



PEDRO CARLOS FAZOLINO PEPE

**“ESCALONAMENTO DINÂMICO DE TENSÃO E FREQUÊNCIA EM
MULTIPROCESSADORES PARA APLICAÇÕES COM ESPECIFICAÇÃO
DE QUALIDADE POR TAXA MÍNIMA DE PROCESSAMENTO DE
ENTRADAS”**

CAMPINAS

2012



UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

PEDRO CARLOS FAZOLINO PEPE

**“ESCALONAMENTO DINÂMICO DE TENSÃO E FREQUÊNCIA EM
MULTIPROCESSADORES PARA APLICAÇÕES COM ESPECIFICAÇÃO
DE QUALIDADE POR TAXA MÍNIMA DE PROCESSAMENTO DE
ENTRADAS”**

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação da UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

ORIENTADORA: PROFA. DRA. ALICE MARIA BASTOS HUBINGER TOKARNIA

Este exemplar corresponde à versão final da Dissertação defendida pelo aluno, e orientada pela Professora Alice Maria Bastos Hubinger Tokarnia

CAMPINAS

2012

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

P391e Pepe, Pedro Carlos Fazolino, 1978-
Escalonamento dinâmico de tensão e frequência em multiprocessadores para aplicações com especificação de qualidade por taxa mínima de processamento de entradas / Pedro Carlos Fazolino Pepe. --Campinas, SP: [s.n.], 2012.

Orientador: Alice Maria Bastos Hubinger Tokarnia.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas embutidos de computação. 2. Multiprocessadores. 3. Multimídia. 4. Consumo de energia. 5. Algoritmos heurísticos. I. Tokarnia, Alice Maria Bastos Hubinger, 1958-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Dynamic voltage and frequency scaling for multiprocessor embedded applications with soft delay deadlines

Palavras-chave em Inglês: Embedded computing systems, Multiprocessors, Multimedia, Energy consumption, Heuristic algorithms

Área de concentração: Engenharia de Computação

Titulação: Mestre em Engenharia Elétrica

Banca examinadora: Eleri Cardozo, Edson Borin

Data da defesa: 17-09-2012

Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Pedro Carlos Fazolino Pepe

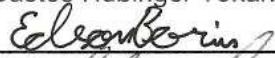
Data da Defesa: 17 de setembro de 2012

Título da Tese: "Escalonamento Dinâmico de Tensão e Frequência em Multiprocessadores para Aplicações com Especificação de Qualidade por Taxa Mínima de Processamento de Entradas"

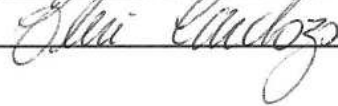
Profa. Dra. Alice Maria Bastos Hubinger Tokarnia (Presidente):



Prof. Dr. Edson Borin:



Prof. Dr. Eleri Cardozo:



Agradecimentos

Gostaria de agradecer e dedicar este trabalho à minha esposa Josieni, que mesmo enfrentando todas as dificuldades que a vida lhe impôs, manteve-se forte e me apoiou em todas as decisões, acertadas ou equivocadas, que me trouxeram até aqui. Agradeço pelos plenos momentos de cumplicidade e felicidade que dividimos durante todos os anos em que pude estar ao seu lado. Deus levou para ele a melhor companheira que um homem pode ter.

Aos meus pais, Vilma e Pedro, que sempre me apoiaram em meus estudos, me dando o suporte e exemplo de caráter e dignidade que qualquer jovem precisa para enfrentar de cabeça erguida os desafios da vida.

Ao meu irmão, Peterson, pelos ótimos momentos de convivência e crescimento que dividimos.

À minha família e à família da Josieni, que nunca faltaram nos momentos mais difíceis, e que sempre pude ter ao meu lado para dividir minhas alegrias e conquistas.

À minha orientadora e amiga, Professora Alice, que acreditou em mim e teve a enorme paciência de desenvolver comigo este trabalho, dividindo seus conhecimentos e aumentando os meus.

Aos meus grandes amigos e aos não tão grandes assim, que de um jeito ou de outro estiveram presentes em todos os momentos da minha vida.

À Faculdade de Engenharia Elétrica e Computação da UNICAMP, por me deixar ser por tanto tempo um “estudante”.

Agradeço a Deus, por ter me dado confiança e perseverança para concluir esta dissertação, combinando minha vida acadêmica, pessoal e profissional, e por me dar forças para continuar lutando por meus ideais.

Pedro Pepe

Resumo

Este trabalho apresenta quatro algoritmos de escalonamento dinâmico de Tensão e Frequência (DVFS) em sistemas multiprocessador baseado em caminhos de execução. Nossos alvos são aplicações multimídia executadas em sistemas embarcados, com especificação de qualidade por taxa mínima de entradas (QoS) processadas. Uma fração mínima de entradas, geralmente quadros de dados, precisa ser completamente processada no tempo máximo de resposta especificado.

O objetivo dos algoritmos é atuar em quatro cenários que correspondem a sistemas com diferentes possibilidades de escalonamento dinâmico de tensão e frequência e diferentes capacidades de monitoramento da qualidade de serviço.

No primeiro cenário, todos os pacotes de dados de entrada recebidos devem ser processados dentro do tempo máximo especificado e o nível de tensão/frequência pode ser ajustado no início da execução da aplicação, sendo o mesmo para todos os processadores. Este cenário é referência para comparação de resultados para os outros cenários.

Para o segundo cenário, o nível de tensão/frequência pode ser definido individualmente para um processador, no início da execução de cada tarefa, e dados de entrada de classes específicas podem ser descartados.

O terceiro cenário possibilita, além do descarte de classes específicas de dados de entrada, o ajuste do nível de tensão/frequência de cada tarefa de acordo com a classe de dados de entrada a ser processada.

O algoritmo desenvolvido para o quarto cenário trata dinamicamente de alterações na distribuição probabilística das classes de entrada, calculando novos níveis de tensão/frequência para as tarefas e classes de entrada de modo que a especificação de qualidade continue a ser satisfeita, de forma eficiente.

Para uma aplicação de cancelamento de eco acústico, executada em 4 processadores, com taxa mínima de processamento igual a 50%, o algoritmo de escalonamento de tensão e frequência, no cenário 3, conseguiu reduzir o consumo de energia em cerca de 71%, comparado ao cenário 1. No cenário 4, simulamos para esta aplicação uma modificação simultânea de 10 pontos percentuais na distribuição das classes de entrada em 3 tarefas causando aumentos do número de descartes. O algoritmo proposto para o cenário 4 manteve a qualidade mínima com um aumento de apenas 6% no consumo de energia, quando comparado ao consumo de energia da configuração inicial definida para o cenário 3.

Abstract

This work presents four execution-path based Dynamic Voltage/Frequency Scaling (DVFS) algorithms for multiprocessor systems. The targets are embedded systems multimedia applications, with minimum input data completion rate specification (QoS). A minimum fraction of input data, usually data frames, should be processed within the specified deadline.

These algorithms aim to operate in four scenarios corresponding to systems with different possibilities of dynamic voltage and frequency scheduling and different QoS monitoring capabilities. In the first scenario, all received data frames should be treated within the deadline and the voltage/frequency operational level can be adjusted at the beginning of the application execution, and must be the same for all processors. This scenario is a reference for comparison of results obtained for the other scenarios. For the second scenario, the voltage/frequency operational level can be set individually for each processor at the beginning of each task execution, and input data frames of specific input classes can be discarded. The third scenario allows, besides discarding specific classes of input data, it is possible to adjust the operation level for each task, according to the class of the input data to be treated. The algorithm for the fourth scenario operates online, computing new voltage/frequency levels and making new decisions about class discarding to cope with changes in probability distribution of input classes. Its goal is to maintain the specified quality with low energy consumption.

In an application of acoustic echo cancellation running on a system with 4 processors, with a rate of inputs completely processed specified as 50%, the algorithm for scenario 3 achieved a reduction in consumption close to 71%, comparing to the results for scenario 1. During simulation, this application has been subjected to simultaneous changes of 10% in the input class distributions of three discarding tasks, reducing system quality. The algorithm for scenario 4, maintained the minimum quality with just 6% increase in power consumption, when compared to the consumption of the initial configuration for scenario 3.

Sumário

LISTA DE FIGURAS	xv
LISTA DE TABELAS.....	xvii
LISTA DE ACRÔNIMOS.....	xix
PUBLICAÇÕES RELACIONADAS A ESTE TRABALHO.....	xxi
1 INTRODUÇÃO	1
2 CONCEITOS BÁSICOS	5
2.1 INTRODUÇÃO	5
2.2 PROCESSAMENTO DE SINAIS	5
2.3 MULTIPROCESSADORES	6
2.4 FONTES DE ALIMENTAÇÃO EM DISPOSITIVOS EMBARCADOS	7
2.4.1 <i>Tensão de Polarização da Fonte de Alimentação</i>	7
2.4.2 <i>Tensão de Operação</i>	8
2.4.3 <i>Tensão de Operação VS Frequência de Operação</i>	11
2.5 ESCALONAMENTO DINÂMICO DE TENSÃO E FREQUÊNCIA (DVFS).....	12
2.6 GERENCIAMENTO DINÂMICO DE POTÊNCIA (DPM).....	13
2.7 CONCLUSÃO.....	13
3 TRABALHOS CORRELATOS.....	15
3.1 INTRODUÇÃO	15
3.2 TRABALHOS ANTERIORES	15
3.2.1 <i>Importância do Consumo de Energia do Processador</i>	15
3.2.2 <i>Técnicas de Escalonamento de Tensão/Frequência</i>	16
3.2.3 <i>Técnicas Ortogonais</i>	19
3.2.4 <i>Escalonamento de Tarefas</i>	21
3.3 CONCLUSÃO.....	22
4 ESPECIFICAÇÃO DO SISTEMA	23
4.1 INTRODUÇÃO	23
4.2 CARACTERÍSTICAS DO SISTEMA	23
4.3 CENÁRIOS DE OPERAÇÃO E CONFIGURAÇÕES	26
4.3.1 <i>C1 - Escalonamento de tensão/frequência da Aplicação</i>	26
4.3.2 <i>C2 - Escalonamento de tensão/frequência por Tarefa e Descarte de classes de entrada</i>	28
4.3.3 <i>C3 - Escalonamento de tensão/frequência por Classe e Descarte de classes de entrada</i>	29
4.3.4 <i>C4 - Monitoramento da distribuição das Classes de Entrada</i>	30
4.4 CONCLUSÃO.....	30
5 DESCRIÇÃO DOS ALGORITMOS.....	31
5.1 INTRODUÇÃO	31
5.2 DESCRIÇÃO DOS ALGORITMOS	31
5.3 ALGORITMO SCN1	32
5.4 ALGORITMO SCN2	33
5.4.1 <i>Configuração Inicial</i>	33
5.4.2 <i>Condição de descarte</i>	35
5.4.3 <i>Identificação dos caminhos de execução</i>	35
5.4.4 <i>Descarte de classes de entrada</i>	35
5.4.5 <i>Modificação do nível Tensão/Frequência de Operação</i>	36
5.4.6 <i>Redução da Qualidade de Serviço</i>	37
5.4.7 <i>Resultados</i>	38
5.5 ALGORITMO SCN3	38
5.6 ALGORITMO SCN4	39
5.6.1 <i>Etapa Offline SCN4</i>	40
5.6.2 <i>Etapa Online SCN4</i>	42
5.7 ALTERAÇÕES PARA EXECUÇÃO DOS ALGORITMOS	45
5.7.1 <i>SCN2</i>	46

5.7.2	SCN3.....	46
5.7.3	SCN4.....	46
5.7.4	<i>Passo a Passo</i>	47
5.8	COMPLEXIDADE	47
5.8.1	<i>Complexidade do algoritmo SCN1</i>	47
5.8.2	<i>Complexidade dos algoritmos SCN2 e SCN3</i>	48
5.8.3	<i>Complexidade do algoritmo SCN4</i>	48
5.9	EXEMPLO	48
5.9.1	SCN1	48
5.9.2	SCN2	50
5.9.3	SCN3	56
5.10	CONCLUSÃO	57
6	RESULTADOS.....	59
6.1	INTRODUÇÃO	59
6.2	DESENVOLVIMENTO	59
6.3	APLICAÇÕES	60
6.3.1	<i>Cancelador de Eco Acústico</i>	62
6.3.2	<i>Sonar</i>	63
6.3.3	<i>Vocoder GSM</i>	64
6.3.4	<i>TGFF</i>	65
6.4	RESULTADOS EXPERIMENTAIS	66
6.4.1	<i>Resultados SCN2 e SCN3</i>	66
6.4.2	<i>Resultados SCN4</i>	68
6.5	CONCLUSÃO.....	75
7	CONCLUSÃO	77
7.1	CONTRIBUIÇÕES	77
7.2	TRABALHOS FUTUROS	78
8	REFERÊNCIAS.....	81
APÊNDICE A1: APLICAÇÕES.....		85
A1.1	CANCELADOR DE ECO ACÚSTICO.....	85
A1.2	SONAR.....	91
A1.3	VOCODER	98
A1.4	TGFF70	106
A1.5	TGFF99	111
APÊNDICE A2: FERRAMENTA DE TESTE DOS ALGORITMOS.....		119
A2.1	ENTRADA DE DADOS.....	119
A2.2	EXECUÇÃO <i>SCN1, SCN2 e SCN3</i>	120
A2.3	EXECUÇÃO <i>SCN4</i>	123
APÊNDICE A3: TABELAS DE RESULTADOS		129
A3.1	RESULTADOS <i>SCN2 e SCN3</i>	129
A3.2	RESULTADOS <i>SCN4</i>	131
A3.2.1	<i>Vocoder</i>	131
A3.2.2	<i>Cancelador de Eco Acústico</i>	143
A3.2.3	<i>TGFF</i>	146

Lista de Figuras

Figura 2.1: Aumento de consumo devido à corrente de polarização da fonte [9].	8
Figura 2.2: Esquema simplificado de um dispositivo conectado a uma bateria.	9
Figura 4.1: Exemplo de especificação do sistema com (a) tempo de execução, energia e distribuição das classes de entrada, (b) grafo de tarefas e (c) grafo escalonado de tarefas .	24
Figura 5.1: Pseudocódigo para o algoritmo <i>SCN2</i> .	34
Figura 5.2: Pseudocódigo complementar do algoritmo <i>SCN3</i> .	39
Figura 5.3: Pseudocódigo para o algoritmo <i>offline SCN4</i> .	41
Figura 5.4: Pseudocódigo para o algoritmo <i>online SCN4</i> .	44
Figura 5.5: Caminhos de execução para a aplicação .	49
Figura 5.6: Descarte de classes de entrada. Inicialmente u_7 , depois u_5 .	52
Figura 5.7: Aumento do nível de operação. Inicialmente u_1 , depois u_2 e u_4 .	54
Figura 5.8: Alteração do nível de operação da classe $c_{4,1}$.	56
Figura 6.1: Fluxo para algoritmos <i>offline</i> .	60
Figura 6.2: Consumo médio de energia por entrada em relação a <i>SCN1</i> .	67
Figura 6.3: Utilização Média do Processador, relativo a <i>SCN1</i> .	68
Figura 6.4: Consumo médio de energia por entrada para <i>SCN4</i> em relação a <i>SCN3</i> , como função do δ , para o Vocoder-2, $Q = 0,5$. .	69
Figura 6.5: Utilização dos processadores, em relação a <i>SCN3</i> , em função do δ , para Vocoder-2, $Q = 0,5$. .	70
Figura 6.6: Qualidade de serviço efetiva para <i>SCN4</i> , como função do δ , para Vocoder-2, $Q = 0,5$. .	71
Figura 6.7: Consumo Médio de Energia por entrada para <i>SCN4</i> em relação a <i>SCN3</i> , como função do δ , para o Vocoder-2, $Q = 0,7$. .	72
Figura 6.8: Utilização dos Processadores, em relação a <i>SCN3</i> , em função do δ , para Vocoder-2, $Q = 0,7$. .	72
Figura 6.9: Qualidade de Serviço efetiva para <i>SCN4</i> , como função do δ , para Vocoder-2, $Q = 0,7$. .	72
Figura 6.10: (a) Consumo de energia, (b) utilização dos processadores e (c) qualidade de serviço efetiva, como função do δ , para Vocoder-1, $Q = 0,7$. .	73
Figura 6.11: (a) Consumo de energia, (b) utilização dos processadores e (c) qualidade de serviço efetiva, como função do δ , para GMDFa-4, $Q = 0,5$. .	74
Figura 6.12: (a) Consumo de energia, (b) utilização dos processadores e (c) qualidade de serviço efetiva, como função do δ , para TGFF70-4, $Q = 0,5$. .	75
Figura A1.1: DFG Cancelador de Eco Acústico com tempos de comunicação entre tarefas [6].	85
Figura A1.2: DFG de processamento de dados de Sonar [41] .	91
Figura A1.3: DFG do Vocoder [4].	99
Figura A1.4: DFG do TGFF70 gerado pela ferramenta TGFF [34] .	106
Figura A1.5: Parâmetros de entrada para geração da aplicação sintética TGFF70 .	107
Figura A1.6: DFG do TGFF99 gerado pela ferramenta TGFF [34] .	111
Figura A1.7: Parâmetros de entrada para geração da aplicação sintética TGFF99 .	112
Figura A2.1: Exemplo de arquivo de dados de entrada para <i>SCN3</i> - DFG com 5 tarefas. .	120
Figura A2.2: Arquivo de dados de experimentação para os algoritmos <i>offline</i> .	120
Figura A2.3: Diagrama do fluxo de execução dos algoritmos <i>offline</i> .	121
Figura A2.4: Arquivo de saída com resultados de experimento para diferentes QoS .	122
Figura A2.5: Arquivo de saída para uso de <i>SCN4</i> . .	123
Figura A2.6: Tela do aplicativo para o algoritmo <i>SCN4</i> .	124

Figura A2.7: Arquivo de caminhos para a execução do experimento	124
Figura A2.8: Tabela de dados para a aplicação do experimento	125
Figura A2.9: Nova distribuição das classes de entrada da tarefa 3, GMDFa-4, $Q = 0,7$...	126
Figura A2.10: Lista de combinações das tarefas que descartam classes de entrada.....	126
Figura A2.11: Fluxo de execução do algoritmo <i>SCN4</i>	127
Figura A2.12: Arquivo de saída do algoritmo <i>SCN4</i>	128
Figura A3.1: Gráficos das médias de (a) consumo de energia, (b) ocupação dos processadores e (c) qualidade efetiva quando apenas uma tarefa descarta classes de dados de entrada.	132
Figura A3.2: Gráficos das médias para (a) consumo de energia, (b) ocupação dos processadores e (c) qualidade efetiva quando apenas uma tarefa descarta classes de dados de entrada.	139

Lista de Tabelas

Tabela 2.1: Modos de operação para processadores Intel Pentium M com construção de 0,13 micron [17]	12
Tabela 4.1: Configuração do sistema para cada cenário	27
Tabela 5.1: Configuração do sistema para execução da aplicação (SCN2)	55
Tabela 5.2: Configuração do sistema para execução da aplicação (SCN3)	57
Tabela 6.1: Aplicações	61
Tabela A1.1: Tempos de execução em SW e HW [6]	86
Tabela A1.2: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia.....	87
Tabela A1.3: Mapeamento das tarefas nos sistemas experimentais	89
Tabela A1.4: Dados da Aplicação Cancelador de Eco Acústico	90
Tabela A1.5: Tempos de execução das tarefas de sonar [41]	92
Tabela A1.6: Mapeamento das tarefas de sonar em 7 processadores [41].....	92
Tabela A1.7: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia.....	94
Tabela A1.8: Dados da Aplicação Sonar	97
Tabela A1.9: Ciclos de relógio e número de instruções para execução das tarefas do Vocoder [14].....	100
Tabela A1.10: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia.....	101
Tabela A1.11: Mapeamento das tarefas nos sistemas experimentais	104
Tabela A1.12: Dados da Aplicação Vocoder	105
Tabela A1.13: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia.....	108
Tabela A1.14: Mapeamento das tarefas nos sistemas experimentais	109
Tabela A1.15: Dados da Aplicação TGFF70	110
Tabela A1.16: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia.....	113
Tabela A1.17: Mapeamento das tarefas nos sistemas experimentais	116
Tabela A1.18: Dados da Aplicação TGFF90	117
Tabela A3.1: Relações de consumo de energia e ocupação dos processadores quando qualidade de serviço mínima $Q_{MIN} = 0,50$	129
Tabela A3.2: Relações de consumo de energia e ocupação dos processadores quando qualidade de serviço mínima $Q_{MIN} = 0,70$	130
Tabela A3.3: Combinações de descartes usadas no Cenário 4 para Vocoder-2, $Q = 0,5$..	131
Tabela A3.4: Resultados para descarte de classes nas tarefas (a) 2, (b) 8, (c) 13, (d) 21, (e) 39, (f) 65 e (g) a média dos valores	133
Tabela A3.5: Resultados para descarte simultâneo das combinações (a) 2/8, (b) 2/13, (c) 2/65, (d) 13/65, (e) 21/39, (f) 8/39, (g) 13/21 e (h) a média dos valores.5	134
Tabela A3.6: Resultados para descarte simultâneo das combinações (a) 2/8/13, (b) 8/13/65, (c) 2/13/21, (d) 21/39/65, (e) 2/21/39, (f) 8/21/39, (g) 2/8/65 e (h) a média dos valores. .	135
Tabela A3.7: Resultados para descarte simultâneo das combinações (a) 2/8/13/65, (b) 8/13/21/65, (c) 2/21/39/65, (d) 2/8/39/65, (e) 8/13/39/65, (f) 2/8/21/39, (g) 2/13/21/39 e (h) a média dos valores.	136
Tabela A3.8: Resultados para descarte simultâneo das combinações (a) 2/8/13/21/39, (b) 2/8/13/21/65, (c) 2/8/13/39/65, (d) 2/8/21/39/65, (e) 2/13/21/39/65, (f) 8/13/21/39/65 e (g) a média dos valores.	137
Tabela A3.9: Resultados para descarte simultâneo de todas as tarefas.	138

Tabela A3.10: Combinações de descartes usadas no Cenário 4 para Vocoder-2, $Q = 0,7$	138
Tabela A3.11: Resultados para descarte de classes nas tarefas (a) 1, (b) 8, (c) 13, (d) 21 e (e) a média dos valores.....	140
Tabela A3.12: Média dos resultados para descarte simultâneo em duas tarefas.....	141
Tabela A3.13: Média dos resultados para descarte simultâneo em três tarefas.	141
Tabela A3.14: Resultados para descarte simultâneo de todas as tarefas.	141
Tabela A3.15: Combinações de descartes usadas no Cenário 4 para Vocoder-1, $Q = 0,7$	142
Tabela A3.16: Média dos resultados para descarte em apenas uma tarefa.	142
Tabela A3.17: Média dos resultados para descarte simultâneo em duas tarefas.....	142
Tabela A3.18: Média dos resultados para descarte simultâneo em três tarefas.	143
Tabela A3.19: Resultados para descarte simultâneo de todas as tarefas.	143
Tabela A3.20: Combinações de descartes usadas no Cenário 4 para GMDFa-4, $Q = 0,5$.	144
Tabela A3.21: Média dos resultados para descarte em apenas uma tarefa.	144
Tabela A3.22: Média dos resultados para descarte simultâneo em duas tarefas.....	144
Tabela A3.23: Média dos resultados para descarte simultâneo em três tarefas.	145
Tabela A3.24: Média dos resultados para descarte simultâneo em quatro tarefas.	145
Tabela A3.25: Média dos resultados para descarte simultâneo em cinco tarefas.....	145
Tabela A3.26: Resultados para descarte simultâneo de todas as tarefas.	146
Tabela A3.27: Combinações de descartes usadas no Cenário 4 para TGFF70-4, $Q = 0,5$	146
Tabela A3.28: Média dos resultados para descarte em apenas uma tarefa.	147
Tabela A3.29: Média dos resultados para descarte simultâneo em duas tarefas.....	147
Tabela A3.30: Média dos resultados para descarte simultâneo em três tarefas.	147

Lista de Acrônimos

ABB – Adaptative Body Biasing

DFG – Data Flow Graph

CHIP – Integrated Circuit

DPM – Dynamic Power Management

DSP – Digital Signal Processing

DVS – Dynamic Voltage Scaling

DVFS – Dynamic Voltage/Frequency Scale

FPGA – Field-programmable Gate Array

GPS – Global Positioning System

HW – Hardware

MPSoC – Multiprocessor system-on-chip

NUMA – Nonuniform Memory Access Multiprocessor

PDA – Personal Digital Assistant

PDFG – Probabilistic Data Flow Graph

PE – Processing Element

QoS – Quality of Service

SMP – Symmetric Multiprocessors

SoC – System-on-Chip

SVS – Static Voltage Scaling

SW – Software

UMA – Uniform Memory Access Multiprocessors

VLSI – Very Large Scale Integration

Publicações Relacionadas a este Trabalho

A. M. Tokarnia, P. C. Pepe, L. D. Pagotto, "Path-Based Dynamic Voltage Frequency Scaling Algorithms for Multiprocessor Embedded Applications with Soft Delay Deadlines", *14th Euromicro Conference on Digital System Design: Methods, Architectures, and Tools - Oulu, Finland*, Aug. 31-Sept. 2, 2011

P. C. Pepe, A. M. Tokarnia, "Escalonamento Dinâmico de Tensão (DVS) Baseado em Caminhos aplicado a Sistemas Embarcados Multiprocessador com Especificação de Qualidade de Serviço", *Anais do Segundo Encontro de Alunos e Docentes do Depto. de Engenharia de Computação e Automação Industrial da FEEC Unicamp – EADCA*, 26 e 27/03/2009

P. C. Pepe, A. M. Tokarnia, M. Ferretti, "Desafios na Redução de Consumo em Dispositivos Alimentados por Bateria", *IX Seminário sobre a Eletro-Eletrônica Aplicada à Mobilidade - Novas Tecnologias*, 31/05/2007 – disponível somente pelo email pedropepe@gmail.com

1 Introdução

Novas tecnologias surgem a cada dia e a integração de funções em um mesmo sistema não pára de crescer, tornando-os cada vez mais complexos e, conseqüentemente, mais ávidos por energia. Sistemas portáteis e de comunicação, que contam com diversas aplicações de *streaming* multimídia e os mais diversos aplicativos de uso pessoal e profissional, devem ser rápidos, eficientes e, ao mesmo tempo, prolongar ao máximo a duração de suas baterias.

Tais sistemas são classificados como sistemas de computação embarcados, e possuem geralmente um ou mais processadores, FPGA's, DSP's e periféricos, como memórias, *flash drivers* e outros *hardwares* de uso específico. São exemplos destes sistemas embarcados os celulares, *notebooks*, *smartphones*, *tablets*, navegadores GPS, câmeras digitais e uma infinidade de outros dispositivos que integram uma ou várias funções em um único sistema portátil.

O uso de processadores com capacidade de escalonamento dinâmico de tensão/frequência (DVFS – *Dynamic Voltage/Frequency Scaling*), como Intel XScale [18], AMD Athlon [1] e Transmeta Crusoe [13], possibilitam a troca do nível de tensão/frequência de operação durante a execução da aplicação, provendo à aplicação a oportunidade de reduzir o consumo de energia ou aumentar a velocidade de processamento de acordo com a necessidade momentânea. Uma vez que o consumo de energia normalmente tem uma dependência quadrática com a tensão, níveis mais baixos de tensão favorecem a economia de energia. Entretanto, diminuir os níveis de energia aumenta o retardo dos transistores, impondo o uso de um menor par tensão/frequência de operação, aumentando o tempo de execução das tarefas [35].

Nos sistemas embarcados, as aplicações para processamento de sinais são principalmente caracterizadas por contarem com um processamento repetitivo sobre entradas recebidas periodicamente, pela incerteza do tempo de execução para cada entrada recebida e pela tolerância a ocasionais falhas no cumprimento do tempo de execução sem que possam ser percebidos pelo usuário [15].

Para estes sistemas existem atualmente diversas formas de reduzir o consumo de energia como, por exemplo, métodos de otimização de *software* [38] e algoritmos de escalonamento de tensão [32][36][42][47]. Além disso, o uso de sistemas multiprocessadores está cada vez mais difundido entre os pesquisadores de sistemas embarcados, e aparece como uma boa alternativa na busca de desempenho com consumo reduzido [36]. Na busca da redução do consumo de energia, alguns algoritmos fazem uso do recurso de desligamento de processadores e outros componentes do sistema, através do gerenciamento dinâmico de potência, DPM (*Dynamic Power Management*) [5][27][35]. Outros sistemas contam ainda com o recurso de alteração da tensão de limiar de polarização dos transistores, possibilitando a algoritmos de redução de consumo de energia a redução da corrente de fuga nos transistores do sistema, quando trabalhando em baixas tensões de operação [5][21][26][29][35].

Tais sistemas podem apresentar diferentes classes de dados de entrada, que exigem tempos de processamento diferentes em uma mesma tarefa da aplicação. Esta distinção nos tempos de processamento se deve aos diferentes conteúdos destes dados de entrada que podem,

por exemplo, formar diferentes quadros de um *streaming* de vídeo, com diferentes detalhamentos de imagem. A distribuição destas classes de dados no tempo (probabilidade de ocorrência) pode ser fixa ou sofrer uma variação causada, por exemplo, por alterações na disponibilidade de sinal para comunicação com a rede de telefonia celular, ou então pela mudança do conteúdo tratado por um sistema multimídia. Este comportamento torna necessário, antes ou durante a operação, calcular o nível de tensão/frequência associado à execução das tarefas do sistema, para que o tempo de processamento e a Qualidade de Serviço (QoS – *Quality of Service*) sejam mantidos. Esta QoS é definida pela relação entre a quantidade de dados de entrada recebida pela aplicação e a quantidade de dados efetivamente tratada pela aplicação dentro do tempo limite para processamento.

Neste trabalho, a aplicação é modelada sobre grafos de tarefas escalonadas em multiprocessadores, tendo cada tarefa classes de dados de entrada com distribuição probabilística de ocorrência. O tempo de execução de cada tarefa é definido em função da classe dos dados de entrada e do nível de tensão/frequência.

As limitações impostas à aplicação consistem em um tempo máximo de execução com tolerância a atrasos, respeitando uma QoS mínima. Então, desde que a qualidade mínima de serviço seja respeitada, é aceitável que alguns *frames* de dados não sejam processados no tempo máximo de execução.

Os algoritmos propostos procuram explorar esta definição de QoS e a divisão dos dados de entrada em *frames*, determinando uma configuração de operação para um sistema multiprocessador. Tal configuração estipula os níveis de tensão/frequência para execução de cada tarefa, além de definir se dados de entradas de uma determinada classe devem ser processados ou descartados.

O uso dos algoritmos depende da capacidade de cada sistema em alterar o conjunto tensão/frequência de operação (DVFS) e o nível de exigência de QoS durante a execução da aplicação. São definidos, então, 4 cenários de trabalho.

No primeiro cenário C1 (algoritmo *SCN1*), que corresponde à menor capacidade de DVFS, o nível de tensão/frequência pode ser ajustado somente no início da execução da aplicação, e este nível deve ser o mesmo para todos os processadores. Ainda, todas as classes de dados de entrada devem ser tratadas, isto é, uma QoS de 100% deve ser obedecida.

Em um segundo cenário C2 (algoritmo *SCN2*), o nível de tensão/frequência de um processador pode ser individualmente ajustado quando uma tarefa começa a ser executada e entradas de dados de determinadas classes podem ser descartadas sem processamento, possibilitando QoS abaixo de 100%.

No terceiro cenário C3 (algoritmo *SCN3*), um mecanismo mais flexível de DVFS é considerado, ajustando o nível de tensão/frequência de cada processador de acordo com a classe de dados de entrada recebida, além de possibilitar o mesmo descarte sem processamento do cenário C2.

Uma versão do algoritmo baseada em *SCN3* é apresentada para o quarto cenário C4 (algoritmo *SCN4*), no qual uma mudança na distribuição probabilística das classes de dados de

entrada dispara o cálculo, durante a execução (*online*), de uma nova configuração de operação. Enquanto nos primeiros três cenários a configuração do sistema é determinada antes da execução da aplicação (*offline*) e continua fixa durante a execução da aplicação, o quarto cenário permite um ajuste do sistema para compensar a variação nas classes de entrada, de modo a manter a QoS e, dentro do possível, o baixo consumo obtido pelo algoritmo *offline*.

Os resultados obtidos para o cenário C1 foram usados como base de comparação para os resultados obtidos com o uso dos algoritmos nos demais cenários, no que diz respeito ao consumo de energia, à QoS e à utilização dos processadores.

Estes algoritmos foram experimentados sobre várias aplicações embarcadas, e resultados significantes de economia de energia foram alcançados. Para o Cancelador de Eco [6], sendo executado sobre um sistema com 8 processadores e QoS mínima de 50%, a redução de consumo de energia foi de 64% quando considerado um sistema compatível com o cenário C2. Assumindo um cenário com as condições de C3, esta redução fica próxima de 16% quando comparado ao consumo no cenário C2, e em 70% quando comparado ao consumo para C1.

Considerando ainda o Cancelador de Eco, agora sendo executado sobre 4 processadores com a mesma QoS mínima de 50%, uma alteração na distribuição das classes de dados de entrada da aplicação (cenário C4) força o sistema a se adequar à nova condição. Para um *delta* de variação negativa de 0,10 simultaneamente em 4 tarefas com descarte de classe (redução momentânea de QoS para 31%), a adequação realizada pelo algoritmo deixou o sistema com um consumo apenas 15% maior do que o consumo de energia anterior obtido para o cenário 3. Além disso, a nova configuração foi obtida após poucas iterações do algoritmo.

A título de comparação, a utilização de caminhos de grafos de tarefas pode reduzir o número de iterações do algoritmo para obter um resultado, quando comparado a algoritmos que fazem uso de programação dinâmica matemática [32].

Esta dissertação está dividida em sete capítulos.

O Capítulo 2 demonstra alguns conceitos usados como base para o desenvolvimento do trabalho, como o uso de multiprocessadores, processamento digital de sinais, escalonamento estático e dinâmico de tensão e frequência de operação e outros conceitos que relacionam consumo de energia com níveis de operação de um sistema embarcado.

O Capítulo 3 apresenta publicações na área de redução de consumo de energia em sistemas embarcados, algumas das quais são amplamente referenciadas durante o trabalho.

O Capítulo 4 descreve o modelo dos sistemas sobre os quais são aplicados os algoritmos desenvolvidos e os cenários considerados para realização do trabalho, assim como o problema a ser abordado.

O Capítulo 5 detalha os algoritmos desenvolvidos neste trabalho para cada cenário de execução das aplicações.

O Capítulo 6 contém os resultados experimentais obtidos para as aplicações em que os algoritmos foram usados, sendo executadas de acordo com os diferentes cenários propostos.

O Capítulo 7 traz um resumo dos resultados, ressalta as contribuições deste trabalho e apresenta planos para futuros desenvolvimentos.

2 Conceitos Básicos

2.1 Introdução

Este capítulo tem como finalidade a apresentação de alguns conceitos inerentes ao funcionamento dos sistemas embarcados. É feita uma breve apresentação dos conceitos de processamento de sinais e multiprocessadores, seguida de uma introdução mais detalhada sobre o uso de fontes de alimentação em dispositivos embarcados. Este capítulo também traz os conceitos de gerenciamento dinâmico de potência e escalonamento dinâmico de tensão/frequência, considerado como um dos melhores recursos para redução de consumo de energia [39].

Embora o termo Energia defina a Potência exercida ao longo do tempo, os termos consumo de Energia e consumo de Potência serão usados neste trabalho como sinônimos na descrição de consumo das aplicações.

2.2 Processamento de Sinais

O processamento de sinais consiste na análise e modificação de sinais, de forma a extrair informações ou alterá-los para uso em outras aplicações. Neste trabalho, os algoritmos propostos destinam-se a aplicações que trabalham com processamento de sinais.

Sinais digitais podem representar os mais diversos tipos de sinais analógicos, ou mesmo digitais, como vídeo, voz e dados. Estes sinais podem ser transmitidos de forma multiplexada no tempo. O envio e recebimento de informações na forma de sinais digitais, além de seu tratamento e decodificação, fazem parte de diversos sistemas de uso diário, como celulares, câmeras digitais, reprodutores de música, PDA's, *tablets* e sistemas portáteis em geral.

Muitas unidades de processamento podem ser usadas no processamento digital de sinais como, por exemplo, DSPs, microcontroladores e outros tipos de processadores. Além disto, podem ser usados dispositivos dedicados implementados em circuitos VLSI, ou mesmo, em FPGA.

Um sinal digital pode ser formado por amostras que representam, de forma codificada, o sinal analógico. O sinal recebido por um dispositivo pode ser dividido em blocos de amostras. Estes dados de entrada do sistema, analisados individualmente ou em *frames*, fornecem informações sobre o sinal do qual foram amostrados. Para algumas aplicações, é possível classificar dados de entrada de acordo com sua periodicidade, tempo de processamento ou outra característica inerente ao tratamento.

Os algoritmos apresentados neste trabalho destinam-se a aplicações que envolvem processamento de sinais, visando melhorar o consumo de energia do sistema, mantendo o tempo de resposta e a qualidade de serviço especificados. Esta qualidade de serviço é definida pela fração de blocos de dados de entrada processada dentro do tempo de resposta da aplicação.

2.3 Multiprocessadores

Segundo conceitos definidos por Pattersson e Hennessy [30], o multiprocessador é, basicamente, um conjunto de processadores que compartilham um espaço de endereçamento ou então trocam mensagens, para resolver um problema em comum. Podem ser constituídos por vários computadores, com processadores individuais conectados por uma rede local (*Clusters*), vários processadores (diferentes encapsulamentos) ligados a um mesmo barramento, ou mesmo vários processadores construídos em um único encapsulamento, acompanhados ou não de outros dispositivos de *hardware* específico (*Multiprocessor system-on-chip - MPSoC*).

