Re^+ \cup \{0\}$ - Seja t_i uma transição pertencente a T, *e* $Tx(t_i)$ a taxa de execução da transição t_i. Define-se

$P(t_i) = Tx(t_i) \times E(t_i)$ como a potência dissipada pelas execuções da ação associada à transição t_i, dada em *Watt*(W).

A *Potência dissipada pelo sistema (PS)* representa a somatório das potências dissipada nas transições $t_i \in T$ do sistema representado.

Definição 4.6.5. *Potência dissipada pelo Sistema - PS*: $T \rightarrow \Re^+ \cup \{0\}$ - Seja t_i uma transição pertencente a *T*, *e* $P(t_i)$ como a potência dissipada pelas execuções da transição t_i. Define-se $PS = \sum_{t_i \in T} P(t_i)$ como a potência total dissipada pelo sistema, dada em *Watt*(W)

A *Energia Consumida pelo Sistema* (*ECS*) representa consumo de energia do sistema represetado, ao longo do tempo. A *ECS* é calculada pela integração da potência dissipada pelo sistema no tempo.

Vale ressaltar que o valor da *ECS* calculado neste trabalho se dá através da análise estacionária. Isto porque o consumo relacionado ao estado transiente é, geralmente, muito pequeno em relação ao consumo em estado estacionário. Porém, como já foi mostrado, o sistema tem dois comportamentos distintos. O primeiro deles é quando o sistema é iniciado, chamado de estado transiente. O estado transiente, geralmente, tem um intervalo de tempo muito curto. A duração do estado transiente vai até quando o sistema alcança a estabilidade, isto é, quando seus parâmetros começam a se tornar constante, como por exemplo, a potência dissipada assuma um valor que não se altere significativamente em relação ao tempo.

Para uma maior precisão do calculo do *ECS* deve-se considerar a energia consumida em estado transiente.

Definição 4.6.6. *Energia Consumida pelo Sistema - ECS* : $T \rightarrow \Re^+ \cup \{0\}$ - Seja $N=(P, T, \pi, I , O, H, M_0, W, E_d, E_e)$ uma Power-GSPN limitada, viva e reversível e *PS* a potência dissipada pelo sistema representado. Define-se $ECS = \int_{t_i}^{t_f} PS \times dt$ como a

energia consumida pelo sistema no intervalo de tempo $\Delta t = t_f - t_i$, que é calculada pela integração da potência do sistema no intervalo de tempo Δt, dada em *Joule*(J)

A *Energia Acumulada* (*EA*) representa a quantidade de energia acumulada que será utilizada para alimentar o sistema.

Definição 4.6.7. *Energia Acumulada - EA* : $T \rightarrow \Re^+ \cup \{0\}$ - Seja *V* a tensão nominal da fonte alimentadora dada em *Volts (V), I* a corrente máxima nominal fornecida pela fonte alimentadora, dada em *Amper* (A), e $\Delta t = t_f - t_i$ o intervalo de tempo de fornecimento da corrente, pela fonte alimentadora, sem depreciação da Tensão. Define-se $EA = \int_{t_i}^{t_f} (V \times I) dt$ como a energia acumulada que é disponibilizada para alimentação do sistema, dada em *Joule*(J).

A *Autonomia do sistema (A)* representa o tempo que uma fonte de energia acumulada consegue fornecer energia, a níveis de tensão (*V*) e corrente (*I*) compatíveis com o a necessidade do sistema. Ao longo do tempo a energia armazenada na fonte acumuladora decairá. Após um determinado intervalo de tempo (Δt) de fornecimento de energia, a fonte acumuladora não terá condições de manter o nível de tensão e corrente necessário para manter o sistema em funcionamento. A partir do momento em que a fonte não consegue suprir a necessidade de fornecimento de energia adequada ao perfeito funcionamento do sistema, este entrará em estado de falha.

Definição 4.6.7. *Autonomia do sistema – A* - $T \rightarrow \Re^+ \cup \{0\}$ - Seja *PS* a potência dissipada pelo sistema e *EA* a energia acumulada pela fonte alimentadora. Define-se $A = \frac{EA}{PS}$ como autonomia da fonte alimentadora considerando aquele sistema. Sendo o valor dado em unidade de tempo(*s*).

Índice de Desempenho do Sistema - (*IDS*) é uma medida que representa o desempenho do sistema sob um determinado aspecto. Em alguns casos, o índice considerado será a taxa de execução geral do sistema.

As métricas definidas podem ser calculadas e auxiliar o processo de tomada de decisão no projeto de sistemas embarcados de maneira que apenas alternativas viáveis, considerando os aspectos de consumo de energia e desempenho, venham ser consideradas para a implementação.

4.7 Metodologia de medição do consumo

Para que o modelo estacionário proposto venha ser avaliado de maneira fidedigna é necessário atribuir valores de consumo de energia, associado às ações, às transições que as representam no modelo.

Em alguns caso os valores de consumo das ações podem ser informados. Contudo isto nem sempre é possível. Existem diversas maneiras de se obter o consumo de energia de uma atividade (operação, procedimento, tarefa, etc.). A forma mais direta (quando possível) é consultar a folha de dados (*data-sheets*) e utilizá-la como referência para obtenção dos valores de consumo de energia das ações. É comum encontrar informações de energia associadas às operações lógicas, aritméticas etc. Para atividades mais complexas e para as quais não se dispõe de *data-sheets* está não é uma alternativa viável.

Outra alternativa para o levantamento do consumo de energia, baseia-se na metodologia apresentada em [64],[66]. Esta metodologia consiste em isolar a ação a ser medida e executá-la em um laço infinito, isto é, apenas esta ação será executada infinitas vezes. Entre cada execução, insere-se operações de saída que provocam pulsos visíveis nos terminais do processador, que servem como indicadores de início e fim da execução. Os sinais inseridos são utilizados como marcadores de sincronismo que possibilitam que um intervalo de observação seja identificado por um osciloscópio ou analisador de sinais. Este intervalo de tempo compreende o tempo de execução da ação sob observação. Enquanto um canal do osciloscópio monitora os marcadores de intervalo de medida, o outro canal monitora a tensão sobre um resistor posto em série com o sistema embarcado, conforme a Figura 4.8. Desta forma, os padrões de consumo da ação em observação, serão registrados como flutuação da tensão sobre o resistor. Através de um protocolo de controle, suportado pelo osciloscópio através de uma porta

serial RS232C, um aplicativo realiza a aquisição da tensão sobre o resistor. Estas informações são armazenadas em um arquivo onde é possível verificar os pulsos que sinalizam o início e término da ação. Todas as medidas de tensão entre os sinais de início e término da ação são selecionadas e analisadas, obtendo assim o valor médio da queda de tensão em cima do resistor.

$$\Delta V_{r(médio)} = \frac{\sum_{t=t_i, t_i \in \Delta t}^{t_f} \Delta V_r(t)}{NA}$$

Onde:

$\Delta V_{r(médio)}$ – valor médio da variação de tensão sobre a resistência;

ΔV_{r} – valor pontual da variação de tensão sobre a resistência de uma amostragem;

NA – número de amostragem da variação de tensão sobre a resistência, em um intervalo de tempo Δt. onde $NA = \frac{\Delta t}{FA}$.

t_i – tempo de início da execução da ação;

t_f – tempo de término da execução da ação;

Δt – Intervalo de tempo entre o início e término da execução da ação, delimitado pelos pulsos de início e término;

FA – freqüência de amostragem da variação de tensão sobre a resistência.

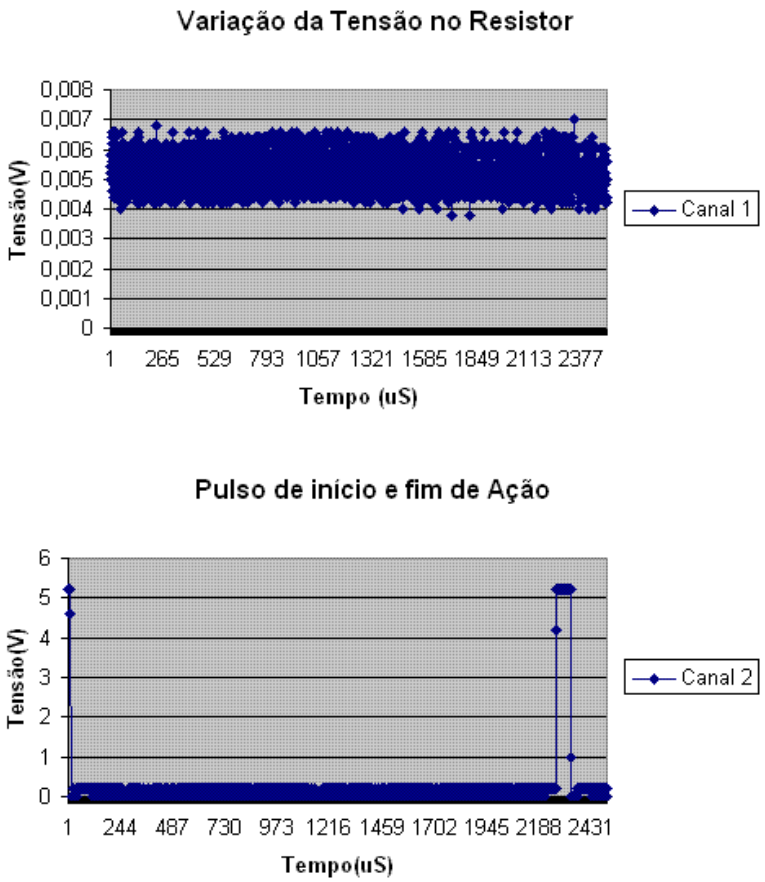


Figura 4.8: a -Gráfico da variação de tensão na resistência ; b - pulsos de sincronismo.

A Figura 4.8.a mostra a variação da tensão sobre o resistor colocado em série com o sistema em análise. A Figura 4.8.b mostra os pulsos de sincronismo antes e após a execução da ação.

Dado que o valor da resistência do resistor é conhecido, obtém-se, facilmente, a corrente que flui para o sistema. Este valor é calculado pela fórmula:

$$i = \frac{\Delta V_{r(médio)}}{r} \ .$$

Onde:

i – valor da corrente que flui pelo resistor, expressa em Amper(A);

$\Delta V_{r(médio)}$ – valor médio da variação de tensão sobre o resistor, expressa em Volt(V);

r – valor da resistência do resistor em série com o sistema embarcado, expressa em OHM(Ω) .

Obtendo-se o valor da corrente i e da tensão de alimentação V_{fonte} do sistema, obtém-se o valor da potência $P_{(ação)}$ dissipada pela execução de uma ação. Este valor e dado por:

$$P_{(ação)} = V_{\text{fonte}} \times i \, .$$

Onde:

V_{fonte} – Valor de tensão de alimentação;

$P_{(ação)}$ – Valor da potência dissipada, expressa em Watt(W), associada à execução da ação.

Tendo-se calculado o valor da potência dissipada $P_{(ação)}$, determina-se o consumo de Energia $E_{(ação)}$ da ação observada através da equação:

$$E_{(ação)} = P_{(ação)} \times \Delta t \, .$$

Onde:

$E_{(ação)}$ – Valor de consumo de energia expressa em Joule (J) para uma execução da ação.

4.8 Um pequeno exemplo didático

Para demonstrar a aplicação do modelo proposto, um pequeno exemplo é apresentado (ver Figura 4.9.). Este modelo contempla ações concorrentes e seqüenciais com distribuição determinística e exponencial, e uma situação de escolha.

O modelo tem cinco tarefas, que têm seus atributos definidos conforme Tabela 4.1. Note que as informações temporais das ações determinísticas foram informadas como retardos. As informações temporais das ações com distribuição exponencial foram apresentadas como taxas (λ). As prioridades são irrelevantes para este modelo. Quanto aos pesos associados às transições, apenas os pesos das transições *tarefas 3* e *tarefa 4* foram informados, pois apenas estas duas tarefas entram em conflito em uma dada marcação da rede.

Tabela 4.1:Atributos das tarefas do exemplo.

Atributos	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
Taxa	0,2		1,5		3,1
Tempo(s)		0,2		3	
Prioridade	1	1	1	1	1
Peso	0	0	6	4	0
Consumo de Energia (mJ)	22	30	12	26	18

Na descrição das operações deste sistema, tem-se o início dado pela execução da ação *Tarefa 1.* Há uma concorrência entre a *Tarefa 2* e a *Tarefa 3* ou *Tarefa 4*. As *Tarefa 3* e *Tarefa 4* estão em conflito. A escolha dentre a execução das *Tarefa 3* e *Tarefa 4* é solucionado probabilisticamente através da atribuição dos pesos, conforme a Tabela 4.1. Estes pesos são usados para o cálculo da probabilidade de execução para o conjunto das ações conflitantes. A ação *Tarefa 5* é executada após a execução da *Tarefa 2* e da *Tarefa 3* ou *Tarefa 4*. Tem-se, após a execução da ação *Tarefa 5,* uma nova execução da ação *Tarefa 1,* retornando, portanto, ao estado inicial.

As ações com comportamento temporal determinístico são aproximadas por fases de exponenciais no modelo Power-GSPN.