Como processadores operando em paralelo normalmente compartilham dados, é necessário coordenar a operação sobre os dados compartilhados, caso contrário, um processador pode, por exemplo, alterar um dado que está sendo utilizado por outro processador. Este processo de coordenação leva o nome de “Sincronização” e requer, sucintamente, a coordenação de dois ou mais processos ou tarefas, que podem ou não ser executados em diferentes processadores. A utilização de estruturas de bloqueio como, por exemplo, semáforos, podem habilitar o acesso aos dados numa memória compartilhada a apenas um processador de cada vez.

Quando os multiprocessadores têm processadores que compartilham o mesmo espaço de endereçamento de memória, podem ser classificados em dois tipos. No primeiro, qualquer processador leva sempre o mesmo tempo para acessar a memória compartilhada, não importando qual endereço requisitado. Tais sistemas são chamados de Multiprocessadores de Acesso Uniforme à Memória (*Uniform Memory Access Multiprocessors – UMA*) ou Multiprocessadores Simétricos (*Simmetric Multiprocessors - SMP*). No segundo tipo de multiprocessador, alguns acessos à memória comum são mais rápidos que outros, dependendo do processador e do endereço. A estes sistemas é dado o nome de Multiprocessadores de Acesso Não Uniforme à Memória (*Nonuniform Memory Access Multiprocessor - NUMA*).

O uso de multiprocessadores se iniciou em grandes computadores, com grande poder de processamento. No entanto, com a popularização da tecnologia MPSoC, os multiprocessadores estão sendo oferecidos em computadores pessoais, como os “multi-core” Intel i3, i5 e i7 [19], Intel Xeon Tri-Core e Quad-Core [20], AMD Phenon Quad-Core [2] além de muitos outros. Atualmente, o uso de multiprocessadores pode ter como objetivo: (1) melhorar o desempenho, (2) reduzir o consumo de energia, ou (3) uma combinação dos objetivos anteriores, comparado a um sistema constituído por um único processador.

Os algoritmos propostos neste trabalho podem ser usados em aplicações que são executadas em sistemas multiprocessador.

2.4 Fontes de Alimentação em Dispositivos Embarcados

O consumo de energia em dispositivos embarcados pode variar de acordo com as operações efetuadas pela aplicação, processadores utilizados, periféricos envolvidos e interconexões entre processadores e periféricos.

Segundo o exposto por Tiwari *et al.* [39], embora a dissipação de potência por corrente de fuga seja um motivo de preocupação, o consumo dinâmico ainda domina a maioria dos sistemas embarcados. A energia consumida é proporcional ao quadrado da tensão de alimentação e, portanto, a redução da tensão de alimentação pode ocasionar uma grande economia. Ainda de acordo com Tiwari *et al.* [39], análises de dissipação de potência em CPU's de alto desempenho mostraram que, comparado a várias técnicas de redução de consumo, o escalonamento de tensão/frequência é uma das mais efetivas e promissoras.

Seguem alguns esclarecimentos sobre a influência das tensões de polarização e de operação no consumo de energia, as quais podem influir diretamente no consumo de um sistema embarcado.

2.4.1 Tensão de Polarização da Fonte de Alimentação

Com o uso de variadas técnicas de redução de consumo e com as cada vez menores tecnologias de circuitos integrados, a corrente de fuga dos circuitos passa a ser responsável por uma considerável parcela do consumo de energia dos sistemas embarcados. Segundo Martin *et al.* [26] esta fuga pode ser minimizada com o uso de técnicas como a polarização adaptativa (*ABB - adaptive body biasing*), que ajusta a tensão de limiar de acionamento dos transistores dinamicamente (*threshold voltage*), reduzindo a fuga de corrente.

Em dispositivos embarcados, dois tipos de circuitos reguladores de alimentação são mais usados: fontes lineares (em reguladores de tensão) e fontes chaveadas (em conversores DC-DC). Qualquer que seja a fonte, existe uma corrente de polarização (*bias*) que, mesmo sem carga conectada à fonte, ocasiona um consumo residual, uma corrente de fuga. Caso uma aplicação demande um baixo consumo de energia, e o sistema seja projetado para suprir apenas esta baixa demanda sem considerar esta corrente de fuga, é possível que o comportamento do sistema, quando em execução, não atenda às especificações de consumo do projeto.

A curva de corrente fornecida pela bateria, I_{bat} , em função da corrente consumida pelo sistema, I_{dd} , é mostrada na Figura 2.1, para os dois tipos de fonte citadas, tanto para seus comportamentos ideais, como em relação a seus comportamentos reais. Nesta figura é possível observar que a corrente de polarização da fonte linear, $I_{bias\ 1}$ tende a ser menor que a corrente de polarização da fonte chaveada, $I_{bias\ 2}$ para valores abaixo de I_L . Ou seja, nesta faixa de valores de I_{dd} , a fonte linear pode ser mais eficiente que a fonte chaveada, embora a relação entre corrente fornecida e corrente consumida, indicada no gráfico da Figura 2.1, aponte para a fonte chaveada

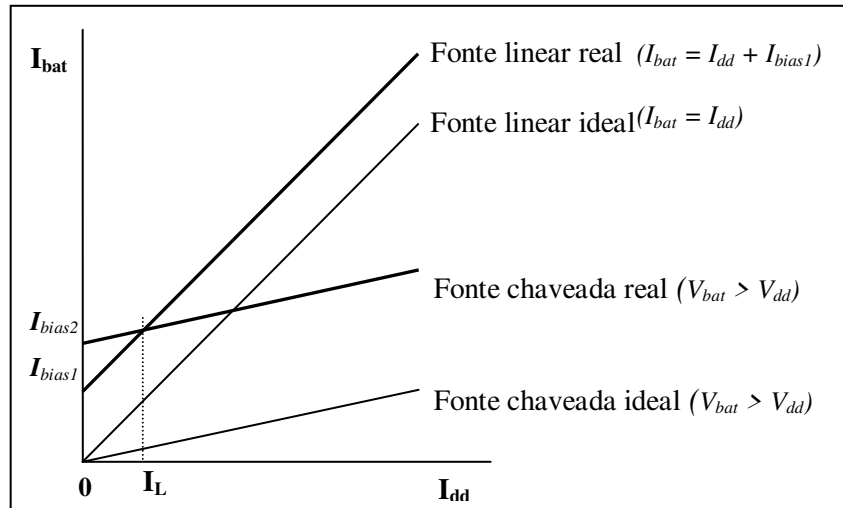


Figura 2.1: Aumento de consumo devido à corrente de polarização da fonte [9].

como melhor opção para redução de consumo [9]. Os valores de I_{bias1} e I_{bias2} podem chegar a alguns miliampères, e valores abaixo de 5 mA são desejados para sistemas comerciais.

As fontes ideais apresentadas na Figura 2.1 são assim chamadas por não apresentarem perdas devido a correntes de fuga. Quanto menor o valor de I_{bias} para uma fonte, maior é sua eficiência, pois o consumo de energia resultante de correntes de fuga será ínfimo quando comparado ao consumo total do sistema. Para outros casos em que o consumo de energia para execução de uma aplicação é muito grande, uma fonte real pode ter comportamento equivalente ao de uma fonte ideal, pois a corrente de fuga é desprezível frente ao consumo total de energia.

Embora a dissipação de potência devido à corrente de fuga do sistema exista, ela não foi considerada neste trabalho, pois os algoritmos trabalham apenas com a flexibilidade do sistema em alterar dinamicamente seus níveis de operação.

2.4.2 Tensão de Operação

Trabalhos anteriores que descrevem técnicas de redução de consumo de energia exploram diretamente a proporcionalidade entre o quadrado da tensão de operação de um processador e seu consumo de energia [36][39][42][47]. Ou seja, duplicando-se a tensão de operação, a potência consumida pelo dispositivo aumenta em, aproximadamente, quatro vezes quando comparada à potência consumida originalmente.

As equações a seguir demonstram que esta proporcionalidade quadrática só é possível quando a tensão da bateria é a mesma usada pelo processador, ou quando é feito o uso de uma fonte chaveada (conversor DC-DC), recurso comum em equipamentos portáteis operados à bateria, como *notebooks*, celulares, *smartphones*, *tablets* e afins, por ser mais eficiente que as fontes lineares. No entanto, dispositivos embarcados contam, muitas vezes, apenas com fontes lineares, como transistores reguladores e diodos Zener, que simplesmente reduzem a tensão da

bateria para alcançar a tensão de operação dos processadores e microcontroladores usados, como em sistemas de alarme automotivo, alarmes residenciais e outros sistemas de menor ou maior complexidade.

A Figura 2.2 descreve, sucintamente, a alimentação dos referidos dispositivos.

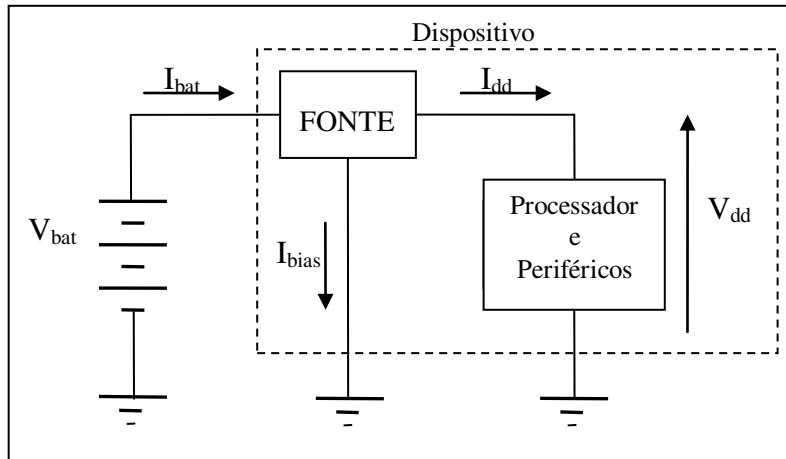


Figura 2.2: Esquema simplificado de um dispositivo conectado a uma bateria

Quando a fonte é linear, a relação da corrente fornecida pela bateria I_{bat} e a corrente fornecida ao sistema I_{dd} é direta, e é dada por

$$I_{bat} \cong I_{dd} + I_{bias\ 1} \quad (\text{eq.1}),$$

onde $I_{bias\ 1}$ é a corrente de fuga para a fonte linear [9].

Em fontes chaveadas, a potência fornecida pela fonte de alimentação é consumida pela fonte do dispositivo, acrescido das perdas por fuga de corrente. No entanto, não existe uma relação direta entre a corrente fornecida pela fonte e a corrente consumida pelo sistema, e esta relação é obtida através da potência. Assim,

$$P_{bat} \cong P_{dd} + P_{bias\ 2} \quad (\text{eq.2}),$$

onde P_{bat} é a potência fornecida pela bateria, P_{dd} é a potência consumida pelo sistema, e $P_{bias\ 2}$ é a potência resultante da corrente de fuga [9].

O consumo de potência de um dispositivo é dado por

$$P_{bat} = V_{bat} \cdot I_{bat} \quad (\text{eq.3}),$$

onde V_{bat} é a tensão de bateria e I_{bat} a corrente elétrica fornecida. Fazendo-se uso das equações (eq.2) e (eq.3), temos a relação entre a corrente de bateria I_{bat} e a corrente do sistema I_{dd} para uma fonte chaveada,

$$\begin{aligned} V_{bat} \cdot I_{bat} &\cong V_{dd} \cdot I_{dd} + V_{bat} \cdot I_{bias\ 2} \\ I_{bat} &\cong \frac{V_{dd}}{V_{bat}} \cdot I_{dd} + I_{bias\ 2} \end{aligned} \quad (\text{eq.4}),$$

onde $I_{bias\ 2}$ é a corrente de fuga para esta fonte.

Ainda, considerando que a corrente elétrica pode ser definida como sendo um fluxo de cargas em um espaço de tempo:

$$I_{dd} = \frac{Q}{t} \quad (\text{eq.5}),$$

a carga Q consumida pelo sistema no intervalo de tempo t é igual à capacitância de carga/descarga multiplicada pela tensão, ou seja,

$$Q = C \cdot V_{dd} \quad (\text{eq.6})$$

e o intervalo de tempo t é o inverso da frequência de operação:

$$t = \frac{1}{f} \quad (\text{eq.7}),$$

podemos definir as equações de potência a seguir para os dois tipos de fonte:

Fonte Linear:

Da equação (eq.1) e da fórmula de potência (eq.3), é possível definir a potência para a fonte linear como

$$P_L \cong V_{bat} \cdot (I_{dd} + I_{bias\ 1}) \quad (\text{eq.8}) \text{ ou}$$

$$P_L \cong V_{bat} \cdot I_{dd} + V_{bat} \cdot I_{bias\ 1} \quad (\text{eq.9})$$

Utilizando as equações (eq.5), (eq.6) e (eq.7), podemos reescrever (eq.9) como:

$$P_L \cong V_{bat} \cdot V_{dd} \cdot C \cdot f + V_{bat} \cdot I_{bias\ 1} \quad (\text{eq.10})$$

Fonte Chaveada:

Das equações (eq.3) e (eq.4),

$$P_C \cong V_{bat} \cdot \left(\frac{V_{dd}}{V_{bat}} \cdot I_{dd} + I_{bias\ 2} \right) \quad (\text{eq.11}) \text{ ou}$$

$$P_C \cong V_{dd} \cdot I_{dd} + V_{bat} \cdot I_{bias 2} \quad (\text{eq.12})$$

Utilizando as equações (eq.5), (eq.6) e (eq.7), podemos reescrever (eq.12) como:

$$P_C \cong V_{dd}^2 \cdot C \cdot f + V_{bat} \cdot I_{bias 2} \quad (\text{eq.13})$$

Supomos que um sistema eletrônico com corrente de fuga na fonte e consumo de periféricos desprezíveis conte com o processador StrongArm referenciado por Schaumont *et al.* [36]. Este sistema também pode ser configurado dinamicamente em dois níveis de operação, tendo o nível de menor consumo uma tensão V_{low} e uma frequência associada f_{low} , enquanto que o nível de maior consumo conta com uma tensão $V_{high} = 2,09 \cdot V_{low}$ e uma frequência $f_{high} = 4,25 \cdot f_{low}$. Se este dispositivo for dotado de uma fonte linear, a mudança do nível de operação de “high” para “low” gera uma redução de aproximadamente 8,88 vezes na potência consumida, de acordo com a equação (eq.10). Entretanto, se este mesmo dispositivo for dotado de uma fonte chaveada, a equação (eq.13) mostra que a redução na potência consumida chega a aproximadamente 18,5 vezes.

2.4.3 Tensão de Operação VS Frequência de Operação

Para a maioria dos dispositivos eletrônicos que fazem uso de processadores ou microcontroladores, a variação da tensão de operação afeta a frequência máxima de processamento e o tempo de resposta alcançada por um processador ou microcontrolador. A Tabela 2.1 mostra as possibilidades de utilização de um processador da família Intel Pentium M utilizado em *notebooks*, relacionando a tensão de alimentação e máxima frequência de operação, além do consumo esperado [17]. Considerando o modo de desempenho máximo, somente utilizando tensão acima de 1,4V é possível obter o desempenho nominal do processador (1,7 GHz), enquanto abaixo desta tensão, o desempenho máximo alcança valores próximos a 75% do desempenho nominal. Utilizando a primeira linha da tabela para comparar os modos de operação, o modo de operação de bateria otimizada pode ter desempenho abaixo de 36% do modo de desempenho máximo.

Assim, torna-se necessário definir uma política correta de alteração dos níveis de operação, de modo a obter um menor consumo de energia sem prejudicar o desempenho do dispositivo eletrônico em uso.

Tanto fontes chaveadas quanto fontes lineares são utilizadas em dispositivos eletrônicos. No entanto, algoritmos de redução de consumo geram melhores resultados quando utilizados em dispositivos dotados de fontes chaveadas, como visto através da equação (eq.13), em que o consumo de energia é proporcional ao quadrado da tensão de operação do sistema de processamento.

Tabela 2.1: Modos de operação para processadores Intel Pentium M com construção de 0,13 micron [17]

Modo de desempenho máximo			Modo de bateria otimizada		
Frequência	Tensão	Consumo de energia	Frequência	Tensão	Consumo de energia
Processadores 0,13 micron					
1,70 GHz	1,48 V	24,5 W	600 MHz	0,956 V	6,0 W
1,60 GHz	1,484 V	24,5 W	600 MHz	0,956 V	6,0 W
1,50 GHz	1,484 V	24,5 W	600 MHz	0,956 V	6,0 W
1,40 GHz	1,484 V	22,0 W	600 MHz	0,956 V	6,0 W
1,30 GHz	1,388 V	22,0 W	600 MHz	0,956 V	6,0 W
1,20 GHz	1,39 V	12,0W	600 MHz	0,956 V	6,0 W
1,10 GHz	1,10 V	12,0W	600 MHz	0,956 V	6,0 W
1 GHz	1,39 V	7,0W	600 MHz	0,844 V	6,0 W
900 MHz	1,004 V	7,0W	600 MHz	0,844 V	6,0 W

2.5 Escalonamento Dinâmico de Tensão e Frequência (DVFS)

Vários processadores podem funcionar com diferentes tensões de operação, resultando em diferentes velocidades de processamento e diferentes níveis de consumo. Este ajuste no nível de operação pode ser feito de acordo com a necessidade momentânea de processamento, aumentando-se o nível de tensão/frequência quando for necessário maior poder de processamento, ou diminuindo o nível de tensão/frequência, para reduzir o consumo de energia, quando o processamento for simples ou puder ser feito com menor velocidade.

Como descrito no item anterior, o consumo do processador pode aumentar com o quadrado do aumento de tensão. Schaumont *et al.*[36] comparam o desempenho e o consumo de energia de processadores operando em diferentes níveis de operação. O anteriormente citado processador StrongArm [36] pode ter um aumento de desempenho de 4,25 vezes ao alterar-se seu nível de operação. Em contrapartida, o consumo de energia aumenta em 18,5 vezes. Estes dados exemplificam a necessidade de uma correta estratégia de alteração dos níveis de operação, de modo a obter um menor consumo com um desempenho satisfatório, de acordo com as restrições de desempenho especificadas para a aplicação.

Quando o comportamento de uma aplicação em um determinado sistema é conhecido, é possível escalar os ajustes de tensão dos processadores do sistema antes que a aplicação seja iniciada. A esta definição de tensões de operação antes da execução da aplicação é dada a denominação de Escalonamento Estático de Tensão/Frequência (*Static Voltage/Frequency Scaling - SVFS*).

No entanto, nem todos os sistemas podem ter seu comportamento definido antes do início da aplicação. Para isso é necessário reconhecer variações dos dados de entrada e necessidades presentes no sistema. Este ajuste dinâmico dos níveis de tensão de operação através do

reconhecimento do estado instantâneo do sistema é denominado Escalonamento Dinâmico de Tensão/Frequência (*Dynamic Voltage/Frequency Scaling - DVFS*).

Vários são os trabalhos que fazem uso de escalonamento de tensão/frequência para redução de consumo do sistema, assim como os algoritmos apresentados no Capítulo 5 deste trabalho. Como citado anteriormente, Tiwari *et al.* [39] descreve o Escalonamento de Tensão/Frequência como um dos mais promissores métodos de redução de consumo.

2.6 Gerenciamento Dinâmico de Potência (DPM)

Além da possibilidade de variar os níveis de operação, um grande número de sistemas embarcados também conta com a possibilidade de alteração do estado de operação do processador e periféricos, possibilitando colocá-los em estado de espera, com consumo de energia muito reduzido.

O Gerenciamento Dinâmico de Potência (DPM – *Dynamic Power Management*) se refere a esquemas de controle de potência durante a execução de programas, e várias arquiteturas de processamento disponibilizam uma estrutura equivalente a uma instrução de *halt*, que reduz a potência da CPU durante períodos de ociosidade. Além disso, sistemas de processamento mais recentes, com variados periféricos integrados no mesmo encapsulamento (SoC), usualmente apresentam um gerenciamento do *clock* controlado via *software*, possibilitando a redução do consumo de energia em periféricos inativos e em seus controladores, além de outros artifícios de desligamento ou redução de consumo para memórias e outras estruturas que venha a consumir energia em períodos de ausência de carga de trabalho no sistema. Estas técnicas de gerenciamento dinâmico são bem conhecidas e utilizadas em sistemas de tempo real, eliminando tanto a dissipação dinâmica quanto a dissipação estática de potência.

Vários trabalhos fazem uso do gerenciamento dinâmico de potência para redução de consumo de energia do sistema, e este recurso é bastante empregado em associação a outras técnicas de redução de consumo de energia [5][27][35], como o DVFS. Esta técnica não é utilizada no desenvolvimento deste trabalho, mas sua apresentação é interessante devido ao uso em diversos outros trabalhos citados no decorrer dos capítulos seguintes.

2.7 Conclusão

Foram apresentamos neste capítulo alguns conceitos relativos ao funcionamento de dispositivos embarcados, que são base de entendimento e desenvolvimento para diversos algoritmos de redução de consumo. Processamento digital de sinais e sistemas multiprocessador serão novamente abordados neste trabalho, e são base para o desenvolvimento do algoritmo, assim com o recurso de DVFS, disponível em diversos sistemas embarcados.

3 Trabalhos Correlatos

3.1 Introdução

Sistemas de computação embarcada destinados ao processamento de sinais podem ser caracterizados pelo processamento repetitivo de entradas recebidas periodicamente, pela incerteza do tempo de resposta às entradas e pela tolerância a ocasionais falhas no cumprimento do prazo de execução, desde que passem despercebidas pelo usuário [15]. Esta observação se aplica, por exemplo, a sinais que representam imagens e sons. Partindo desta premissa, este capítulo tem a finalidade de apresentar abordagens e técnicas de redução de consumo de energia presentes na literatura, principalmente sobre o uso de DVFS e qualidade mínima de serviço, técnicas estas, também utilizadas neste trabalho. As abordagens aqui apresentadas utilizam algumas técnicas de redução de consumo de energia, incluindo o uso de DVFS, aplicada isoladamente ou combinada a outras técnicas de redução de consumo.

3.2 Trabalhos Anteriores

No decorrer dos últimos anos, houve uma grande evolução dos sistemas embarcados, principalmente devido à grande integração de vários componentes em um único *chip*, possibilitando um maior processamento com menores índices de consumo de energia.

3.2.1 Importância do Consumo de Energia do Processador

Lee e Kim [24] fazem uma comparação entre os sistemas embarcados atuais e os sistemas de anos atrás. Alguns sistemas embarcados, como os existentes em pequenos robôs, contavam com atuadores que tinham um consumo de energia bem maior que o processador de controle, e os projetistas ignoravam o consumo do processador durante o desenvolvimento. Com a evolução da integração e inteligência destes sistemas, já não é mais possível desprezar o consumo de um poderoso processador de controle, frente aos cada vez mais econômicos atuadores, sensores e outros periféricos.

Ainda citando o exemplo de sistemas embarcados em um robô, Mei *et al.* [27] apresentam o Pioneer 3DX, um robô composto de computação embarcada, microcontroladores, sensores e motores, onde o consumo de energia envolvido em seu movimento pode chegar a apenas 12% do consumo de energia total do sistema, enquanto o consumo de energia do processador de controle embarcado pode chegar a 65%. Os autores ainda combinam o planejamento dos movimentos do robô com a utilização de algoritmos de escalonamento em tempo real e gerenciamento dinâmico de potência (DPM – *Dynamic Power Management*) para reduzir o consumo de energia na execução dos movimentos do robô.

Quando são utilizados sistemas em que a configuração de operação pode ser alterada durante a execução de aplicações, um vasto campo de técnicas de redução de consumo é aberto. Tiwari *et al.* [39] analisam o consumo de energia em CPUs de alto desempenho, e fazem a análise comparativa de várias técnicas que visam a redução de consumo de energia. Técnicas de escalonamento de tensão (*Voltage Scaling*), desligamento parcial da rede de *clock* (*Clock Gating*), bibliotecas de células de *hardware* (*Libraries*), curvas de atraso de potência (*Power-Delay Curves*), dimensionamento automático de transistores (*Automated Transistor Sizing*), síntese dos circuitos lógicos de baixo consumo de energia (*Low Power Logic Synthesis*), otimização individual de circuitos (*Specific Circuit Level Techniques*), gerenciamento de potência do sistema (*System Power Management*) e redução do consumo de energia baseado no código (*Software Based Power Reduction*) são analisadas. A conclusão de Tiwari *et al.*, é que o escalonamento de tensão aparece com uma das mais efetivas e promissoras técnicas de redução de consumo de energia.

O escalonamento dinâmico de tensão e frequência (DVFS) é amplamente aplicado em diversos sistemas, isoladamente ou combinado a outras técnicas de redução, a técnicas de otimização de código e a uma qualidade mínima de serviço exigida pela aplicação, além de outras combinações.

3.2.2 Técnicas de Escalonamento de Tensão/Frequência

Várias são as técnicas de Escalonamento de Tensão/Frequência, assim como variados são os resultados obtidos. Estes resultados podem apresentar desde uma solução ideal, com máxima redução de consumo para sistemas com dados de entrada com pouca variação, até uma solução *online* menos eficiente, mas que pode ser usada em um sistema com entradas continuamente variáveis.

SOLUÇÃO EXATA

A busca por uma solução exata para um sistema pode ser exemplificada pelo trabalho de Cao *et al.* [8], onde é proposto um método de programação linear inteira *offline* para obter um mínimo consumo de energia para aplicações com rigorosos prazos de tratamento. Também é descrito um algoritmo *online* baseado em uma robusta programação linear sequencial, além da comparação de resultados com outros algoritmos de escalonamento de tensão. Os autores apresentam resultados interessantes para sistemas com apenas um processador, onde são tratadas *online* alterações nos dados de entrada recebidos. O levantamento de dados *offline* para este tratamento indicou um caminho a ser seguido para o desenvolvimento dos algoritmos propostos neste trabalho.

Técnicas de programação matemática também são combinadas ao escalonamento de tensão e frequência para a obtenção de uma solução ideal, como no trabalho de Qiu *et al.* [32], que propõem uma técnica *offline* de programação matemática dinâmica de escalonamento de tensão para sistemas multiprocessador. Estes sistemas são submetidos à taxas mínimas de tratamento dos dados de entrada, garantindo que a qualidade mínima especificada para o serviço

seja satisfeita. Seu modelo de sistema é baseado na distribuição dos tempos de execução. A complexidade varia como uma potência do número de classes de entrada, e também depende do número de antecessores/sucessores de uma tarefa, do quadrado do número de nós, do tempo máximo de resposta, e do número de níveis de tensão. O modelo de sistema, com tolerância a falhas de tratamento e divisão dos dados recebidos em classes de entrada, foi usado como parte do trabalho aqui apresentado.

Para sistemas complexos, com grande número de tarefas e arestas de ligação, contando com várias classes de entrada por tarefa, a complexidade do algoritmo usado pode definir o tempo e/ou poder de processamento necessário para se obter uma solução ótima de escalonamento de tensão e frequência.

SOLUÇÃO APROXIMADA OFFLINE

No entanto, é possível que uma solução aproximada do escalonamento de Tensão/Frequência para um sistema possa gerar resultados de redução de consumo de energia próximos aos dos algoritmos que buscam soluções de escalonamento ideais, gastando menor tempo e poder de processamento.

Este escalonamento aproximado pode ser obtido antes da aplicação ser executada no sistema. Hua *et al.* [15] exploram a tolerância do usuário a ocasionais falhas, garantindo uma qualidade mínima de serviço (QoS) através de heurísticas de escalonamento de tensão, aplicados a um sistema multimídia. O resultado obtido por estas heurísticas aponta um nível de operação ideal, que pode não existir no sistema em questão. De modo a contornar a inexistência deste nível de operação, os autores executam a tarefa nos dois níveis imediatamente inferior e superior ao nível ideal a ela definido para execução, encontrando o momento da troca de um nível para outro, “simulando” o nível desejado. Estas heurísticas se aplicam a sistemas com um único processador, com um único caminho de tarefas e com uma distribuição fixa de classes de entrada. Este algoritmo permite alcançar bons resultados para alguns sistemas, mas conforme mostrado por Qiu *et al.* [32] não pode ser facilmente adaptado para sistemas multiprocessadores.

Hwang *et al.* [16] descrevem uma implementação, baseada em *hardware*, de uma unidade de gerenciamento de energia dedicada a recolher e analisar padrões associados ao acesso a dispositivos de entrada/saída. O consumo de energia de um dispositivo em modo de espera é muitas vezes maior que o consumo do mesmo dispositivo desligado. No entanto, o tempo de resposta do dispositivo em espera é significativamente menor. Com base nestas considerações, esse trabalho apresenta um gerenciamento dinâmico de potência que usa predição para compensar a latência de ativação dos dispositivos.

SOLUÇÃO APROXIMADA ONLINE

É comum, entretanto, que os sistemas não tenham sempre as mesmas características de distribuição dos dados de entrada, isto é, o comportamento do sistema não pode ser definido antes de sua execução. Assim, técnicas de redução de consumo com soluções obtidas em tempo de execução são abordadas, como a técnica apresentada por Dhiman e Rosing [11], que propõe

uma aprendizagem em tempo de execução de um sistema multi-tarefas tolerante a falhas, em que informações sobre o desempenho do sistema são coletadas. De acordo com o histórico de desempenho deste sistema de comportamento periódico, é possível escalonar a tensão/frequência de operação (DVFS) do sistema para que ele funcione de acordo com suas necessidades momentâneas, reduzindo o consumo de energia. A redução de consumo de energia chega a 49%, com uma redução significativa no sobre processamento causado pelo algoritmo, quando comparado às técnicas mais recentes de redução de consumo.

Pillai e Shin [31] fazem uso de um algoritmo de DVFS que altera o escalonador e o serviço de gerenciamento de tarefas do sistema operacional em sistemas embarcados de tempo real (Linux e Windows CE), de modo a garantir o cumprimento das restrições de tempo da aplicação e ainda reduzir o consumo de energia. Através de simulações e de um protótipo de sistema, são apresentados resultados para o algoritmo proposto que podem levar sistemas embarcados a uma redução de até 40% no consumo de energia, alcançando resultados próximos aos menores resultados teoricamente esperados.

Lee e Shin [23] propõem reduzir o consumo de energia através da divisão do tempo de execução disponível para uma tarefa analisando o tempo necessário para tratar as tarefas que são executadas com prioridade (preempção) durante o tratamento de uma tarefa periódica, em um sistema com possibilidade de escalonamento de tensão/frequência. O objetivo é alcançar uma tensão média intermediária para a execução da tarefa mantendo o tempo de execução dentro das restrições do sistema. O levantamento dos resultados foi realizado com um simulador em que qualquer valor de tensão e frequência pode ser utilizado para execução das tarefas. Aplicado a sistemas reais, pode apresentar uma solução em que o nível de operação esperado não está disponível, uma vez que sistemas reais contam com um número finito de pares tensão/frequência.

Sistemas com fontes de energia alternativas também são alvos de algoritmos de escalonamento de tensão e frequência. Liu *et al.* [25] combinam um algoritmo de DVFS a um escalonamento adaptativo de tarefas, para uso em sistemas de tempo real que fazem coleta de energia, como sistemas alimentados por energia solar. A proposta é utilizar o máximo tempo disponível para tratamento das tarefas para redução do consumo de energia. No entanto, em situações críticas de armazenamento, prazos de tratamento podem não ser respeitados. Em uma situação de capacidade mínima de energia armazenada, o algoritmo proposto reduz as perdas de prazo no tratamento de dados em pelo menos 20%, quando comparado a outros algoritmos de mesma função, mesmo sob utilização de vários processadores.

Rakhmatov e Vrudhula [33] consideram o perfil de carga e descarga da bateria que alimenta o sistema para realizar o escalonamento das tarefas. No trabalho de Schmitz *et al.* [37] é considerado o perfil de consumo de potência do elemento de processamento (PE, *Processing Element*), para a configuração da tensão/frequência de operação e o escalonamento de cada tarefa. Yan, Zhong e Jha [42][45] apresentam algoritmos de escalonamento de tensão/frequência baseados na percepção de latência pelo usuário em aplicativos interativos. O nível de tensão/frequência é ajustado para fornecer um desempenho de acordo com o tempo de resposta necessário para que o usuário não perceba “atrasos” na aplicação.

A possibilidade de trabalhar com a tolerância da aplicação a ocasionais falhas, garantindo uma qualidade mínima de serviço, combinada à análise dos resultados do tratamento de dados durante a aplicação, é amplamente utilizada na determinação do escalonamento de tensão e frequência para a execução da aplicação também em outros trabalhos.

Além do anteriormente citado trabalho de Dhiman e Rosing [11], o trabalho de Wang *et al.* [40] faz uso da tolerância da aplicação a ocasionais falhas e apresenta um algoritmo de DVFS que utiliza a carga de trabalho para tratamento de um *streaming* de vídeo para reduzir o consumo de energia. O algoritmo utiliza um módulo que monitora a carga de trabalho média e fornece estes dados a um módulo de DVFS, que controla a utilização dos processadores e altera os níveis de operação de acordo com os dados da carga de trabalho fornecidos. Os resultados são comparados a um algoritmo de tratamento que não utiliza DVFS e a outros dois algoritmos, chegando a reduções de consumo de 12,5%.

Yang e Song [44] propõem em seu trabalho um algoritmo de DVFS para *players* portáteis de vídeo. Através de uma regressão logarítmica, foi obtida uma relação entre o tamanho e o tempo de decodificação dos *frames* de vídeo. Baseado nestes modelos, os autores apresentam um algoritmo de seleção de nível de operação baseado em uma estimativa do tempo de decodificação dos *frames*, tolerante a algumas perdas de prazo de tratamento. Os resultados experimentais mostram redução no consumo de até 24% quando comparados a sistemas de decodificação sem o uso de DVFS.

Sassolas *et al.* [35] apresentam um algoritmo de escalonamento global *online* para aplicações de streaming. O algoritmo leva em conta as dependências de dados entre as tarefas no *pipeline* para otimizar o uso do processador e reduzir o consumo de energia usando os modos DVFS e DPM disponíveis no sistema. O Algoritmo de escalonamento usa o DVFS para fazer o balanceamento do *pipeline* da aplicação, aumentando ou diminuindo o tempo de execução das tarefas, e tratando as preempções do sistema, de modo a reduzir o consumo de energia. No entanto, é possível que alguns recursos do sistema fiquem ociosos devido às dependências de dados entre as tarefas do sistema. Neste ponto, o escalonador global usa o DPM para desligar estes recursos, reduzindo o consumo. Em uma plataforma virtual de simulação com 13 PEs, uma redução de 45% foi observada no consumo de energia, com alta taxa de utilização dos processadores, quando comparado ao sistema sem o uso do algoritmo.

3.2.3 Técnicas Ortogonais

Algumas técnicas de redução do consumo de energia são ortogonais a este trabalho, ou seja, podem ser aplicadas em conjunto com nossos algoritmos para reduzir ainda mais o consumo de energia.

Simunic *et al.* [38] apresentam uma metodologia de exploração do código fonte e otimização do compilador, que melhoram o desempenho do *software*, reduzindo o consumo de energia. A otimização é feita baseada em um modelo de consumo de energia com precisão de ciclo de execução, identificando seções críticas de consumo de energia do código, além de possibilitar a

avaliação do impacto de cada otimização realizada. Trata-se de uma ferramenta de levantamento do perfil do código, baseada em três categorias de otimização: alterações no algoritmo, alterações na representação dos dados e otimizações em nível de instrução. E esta ferramenta fornece ainda o percentual de tempo e consumo de energia de cada rotina dos componentes do sistema, levantando o perfil de consumo. Assim, é possível identificar as rotinas mais críticas do código, realizar alterações e verificar a eficácia, não só no consumo de energia do processador, mas também em memórias e barramento de comunicação.

Outro modo de reduzir o consumo é atuar diretamente na construção do sistema que vai executar uma aplicação. De acordo com Schaumont *et al.* [36], é possível reduzir o consumo de energia de um sistema que suporta uma aplicação quando a carga de trabalho gerada passa a ser dividida entre vários processadores. Para um sistema de leitura de impressões digitais, é proposta a substituição de um processador único por um sistema multiprocessador de menor consumo. A substituição por um sistema com dois processadores gerou uma redução de 12% no consumo de energia, com um tempo de tratamento 16% menor. Ao substituir o processador único por 4 processadores, houve um aumento no tempo de tratamento de apenas 2,2%, mas uma redução no consumo de energia de 77%. Do mesmo modo, esta abordagem pode ser aplicada sobre um sinal digital, que pode ser tratado por partes, em diferentes processadores. No mercado de computadores e servidores, esta substituição de um processador por um sistema multiprocessador é atualmente comum [2][19][20].

O anteriormente citado trabalho de Hwang *et al.* [16] descreve uma implementação, baseada em *hardware*, de uma unidade de gerenciamento de energia dedicada a recolher e analisar padrões associados ao acesso a dispositivos de entrada/saída. Pode ser usado ortogonalmente ao trabalho aqui apresentado.

Zhuo e Chakrabarti [47], entretanto, aplicam o escalonamento dinâmico de tensão e frequência para execução de tarefas no processador considerando o consumo de energia dos periféricos do sistema. Eles observam que caso uma tarefa tenha sua execução “esticada” para diminuir o consumo do processador durante sua execução, mas faça uso de um periférico de alto consumo de energia, o consumo geral do sistema pode aumentar ao invés de diminuir.

Bhatti *et al.* [5] propõem um esquema de gestão abrangente de energia, baseado em sistemas multiprocessador, nomeado Gerenciamento Híbrido de Potência (*Hybrid Power Management*), que utiliza técnicas disponíveis de DPM e DVFS e adapta-se, em tempo de execução, a qualquer carga de trabalho, utilizando-se da melhor política de DPM ou DVFS para o momento. A decisão de aplicar uma ou outra técnica é tomada por um algoritmo que avalia o desempenho das técnicas conhecidas e, partindo de um histórico de observação destas técnicas e da condição atual do sistema, define qual será a técnica utilizada.

Outros trabalhos consideram a tensão de limiar de polarização dos transistores no escalonamento de tensão, como no trabalho de Jejurikar e Gupta [21], que apresenta um algoritmo que levanta fatores de redução de velocidade de tratamento das tarefas baseado na contribuição da tarefa à corrente de fuga do processador e consumo em espera dos recursos do

sistema, além de procrastinar o tratamento de tarefas. Simulações apresentaram ganhos de até 15% sobre algoritmos tradicionais de DVFS.

Andrei *et al.* [3] trabalham sobre um sistema que possibilita a variação dos limiares de polarização, e propõem um algoritmo de redução de consumo *quasi-static* que faz uso de configurações definidas *offline*. O algoritmo utiliza os resultados de ajuste do nível de operação e do limiar de polarização dos transistores para alterar o sistema dinamicamente, de acordo com os tempos de execução observados para as tarefas do sistema, obtendo redução de consumo da ordem de 78% sobre a configuração nominal do sistema.

Kim e Roy [22] e Nose *et al.* [29] apresentam uma abordagem para tecnologias de construção de circuitos de pequenas dimensões ($< 70\text{ nm}$), com tensões de alimentação abaixo de 0,9V. São trabalhadas as tensões de limiar de polarização dos transistores ao invés dos níveis de operação do sistema, de modo a diminuir a corrente de fuga nos circuitos, chegando a resultados comparáveis aos obtidos por técnicas de DVFS. Como pode ser aplicada de forma independente, a alteração do limiar de polarização pode ser combinada a técnicas de DVFS, obtendo resultados ainda melhores na redução do consumo de energia [26].

3.2.4 Escalonamento de Tarefas

No citado trabalho de Lee e Kim [24], é apresentado um algoritmo de escalonamento em tempo de execução que simultaneamente seleciona os níveis de operação do processador e os períodos de execução das tarefas de controle, sempre considerando o consumo de energia de todo o sistema. Um índice de desempenho, considerando tanto os termos de energia do processador quanto dos sistemas de controle é proposto como base para execução do algoritmo do escalonador. Os resultados apresentados para duas variações do escalonador proposto mostraram um comportamento melhor que um escalonador denominado clarividente, porém com pior desempenho quando o escalonador clarividente tem dados ótimos para sua execução.

A análise do perfil de comportamento do sistema também pode ser levada em conta para a aplicação de algoritmos de DVFS. Yang *et al.* [43] propõem duas abordagens sobre padrões regulares de execução de tarefas de tempo real que tratam dados de entrada *multiframe*. A primeira abordagem é baseada na tarefa e aloca sempre o mesmo tempo de execução para instâncias de execução de uma mesma tarefa. A segunda abordagem, baseada no *frame* de dados recebido, consiste de uma fase *offline*, em que um vetor de velocidades de execução é definido para cada tarefa *multiframe* e uma fase *online*, em que uma política de escalonamento baseada em EDF (*earliest-deadline-first*) é usada. Os resultados mostram que as abordagens propostas usam menos espaço de memória e ocasionam menor sobre processamento, quando executadas *online*. No entanto, a redução no consumo de energia ficou aquém do resultado ótimo para o sistema.

Como proposto por Schaumont *et al.* [36], o gerenciamento do consumo de energia pode ser considerado desde a concepção do sistema, ao considerar o uso de um sistema multiprocessador para dividir a carga de tratamento dos dados de entrada recebidos. Além de

fazer esta consideração, o trabalho de Bilavarn *et al.* [7] enfatiza a necessidade de uma aplicação baseada em técnicas de gerenciamento do consumo de energia para implementar o novo padrão de serviços de vídeo embarcado H.264, além de apresentar uma implementação que explora tanto a execução paralela em uma plataforma multiprocessador quanto a aplicação de DVFS para o gerenciamento do consumo de energia.

3.3 Conclusão

Neste capítulo foram apresentados diversos trabalhos que fazem uso de variadas técnicas de redução de consumo de energia, principalmente técnicas baseadas em DVFS, que é base deste trabalho. Foram apresentados também trabalhos que fazem uso da qualidade mínima de serviço exigida pelo sistema para aplicação de técnicas de redução de consumo de energia, característica esta que também é utilizada neste trabalho.

Nos algoritmos baseados em DVFS apresentados neste trabalho, os ajustes de tensão/frequência acompanham a execução da aplicação. Os ajustes são feitos de acordo com as tarefas executadas e as entradas recebidas. Além disso, outras características se destacam: (a) existem quatro cenários aos quais podem ser aplicados, de acordo com a capacidade de DVFS e de monitoramento de qualidade, sendo possível utilizá-los em um grande número de sistemas (b) os algoritmos para os três cenários *offline* permitem a determinação rápida da configuração de tensão/frequência e podem ser usado repetidamente durante o projeto do sistema, (c) no quarto cenário, alterações dinâmicas na distribuição probabilística das classes de entrada são levados em conta, e (d) os nossos experimentos comparam resultados de distribuições estáticas e dinâmicas de classes de entrada, usando a variação de probabilidade de ocorrência da classe de entrada como parâmetro.

4 Especificação do Sistema

4.1 Introdução

Se uma aplicação exige grande desempenho para processamento digital de sinais, ou então uma grande velocidade de resposta na interface homem-máquina, o sistema embarcado onde esta aplicação vai ser executada deve ser capaz de realizar o processamento requerido e prover as condições necessárias para que a aplicação funcione a contento. Do mesmo modo, esta caracterização se torna necessária para identificar sistemas embarcados que podem ser trabalhados para, quando executando uma aplicação, reduzir o consumo de energia através do uso de algoritmos de redução de consumo.

Neste capítulo, serão descritos os sistemas embarcados sobre os quais os algoritmos desenvolvidos podem ser usados. As características necessárias à execução de cada algoritmo foram agrupadas em cenários de operação, que podem requerer desenvolvimentos específicos em *hardware* ou *software*.

4.2 Características do Sistema

O modelo de sistema de computação embarcada, sobre o qual os algoritmos desenvolvidos podem ser utilizados, é descrito através das suas características de funcionamento. A Figura 4.1 mostra um exemplo destes sistemas de computação embarcada.

Neste caso, várias tarefas de uma aplicação processam os *frames* de entrada da aplicação. Tarefas subsequentes, recebem como entrada os resultados de tratamentos provenientes de uma ou mais tarefas predecessoras, conforme definido por um Grafo de Tarefas, como mostrado na Figura 4.1(b). Estes grafos de tarefas são Grafos de Fluxo de Dados (DFGs – *Data Flow Graphs*) Acíclicos, com os nós representando as tarefas e as arestas representando as dependências de dados encontradas no tratamento dos *frames* de entrada.

Na execução de uma aplicação, suas tarefas são executadas respeitando a ordem parcial definida por seu DFG. Como os sistemas embarcados contam com um número limitado de processadores, é necessário definir o processador em que cada tarefa será executada. Assim, uma tarefa u_i é mapeada em um dos processadores PE_{ϖ} ($\varpi = 1, \dots, W$) disponíveis no sistema. No exemplo de sistema embarcado da Figura 4.1, a aplicação é executada em dois processadores ($W = 2$), como descrito na Figura 4.1(c). Quando as tarefas são mapeadas em processadores, o grafo de tarefa é modificado com a introdução de arestas de prioridade que definem as sequências de execução nos processadores. Assim, no novo grafo, uma aresta entre duas tarefas pode indicar não só uma dependência de dados, mas também uma sequência escolhida para execução das tarefas, que independe da existência de dependência de dados entre elas. De qualquer forma, uma aresta e_{ir} de u_i a u_r indica que u_i é executada antes de u_r .

Depois da introdução das arestas de prioridade, o grafo de tarefas é simplificado com a remoção de arestas redundantes. Uma aresta e_{ir} entre duas tarefas u_i e u_r é considerada redundante se existe uma sequência de arestas que vai de u_i a u_r sem passar por e_{ir} . Este novo grafo é chamado de Grafo Escalonado. No grafo escalonado da Figura 4.1(c), a aresta de u_3 a u_2 , que existia na Figura 4.1(b), foi removida.

Sobre este mesmo grafo da Figura 4.1(c), são obtidos os caminhos de execução do sistema, que definem a sequência de execução das tarefas da aplicação para tratamento dos dados de entrada recebidos. Estes caminhos seguem as arestas de prioridade definidas para o sistema.

Os algoritmos de DVFS desenvolvidos neste trabalho são aplicados sobre o grafo escalonado dos sistemas. A primeira etapa dos algoritmos de DVFS (*Dynamic Voltage Frequency Scaling*) desenvolvidos neste trabalho é obter um grafo de tarefas escalonado que descreva o sistema, utilizando como informações o grafo de tarefas e o escalonamento nos processadores.

O tratamento destes *frames* de entrada pode ser diferenciado para cada tarefa, separando estes *frames* em classes de entrada. Um *frame* de entrada de uma tarefa $u_i (i = 1, \dots, n)$ é classificado em uma das classes de entrada $c_{i,j} (j = 1, \dots, k_i)$, de acordo com seu tempo de execução na tarefa, com um maior valor de j correspondendo a um maior tempo de execução. A probabilidade de ocorrência de uma classe de entrada $c_{i,j}$ é indicada por $p(c_{i,j})$. Portanto, a soma das probabilidades correspondentes a todas as classes de entrada para a tarefa u_i é dada por $\sum_1^{k_i} p(c_{i,j}) = 1$. A probabilidade de uma tarefa u_i ter, para uma entrada, um tempo de execução menor ou igual ao tempo correspondente a uma entrada da classe $c_{i,j}$ é dada por $P_{i,j} = \sum_{t=1}^j p(c_{i,t})$. As probabilidades definidas para cada classe de entrada são mostradas na Figura 4.1(a).

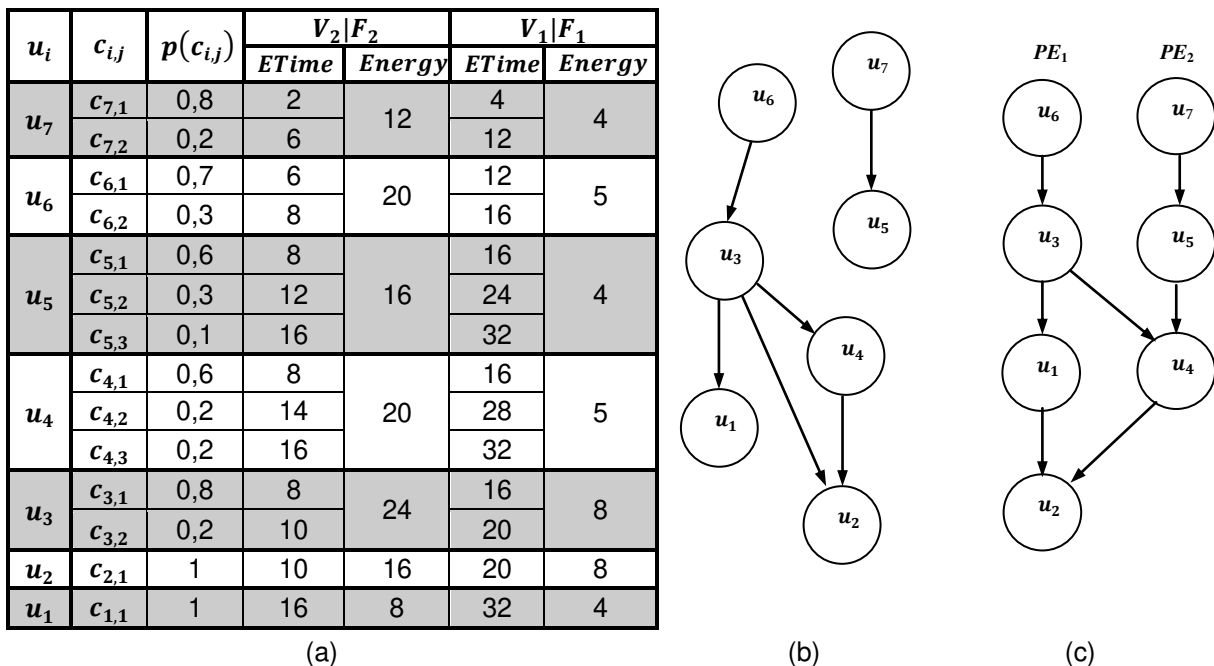


Figura 4.1: Exemplo de especificação do sistema com (a) tempo de execução, energia e distribuição das classes de entrada, (b) grafo de tarefas e (c) grafo escalonado de tarefas

Para permitir o escalonamento de tensão/frequência, um sistema embarcado pode contar com vários níveis de tensão/frequência de operação, onde uma maior tensão de operação possibilita uma maior frequência de trabalho, acelerando a execução das tarefas. Do mesmo modo, um valor de tensão menor reduz a velocidade de trabalho do sistema. Para os sistemas embarcados aqui tratados, estes níveis de tensão/frequência podem ou não ser ajustados durante a execução da aplicação, dependendo de seu cenário de operação. Estes cenários serão descritos na seção 4.3.

O par tensão/frequência do processador pode assumir L diferentes níveis de operação. O nível λ corresponde ao par $V_\lambda | F_\lambda$, sendo que $V_1 < V_2 < \dots < V_L$. O tempo de execução da tarefa u_i para processar uma classe de entrada $c_{i,j}$ em um processador operando no nível λ é representado por $ETime_\lambda(c_{i,j})$, e depende da classe de entrada em que o *frame* recebido se encaixa. Considerando que as classes com maiores índices correspondem a maiores tempos de execução, temos, para o nível λ , $ETime_\lambda(c_{i,1}) < ETime_\lambda(c_{i,2}) < \dots < ETime_\lambda(c_{i,k_i})$.

É também necessário descrever o modelo de consumo de energia para este sistema embarcado. A energia média consumida para tratamento das entradas da tarefa u_i , considerando que entradas de todas as classes são processadas, é representada por $Energy_\lambda(u_i)$, onde λ corresponde ao par $V_\lambda | F_\lambda$.

A energia média para a execução de uma classe de entrada é obtida calculando-se a potência média para a execução da tarefa e multiplicando esta potência média pelo tempo de execução da classe na tarefa, como mostrado nas seguintes equações.

$$Power_\lambda(u_i) = Energy_\lambda(u_i) / \left(\sum_{j=1}^{k_i} p(c_{i,j}) \cdot ETime_\lambda(c_{i,j}) \right) \quad (\text{eq.14})$$

$$Energy_\lambda(c_{i,j}) = Power_\lambda(u_i) \cdot ETime_\lambda(c_{i,j}) \quad (\text{eq.15})$$

Estas equações serão novamente referenciadas na próxima seção deste capítulo, para descrição do consumo de energia dos sistemas embarcados em cada cenário de operação.

A Figura 4.1(a) mostra os tempos de execução de classes de entrada e a energia média consumida para execução das entradas em uma tarefa, para dois níveis de operação de tensão/frequência. Este modelo é similar ao usado por Hua *et al.* [15] e Qiu *et al.* [32].

Consideramos ainda que a especificação do sistema inclua duas restrições para o correto funcionamento. O prazo M , especifica o tempo máximo para finalizar a execução do tratamento de cada *frame* de entrada e, usualmente, é consequência da taxa de recebimento destes *frames*. A segunda restrição é a qualidade de serviço Q , que especifica a fração mínima de *frames* de entrada para os quais o processamento deve ser completamente realizado dentro do prazo M . Em sistemas embarcados de vídeo e áudio, é comum o processamento de imagens e sons utilizar uma qualidade de serviço abaixo de 100% [15]. A maioria destes sistemas faz uso da mínima qualidade de serviço necessária para que um usuário comum não perceba, ou não se incomode, com falhas no tratamento. Segundo Hua *et al.* [15], uma taxa de perda de até 10% pode ser tolerada em pacotes de aplicações de áudio, enquanto a tolerância a perdas em pacotes de voz

com baixa taxa de transmissão, como uma conversa pela internet, pode ser ainda maior. De acordo com Yan, Zhong e Jha [42][45], um atraso de resposta de até 100 *ms* em aplicativos interativos, como *video games*, pode não ser percebido pelo usuário. Em outros tipos de interações, atrasos de resposta ainda maiores podem passar despercebidos ao usuário.

Assim, o modelo de sistema embarcado, sobre os quais os algoritmos serão utilizados, pode ser representado por um conjunto de grafos de tarefas mapeadas em um sistema multiprocessador e escalonadas, Figuras 4.1(b) e 4.1(c), por uma distribuição probabilística de classes de entrada, Figura 4.1(a), por um prazo M , e por uma taxa mínima de conclusão, ou qualidade de serviço Q .

4.3 Cenários de Operação e Configurações

A utilização dos algoritmos desenvolvidos pode ser feita em diversos tipos de sistemas de computação embarcada, e esta seção divide em 4 diferentes cenários o funcionamento destes sistemas, de acordo com a capacidade de configuração de DVFS e monitoramento da qualidade de serviço. Foram desenvolvidos algoritmos para cada um dos cenários de operação.

Para cada cenário, o algoritmo determina uma configuração de DVFS para o sistema, com o objetivo de satisfazer as restrições de prazo e qualidade de serviço, consumindo uma quantidade mínima de energia.

Os algoritmos desenvolvidos para os 3 primeiros cenários são executados *offline*, isto é, a configuração de DVFS para o sistema está pronta antes do início da aplicação. O algoritmo desenvolvido para o quarto cenário define, durante a execução da aplicação (*online*), a configuração de DVFS para o sistema.

Para o uso dos algoritmos nas aplicações, são necessárias algumas pequenas alterações nas tarefas e, quando a aplicação for executada sobre um sistema operacional, também este deve receber pequenas modificações. As necessidades de alteração serão descritas no próximo capítulo.

4.3.1 C1 - Escalonamento de tensão/frequência da Aplicação

Este primeiro cenário de aplicação foi incluído no trabalho para prover uma base de comparação com os algoritmos desenvolvidos para os outros cenários. Aqui supomos que níveis de tensão/frequência dos processadores podem ser configurados apenas no início da execução da aplicação e, além disso, um mesmo nível deve ser usado para todos os processadores. Todos os *frames* de entrada devem ser processados no prazo M , isto é, a qualidade de serviço é igual a 100% ($Q = 1$). A configuração do sistema para este cenário contém apenas o nível de tensão/frequência λ , a ser aplicado a todos os processadores, conforme mostrado na coluna “DVFS e Configuração de Entrada” da Tabela 4.1. Esta mesma tabela também apresenta as principais diferenças entre os 4 cenários, comentadas no decorrer da seção.

Tabela 4.1: Configuração do sistema para cada cenário

Cenário	Algo ritmo	Ajuste V/F	QoS	DVFS e Configuração de Entrada	Distribuição das entradas em classes
C1	SCN1	Aplicação	100%	λ	Estática
C2	SCN2	Tarefa	$\leq 100\%$	(u_i, l_i, λ_i) , para $i = 1, \dots, n$	Estática
C3	SCN3	Classe	$\leq 100\%$	$(u_i, l_i, (\lambda_{i,j}, j = 1, \dots, l_i))$, $i = 1, \dots, n$	Estática
C4	SCN4	Classe	$\leq 100\%$	$(u_i, l_i, (\lambda_{i,j}, j = 1, \dots, l_i))$, $i = 1, \dots, n$	Dinâmica

O algoritmo para este cenário inicialmente identifica todos os caminhos possíveis no grafo de tarefas escalonado. O tempo de execução de um caminho é calculado como

$$ETime_{\lambda}(Path) = \sum_{u_i \text{ on this Path}} ETime_{\lambda}(c_{i,k_i}) \quad (\text{eq.16})$$

Em seguida, é determinado o menor nível de tensão/frequência, comum a todos os processadores, para que o tempo de execução de nenhum caminho exceda M . Este nível de tensão/frequência determina a configuração inicial do sistema, a ser usada pela aplicação.

Para este cenário, o consumo de energia para tratamento dos *frames* de entrada é dada por

$$Energy(C1) = \sum_{i=1}^n Energy_{\lambda}(u_i) \quad (\text{eq.17})$$

Além do consumo de energia, outro parâmetro de comparação dos resultados obtidos pelos algoritmos é usado. Ao tentar reduzir o consumo de energia, os algoritmos “esticam” as tarefas, de modo que a execução seja feita em um nível de operação suficiente para cumprir a restrição de tempo M . Como consequência, a “taxa de utilização dos processadores” do sistema aumenta. Uma taxa de utilização mais alta significa que um algoritmo obtém uma melhora no compromisso entre tempo de execução e consumo de energia para os valores especificados de M e Q , deixando os processadores do sistema ociosos por menos tempo.

A taxa de utilização média dos processadores é calculada através da relação entre o tempo em que os processadores do sistema fazem o tratamento das tarefas e o tempo total disponível para execução das tarefas, M . São consideradas as tarefas em que ocorre descarte de classes de entrada quando operando nos cenários C2 e C3. A Taxa de Utilização Média dos processadores, para este cenário 1, é definida como

$$U(C1) = (1/W \cdot M) \sum_{i=1}^n \sum_{j=1}^{k_i} p(c_{i,j}) \cdot ETime_{\lambda}(c_{i,j}) \quad (\text{eq.18})$$

onde W é o número de processadores.

4.3.2 C2 - Escalonamento de tensão/frequência por Tarefa e Descarte de classes de entrada

Neste cenário, é possível configurar o nível de tensão/frequência de um processador, independentemente dos outros processadores, quando uma tarefa nele mapeada é iniciada. Além disso, classes de entrada podem ser identificadas e entradas de determinadas classes podem ser descartadas sem tratamento.

Uma configuração para estes segundo cenário tem o formato (u_i, l_i, λ_i) , para $i = 1, \dots, n$, onde o nível λ_i de tensão/frequência é definido para a execução da tarefa u_i e a classe de entrada l_i é a classe de maior índice, portanto, maior tempo de processamento, que não é descartada. Entradas de classes com tempo de processamento maior que c_{i,l_i} são descartadas sem tratamento. A descrição detalhada deste algoritmo é apresentada no próximo capítulo.

Para o cenário C2, as equações de consumo de energia são diferentes do cenário C1, devido ao descarte de classes de entrada e do uso de diferentes níveis de operação λ_i entre as tarefas e processadores, como descrito nas colunas “QoS” e “Ajuste V/F”, da Tabela 4.1. Assim, o consumo médio efetivo de energia de uma tarefa u_i , considerando o nível de tensão/frequência λ_i , e o descarte de entradas de classes com índices maiores que l_i , é dado por

$$EffEnergy_{\lambda_i, l_i}(u_i) = (Power_{\lambda_i}(u_i)/P_{i,l_i}) \cdot \sum_{j=1}^{l_i} (p(c_{i,j}) \cdot ETime_{\lambda_i}(c_{i,j})) \quad (eq.19)$$

onde $Power_{\lambda_i, l_i}(u_i)$ é calculado de acordo com (eq.14). A notação λ_i indica que o nível de tensão/frequência é definido apenas para a tarefa u_i .

A taxa efetiva de conclusão de tratamento dos *frames* de entrada, considerando que ocorre descarte de classes de entrada neste segundo cenário, difere do cenário C1, como mostrado na terceira coluna da Tabela 4.1. A taxa efetiva de conclusão para C2 é dada por

$$Q_{eff} = \prod_{i=1}^n P_{i,l_i} \quad (eq.20)$$

O consumo médio efetivo por *frame* de entrada é calculado levando em consideração a energia gasta com o tratamento de entradas por todas as tarefas (tratamento completo das entradas) e a energia gasta com entradas que foram processadas por algumas tarefas, mas que foram descartadas por alguma outra tarefa antes do tratamento estar completo. Assim,

$$Energy(C2) = Energy_{Complete} + Energy_{Partial} \quad (eq.21)$$

onde,

$$Energy_{Complete} = Q_{eff} \cdot \sum_{i=1}^n EffEnergy_{\lambda_i, l_i}(u_i) \quad (eq.21-1)$$

e

$$Energy_{Partial} = \sum_{\substack{\text{combinations} \\ \text{C}_y \text{ of } u_d}} \left[\left(Q_{eff} \cdot \prod_{u_d \text{ of } C_y} \frac{1-P_{d,l_d}}{P_{d,l_d}} \right) \cdot \sum_{\substack{u_i \text{ executed} \\ \text{when tasks of } C_y \\ \text{discard inputs}}} EffEnergy_{\lambda_i, l_i}(u_i) \right] \quad (eq.21-2)$$

onde u_d é a tarefa que descarta entradas, isto é, $l_d \neq k_d$.

Os caminhos de execução, definidos para cada aplicação, podem ou não ter tarefas em que *frames* de entrada podem ser descartados antes de seu tratamento ser iniciado. O primeiro termo de (eq.21), $Energy_{Complete}$, é a soma da energia consumida quando um *frame* de entrada é tratado por todas as tarefas dos caminhos de execução que não contém tarefas que fazem descartes de *frames* de entrada. Neste termo ainda é considerada a qualidade efetiva do sistema, Q_{eff} .

O segundo termo de (eq.21), $Energy_{partial}$, considera todas as combinações de descarte possíveis quando um ou mais caminhos de execução contam com tarefas que fazem descarte de *frames* de entrada. É considerado, para cada caminho, o consumo de energia usado no tratamento do *frame*, até que ele seja descartado. Este mesmo *frame* pode ser tratado em outros caminhos de execução que também façam seu descarte, ou em caminhos em que tarefas sucessivas possam descartá-lo, gerando uma combinação dos pontos de interrupção de tratamento. Todas estas combinações são consideradas no cálculo de consumo de energia e são representadas pelo segundo termo de (eq.21).

Também devido ao descarte e a diferença de níveis de operação entre os processadores, a utilização média dos processadores difere do cenário C1, e é dada por

$$U(C2) = (1/M \cdot W) \cdot \sum_{i=1}^n \sum_{j=1}^{l_i} p(c_{i,j}) \cdot ETime_{\lambda_i}(c_{i,j}) / P_{i,l_i} \quad (eq.22)$$

onde W é o número de processadores.

4.3.3 C3 - Escalonamento de tensão/frequência por Classe e Descarte de classes de entrada

Neste terceiro cenário, o nível de tensão/frequência é definido para cada classe de entrada a ser tratada pelo sistema, coluna "Ajuste V/F" da Tabela 4.1. Como em C2, entradas pertencentes a classes específicas podem ser descartadas. Uma configuração para o cenário C3 tem o formato $(u_i, l_i, (\lambda_{i,j}, \text{para } j = 1, \dots, l_i))$, para $i = 1, \dots, n$, onde l_i é a maior classe de entrada tratada pela tarefa u_i e $\lambda_{i,j}$ define o nível de tensão/frequência atribuído à tarefa u_i para tratamento de cada classe de entradas.

O algoritmo para este cenário é o mesmo usado para o cenário 2, acrescido de um tratamento final, aplicado às diferentes classes de entrada de uma mesma tarefa. Uma descrição mais detalhada desta etapa do algoritmo será apresentada no próximo capítulo.

Para este cenário, o consumo efetivo de energia da tarefa u_i , quando processando apenas classes de entrada menores ou iguais a l_i , com as classes de entrada $c_{i,j}$ sendo tratadas no nível de tensão/frequência $\lambda_{i,j}$, é dado por

$$EffEnergy_{\lambda_i, l_i}(u_i) = (1/P_{i,l_i}) \cdot \sum_{j=1}^{l_i} Power_{\lambda_{i,j}}(u_i) \cdot p(c_{i,j}) \cdot ETime_{\lambda_{i,j}}(c_{i,j}) \quad (eq.23)$$

onde $Power_{\lambda_{i,j}}(u_i)$ é calculado de acordo com (eq.14), considerando a execução da tarefa u_i no nível de tensão/frequência $\lambda_{i,j}$, para uma entrada de classe $c_{i,j}$. Aqui é considerado o consumo de energia para a execução de cada classe de entrada, agora levando em conta o tempo de tratamento e o nível de operação em que cada uma das classes é tratada, podendo ser diferentes para uma mesma tarefa.

Fazendo uso de (eq.23) para calcular $EffEnergy_{\lambda_i,l_i}(u_i)$, as equações (eq.20) e (eq.21) também se aplicam a este cenário. A utilização média do processador para o cenário 3, mostrada a seguir, é um pouco diferente, pois leva em conta que uma classe $c_{i,j}$ pode ser executada em um nível diferente de tensão/frequência $\lambda_{i,j}$.

$$U(C3) = \left(\frac{1}{W \cdot M}\right) \cdot \sum_{i=1}^n \sum_{j=1}^{l_i} p(c_{i,j}) \cdot ETime_{\lambda_{i,j}}(c_{i,j}) / P_{i,l_i} \text{ (eq.24)}$$

onde W é o número de processadores.

4.3.4 C4 - Monitoramento da distribuição das Classes de Entrada

Neste cenário 4, é assumido que o algoritmo para o cenário 3 é usado para encontrar a configuração inicial de DVFS de acordo com a distribuição inicial de classes de entrada. Para este novo cenário, a distribuição probabilística das entradas em classes pode ser alterada durante a execução da aplicação, abrindo a possibilidade de uso do algoritmo. Esta diferença para os cenários anteriores é apresentada na coluna “Distribuição das entradas em classes” da Tabela 4.1. É necessário monitorar a distribuição de entradas em classes para as tarefas que fazem descartes ($l_i \neq k_i$).

No entanto, para que o algoritmo do cenário 4 seja usado, o algoritmo do cenário 3 deve ser estendido para que algumas informações sejam coletadas e fiquem armazenadas para uso *online*. Mais detalhes sobre esta extensão do algoritmo *offline* do cenário 3 serão apresentados no próximo capítulo.

4.4 Conclusão

Foi apresentada neste capítulo uma descrição do modelo de sistemas embarcados abordado neste trabalho, assim como os possíveis cenários de operação definidos como base para o trabalho. As características e cenários de operação aqui descritos cobrem uma variada gama de sistemas embarcados, com diferentes capacidades de ajustes de tensão/frequência durante a execução das tarefas de uma aplicação, o que torna os algoritmos propostos uma solução a ser considerada na incessante busca da redução de consumo de energia.

5 Descrição dos Algoritmos

5.1 Introdução

Apresentados os cenários de aplicação do algoritmo, é necessário descrever os algoritmos desenvolvidos para cada cenário, visando soluções de configuração para redução do consumo de energia sem perda de eficiência de funcionamento. Além de descrever detalhes de implementação, são apresentados os cálculos de complexidade de cada algoritmo, mostrando a possibilidade de rápida convergência a uma solução, isto é, uma configuração de tensões/frequências para redução do consumo de energia do sistema.

5.2 Descrição dos Algoritmos

Os algoritmos desenvolvidos neste trabalho são usados em quatro cenários, que caracterizam a flexibilidade de uso de DVFS e a possibilidade de identificação e descarte de *frames* de entrada, como descrito na Tabela 4.1. Sucintamente, estes quatro cenários podem ser apresentados como:

- Cenário C1: Configurado em um nível de tensão/frequência mínimo em todos os processadores, igual para todas as tarefas de uma aplicação, suficiente para cumprir o tempo máximo M de execução, com QoS de 100% dos *frames* tratados.
- Cenário C2: O nível de tensão/frequência do processador pode ser alterado antes do início da execução de cada tarefa u_i e há possibilidade de descartar *frames* de classes de entrada maiores que l_i , sendo $l_i \neq k_i$.
- Cenário C3: É possível identificar e ajustar o par tensão/frequência de acordo com a classe do *frame* de entrada a ser tratado em cada tarefa. Portanto, uma mesma tarefa pode ser executada com tensões de operação distintas para entradas de classes distintas.
- Cenário C4: Há a possibilidade de alteração dinâmica nas configurações do sistema, de modo a tratar alterações na distribuição probabilística dos *frames* em classes de entrada durante a execução da aplicação.

O objetivo de cada algoritmo é determinar configurações do par tensão/frequência para execução das tarefas do sistema, de modo a reduzir o consumo de energia cumprindo, ainda, as restrições de qualidade e prazo impostas ao sistema. Esta configuração não só busca deixar o sistema dentro das restrições como também deixá-lo o mais próximo possível destes limites, visando o melhor aproveitamento dos recursos disponíveis. As atribuições de parâmetros às tarefas, como par tensão/frequência de execução e limite de tratamento de classes, além de outras especificidades de cada cenário, serão discutidas nas próximas seções.

Para aplicação dos algoritmos aos cenários descritos, é preciso reunir as informações que serão usadas como entradas para os algoritmos. São consideradas entradas a tabela de tempos de execução, consumo de energia e distribuição das entradas em classes (exemplo na Figura

4.1(a)), o mapeamento das tarefas nos processadores (exemplo na Figura 4.1(c)), o prazo M e a qualidade mínima de serviço Q_{MIN} . Além destas entradas, através do mapeamento das tarefas nos processadores, o algoritmo faz o cálculo para obtenção dos caminhos de execução da aplicação, inserindo arestas de prioridade, anteriormente descritas no item 4.2, que definem as sequências de tratamento das tarefas nos processadores, definindo os caminhos de execução para a aplicação.

A distribuição das entradas do sistema em classes é conhecida antes do início da execução da aplicação e é considerada fixa durante toda a execução *offline* dos algoritmos. Como saídas, os algoritmos fornecem a configuração do par tensão/frequência para execução das tarefas da aplicação, o tempo máximo para tratamento de um *frame* de entrada, respeitando o prazo M , o consumo médio de energia por *frame* e a taxa de utilização dos processadores, sendo os dois últimos calculados apenas para efeito comparativo.

Os caminhos de execução, calculados na primeira etapa de execução dos algoritmos, constituem todas as sequências de tarefas, construídas escolhendo um único sucessor para cada tarefa, de uma tarefa inicial até uma tarefa final. O *tempo de um caminho* é definido como a soma dos tempos de execução das tarefas deste caminho.

No cenário C4 é possível considerar variações *online* na distribuição de entradas em classes. A proposta é monitorar esta variação e adequar o funcionamento do sistema a esta nova distribuição, mantendo um consumo reduzido de energia, baseando-se em informações do sistema coletadas antes do início da execução da aplicação (*offline*).

Apenas sistemas do cenário C4 precisam de ajustes baseados na configuração inicial dos níveis de tensão/frequência durante a execução da aplicação, utilizando um algoritmo com etapas *offline* e *online*. Para os outros 3 cenários, o algoritmo conta apenas com etapas *offline*.

5.3 Algoritmo SCN1

Como comentado no capítulo anterior, o algoritmo SCN1 foi incluído neste trabalho para prover uma base de comparação com os algoritmos desenvolvidos para os outros cenários. Os níveis de tensão/frequência dos processadores para este cenário podem ser configurados apenas no início da execução da aplicação. Esta configuração inicial tem o objetivo de prover o menor valor de tensão/frequência, identificado pelo nível λ , comum a todos os processadores, para execução de todas as tarefas da aplicação, de modo a tratar todos os *frames* de entrada ($Q_{eff} = 100\%$) no prazo M .

Devido à sua baixa complexidade, o algoritmo para este cenário foi descrito no capítulo anterior em duas etapas simples de execução. Nesta seção, estas etapas são detalhadas.

Inicialmente o algoritmo identifica todos os caminhos no grafo de tarefas escalonado, exemplificado pela Figura 4.1(c). Através da tabela de tempo de execução, exemplificada pela Figura 4.1(a), e da equação para cálculo do tempo de execução dos caminhos (eq.16), rerepresentada a seguir, o algoritmo calcula o tempo de execução de cada caminho utilizando o menor nível de tensão/frequência λ_1 para todos os processadores.

$$ETime_{\lambda}(Path) = \sum_{u_i \text{ on } this \ Path} ETime_{\lambda}(c_{i,k_i}) \quad (\text{eq.16})$$

Em seguida, é verificado se o tempo de execução de todos os caminhos utilizando este nível de tensão/frequência é menor que M . Caso algum caminho exceda M , o par tensão/frequência é elevado ao próximo nível de operação λ_2 e o tempo de execução é novamente calculado. Este processo é repetido até que seja encontrado o menor nível λ_i , comum a todos os processadores, que respeite a restrição de prazo M , definindo a configuração de execução para o cenário C1.

5.4 Algoritmo SCN2

No cenário C2, para cada tarefa u_i , o nível de tensão/frequência do processador pode ser alterado antes do início da execução e é possível descartar entradas de classes superiores a l_i . O algoritmo utilizado é descrito pelo pseudocódigo da Figura 5.1.

Este algoritmo é baseado em caminhos de execução e o tempo de execução destes caminhos, calculado através de (eq.16), é comparado ao prazo M . A restrição de qualidade de serviço do sistema, Q_{MIN} , é verificada pela quantidade de *frames* de entrada completamente tratados dentro do prazo M . Para um melhor entendimento do algoritmo, alguns blocos principais de execução foram separados e detalhados de acordo com sua função principal.

5.4.1 Configuração Inicial

Esta configuração inicial é realizada nas linhas 1 a 3 do pseudocódigo da Figura 5.1. Inicialmente, a execução de todas as tarefas, em todos os processadores, é definida para ser realizada no menor nível de tensão/frequência do sistema ($\lambda_i = 1$), realizando ainda o tratamento de 100% dos *frames* de entrada ($l_i = k_i$), ou seja, $Q_{eff} = 1$.

A linha 4 do pseudocódigo traz o cálculo do parâmetro $FTask(u_i)$, que avalia quantas tarefas deixam de ser executadas se u_i descarta um *frame*. É definido por

$$FTask(u_i) = \text{numero de tarefas dependentes de } u_i / n \quad (\text{eq.25})$$

onde n é o número total de tarefas da aplicação.