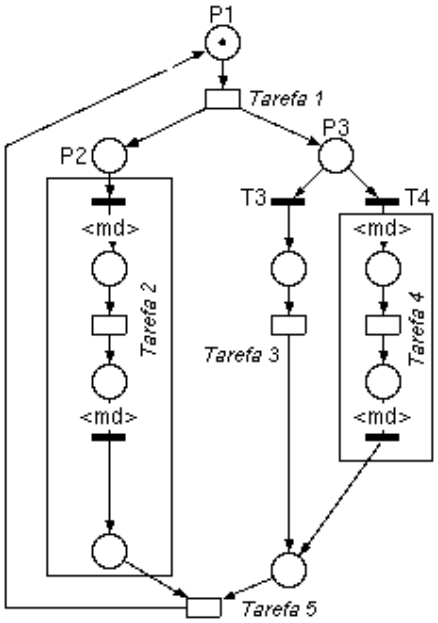


Figura 4.9:Representação em Power-GSPN do sistema exemplo.

Após a atribuição das informações básicas de tempo e consumo de energia aos elementos, encontra-se uma solução em estado estacionário (utilizando a ferramenta *TimeNET 3.0)*[65]. A ferramenta TimeNET 3.0 dá suporte a realização de experimentos, variando-se um parâmetro por vez.

Um experimento foi realizado através da variação do número de fases *N*. O objetivo deste experimento foi observar a variação da precisão das estimativas computadas de acordo com a variação do número de fase (*N*) da aproximação. Sabe-se que quanto maior o número de fases(*N*) mais aproximado será o modelo da distribuição determinística. A partir de um determinado número de fases *N*, a precisão praticamente não se altera. O resultado é mostrado na Figura 4.10.

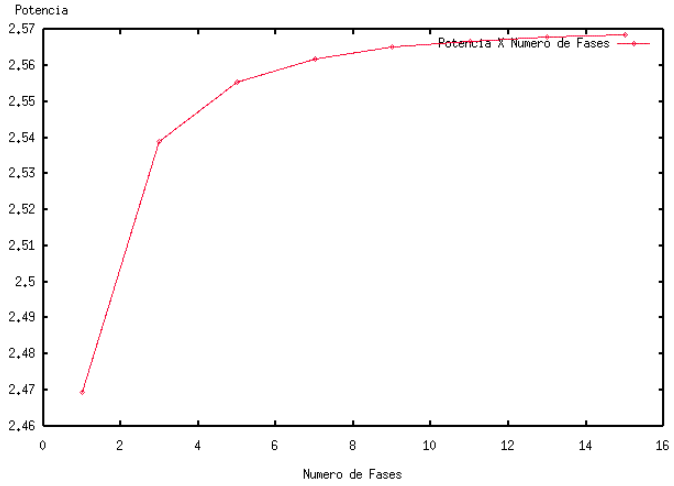


Figura 4.10:Precisão da estimativa X N° de Fases da aproximação determinística.

Para este exemplo, a Figura 4.10 mostra que o valor da estimativa de potência pouco varia em relação ao aumento do número de fases. Observa-se um comportamento assintético, tendendo para uma potência de 2,57mW. Considerando-se o número de fases igual a dez (*N=10),* tem-se um valor muito próximo daquele (2,57mW). Portanto, constata-se que, para o número de fases além de dez (*N=10),* não melhora significativamente a precisão dos resultados.

A Tabela 4.2 apresenta as estimativas obtidas para as métricas definidas. Os valores apresentados na Tabela 4.2 foram computados considerando-se o número de fases *N* igual a 10 e a probabilidade de execução da *Tarefa 3* e *Tarefa 4* igual a 0,5, respectivamente.

Tabela 4.2:Atributos e estimativas das tarefas do exemplo .

	Tarefa 1	**Tarefa 2**	**Tarefa 3**	**Tarefa 4**	**Tarefa 5**
Taxa	0,2		1,5		3,1
Tempo(s)		0,2		3	
prioridade	1	0	0	1	1
Peso	0	0	6	4	0
Consumo de Energia(J)	22	30	12	26	18
PHT(t_i)	**0,23**	**0,46**	**0,0863**	**0,1726**	**0,2877**
Tx(t_i)	**0,115**	**0,115**	**0,0575**	**0,0575**	**0,115**
P(ti)	**2,53**	**3,45**	**0,69**	**1,49**	**2,07**

A estimativa de potência total para o sistema modelado é:

$PS = \sum_{t_i \in T} P(t_i)$, PS = 10,242 mW.

Este sistema será alimentado por fonte química acumuladora cuja especificação é:

- Tensão de 1,5V
- Capacidade 3000mAh

Um arranjo em série de quatro (4) destes acumuladores provê uma tensão resultante de *6V*. Para calcular a autonomia do sistema quando alimentado por este arranjo de acumuladores, é necessário calcular a energia armazenada na fonte (*EA)*, onde:

$EA = \int_{t_i}^{t_f} (V \times I) dt$. $\qquad EA = 64800J$

Para o cálculo da autonomia do sistema considerando-se esta fonte de alimentação, tem-se:

$A = \frac{EA}{PS}$ $\qquad A = \frac{64800}{0{,}010242} = 6.326.889{,}28s = 1.757{,}47h = 73{,}23dias$

Para a fonte especificada, a autonomia estimada é de 73,23 dias ou 1.757,47 horas de uso.

Considerando-se o modelo proposto, experimentos podem ser realizados variando-se os pesos associados às transições conflitantes, de maneira que diversos cenários podem ser avaliados.

A Figura 4.11 mostra o resultado do experimento realizado, onde se variou os pesos associados às *Tarefas 3* e *Tarefa 4*. No experimento, a *Tarefa 3* inicia com

probabilidade de execução um(1), enquanto a *Tarefa 4* com zero(0). Ao longo do experimento, diversos cenários são analisados, onde as probabilidades de execução vão se invertendo até que a *Tarefa 4* atinja a probabilidade de execução um(1) e a *Tarefa 3* zero(0). No gráfico da Figura 5.11 (automaticamente gerado pelo *TimeNET 3.0*), atribui-se a variação de probabilidade de execução da *Tarefa 4* ao eixo X e a métrica potência dissipada ao eixo Y.

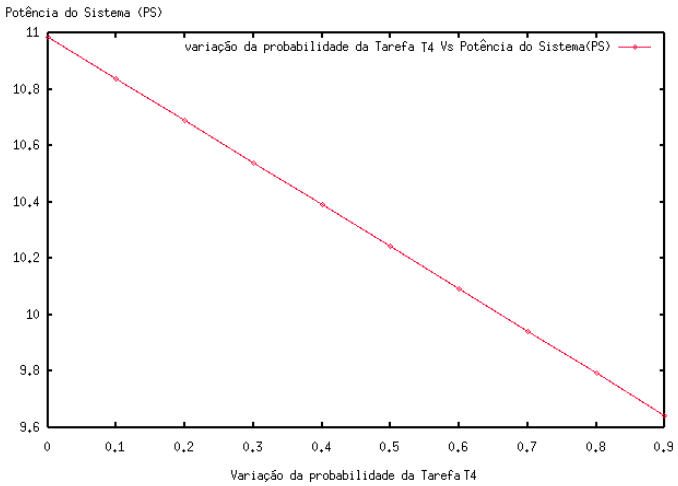


Figura 4.11:Variação das probabilidades da Tarefa 4 X Potência (PS).

Observa-se, na Figura 4.11, que à medida que a *Tarefa 4* tem aumentada sua probabilidade de execução, a potência dissipada pelo sistema diminui.

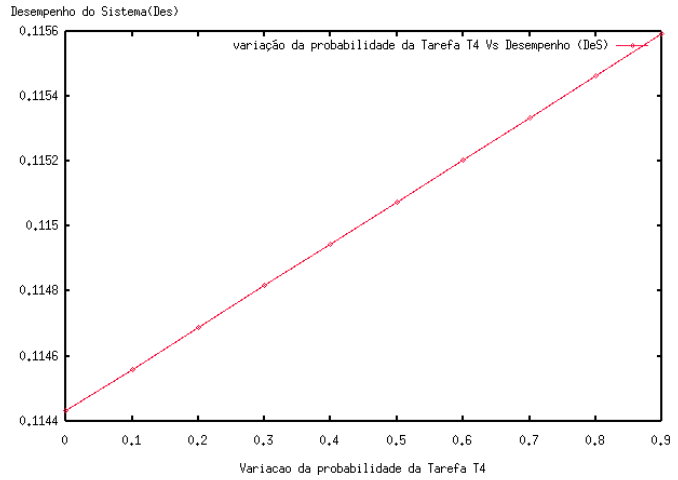


Figura 4.12:Variação das probabilidades Tarefa 4 X Taxa de execução (Tx).

Na Figura 4.12 atribui-se a variação da probabilidade da *Tarefa 4* ao eixo X e a métrica de *taxa de execução,* a qual neste estudo de caso é o *índice de desempenho* do sistema, ao eixo Y. Observa-se que com o aumento da probabilidade de execução da *Tarefa 4,* a taxa de execução do sistema aumenta proporcionalmente. De acordo com o experimento, concluí-se que a *Tarefa 4* além de ser mais rápida que a *Tarefa 3* dissipa menos potência.

4.9 Resumo

Neste capítulo apresentou-se um modelo para avaliação e estimativas de consumo de energia e desempenho em sistemas embarcados. Inicialmente, foi definido o modelo denominado Power-GSPN. O modelo proposto é uma rede GSPN estendido com funções de consumo de energia mapeadas às transições e lugares da rede.

Apresentou-se também, de maneira informal como uma especificação de um sistema pode ser representada no modelo Power-GSPN. Apesar do modelo Power-GSPN ter apenas transições imediatas e transições exponenciais, diversos outros tipos de distribuições podem ser modeladas através de aproximações. Neste trabalho, tempos determinísticos foram aproximados por fases de exponenciais. A técnica adotada para medição da energia atribuída aos elementos do modelo, também foi apresentada. O consumo de energia de cada ação do sistema é utilizado para caracterizar a transição equivalente no modelo Power-GSPN. Um conjunto de métricas de desempenho, consumo de energia, potência dissipada e autonomia foram definidas, a fim de se obter as estimativas do sistema.

Um pequeno exemplo foi apresentado para ilustrar o modelo proposto, assim como para o método descrito. Finalmente, a solução estacionária foi apresentada.

Capítulo 5 - Estudo de Caso

Este capítulo, objetivando a validação da metodologia proposta, apresenta os resultados da aplicação da metodologia. Inicialmente, uma descrição breve dos sistemas utilizados como estudo de caso será apresentado. A seguir, serão descritos os modelos dos sistemas estudados. Finalmente, os resultados serão apresentados e comentados.

5.1 Introdução

Os exemplos utilizados neste capítulo são sistemas embarcados reais. Neste sentido, procura-se demonstrar a efetividade do modelo e da metodologia desenvolvida para se estimar o consumo de energia em sistemas embarcados. Três estudos de casos serão mostrados neste capítulo. O primeiro deles é um sistema de verificação de Código de Redundância Cíclica de 8 *bits* (*CRC8 - Cyclic Redundancy Codes*). Este código é muito utilizado para transmissão de dados em sistemas, onde se verifica a integridade dados transmitidos. O segundo estudo de caso é um sistema de classificação (*Bubble sort*). O Terceiro estudo de caso é um Oxímetro de Pulso, equipamento utilizado para medir o nível ou quantidade de oxigênio no sangue.

Este capítulo aplica os passos gerais da metodologia proposta. Em seguida, apresentam-se os modelos. Posteriormente, o modelo será avaliado em estado estacionário. Os resultados desta avaliação provêem as informações necessárias para o cálculo das métricas. Finalmente, os resultados serão comentados para possibilitar uma compreensão adequada da avaliação realizada.

5.2 Sistema de verificação de Código de Redundância Cíclica de 8 bits (CRC-8).

No contexto de sistemas computacionais, integridade pode ser informalmente definida como a habilidade para assegurar que uma informação não seja alterada por uma entidade sem autorização. Controles de acesso previnem mudanças sem autorização e é uma forma de proteção da integridade. Outros mecanismos detectam mudanças sem autorização em lugar de preveni-las. Verificação através de *Checksums* e código de redundância cíclica (CRC) são exemplos dos primeiros mecanismos de detecção de integridade. A integridade representa um papel essencial onde a garantia de que o conteúdo de uma mensagem está correto é mais importante que manter a mensagem confidencial.

CRC é uma abreviação de *Cyclic Redundancy Codes* e baseia-se na anexação de informação extra que pode ser usada em qualquer situação, não apenas no contexto de transmissão de dados. Através desta informação, pode-se testar se um arquivo está intacto antes de utilizá-lo. Qualquer alteração no arquivo altera o valor código CRC. O código CRC é um número calculado aplicando-se diversas operações aritméticas aos bytes da informação.

A rotina CRC foi escolhida como estudo de caso dado a incerteza dos dados de entrada, pois desta forma podemos estimar a variação do consumo de energia, potência dissipada e desempenho, em função das possíveis entradas. Serão realizados experimentos para varrer as possibilidades de dados de entrada. Esta é uma das grandes vantagens da abordagem proposta, pois torna possível a avaliação da potência e desempenho não apenas para um contexto ou entrada específica.

Apresenta-se a seguir o código da rotina CRC-8 implementada em *Assembly* do processador 8051.

```
CRC8_ADD_CALC:
proc
                mov  0AH,acc        salva Acc
                mov  0BH,r0         salva R0
                mov  0CH,acc        salva Valor
                mov  r0,#8
loop1:          xrl  a,b            calcula o CRC
                rrc  a              move para o Carry
                mov  a,b                    pega o último
valor CRC
                jnc  loop2                  salta se Bit =
0
                xrl  a,#0b1H;#18h           atualiza o
valor CRC
loop2:          rrc  a              posição do novo
CRC
```

Figura 5.1:Código da rotina CRC-8 implementada em Assembly.