Quando entradas de uma classe maior que l_i são descartadas por u_i , todas as tarefas dependentes que continuariam a tratar estas entradas não são executadas. Assim, para maximizar a economia de energia, um dos critérios usados pelo algoritmo para escolha da tarefa u_i em que pode ocorrer descarte de classes de entrada é o número de tarefas dependentes de u_i , quantificado pelo parâmetro $FTask$.

```

/* Entradas: Espec. do sistema, tempo máximo  $M$ , taxa mínima processamento de entradas  $Q_{MIN}$  */
1. Identifica todos os caminhos
2.  $\lambda_i = 1$ , para todo  $i$  (determina a menor tensão para todas as tarefas)
3.  $l_i = k_i$ , para todo  $i$ ,  $Q_{eff} = 1$  (sem descarte de classes de entradas)
4. Para cada  $u_i$ ,
   4.1  $FTask(u_i)$  = número de tarefas em caminhos dependentes /  $n$ 
5. Se ( $Q_{eff} = Q_{MIN}$ ), vai para 9

6. Identifica o conjunto de caminhos  $ED$ , isto é, caminhos com tempo de execução acima de  $M$ 
   6.1 Se não existem caminhos  $ED$ , finaliza o algoritmo, indo para linha 14
7. Para cada  $u_i$  com  $l_i \neq 1$  nestes caminhos
   7.1  $FP(u_i) = (P_{i,l_i-1}/P_{i,l_i})$ 
   7.2  $FT1(u_i) = (ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i}(c_{i,l_i-1}))$ 
8. Enquanto existe  $u_i$  tal que ( $u_i$  está num caminho  $ED$ ) AND ( $l_i \neq 1$ ) AND ( $Q_{eff} \cdot FP(u_i) \geq Q_{MIN}$ )
   8.1 Selecione  $u_i$  com o máximo valor para ( $FT1(u_i) \cdot FP(u_i) \cdot FTask(u_i)$ )
     8.1.1  $Q_{eff} = Q_{eff} \cdot FP(u_i)$ ,  $l_i = l_i - 1$ 
     8.1.2 Atualiza  $FP(u_i)$  e  $FT1(u_i)$ , Atualiza o conjunto de caminhos  $ED$ 

9. Identifica o conjunto de caminhos  $ED$ 
   9.1 Se não existem caminhos  $ED$ , vai para 14

10. Para cada  $u_i$  com  $\lambda_i < L$  nestes caminhos
    10.1  $FT2(u_i) = ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i+1}(c_{i,l_i})$ 
    10.2  $FE(u_i) = -EffEnergy_{\lambda_i,l_i}(u_i) + EffEnergy_{\lambda_i+1,l_i}(u_i)$ 
11. Enquanto existe  $u_i$  tal que ( $u_i$  está num caminho  $ED$ ) AND ( $\lambda_i \neq L$ )
    11.1 Selecione  $u_i$  com máximo valor de  $FT2(u_i)/FE(u_i)$ 
      11.1.1  $\lambda_i = \lambda_i + 1$ ,
      11.1.2 Atualiza  $FT2(u_i)$ , Atualiza  $FE(u_i)$ , Atualiza o conjunto de caminhos  $ED$ 

12. Se existe um caminho  $ED$ 
    (SEM SOLUÇÃO), Fim de  $SCN2$ .
    12.1 Senão (SOLUÇÃO): Configuração do Sistema é ( $u_i, l_i, \lambda_i$ ), para  $i = 1, \dots, n$ .

13. Se ( $Q_{eff} = Q_{MIN}$ ), vai para 16.

14. Para cada  $u_i$  com  $l_i \neq 1$ 
    14.1  $FP(u_i) = (P_{i,l_i-1}/P_{i,l_i})$ 
15. Enquanto existir  $u_i$  tal que ( $l_i \neq 1$ ) AND ( $Q_{eff} \cdot FP(u_i) \geq Q_{MIN}$ )
    15.1 Selecione  $u_i$  com máximo valor de ( $FP(u_i) \cdot FTask(u_i)$ )
      15.1.1  $Q_{eff} = Q_{eff} \cdot FP(u_i)$ ,  $l_i = l_i - 1$ 
      15.1.2 Atualiza  $FP(u_i)$ 

16. Configuração final do sistema para  $C2$ .
    A linha seguinte é usada apenas para avaliação de resultados. Não é parte do algoritmo:
    Calcula  $Energy(SCN2)$  e  $U(SCN2)$ .

```

Figura 5.1: Pseudocódigo para o algoritmo $SCN2$

5.4.2 Condição de descarte

A linha 5 do pseudocódigo da Figura 5.1 verifica se a qualidade requerida pelo sistema, Q_{MIN} , é igual à qualidade efetiva inicial atribuída ao sistema $Q_{\text{eff}} = 1$. Esta verificação é feita para evitar que o algoritmo inicie a busca por tarefas que farão descarte de classes de entrada, já que descartes não são necessários.

5.4.3 Identificação dos caminhos de execução

Quando um caminho de execução, na configuração atribuída pelo algoritmo, excede o prazo M , este caminho é chamado de *ED Path (Exceeded Deadline)* ou caminho *ED*. Esta identificação é feita nas linhas 6 e 9, e estes caminhos são utilizados em duas técnicas de redução de consumo de energia, combinadas no algoritmo proposto.

5.4.4 Descarte de classes de entrada

As linhas 6 a 8 do pseudocódigo da Figura 5.1 trabalham com a possibilidade de descarte de *frames* de entrada de determinadas classes em algumas tarefas. Depois de identificar os caminhos com tempo de execução acima do prazo M (linha 6), são calculados os valores dos parâmetros $FP(u_i)$ e $FT1(u_i)$ para cada uma das tarefas que estão nestes caminhos, linha 7:

$$FP(u_i) = (P_{i,l_i-1}/P_{i,l_i}) \quad (\text{eq.26})$$

$$FT1(u_i) = (ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i}(c_{i,l_i-1})) \quad (\text{eq.27})$$

onde P_{i,l_i-1} representa a probabilidade acumulada de execução da tarefa u_i com a redução do valor de l_i e P_{i,l_i} representa a probabilidade acumulada de execução da tarefa u_i na configuração atual, e os termos $ETime_{\lambda_i}(c_{i,l_i-1})$ e $ETime_{\lambda_i}(c_{i,l_i})$ representam, respectivamente, os tempos de execução nas duas configurações de probabilidade descritas.

O parâmetro $FP(u_i)$ quantifica a variação de qualidade de serviço atrelada ao descarte de *frames* de uma classe de entrada (redução da qualidade de serviço Q_{eff} para $l_i = l_i - 1$). O parâmetro $FT1(u_i)$ mostra a redução do tempo de tratamento, mantendo o mesmo nível λ_i de tensão/frequência, com o descarte de *frames* de uma classe de entrada da tarefa u_i ocasionado pela redução de l_i .

O produto dos parâmetros $FP(u_i)$, $FT1(u_i)$ e $FTask(u_i)$, constitui nosso critério de “custo/benefício” para seleção de uma tarefa de um caminho que excede M para reduzir o número de classes de entrada tratadas. Assim, enquanto existir uma tarefa u_i pertencente a um *ED path*, o valor de l_i para esta tarefa for diferente de 1, e a qualidade efetiva Q_{eff} ainda for maior ou igual a Q_{MIN} , linha 8, a tarefa u_i com maior valor de $FT1(u_i) \cdot FP(u_i) \cdot FTask(u_i)$, linha 8.1, é repetidamente selecionada para reduzir a classe máxima l_i de tratamento, linha 8.1.1, atualizando

os valores de $FP(u_i)$, $FT1(u_i)$ e os caminhos que excedem M , linha 8.1.2. Os tempos dos caminhos são novamente calculados devido aos novos tempos de execução das tarefas que tiveram seus valores de l_i reduzidos.

Vale destacar que nem sempre a mesma tarefa tem sua classe reduzida em sequência, pois a atualização dos valores dos parâmetros, linha 8.1.2, pode alterar a tarefa u_i com o melhor custo/benefício de maior redução de tempo de tratamento, mínima redução de qualidade e maior redução do consumo de energia.

Encontrada uma configuração para o sistema em que não existem mais caminhos que excedam M , a linha 9.1 completa a execução desta etapa do algoritmo, levando o algoritmo à etapa de Redução da Qualidade de Serviço, linhas 14-15, descritas posteriormente no item 5.4.6. Entretanto, é possível que apenas o descarte de classes não seja suficiente para encontrar uma solução de configuração para o sistema. Assim sendo, o algoritmo passa a trabalhar com a mudança do nível de operação para tratamento de cada tarefa.

5.4.5 Modificação do nível Tensão/Frequência de Operação

Esta técnica de redução do consumo de energia consiste em selecionar uma tarefa a ser executada em um maior nível de tensão/frequência para que o prazo M seja respeitado.

Caso seja verificada na linha 9 a existência de caminhos que excedem M , são calculados os parâmetros $FT2(u_i)$ e $FE(u_i)$, linhas 10.1 e 10.2:

$$FT2(u_i) = ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i+1}(c_{i,l_i}) \quad (\text{eq.28})$$

$$FE(u_i) = -EffEnergy_{\lambda_i,l_i}(u_i) + EffEnergy_{\lambda_i+1,l_i}(u_i) \quad (\text{eq.29})$$

onde $ETime_{\lambda_i}(c_{i,l_i})$ representa o tempo de execução da classe de entrada c_{i,l_i} em uma tarefa u_i executada no nível de tensão/frequência λ_i e $ETime_{\lambda_i+1}(c_{i,l_i})$ representa o tempo de tratamento da mesma classe de entrada no próximo nível superior de tensão/frequência, $\lambda_i + 1$. Os termos $EffEnergy_{\lambda_i,l_i}(u_i)$ e $EffEnergy_{\lambda_i+1,l_i}(u_i)$ representam o consumo de energia efetivo para execução da classe de entrada c_{i,l_i} no nível atual de operação λ_i e no próximo nível superior de operação $\lambda_i + 1$, respectivamente. O cálculo da energia efetiva é feito através de (eq.23), rerepresentada a seguir.

$$EffEnergy_{\lambda_i,l_i}(u_i) = (1/P_{i,l_i}) \cdot \sum_{j=1}^{l_i} Power_{\lambda_{ij}}(u_i) \cdot p(c_{i,j}) \cdot ETime_{\lambda_{ij}}(c_{i,j}) \quad (\text{eq.23})$$

O parâmetro $FT2(u_i)$ mostra a redução do tempo de tratamento de uma classe de entradas c_{i,l_i} quando tratadas no nível de tensão/frequência imediatamente superior ao nível atual. O parâmetro $FE(u_i)$ indica o aumento no consumo de energia com a alteração de nível de tensão/frequência para tratamento da classe de entradas c_{i,l_i} . Assim, a escolha da tarefa objetiva

maximizar a redução do tempo de execução (variável $FT2$) e minimizar o crescimento do consumo de energia (variável FE), ou seja, maximizar a fração $FT2(u_i)/FE(u_i)$.

Com os valores de $FT2(u_i)$ e $FE(u_i)$, é possível encontrar a tarefa u_i pertencente a um caminho que excede M que fornece a melhor relação “custo/benefício” ao ter seu nível de tensão/frequência λ_i alterado. Enquanto as restrições da linha 11 do pseudocódigo da Figura 5.1 forem respeitadas, é repetidamente selecionada a tarefa u_i com maior valor de $FT2(u_i)/FE(u_i)$, linha 11.1, que vai ter seu nível de tensão/frequência λ_i alterado para $\lambda_i + 1$ na linha 11.1.1. Após a alteração do nível de tensão/frequência de operação de u_i , a linha 11.1.2 faz a atualização dos parâmetros $FT2(u_i)$, $FE(u_i)$, e verifica novamente os caminhos que excedem M .

Se as restrições da linha 11 não mais são satisfeitas e ainda existem caminhos que excedem M para o sistema, a linha 12.1 finaliza o algoritmo, sem que uma solução tenha sido encontrada. No entanto, se foi encontrada uma configuração para o sistema em que não existem mais caminhos que excedam o prazo M , a linha 13 verifica se a qualidade de serviço mínima é igual à qualidade efetiva atual do sistema, $Q_{eff} = Q_{MIN}$. Se a aplicação exige a qualidade de serviço igual a Q_{eff} de tratamento dos *frames* de entrada, a solução foi encontrada. Entretanto, se a qualidade de serviço está acima da qualidade de serviço mínima definida para a aplicação, mesmo com uma configuração que satisfaz o prazo M , ainda é possível reduzir o consumo de energia para execução da aplicação.

5.4.6 Redução da Qualidade de Serviço

A aplicação requer que apenas uma fração Q_{MIN} dos *frames* de entrada seja completamente tratada antes do prazo M . É possível que neste ponto, os níveis de tensão/frequência atribuídos a cada tarefa propiciem a execução da aplicação dentro do prazo M , mas tratando mais *frames* de entrada do que o necessário. A etapa final do algoritmo, linhas 14-15, ainda pode reduzir o consumo de energia, verificando se existem tarefas u_i em que é possível descartar classes de entradas para reduzir l_i , aproximando Q_{eff} de Q_{MIN} e reduzindo o consumo.

Através dos parâmetros $FP(u_i)$ e $FTask(u_i)$, apresentados nesta seção, é possível selecionar, entre as tarefas do sistema, a tarefa com menor perda de qualidade e maior redução de consumo de energia, quando reduzido seu limite l_i para tratamento das classes de entrada, linha 14.1. Enquanto as restrições da linha 15 são respeitadas, a tarefa com maior valor do produto $FP(u_i) \cdot FTask(u_i)$ é repetidamente escolhida na linha 15.1. Para esta tarefa, é alterado o limite de tratamento de classe de entrada, $l_i = l_i - 1$, além da atualização correspondente da qualidade de serviço efetiva do sistema, $Q_{eff} = Q_{eff} \cdot FP(u_i)$, linha 15.1.1. Com o novo valor de l_i para u_i , o valor de $FP(u_i)$ é atualizado para a próxima iteração.

5.4.7 Resultados

Quando não é mais possível encontrar uma tarefa u_i para reduzir o limite l_i para processamento de classes de entrada, o algoritmo fornece a configuração λ_i dos processadores para o tratamento de cada classe de entrada, de acordo com l_i , em cada tarefa u_i nos caminhos de execução do sistema, assim como quais tarefas farão descarte de classes de entrada, definindo a configuração de execução para a aplicação, (u_i, l_i, λ_i) , para $i = 1, \dots, n$, na linha 16. A etapa final, que calcula o consumo de energia e a utilização dos processadores, é usada apenas para avaliação de resultados, não sendo parte do algoritmo.

Este mesmo cenário foi tratado por Qiu *et al.* [32], empregando um método de programação dinâmica, obtendo-se a solução ótima para o sistema. No entanto, para o exemplo apresentado no referido trabalho, é necessário um número de operações muitas vezes maior para uso do algoritmo proposto por Qiu *et al.*, quando comparado ao processamento exigido pelo algoritmo aqui apresentado.

5.5 Algoritmo SCN3

Para o cenário C3 é possível identificar e ajustar o par tensão/frequência de acordo com a classe do *frame* de entrada a ser tratado por uma tarefa. O algoritmo SCN3 utilizado para este cenário continua a ser representado pelo pseudocódigo da Figura 5.1 utilizado no cenário C2. É apenas necessário adicionar a este algoritmo a extensão, representada na Figura 5.2, que tem como objetivo definir, para a execução de uma classe de entrada $c_{i,j}$, $j \leq l_i$, o menor nível de tensão/frequência de modo que o tempo de processamento de $c_{i,j}$ não exceda o tempo de processamento de c_{i,l_i} . O intuito desta nova etapa é reduzir ainda mais o consumo de energia do sistema aproveitando a maior flexibilidade de DVFS do cenário C3. Uma solução para este problema, para um sistema com um único processador, foi proposta por Cao *et al.* [8] fazendo-se uso de Programação Linear.

Os dados de entrada do sistema estão distribuídos em classes de entrada $c_{i,j}$, $j \leq l_i$, diferenciadas pelo tempo de tratamento na tarefa u_i . Definidas as tensões de operação para cada tarefa através do algoritmo SCN2, é possível levantar o tempo máximo para execução de cada tarefa do sistema, determinado pelo tempo máximo de tratamento da classe de entrada c_{i,l_i} . Para as tarefas em que não há descarte de classes de entrada, ($l_i = k_i$), o tempo máximo para a execução da tarefa é o tempo de tratamento da classe c_{i,k_i} . Quando $l_i < k_i$, o tempo máximo de execução da tarefa é determinado pelo tempo de tratamento da classe c_{i,l_i} .

Entretanto, é possível que, com o tempo disponível para tratamento da classe de maior índice c_{i,l_i} , classes de entrada com menores índices ($c_{i,l_{i-1}}, c_{i,l_{i-2}}, \dots$) possam ser tratadas dentro deste mesmo tempo em níveis inferiores de tensão/frequência ($\lambda_{i-1}, \lambda_{i-2}, \dots$), reduzindo o consumo de energia. Assim, para o cenário C3, temos a nova etapa do algoritmo, descrita na Figura 5.2.

```

/* Executar SCN2 antes do código a seguir */
17. Para cada  $u_i$ 
    17.1 Para  $j = 1, \dots, l_i$ 
        17.1.1  $\lambda_{i,j} = \lambda_i$ 
        17.1.2  $temp = \lambda_i$ 
        17.1.3 Enquanto  $temp > 1$ 
            17.1.3.1 Se  $ETime_{temp-1}(c_{i,j}) \leq ETime_{\lambda_{i,l_i}}(c_{i,l_i})$ ,
                 $\lambda_{i,j} = temp - 1$ 
                 $temp = temp - 1$ 
18. A configuração do sistema é  $(u_i, l_i, (\lambda_{i,j}, j = 1, \dots, l_i)), i = 1, \dots, n$ . Fim de SCN3
A linha seguinte é usada apenas para avaliação de resultados. Não é parte do algoritmo:
Calcula  $Energy(SCN3)$  e  $U(SCN3)$ .

```

Figura 5.2: Pseudocódigo complementar do algoritmo SCN3

A linha 17 apresenta o primeiro laço de tratamento, envolvendo todas as tarefas da aplicação. A linha 17.1 mostra o segundo laço, onde todas as classes de entrada de cada tarefa são verificadas quanto à possibilidade de redução do nível tensão/frequência de operação. Assume-se que todas as classes estão sendo tratadas no mesmo nível λ_i , associado à tarefa u_i pelo algoritmo de SCN2, linha 17.1.1. A variável auxiliar $temp$ é utilizada para manter esta configuração durante os cálculos do algoritmo, linha 17.1.2. A linha 17.1.3 exibe o laço de verificação da possibilidade de redução do nível de tensão/frequência de operação para cada classe de entrada. Em busca do menor nível de operação, é verificado se o tempo de tratamento da classe de entrada $c_{i,j}$ em níveis inferiores a λ_i é menor que o tempo de tratamento da classe de entrada c_{i,l_i} , linha 17.1.3.1. Caso a condição seja verdadeira, o novo nível de tensão/frequência é atribuído ao sistema para tratamento da classe de entrada $c_{i,j}$. A verificação pode ser feita até que $temp$ tenha valor unitário. Como resultado, uma mesma tarefa pode ter um nível de tensão/frequência $\lambda_{i,j}$ para execução de cada classe $c_{i,j}$. O nível de tensão/frequência indicado por $\lambda_{i,j}$ indica a tensão j na qual o processador deve tratar a classe $c_{i,j}$.

A linha 18 atualiza a configuração para execução da aplicação, definindo um nível de tensão/frequência específico para tratamento de cada classe de entrada. Na linha seguinte, o cálculo dos valores de energia consumida por $frame$ e taxa de utilização dos processadores é feito apenas para avaliação dos resultados.

5.6 Algoritmo SCN4

No cenário C4 é possível avaliar a distribuição dos $frames$ de entrada durante a execução do sistema. O algoritmo SCN4 destinado a este cenário, altera a configuração do sistema em tempo de execução. É necessário que este algoritmo possa ser executado de maneira rápida e eficiente, garantindo a satisfação das restrições especificadas e mantendo o objetivo de redução no consumo de energia. Assumimos que a melhor configuração *offline* para o sistema foi encontrada anteriormente, através do algoritmo SCN3.

Dependendo do conhecimento a respeito das alterações na distribuição probabilística das classes de entrada, é possível definir o uso dos algoritmos para alguns sub-cenários do Cenário 4:

Cenário 4.1 - Modificações conhecidas na distribuição probabilística das classes de entradas. As configurações de funcionamento para o sistema para as distribuições conhecidas são definidas uma a uma pela aplicação do algoritmo *SCN3*. Dependendo do conhecimento a respeito do tempo no qual a variação na distribuição das entradas ocorre, dois novos cenários são observados:

Cenário 4.1.1 - As modificações na distribuição das classes de entrada são conhecidas e o momento em que estas mudanças ocorrem também é conhecido. Para este cenário, as configurações definidas pelo algoritmo de *SCN3* para cada “horário” são carregadas e o sistema é alterado para suprir a condição momentânea de funcionamento. Esta situação pode ocorrer quando um *smartphone*, durante o download de um aplicativo, apaga o display e desliga alguns periféricos, após um período de inatividade do usuário.

Cenário 4.1.2 - As modificações na distribuição das classes de entrada são conhecidas, mas o momento em que ocorrem não é definido. Neste cenário é necessária a inclusão da tarefa de reconhecimento dos *frames* de entrada para classificação das entradas e, quando reconhecida a nova distribuição, a configuração do sistema definida pelo algoritmo de *SCN3* para a nova distribuição das classes de entrada é carregada. Sensores remotos de monitoramento de presença (alarmes) podem alternar estados de monitoramento e transmissão de mensagens, respondendo a alterações nas classes de entrada que representam a detecção de movimentos.

Cenário 4.2 - Modificações na distribuição das classes de entrada não são conhecidas. Novas distribuições dos *frames* em classes de entrada serão tratadas pelo sistema de acordo com a identificação de cada uma delas. Neste caso, não é possível definir uma configuração através do algoritmo *SCN3*, sendo necessário um novo algoritmo, com atuação durante a execução da aplicação.

Este algoritmo para o cenário C4 está dividido em duas etapas, uma executada estaticamente (*offline*) e outra executada em conjunto com a aplicação (*online*). A etapa *offline*, executada ao final da execução do algoritmo *SCN3*, encontra uma “configuração estendida” do sistema definida por *SCN3*, que será utilizada no caso de alteração da distribuição das classes de entrada e conseqüente ativação da etapa *online* do algoritmo. O intuito da configuração estendida é possibilitar maior velocidade na reconfiguração do sistema quando ocorrer uma alteração na distribuição probabilística das classes de entrada.

5.6.1 Etapa Offline SCN4

O pseudocódigo da Figura 5.3 descreve a etapa *offline* de *SCN4*, que após a execução de *SCN3* encontra uma configuração estendida para o sistema. Como feito no início do algoritmo para

Algoritmo Offline

1. Identifica todos os potenciais caminhos ED (considerando execução das tarefas com $V_1|F_1$ e $Q_{\text{eff}} = 100\%$), seleciona somente os caminhos mais longos.
2. Determine para todo u_i , considerando a configuração de $SCN3$:
 - 2.1. $FTask(u_i)$
 - 2.2. $FP(u_i) = (P_{i,l_i-1}/P_{i,l_i})$
 - 2.3. $FT2(u_i) = ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i+1}(c_{i,l_i})$
 - 2.4. $FE(u_i) - EffEnergy_{\lambda_i,l_i}(u_i) + EffEnergy_{\lambda_i+1,l_i}(u_i)$
3. Determine para todo u_i com $(l_i \neq k_i)$, considerando a configuração de $SCN3$:
 - 3.1. $FP^+(u_i) = (P_{i,l_i}/P_{i,l_i+1})$
 - 3.2. $FT1^+(u_i) = ETime_{\lambda_i}(c_{i,l_i+1}) - ETime_{\lambda_i}(c_{i,l_i})$
4. Salva a *Configuração Estendida do Sistema* incluindo $FTask(u_i), FP(u_i), FT1^+(u_i), FP^+(u_i), FT2(u_i), FE(u_i)$

Figura 5.3: Pseudocódigo para o algoritmo *offline* $SCN4$

o cenário C2, na linha 1 são levantados os caminhos da aplicação que excedem M (ED paths), considerando a execução de todas as tarefas, em todos os processadores, no menor nível de tensão/frequência do sistema ($\lambda_i = 1$), realizando ainda o tratamento de 100% dos *frames* de entrada ($l_i = k_i$) assumindo assim uma qualidade de serviço $Q_{\text{eff}} = 1$. Como o prazo de execução M não é alterado, somente as tarefas nos caminhos excedem M são consideradas durante a execução do algoritmo.

A linha 2, no entanto, considera a configuração final obtida para o cenário C3 para continuação da execução desta etapa *offline*, calculando para todas as tarefas da aplicação os parâmetros $FTask(u_i), FP(u_i), FT2(u_i)$ e $FE(u_i)$, conforme definidos por $SCN3$. A linha 3, partindo também da configuração final obtida por $SCN3$, calcula dois novos parâmetros, $FP^+(u_i)$ e $FT1^+(u_i)$, linhas 3.1 e 3.2:

$$FP^+(u_i) = (P_{i,l_i}/P_{i,l_i+1}) \quad (\text{eq.30})$$

$$FT1^+(u_i) = ETime_{\lambda_i}(c_{i,l_i+1}) - ETime_{\lambda_i}(c_{i,l_i}) \quad (\text{eq.31})$$

onde P_{i,l_i+1} representa a probabilidade acumulada de execução da tarefa u_i considerando um possível aumento do valor de l_i e P_{i,l_i} representa a probabilidade acumulada de execução da tarefa u_i na configuração atual, e os termos $ETime_{\lambda_i}(c_{i,l_i+1})$ e $ETime_{\lambda_i}(c_{i,l_i})$ representam, respectivamente, os tempos de execução nas duas configurações de probabilidade descritas.

A linha 4 do pseudocódigo da Figura 5.3 salva a recém calculada “configuração estendida”, gerando uma base de resultados para realização da etapa *online* do algoritmo $SCN4$.

A escolha apenas dos caminhos mais longos que excedam M (linha 1), obtidos *offline*, foi motivada pela maior possibilidade de serem estes os caminhos afetados por alterações *online* na distribuição das classes de entrada e também para reduzir o universo de busca de tarefas durante

a execução da etapa *online*. Entretanto, *SCN4* ainda faz uso de todos os caminhos que excedem M , e os resultados apresentados no próximo capítulo foram obtidos sob esta condição.

Em uma próxima etapa de desenvolvimento deste algoritmo, está previsto o uso apenas dos caminhos mais longos que excedam M . Duas possibilidades de escolha levantadas foram a escolha dos caminhos mais longos, até que todas as tarefas do sistema estejam representadas nestes caminhos, ou então o uso de uma percentagem do total de caminhos que excedem M (50%, 30% ...), correndo-se o risco de problemas na exatidão dos resultados. A escolha da melhor possibilidade ou de outra que venha a ser levantada, será definida empiricamente.

5.6.2 Etapa Online SCN4

Durante a execução da aplicação, a distribuição probabilística das classes de entrada de tarefas que fazem descarte é monitorada. Este monitoramento é feito pelas tarefas que recebem os dados de entrada, contabilizando em cada classe de entrada os dados recebidos. Se ocorrerem mudanças nesta distribuição, e a qualidade de serviço ficar abaixo do valor mínimo especificado para o sistema, o algoritmo *online*, descrito pelo pseudocódigo da Figura 5.4, é iniciado pelas tarefas de monitoramento e procura por uma nova configuração para o sistema. Esta nova configuração é definida para que o sistema volte a cumprir as restrições de funcionamento e, ainda assim, possa continuar com um consumo reduzido de energia. O levantamento da configuração estendida durante a etapa *offline* do algoritmo visa à redução do tempo de busca por uma nova configuração *online*.

Portanto, o algoritmo *online* é executado quando a distribuição de classes de entrada para uma tarefa u_i que descarta classes se altera, como descrito na linha 1 da Figura 5.4. Esta nova distribuição, quando alterada em tarefas que descartam classes de entrada, modifica a Qualidade de Serviço do sistema. A nova qualidade efetiva de serviço Q_{eff} é calculada com base nos valores de l_i , linha 1.1.1, para todas as tarefas que fazem descarte de entradas e que tiveram a distribuição probabilística de suas classes de entrada alterada, linha 1.1. A linha 1.1.2 atualiza os valores de parâmetros $FP^+(u_i)$ e $FP(u_i)$ das tarefas com alteração na distribuição das classes de entrada.

Como se trata da adequação dos parâmetros à nova distribuição de classes de entrada, a linha 1.2 marca estas tarefas como “não visitadas” para que possam ser utilizadas no restante da execução do algoritmo. Todas as tarefas alteradas posteriormente pelo algoritmo são marcadas com “visitadas”.

Quando o valor de Q_{eff} fica abaixo do valor mínimo especificado, linha 1.3, são necessárias alterações na configuração do sistema para que a aplicação volte a respeitar as restrições de funcionamento. O algoritmo determina uma nova configuração de DVFS para o sistema fazendo uso de duas técnicas: (1) tarefas que descartam classes de entrada deixam de descartá-las e (2) o par tensão/frequência de algumas tarefas nos caminhos que excedem M é aumentado para reduzir o tempo de execução da tarefa e, conseqüentemente, do caminho em que a tarefa é executada. O laço da linha 1.3 engloba os dois tipos de alteração do sistema, e é executado até

que a aplicação volte a respeitar as restrições de funcionamento ou até que todas as possibilidades de alteração sejam exauridas e um alerta de qualidade de serviço abaixo do especificado seja dado na linha 1.4.

Embora a distribuição das classes de entrada se altere, o tempo de tratamento de cada classe em cada tarefa continua o mesmo. Com estas informações, o tempo de execução e a qualidade de serviço de cada nova configuração considerada pelo algoritmo *online* são verificados antes que a configuração do sistema seja alterada. Somente quando é encontrada uma configuração definida como solução para a nova distribuição das classes de entrada, o sistema é alterado. Para isso, uma “configuração de trabalho” ou “configuração sombra” é copiada da configuração atual, linha 1.3.1.

Através dos parâmetros levantados pela etapa *offline* do algoritmo, é possível encontrar a tarefa que faz descarte de classes com o “melhor custo/benefício” ao deixar de descartar classes de entrada, com aumento mínimo do tempo de tratamento e máximo aumento da qualidade de serviço, tendo ainda o menor número de tarefas sucessoras (baixo impacto no consumo de energia). Assim, na linha 1.3.2, é escolhida a tarefa com menor valor para a multiplicação $FT1^+(u_i) \cdot FP^+(u_i) \cdot FTask(u_i)$. Uma vez que a tarefa é selecionada, os parâmetros l_i e λ_{l_i} são atualizados na linha 1.3.2.1, a tarefa é marcada como “visitada” na linha 1.3.2.2, os parâmetros $FP(u_i)$, $FT2(u_i)$ e $FE(u_i)$ são atualizados em 1.3.2.3 e, caso u_i ainda descarte tarefas, são atualizados os parâmetros $FT1^+(u_i)$ e $FP^+(u_i)$, linha 1.3.2.4. Vale destacar que, como o algoritmo usa uma configuração de trabalho, alterada constantemente até chegar a uma solução para o sistema, marcar uma tarefa como “visitada” ou “não visitada”, possibilita ao algoritmo *online* uma maior velocidade de execução, não tendo que trabalhar novamente com tarefas “visitadas” em iterações anteriores, repetindo tentativas de configuração que não levaram a um bom resultado.

A linha 1.3.3 identifica, dentre os caminhos levantados na etapa *offline*, os caminhos que deixaram de cumprir o prazo M . No laço da linha 1.3.4, entre as tarefas u_i que podem ter seu nível de tensão/frequência aumentado e que estão nos caminhos que excedem M identificados na linha 1.3.3, é escolhida a tarefa u_j com o maior valor da fração $FT2(u_j)/FE(u_j)$. Esta tarefa tem o “melhor custo/benefício” para ter seu nível de operação aumentado, pois tem a melhor relação entre a redução do tempo de execução e o aumento de consumo de energia. Assim, a tensão/frequência de operação λ_{j,l_j} da tarefa u_j é aumentada na linha 1.3.4.1.1 e, caso o tempo de execução das classes de entrada de índices inferiores fique abaixo do tempo de execução para l_j , estas tarefas também tem seu par tensão/frequência de operação aumentada na linha 1.3.4.1.2. A linha 1.3.4.1.3 faz a atualização dos parâmetros $FT1^+(u_i)$, $FT2(u_i)$ e $FE(u_i)$, influenciados pela alteração de λ_{j,l_j} . A linha 1.3.4.2 faz a atualização dos caminhos que excedem M e, caso ainda exista algum caminho que não cumpra o prazo M , uma nova iteração do algoritmo é realizada, o pseudocódigo volta para a linha 1.3.1, e a configuração de trabalho é atualizada.

Caso não sejam identificados caminhos que excedam M nas linhas 1.3.3 ou 1.3.4.2, a sequência do pseudocódigo da Figura 5.4 leva à linha 1.3.5, onde a configuração do sistema é atualizada com a nova configuração de trabalho, calculada para que o sistema execute a aplicação dentro das restrições de funcionamento, com a nova distribuição de classes de entrada.

Algoritmo Online

1. Se (uma nova distribuição das classes de entrada é detectada)
 - 1.1. Para todo u_i com ($l_i \neq k_i$) AND alteração na distribuição de classes
 - 1.1.1. $Q_{eff} = Q_{eff} \cdot P^{new}(i, l_i) / P^{old}(i, l_i)$
 - 1.1.2. Atualiza $FP^+(u_i)$, $FP(u_i)$
 - 1.2. Para todo u_i com ($l_i \neq k_i$), Marca u_i com **não visitada**
 - 1.3. Enquanto ($Q_{eff} < Q_{MIN}$) e existe uma tarefa *não visitada*
 - 1.3.1. *Configuração Sombra = Configuração Estendida do Sistema*
 - 1.3.2. Selecione u_i **não visitada** com **mínimo** valor de ($FT1^+(u_i) \cdot FP^+(u_i) \cdot FTask(u_i)$)
 - 1.3.2.1. $l_i = l_i + 1$; $\lambda_{l_i} = \lambda_{l_i-1}$ (*)
 - 1.3.2.2. Marca u_i como **visitada**
 - 1.3.2.3. Atualiza $FP(u_i)$, $FT2(u_i)$, $FE(u_i)$ (*)
 - 1.3.2.4. Se $l_i \neq k_i$, atualiza $FT1^+(u_i)$, $FP^+(u_i)$ (*)
 - 1.3.3. Identifica o conjunto de caminhos *ED*
 - 1.3.4. Enquanto existir u_j num caminho *ED*, com ($\lambda_{j,l_j} \neq L$)
 - 1.3.4.1. Seleciona u_j com **máximo** $FT2(u_j) / FE(u_j)$
 - 1.3.4.1.1. $\lambda_{j,l_j} = \lambda_{j,l_j} + 1$ (*)
 - 1.3.4.1.2. Aumenta nível tensão/frequência das outras classes, se necessário (*)
 - 1.3.4.1.3. Atualiza $FT1^+(u_i)$, $FT2(u_i)$, $FE(u_i)$ (*)
 - 1.3.4.2. Atualiza o conjunto de caminhos *ED*
 - 1.3.5. Se não existem mais caminhos *ED*
 - 1.3.5.1. *Configuração Estendida do Sistema = Configuração Sombra*, etapas com (*) modificam a configuração do sistema
 - 1.3.5.2. $Q_{eff} = Q_{eff} / FP(u_i)$
 - 1.3.5.3. Se ($l_i \neq k_i$), Marca u_i como **não visitada** (novamente)
- 1.4. Se ainda existe um caminho *ED* (depois do laço 1.3)
 - 1.4.1. *Aviso: Q_{eff} abaixo do nível especificado*
- 1.5. Se ($Q_{eff} > Q_{MIN}$)
 - 1.5.1. Enquanto existe u_i com ($Q_{eff} \cdot FP(u_i) \geq Q_{MIN}$) AND ($l_i \neq 1$)
 - 1.5.1.1. Seleciona u_i com **máximo** ($FP(u_i) \cdot FTask(u_i)$)
 - 1.5.1.1.1. $Q_{eff} = Q_{eff} \cdot FP(u_i)$
 - 1.5.1.1.2. $l_i = l_i - 1$
 - 1.5.1.1.3. Atualiza $FP(u_i)$, $FT2(u_i)$, $FE(u_i)$

A linha seguinte é usada apenas para avaliação de resultados. Não é parte do algoritmo: Salva o valor final de Q_{eff} , calcula *Energy (SCN3)* e *U(SCN3)*.

Figura 5.4: Pseudocódigo para o algoritmo *online* SCN4

Outra situação é quando uma nova distribuição das classes de entrada aumenta o valor da qualidade de serviço da aplicação para um valor acima do necessário. Esta mesma condição pode ocorrer ao final do laço da linha 1.3, pois a configuração de trabalho do sistema foi alterada até que a condição $Q_{eff} < Q_{MIN}$ fosse falsa. Nestes casos, é possível considerar novos descartes de classes de entrada para reduzir a qualidade e economizar energia.

De acordo com a linha 1.5.1, enquanto existir uma tarefa que possa fazer descarte de classes de entrada e que o descarte não coloque Q_{eff} abaixo de Q_{MIN} , a linha 1.5.1.1 seleciona a tarefa com o “melhor custo/benefício” para a menor redução de Q_{eff} e descarte da maior quantidade de tarefas sucessoras, diminuindo o consumo de energia. Assim, é escolhida a tarefa u_i com o maior valor para a multiplicação $FP(u_i) \cdot FT_{\text{ask}}(u_i)$. Escolhida a tarefa que passa a descartar classe de entrada, são atualizados a qualidade de serviço efetiva Q_{eff} na linha 1.5.1.1.1, o limite de descarte l_i em 1.5.1.1.2 e os parâmetros $FP(u_i)$, $FT_2(u_i)$ e $FE(u_i)$ na linha 1.5.1.1.3, atualizando a configuração do sistema para a próxima iteração de descarte de classe de entrada.