O sistema CRC-8, escrito em assembly, mostrado na Figura 5.1, inicia salvando o contexto atual do *acumulador* e do registrador R_0, tarefa esta, realizada pelo conjunto das quatros primeiras instruções. Nas próximas quatro instruções, calcula-se o valor do CRC. Caso o valor do *bit* corrente seja um (1), então o valor *CRC* calculado é somado ao último valor *CRC* acumulado, caso contrário o valor *CRC* deste *bit* é descartado. Esta *soma* é realiza em uma única instrução. Este procedimento será repetido para os outros sete *bits* restantes do *Byte*. Nas cinco instruções seguintes, verifica-se o final da palavra, para fazer nova interação ou sair do laço principal. Por fim, as três últimas instruções restauram o contexto do *Acumulador* e R_0 que foi salvo na entrada do sistema.

5.2.1 O modelo Power-GSPN do CRC-8

O código CRC pode ser dividido em cinco (5) blocos básicos, que realizam tarefas específicas. São elas:

- Salva contexto;
- Cálculo *CRC*;
- Soma;

- Laço;
- Restaura contexto.

A rotina do cálculo do CRC tem dois condicionais. O primeiro verifica se o valor do *bit* calculado é igual a 1. Neste caso, o valor *CRC* calculado será adicionado ao último valor do CRC. O segundo condicional determina quando chegou o último *bit* do *Byte,* para finalizar a estrutura de repetição.

Dois experimentos foram realizados, no primeiro deles variou-se a probabilidade dos *bits* do *Byte* ser igual a um. Apenas quando os *bits* são iguais a um é que se executa a *soma* na rotina *CRC-8*. No segundo experimento, variou-se a probabilidade do tamanho da palavra de entrada. Este experimento reflete na quantidade de interações do laço principal. No caso do *CRC-8*, como o próprio nome já sugere, é utilizada uma palavra de oito *bits* (*Byte)*, onde são feitas oito interações. Neste segundo experimento, serão estimadas a potência e desempenho para diversos tamanhos de palavra.

Como as cinco tarefas têm o tempo de execução determinístico, será utilizado o recurso da aproximação por fases, descrito no Capítulo 4, para a criação do modelo . Na Figura 5.2 apresenta-se o modelo Power-GSPN do sistema *CRC-8.* Nota-se que, toda ação do sistema é representada por uma transição, conforme associação entre a Figura 5.2 e a Tabela 5.1.

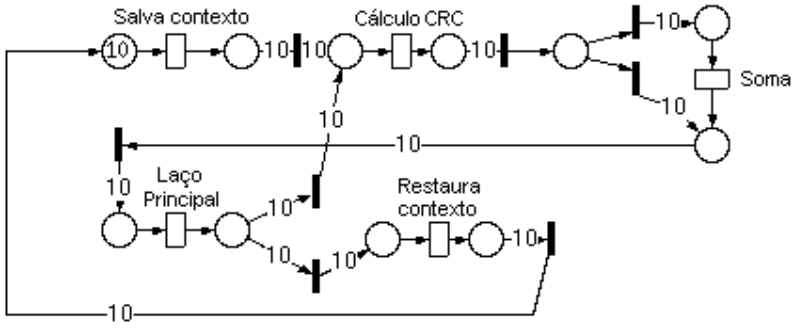


Figura 5.2:Modelo do CRC-8 em Power-GSPN.

Com o modelo do CRC-8 construído, deve-se atribuir às transições informações de tempo, prioridade e consumo de energia. As informações temporais são facilmente obtidas pela soma dos tempos das instruções que compõe cada tarefa. As prioridades, neste modelo, não têm relevância dado que as tarefas são executadas seqüencialmente não existindo concorrência entre elas. Os valores de consumo de energia das tarefas foram obtidos segundo a técnica mostrada no Capítulo 4.

Tabela 5.1:Informações de Tempo, prioridade e Consumo de energia das tarefas do CRC.

Tarefas	Instruções	Tempo(μs)	Prioridade	Energia (ηJ)
Salva contexto	MOV 0AH,Acc	5	1	263
	MOV 0BH,R_0			
	MOV 0CH,Acc			
	MOV R_0,#8			
Cálculo CRC	XRL A,B	4	1	202,2
	RRC A			
	MOV A,B			
	JNC Loop2			
Soma	XRL A,#0B1H	1	1	49,7
Laço principal	RRC A	7	1	308,4
	MOV B,A			
	MOV Acc,0AH			
	RR A			
	MOV 0AH,Acc			
	DJNZ R_0,Loop1			
Restaura contexto	MOV Acc, 0CH	6	1	236,3
	MOV R_0, 0BH			
	MOV Acc, 0AH			

5.2.2 Comparação da potência medida com a estimada

Objetivando demonstrar a fidelidade da metodologia proposta, será confrontada apenas para este estudo, a potência medida em um contexto específico do sistema, com as estimativas obtidas pelo calcula das métricas em regime estacionário.

Para a comprovação será utilizado o contexto onde a palavra de entrada tem apenas um bit de valor lógico um. Neste contexto, o sistema CRC-8, executará uma vez cada tarefa e potência dissipada será calculado pelo somatório da energia consumida de todas as tarefas dividido pelo somatório do tempo de execução de cada tarefa, dado por:

$$PS = \frac{\sum_{i=1}^{5} E(t_i)}{\sum_{i=1}^{5} t_i} = \frac{((263)+(202,2)+(49,7)+(308,4)+(236,3))}{(5)+(4)+(1)+(7)+(6)} = 46,06957mW$$

Agora será computadas as estimativas de potência dissipada para o mesmo contexto descrito, apenas um *bit* com valor lógico 1.

Tabela 5.2:Métricas de estimativas de potência e desempenho das tarefa do CRC8.

Tarefas	Tempo (μs)	**Probabilidade**	Energia (ηJ)	**Taxa de Execução(e/s)**	**Potência Dissipada ($_m$W)**
Salva Contexto	5	**0,217391**	263	**43.478,260870**	**11,434783**
Cálculo	4	**0,173913**	202,2	**43.478,260870**	**8,782609**
Soma	1	**0,043478**	49,7	**43.478,260870**	**2,173913**
Laço principal	7	**0,304348**	308,4	**43.478,260870**	**13,391304**
Restaura contexto	6	**0,260870**	236,3	**43.478,260870**	**10.260870**

Tendo obtido a potência dissipada para cada tarefa, calcula-se a potência total dissipada pelo sistema, conforme a equação:

$$PS = \sum_{t_i \in T} P(t_i) \qquad \textit{Potência dissipada pelo Sistema - } PS = 46{,}043478 \text{m}W$$

$$\text{Diferença da medição X estimativa} = \frac{Valor\ Medido}{Valor\ Estimado} \times 100 = \frac{46{,}06957}{46{,}043478} \times 100 = 0{,}01\%$$

Verifica-se que as estimativas obtidas através do cálculo em regime estacionário das métricas definida pelo método proposto é fiel ao sistema real.E além da fidelidade, se tem uma precisão bem apurada dos resultado.

5.2.3 Estimativas de potência e desempenho

Tendo atribuído as informações relativas ao tempo, prioridade e consumo de energia às transições do modelo, associam-se aos elementos da rede às fórmulas das métricas para obtenção das estimativas de potência e desempenho, conforme as equações apresentadas no Capítulo 4.

A partir da solução da analise estacionária, obtêm-se as probabilidades de habilitação das transições $PHT(t_i)$**,** para todo $t_i \in T$. Utilizando-se estas informações é possível calcular a taxa de execução das transições. Dado que determinadas transições

representam a execução de uma tarefa (bloco binário), é possível calcular sua taxa de execução.

$$Tx(t_i) = PHT(t_i) \times \lambda(t_i)$$

Onde :

$PHT(t_i)$ **-** representa a probabilidade de habilitação de uma transição t_i.

λ - é a taxa de disparo da transição exponencial, $\lambda = \frac{1}{D}$.

D – é o retardo (*delay*) de uma transição determinística.

Após o cálculo da taxa de execução $Tx(t_i)$ das transições, que é dado em número de execuções por segundo(s^{-1}), calcula-se a potência dissipada associada a cada transição ($P(t_i)$), conforme equação:

$$P(t_i) = Tx(t_i) \times E(t_i)$$

Onde:

$E(t_i)$ **-** representa o consumo de energia para cada disparo de uma transição.

$P(t_i)$ – representa a potência dissipada associada a transição $t_i \in T$.

Dado que a execução das transições com suas características representam a execução das tarefas. Os resultados da análise de consumo de energia e desempenho do sistema podem ser obtidos. Os resultados desta análise são apresentados na Tabela 5.3.

Tabela 5.3:Métricas de estimativas de potência e desempenho por tarefa do CRC8.

Tarefas	Tempo (μs)	**Probabilidade**	Energia (ηJ)	**Taxa de Execução(e/s)**	**Potência Dissipada ($_m$W)**
Salva Contexto	5	**0,2089**	263	**41.775**	**10,987**
Cálculo	4	**0,188**	202,2	**46.997**	**9,492**
Soma	1	**0,0235**	49,7	**23.499**	**1,175**
Laço principal	7	**0,329**	308,4	**46.997**	**14,475**
Restaura contexto	6	**0,2506**	236,3	**41.775**	**9,86**

Tendo obtido a potência dissipada para cada tarefa, calcula-se a potência total dissipada pelo sistema, conforme a equação:

$$PS = \sum_{t_i \in T} P(t_i) \qquad \textit{Potência dissipada pelo Sistema - } PS = 45{,}989 \mathrm{m}W$$

O índice de desempenho definido neste estudo de caso é a taxa de execução do sistema, que é obtido, para este cenário, através da taxa de execução da tarefa *restaura contexto*, pois esta é a última tarefa a ser executada, a qual sua taxa de execução é mostrada na Tabela 5.3. Esta informação representa a taxa de execução do sistema dado que a execução da tarefa *restaura contexto* só é possível após a execução de todas as demais tarefas.

Índice de Desempenho do Sistema (Taxa de execução – Tx) = 41.775 s^{-1}

Estas estimativas foram feitas considerando-se a probabilidade dos dados de entrada (*bits*) serem 50% igual a um (1) e, consequentemente, 50% serem igual a zero(0). Além de considerar a palavra de entrada de oito *bits* (*Byte*), a probabilidade atribuída ao corpo do laço foi de $\frac{8}{9}$, isto é, a cada oito interações a estrutura de repetição é descontinuada.

Para este estudo de caso, em princípio, não seria muito relevante o cálculo da autonomia, visto que este sistema geralmente está associado a um sistema maior e que ele nem sempre esta sendo executado. A estimativa da autonomia será realizada para o caso do Oxímetro de Pulso.

5.2.4 Experimentos com variações de parâmetros

O resultado da avaliação apresentado na seção anterior foi obtido considerando-se os parâmetros com valores fixos. Foi atribuída uma probabilidade de 50% para ocorrência para valores zero (0) e, consequentemente, outros 50% para valores um (1) de cada *bit* do *Byte*. Nesta seção serão realizados experimentos onde, apenas um parâmetro será variado por vez (de um valor mínimo até um valor máximo). Serão traçados gráficos

com os resultados da avaliação (potência e desempenho) em função da variação do parâmetro.

1° Experimento: Variar a probabilidade do bit de entrada assumir nível lógico um (1).

Alterar a probabilidade dos *bits* do *Byte* de entrada assumir nível lógico um (1) influencia a probabilidade de execução da tarefa *Soma*. Esta tarefa é executada apenas quando o *bit* de entrada tem nível lógico um (1). A Figura 5.3 apresenta a potência dissipada total (*PS*) em função da probabilidade dos *bits* assumirem nível lógico um (1).

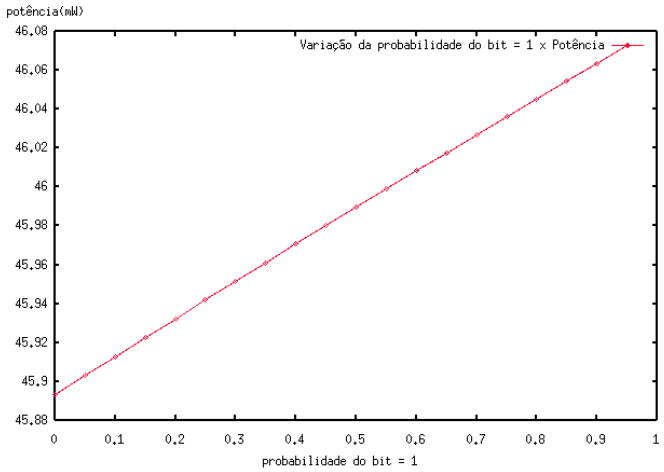


Figura 5.3: Variação da probabilidade do bit ser um (1) X potência dissipada.

Na Figura 5.3, observa-se que quanto maior a probabilidade do *bit* assumir nível lógico um (1), maior será a potência dissipada. Este aumento está diretamente relacionado ao maior número de execuções tarefa de *soma*.

A Figura 5.4 mostra a taxa de execução do sistema *Tx* em função da probabilidade dos *bits* assumirem nível lógico um (1).