A última etapa do algoritmo *online*, descrita na última linha do pseudocódigo da Figura 5.4, é realizada apenas para avaliação dos resultados, computando o valor final de Q_{eff} , o consumo de energia e a utilização dos processadores para a nova distribuição de classes de entrada para SCN4.

5.7 Alterações para execução dos algoritmos

Sistemas embarcados em que parte do processamento é dedicado ao tratamento de pacotes digitais são o foco de aplicação dos algoritmos apresentados. No entanto, estes sistemas devem ser levemente alterados para que possam executar os algoritmos propostos.

Nestes sistemas, *frames* de dados são recebidos em intervalos periódicos, e os dados recebidos são divididos em classes de dados de entrada. Um exemplo destes sistemas são os celulares, desde os mais básicos até os *smartphones*, em que os *frames* de dados são diferenciados dos de voz, através do meio de recepção e da identificação nos cabeçalhos dos *frames* recebidos. Mais ainda, através deste cabeçalho, é possível diferenciar dados de vídeo, imagens, sincronização de agendas e outros dados recebidos por um *smartphone*. O mesmo princípio de identificação pode ser aplicado aos *frames* tratados em um *notebook*, onde a configuração do sistema pode ser alterada de acordo com o tipo de entrada de dados identificada através dos cabeçalhos.

Os mais recentes codificadores de vídeo para *streaming* codificam *frames* próximos em grupos para reduzir a taxa de transmissão dos vídeos. No entanto, isto leva a formação de estruturas de grupos de imagens, onde alguns *frames* de vídeo precisam da reconstrução de vários *frames* intermediários para serem decodificados, enquanto outros *frames*, de menor “densidade de imagem” precisam da reconstrução de apenas alguns *frames* intermediários. Este comportamento do *streaming* de vídeo recebido altera o tratamento feito pelo sistema para cada tipo de *frame* de entrada, resultando em uma grande variação da carga de trabalho do sistema de decodificação.

Para todos os algoritmos propostos, as alterações necessárias em cada tarefa da aplicação dependem da implementação das tarefas no sistema. Ao mapear as tarefas da aplicação nos processadores do sistema, pode ser definida uma dependência entre duas tarefas que não tem dependência de dados, determinando uma aresta de prioridade. A tarefa sucessora deve então ser alterada para que, quando a tarefa predecessora descartar dados, esta tarefa não seja

executada e o caminho definido pela aresta de prioridade tenha sua execução interrompida, mesmo contando com tarefas que não dependem dos resultados de tarefas predecessoras. No entanto, esta alteração só é necessária quando o sistema embarcado estiver operando com um sistema operacional, que executa as tarefas liberadas para execução.

No caso de o sistema embarcado não utilizar um sistema operacional sendo a sequência de execução das tarefas decidida estaticamente, não existe necessidade de alteração das tarefas, já que o escalonamento nos processadores está definido diretamente e qualquer descarte interrompe o caminho de execução.

5.7.1 SCN2

Se a configuração definida pelo algoritmo *SCN2* para o Cenário 2 for armazenada na memória utilizada pelo processador, este cenário pode ser implementado através da modificação do sistema operacional, de forma que a tensão de operação seja ajustada, de acordo com a configuração, antes da execução de cada tarefa. A seção do código de iniciação de cada tarefa também deve ser modificada, de modo que as entradas de classes específicas sejam descartadas. Como alternativa, é possível alterar apenas os códigos de iniciação das tarefas, desde que o pedido ao sistema operacional para modificação do nível de tensão/frequência, de acordo com a configuração definida pelo algoritmo, esteja inserido nestes códigos de iniciação. Além disso, estas tarefas devem ser alteradas para obedecer ao grafo escalonado, onde as arestas de prioridade se sobrepõem às arestas de dependência.

5.7.2 SCN3

Assim como feito para *SCN2*, a configuração resultante da execução de *SCN3* para o cenário C3 é guardada na memória usada por cada processador, além da modificação da seção de iniciação de cada tarefa. Depois de identificada a classe de um *frame* de entrada, o código de iniciação pode descartar o *frame* ou pedir para que o nível de tensão/frequência do processador seja alterado para tratar este *frame*, de acordo com a tabela de configuração definida por *SCN3*. Novamente, as tarefas devem obedecer às arestas de prioridade em detrimento de suas dependências de tarefas predecessoras para início de execução.

5.7.3 SCN4

Além da armazenagem dos dados resultantes da execução de *SCN3* e da etapa *offline* de *SCN4*, a implementação deste cenário requer modificações no código das tarefas do sistema, para a monitoração da distribuição probabilística das classes de entrada. Qualquer alteração detectada dispara a execução da etapa *online* do algoritmo.

5.7.4 *Passo a Passo*

De modo geral, para uso dos algoritmos aqui apresentados, são necessárias poucas etapas de alteração do sistema, resumidas a seguir:

- Levantamento dos dados do sistema para uso dos algoritmos;
- Execução do algoritmo para obtenção das tabelas de configuração de funcionamento;
- Alteração no código do sistema para incluir:
 - ✓ Ajustes do nível de operação de acordo com a tarefa ou classe a ser executada
 - ✓ Descarte de classes de entrada
 - ✓ Monitoramento de alterações na distribuição das classes de entrada
- Geração do código do sistema, incluindo o algoritmo, para execução do sistema.

5.8 *Complexidade*

Em busca da redução de consumo sem perda de desempenho, é possível fazer uso de técnicas como a programação linear [8] ou a programação matemática dinâmica [32], obtendo a configuração ótima de tensão/frequência para o sistema. No entanto, para sistemas complexos, com grande número de tarefas com múltiplas tarefas ascendentes e descendentes e várias classes de entrada por tarefa, o tempo de processamento para se obter uma solução de escalonamento de tensão e frequência ideal pode ser muito grande, principalmente se ajustes forem necessários durante a execução da aplicação (*online*).

O algoritmo apresentado por Qiu *et al.* [32] para sistemas multiprocessador apresenta uma complexidade $O(K^{t+1} \cdot n^2 \cdot M \cdot L)$, onde K representa o maior tempo de execução das variações de uma tarefa e t representa o menor número de tarefas que tem mais de uma tarefa descendente ou mais de uma ascendente. A variável n representa o número de tarefas do sistema, M representa o tempo máximo de execução e L , a quantidade de níveis de tensão/frequência de operação.

Com o uso dos algoritmos propostos, de menor complexidade, embora uma solução ótima não seja obtida, os resultados alcançados mostram reduções de consumo de mais de 60% (*SCN3*) em relação a uma configuração mínima de tensão/frequência comum a todas as tarefas.

Para cada um dos cenários analisados, foram levantadas as complexidades de processamento.

5.8.1 *Complexidade do algoritmo SCN1*

Devido ao tratamento baseado em caminhos de execução, a complexidade do algoritmo para este cenário é $O(n \cdot L \cdot \#P)$, onde n é o número de tarefas do sistema, L , o número de níveis de tensão/frequência, e $\#P$, o número de caminhos.

5.8.2 Complexidade dos algoritmos SCN2 e SCN3

Considerando os *loops* das linhas 8, 10 e 11 na Figura 5.1, a complexidade deste algoritmo para o SCN2 é dada por $O(n^3 \cdot \#P \cdot \max(L, K))$, onde K é o número máximo de classes de entrada de um tarefa. A complexidade do cálculo de energia não é levada em conta, pois este cálculo não tem influência na determinação da configuração do sistema. Como SCN3 faz uso dos resultados de SCN2 e os cálculos exclusivos para o cenário 3 tem complexidade mais baixa, a complexidade para SCN3 é a mesma de SCN2.

5.8.3 Complexidade do algoritmo SCN4

A complexidade para a etapa *offline* do algoritmo para SCN4 é $O(n \cdot \#P)$, onde n é o número de tarefas do sistema e $\#P$ o número de caminhos.

Usualmente, o número de tarefas que descartam classes de entrada é muito pequeno quando comparado ao número total de tarefas do sistema. Por esta razão, consideramos que a complexidade do algoritmo *online* para SCN4 é determinada pela busca à tarefa u_j , linha 1.3.4. Esta complexidade é $O(n^2 \cdot \#P)$. Foram usadas várias estratégias para redução da complexidade, como a redução do número de caminhos considerados potenciais caminhos de não cumprimento do prazo de tratamento (*ED paths*) e determinação *offline* de listas de tarefas utilizadas *online* nas linhas 1.3.2, 1.3.4.1, e 1.5.1.1.

5.9 Exemplo

Fazendo uso do sistema exemplificado pela Figura 4.1, é possível aplicar os algoritmos apresentados, definindo configurações de operação do sistema nos três cenários *offline* levantados para este trabalho. No entanto, para uso dos algoritmos sobre o sistema exemplo, é importante conhecer duas restrições de funcionamento da aplicação: a qualidade mínima de serviço, Q_{MIN} , e o prazo máximo para tratamento dos *frames* de entrada, M . Para o exemplo, assumimos:

$$Q_{MIN} = 70\% \text{ e } M = 72 \text{ unidades de tempo.}$$

5.9.1 SCN1

Assumindo que a aplicação rode em um sistema que disponha de ajustes mínimos de DVFS, caracterizando o cenário C1 de operação, é preciso encontrar o menor par tensão/frequência, comum aos dois processadores do sistema (PE_1 e PE_2), que possibilite à aplicação o tratamento dos frames de entrada respeitando Q_{MIN} e M .

Seguindo o algoritmo para este cenário, é preciso calcular o tempo máximo de execução de cada caminho da aplicação. Observando a Figura 4.1(b), são 4 os caminhos de execução da aplicação:

$$u_6 > u_3 > u_1$$

$$u_6 > u_3 > u_2$$

$$u_6 > u_3 > u_4 > u_2$$

$$u_7 > u_5$$

No entanto, ao mapear as tarefas nos processadores, Figura 4.1(c), temos apenas três caminhos de execução para a aplicação, como descrito na Figura 5.5:

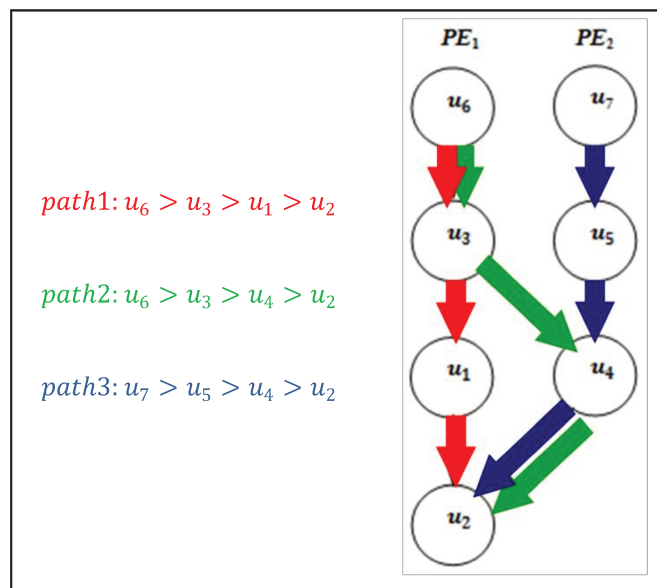


Figura 5.5: Caminhos de execução para a aplicação

Estes três caminhos de execução contêm os quatro caminhos de execução originais definidos para a Figura 4.1(b). No caso deste exemplo, a aresta entre u_3 e u_2 é redundante quando as tarefas são mapeadas nos processadores, pois a aresta de ligação entre u_3 e u_2 aparece implicitamente quando o caminho $u_6 > u_3 > u_1 > u_2$ é executado.

Considerando o menor par tensão/frequência de operação, e os tempos de execução neste nível de operação para tratamento de todas as classes de dados de entrada em cada tarefa do primeiro caminho, podemos calcular a quantidade de unidades de tempo (u.t.) necessária para execução do caminho. Para tratamento das classes $c_{6,1}$ e $c_{6,2}$ pela tarefa u_6 , operando em $V_1|F_1$, são necessárias 16 unidades de tempo. Para tratamento das classes $c_{3,1}$ e $c_{3,2}$ na tarefa u_3 , no mesmo nível $V_1|F_1$ de operação, são necessárias 20 unidades de tempo. Para tratamento da classe única $c_{1,1}$ na tarefa u_1 são necessárias 32 unidades e para tratamento da classe única $c_{2,1}$ na tarefa u_2 , 20 unidades. Assim, o tempo total para execução do primeiro caminho $u_6 > u_3 >$

$u_1 > u_2$ é de 88 unidades de tempo. Do mesmo modo, são calculados os tempos de execução para os outros dois caminhos:

$$u_6 > u_3 > u_4 > u_2 = 88 \text{ u. t.}$$

$$u_7 > u_5 > u_4 > u_2 = 96 \text{ u. t.}$$

Nenhum dos 3 caminhos respeita a restrição de tempo $M = 72 \text{ u. t.}$ Assim, é necessário alterar o par tensão/frequência de operação, comum a todas as tarefas, para $V_2|F_2$. Considerando que todas as classes de dados de entrada são tratadas, o tempo de execução para cada caminho é recalculado considerando o novo nível de operação:

$$u_6 > u_3 > u_1 > u_2 = 8 + 10 + 16 + 10 = 44 \text{ u. t.}$$

$$u_6 > u_3 > u_4 > u_2 = 8 + 10 + 16 + 10 = 44 \text{ u. t.}$$

$$u_7 > u_5 > u_4 > u_2 = 6 + 16 + 16 + 10 = 48 \text{ u. t.}$$

Operando com este segundo par tensão/frequência, $V_2|F_2$, todos os caminhos de execução são terminados dentro da restrição de tempo $M = 72 \text{ u. t.}$, definindo a solução do algoritmo para este sistema.

É então definida a configuração final do sistema para execução da aplicação: todas as tarefas operando no nível de tensão/frequência $V_2|F_2$, sem descarte de tarefas ($Q_{\text{eff}} = 1$).

5.9.2 SCN2

Considerando que o sistema tenha uma maior flexibilidade dos ajustes para DVFS, e exista a possibilidade de descarte de classes de entrada, é possível fazer uso de SCN2.

De acordo com o algoritmo descrito na Figura 5.1, o sistema é configurado para que todas as tarefas sejam executadas na tensão mínima de operação ($\lambda_i = 1$), tratando todas as classes de entrada ($l_i = k_i$). Em seguida, são identificados os caminhos do sistema mapeados nos processadores, já apresentados no item anterior, Figura 5.5.

$$\text{path1: } u_6 > u_3 > u_1 > u_2$$

$$\text{path2: } u_6 > u_3 > u_4 > u_2$$

$$\text{path3: } u_7 > u_5 > u_4 > u_2$$

Continuando a execução do algoritmo, são calculados os valores de F_{task} para todas as tarefas, de acordo com (eq.25).

$$F_{\text{task}}(u_1) = 0,143 \quad (\text{tarefa 2})$$

$$F_{\text{task}}(u_2) = 0 \quad (\text{nenhuma tarefa})$$

$$\begin{aligned}
F_{task}(u_3) &= 0,428 && \text{(tarefas 1, 2 e 4)} \\
F_{task}(u_4) &= 0,143 && \text{(tarefa 2)} \\
F_{task}(u_5) &= 0,286 && \text{(tarefas 4 e 2)} \\
F_{task}(u_6) &= 0,571 && \text{(tarefas 1, 2, 3 e 4)} \\
F_{task}(u_7) &= 0,428 && \text{(tarefas 4, 5 e 2)}
\end{aligned}$$

São então levantados os caminhos que excedem M . No caso desta aplicação, todos os caminhos excedem M .

$$path1: 16 + 20 + 32 + 20 = 88$$

$$path2: 16 + 20 + 32 + 20 = 88$$

$$path3: 12 + 32 + 32 + 20 = 96$$

São calculados então os valores de $FP(u_i) = (P_{i,l_i-1}/P_{i,l_i})$ e $FT1(u_i) = (ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i}(c_{i,l_i-1}))$ para todas as tarefas em que é possível descartar entradas:

$$\begin{aligned}
FP(u_3) &= 0,8 && FT1(u_3) = 4 \\
FP(u_4) &= 0,8 && FT1(u_4) = 4 \\
FP(u_5) &= 0,9 && FT1(u_5) = 8 \\
FP(u_6) &= 0,7 && FT1(u_6) = 4 \\
FP(u_7) &= 0,8 && FT1(u_7) = 8
\end{aligned}$$

Como as restrições da linha 8 do algoritmo *SCN2* (Figura 5.1) são respeitadas, são calculados os valores de cada tarefa para a multiplicação $FT1(u_i) \cdot FP(u_i) \cdot FTask(u_i)$, mostradas a seguir em ordem crescente:

Tarefa 4: 0,46

Tarefa 3: 1,37

Tarefa 6: 1,60

Tarefa 5: 2,06

Tarefa 7: 2,74

O maior valor foi obtido para a tarefa u_7 . Multiplicando $FP(u_7)$ pela qualidade efetiva atual do sistema, o resultado obtido ainda é maior que Q_{MIN} . Assim,

$$Q_{eff} = FP(u_7) \cdot Q_{eff} = 0,8$$

$$l_7 = l_7 - 1$$

A tarefa u_7 não pode mais fazer descartes ($l_7 = 1$). Os valores de $FT1$ e FP não são atualizados para u_7 e ela não é mais considerada para descarte.

O tempo máximo para tratamento de u_7 passou de 12 para 4 *unidades de tempo* (*u.t.*), como observado na Figura 5.6, uma vez que a tarefa passou a descartar a classe de maior tempo de tratamento. Como u_7 não faz parte de $path1$ e $path2$, apenas o tempo de $path3$ foi alterado:

$$path3: 4 + 32 + 32 + 20 = 88$$

Todos os caminhos ainda excedem o prazo máximo M .

Retornando à linha 8, a próxima tarefa a ter classes de entrada descartadas é u_5 , pois tem o maior valor da multiplicação $FT1(u_i) \cdot FP(u_i) \cdot FTask(u_i)$, uma vez que u_7 não pode mais descartar. O resultado de $Q_{eff} \cdot FP(u_5) = 0,72$ ainda é maior que Q_{MIN} , então u_5 passa a descartar classes de entrada ($l_5 = l_5 - 1$).

Como u_5 ainda pode aumentar a quantidade de classes descartadas, ($l_5 = 2$), são recalculados os valores $FP(u_5) = 0,67$ e $FT1(u_5) = 8$. Para u_5 , $FT1(u_5) \cdot FP(u_5) \cdot FTask(u_5) = 1,53$, considerando os novos valores $FP(u_5)$ e $FT1(u_5)$.

O tempo de tratamento de u_5 passou de 32 para 24 *u.t.*, como descrito na Figura 5.6. Novamente apenas o caminho $path3$ teve alteração no tempo de execução.

$$path3: 4 + 24 + 32 + 20 = 80$$

Todos os caminhos ainda tem tempo de execução acima de M .

Voltando à linha 8, nenhuma outra tarefa pode descartar classes de entrada sem que a qualidade efetiva Q_{eff} se torne menor que Q_{MIN} . Com os descartes apenas em u_7 e u_5 , o valor de Q_{eff} é 0,72.

u_i	$c_{i,j}$	$p(c_{i,j})$	$V_2 F_2$		$V_1 F_1$	
			<i>ETime</i>	<i>Energy</i>	<i>ETime</i>	<i>Energy</i>
u_7	$c_{7,1}$	0,8	2	12	4	4
	$c_{7,2}$	0,2	6		12	
u_6	$c_{6,1}$	0,7	6	20	12	5
	$c_{6,2}$	0,3	8		16	
u_5	$c_{5,1}$	0,6	8	16	16	4
	$c_{5,2}$	0,3	12		24	
	$c_{5,3}$	0,1	16		32	
u_4	$c_{4,1}$	0,6	8	20	16	5
	$c_{4,2}$	0,2	14		28	
	$c_{4,3}$	0,2	16		32	
u_3	$c_{3,1}$	0,8	8	24	16	8
	$c_{3,2}$	0,2	10		20	
u_2	$c_{2,1}$	1	10	16	20	8
u_1	$c_{1,1}$	1	16	8	32	4

Figura 5.6: Descarte de classes de entrada. Inicialmente u_7 , depois u_5

Como ainda há caminhos acima de M , o algoritmo passa à linha 10, onde são calculados os valores de $FT2(u_i) = ETime_{\lambda_i}(c_{i,l_i}) - ETime_{\lambda_i+1}(c_{i,l_i})$ e $FE(u_i) = -EffEnergy_{\lambda_i,l_i}(u_i) + EffEnergy_{\lambda_i+1,l_i}(u_i)$:

$FT2(u_1) = 16$	$FE(u_1) = 4$
$FT2(u_2) = 10$	$FE(u_2) = 8$
$FT2(u_3) = 10$	$FE(u_3) = 16$
$FT2(u_4) = 16$	$FE(u_4) = 15$
$FT2(u_5) = 12$	$FE(u_5) = 12$
$FT2(u_6) = 18$	$FE(u_6) = 15$
$FT2(u_7) = 2$	$FE(u_7) = 8$

Uma vez que as restrições da linha 11 de *SCN2* são respeitadas, são calculados os valores de cada tarefa para a divisão $FT2(u_i)/FE(u_i)$, mostradas a seguir em ordem crescente:

- Tarefa 7: 0,25
- Tarefa 6: 0,53
- Tarefa 3: 0,625
- Tarefa 5: 1,00
- Tarefa 4: 1,07
- Tarefa 2: 1,25
- Tarefa 1: 4

Assim, o maior valor para a divisão $FT2(u_i)/FE(u_i)$ é obtido por u_1 , sendo esta a tarefa escolhida para ter seu par tensão/frequência de operação alterado:

$$\lambda_1 = \lambda_1 + 1$$

O nível de operação de u_1 não pode mais ser aumentado, e esta tarefa é desconsiderada para próximas iterações, não sendo necessária a atualização dos valores de $FT2(u_1)$ e $FE(u_1)$. Com o aumento do nível tensão/frequência de operação, o tempo de execução de u_1 passa de 32 para 16 *u. t.*, como apresentado na Figura 5.7. No entanto, u_1 faz parte apenas de *path1*, que tem seu tempo de execução reduzido:

$$path1: 16 + 20 + \mathbf{16} + 20 = 72$$

Com esta alteração, *path1* respeita a restrição de tempo M , mas ainda há caminhos que excedem M .

u_i	c_{ij}	$p(c_{ij})$	$V_2 F_2$		$V_1 F_1$	
			<i>ETime</i>	<i>Energy</i>	<i>ETime</i>	<i>Energy</i>
u_7	$c_{7,1}$	0,8	2	12	4	4
	$c_{7,2}$	0,2	6		12	
u_6	$c_{6,1}$	0,7	6	20	12	5
	$c_{6,2}$	0,3	8		16	
u_5	$c_{5,1}$	0,6	8	16	16	4
	$c_{5,2}$	0,3	12		24	
	$c_{5,3}$	0,1	16		32	
u_4	$c_{4,1}$	0,6	8	20	16	5
	$c_{4,2}$	0,2	14		28	
	$c_{4,3}$	0,2	16		32	
u_3	$c_{3,1}$	0,8	8	24	16	8
	$c_{3,2}$	0,2	10		20	
u_2	$c_{2,1}$	1	10	16	20	8
u_1	$c_{1,1}$	1	16	8	32	4

Figura 5.7: Aumento do nível de operação. Inicialmente u_1 , depois u_2 e u_4

Então, de volta à linha 11, a tarefa com maior valor para a divisão $FT2(u_i)/FE(u_i)$, que pode ter o nível de operação aumentado, é u_2 . Assim,

$$\lambda_2 = \lambda_2 + 1$$

O nível de operação de u_2 não pode ser mais aumentado, e esta tarefa também é desconsiderada para próximas iterações, não sendo necessária a atualização dos valores de $FT2(u_2)$ e $FE(u_2)$.

Com esta alteração, o tempo para execução da tarefa u_2 foi reduzido de 20 para 10 *u. t.*, e o tempo de todos os caminhos é alterado:

$$path1: 16 + 20 + 16 + \mathbf{10} = 62$$

$$path2: 16 + 20 + 32 + \mathbf{10} = 78$$

$$path3: 4 + 24 + 32 + \mathbf{10} = 70$$

O caminho *path2* ainda é um caminho que excede M , então é necessária outra alteração nos níveis de operação.

De volta à linha 11, a tarefa com maior valor para a divisão $FT2(u_i)/FE(u_i)$, é u_4 e esta tarefa também está no caminho *path2*. Então,

$$\lambda_4 = \lambda_4 + 1$$

O nível de operação de u_4 chegou à maior configuração do par tensão/frequência, e esta tarefa não é mais considerada nos cálculos do algoritmo, dispensando a atualização dos valores de $FT2(u_4)$ e $FE(u_4)$.

Com a alteração do nível de operação, o tempo para execução da tarefa u_4 foi reduzido de 32 para 16 $u.t.$, e apenas o tempo de $path1$ continua sem alteração:

$$path2: 16 + 20 + \mathbf{16} + 10 = 62$$

$$path3: 4 + 24 + \mathbf{16} + 10 = 54$$

Com esta última alteração, todos os caminhos de execução da aplicação respeitam a restrição de tempo M . Como a qualidade efetiva Q_{eff} ainda é maior que Q_{MIN} , é possível buscar o descarte de classes de entrada, de modo a reduzir o consumo de energia.

De acordo com a linha 14, são novamente calculados os valores de $FP(u_i)$ para as tarefas que ainda possam descartar classes de entrada. Levando em conta os descartes realizados e as alterações nos níveis de tensão/frequência, os valores são:

$$FP(u_3) = 0,8$$

$$FP(u_4) = 0,8$$

$$FP(u_5) = 0,67$$

$$FP(u_6) = 0,7$$

No entanto, nenhuma das tarefas da aplicação que ainda podem fazer descarte de classes de entrada satisfaz a restrição do sistema $Q_{eff} \cdot FP(u_i) \geq Q_{MIN}$ e a execução do algoritmo é encerrada.

Assim, está definida a configuração do sistema para execução da aplicação, de acordo com a Tabela 5.1.

Tabela 5.1: Configuração do sistema para execução da aplicação (SCN2)

Tarefas	Classes	$V_\lambda F_\lambda$
7	$c_{7,1}$	$V_1 F_1$
	$c_{7,2}$	Desc.
6	$c_{6,1}$	$V_1 F_1$
	$c_{6,2}$	$V_1 F_1$
5	$c_{5,1}$	$V_1 F_1$
	$c_{5,2}$	$V_1 F_1$
	$c_{5,3}$	Desc.
4	$c_{4,1}$	$V_2 F_2$
	$c_{4,2}$	$V_2 F_2$
	$c_{4,3}$	$V_2 F_2$
3	$c_{3,1}$	$V_1 F_1$
	$c_{3,2}$	$V_1 F_1$
2	$c_{2,1}$	$V_2 F_2$
1	$c_{1,1}$	$V_2 F_2$

5.9.3 SCN3

Considerando possível configurar diferentes níveis de operação para tratamento das classes de entrada de uma mesma tarefa, caracterizando o cenário C3, o algoritmo *SCN3* pode ser usado.

Analisando-se as tarefas da aplicação, apenas a tarefa u_4 pode ser trabalhada por *SCN3*, pois $\lambda_4 \neq 1$ e $l_4 \neq 1$. As outras tarefas da aplicação ou não contam com um nível inferior de operação ou não fazem tratamento de diferentes classes de entrada.

O tempo máximo para tratamento de u_4 é de 16 *u.t.* com o processador operando em $V_2|F_2$. A classe $c_{4,2}$ tem tempo de tratamento de 14 *u.t.* e, operando no nível inferior de tensão/frequência, $V_1|F_1$ o seu tempo de tratamento passa a ser de 28 *u.t.*, superior ao tempo máximo de tratamento da tarefa.

A classe $c_{4,1}$ tem tempo de tratamento de 8 *u.t.* No entanto, o tempo de tratamento dos dados desta classe, se o processador estiver operando em $V_1|F_1$, é de 16 *u.t.*, tempo este que respeita a restrição de tempo para tratamento de u_4 . Assim, o processador que trata a tarefa u_4 pode ser configurado para operar em $V_2|F_2$ quando receber dados de entrada das classes $c_{4,2}$ e $c_{4,3}$ e operar em $V_1|F_1$ quando receber dados da classe $c_{4,1}$, como descrito na Figura 5.9.

Como resultado do algoritmo *SCN3*, a configuração para execução da aplicação é dada na Tabela 5.2.

u_i	c_{ij}	$p(c_{ij})$	$V_2 F_2$		$V_1 F_1$	
			<i>ETime</i>	<i>Energy</i>	<i>ETime</i>	<i>Energy</i>
u_7	$c_{7,1}$	0,8	2	12	4	4
	$c_{7,2}$	0,2	6		12	
u_6	$c_{6,1}$	0,7	6	20	12	5
	$c_{6,2}$	0,3	8		16	
u_5	$c_{5,1}$	0,6	8	16	16	4
	$c_{5,2}$	0,3	12		24	
	$c_{5,3}$	0,1	16		32	
u_4	$c_{4,1}$	0,6	8	20	16	5
	$c_{4,2}$	0,2	14		28	
	$c_{4,3}$	0,2	16		32	
u_3	$c_{3,1}$	0,8	8	24	16	8
	$c_{3,2}$	0,2	10		20	
u_2	$c_{2,1}$	1	10	16	20	8
u_1	$c_{1,1}$	1	16	8	32	4

Figura 5.8: Alteração do nível de operação da classe $c_{4,1}$

Tabela 5.2: Configuração do sistema para execução da aplicação (SCN3)

Tarefas	Classes	$V_\lambda F_\lambda$
7	$c_{7,1}$	$V_1 F_1$
	$c_{7,2}$	Desc.
6	$c_{6,1}$	$V_1 F_1$
	$c_{6,2}$	$V_1 F_1$
5	$c_{5,1}$	$V_1 F_1$
	$c_{5,2}$	$V_1 F_1$
	$c_{5,3}$	Desc.
4	$c_{4,1}$	$V_1 F_1$
	$c_{4,2}$	$V_2 F_2$
	$c_{4,3}$	$V_2 F_2$
3	$c_{3,1}$	$V_1 F_1$
	$c_{3,2}$	$V_1 F_1$
2	$c_{2,1}$	$V_2 F_2$
1	$c_{1,1}$	$V_2 F_2$

5.10 Conclusão

Este capítulo apresentou os algoritmos desenvolvidos para aplicação em 4 cenários distintos de operação, desde sistemas de segurança, rastreamento automotivo e celulares básicos, até *smartphones*, *tablets* e *notebooks*, desde que cada entrada recebida tenha sua identificação de classe. De modo geral, os algoritmos propostos são facilmente utilizados em sistemas que trabalham com processamento digital de sinais. Três destes algoritmos são algoritmos estáticos (*offline*), para aplicações que não tem alteração na distribuição probabilística das classes de entrada durante a execução. O quarto algoritmo tem execução dinâmica (*online*), desenvolvido para aplicações em que o recebimento de *frames* de entrada pode sofrer alterações na distribuição em classes de entrada.

Como descrito na seção 5.7, os algoritmos apresentados demandam uma quantidade relativamente pequena de alterações no código original da aplicação, tornando simples sua implementação, possibilitando assim a redução do consumo de energia.

A complexidade dos algoritmos é baixa, quando comparados a outros algoritmos que fazem uso de métodos matemáticos, podendo ser utilizados em sistemas com pouca capacidade de processamento ou, no caso do algoritmo *SCN4*, ser executado em conjunto com a aplicação, mantendo o baixo consumo de energia.

6 Resultados

6.1 Introdução

Este capítulo descreve as aplicações utilizadas para testes e validação dos algoritmos propostos, assim como os resultados experimentais obtidos por nossos algoritmos de DVFS. Foram considerados diversos casos de execução de aplicações reais e sintéticas, desde aplicações simples com apenas um caminho de execução e sem descartes até aplicações mais complexas, contando com dezenas de tarefas e milhares de caminhos de execução, respeitando restrições de qualidade de serviço.

Os resultados apresentados neste capítulo, quando comparados aos resultados obtidos para o cenário C1, mostram redução significativa do consumo de energia e um aumento da taxa de utilização dos processadores, sem perdas de qualidade nos resultados. A redução de consumo chega a mais de 70% para a aplicação GMDFa-1, dedicada ao cancelamento de eco acústico, com um aumento de 38% na utilização dos processadores. Para uma aplicação de codificação e decodificação de voz, Vocoder-2, a utilização dos processadores aumentou em 250%, chegando a uma redução de consumo também próxima a 70%. Como comentado anteriormente, o aumento da ocupação dos processadores não é objetivo dos algoritmos, mas aparece como uma forma de apresentar a configuração dos processadores para execução das tarefas da aplicação obtida pelos algoritmos, buscando a redução do consumo de energia.

6.2 Desenvolvimento

A codificação dos algoritmos foi realizada em linguagem C, sendo que a codificação dos algoritmos *SCN1*, *SCN2* e *SCN3* foi feita utilizando-se o programa Dev C++ [10] com compilador Gnu C. Esta compilação gerou um arquivo executável por linha de comando, com a utilização de dois arquivos de entrada de dados, gerando um arquivo de saída com as configurações *offline* para os três algoritmos. A Figura 6.1 mostra um diagrama simplificado do fluxo para execução dos algoritmos *offline*. A codificação do algoritmo *SCN4* foi realizada fazendo-se uso do programa LabWindows [28], da National Instruments, com geração de um executável com tela de interface para controle dos experimentos e captação dos resultados após aplicação de *SCN4*. No total, são pouco mais de 2000 linhas de código, sendo quase 1000 linhas para codificação dos algoritmos *SCN1*, *SCN2* e *SCN3* e pouco mais de 1000 linhas para *SCN4*, sendo que por volta de 200 linhas são dedicadas à etapa *offline* de *SCN4*.

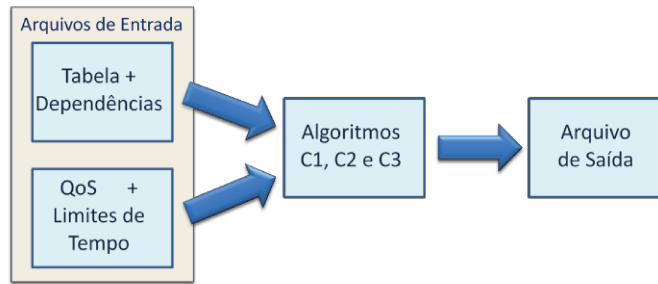


Figura 6.1: Fluxo para algoritmos *offline*

O Apêndice A2 traz mais informações sobre os algoritmos, com uma descrição mais detalhada dos vários blocos de código implementados, assim como exemplos dos arquivos de entrada utilizados para uso dos algoritmos nas aplicações e uma descrição mais precisa das etapas de experimentação realizadas para obtenção dos resultados apresentados neste capítulo. O email de contato para obtenção dos códigos para uso em trabalhos futuros também está disponível no Apêndice A2.

Devido à grande quantidade de casos de execução das aplicações de teste, foram utilizados diversos PCs, com sistemas operacionais Windows XP ou Windows 7. Utilizando um notebook com processador Intel Core i3 rodando a 2,53GHz, com 2 GB de memória RAM e sistema operacional Windows 7 de 64 bits, o tempo para obter os resultados de configuração para os Cenários 2 e 3 para a aplicação Vocoder-2 (133 tarefas) ficou em torno de 15 segundos, sendo que a extensão dos dados *offline* para *SCN4* levou mais 2 segundos, fazendo uso dos dados calculados por *SCN3*. Para obter uma nova configuração para o Cenário 4 para esta mesma aplicação, são necessários cerca de 2 segundos após a alteração da distribuição das classes de entrada. Maiores detalhes destes experimentos serão apresentados nos próximos itens e no Apêndice A2.

6.3 Aplicações

Os algoritmos propostos foram desenvolvidos para uso em aplicações em sistemas que precisam garantir uma qualidade de serviço mínima com baixo consumo de energia, por exemplo, para prolongar a duração da carga da bateria. Foram escolhidas cinco aplicações, sendo três delas reais e duas sintéticas. As aplicações reais consideradas são cancelamento acústico de eco, denominada GDMFa [6], processamento de dados de sonar, identificada apenas como Sonar [41] e codificação/decodificação de voz em um sistema GSM, tratada como Vocoder [14]. Mais duas aplicações, sintéticas, foram obtidas através do TGFF [12][34], identificadas por TGFF70 e TGFF99.

A Tabela 6.1 faz um resumo das configurações utilizadas para cada aplicação durante os experimentos de execução dos algoritmos. Nesta tabela estão descritos o número de nós e arestas do grafo de tarefas original, o número de processadores (PE - *Processing Element*) e o número de arestas e caminhos após o mapeamento e escalonamento. Nos gráficos de resultados,

Tabela 6.1: Aplicações

Aplicações (nós, arestas)	Nº PEs	Tempo de resposta (ms)	Após Escalonamento	
			Nº Arestas	Nº Caminhos
GMDFa (31, 53)	1	8,0	30	1
	2	7,0	46	4
	4	5,0	52	8
	8	3,5	52	16
Sonar (119, 116)	1	1350	118	1
	7	1350	93	150800
Vocoder (133, 135)	1	20,0	132	1
	2		165	131072
TGFF70 (70, 84)	1	3,0	69	1
	2	1,5	109	693
	4	1,0	129	1944
	8	0,6	129	573
TGFF99 (99, 100)	1	5,0	98	1
	2	2,5	156	6039
	5	1,3	170	3586
	8	1,1	197	10474

as aplicações são identificadas pelo nome e pelo número de processadores. Os processadores utilizados nos experimentos são sempre iguais entre si, caracterizando sistemas homogêneos.

Dependendo das arestas de dependência de uma aplicação e do número de processadores nos quais esta aplicação está mapeada, um maior número de tarefas pode ser executado em paralelo, diminuindo o tempo de resposta total da aplicação. Deste modo, o tempo de resposta definido originalmente pode ser longo o suficiente para que os algoritmos propostos encontrem sempre a mesma solução ótima, onde o menor nível de operação seja utilizado para todas as tarefas, com o mínimo consumo de energia. Assim, uma mesma aplicação, quando mapeada em diferentes números de processadores, pode ter atribuídos a ela diferentes tempos de resposta, como descrito na Tabela 6.1.

Por se tratarem de aplicações utilizadas em outros trabalhos para fins diversos, nem todos os dados sobre o funcionamento das aplicações estavam disponíveis. Além disso, para algumas aplicações, a distribuição das classes de entrada, ou a qualidade mínima de execução, ou mesmo o consumo de energia para execução das tarefas não estava disponível, necessitando de um complemento de informações para que se adequassem a um sistema embarcado real, como o descrito no Capítulo 4.

A distribuição em classes de entrada dos dados recebidos por cada aplicação foi feita baseando-se nos tempos de execução de cada tarefa das aplicações usadas como exemplo, onde o tempo original de execução da tarefa é o tempo para sua execução quando o processador está operando em um nível de tensão/frequência intermediário para cada aplicação, com tratamento de 100% das entradas de cada tarefa. As probabilidades de ocorrência de cada classe de entrada, variação de tempos em relação ao tempo original de cada tarefa para cada classe de entrada, além dos valores de consumo de energia atribuídos a cada tarefa, foram gerados aleatoriamente

para cada aplicação que não dispunha destes dados. No entanto, os limites desta aleatoriedade foram controlados, de modo que não existissem variações muito grandes dos valores de consumo de energia, tempo de execução e probabilidades de ocorrência das classes de entrada.

As tabelas de cada aplicação, no formato apresentado na Figura 4.1(a), com os tempos de execução e distribuição probabilística dos dados de entrada de cada tarefa, além do consumo de energia para tratamento das tarefas em cada par tensão/frequência de operação dos processadores do sistema, estão disponíveis no Apêndice A1, assim como os parâmetros de “aleatoriedade” que foram usados para obtenção dos dados não disponíveis em cada aplicação. As próximas seções deste capítulo apresentam as aplicações utilizadas para o desenvolvimento deste trabalho e os resultados obtidos.

6.3.1 Cancelador de Eco Acústico

Bianco *et al.* [6] utilizam um sistema de Cancelamento de Eco Acústico em seu trabalho para apresentar uma solução que define uma melhor partição entre estruturas de *hardware* e *software* para execução das tarefas da aplicação abordada, cumprindo as restrições de tempo de tratamento e reduzindo a área de *hardware* necessária. Emparamos a estrutura deste Cancelador de Eco Acústico, nomeado pelos autores como GMDFa, para uso como exemplo de aplicação real neste trabalho.

Identificado em nosso trabalho como GMDFa, este cancelador de eco acústico conta originalmente com 30 tarefas, distribuídas de acordo com o DFG apresentado no Apêndice A1, Figura A1.1, a serem executadas em até 8 ms, o tempo de resposta M imposta pela frequência de amostragem do sistema. Bianco *et al.* também apresentam o tempo de execução das tarefas correspondente a implementações em *software*, e foram estes os dados de tempo utilizados para construção das tabelas para aplicação do algoritmo. Os detalhes da construção destas tabelas de dados também são apresentados no Apêndice A1.

Para experimentação do uso dos algoritmos, a aplicação foi executada em 4 sistemas diferentes, contando com 1, 2, 4, ou 8 processadores. Os resultados dos experimentos serão identificados pela quantidade de processadores utilizados: GMDFa-1, GMDFa-2, GMDFa-4 e GMDFa-8. Uma tarefa extra foi adicionada, ao final do DFG do GMDFa apenas como ponto final de execução dos caminhos, para fins computacionais, passando o DFG a contar com 31 tarefas. Considerou-se que cada um dos processadores do sistema conta com três pares de tensão/frequência para execução de cada classe de entrada das tarefas da aplicação.

Devido ao mapeamento em mais de um processador, o tempo de resposta M foi mantido em 8 ms apenas para GMDFa-1. Para as outras variações da aplicação, este valor foi reduzido, pois a simples divisão das tarefas em um maior número de processadores já possibilitava ao sistema a execução das tarefas da aplicação dentro da restrição original de tempo $M = 8$ ms e a atuação do algoritmo seria apenas selecionar o menor nível tensão/frequência e calcular os descartes possíveis. Assim, os tempos de resposta para GMDFa-2, GMDFa-4 e GMDFa-8 são, respectivamente, $M = 7$ ms, $M = 5$ ms e $M = 3,5$ ms.

O trabalho de Bianco *et al.*, assim como todas as outras aplicações usadas como exemplo neste trabalho, não considera a distribuição de dados em classes de entrada ou o descarte de dados de entrada mas, como sistemas reais alimentados por bateria, como celulares, *notebooks*, etc., contam com diferentes classes de dados de entrada, consideramos neste trabalho a distribuição dos dados de entrada em até 5 classes, além de duas condições de qualidade de serviço mínima para os sistemas: $Q_{MIN} = 0,50$ e $Q_{MIN} = 0,70$.

A distribuição de dados em classes de entrada e a quantidade de classes de entrada são diferentes para cada uma das tarefas, pois um mesmo dado de entrada pode ser tratado de maneiras diferentes pelas tarefas. Neste exemplo, a quantidade de classes de entrada por tarefa varia de 1 a 4. A distribuição probabilística e as condições de aleatoriedade para geração destas distribuições são apresentadas no Apêndice A1.

Para a aplicação em questão, é considerado o tempo de comunicação entre as tarefas quando executadas em diferentes processadores. Estes tempos, também fornecidos no trabalho de Bianco *et al.*, foram somados ao tempo de execução da tarefa que recebe os dados.

Originalmente, o DFG desta aplicação conta com, no máximo, 8 tarefas descendentes de uma mesma tarefa e também com 8 tarefas que podem ser ascendentes a uma única tarefa. Com o mapeamento das tarefas nas várias quantidades de processadores propostas, esta ascendência e descendência de tarefas varia de 8 a apenas uma tarefa, para o caso do mapeamento em um único processador. De acordo com a quantidade de processadores, a aplicação GMDFa pode ter até 52 arestas sendo percorridas por 16 caminhos, como descrito na Tabela 6.1.

6.3.2 Sonar

Woodside e Monforton [41] apresentam um alocador estático de tarefas que pode ser usado para definir a alocação das tarefas (processos) de uma aplicação em um sistema de processamento distribuído. Em seu artigo, é apresentado um sistema real de processamento digital de sinais (DSP) para tratamento de dados de sonar, sobre o qual são apresentados os resultados do trabalho. Este exemplo de aplicação real também será usado em nosso trabalho, e será identificado como Sonar.

O DFG desta aplicação conta com 119 tarefas e é apresentado no Apêndice A1, Figura A1.2. A distribuição de tarefas em diferentes números de processadores também é fornecida por Woodside e Monforton, sendo que utilizaremos como exemplo neste trabalho as distribuições sobre 1 e 7 PEs, identificados como Sonar-1 e Sonar-7, respectivamente. A restrição $M = 1350\ ms$ é a mesma para as duas distribuições, e sua definição também foi baseada nos dados do artigo, que indicam os tempos de resposta de $2205,4\ ms$ e $1728\ ms$ para as distribuições das tarefas sobre 1 e 7 PEs, respectivamente.

Também para esta aplicação, foram considerados processadores com três pares de tensão/frequência de operação, com possibilidade de configuração para tratamento de cada classe de entrada de uma mesma tarefa em níveis diferentes de operação.

A distribuição probabilística de entradas conta com até 4 classes para uma mesma tarefa. As tabelas de distribuição e os critérios de aleatoriedade para obtenção dos dados estão dispostos no Apêndice A1. Como citado, consideramos para os experimentos duas qualidades mínimas de serviço para o sistema: $Q_{MIN} = 0,50$ e $Q_{MIN} = 0,70$.

Originalmente, o DFG desta aplicação conta com, no máximo, 4 tarefas descendentes de uma mesma tarefa e 3 tarefas que podem ser ascendentes a uma única tarefa. De acordo com a quantidade de processadores, a aplicação Sonar pode ter até 118 arestas, e ter sua execução em mais de 150.000 caminhos, como descrito na Tabela 6.1.

6.3.3 **Vocoder GSM**

Um sistema de codificação/decodificação de voz, conhecido como Vocoder, foi proposto por Gerstlauer *et al.* [14], baseado no padrão de comunicação de telefonia celular GSM Europeu. O vocoder foi projetado a partir de uma metodologia proposta na *University of California, Irvine*, onde uma especificação ainda abstrata do sistema é escrita na linguagem SpecC [46] e, partindo do executável desta especificação, é possível considerar diferentes alternativas de arquitetura para o sistema (componentes e comunicação), refinando gradualmente a especificação do vocoder até mapeá-la em uma implementação final de *hardware* e *software*. Esta implementação tem o intuito de encontrar a melhor relação entre implementações em *software* e espaço de silício utilizado para *hardware*, satisfazendo otimamente as restrições de funcionamento.

A estrutura deste Vocoder também foi utilizada para aplicação dos algoritmos propostos neste trabalho mas, diferente das duas aplicações descritas anteriormente, o trabalho de Gerstlauer *et al.* não descreve um DFG para o sistema. No entanto, os dados do trabalho tornam possível a montagem deste DFG, que foi feita pela aluna de Mestrado da UNICAMP, Daiane Angolini, que também utiliza este exemplo de aplicação real em seu trabalho [4]. O resultado deste excelente trabalho de levantamento de dados está na Figura A1.3 do Apêndice A1, que descreve o DFG do Vocoder.

Esta aplicação conta com 133 tarefas em seu DFG e o maior número de tarefas que uma tarefa tem como descendentes são duas, sendo que nenhuma tarefa do sistema tem mais que duas tarefas ascendentes. Apenas duas distribuições em processadores foram consideradas, utilizando-se sistemas com 1 e 2 processadores. As variações correspondentes foram denominadas Vocoder-1 e Vocoder-2.

O intervalo de tempo mínimo entre amostragens de sinal definida pelo padrão GSM Europeu, $M = 20\text{ ms}$, foi utilizado para as duas variações da aplicação. O sistema novamente é homogêneo em relação a seus processadores, mas agora com 4 pares de tensão/frequência de operação, também configuráveis em diferentes níveis de operação, para execução de classes de entrada de uma tarefa.

A distribuição probabilística de dados se divide em 3 classes de entrada para cada tarefa do sistema, sendo que as tabelas de distribuição e os critérios de aleatoriedade para obtenção dos dados podem ser encontrados no Apêndice A1. Os dados para definição dos tempos de execução

e consumo de energia de cada tarefa também foram obtidos de tabelas do trabalho de Gerstlauer *et al.* [14]. Estas tabelas e a forma de obtenção destes dados também estão no Apêndice A1.

Para esta aplicação, consideram-se novamente as duas qualidades mínimas de serviço, $Q_{MIN} = 0,50$ e $Q_{MIN} = 0,70$. As variações desta aplicação trabalham com até 165 arestas e mais de 130.000 caminhos de execução, como descrito na Tabela 6.1.

6.3.4 TGFF

Rodhes e Dick [12][34], apresentaram um gerador de grafos de tarefas que possibilita, a desenvolvedores e pesquisadores, a obtenção de “sistemas” sintéticos para uso em seus trabalhos. Esta ferramenta é conhecida por TGFF (*Task Graph for Free*) e fornece, além do grafo de tarefas, a lista das arestas entre as tarefas e dados específicos para cada tarefa que, no caso do nosso trabalho, foram interpretados como o tempo de execução de cada tarefa. A ferramenta possibilita, inclusive, definir os limites de quantidade de tarefas ascendentes e descendentes que estarão presentes no grafo gerado.

Foram geradas duas “aplicações” sintéticas para obtenção de resultados com o uso dos algoritmos propostos neste trabalho. A primeira delas, nomeada de TGFF70, conta com 70 tarefas e um limite máximo de duas tarefas ascendentes para cada tarefa e a mesma quantidade de tarefas descendentes para uma única tarefa.

Assim como nas aplicações reais, alguns sistemas foram considerados, contando com 1, 2, 4 e 8 processadores. Temos então TGFF70-1, TGFF70-2, TGFF70-4 e TGFF70-8. Os processadores são iguais, e contam com três pares de tensão/frequência de operação, possibilitando a configuração de diferentes níveis de operação para as diferentes classes de entrada de uma tarefa. Para esta aplicação com 70 tarefas, a distribuição probabilística dos dados de entrada é feita em até 3 classes para cada tarefa e os dados para geração destas tabelas, assim como modelos dos arquivos para geração dos exemplos através do TGFF estão dispostos no Apêndice A1.

Através do TGFF também são gerados limites para tempo de resposta da “aplicação”, definida para o TGFF70 como $M = 1000 u. t.$ (unidades de tempo) mas, para completa exercitação dos algoritmos, esta restrição de tempo foi assumida somente para TGFF70-4 e as restrições de tempo para TGFF70-1, TGFF70-2 e TGFF70-8 assumiram os valores $M = 3000 u. t.$, $M = 1500 u. t.$ e $M = 650 u. t.$, respectivamente.

Para o experimento com esta aplicação, repetem-se as duas qualidades mínimas de serviço, $Q_{MIN} = 0,50$ e $Q_{MIN} = 0,70$, para os quatro casos de distribuição nos processadores. A aplicação conta com até 129 arestas e quase 2.000 caminhos de execução, como descrito na Tabela 6.1.

A outra “aplicação” gerada pelo TGFF foi denominada TGFF99 e conta com 99 tarefas. Do mesmo modo que TGFF70, tem um máximo de duas tarefas ascendentes para cada tarefa e a mesma quantidade de tarefas descendentes para uma única tarefa.

O mapeamento das tarefas em sistemas homogêneos com 1, 2, 5 e 8 processadores é indicado por TGFF99-1, TGFF99-2, TGFF99-5 e TGFF99-8. Novamente, estes sistemas contam com três pares de tensão/frequência de operação, e podem ser configurados em diferentes níveis para classes de uma mesma tarefa.

No caso deste experimento, a restrição de tempo fornecida pelo TGFF foi de 1400 *u. t.*, mas este valor foi alterado para 1300 *u. t.*, de modo a garantir que os algoritmos fossem exercitados em qualquer condição de distribuição das classes de entrada. Este valor de $M = 1300$ *u. t.* foi assumido para TGFF99-5 e TGFF99-8, sendo que os sistemas com 1 e 2 processadores tiveram suas restrições de tempo definidas por $M = 5000$ *u. t.* e $M = 2500$ *u. t.*, respectivamente.

Os dados de entrada para a aplicação TGFF99 podem estar distribuídos em até 5 classes por tarefa. Repetem-se as duas qualidades mínimas de serviço, $Q_{MIN} = 0,50$ e $Q_{MIN} = 0,70$, para os quatro experimentos. A aplicação conta com quase 200 arestas e mais de 10.000 caminhos de execução, como descrito na Tabela 6.1. O Apêndice A1 novamente traz os arquivos para geração no TGFF, o DFG gerado, as tabelas e os dados usados para montagem dos dados da aplicação.

6.4 Resultados Experimentais

Como descrito nos itens anteriores, todas as aplicações contam com um grafo de tarefas e restrições de qualidade de serviço e tempo máximo de resposta, além de informações sobre suas tarefas, processadores e consumo de energia. Para melhor verificação dos resultados dos algoritmos, foi considerada a execução das aplicações sobre sistemas homogêneos com diferentes números de processadores que permitem a alteração dinâmica do par tensão/frequência de operação, além de diferentes restrições de qualidade de serviço.

6.4.1 Resultados SCN2 e SCN3

As Figuras 6.2 e 6.3 mostram, respectivamente, a energia média consumida por entrada e a utilização dos processadores obtidas para as configurações definidas por SCN2 e SCN3. As configurações definidas por SCN1 são a referência para estas comparações. As qualidades de serviço mínimas são $Q_{MIN} = 0,50$ e $Q_{MIN} = 0,70$ para todas as aplicações, para efeito de comparação e facilidade do controle dos resultados.

Existe uma redução de consumo com o ajuste individual do par tensão/frequência de cada processador de acordo com a classe de dados de entrada tratada, como pode ser observado pela diferença entre os resultados de consumo de energia das configurações definidas por SCN2 e SCN3 na Figura 6.2. Para SCN2 com $Q = 0,70$, o consumo de energia da maioria das aplicações é apenas a metade do resultado de SCN1. Para SCN3, há ainda uma maior redução do consumo. Para GMDFa-1, a configuração ditada por SCN2 requer apenas 60% da energia despendida na configuração ditada por SCN1, enquanto a configuração encontrada por SCN3 reduz em mais 23% o consumo de energia.

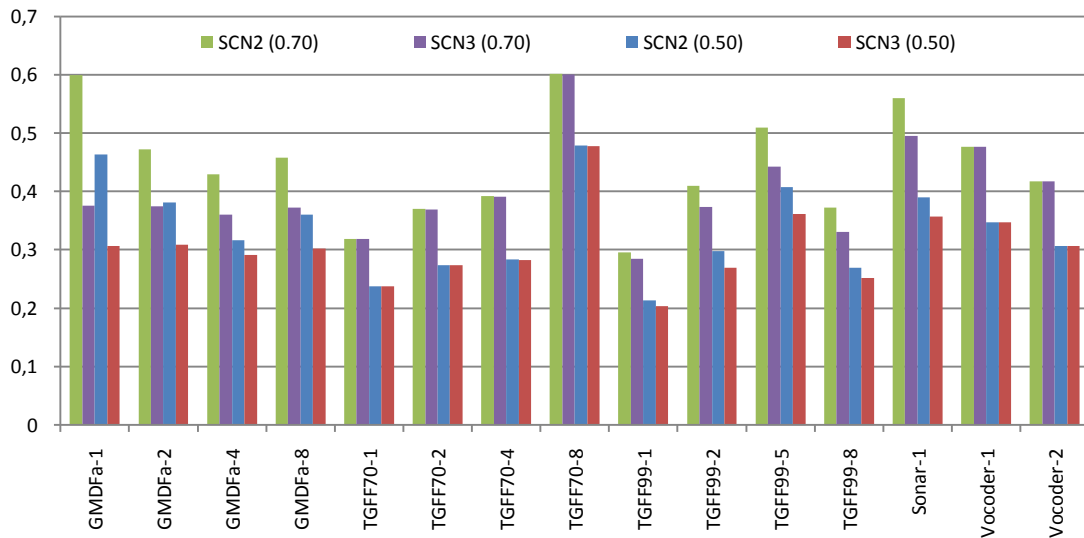


Figura 6.2: Consumo médio de energia por entrada em relação a *SCN1*

Para $Q = 0,50$, o consumo de energia relativo é menor que 40% para a maioria das aplicações. Isto indica que os algoritmos aproveitam a menor qualidade de serviço exigida para reduzir o consumo de energia.

Como comentado no Capítulo 4, uma taxa alta de utilização dos processadores indica um melhor aproveitamento do poder de processamento do sistema, diminuindo a ociosidade dos processadores. Para as configurações de *SCN2* e *SCN3*, a utilização média dos processadores aumenta em média de 25 para 50% quando comparadas à configuração definida por *SCN1*. Isto indica que o algoritmo utiliza menores níveis de tensão/frequência para aumentar o tempo de execução das tarefas ocupando o “tempo livre” criado pelo descarte de tarefas. Os casos de execução da aplicação Vocoder em sistemas com 1 e 2 processadores, Vocoder-1 e Vocoder-2, tem o maior aumento da taxa de utilização dos processadores, chegando próximo a 200% e 250%, respectivamente, quando comparados às configurações definidas por *SCN1*, como observado na Figura 6.3.

Para todas as aplicações, é possível observar que a taxa de utilização dos processadores aumenta pouco quando a qualidade de serviço exigida é reduzida de 0,70 para 0,50. A configuração do nível de tensão/frequência dos processadores, para cada classe em cada tarefa da aplicação, tem um maior impacto no aumento da taxa de utilização dos processadores do que a diminuição da restrição de qualidade.

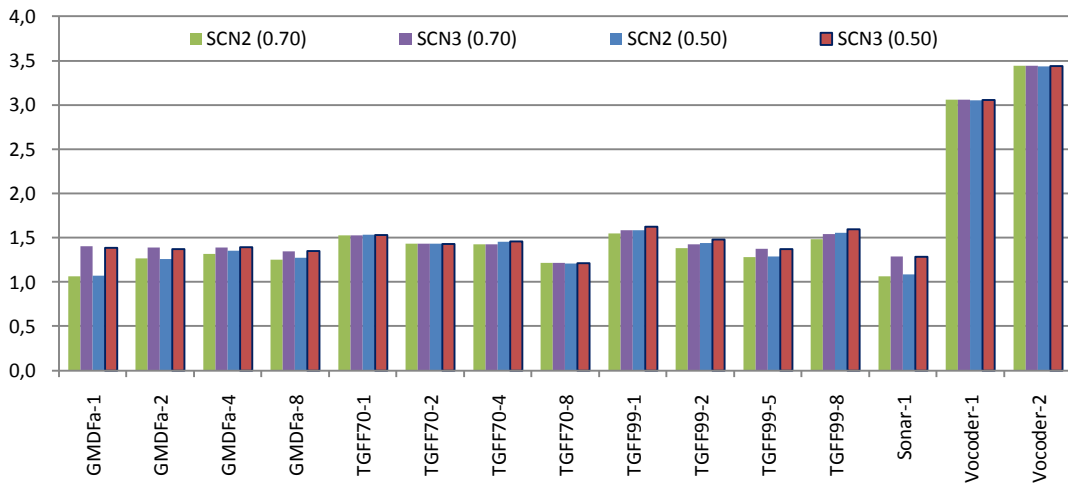


Figura 6.3: Utilização Média do Processador, relativo a SCN1

6.4.2 Resultados SCN4

Quando ocorrem alterações na distribuição de classes de entrada de uma aplicação durante a operação do sistema, caracterizando o cenário C4, é necessário que o sistema se adapte à nova condição das entradas, de modo a cumprir as restrições especificadas e, se possível, manter o baixo consumo, definido pela configuração obtida por SCN3. Devido à grande quantidade de resultados obtidos, a aplicação Vocoder-2, $Q_{MIN} = 0,50$, será utilizada como exemplo do experimento para levantamento dos dados que compõe os resultados de SCN4 para o cenário C4.

Esta aplicação foi escolhida por conter o maior número de tarefas e arestas de ligações, além da grande quantidade de caminhos de execução. No entanto, outras variações desta e de outras aplicações foram usadas no levantamento de resultados para este trabalho. As tabelas completas com os resultados para esta e outras aplicações estão descritas no Apêndice A3 e apenas gráficos de resultados de alguns experimentos para as outras aplicações serão apresentados aqui.

O experimento com o cenário C4 é realizado através da introdução de uma variação máxima (*delta*) nas probabilidades de classes de entrada de algumas tarefas que fazem descartes de classes de entrada. O “sinal do *delta*” indica se a mudança na distribuição das classes de entrada levou a um aumento de Q_{eff} (positivo) ou a uma degradação da qualidade de serviço (negativo). A variação foi aplicada em tarefas que fazem descarte de classes de entrada devido ao fato de que alterações de distribuição em tarefas que não fazem descarte não alteram Q_{eff} , embora alterem o consumo de energia para execução da tarefa.

As Figuras 6.4 e 6.5 representam, respectivamente, o consumo de energia e a utilização do processador para a aplicação Vocoder-2 quando é usado o algoritmo SCN4, que reconfigura o sistema de modo a tratar variações na distribuição de probabilidade das classes de entrada. Os valores de consumo de energia e utilização dos processadores são relativos à configuração

definida por *SCN3* para a distribuição probabilística inicial das classes de entrada e uma qualidade mínima de serviço de 0,50. Dentre as 133 tarefas do sistema, são seis as tarefas que originalmente fazem descarte de classes de entrada, configuração esta definida por *SCN3*.

Cada curva nas Figuras 6.4 e 6.5, referentes à aplicação Vocoder-2, $Q_{MIN} = 0,50$, corresponde a um número distinto de tarefas que tiveram a distribuição de classes de entrada alterada, chegando ao máximo de seis tarefas. Foram realizados experimentos alterando a distribuição das classes de entrada de apenas uma tarefa, até alteração da distribuição das seis tarefas ao mesmo tempo, utilizando vários valores de variação do *delta*.

Cada ponto nas curvas é a média de consumo de energia e utilização dos processadores num conjunto de combinações de tarefas com descarte, nas quais ocorreu a variação das classes de entrada. Como exemplo, os pontos da linha correspondente a 4 tarefas, são a média dos resultados de consumo de energia e utilização dos processadores para cada *delta* de sete das quinze combinações 4 a 4 das 6 tarefas com descarte para a aplicação Vocoder-2, $Q_{MIN} = 0,50$. As tabelas com as combinações usadas e os gráficos dos resultados para cada combinação estão no Apêndice A3.

O limite de sete combinações por valor de *delta* para experimentação do algoritmo, quando uma quantidade maior de combinações é possível, foi estabelecido após a comparação dos resultados usando esta quantidade com os resultados exaustivos, obtidos utilizando todas as combinações possíveis, onde não foram observadas diferenças significantes. Cada combinação de descartes requer uma execução do algoritmo. Cada ponto do gráfico, como o descrito no parágrafo anterior, é a média dos resultados obtidos para até sete combinações de descarte. Cada linha do gráfico apresentado nas Figuras 6.4 e 6.5 é o resultado das médias das experimentações do algoritmo *SCN4* para diversos valores de *delta*. Ou seja, para montar os gráficos de resultados apresentados, o algoritmo foi executado centenas de vezes. As tabelas com estes resultados são apresentadas no Apêndice A3.

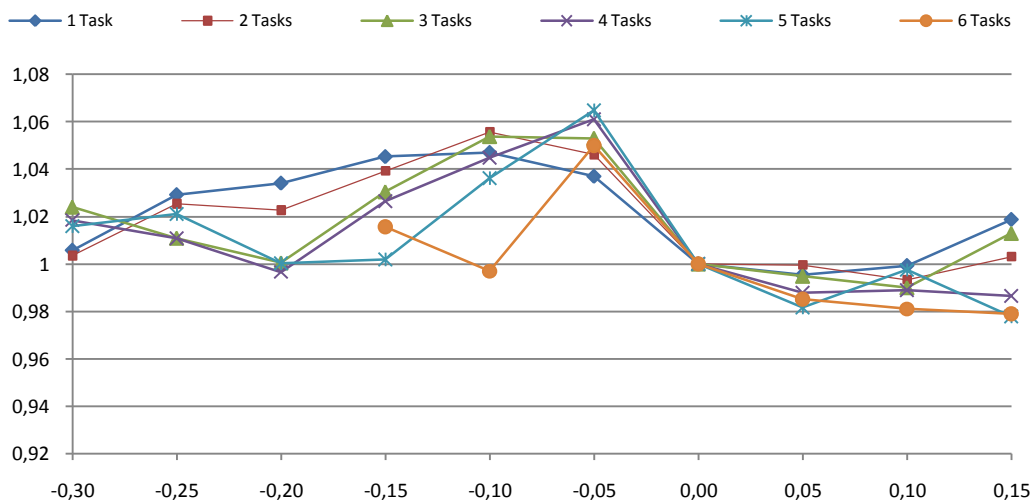


Figura 6.4: Consumo médio de energia por entrada para *SCN4* em relação a *SCN3*, como função do *delta*, para o Vocoder-2, $Q = 0,5$.

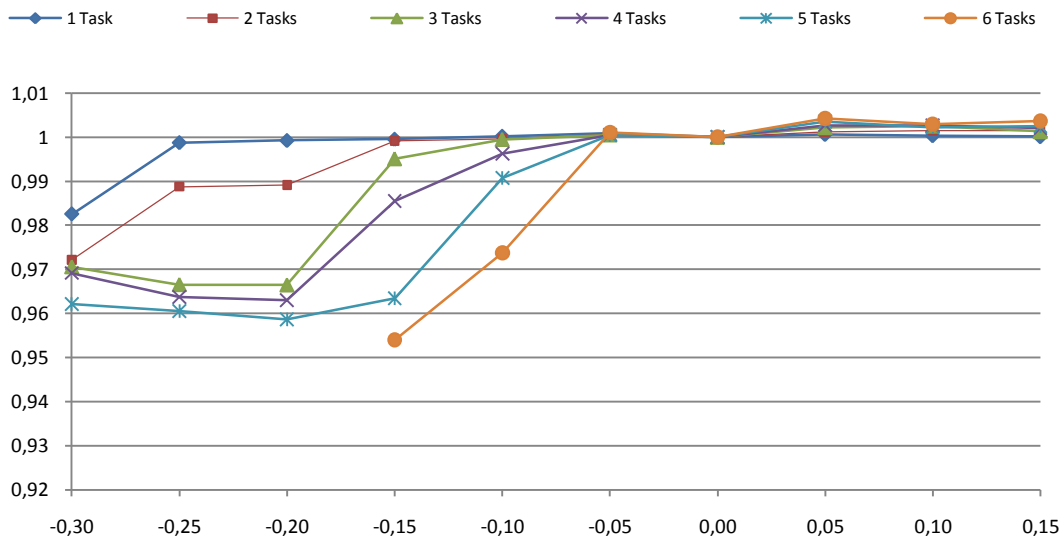


Figura 6.5: Utilização dos processadores, em relação a *SCN3*, em função do *delta*, para Vocoder-2, $Q = 0,5$.

A variação do *delta* é a mesma para todas as tarefas que fazem descarte de classes de entrada, de modo a facilitar o controle dos testes e a apresentação dos resultados no formato de gráficos. No entanto, qualquer variação de *delta* na distribuição das classes de entrada é tratada pelos algoritmos propostos. A variação do *delta* em passos de 0,05 se deve unicamente a facilitar a apresentação dos resultados. Como esperado, para valores de *delta* igual a zero, o algoritmo identifica que não houve mudança da qualidade de serviço da aplicação e não executa qualquer ação.

A faixa de variação do *delta*, partindo de -0,30 e chegando a +0,15 para tarefas que sofrem descarte, foi levantada levando-se em conta a faixa dos resultados de descarte observados na configuração obtida por *SCN3*. Esta variação indica que distribuição probabilística das classes da tarefa u_i mudou, alterando a qualidade efetiva (Q_{eff}) do sistema.

Em algumas aplicações, tarefas que fazem descarte de classes de entrada tem um somatório de probabilidades (definido por l_i) igual ou maior que 85%. Nestes casos, não é possível somar um *delta* de 0,15, pois as tarefas deixariam de descartar. No código de automação do algoritmo é feito um teste antes do incremento de 0,05 no valor do *delta*, onde é verificado se o resultado da soma vai ultrapassar o valor de 0,99 (99%). Caso isto ocorra, as outras tarefas continuam a sofrer incrementos no valor do *delta*, enquanto a tarefa em questão mantém o último valor (de 0,95 a 0,99).

Não foi imposto um limite inferior para aplicação do *delta*, chegando a -0,30 para todas as tarefas. Como pode ser observado nas Figuras 6.4 e 6.5, a curva que define a variação simultânea de *delta* na distribuição para 6 tarefas de Vocoder-2, se inicia apenas com o *delta* em -0,15 pois, para valores inferiores, não foi encontrada uma solução. Nos experimentos realizados, a redução drástica imposta à qualidade de serviço por valores maiores de *delta* impossibilitou *SCN4* de encontrar uma nova configuração para o sistema que satisfizesse a qualidade mínima.

É importante observar que, com o aumento de δ , é esperado que o consumo de energia diminua, pois existe folga em relação à qualidade mínima para descartes de classes de entrada e redução do par tensão/frequência de operação. No entanto, como observado na Figura 6.4, o consumo de energia aumenta para $\delta +0,15$, quando ocorrem descartes em uma, duas ou três tarefas simultaneamente. Isto ocorre porque não há folga para reduções de nível de operação, visto que são poucas as tarefas com descarte sobre as quais o algoritmo pode trabalhar. Para um número maior de tarefas com descarte, a redução do consumo de energia ocorre como esperado. A mesma condição pode ser observada quando o δ é $-0,05$ na Figura 6.6, onde ocorre um aumento da qualidade de serviço final. Uma tarefa que fazia descarte de classe de entrada deixou de descartar com a nova distribuição, mas não foi possível encontrar outra tarefa em que o descarte de classes de entrada mantivesse o sistema dentro de suas restrições de funcionamento, uma vez que a configuração definida por *SCN3* havia encontrado a melhor classe para descarte.

Considerando ainda a aplicação Vocoder-2, com $Q_{MIN} = 0,50$, para δ variando de $-0,30$ a $+0,15$, o consumo de energia nunca excede o obtido pela configuração de *SCN3* em mais de 6%. Se uma nova variação na distribuição probabilística das classes de entrada ocorrer, o algoritmo não vai aumentar o consumo do sistema em mais de 6% novamente. Supondo uma terceira variação, o novo consumo do sistema não vai exceder em mais de 18% o consumo inicial obtido com a configuração definida *offline* por *SCN3*.

A Figura 6.6 mostra que a qualidade de serviço Q_{eff} é mantida sempre acima de 0,50, de acordo com a restrição de funcionamento da aplicação.

A mesma aplicação Vocoder-2 foi considerada com $Q_{MIN} = 0,70$, e as Figuras 6.7, 6.8 e 6.9 mostram os resultados obtidos para consumo de energia, utilização dos processadores e qualidade efetiva em relação às variações de δ em até 4 tarefas com descarte. Para esta aplicação, a variação positiva do δ foi interrompida em $+0,10$, pois nenhuma das tarefas que fazem descarte pode ter uma variação de δ acima deste valor.

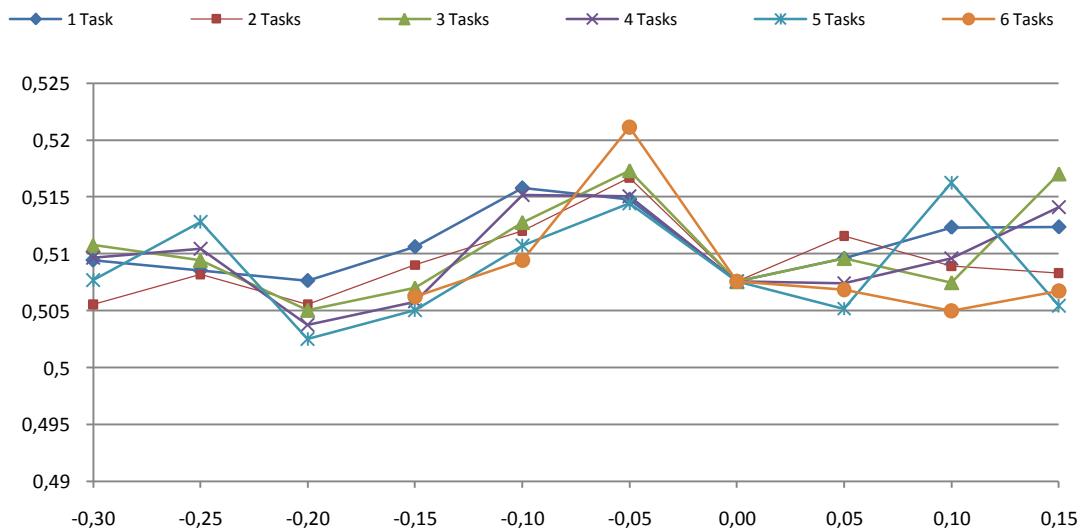


Figura 6.6: Qualidade de serviço efetiva para *SCN4*, como função do δ , para Vocoder-2, $Q = 0,5$.

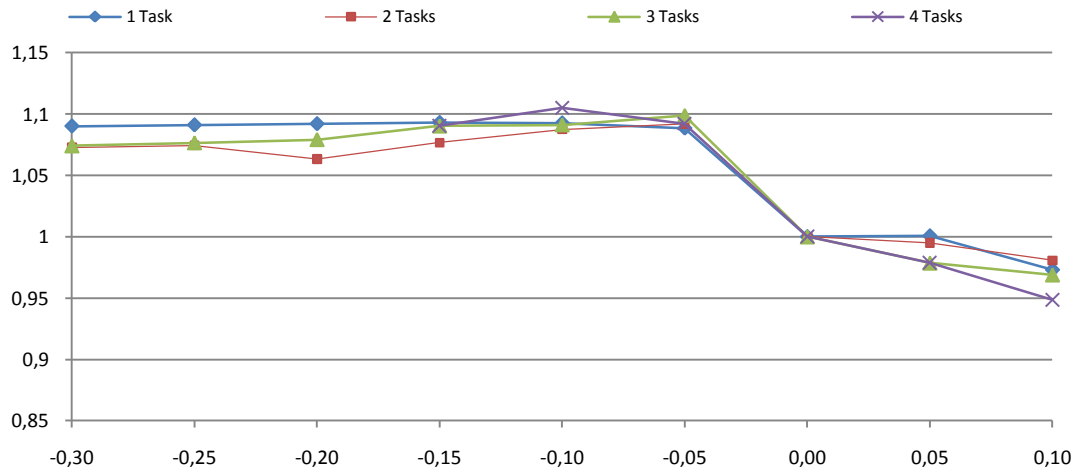


Figura 6.7: Consumo Médio de Energia por entrada para *SCN4* em relação a *SCN3*, como função do *delta*, para o Vocoder-2, $Q = 0,7$.