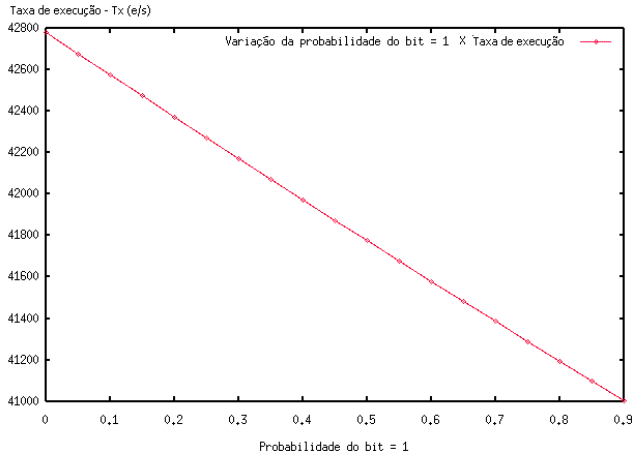


Figura 5.4:Variação da probabilidade do bit ser um X Taxa de Execução do sistema.

A Figura 5.4 demonstra ao aumentar a probabilidade do bit assumir nível lógico um, menor será o desempenho dado um maior número de execuções da tarefa *soma.*

2° Experimento:Variar a probabilidade do tamanho da palavra de entrada.

A influência do tamanho da palavra de entrada pode ser avaliada alterando-se a probabilidade de interações do laço principal. Para uma probabilidade de valor igual zero (0%), o tamanho da palavra correspondente é de apenas um *bit*. O laço principal fará apenas uma interação. Quanto maior o tamanho da palavra de entrada a ser processada, maior o número de interações do laço principal. A Figura 5.5 mostra a potência dissipada (*PS*) em função do tamanho da palavra.

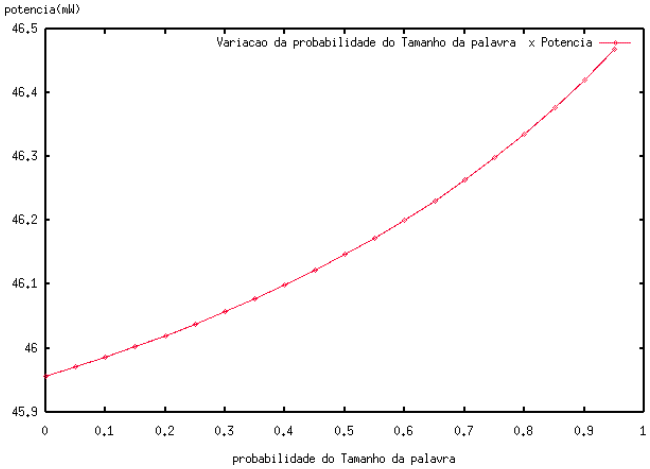


Figura 5.5:Variação do tamanho da palavra de entrada X Potência dissipada pelo sistema PS.

A Figura 5.5 mostra que a potência dissipada cresce, à medida que o tamanho da palavra de entrada aumenta, pois maior o número de interações do laço principal. Este experimento é interessante, pois temos as estimativas de potência dissipada para palavras de um *bit* em diante.

A Figura 5.6 apresenta o desempenho do sistema, no caso à taxa de execução (*Tx*), em função do tamanho da palavra.

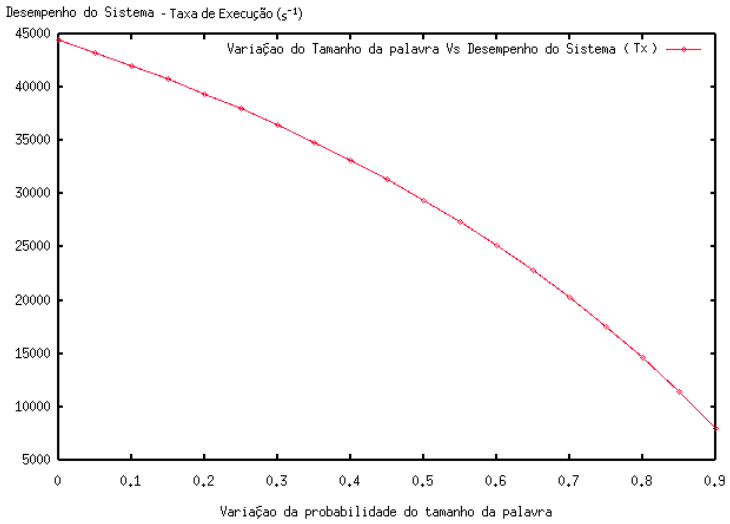


Figura 5.6:Variação do tamanho da palavra de entrada X Taxa de execução (Tx).

A Figura 5.6. demonstra que o índice de desempenho do sistema (Taxa de execução) degrada à medida que o tamanho da palavra de entrada cresce, pois maior será o número de interações do laço principal. Este experimento é interessante, pois se tem estimativas de potência dissipada e desempenho para diversos tamanhos de palavra de entrada.

5.3 Código da rotina de ordenação Bubble Sort.

O *Bubble Sort* é um algoritmo de ordenação simples que precisa de $O(n^2)$ comparações para ordenar *n* itens. Ele é um dos mais simples algoritmos de ordenação conhecidos, mas geralmente é considerado de baixo rendimento para trabalhos que ordenam um grande número de elementos. O algoritmo compara e troca a posição dos elementos que estão desordenados. Apesar de serem muito simples, o *bubble sort* tende a ser um dos

mais eficientes algoritmos de ordenação que se destinam a ordenar um pequeno número de elementos. O algoritmo *bubble sort* faz:

1. Compara elementos adjacentes. Se o primeiro é maior do que o segundo, troca eles.

2. Faz isto para cada par de elementos adjacentes, começando com os dois primeiros e terminando com os dois últimos. Neste ponto o último elemento deve ser o maior de todos.

3. Repete os passos para todos os elementos exceto para o último.

4. Continua repetindo, cada vez com um elemento a menos, até que não existam mais pares para se comparar.

É mostrado o código da rotina em Assembly do *Bubble Sort* na Figura 5.7.

```
Bubble_sort: ;      proc
                              mov r4,#00H
loop:                         mov a,r4
                              add a,40H
                              mov r4,a
                              mov 00H,r4
                              mov r1,00H
inner_loop:                   inc r1
                              mov a,@r0
                              clr c
                              subb a,@r1
                              jnc exit
                              mov b,@r0
                              mov a,@r1
                              mov @r0,a
                              mov @r1,b
exit:                         mov a,r1
                              clr c
                              subb a,40H
                              cjne a,41H,inner_loop
                              mov a,r4
                              clr c
                              subb a,40H
                              mov r4,a
                              inc r4
                              mov a,r4
                              cjne a,41H,loop
                              ret
        endproc
end
```

Figura 5.7:Código em Assembly da rotina de ordenação Bubble sort.

O Sistema *Bubble sort*, escrito em assembly, mostrado na Figura 5.7, inicia limpando o conteúdo do registrado R_4 a essa tarefa denominou-se de *início*. Em seguida, tem-se a estrutura de repetição do *laço principal* e depois um *laço interno*. Faz-se a comparação da grandeza dos elementos, caso estejam desordenados é realizado a *troca* da posição dos elementos. Tem-se a finalização do *laço interno*, em seguida a finalização do *laço principal*.

O sistema de ordenação *Bubble sort* tem dois condicionais. O primeiro compara dois elementos para verificar se estão ordenados. No caso os elementos não estejam ordenados executa-se a tarefa *troca,* que muda a posição entre os elementos. O segundo condicional determina o final do conjunto de elementos a serem ordenados, finalizando a estrutura de repetição do *laço principal*. Dois experimentos foram realizados, no primeiro deles variou-se a probabilidade do conjunto de elementos estarem ordenados. Apenas quando os elementos estão desordenados é que se executa a *troca*. No segundo experimento, variou-se a probabilidade da quantidade de elementos a ser ordenado. O primeiro experimento reflete na quantidade de *trocas* da posição dos elementos que será realizada. O segundo experimento será estimado a potência e o desempenho para diversos números de elementos de entrada.

Como as seis tarefas têm o tempo de execução determinístico, será utilizado o recurso da aproximação por fases para a criação do modelo, descrito no Capítulo 4. Na Figura 5.8, apresenta-se o modelo Power-GSPN do sistema *Bubble sort.* Nota-se que toda ação do sistema é representada por uma transição, conforme associação entre a Figura 5.8 e a Tabela 5.4.

5.3.1 A modelo Power-GSPN do *Bubble Sort*

O código do *Bubble sort* pode ser dividido seis (6) blocos básicos, que realizam tarefas específicas. São elas:

- Início;
- Laço principal;

- Laço interno;
- Troca;
- Final do laço interno;
- Final do laço principal.

Como as seis tarefas têm o tempo determinístico será feita a aproximação por fase. O modelo é composto por seis tarefas determinísticas, e três escolhas. As escolhas são:

- Executar a troca de posições entre os elementos, quando o elemento seguinte é maior que seu antecessor;
- O fim do laço interno;
- Determinar o fim dos elementos e sair do laço principal da rotina de ordenação.

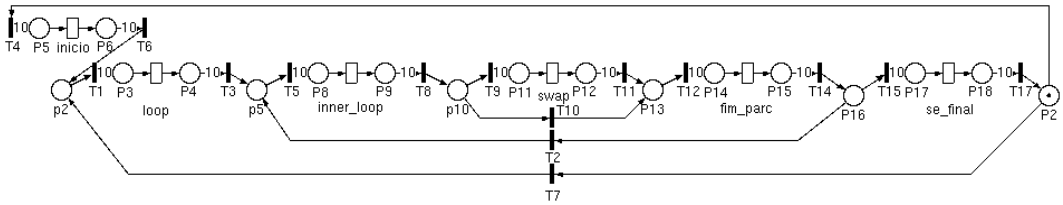


Figura 5.8:Modelo do Bubble sort em Power-GSPN.

Com o modelo do CRC-8 construído, deve-se atribuir às transições informações de tempo, prioridade e consumo de energia. As informações temporais são facilmente obtidas pela soma dos tempos das instruções que compõe cada tarefa. As prioridades, neste modelo, não têm relevância dado que as tarefas são executadas seqüencialmente não existindo concorrência entre elas. Os valores de consumo de energia das tarefas foram obtidas segundo a técnica mostrada no Capítulo 4.

Tabela 5.4:Informações de tempo, consumo de energia e prioridade das tarefas do bubble sort

Tarefas	Tempo(μs)	Prioridade	Energia (ηJ)
Início	1	1	55,3
Laço principal	7	1	365,3
Laço interno	6	1	313,9
Troca	6	1	295,6
Fim do laço interno	5	1	268,9
Fim do laço principal	7	1	430,8

5.3.2 Estimativas de potência e desempenho

Tendo atribuído as informações relativas ao tempo, prioridade e consumo de energia às transições do modelo e formuladas as expressões que representam as métricas para obtenção das estimativas de potência e desempenho, o modelo é avaliado.

A partir da solução da analise estacionária, obtêm-se as probabilidades de habilitação das transições $PHT(t_i)$**,** para todo $t_i \in T$. Utilizando-se estas informações é possível calcular a taxa de execução das transições. Dado que determinadas transições representam a execução de uma tarefa (bloco), é possível calcular a taxa de execução das tarefas.

$$Tx(t_i) = PHT(t_i) \times \lambda(t_i)$$

Onde :

$PHT(t_i)$ **-** representa a probabilidade de habilitação de uma transição t_i.

λ - é a taxa de disparo da transição exponencial, $\lambda = \frac{1}{D}$.

D – é o retardo (*delay*) de uma transição determinística.

Após o cálculo da taxa de execução $Tx(t_i)$ das transições, que é dado em número de execuções por segundo($\frac{1}{s}$), calcula-se a potência dissipada associada a cada transição, conforme equação:

$$P(t_i) = D(t_i) \times E(t_i)$$

Onde:

$E(t_i)$ **-** representa o consumo de energia para cada disparo de uma transição.

$P(t_i)$ – representa a potência dissipada associada a transição $t_i \in T$.

Dado que a execução das transições com suas características representam a execução das tarefas, os resultados da análise de consumo de energia e desempenho do sistema podem ser obtidos. Os resultados desta análise são apresentados na Tabela 5.5.

Tabela 5.5:Métricas de estimativas de potência e desempenho das tarefa do *bubble sort*.

Tarefas	Tempo(μs)	Prioridade	Energia (ηJ)	**Probabilidade**	**Desempenho**	**Potência dissipada**
Início	1	1	55,3	**0,01176**	**11764,7**	**0,647**
Laço principal	7	1	365,3	**0,1647**	**23529,41**	**8,611**
Laço interno	6	1	313,9	**0,2823**	**47058,82**	**14,776**
Troca	6	1	295,6	**0,1412**	**23529,41**	**6,965**
Fim do laço interno	5	1	268,9	**0,2353**	**47058,82**	**12,659**
Fim do laço principal	7	1	430,8	**0,1647**	**23529,41**	**10,141**

Tendo obtido a potência dissipada para cada tarefa, calcula-se a potência total dissipada pelo sistema, conforme a equação:

$$PS = \sum_{t_i \in T} P(t_i) \qquad \text{Potência dissipada pelo Sistema} - PS = 53{,}97 \text{ m}W$$

O índice de desempenho, neste estudo de caso, é obtido através da taxa de execução da tarefa *iníco*, pois esta é a primeira tarefa a ser executada. Esta informação representa a taxa de execução do sistema dado que uma nova execução da tarefa *início* só é possível após a execução de todas as demais tarefas.

Índice de Desempenho do Sistema (Taxa de execução – Tx) = 11.764,7 s^{-1}

Assim como o estudo de caso do CRC8, não faz muito sentido nos calcularmos a autonomia do sistema, visto que esta rotina geralmente está associada a um sistema maior e que nem sempre está sendo executada.

Estas estimativas foram feitas considerando-se a probabilidade dos elementos de entrada estarem 50% ordenados. Além de considerar a palavra de entrada de oito *bits* (*Byte*), probabilidade atribuída ao corpo do laço foi de $\frac{8}{9}$, isto é, a cada oito interações a estrutura de repetição é descontinuada.

5.3.3 Experimentos com variação de parâmetros

O resultado da avaliação apresentado na seção anterior foi obtido considerando-se os parâmetros com valores fixos. Foi atribuída uma probabilidade de 50% que os elementos do conjunto de entrada estejam ordenados. Para o número de elementos do conjunto que será ordenado, atribuiu-se uma probabilidade de $\frac{8}{9}$. Nesta seção serão realizados experimentos onde, apenas um parâmetro será variado por vez (de um valor mínimo até um valor máximo). Serão traçados gráficos com os resultados da avaliação (potência e desempenho) em função da variação do parâmetro.

1° Experimento: Variar a probabilidade dos elementos do conjunto de entrada estar ordenados.

Alterando a probabilidade dos elementos do conjunto de entrada estarem ordenados influi na probabilidade de execução da tarefa *Troca*. Esta tarefa é executada apenas quando os dois elementos comparados estão desordenados. A Figura 5.9 apresenta a potência dissipada total (P*S*) em função da probabilidade dos elementos estarem ordenados.

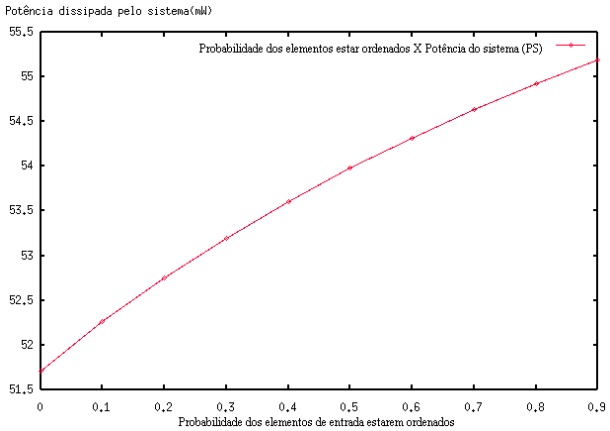


Figura 5.9:Variação da probabilidade dos elementos estarem ordenados X potência dissipada pelo sistema (PS) (*bubble sort*).

Na Figura 5.9, observa-se que com o aumento da probabilidade dos elementos estarem ordenados, diminui a potência dissipada. Esta diminuição da potência está diretamente relacionada ao maior número de execuções da tarefa de *troca* de elementos e o *laço interno*.

A Figura 5.10 mostra o índice de desempenho do sistema (*IDS*). Para este estudo de caso é a taxa de execução do sistema (*Tx)*. Obtêm-se o respectivo valor através da taxa de execução da tarefa *início*. A tarefa início é a única tarefa que esta fora do laço principal. De forma que, a taxa de execução da tarefa *início* reflete a taxa de execução global do sistema.

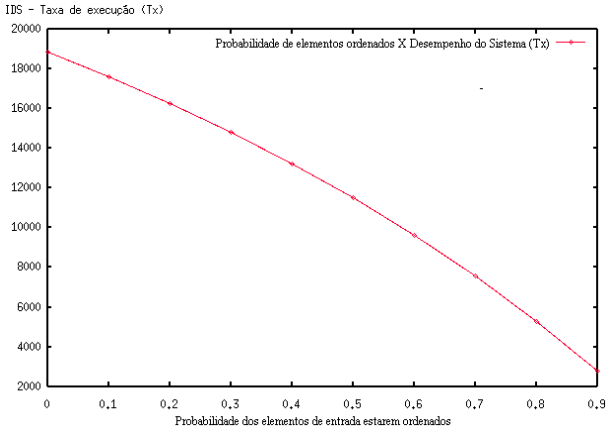


Figura 5.10:Variação da probabilidade dos elementos estarem ordenados X Taxa de execução do sistema. (Tx) (*bubble sort*).

A Figura 5.10 demonstra que quanto mais ordenados estejam os elementos de entrada, maior o desempenho do sistema, isto é, mais rápido serão obtido os resultados.

2° Experimento: Variar a probabilidade da quantidade de elementos do conjunto de entrada.

A influência da quantidade de elementos do conjunto de entrada pode ser avaliada alterando-se a probabilidade de interações do *laço principal*. Para uma probabilidade igual zero (0%), o tamanho da palavra correspondente é de dois elementos. O laço principal fará apenas uma interação. Quanto maior o número de elementos do conjunto de entrada a ser processada, maior o número de interações do laço principal. A Figura 5.11 mostra a potência dissipada (*PS*) em função da quantidade de elementos do conjunto de entrada.

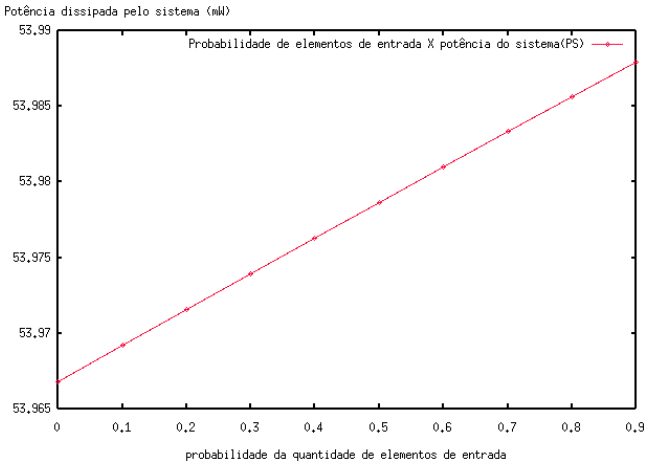


Figura 5.11:Variação do número de elementos do conjunto de entrada X Potência dissipada pelo sistema (PS) - (bubble sort).

A Figura 5.11 mostra que o aumento do número de elementos do conjunto de dados a serem ordenados gera um aumento na potência dissipada. Este fato ocorre devido ao maior o número de interações no laço principal. Este experimento é interessante, pois provê estimativas de potência dissipada para diversos tamanhos de conjuntos entrada.

A Figura 5.12 apresenta o desempenho do sistema, a taxa de execução (*Tx*), em função do número de elementos do conjunto de entrada.

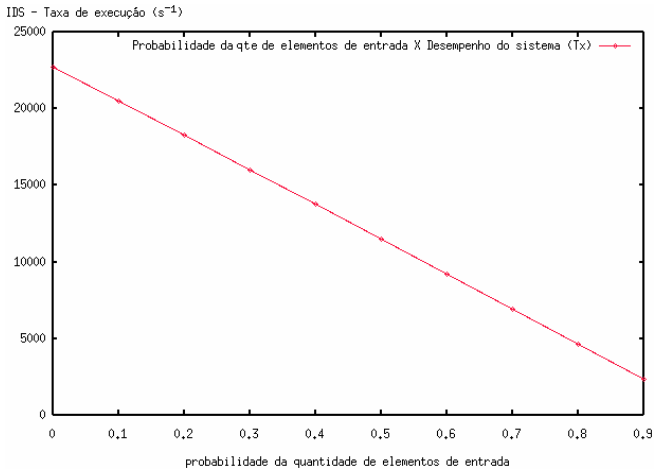


Figura 5.12:Variação do tamanho da palavra de entrada X Taxa de execução do sistema (Tx) - (bubble sort).

A Figura 5.12 apresenta a variação do desempenho do sistema (taxa de execução) como o aumento do número de elementos dos conjuntos de dados a serem ordenados. Nota-se que, à medida que o número de elementos dos conjuntos a serem ordenados aumenta, degrada-se o desempenho do sistema, isto é, a taxa com que o sistema consegue ordenar os conjuntos diminui em função do aumento do tamanho dos conjuntos. Este experimento possibilita a estimativa de desempenho para diversos tamanhos de conjuntos entrada.

5.4 Oxímetro de pulso

O oxímetro de pulso [67] é um equipamento que utiliza em duas categorias tecnológicas distintas: a tecnologia analógica e a digital. A aquisição e condicionamento do sinal proveniente do meio biológico é feita por estruturas analógicas devido à natureza analógica do sinal de interesse. A extração da informação de saturação e freqüência cardíaca é realizada mediante a digitalização do sinal e seu processamento por algoritmos computacionais específicos. Sendo assim, o instrumento é formado por um módulo analógico e um microprocessado (digital). Na Figura 5.13

observar-se a estrutura genérica de um oxímetro de pulso. Situações de baixa perfusão refletem-se no sinal como uma baixa neste índice de modulação.

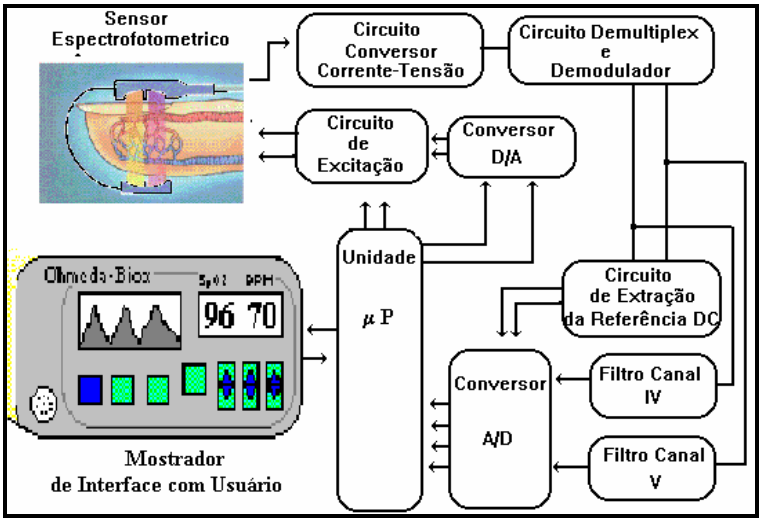


Figura 5.13:Estrutura básica de um oxímetro de pulso.

A unidade microprocessada cumpre o papel de controlar os padrões de excitação de forma a manter a melhor situação de leitura possível. A interface com o usuário é constituída de mostradores de sete segmentos e chaves de programação de alarmes máximos e mínimos para a saturação e freqüência cardíaca. Alguns oxímetros oferecem também a apresentação do sinal fotopletismográfico em um mostrador gráfico.

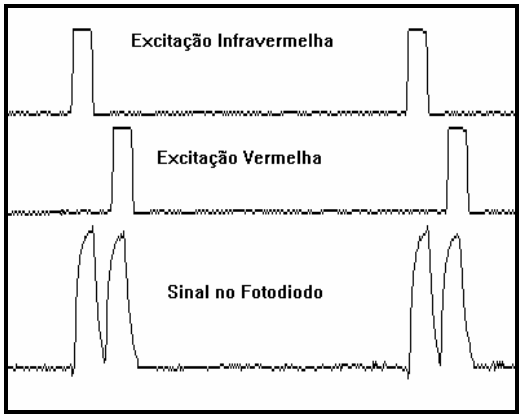


Figura 5.14:Sinais na sonda de um oxímetro de pulso.

A oximetria de pulso [67] mostrou-se uma técnica altamente eficiente na superação da problemática da absorção tecidual e do componente venoso da saturação,

sendo por isto consagrada como o estado da arte em oximetria. O atual desafio na busca de sistemas mais eficientes de leitura não evasiva de SaO_2 é o combate aos chamados artefatos de movimento - surtos de absorção, devido ao movimento relativo entre paciente e sensor. Gerações de Oxímetro foram desenvolvidos na tentativa de suplantar os efeitos dos artefatos de movimento, inicialmente com a melhoria da ergonometria dos sensores e sincronização da leitura com um sinal eletrocardiográfico. Atualmente, com a crescente disposição de microprocessadores dedicados ao processamento digital de sinais (DSP), a custos relativamente baixos, sistemas baseados em filtros de convolução e adaptativos indicam uma nova tendência.

5.4.1 O modelo Power-GSPN do Oxímetro de pulso

O sistema do Oxímetro de Pulso é dividido em três partes distintas, denominadas *planos*. Os *planos* são programas que executam independente um dos outros. Cada um destes *planos* é dividido em sub-tarefas. Seguindo este conceito, nomeia-se os planos de acordo com a atividade macro que eles desempenham. São elas:

1°Plano – Rotina de Excitação;

2°Plano – Rotina de Amostragem e controle;

3°Plano – Programa Gerenciador.

Neste estudo de caso, considerou-se para modelagem e estimativas apenas o 1° plano e o 2° plano, dado que o 3° plano diz respeito à inicialização do sistema, atendimento aos comandos do usuário entre outras atividades de menor relevância neste contexto.

No 1° Plano – Rotina de Excitação são realizadas as seguintes tarefas:

- Geração dos pulsos de excitação, vermelho e infravermelho;
- Temporização entre os pulsos de excitação vermelho e infravermelho;
- Controle da freqüência dos pulsos de excitação: vermelho e infravermelho;

- Leitura do conversor digital/ analógico.

No 2° Plano – Rotina de Amostragem e controle são realizadas as seguintes tarefas:

- Disparo de Conversão A/D;
- Aguarda conversão;
- Lê conversor;
- Carrega vetores;
- Regula varredura;
- Mostra resultados/curva;
- Controle;
- Cardiobeep

O plano de excitação e o plano de amostragem e controle, tem sua execução iniciada pela geração de solicitação de interrupção através dos temporizadores T_0 e T_1 respectivamente. O temporidador T_0 opera em uma freqüência d*e* $400Hz$ e o temporizador T_1 em $60Hz$.