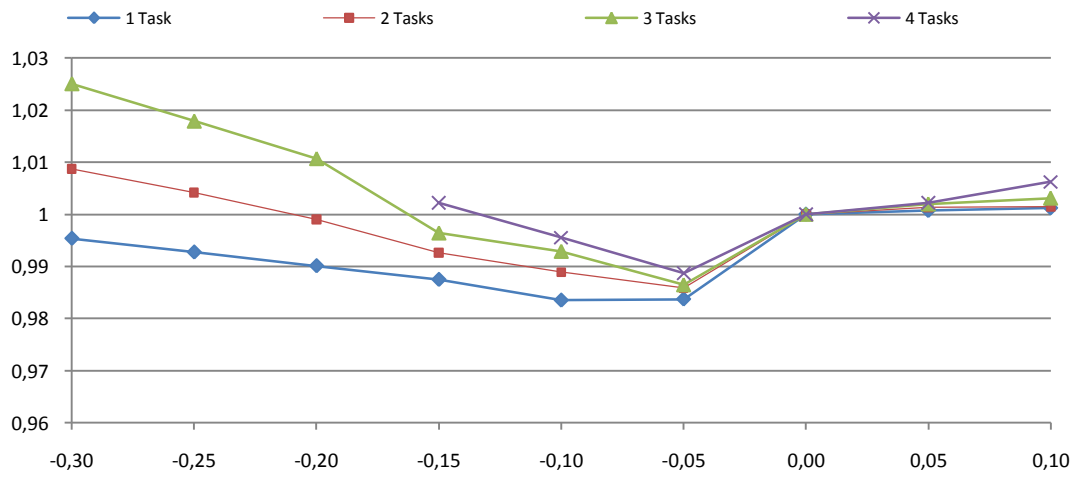


Figura 6.8: Utilização dos Processadores, em relação a *SCN3*, em função do *delta*, para Vocoder-2, $Q = 0,7$.

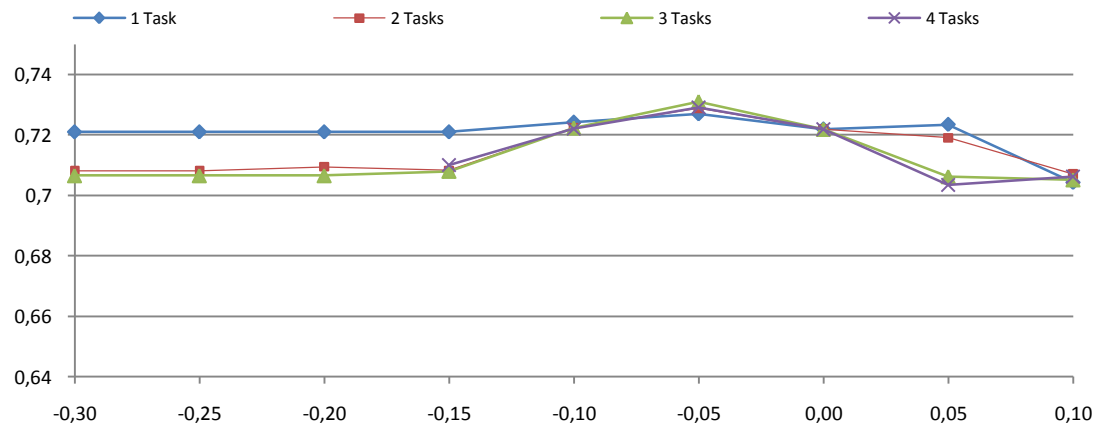


Figura 6.9: Qualidade de Serviço efetiva para *SCN4*, como função do *delta*, para Vocoder-2, $Q = 0,7$.

A Figura 6.10 mostra os resultados para Vocoder-1, também com $Q_{MIN} = 0,70$, quando ocorrem variações na distribuição das classes de entrada de até 4 tarefas com descarte. Pelo mesmo motivo que para o Vocoder-2 com $Q_{MIN} = 0,70$, não aparecem nos gráficos valores de δ acima de +0,10. Novamente são observados alguns valores negativos de δ para os quais não foi possível encontrar uma configuração de funcionamento em que as restrições do sistema fossem satisfeitas. No entanto, para os casos em que uma solução é possível, pode-se observar que foi mantido um consumo contido de energia (aumentos inferiores a 2%), com melhora da utilização dos processadores, respeitando as restrições de funcionamento do sistema.

Foram ainda realizados experimentos de variação da distribuição das classes de entrada para outras aplicações. A Figura 6.11 mostra os resultados obtidos para GMDFa-4, $Q_{MIN} = 0,50$, onde até 6 tarefas podem descartar classes de entrada. Os resultados obtidos seguiram o comportamento observado para as outras aplicações abordadas para experimentação de *SCN4*, mantendo um consumo contido de energia, com pouca redução na utilização dos processadores, respeitando as restrições de funcionamento do sistema. Para todas as condições de experimentação do δ foram encontradas configurações de funcionamento em que as restrições de funcionamento são respeitadas.

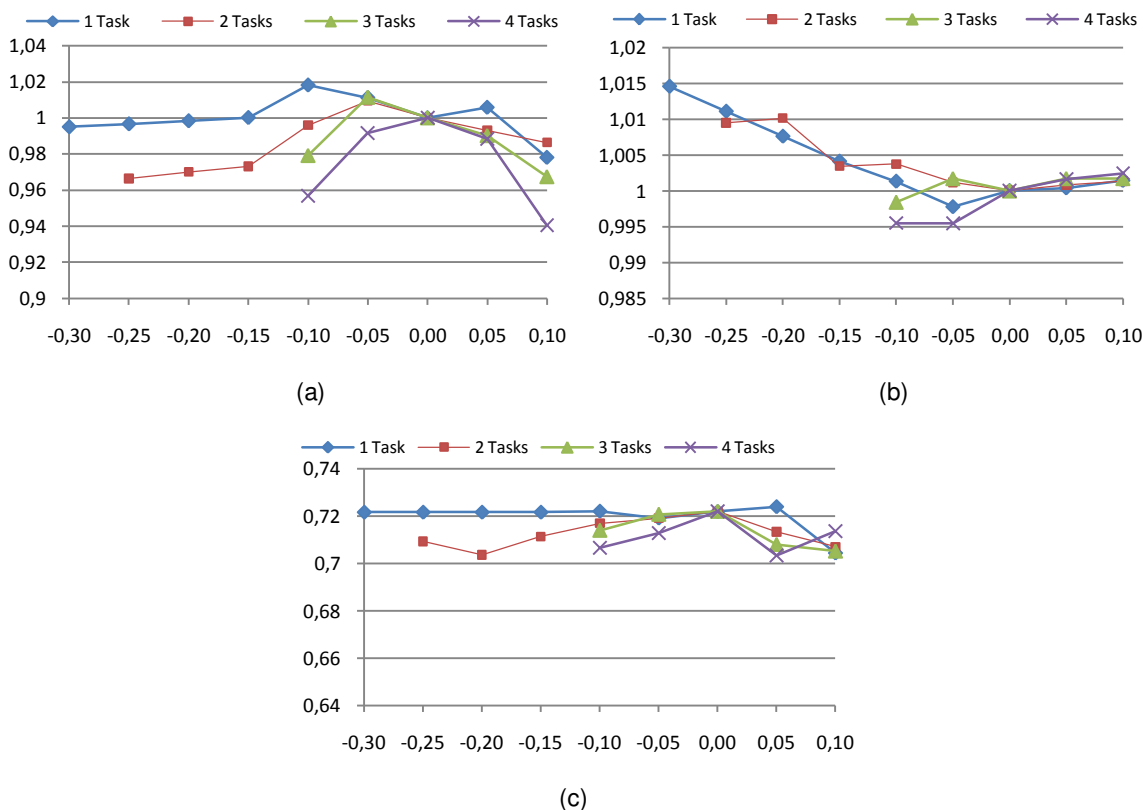


Figura 6.10: (a) Consumo de energia, (b) utilização dos processadores e (c) qualidade de serviço efetiva, como função do δ , para Vocoder-1, $Q = 0,7$.

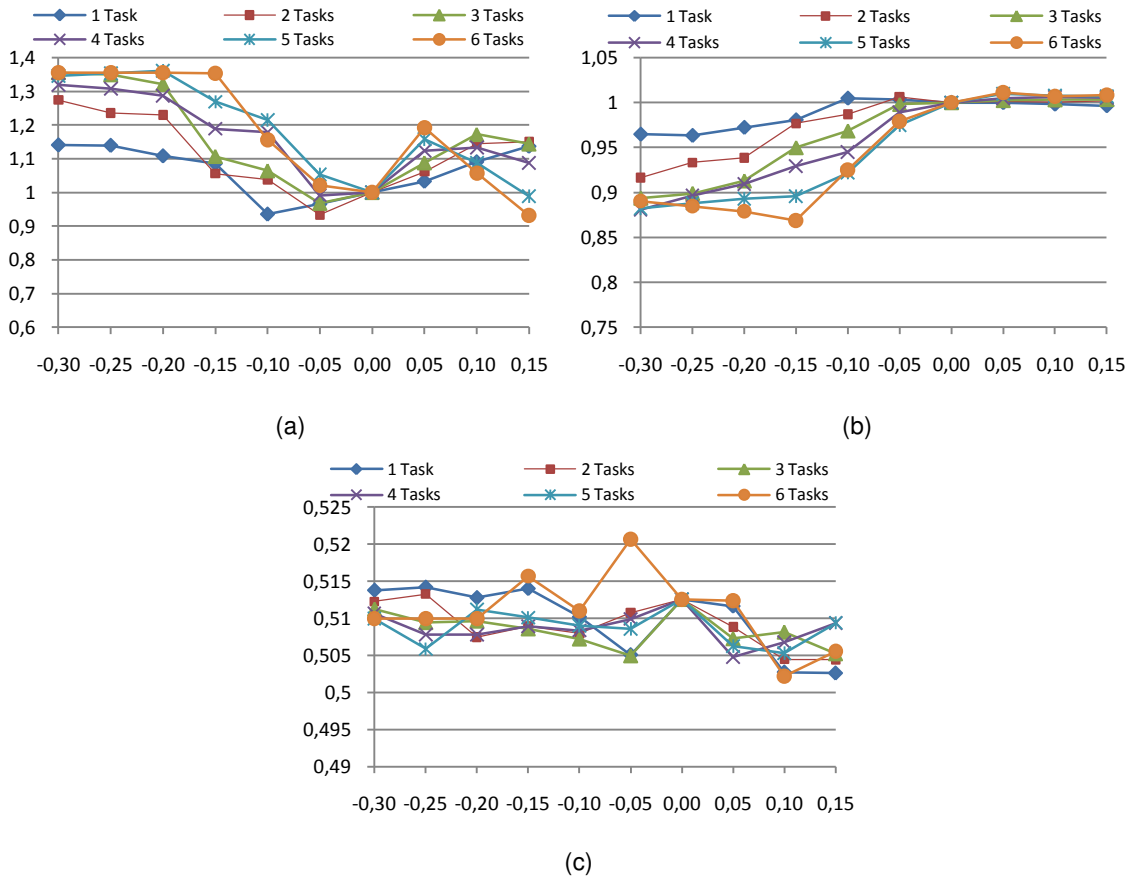


Figura 6.11: (a) Consumo de energia, (b) utilização dos processadores e (c) qualidade de serviço efetiva, como função do δ , para GMDFa-4, $Q = 0,5$.

A Figura 6.12 mostra os resultados obtidos para a aplicação sintética TGFF70-4, onde até 3 tarefas podem sofrer descarte de classes de entrada. Por contar com apenas 3 tarefas que fazem descarte de classes de entrada, o algoritmo tem poucas “oportunidades” de alteração de descartes, trabalhando mais no aumento ou diminuição dos níveis de operação. Este comportamento do algoritmo pode ser observado na Figura 6.12(a), onde maiores variações no consumo de energia são observadas. Contudo, os resultados obtidos mostram que, mesmo trabalhando com poucas tarefas passíveis de descarte, o algoritmo manteve o sistema funcionando dentro das restrições de tempo e qualidade, com consumo contido de energia e boa utilização dos processadores. Também por descartar classes em apenas três tarefas, os valores de δ diferem das variações apresentadas para as outras aplicações, pois foram trabalhados especificamente para esta aplicação para que mais experimentos pudessem ser realizados.

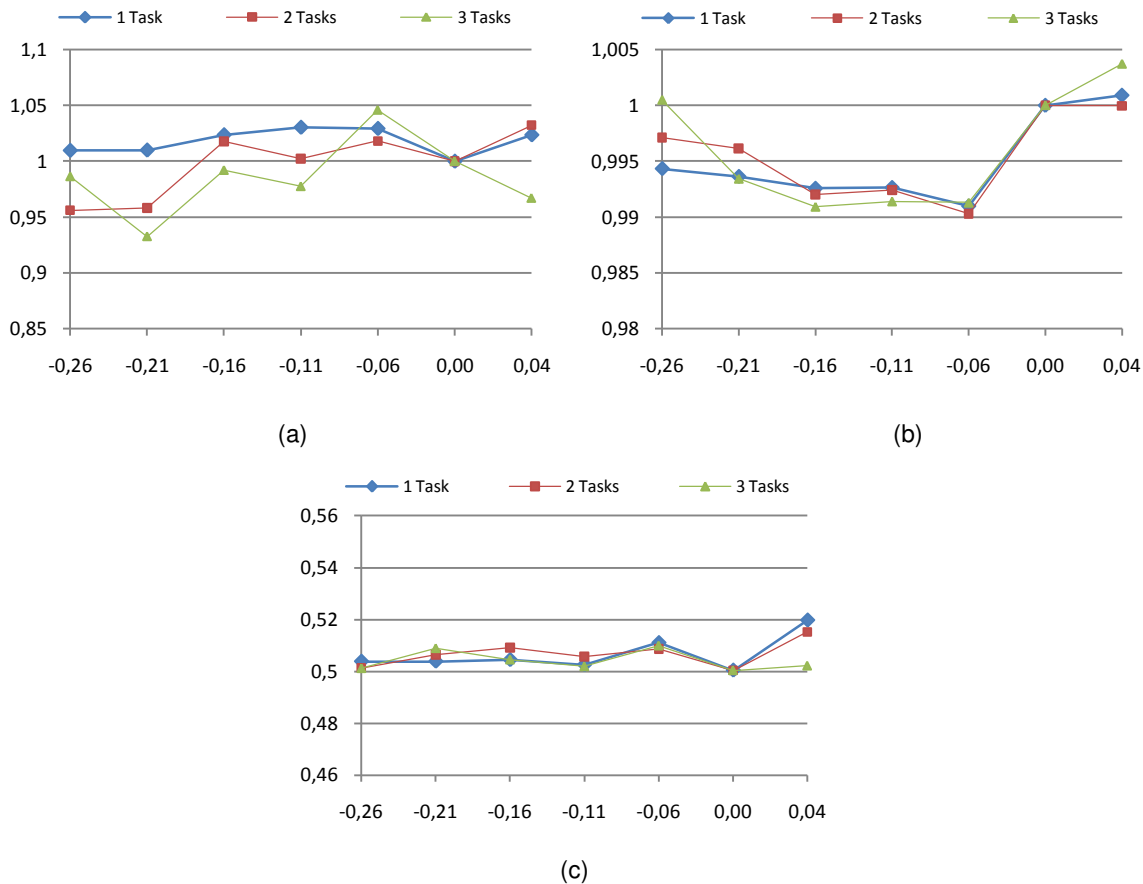


Figura 6.12: (a) Consumo de energia, (b) utilização dos processadores e (c) qualidade de serviço efetiva, como função do δ , para TGFF70-4, $Q = 0,5$.

6.5 Conclusão

Foram apresentados neste capítulo os resultados experimentais para os algoritmos apresentados neste trabalho, através de dezenas de variações de 5 aplicações de sistemas embarcados, sendo três aplicações reais [6][14][41] e duas sintéticas, geradas através de TGFF [12][34].

Foram observados que, em relação aos resultados levantados para o cenário C1, a redução do consumo de energia chegou a 60% para o cancelador de eco quando rodando a configuração definida por *SCN2* em um sistema com apenas um processador e $Q_{MIN} = 0,70$. Rodando a configuração definida por *SCN3* o consumo de energia ainda é reduzido em mais 23%. Considerando ainda o cancelador de eco com a mesma restrição de qualidade, mas agora sendo executado em um sistema com 8 processadores, o consumo de energia com a configuração definida por *SCN2* é 54% menor. Utilizando a configuração definida por *SCN3*, uma redução próxima a 67% é observada em relação ao consumo de energia do cenário C1.

Os resultados obtidos mostram que os algoritmos propostos também definem uma configuração que leva a um melhor uso dos processadores, melhorando a utilização dos processadores em até 250%, quando comparados os resultados de *SCN3* aos obtidos para o cenário C1, quando a aplicação Vocoder é executada em um sistema com dois processadores.

Também para Vocoder-2, com $Q_{MIN} = 0,50$, o uso do algoritmo *SCN4* gerou resultados que mostram que uma nova configuração de funcionamento que atenda à especificação de tempo de resposta e qualidade da aplicação pode ser obtida com um aumento máximo de 18% para até três variações das distribuições das classes de entrada do sistema sem, no entanto, reduzir a utilização do processador em mais de 5% para cada variação.

O próximo capítulo descreve as contribuições deste trabalho e algumas sugestões para futuros trabalhos, buscando a melhoria dos algoritmos apresentados.

7 Conclusão

Este capítulo descreve as contribuições deste trabalho assim como sugestões de desenvolvimento dos algoritmos propostos em futuros trabalhos.

7.1 Contribuições

São altos os investimentos na busca por redução de consumo de baterias em sistemas portáteis e de comunicação, levando pesquisadores de diferentes áreas a desenvolver algoritmos de redução de consumo de energia, baseados em diferentes estruturas de funcionamento de sistemas embarcados, inclusive alterando a construção destes sistemas. A necessidade de redução de consumo de energia, mantendo o desempenho especificado para o sistema foi o tema deste trabalho.

Os algoritmos propostos neste trabalho exploram a possibilidade dos sistemas embarcados de trabalhar em diferentes níveis de operação (DVFS), atribuindo um nível a cada classe de dados de entrada das tarefas da aplicação. Por sua simplicidade de implementação e convergência a uma configuração de menor consumo de energia, a utilização dos algoritmos propostos se estende a quase todos os sistemas embarcados cujas classes de entradas possam ser identificadas, mesmo que não sejam alimentados por bateria. Para avaliação dos algoritmos propostos, aplicações reais e sintéticas foram consideradas em sistemas com diferentes números de processadores, possibilitando uma análise dos resultados obtidos.

Comparando os resultados de consumo de energia e utilização dos processadores do sistema com as configurações de funcionamento obtidas pelos algoritmos *SCN2* e *SCN3* aos resultados para uma configuração com um nível de operação mínimo, comum a todos os processadores do sistema (cenário C1), observaram-se reduções no consumo médio de energia por entrada de até 71%, além de uma maior utilização dos processadores do sistema.

Também foi apresentado um algoritmo para reconfiguração do sistema para que a aplicação volte a responder às restrições de funcionamento quando uma variação na distribuição probabilística dos dados em classes de entrada for observada (*SCN4*). Quando uma aplicação tem uma mudança na distribuição dos dados de entrada, o algoritmo proposto rapidamente adequa o sistema à nova necessidade de funcionamento, aumentando o desempenho da aplicação e buscando minimizar o aumento do consumo de energia. Por outro lado, com uma melhora na qualidade dos dados de entrada processados, *SCN4* tem a oportunidade de buscar a redução do consumo de energia, fazendo novos descartes. Assim, para pequenas variações na distribuição de classes de entrada de uma aplicação, este algoritmo possibilita aos sistemas embarcados a adequação de seu desempenho.

7.2 Trabalhos Futuros

Existem ainda algumas possibilidades de melhoria no funcionamento dos algoritmos propostos.

Embora os algoritmos propostos cubram uma vasta gama de aplicações, seu uso em aplicações que dependem de realimentação de dados, como filtros, ainda não é possível. DFGs cíclicos, onde uma tarefa gere resultados que serão usados por tarefas ascendentes a esta, ocasionando uma dependência iterativa, constituem uma situação interessante de evolução dos algoritmos propostos.

O mapeamento das tarefas nos múltiplos processadores não foi um dos focos de desenvolvimento deste trabalho. O desenvolvimento de um algoritmo de mapeamento direcionado à redução do consumo de energia, que trabalhe junto com os algoritmos propostos neste trabalho pode melhorar os resultados obtidos.

Após o levantamento da configuração do sistema para menor consumo de energia, definida por *SCN3*, todos os caminhos de execução da aplicação respeitam a restrição de tempo M . No entanto, alguns caminhos podem estar com tempos de execução bem abaixo do valor de M , possibilitando a execução de uma nova etapa em busca da redução do consumo de energia, fazendo-se uso desta “folga” de tempo. Uma nova etapa para *SCN3* definiria uma “folga mínima” e faria a busca de tarefas que fazem parte apenas de caminhos com folgas maiores ou iguais a esta folga mínima. Em seguida, mediante critério de classificação a ser definido, seria escolhida uma tarefa que teria seu par tensão/frequência reduzido durante sua execução, reduzindo o consumo através do aumento do tempo de execução do caminho sem, no entanto, exceder M . É certo que este ajuste da configuração deve levar em conta os possíveis diferentes níveis de execução das classes de entrada dos dados desta tarefa.

Para sistemas em que o tempo requerido na troca de níveis de operação seja desprezível, é possível adaptar o algoritmo para que encontre a tensão ideal de operação para o sistema, sem considerar o conjunto de níveis de operação existente. Para isso, os níveis imediatamente inferior e imediatamente superior ao “nível ideal” obtido pelo algoritmo para a execução da tarefa, seriam usados para sua execução. A tarefa seria executada por um tempo no nível inferior e o restante do tempo no nível superior. Técnica semelhante foi utilizada por Hua *et al.* [15].

SCN4 usa, como base de cálculo e para busca de tarefas para alteração, os caminhos que “potencialmente excedem M ” da aplicação. Foram levantadas algumas possíveis alterações deste mecanismo no item 5.6.1 do capítulo 5, mas não foram realizados testes. Em trabalhos futuros, este mecanismo pode ser refinado e outras técnicas, diferentes das propostas, devem surgir à medida que os testes forem realizados. Ainda, quando o sistema sofre uma grande perda de qualidade, a ponto de não ser possível encontrar uma nova configuração que atenda as restrições de funcionamento, o algoritmo só informa esta impossibilidade quando exaurir todas as tentativas de reconfiguração. O algoritmo pode ser trabalhado para que esta impossibilidade de solução possa ser detectada previamente, e isso pode ser feito com o armazenamento de algumas

configurações estáticas para aproximação, detectando mais rapidamente a não existência de uma solução.

A utilização de algoritmos de predição das alterações na distribuição das classes de entrada, em conjunto com *SCN4*, possibilitaria definir uma configuração para o sistema antes que as alterações ocorram, acelerando a adaptação das aplicações às novas condições de funcionamento.

Uma interface gráfica foi criada para levantamento e controle dos resultados, mas a geração das tabelas de resultados, assim como os gráficos que representam visualmente estas tabelas, foram montadas através da importação de arquivos texto de resultados. Em trabalhos futuros, existe a intenção de automatizar a geração destes gráficos através da interface gráfica desenvolvida.

De modo geral, várias rotinas do código podem apresentar melhor desempenho, acelerando a execução dos algoritmos, através da aplicação de técnicas de escrita de *software*, pois este trabalho não teve como foco a otimização de *software* para desempenho.

8 Referências

- [1] AMD, *Mobile AMD Athlon 4 - Processor Model 6 CPGA Data Sheet*, AMD Advanced Micro Devices Inc., 2001.
- [2] AMD Phenom, <http://www.amd.com/br/products/desktop/processors/phenom/Pages/AMD-phenom-processor-X4-X3-at-home.aspx>, em 15/08/12
- [3] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, B. M. Al Hashimi, "Quasi-Static Voltage Scaling for Energy Minimization with Time Constraints", *Proceedings of the Conference on Design, Automation and Test in Europe, Vol. 1*, 514-519, 2005.
- [4] D. Angolini, A. M. Tokarnia, "Síntese das arquiteturas de comunicação e de memória em MPSoCs" *Anais do Primeiro Encontro de Alunos e Docentes do Depto. de Engenharia de Computação e Automação Industrial da FEEC Unicamp – EADCA*, 12 e 13/03/2009.
- [5] K. Bhatti, C. Belleudy, M. Auguin, "Power Management in Real Time Embedded Systems through Online and Adaptive Interplay of DPM and DVFS Policies," *Proceedings of 2010 IEEE/IFIP 8th Int'l Conf. on Embedded and Ubiquitous Computing (EUC)*, 2010.
- [6] L. Bianco, M. Auguin, G. Gogniat, A. Pegatoquet, "A Path Analysis based Partitioning for Time Constrained Embedded Systems", *Proceedings of 6th International Workshop on Hardware/Software Codesign*, 85-89, 1998.
- [7] S. Bilavarn, C. Belleudy, M. Auguin, T. Dupont, A.-M. Fouillart, "Embedded Multicore Implementation of a H.264 Decoder with Power Management Considerations," *Proceedings of 11th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, 124-130, 2008.
- [8] Z. Cao, B. Foo, L. He, M. van der Schaar, "Optimality and Improvement of Dynamic Voltage Scaling Algorithms for Multimedia Applications", *Proceedings of the 45th Annual Design Automation Conference*, 179-184, 2008.
- [9] W. J. Dally, J. Poulton, *Digital Systems Engineering*, Cambridge University Press, Cambridge, UK, 1998.
- [10] Dev-C++, BloodShed Software, <http://www.bloodshed.net/devcpp.html>, em 08/07/12.
- [11] G. Dhiman, T. S. Rosing, "Dynamic Voltage Frequency Scaling for Multi-tasking Systems Using Online Learning", *Proceedings of the 2007 International Symposium on Low Power Electronics and Design*, 207-212, 2007.
- [12] R.P. Dick, D.L. Rhodes, W. Wolf, "TGFF: Task Graphs for Free", *Proceedings of the Sixth International Workshop on Hardware/Software Codesign*, 97-101, 1998.
- [13] M. Feischmann, *LongRun Power Management: Dynamic Power Management for Crusoe Processors*, Transmeta Corporation, 2001.
- [14] A. Gerstlauer, S. Zhao, D. D. Gajski, "Design of a GSM Vocoder using SpecC Methodology", *Technical Report ICS-99-11*, February 26, 1999.
- [15] S. Hua, G. Qu, S. S. Bhattacharyya, "Energy Reduction Techniques for Multimedia Applications with Tolerance to Deadline Misses", *Proceedings of Design Automation Conf.*, 131-136, 2003.

- [16] Y.-S. Hwang, S.-K. Ku, C.-M. Jung, K.-S. Chung, "Predictive Power Aware Management for Embedded Mobile Devices," *Proceedings of 2008 Asia and South Pacific Design Automation Conference*, 2008.
- [17] Intel, <http://www.intel.com/support/pt/processors/mobile/pm/sb/cs-007983.htm>, em 26/07/12.
- [18] Intel, *Intel XScale Core Developer's Manual*, Intel Corporation, 2004.
- [19] Intel Core, <http://www.intel.com/content/www/us/en/processors/core/core-processor-family.html>, em 15/08/12
- [20] Intel Xeon, <http://www.intel.com/content/www/us/en/processors/xeon/xeon-e7-transforming-mission-critical-computing-brief.html>, em 15/08/12
- [21] R. Jejurikar, R. Gupta, "Dynamic Voltage Scaling for Systemwide Energy Minimization in Real Time Embedded Systems", *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, 78-81, 2004.
- [22] C. Kim , K. Roy, "Dynamic V_{TH} Scaling Scheme for Active Leakage Power Reduction", *Proceedings of the Conference on Design, Automation and Test in Europe*, p.163, 2002.
- [23] C.-H. Lee, K. G. Shin, "On-line dynamic voltage scaling for hard real-time systems using the EDF algorithm," *Proceedings of 25th IEEE Int'l Real-Time System Symposium*, 2004.
- [24] S. Lee, B. K. Kim, "Coscheduling of Processor Voltage and Control Task Period for Energy-efficient Control Systems", *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 9, No. 3, Article 15, 2010.
- [25] S. Liu, Q. Wu, Q. Qiu, "An Adaptive Scheduling and Voltage/Frequency Selection Algorithm for Real-time Energy Harvesting Systems", *Proceedings of the 46th Annual Design Automation Conference*, 782-787, 2009.
- [26] S. Martin, K. Flautner, T. Mudge, D. Blaauw, "Combined Dynamic Voltage Scaling and Adaptative Body Biasing for Lower Power Microprocessors under Dynamic Workloads", *International Conference on Computer-Aided Design (ICCAD) Proc.*, 721-725, 2002.
- [27] Y. Mei, Y. -H. Lu, Y. C. Hu, C. S. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques", *IEEE Proceedings of the 12th International Conference on Advanced Robotics*, 492-497, 2005.
- [28] NI Lab Windows, National Instruments, <http://www.ni.com/lwcvj>, em 08/07/12
- [29] K. Nose, M. Hirabayashi, H. Kawaguchi, S. Lee, T. Sakurai, " V_{TH} -hopping scheme for 82% power saving in low-voltage processors", *Proc. of Custom Integrated Circuits Conf.*, 2001.
- [30] D. A. Patterson, J. L. Hennessy, *Computer Organization and Design: The hardware / Software Interface, 3rd Edition*, Morgan Kaufmann, 2005.
- [31] P. Pillai, K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems", *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, 89-102, 2001.
- [32] M. Qiu, C. Xue, Z. Shao, E. H.-M. Sha, "Energy Minimization with Soft Real-time and DVS for Uniprocessor and Multiprocessor Embedded Systems", *Proceedings of European Design and Automation Association*, 1641-1646, 2007.

- [33] D. Rakhmatov, S. Vruthula, "Energy Management for Battery-Powered Embedded Systems", *ACM Transactions on Embedded Computing Systems (TECS)*, 277-324, 2003.
- [34] D. Rhodes, R. Dick, "Task Graphs For Free", <http://ziyang.eecs.umich.edu/~dickrp/tgff>, em 01/08/11.
- [35] T. Sassolas, N. Ventroux, N. Boudoani, G. Blanc, "A Power-Aware Online Scheduling Algorithm for Streaming Applications in Embedded MPSoC," *Proceedings of 20th International Conf. on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2010.
- [36] P. Schaumont, B. C. Lai, W. Qin, I. Verbauwhede, "Cooperative multithreading on Embedded Multiprocessor Architectures Enables Energy-scalable Design", *Proceedings of Design Automation Conf.*, 27-30. 2005.
- [37] M. T. Schmitz, B. M. Al-Hashimi, P. Eles "Energy-Efficient Mapping and Scheduling for DVS Enabled Distributed Embedded Systems", *Proceedings of the Conference on Design, Automation and Test in Europe*, 514-521, 2002.
- [38] T. Simunic, L. Benini, G. Micheli, M. Hans, "Source Code Optimization and Profiling of Energy Consumption in Embedded Systems", *Proceedings of International Symp. on System Synthesis*, 193-198, 2000.
- [39] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, F. Baez, "Reducing Power in High-Performance Microprocessors", *Proceedings of 35th Association for Computer Machinery / Institute of Electrical and Electronics Engineers (ACM/IEEE) Design Automation Conference*, 732-737, 1998.
- [40] H. M. Wang, H. S. Choi, J. T. Kin, "Workload-Based Dynamic Voltage Scaling with the QoS for Streaming Video", *Proceedings of 4th IEEE International Symposium on Electronic Design, Test & Applications*, 236-239, 2008.
- [41] C. M. Woodside, G. G. Monforton, "Fast Allocation of Processes in Distributed and Parallel Systems", *IEEE Transactions on Parallel and Distributed Systems*, 164-174, 1993.
- [42] L. Yan, L. Zhong, N. Jha, "User-perceived Latency Driven Scaling for Interactive Applications", *Proceedings of Design Automation Conf.*, 624-627, 2005.
- [43] C. Y. Yang, J.J. Chen, T. W. Kuo, "Energy-Efficiency for Multiframe Real-Time Tasks on a Dynamic Voltage Scaling Processor", *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, 211-220, 2009.
- [44] A. Yang, M. Song, "Aggressive Dynamic Voltage Scaling for Energy-Aware Video Playback Based on Decoding Time Estimation", *Proceedings of the seventh ACM International Conference on Embedded Software*, 1-10, 2009.
- [45] L. Zhong, and N.K. Jha, "Dynamic Power Optimization Targeting User Delays in Interactive Systems", *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, Nov. 2000.
- [46] J. Zhu, R. Damer, D. Gajski, "Syntax and Semantics of the SpecC Language," *Proceedings of the Synthesis and System Integration of Mixed Technologies*, 1997.
- [47] J. Zhuo, C. Chakrabarti, "System-level Energy-Efficient Dynamic Task Scheduling", *Proceedings of Design Automation Conference*, 628-631, 2005.

Apêndice A1: Aplicações

Neste apêndice serão apresentados detalhes sobre as aplicações utilizadas neste trabalho para a obtenção dos resultados do capítulo 6. Estão aqui disponíveis os grafos de tarefas e as tabelas de parâmetros da aplicação sobre as quais foram desenvolvidos os exemplos. Este apêndice inclui também as tabelas completas de dados que foram montadas para simulação dos algoritmos, assim como as condições de “aleatoriedade” sobre as quais foram montados os dados não disponíveis nas tabelas originais.

A1.1 Cancelador de Eco Acústico

O sistema apresentado por Bianco *et. al.* [6] é um sistema de Cancelamento de Eco Acústico, e sua estrutura de tarefas pode se descrita pelo DFG (*Data Flow Graph*) da Figura A1.1.

Identificado como GMDFa, este sistema conta com 30 tarefas para tratamento dos dados recebidos em “X” e, de modo a caracterizar uma tarefa de aglutinação dos resultados em “Y” ao final do tratamento na Figura A1.1, foi incluída mais uma tarefa, passando o sistema a contar com 31 tarefas. Como citado anteriormente, o tempo de tratamento dos dados de entrada por estas 31 tarefas deve ficar abaixo de $M = 8\text{ ms}$, frequência de amostragem especificada para o sistema.

Bianco *et. al.* fornecem também o tempo de execução em software para cada uma das tarefas, e o tempo de execução em dois aceleradores de hardware para algumas das tarefas. Estas informações de tempo, em μs , são apresentados na Tabela A1.1. Além das informações de tempo da Tabela A1.1, os tempos de comunicação entre as diversas tarefas também são fornecidos na Figura A1.1, e são considerados apenas quando há troca de dados entre hardware e software, sendo usados como referência neste trabalho para definição dos tempos de

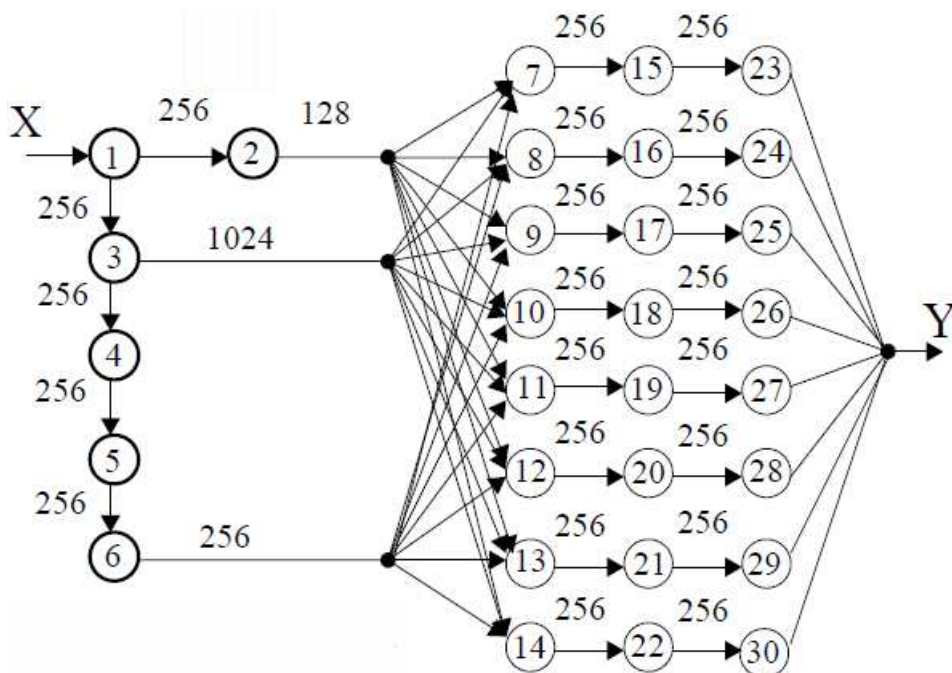


Figura A1.1: DFG Cancelador de Eco Acústico com tempos de comunicação entre tarefas [6].

Tabela A1.1: Tempos de execução em SW e HW [6]

Tarefa	Tipo	SW	HW1	HW2
1	fft	406/ 0,15		87/ 1,8
2	div	472/ 0,13		
3	conv/ add2d	730/ 0,21	123/ 0,86	
4	ifft	470/ 0,16		87/ 1,8
5	mult/ sub	62/0,08	16/ 0,86	
6	fft	406/ 0,15		87/ 1,8
7 a 14	conv/ mult	84/0,11	23/ 0,86	
15 a 22	ifft	470/ 0,16		87/ 1,8
23 a 30	fft/add1d	451/ 0,16		98/ 1,87

comunicação entre tarefas, quando executadas em diferentes processadores. Estes tempos foram divididos por 10 e a unidade de tempo usado foi μs . Se o sistema conta com apenas um processador, ou a troca de dados é feita entre tarefas mapeadas em um mesmo processador, estes tempos são desconsiderados.

No entanto, os dados fornecidos não são suficientes para utilização da aplicação de cancelamento de eco neste trabalho, e outros dados de funcionamento do sistema foram criteriosamente gerados a partir dos dados disponíveis em [6].