O plano de excitação tem prioridade sobre o plano de amostragem e controle. Se a plano de amostragem e controle estiver sendo executada e o temporizador T_0 gerar um interrupção para iniciar a execução do plano de Excitação, então o processador finalizará a tarefa que esta sendo executada, plano de amostragem e controle, e executará todas as atividades do plano excitação. Ao finalizar o plano de excitação, o processador retornará executar as tarefas do plano de amostragem e controle.

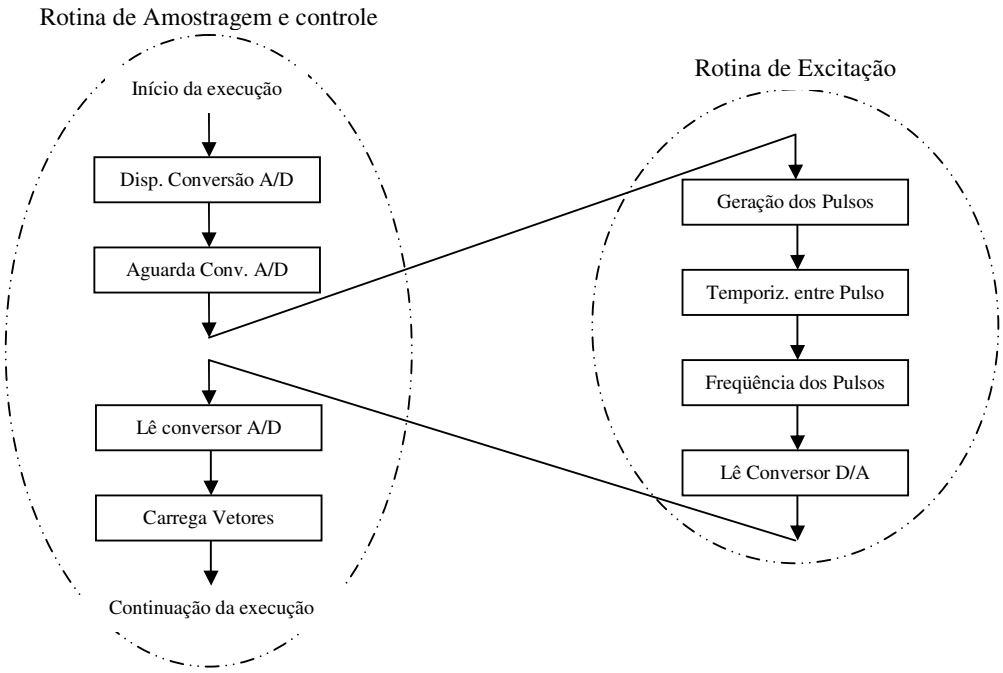


Figura 5.15:Plano de excitação interrompendo o plano de Amostragem e controle.

Na geração do modelo Power-GSPN do Oxímetro de Pulso, necessitou-se modelar os dois temporizadores. Cada temporizador é responsável pela ativação de um *plano*. Também foi modelado o compartilhamento do processador para execução das tarefas de acordo com as prioridades. A forma de modelagem do temporizador e do compartilhamento do processador foi abordada no Capítulo 4.

O objetivo deste estudo de caso é analisar e estimar consumo de energia, potência, desempenho e autonomia do oxímetro de pulso, variando-se os parâmetros de configurações e analisar a variação das estimativas, de forma a se obterem estimativas para diversas configurações possíveis.

Os parâmetros de configurações do oxímetro de pulso são:

- Freqüência de execução da tarefa de carga de vetores;
- Freqüência de execução da tarefa de controle;

Freqüência de execução tarefa de carga de vetores – define-se este parâmetro, como a número de capturas, por unidade de tempo (segundos), do resultado da conversão analógica para digital. Um aumento na freqüência desta tarefa, tende a uma maior precisão dos resultados, porém há um custo de potência associado que deve ser ponderado.

Freqüência de execução da tarefa de Controle – Associa-se a este parâmetro o número de vezes de execução da tarefa de controle por segundo. Quanto maior a freqüência desta tarefa, maior a precisão dos resultados do Oxímetro, por outro também há um custo de potência associado que deve ser ponderado.

Há uma outra variável que influencia o cálculo da estimativa de potência do modelo do Oxímetro, que é o número de batimentos cardíaco por minuto do paciente. A tarefa *cardiobip* é executada de acordo com o batimento cardíaco do paciente.

Freqüência de execução do Cardiobeep – A freqüência da tarefa *Cardiobeep* está ligada ao batimento cardíaco do paciente. Quando o paciente está com um batimento cardíaco mais elevado, aumenta-se proporcionalmente a execução da tarefa *Cardiobeep*.

Na Figura 5.16, ver-se nas extremidades da direita e esquerda os dois Temporizadores, que no modelo estão nomeados como *Temporizador 1* e *Temporizador 2.* São eles quem disparam a execução dos planos. Ao centro está o processador que é compartilhado pelos dois planos. Nota-se que o plano de excitação (à direita), quando executado, consome o recurso do processador, liberando-a após a execução de todas as tarefas. Diferentemente, no plano de amostragem e controle, cada tarefa após ser executada libera o processador. Desta forma, o plano de excitação pode interromper a execução do plano de amostragem e controle para ser executado. Como já foi dito antes, o plano de excitação tem prioridade sobre o plano de amostragem e controle.

No plano de excitação, as tarefas são seqüenciais e uma vez iniciadas a execução do plano, só finaliza quando são executadas todas as tarefas.

No plano de amostragem e controle, as três primeiras tarefas são seqüenciais, depois se tem um parâmetro de configuração, que é a tarefa de carregar vetores. Em seguida, mais duas tarefas seqüenciais e outro parâmetro de configuração, tarefa de *controle*. Por último vem a variável do *Cardiobeep*.

Para implementar a variação dos parâmetros, inseriu-se situações de conflito ao modelo. Variando o valor dos pesos das transições em conflito, altera-se a probabilidade de execução das tarefas conflitantes.

A Figura 5.16 mostra o modelo Power-GSPN do Oxímetro de Pulso. Ao centro, localiza-se um lugar com uma marca, representando o processador. Ao lado direito, tem-se o plano de excitação junto com o *Temporizador* T_1, mais a esquerda. Nota-se que, no plano de excitação (esquerda), quando executa a primeira tarefa, consome-se a marca no lugar que simboliza o processador, desabilitando a execução do plano de amostragem e controle. Devolve-se a marca do lugar, que representa o processador, apenas quando conclui a execução da última tarefa do plano de excitação.

Diferentemente do plano de excitação, as tarefas do plano de amostragem e controle, à medida que se executa cada tarefa, a marca é devolvida ao lugar *processador*, permanecendo o plano de excitação habilitado. Desta forma, a qualquer momento o plano de excitação pode interromper a execução do plano de amostragem e controle, uma vez que ele tem uma maior prioridade de execução.

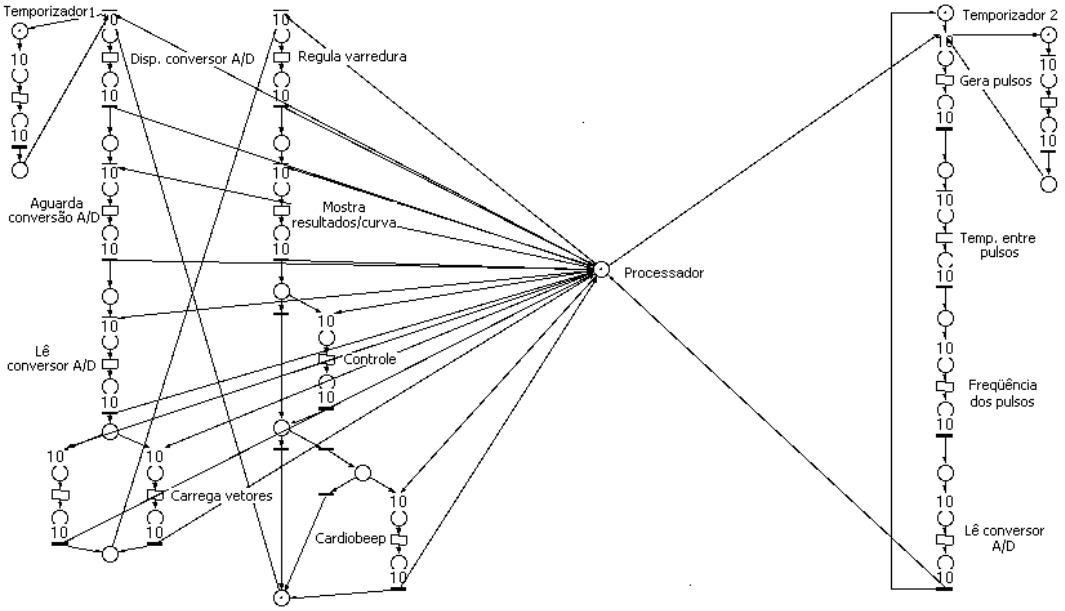


Figura 5.16:Modelo Power-GSPN do Oxímetro de Pulso.

Com o modelo Power-GSPN do Oxímetro de Pulso construído, deve-se atribuir às transições informações de tempo, prioridade e consumo de energia. As informações temporais são facilmente obtidas pela soma dos tempos de execução das instruções que compõe cada tarefa. As prioridades, neste modelo, são relevantes, dado que as tarefas podem ser executadas paralelamente existindo concorrência entre elas. Os valores de consumo de energia das tarefas foram obtidos segundo técnica mostrada no Capítulo 4.

Tabela 5.6:Informações de tempo, prioridade, consumo de energia das tarefas Oxímetro de pulso.

Tarefa		Tempo(μs)	Prioridade	Energia (ηJ)
1° Plano – Rotina de Excitação	Geração dos Pulsos	1	1	52,36
	Largura entre os Pulsos	164	1	8524,35
	Freqüência dos Pulsos	330	1	17379,9
	Leitura do Conversor A/D	5	1	222,23
2° Plano – Rotina de Amostragem e Controle	Conversão A/D	1	2	55,58
	Aguarda conversão	100	2	2841,45
	Lê conversor	15	2	760,96
	Carrega Vetores	12	2	429,76
	Regula Varredura	11	2	443,87
	Mostra resultados/curva	24	2	1117,35
	Controle	330	2	7088,97
	Cardiobeep	22	2	934,94

5.4.2 Estimativas de potência e desempenho

Tendo atribuído as informações relativas ao tempo, prioridade e consumo de energia às transições modelo, definem-se as métricas para obtenção das estimativas de potência e desempenho.

A partir da solução da análise estacionária, obtêm-se as probabilidades de habilitação das transições $PHT(t_i)$, para todo $t_i \in T$. Utilizando-se estas informações é possível calcular a taxa de execução das transições. Dado que determinadas transições representam a execução de uma tarefa (conjunto de instruções), é possível calcular sua taxa de execução.

$$Tx(t_i) = PHT(t_i) \times \lambda(t_i)$$

Onde :

$PHT(t_i)$ - representa a probabilidade de habilitação de uma transição t_i.

λ - é a taxa de disparo da transição exponencial, $\lambda = \frac{1}{D}$.

D – é o retardo (*delay*) de uma transição determinística.

Após o cálculo da taxa de execução $Tx(t_i)$ das transições, calcula-se a potência dissipada associada a cada transição, conforme equação:

$$P(t_i) = Tx(t_i) \times E(t_i)$$

Onde:

$E(t_i)$ - representa o consumo de energia para cada disparo de uma transição.

$P(t_i)$ – representa a potência dissipada associada a transição $t_i \in T$.

Dado que a execução das transições com suas características representam à execução das tarefas. Os resultados da análise de consumo de energia e desempenho do sistema podem ser obtidos. Os resultados desta análise são apresentados na Tabela 5.7.

Tabela 5.7:Métricas de estimativas de potência e desempenho das tarefa do Oxímetro de pulso

Tarefa		Tempo (μs)	Probabilidade	Energia (ηJ)	**Taxa de Execução (e/s)**	**Potência Dissipada**(mW)
1° Plano Excitação	Geração dos Pulsos	1	0,000400	52,36	**399,6**	**0,020779**
	Largura entre os Pulsos	164	0,065534	8524,35	**399,6**	**3,406210**
	Freqüência dos Pulsos	330	0,131868	17379,89	**399,6**	**6,945087**
	Leitura Conversor A/D	5	0,001998	222,23	**399,6**	**0,088712**
2° Plano Amostragem e Controle	Conversão A/D	1	0,00006	55,58	**59,81**	**0,003349**
	Aguarda conversão	100	0,005981	2841,45	**59,81**	**0,169922**
	Lê conversor	15	0,000897	760,96	**59,81**	**0,045516**
	Carrega Vetores	12	0,000638	429,76	**53,125**	**0,012691**
	Regula Varredura	11	0,000658	443,87	**59,81**	**0,026556**
	Mostra resultados/curva	24	0,001435	1117,35	**59,81**	**0,066808**
	Controle	330	0,009789	7088,97	**29,66**	**0,210332**
	CardioBeep	22	0,000603	934,94	**27,41**	**0,025649**

Tendo obtido a potência dissipada para cada tarefa, calcula-se a potência total dissipada pelo sistema, conforme a equação:

$$PS = \sum_{t_i \in T} P(t_i) \qquad \textit{Potência dissipada pelo Sistema - } PS = 11{,}02161\text{mW}$$

Para este estudo de caso, em princípio, não é relevante o cálculo do índice de desempenho geral do sistema, visto que se os dois planos tem uma taxa de execução fixa, definidas por seu respectivo temporizador.

Realizou-se o cálculo da autonomia, visto que este sistema geralmente está associado a um alimentador acumulativo, pois se trata de um equipamento portátil.

O oxímetro será alimentado por fonte química acumuladora de energia. A especificação do acumulador é:

- Tensão de 12V
- Capacidade 2000mAh

Para calcular a autonomia do sistema quando alimentado por este acumulador, é necessário calcular a energia armazenada na fonte (*EA)*, onde:

$$EA = \int_{t_i}^{t_f} (V \times I)dt \,. \qquad EA = 86.400J$$

Para o cálculo da autonomia do sistema, considerando-se esta fonte de alimentação, tem-se:

$$A = \frac{EA}{PS} \qquad A = \frac{86400}{11{,}02161} = 7839145s = 2.177{,}54h = 90{,}73dias$$

Para a fonte especificada, a autonomia estimada é de 90,73 dias ou 2.177,54 horas de uso.

5.4.3 Experimentos com variações de parâmetros

O resultado da avaliação apresentado na seção anterior foi obtido considerando-se os parâmetros com valores fixos. Foi atribuída uma probabilidade de 90% para tarefa de *carregar vetores,* 50% de probabilidade para as tarefas de *controle* e *Cardiobeep*. Nesta seção serão realizados experimentos onde, apenas um parâmetro pode ser variado por vez (de um valor mínimo até um valor máximo). Serão traçados gráficos com os resultados da avaliação em função da variação do parâmetro.

A freqüência de execução do plano de amostragem e controle é igual a 60 *Hz.* A princípio, as tarefas pertencentes a este plano teriam uma taxa de execução igual $60s^{-1}$. Através de escolhas, pode-se diminuir a freqüência de execução das tarefas contidas neste plano. De acordo com a probabilidade das escolhas, a taxa de execução de uma tarefa pode variar de zero, probabilidade zero (0%), até sessenta execuções por segundo ($60s^{-1}$), para uma probabilidade de 100% da freqüência do plano de amostragem e controle.

Para os três experimentos realizados no estudo de caso do Oxímetro de pulso, variam-se as probabilidades de 0% até 90% da freqüência do plano de amostragem e

controle, com um incremento de 30% a cada interação do experimento, totalizando quatro interações. Em termos de execuções por segundo, tem-se uma variação de 0 s^{-1} até 54s^{-1},com o incremento de 18 s^{-1}.

1° Experimento: Variar a probabilidade de execução da tarefa c*arrega vetores* .

Neste experimento varia-se a probabilidade de execução da tarefa c*arrega vetores* e verifica-se a variação da potência dissipada (PS).

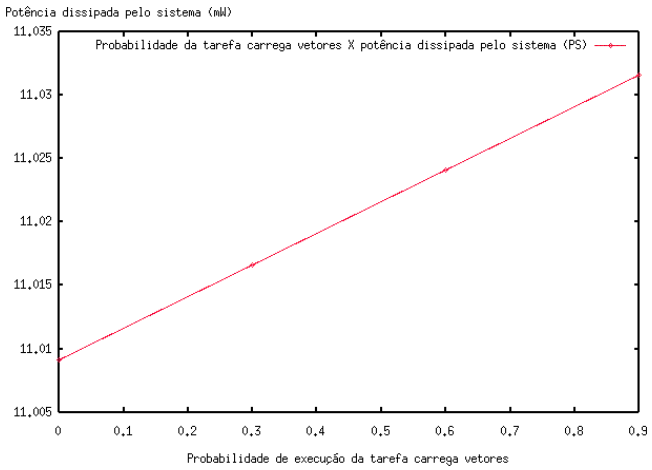


Figura 5.17:Variação da probabilidade da tarefa carregar vetores X Potência dissipada pelo sistema (PS)- (Oxímetro).

Na Figura 5.17 observa-se que quanto maior a probabilidade da execução da tarefa *carregar vetores*, maior é a potência dissipada. Este aumento está diretamente relacionado ao maior número de execuções tarefa de *carregar vetores*. O gráfico da Figura 5.17 é útil para o projetista analisar a variação do consumo por uma maior precisão do sistema que é dada através do aumento da taxa de execução da tarefa carrega vetores.

2° Experimento: Variar a probabilidade de execução da tarefa *controle.*

Alterando a probabilidade execução da tarefa *controle*, influencia a precisão das amostras capturadas pelos sensores, porém aumento da taxa de execução desta tarefa tem um custo de potência associado. A Figura 5.18 apresenta a potência dissipada total (*PS*) em função da probabilidade de execução da tarefa *controle*.

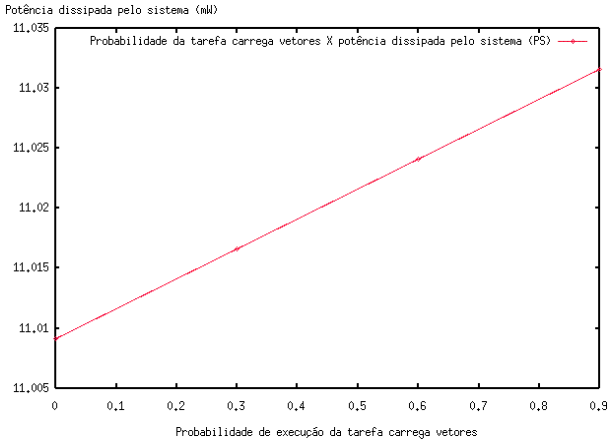


Figura 5.18:Variação da probabilidade da tarefa controle X Potência dissipada pelo sistema (PS)- (Oxímetro).

Na Figura 5.18, observa-se que quanto maior a probabilidade da execução da tarefa *controle*, maior é a potência dissipada. Este aumento está diretamente relacionado ao maior número de execuções tarefa de *controle*. Para uma menor potência dissipada, deve-se verificar qual a menor taxa de execução viável para a tarefa *controle*.

3° Experimento: Variar a probabilidade de execução tarefa *Cardiobeep.*

A freqüência da tarefa *Cardiobeep* está ligada ao batimento cardíaco do paciente. Com a variação do batimento cardíaco a Taxa de execução da tarefa *Cardiobeep* é incrementada.

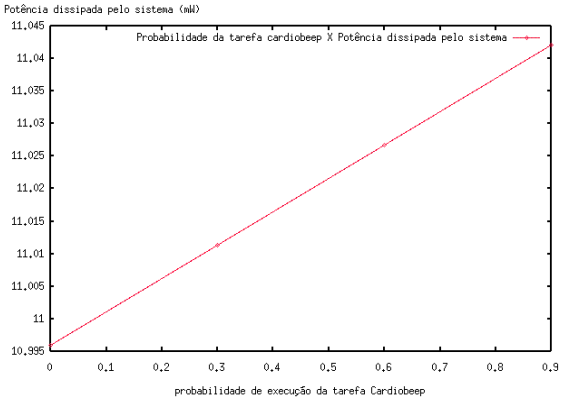


Figura 5.19:Variação da probabilidade da tarefa controle X Potência dissipada pelo sistema (PS) – (Oxímetro).

A Figura 5.19 mostra que quanto maior a probabilidade da execução de *Cardiobeep* maior é a potência dissipada pelo sistema. Este gráfico serve como referência da variação de potência dada a variação do batimento cardíaco.

5.5 Resumo

Neste capítulo realizaram-se três estudos de caso de sistemas do mundo real, a fim de aplicar e validar a metodologia desenvolvida. Foram apresentados todos os passos e tarefas necessárias para obtenção das estimativas.

Primeiramente mostrou-se como representar os sistemas em estudas em Power-GSPN. Em seguida, demonstrou-se a forma de obtenção das informações temporais e de consumo de energia para atribuição aos elementos do modelo. Na próxima etapa, foram definidas as métricas para obtenção das estimativas requeridas. Avaliou-se os modelos Power-GSPN em estado estacionário, onde as métricas foram calculadas e as estimativas obtidas.

Finalizando o Capítulo, foram feitos experimentos onde se variou os parâmetros de entrada do sistema, uma a um, através do uso da ferramenta *TimeNET 3.0.* Mostrou-se, em forma gráfica, a variação da potência dissipada e o desempenho do sistema pela variação do parâmetros de entrada.

Capítulo 6 - Conclusão e Trabalhos Futuros

Este capítulo traz um resumo do trabalho apresentado, mostrando como os objetivos propostos foram atingidos, exibindo também quais os problemas encontrados durante o desenvolvimento do mesmo. Em seguida, são apontadas as contribuições desta pesquisa para esta área de conhecimento de redes de Petri e projeto de sistemas de embarcados. Finalmente, são apontadas possibilidades de melhorias no trabalho e novas linhas de investigação para trabalhos futuros dentro do mesmo tema.

6.1 Conclusão

Este trabalho teve como objetivo principal propor uma metodologia de estimativas de consumo de energia através do uso das redes de Petri estocásticas generalizadas (GSPN). Para obter-se as estimativas, faz-se necessário a elaboração e definição de métricas de consumo de energia e desempenho. A metodologia proposta insere-se no contexto de projetos de sistemas embarcados, principalmente quando se tem restrições quanto ao consumo de energia.

Para atingir os objetivos esperados, fez-se necessário a criação de uma extensão do modelo GSPN, que foi chamado de Power-GSPN. De forma a mapear funções de consumo de energia, definiram-se duas funções de consumo: a primeira delas foi consumo de energia que está associado à mudança de estados do sistema, denominado de *consumo de energia dinâmico*. A segunda função de mapeamento do consumo de energia está relacionada à permanência do sistema no estado atual, isto é, quando

nenhuma operação ou tarefa é realizada. A esta segunda função de consumo, denominou-se *consumo de energia estático.* Mostrou-se que, neste trabalho, foi considerado apenas o consumo de energia dinâmico, tendo em vista que as tecnologias atuais têm um baixo consumo de energia estático, quando comparado ao consumo dinâmico, o qual pode ser desprezado.

Para obter-se as estimativas de consumo e desempenho de sistemas, mostrou-se que primeiramente deve-se representar o sistema em um modelo Power-GSPN. Mostrar-se diversas formas de modelagem para obtenção do modelo em Power-GSPN.

Foram apresentadas técnicas de medição de consumo de energia das atividades realizadas pelo sistema, pois se faz necessário a atribuição de informações temporais e de consumo de energia aos elementos do modelo Power-GSPN. Em seguida, foram definidas as métricas de consumo de energia e desempenho.

Para calcular as métricas definidas, apresentaram-se as formas de avaliação do modelo em estado transiente e em estado estacionário.

A fim de aplicar e validar a metodologia desenvolvida, três estudos de caso foram apresentados. A avaliação dos estudos de casos realizados ilustra a metodologia proposta. A partir de um código ou uma de especificação, gerou-se um modelo Power-GSPN representando o sistema. Posteriormente, fez-se o levantamento das informações temporais e de consumo de energia do referido sistema. Atribuíram-se aos elementos do modelo as informações temporais e de consumo de energia levantadas. Aplicaram-se as métricas definidas e avaliou-se o sistema em estado estacionário, obtendo-se as estimativas requeridas. Através da ferramenta *TimeNET* 3.0, vários experimentos foram realizados a fim de contemplar as variações de entrada dos sistemas. Para uma melhor visualização dos resultados, geraram-se gráficos dos experimentos realizados. Os gráficos mostram a variação das estimativas de acordo com a variação dos diversos parâmetros dos sistemas modelados.

O trabalho desenvolvido é de grande valia tanto para os profissionais de projetos de sistemas embarcados, quanto para a comunidade científica pela abordagem

estocástica utilizada. Para os profissionais, a metodologia facilita o encontro de uma das soluções viáveis para a implementação do projeto, ou então valida as soluções encontradas. O formalismo matemático suportado pelas redes de Petri é de grande importância, pois é possível avaliar o modelo em regime transiente e estacionário, permitindo ainda a simulação.

6.2 Trabalhos Futuros

Como para trabalho futuro é importante considerar a função que mapeia o consumo de energia estático, isto é, o consumo dado pela permanência no estado quando o sistema não executa nenhuma operação. Esta função não foi considerada neste trabalho porque o consumo de energia estático e muito pequeno em relação ao consumo de energia dinâmico, porém para uma maior precisão dos resultados o consumo de energia estático deve ser computado.

Outro trabalho futuro importante a ser desenvolvido, é o aperfeiçoamento das técnicas de estimativas de consumo. Novos modelos podem ser definidos, inclusive para abordagem não estocástica, ou com outra semântica de disparo das ações. O modelo de redes de Petri temporizado (Timed Petri Net - TPN) é interessante pelas características determinística e pelo uso da semântica de passo. Onde teria espaço para a criação de uma ferramenta de auxílio para dar suporte as rede de Petri escritas em TPN.

Referências Bibliográficas

[1] Holger Hermanns, Joost-Pieter Katoen, *Lectures on Formal Methods and Performance Analysis*, First EEF/Euro Summer School on Trends in Computer. Scienc Berg en Dal, The Netherlnds, July 3-7, 2000 Revised Lectures.

[2] Boudewijn R. Haverkort, Markovian *Models for Performance and Dependability Evaluation,Laboratory for Performance Evaluation and Distributed Systems*, Department of Computer Science, RWTH Aachen, 52056 Aachen, Germany

[3] B.R. Haverkort and K.S. Trivedi. *Specification and generation of Markov reward models*. *Discrete-Event Dynamic Systems: Theory and Applications*, 3:219–247,1993.

[4] Albano Portela Machado,*Uma metodologia para estimativa de área baseada em redes de petri temporizadas para ambientes de sistemas de hardware/software co-design*. Dissertação de Mestrado, Universidade Federal de Pernanbuco, Recife, PE, Brasil, Dez. Agosto, 2004.

[5] Maciel, P. *Introdução as redes de Petri e aplicações*. UNICAMP, Campinas 1996.

[6] Maciel, P. *Petri Net Based Estimators for Hardware/Software Co-Design*. PhD thesis, Universidade Federal de Pernanbuco, Recife, PE, Brasil, Dez. 1999.

[7] Mauro José Carlos e Silva, *Modelagem de Políticas de Estoques: uma abordagem baseada em Redes de Petri*. Dissertação de Mestrado, Universidade Federal de Pernanbuco, Recife, PE, Brasil, Dez. Novembro, 2004.

[8] Julio Alexandrino de Oliveira Filho, *Geração automática de interface para incorporação de IP-CORE em ambientes SoC. Dissertação de Mestrado*, Universidade Federal de Pernanbuco, Recife, PE, Brasil, Dez. Março, 2003.

[9] M. Ajmone Marsan ,Gianfranco Balbo, Gianni Conte, Susanna Donateili, Giuliana Franceschinis, *Modelling With Generalised Stochastic Petri Nets*, Università degli studi di Tonno Dipartimento di Informatica.

[10] Gary K. Yeap, *Pratical Low Power Digital VLSI Design*, KAP

[11] Choi, K. e Chatterjee, A. (2001). "*Efficient Instruction-Level Optimization Methodology for Low-Power Embedded Systems*". International Symposium on System Synthesis, Montreal, Canada, Out. 2001. Proceedings, ACM, 2001. pp 147-152.

[12] Chen, R., Irwin, M.J. e Bajwa, R. (2001). "*Architecture-Level Power Estimation and Design Experiments*". ACM Transactions on Design Automation of Electronic Systems, Vol. 6, No. 1, Jan. 2001. pp 50-66.

[13] Zhang, Y., Hu, X. e Chen, D. (2002). "*Task Scheduling and Voltage Selection for Energy Minimization*". DAC'2002 – Design Automation Conference, New Orleans, EUA, Jun. 2002. Proceedings, ACM Press, 2002.

[14] Landman, P. e Rabaey, J. (1996). "*Activity-Sensistive Architectural Power Analysis*". IEEE Transactions on CAD of Integrated Circuits, Vol. 15, No. 6, Jun. 1996, pp 571-587

[15] Beck Filho, A.C., Wagner, F.R. e Carro, L. (2003). "*CACO-PS: A General Purpose Cycle-Accurate Compiled-Code Power Simulator*". 16th Symposium on Integrated Circuits and Systems Design. São Paulo, Brasil, Set. 2003. Proceedings, IEEE Computer Society Press, 2003.

[16] Tiwari, V., Malik, S. e Wolfe, A. (1994). "*Power Analysis of Embedded Software: a First Step Towards Software Power Minimization*". IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol. 2, No. 4, Abr. 1994, pp 437-445.

[17] Marco Ajmone Marsan, Gianni Conte, Gianfranco Balbo , *A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems*, Politecnico di Torino, Universita' di Torino, Turin, Italy.

[18] Manish Malhotra, Andrew Reibman, *Selecting and implementing phase approximations for semi-markov models*, AT&T Beil Laboratories HolmdelDept. of Computer Science Box 90129, Duke University Durham

[19] Dalal, V. e Ravikumar, C.P. (2001). "*Software Power Optimizations in an Embedded System*". VLSI Design Conference, Bangalore, India, Jan. 2001. Proceedings, IEEE Computer Science Press. 2001, pp 254-259.

[20] G. Ciardo. *Discrete-time Markovian stochastic Petri nets*. In W. Stewart, editor, *Numerical Solution of Markov Chains*, pages 339–358, Raleigh, NC, 1995. Kluwer.

[21] Gervini, A.I., Corrêa, E.F., Carro, L. e Wagner, F.R. (2003). "*Avaliação de Desempenho, Área e Potência de Mecanismos de Comunicação em Sistemas Embarcados*". SEMISH'2003 – XXX Seminário Integrado de Software e Hardware.Campinas, Ago. 2003. Anais, SBC, 2003.

[22] Christos G. Cassandras, Stéphane Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers.

[23] Chen, R., Irwin, M.J. e Bajwa, R. (2001). "*Architecture-Level Power Estimation and Design Experiments*". ACM Transactions on Design Automation of Electronic Systems, Vol. 6, No. 1, Jan. 2001. pp 50-66.

[24] T. Murata, "*Petri nets: Properties, analysis and applications*," *Proceedings of the IEEE,* vol. *77,* pp. 541 580, April 1989.

[25] J. Desel and J. Esparza, *Free Choice Petri Nets.* Cambridge University Press, 1995. Cambridge, Great Britain.

[26] T. S.F.L. Fernandes, W.J. Silva, M. Silva, N.S. Rosa, P.R.M. Maciel, and D.F.H. Sadok. *On the generalised stochastic petri net modeling of message-oriented middleware systems.* In: 23rd IEEE International performance Computing and Communications Conference - International Workshop on Middleware Performance (IWMP 2004), 1:783{788, 2004.

[27] J. Desel, *Lectures on Petri Nets 1, Basic Modeis,* vol. 1491 of *Lecture Notes In Computer Science,* ch. Basic Linear Algebraic Techniques for Place/Transition Nets, pp. 257 308. Springer Verlag, 1998.

[28] Gajski, D. D., Varid, F., Narayan, S., and Gong, J. *Specification and Design of Embedded Systems.* PTR Prentice Hall. 1994.

[29] E. Best, "Structural theory of petri nets: The free choice hiatus," *Lecture Notes in Computer Science,* vol. 254, pp. 168 206, 1987. Springer-Verlag.

[30] A. Yakovlev and A. M. Koelmans, *Petri Nets and Digital Hardware Design,* vol. 1492 of *Lecture notes in Computer Science.* Springer-Verlag, 1998.

[31] Machado, A. P., Maciel, P.and Guilhermino, A. "*A Petri net Based Method for Storage Units Estimation*". *In Proc. of International Conference on Systems, Man and Cybernetics*, pages 1025-1031, vol. 2. Washington, D.C., USA, Oct. 5-8 2003. SMC Society, IEEE.

[32] Maciel, P. *Petri Net Based Estimators for Hardware/Software Co-Design*. PhD thesis, Universidade Federal de Pernanbuco, Recife, PE, Brasil, Dez. 1999.

[33] Maciel, P., Barros, E., Lima, M., Silva, D. and Rosenstiel, W. "*Resource Sharing Estimation by Petri Nets in PISH Co-Design System*". In *Proc. of High Performance Computing 2000*, Washington D.C., EUA, April, 2000.

[34] Maciel, P., Barros, E., and Rosenstiel, W. "*A Petri Net Based Approach for estimating Area in Hardware/Software Co-Design*". In *Proc. of High Performance Computing '99*, San Diego, California, USA, April 10-14, 1999.

[35] Maciel, P., Barros, E., and Rosenstiel, W. "*A Petri Model for Hardware/Software Co-Design*". In *Design Automation for Embedded Systems*, 4: 243-310, 1999.

[36] Maciel, P., Barros, E., and Rosenstiel, W. "*Computing Comunication Cost by Petri Nets for Hardware/Software Co-Design*". In *Proc. of 8th IEEE International Workshop on Rapid System Prototyping*, Chapel Hill, NC, EUA, Jun. 24-26, 1997.

[37] Maciel, P., Barros, E., Silva, M., and Filho, F. Cruz. "*Resource Sharing Estimation by Petri Nets in PISH Hardware/Software Co-Design System*". In *Proc. of International Conference on Systems, Man and Cybernetics*, pages 3165-3170, Nashville, TN, EUA, October 8-11, 2000. SMC Society, IEEE.

[38] Maciel, P., Filho, F. Cruz, and Barros, E. "*Estimates Based on Petri Nets for Hardware/Software Co-Design*". In *Proc. of 4th Portuguese Conference on Automatic Control*, pages 546-551, Guimarães, Portugal, October 4-6, 2000. APCA, IFAC.

[39] Dalal, V. e Ravikumar, C.P. (2001). "*Software Power Optimizations in an Embedded System*". VLSI Design Conference, Bangalore, India, Jan. 2001. Proceedings, IEEE Computer Science Press. 2001, pp 254-259

[40] Edwards, S., Lavagno, L., Lee, E.A. e Sangiovanni-Vincentelli, A. (1997). "*Design of Embedded Systems: Formal Models,Validation, and Synthesis*". Proceedings of the IEEE, Vol. 85, No. 3, Mar. 1997. pp 366-390

[41] Ishihara, T. e Yasuura, H. (2000). "*A Power Reduction Technique with Object Code Merging for Application Specific Embedded Processors*". DATE'00 – Design, Automation and Test in Europe, Paris, França, Mar. 2000. Proceedings, IEEE Computer Society Press, 2000.

[42] Stravers, P. *Embebed System Design*, 1994. Sait, S. e Habib, Y.

[43] Giallfrallco Ciardo, Jogesh Muppala, Kishor S. Trivedi, *On the Solution of GSPN Reward Modeis,* Software Productivity Consortium Herndon, VA 22070, USA, Duke University Durham, NC 27706, USA, August 21, 1996

[44] *VLSI Physical Design Automation: Theory and Practice*, 1997.

[45] Reisig, W. *Petri Nets: An Introduction*. Springer-Verlag, 1982.

[46] Murata, T. "*State Equation, Controllability, and Maximal of Petri Nets". IEEE Trans. on Automatic Control*, 1977.

[47] Márcio Augusto de Souza, *Análise de Técnicas de Monitoração e Métricas de Desempenho Para a Avaliação do Escalonamento de Processos,* São Carlos – SP 2001

[48] Eduardo Tavares, Raimundo Barreto, Meuse Oliveira Júnior, Paulo Maciel, Marília Neves,Ricardo Lima, *An Approach for Pre-Runtime Scheduling in Embedded Hard Real-Time Systems with Power Constraints,*

[49] Fernando F. Carvalho, Meuse N. Oliveira Júnior, Paulo R Martins Maciel, Raimundo S. Barreto *Towards A Software Power Cost Analysis Framework Using Colored Petri Net,* Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife-PE-Brazil.

[50] Meuse N. Oliveira J´unior, Paulo Maciel, Raimundo Barreto and Fernando Carvalho *A Software Power Cost Analysis Based on Colored Petri Net***,** Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife-PE-Brazil.

[51] Sangiovanni-Vincentelli, A. e Martin, G. (2001). "*Platform-Based Design and Software Design Methodology for Embedded Systems*". IEEE Design & Test of Computers, Vol. 18, No. 6, Nov/Dez. 2001. pp 23-33.

[52] Holger Hermanns**,** *Process Algebra and Markov Chains,Formal Methods and Tools Group*, Faculty of Computer ScienceUniversity of Twente, AE Enschede, The Netherlands

[53] James F. Watson III, Adan A. Desrochers, *Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Nonexponential Transition Functions,* IFPE Transactions On Systems. Man. And Cybernetics VOL 21, N° 5, September/October 1991

[54] Alexandre I. Gervini1, Edgard de F. Corrêa1,2, Luigi Carro3, Flávio R. Wagner1, *Avaliação de Desempenho, Área e Potência de Mecanismos de Comunicação em Sistemas Embarcados*.

[55] Luigi Carro, Flávio Rech Wagner, *Sistemas Computacionais Embarcados.*

[56] Stravers, P. *Embebed System Design*, 1994.

[57] Sait, S. e Habib, Y. *VLSI Physical Design Automation: Theory and Practice*, 1997.

[58] ITRS (2001). International Technology Roadmap for Semiconductors, versão 2001. Disponível em http://public.itrs.net/

[59] Ulrich Herzog,, *Formal Methods for Performance Evaluation,* Universität Erlangem-Nürnberg, Institut für informatik, Germany.

[60] W. J. Stewart. *Introduction to the numerical solution of Markov Chains.* Princeton University Press, 1994.

[61] W.K. Grassmann. *Finding transient solutions in Markovian event systems through randomization*. In W.J. Stewart, editor, *Numerical Solution of Markov Chains*,pages 357–371. Marcel Dekker, 1991.

[62] D. Gross and D.R. Miller. *The randomization technique as a modeling tool and solution procedure for transient Markov processes*. *Operations Research*, 32(2):343–361, 1984.

[63] A. Jensen. *Markov chains as an aid in the study of Markov processes*. *Skand. Aktuarietidskrift*, 3:87–91, 1953.

[64] N. Kavvadias S. Nikolaidis and P. Neofotistos. *Instruction-level power measurement methodology*. Technical report, Electronics Lab., Physics Dept., Aristotle University of Thessaloniki, Greece, March 2002.

[65] TimeNet 3.0 tool. More info avalible in http://pdv.cs.tu-berlin.de/~timenet/overv.html

[66] F. Vahid T. Givargis and J. Henkel. *Instruction-based system-level power evaluation of system-on-chip peripheral cores*. *IEEE Trans. on Very Large Scale Integration Systems*, 10(6):856–863, Dec 2002.

[67] Meuse Nogueira de Oliveira Junior, "*Desenvolvimento de um protótipo para a medida não invasiva da saturação arterial de oxigênio em humanos – Oxímetro de pulso*", Departamento de Biofísica e Rádio Biologia,

[68] Bergamaschi, R.A. et alii (2001). "*Automating the Design of SOCs Using Cores*". IEEE Design & Test of Computers, Vol. 18, No. 5, Set/Out. 2001. pp 32-45.