Os algoritmos apresentados neste trabalho tem como proposta a sua utilização em sistemas que contam com um ou mais processadores, os quais podem operar em diferentes níveis de tensão/frequência e que admitem variações na qualidade do sinal processado. Para isso, partindo da tabela de tempos de execução em software, definidos na terceira coluna da Tabela A1.1, foi definida uma nova tabela, com tempos de execução em três diferentes níveis de tensão/frequência, $V_1|F_1$, $V_2|F_2$ e $V_3|F_3$, além da variação de uma a 4 classes de dados de entrada para cada tarefa. Os dados gerados para cada tarefa estão descritos na Tabela A1.2.

Para a montagem desta tabela, foram utilizados alguns critérios. Às tarefas do sistema foram atribuídas de uma a quatro classes de dados de entrada, sendo que a igualdade entre as tarefas foi definida através da identificação dada às etapas na segunda coluna da Tabela A1.1. Considerando como exemplo a tarefa 1, lhe foram atribuídas 3 classes de dados entrada.

O tempo para execução de uma tarefa, definido na Tabela A1.1, foi usado como o maior tempo de tratamento de dados de entrada no nível intermediário de tensão/frequência. Utilizando novamente a tarefa 1, a Tabela A1.1 define seu tempo de execução em $406 \mu s$. Então, este foi o maior tempo de tratamento de dados de entrada na tarefa 1, sendo esta executada no nível $V_2|F_2$ de operação.

Tabela A1.2: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia

Tarefa	Prob	V3 F3		V2 F2		V1 F1		
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.	
1	0,62	134	21	178	12	267		5
	0,94	199		265		397,5		
	1	305		406		609		
2	0,73	287	25	383	14	574,5		6
	1	354		472		708		
3	0,53	228	32	304	18	456		8
	0,8	334		445		667,5		
	1	548		730		1095		
4	0,51	194	27	259	15	388,5		7
	0,83	260		347		520,5		
	1	353		470		705		
5	1	47	11	62	6	93		3
6	0,62	152	21	202	12	303		5
	0,94	185		246		369		
	1	305		406		609		
7	0,79	44	16	59	9	88,5		4
	1	63		84		126		
8	0,79	44	16	59	9	88,5		4
	1	63		84		126		
9	0,79	44	16	59	9	88,5		4
	1	63		84		126		
10	0,79	44	16	59	9	88,5		4
	1	63		84		126		
11	0,79	44	16	59	9	88,5		4
	1	63		84		126		
12	0,79	44	16	59	9	88,5		4
	1	63		84		126		
13	0,79	44	16	59	9	88,5		4
	1	63		84		126		
14	0,79	44	16	59	9	88,5		4
	1	63		84		126		
15	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
16	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
17	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
18	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
19	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		

Tarefa	Prob	V3 F3		V2 F2		V1 F1		
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.	
20	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
21	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
22	0,69	190	27	253	15	379,5		7
	0,91	230		307		460,5		
	1	353		470		705		
23	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
24	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
25	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
26	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
27	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
28	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
29	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
30	0,64	104	23	138	13	207		6
	0,9	156		208		312		
	0,96	253		337		505,5		
31	1	23	34	31	19	46,5		8

Com formato diferente do apresentado na Figura 4.1(a), a Tabela A1.2 fornece os valores acumulados de probabilidade e não os valores de cada classe. A diferença na apresentação dos dados se deve ao uso desta tabela como arquivo de entrada para execução do algoritmo, que trabalha com os valores acumulados das classes de dados de entrada. A correspondência entre os dois é simples. Tomado como exemplo a tarefa 1, os valores acumulados são 0,62, 0,94 e 1,00. Portanto, os valores para cada classe são $c_{1,1} = 0,62$, $c_{1,2} = 0,32$ e $c_{1,3} = 0,04$.

Os valores de probabilidade de descarte de classes de dados de entrada para cada tarefa foram gerados aleatoriamente, através da função ALEATÓRIO() do Microsoft ExcelTM. No entanto, esta aleatoriedade foi controlada, definindo valores coerentes (sem diferenças gritantes) na distribuição probabilística das classes em cada tarefa. As classes de dados de entrada $c_{i,1}$ ($i = 1 \dots 31$) foram definidas aleatoriamente dentro dos limites 0,35 a 0,75, de modo que a probabilidade de ocorrência das classes fosse sempre maior para a classe de menor tempo de tratamento. No entanto, esta proporcionalidade nada influi no funcionamento do algoritmo e foi adotada meramente para melhor controle da variação das classes de maior l_i , que são as classes descartadas pela tarefa.

As classes seguintes foram definidas com base no valor da classe anterior, $c_{i,j-1}$, com o critério de estarem aleatoriamente entre o valor de $c_{i,j-1} + 0,01$ e $c_{i,j-1} + 0,20$. O valor de c_{i,k_i} é sempre 1,00. Quando uma tarefa tem apenas uma classe de entrada, a tarefa sempre faz o tratamento de 100% dos dados de entrada ($c_{i,k_i} = 1$).

Com dados de entrada de classes diferentes, os tempos de tratamento também são diferentes para uma mesma tarefa. Novamente a ferramenta de aleatoriedade foi usada para geração dos dados de tempo de tratamento, sempre com base no tempo original para cada tarefa obtido da Tabela A1.1. O tempo de c_{i,k_i} para o nível de operação $V_2|F_2$ foi a base de cálculo para todas as tarefas e o valor para cada classe $c_{i,j}$ é aleatoriamente 15 a 40% menor que o tempo de tratamento de $c_{i,j+1}$. Novamente usando como exemplo a tarefa 1, o tempo de tratamento de $c_{1,3} = 406$ foi retirado da Tabela A1.1. O tempo de tratamento de $c_{1,2} = 265$ é de 15 a 40% menor que $c_{1,3}$, e o tempo de $c_{1,1} = 178$ é de 15 a 40% menor que o tempo de tratamento de $c_{1,1}$.

Definida a distribuição probabilística das classes de dados de entrada e os tempos de tratamento de cada classe para cada tarefa no nível de operação intermediário $V_2|F_2$, são definidos os valores de tempo para os níveis $V_1|F_1$ e $V_3|F_3$. Sempre para a mesma classe de dados, a tarefa faz o tratamento dos dados em 75% do tempo quando operando no maior nível de operação $V_3|F_3$ e leva 50% a mais de tempo quando operando no par tensão/frequência de menor velocidade e menor consumo. Novamente citando a tarefa 1, quando está operando em $V_2|F_2$, faz o tratamento dos dados de entrada da classe $c_{1,3}$ em $406 \mu s$. Se operando em $V_1|F_1$, o tempo de tratamento é de $609 \mu s$ e quando operando no nível de maior velocidade e consumo de energia, $V_3|F_3$, o tempo de tratamento é de $305 \mu s$. O critério para escolha da relação entre os tempos de tratamento de cada nível foi definido aleatoriamente. Todos os valores de tempo foram truncados para valores inteiros.

O trabalho de Bianco *et. al.*, não nos fornece dados de consumo de energia para tratamento dos dados recebidos para cancelamento de eco acústico. Assim, foi necessário o levantamento de valores aleatórios de consumo de energia para cada tarefa. Novamente o nível de tensão/frequência intermediário $V_2|F_2$ foi usado como base para criação dos valores de consumo de energia. Os valores de consumo para $V_1|F_1$ são apenas um quarto dos valores de $V_2|F_2$. Já para $V_3|F_3$, o consumo para tratamento em cada tarefa é quatro vezes maior que o consumo de $V_2|F_2$. A

relação entre os valores de consumo novamente foi gerada aleatoriamente, assim como os valores de $V_2|F_2$.

As fórmulas de aleatoriedade utilizadas para levantamento de cada valor das células da tabela A1.2, assim como todas as fórmulas usadas nas tabelas deste apêndice, além dos arquivos Excel originais, podem ser obtidas através do email pedropepe@gmail.com.

Com a tabela de distribuição probabilística das classes de dados de entrada, tempos de execução e consumo de energia para cada tarefa, foram definidos os 4 sistemas nos quais as tarefas do sistema foram mapeadas. Os sistemas contam com 1, 2, 4 e 8 processadores, e a distribuição das tarefas nos processadores seguem o mapeamento da Tabela A1.3. Cada coluna indica o sistema usado e em qual processador cada tarefa está mapeada.

Quando mapeadas em sistemas com mais de um processador, o tempo de comunicação de dados entre as tarefas foi considerado e somado ao tempo de execução da tarefa que recebe os dados, explicito na Tabela A1.1. Quando esta tarefa faz o tratamento dos dados de entrada em níveis de tensão/frequência diferentes, o tempo extra de comunicação também sofre alteração, pois foi considerado que o meio de comunicação seria interno ao processador que recebe os dados, e a mudança de nível de operação também altera a velocidade de comunicação. Esta

Tabela A1.3: Mapeamento das tarefas nos sistemas experimentais

Tarefa	GMDFa-1	GMDFa-2	GMDFa-4	GMDFa-8
1	1	1	1	1
2	1	2	2	2
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	2
9	1	1	2	3
10	1	1	2	4
11	1	2	3	5
12	1	2	3	6
13	1	2	4	7
14	1	2	4	8
15	1	1	1	1
16	1	1	1	2
17	1	1	2	3
18	1	1	2	4
19	1	2	3	5
20	1	2	3	6
21	1	2	4	7
22	1	2	4	8
23	1	1	1	1
24	1	1	1	2
25	1	1	2	3
26	1	1	2	4
27	1	2	3	5
28	1	2	3	6
29	1	2	4	7
30	1	2	4	8
31	1	1	1	1

suposição foi unicamente para facilitar a montagem das tabelas de entrada para o código.

Após mapeamento, o DFG original foi alterado e 4 novos DFGs foram obtidos. A Tabela A1.4 faz um resumo das quantidades de arestas e caminhos de execução para a aplicação em cada um dos quatro sistemas, assim como os tempos de resposta estabelecidos, citados anteriormente.

Tabela A1.4: Dados da Aplicação Cancelador de Eco Acústico

Aplicação (nós, arestas)	Nº PEs	Tempo de resposta (ms)	Após Escalonamento	
			Nº Arestas	Nº Caminhos
<i>GMDFa</i> (31, 53)	1	8,0	30	1
	2	7,0	46	4
	4	5,0	52	8
	8	3,5	52	16

A1.2 Sonar

Apresentado por Woodside e Monforton [41], a aplicação denominada Sonar pode ser descrita pela estrutura de tarefas apresentada no DFG da Figura A1.2.

A aplicação conta com 119 tarefas divididas em cinco estruturas paralelas para tratamento dos dados de entrada. Como citado anteriormente, o tempo de tratamento dos dados de entrada por estas 119 tarefas deve ficar abaixo de $M = 1350 \text{ ms}$.

Além do DFG, o trabalho de Woodside e Monforton [41] também fornece o tempo de execução das tarefas, mostrados na terceira coluna da Tabela A1.5, e utilizados para obtenção dos dados necessários para uso da aplicação Sonar em nosso trabalho.

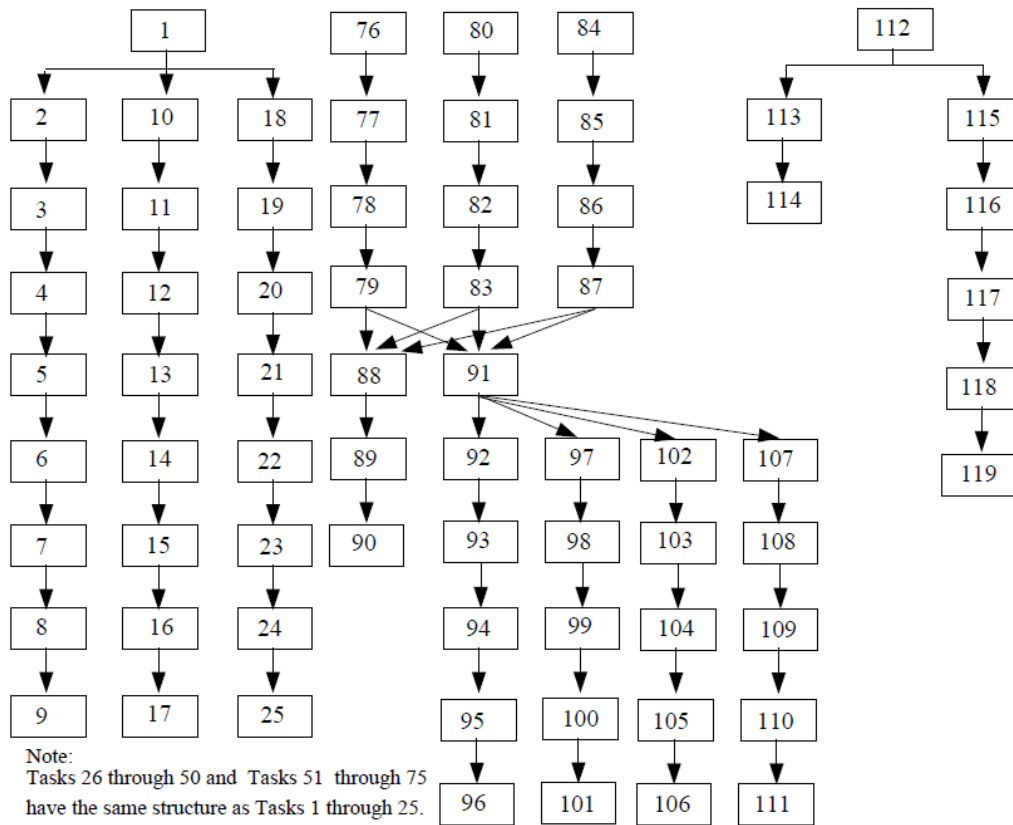


Figura A1.2: DFG de processamento de dados de Sonar [41]

Tabela A1.5: Tempos de execução das tarefas de sonar [41]

Task	Function	p(i) (ms)	Bytes to next task
1,26,51,76,80,84	RemoveDC	170,24	2048
2,10,18,27,35,43, 52,60,68,77,81,85	Bandshift	62,72	2048
3,11,19,28,36,44, 53,61,69,78,82,86	FIR Filter	227,84	512
4,12,20,29,37,45, 54,62,70,79,83,87	FFT	409,6	32K
5,13,21,30,38,46,55, 63,71,92,97,102,107	Hanning Window	21,28	32K
6,14,22,31,39,47,56, 64,72,93,98,103,108	Power	11,2	16K
7,15,23,32,40,48,57, 65,73,94,99,104,109	Integrate	3,07	1024
8,16,24,33,41,49,58, 66,74,95,100,105,110	Normalize	0,31	1024
9,17,25,34,42,50,59, 67,75,96,101,106,111	Quantize	0,28	0
88	Cross Multiply	22,24	16K
89	Integrate2	6,14	16K
90	Bearing	1,23	0
91	Beamform	99,68	16K
112	Form Cell	75	4096
113	Power2	25,5	1
114	Normalize2	12	0
115	Cancatenate	75	8K
116	FFT2	1728	8K
117	Power3	52,5	8K
118	Normalize3	22,5	8K
119	Quantize2	21	0

Outro dado retirado do trabalho de Woodside e Monforton [41] é o mapeamento das tarefas em sistemas com 5 a 9 processadores. No presente trabalho, os mapeamentos em 1 processador (não abordado em Woodside e Monforton) e 7 processadores (sem perdas de comunicação) foram os mapeamentos utilizados. A distribuição das tarefas nos 7 processadores é apresentado na Tabela A1.6.

Tabela A1.6: Mapeamento das tarefas de sonar em 7 processadores [41]

PE	Tarefas	Workload (ms)
P1	1-18,77,112-114	1989,58
P2	81,11-119	2044,9
P3	19-36,43	1991
P4	37-42,44-50,52-59,85	2007,9
P5	51,60-76	1864
P6	78,79,82,83,86-111	2205,4
P7	80,84	391,7

No entanto, como observado para a aplicação de cancelamento de eco acústico, os dados fornecidos não são suficientes para utilização da aplicação Sonar neste trabalho, e outros dados de funcionamento do sistema foram criteriosamente criados a partir dos dados apresentados por Woodside e Monforton [41].

Como feito anteriormente para o cancelador de eco acústico, partindo da tabela de tempos de execução das tarefas, definidos na terceira coluna da Tabela A1.5, foi definida uma nova tabela, com tempos de execução em três diferentes níveis de tensão/frequência, $V_1|F_1$, $V_2|F_2$ e $V_3|F_3$, também com variação de uma a quatro classes de dados de entrada para cada tarefa. Os dados gerados para cada tarefa estão descritos na Tabela A1.7.

Novamente o tempo para execução de uma tarefa, definido na Tabela A1.5, foi usado como o tempo para tratamento de 100% dos dados de entrada com tensão/frequência $V_2|F_2$.

As tarefas do sistema foram atribuídas de uma a quatro classes de dados de entrada, sendo que tarefas do mesmo tipo, identificadas através do agrupamento das tarefas fornecido na Tabela A1.5, apresentam na Tabela A1.7 a mesma distribuição de classes de entrada e os mesmos valores de tempo de execução e consumo de energia.

A Tabela A1.7 (dividida em 6 conjuntos de tarefas) também difere do formato da tabela apresentada na Figura 4.1(a), fornecendo valores acumulados de probabilidade e não os valores de cada classe, também por ser a base do arquivo de entrada para a execução dos algoritmos para a aplicação Sonar.

Os valores de probabilidade de descarte de classes de dados de entrada para cada tarefa foram gerados aleatoriamente, através da função ALEATÓRIO() do Microsoft ExcelTM. No entanto, esta aleatoriedade foi controlada, definindo valores coerentes (sem diferenças gritantes) para a probabilidade de ocorrência de cada classe em cada tarefa. As classes de dados de entrada $c_{i,1}$ ($i = 1 \dots 119$), foram definidas aleatoriamente dentro dos limites 0,40 a 0,75. As classes seguintes foram definidas com base no valor da classe anterior, $c_{i,j-1}$, com o critério de estarem aleatoriamente entre o valor de $c_{i,j-1} + 0,01$ e $c_{i,j-1} + 0,20$. O valor de c_{i,k_i} é sempre 1,00. Quando uma tarefa tem apenas uma classe de entrada, a tarefa sempre faz o tratamento de 100% dos dados de entrada ($c_{i,k_i} = 1$). No caso desta aplicação, quando a tarefa conta com apenas duas classes de dados de entrada, a aleatoriedade do valor de $c_{i,1}$ ($i = 1 \dots 119$), está definida entre 0,60 e 0,90, evitando que estas tarefas tenham peso muito grande quando descartarem classes de dados de entrada.

Com dados de entrada diferentes, os tempos de tratamento também são diferentes para classes de dados de entrada de uma mesma tarefa. Novamente a ferramenta de aleatoriedade foi usada para geração dos dados de tempo de tratamento, sempre com base no tempo original para cada tarefa obtido da Tabela A1.5. O tempo de c_{i,k_i} para o nível de operação $V_2|F_2$ foi a base de cálculo para todas as tarefas e o valor para cada classe $c_{i,j}$ é aleatoriamente 15 a 40% menor que o tempo de tratamento de $c_{i,j+1}$.

Tabela A1.7: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia

Tarefa	Prob	V3 F3		V2 F2		V1 F1		Tarefa	Prob	V3 F3		V2 F2		V1 F1	
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.			Tempo	Energ.	Tempo	Energ.	Tempo	Energ.
1	1	85,12	36	170,24	9	340,48	1	22	0,66	3,87	4	7,73	1	15,46	1
2	0,67	26,02	76	52,03	19	104,06	5	23	1	5,60		11,20		22,40	
	1	31,36		62,72		125,44			0,46	0,80	4	1,60	1	3,20	1
3	0,45	54,59	4	109,18	1	218,36	1	24	0,79	1,27		2,54		5,08	
	0,81	81,57		163,14		326,28			1	1,54		3,07		6,14	
	1	113,92		227,84		455,68			0,46	0,07	28	0,13	7	0,26	2
4	0,49	79,04	68	158,07	17	316,14	4	25	0,7	0,09		0,17		0,34	
	0,76	96,62		193,24		386,48			0,89	0,11		0,21		0,42	
	0,93	159,59		319,18		638,36			1	0,16		0,31		0,62	
	1	204,80		409,60		819,20			1	0,14	8	0,28	2	0,56	1
5	1	10,64	48	21,28	12	42,56	3	26	1	85,12	36	170,24	9	340,48	2
	0,66	3,81	4	7,61	1	15,22	1	27	0,67	26,02	76	52,03	19	104,06	5
1	5,60		11,20		22,40		1		31,36		62,72		125,44		
7	0,46	0,93	4	1,86	1	3,72	1	28	0,45	54,59	4	109,18	1	218,36	1
	0,79	1,20		2,39		4,78			0,81	81,57		163,14		326,28	
	1	1,54		3,07		6,14			1	113,92		227,84		455,68	
8	0,46	0,08	28	0,16	7	0,32	2	29	0,49	79,04	68	158,07	17	316,14	4
	0,7	0,10		0,20		0,40			0,76	96,62		193,24		386,48	
	0,89	0,13		0,26		0,52			0,93	159,59		319,18		638,36	
	1	0,16		0,31		0,62			1	204,80		409,60		819,20	
9	1	0,14	8	0,28	2	0,56	1	30	1	10,64	48	21,28	12	42,56	3
10	0,67	21,56	76	43,11	19	86,22	5	31	0,66	3,81	4	7,61	1	15,22	1
	1	31,36		62,72		125,44			1	5,60		11,20		22,40	
11	0,45	48,17	4	96,33	1	192,66	1	32	0,46	0,93	4	1,86	1	3,72	1
	0,81	79,46		158,91		317,82			0,79	1,20		2,39		4,78	
	1	113,92		227,84		455,68			1	1,54		3,07		6,14	
12	0,49	76,12	68	152,23	17	304,46	4	33	0,46	0,08	28	0,16	7	0,32	2
	0,76	92,19		184,38		368,76			0,7	0,10		0,20		0,40	
	0,93	151,83		303,65		607,30			0,89	0,13		0,26		0,52	
	1	204,80		409,60		819,20			1	0,16		0,31		0,62	
13	1	10,64	48	21,28	12	42,56	3	34	1	0,14	8	0,28	2	0,56	1
14	0,66	3,56	4	7,12	1	14,24	1	35	0,67	21,56	76	43,11	19	86,22	5
	1	5,60		11,20		22,40			1	31,36		62,72		125,44	
15	0,46	0,95	4	1,89	1	3,78	1	36	0,45	48,17	4	96,33	1	192,66	1
	0,79	1,20		2,39		4,78			0,81	79,46		158,91		317,82	
	1	1,54		3,07		6,14			1	113,92		227,84		455,68	
16	0,46	0,07	28	0,14	7	0,28	2	37	0,49	76,12	68	152,23	17	304,46	4
	0,7	0,09		0,17		0,34			0,76	92,19		184,38		368,76	
	0,89	0,10		0,19		0,38			0,93	151,83		303,65		607,30	
	1	0,16		0,31		0,62			1	204,80		409,60		819,20	
17	1	0,14	8	0,28	2	0,56	1	38	1	10,64	48	21,28	12	42,56	3
18	0,67	23,77	76	47,53	19	95,06	5	39	0,66	3,56	4	7,12	1	14,24	1
	1	31,36		62,72		125,44			1	5,60		11,20		22,40	
19	0,45	51,50	4	102,99	1	205,98	1	40	0,46	0,95	4	1,89	1	3,78	1
	0,81	78,46		156,91		313,82			0,79	1,20		2,39		4,78	
	1	113,92		227,84		455,68			1	1,54		3,07		6,14	
20	0,49	89,31	68	178,61	17	357,22	4	41	0,46	0,07	28	0,14	7	0,28	2
	0,76	114,89		229,77		459,54			0,7	0,09		0,17		0,34	
	0,93	158,06		316,11		632,22			0,89	0,10		0,19		0,38	
	1	204,80		409,60		819,20			1	0,16		0,31		0,62	
21	1	10,64	48	21,28	12	42,56	3								

Tabela A1.7 (continuação)

Tarefa	Prob	V3 F3		V2 F2		V1 F1	
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.
42	1	0,14	8	0,28	2	0,56	1
43	0,67	23,77	76	47,53	19	95,06	5
	1	31,36		62,72		125,44	
44	0,45	51,50	4	102,99	1	205,98	1
	0,81	78,46		156,91		313,82	
	1	113,92		227,84		455,68	
45	0,49	89,31	68	178,61	17	357,22	4
	0,76	114,89		229,77		459,54	
	0,93	158,06		316,11		632,22	
	1	204,80		409,60		819,20	
46	1	10,64	48	21,28	12	42,56	3
47	0,66	3,87	4	7,73	1	15,46	1
	1	5,60		11,20		22,40	
48	0,46	0,80	4	1,60	1	3,20	1
	0,79	1,27		2,54		5,08	
	1	1,54		3,07		6,14	
49	0,46	0,07	28	0,13	7	0,26	2
	0,7	0,09		0,17		0,34	
	0,89	0,11		0,21		0,42	
	1	0,16		0,31		0,62	
50	1	0,14	8	0,28	2	0,56	1
51	1	85,12	36	170,24	9	340,48	2
52	0,67	26,02	76	52,03	19	104,06	5
	1	31,36		62,72		125,44	
53	0,45	54,59	4	109,18	1	218,36	1
	0,81	81,57		163,14		326,28	
	1	113,92		227,84		455,68	
54	0,49	79,04	68	158,07	17	316,14	4
	0,76	96,62		193,24		386,48	
	0,93	159,59		319,18		638,36	
	1	204,80		409,60		819,20	
55	1	10,64	48	21,28	12	42,56	3
56	0,66	3,81	4	7,61	1	15,22	1
	1	5,60		11,20		22,40	
57	0,46	0,93	4	1,86	1	3,72	1
	0,79	1,20		2,39		4,78	
	1	1,54		3,07		6,14	
58	0,46	0,08	28	0,16	7	0,32	2
	0,7	0,10		0,20		0,40	
	0,89	0,13		0,26		0,52	
	1	0,16		0,31		0,62	
59	1	0,14	8	0,28	2	0,56	1
60	0,67	21,56	76	43,11	19	86,22	5
	1	31,36		62,72		125,44	
61	0,45	48,17	4	96,33	1	192,66	1
	0,81	79,46		158,91		317,82	
	1	113,92		227,84		455,68	
62	0,49	76,12	68	152,23	17	304,46	4
	0,76	92,19		184,38		368,76	
	0,93	151,83		303,65		607,30	
	1	204,80		409,60		819,20	
63	1	10,64	48	21,28	12	42,56	3
64	0,66	3,56	4	7,12	1	14,24	1
	1	5,60		11,20		22,40	
65	0,46	0,95	4	1,89	1	3,78	1
	0,79	1,20		2,39		4,78	
	1	1,54		3,07		6,14	
66	0,46	0,07	28	0,14	7	0,28	2
	0,7	0,09		0,17		0,34	
	0,89	0,10		0,19		0,38	
	1	0,16		0,31		0,62	
67	1	0,14	8	0,28	2	0,56	1
68	0,67	23,77	76	47,53	19	95,06	5
	1	31,36		62,72		125,44	
69	0,45	51,50	4	102,99	1	205,98	1
	0,81	78,46		156,91		313,82	
	1	113,92		227,84		455,68	
70	0,49	89,31	68	178,61	17	357,22	4
	0,76	114,89		229,77		459,54	
	0,93	158,06		316,11		632,22	
	1	204,80		409,60		819,20	
71	1	10,64	48	21,28	12	42,56	3
72	0,66	3,87	4	7,73	1	15,46	1
	1	5,60		11,20		22,40	
73	0,46	0,80	4	1,60	1	3,20	1
	0,79	1,27		2,54		5,08	
	1	1,54		3,07		6,14	
74	0,46	0,07	28	0,13	7	0,26	2
	0,7	0,09		0,17		0,34	
	0,89	0,11		0,21		0,42	
	1	0,16		0,31		0,62	
75	1	0,14	8	0,28	2	0,56	1
76	0,42	40,36	40	80,71	10	161,42	3
	0,7	55,13		110,25		220,50	
	0,87	71,46		142,91		285,82	
	1	85,12		170,24		340,48	
77	1	31,36	72	62,72	18	125,44	5
78	0,7	91,47	72	182,94	18	365,88	5
	1	113,92		227,84		455,68	
79	0,59	104,19	72	208,38	18	416,76	5
	0,82	160,09		320,18		640,36	
	1	204,80		409,60		819,20	
80	0,42	40,36	40	80,71	10	161,42	3
	0,7	55,13		110,25		220,50	
	0,87	71,46		142,91		285,82	
	1	85,12		170,24		340,48	
81	1	31,36	72	62,72	18	125,44	5
82	0,7	91,47	72	182,94	18	365,88	5
	1	113,92		227,84		455,68	

Tabela A1.7 (continuação)

Tarefa	Prob	V3 F3		V2 F2		V1 F1	
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.
83	0,59	104,19	72	208,38	18	416,76	5
	0,82	160,09		320,18		640,36	
	1	204,80		409,60		819,20	
84	0,42	40,36	40	80,71	10	161,42	3
	0,7	55,13		110,25		220,50	
	0,87	71,46		142,91		285,82	
	1	85,12		170,24		340,48	
85	1	31,36	72	62,72	18	125,44	5
86	0,7	91,47	72	182,94	18	365,88	5
	1	113,92		227,84		455,68	
87	0,59	104,19	72	208,38	18	416,76	5
	0,82	160,09		320,18		640,36	
	1	204,80		409,60		819,20	
88	0,52	5,37	68	10,74	17	21,48	4
	0,77	6,73		13,46		26,92	
	0,89	8,59		17,18		34,36	
	1	11,12		22,24		44,48	
89	1	3,07	48	6,14	12	12,28	3
90	0,79	0,40	60	0,80	15	1,60	4
	1	0,62		1,23		2,46	
91	0,74	23,79	64	47,58	16	95,16	4
	0,95	33,26		66,52		133,04	
	1	49,84		99,68		199,36	
92	0,53	4,61	56	9,21	14	18,42	4
	0,76	6,63		13,25		26,50	
	0,95	7,81		15,61		31,22	
	1	10,64		21,28		42,56	
93	1	5,60	44	11,20	11	22,40	3
94	0,83	1,02	12	2,03	3	4,06	1
	1	1,54		3,07		6,14	
95	0,63	0,09	24	0,18	6	0,36	2
	0,9	0,11		0,21		0,42	
	1	0,16		0,31		0,62	
	0,59	0,06	76	0,11	19	0,22	5
96	0,86	0,09		0,17		0,34	
	0,96	0,11		0,21		0,42	
	1	0,14		0,28		0,50	
	0,53	4,61	56	9,21	14	18,42	4
97	0,76	6,63		13,25		26,50	
	0,95	7,81		15,61		31,22	
	1	10,64		21,28		42,56	
	1	5,60	44	11,20	11	22,40	3
98	0,83	1,02	12	2,03	3	4,06	1
99	1	1,54		3,07		6,14	
	0,63	0,09	24	0,18	6	0,36	2
100	0,9	0,11		0,21		0,42	
	1	0,16		0,31		0,62	
	0,59	0,06	76	0,11	19	0,22	5
101	0,86	0,09		0,17		0,34	
	0,96	0,11		0,21		0,42	
	1	0,14		0,28		0,50	
102	0,53	4,61	56	9,21	14	18,42	4
	0,76	6,63		13,25		26,50	
	0,95	7,81		15,61		31,22	
	1	10,64		21,28		42,56	
103	1	5,60	44	11,20	11	22,40	3
104	0,83	1,02	12	2,03	3	4,06	1
	1	1,54		3,07		6,14	
105	0,63	0,09	24	0,18	6	0,36	2
	0,9	0,11		0,21		0,42	
	1	0,16		0,31		0,62	
106	0,59	0,06	76	0,11	19	0,22	5
	0,86	0,09		0,17		0,34	
	0,96	0,11		0,21		0,42	
107	1	0,14		0,28		0,50	
	0,53	4,61	56	9,21	14	18,42	4
	0,76	6,63		13,25		26,50	
	0,95	7,81		15,61		31,22	
108	1	10,64		21,28		42,56	
	1	5,60	44	11,20	11	22,40	3
	0,83	1,02	12	2,03	3	4,06	1
109	1	1,54		3,07		6,14	
	0,63	0,09	24	0,18	6	0,36	2
	0,9	0,11		0,21		0,42	
110	1	0,16		0,31		0,62	
	0,59	0,06	76	0,11	19	0,22	5
	0,86	0,09		0,17		0,34	
	0,96	0,11		0,21		0,42	
111	1	0,14		0,28		0,50	
	0,48	12,35	64	24,70	16	49,40	4
	0,7	16,81		33,62		67,24	
	0,87	27,09		54,18		108,36	
112	1	37,50		75,00		150,00	
	1	12,75	36	25,50	9	51,00	2
	0,88	4,07	4	8,14	1	16,28	1
113	1	6,00		12,00		24,00	
	0,59	15,55	52	31,09	13	62,18	3
114	0,86	23,93		47,85		95,70	
	1	37,50		75,00		150,00	
	0,41	323,42	36	646,84	9	1293,68	2
	0,82	450,67		901,34		1802,68	
115	0,94	712,35		1424,70		2849,40	
	1	864,00		1728,00		3456,00	
	1	26,25	4	52,50	1	105,00	1
116	0,76	7,56		15,12	11	30,24	3
117	1	11,25	0	22,50		45,00	
	0,43	5,62	60	11,23	15	22,46	4
118	0,71	7,34		14,68		29,36	
	1	10,50		21,00		42,00	

Definidas as probabilidades de distribuição das classes de entrada e os tempos de tratamento de cada classe para cada tarefa no nível de operação intermediário $V_2|F_2$, são definidos os valores de tempo para os níveis $V_1|F_1$ e $V_3|F_3$. Sempre para a mesma classe de dados, a tarefa faz o tratamento dos dados na metade do tempo quando operando no maior nível de operação $V_3|F_3$ e leva o dobro do tempo quando operando no par tensão/frequência de menor velocidade e menor consumo $V_1|F_1$.

Assim como o trabalho de Bianco *et. al.*, Woodside e Monforton não fornecem dados de consumo de energia para tratamento dos dados recebidos pelo sonar. Assim, foi necessário o levantamento de valores aleatórios de consumo de energia para cada tarefa. Novamente o par

intermediário $V_2|F_2$ foi usado como base para criação dos valores de consumo de energia. Os valores de consumo para $V_1|F_1$ são apenas um quarto dos valores de $V_2|F_2$. Já para $V_3|F_3$, o consumo para tratamento em cada tarefa é quatro vezes maior que o consumo de $V_2|F_2$. A relação entre os valores de consumo de energia para cada nível foi baseada na relação quadrática entre o nível de operação (tempo de execução) e o consumo de energia, abordado no Capítulo 2. Aumentando o nível de operação, o tempo de execução cai pela metade, enquanto o consumo de energia aumenta em 4 vezes.

Após o mapeamento em 1 e 7 processadores, o DFG original foi alterado e 2 novos DFGs foram obtidos. A Tabela A1.8 mostra as quantidades de arestas e caminhos de execução para a aplicação nos dois sistemas, assim como os tempos de resposta estabelecidos.

Tabela A1.8: Dados da Aplicação Sonar

Aplicação (nós, arestas)	Nº PEs	Tempo de resposta (ms)	Após Escalonamento	
			Nº Arestas	Nº Caminhos
<i>Sonar</i> (119, 116)	1	1350	118	1
	7	1350	93	150800

A1.3 Vocoder

Gerstlauer *et al.* [14] apresentam um sistema de codificação/decodificação de voz denominado Vocoder, que conta com um estrutura de 133 tarefas, onde o tempo para tratamento dos dados de entrada, imposto pela frequência de amostragem, é de $M = 20$ ms.

A estrutura, baseando-se nas informações fornecidas por Gerstlauer *et al.* [14], é descrita pelo DFG da Figura A1.3, elaborado pela aluna de mestrado da Unicamp, Daiane Angolini [4].

Gerstlauer *et al.* também fornecem a quantidade de ciclos de relógio utilizados pelo sistema para executar cada uma das tarefas, além do número de instruções de cada tarefa, como descrito na Tabela A1.9. Os valores máximos de ciclos de execução da coluna *cycles* foram usados como base para o cálculo do tempo de execução de cada tarefa, necessários neste trabalho.

Novamente alguns dados foram criados para preenchimento das tabelas de entrada dos algoritmos mas, diferentemente das aplicações de cancelamento de eco e sonar, a criação dos dados de consumo de energia foi baseada em dados encontrados no trabalho de Gerstlauer *et al.* [14]. Apenas a distribuição das classes de entrada para cada tarefa ainda foi feita aleatoriamente. A todas as tarefas do sistema foram atribuídas 3 classes de dados de entrada.

Os valores de probabilidade de descarte de classes de dados de entrada para cada tarefa foram gerados através da função ALEATÓRIO() do Microsoft ExcelTM. No entanto, esta aleatoriedade foi controlada, definindo valores coerentes (sem diferenças gritantes) para a probabilidade de ocorrência de cada classe em cada tarefa. As classes de dados de entrada $c_{i,1}$ ($i = 1 \dots 133$) foram definidas aleatoriamente dentro dos limites 0,40 a 0,75. As classes seguintes foram definidas com base no valor da classe anterior, $c_{i,j-1}$, com o critério de estarem aleatoriamente entre o valor de $c_{i,j-1} + 0,01$ e $c_{i,j-1} + 0,20$. O valor de c_{i,k_i} é sempre 1,00.

Desde que temos disponível o número de ciclos de execução para cada tarefa, foram considerados processadores com 4 pares tensão/frequência de operação, sendo que são levadas em conta apenas as frequências em que estes processadores podem operar: 60, 100, 400 e 800MHz. Estes valores foram definidos apenas buscando a máxima exercitação dos algoritmos, sem levar em conta qualquer proximidade com o comportamento de processadores reais. De modo a manter a coerência de nomenclaturas, foram definidos os níveis de operação $V_1|F_1$ (60MHz), $V_2|F_2$ (100MHz), $V_3|F_3$ (400MHz) e $V_4|F_4$ (800MHz).

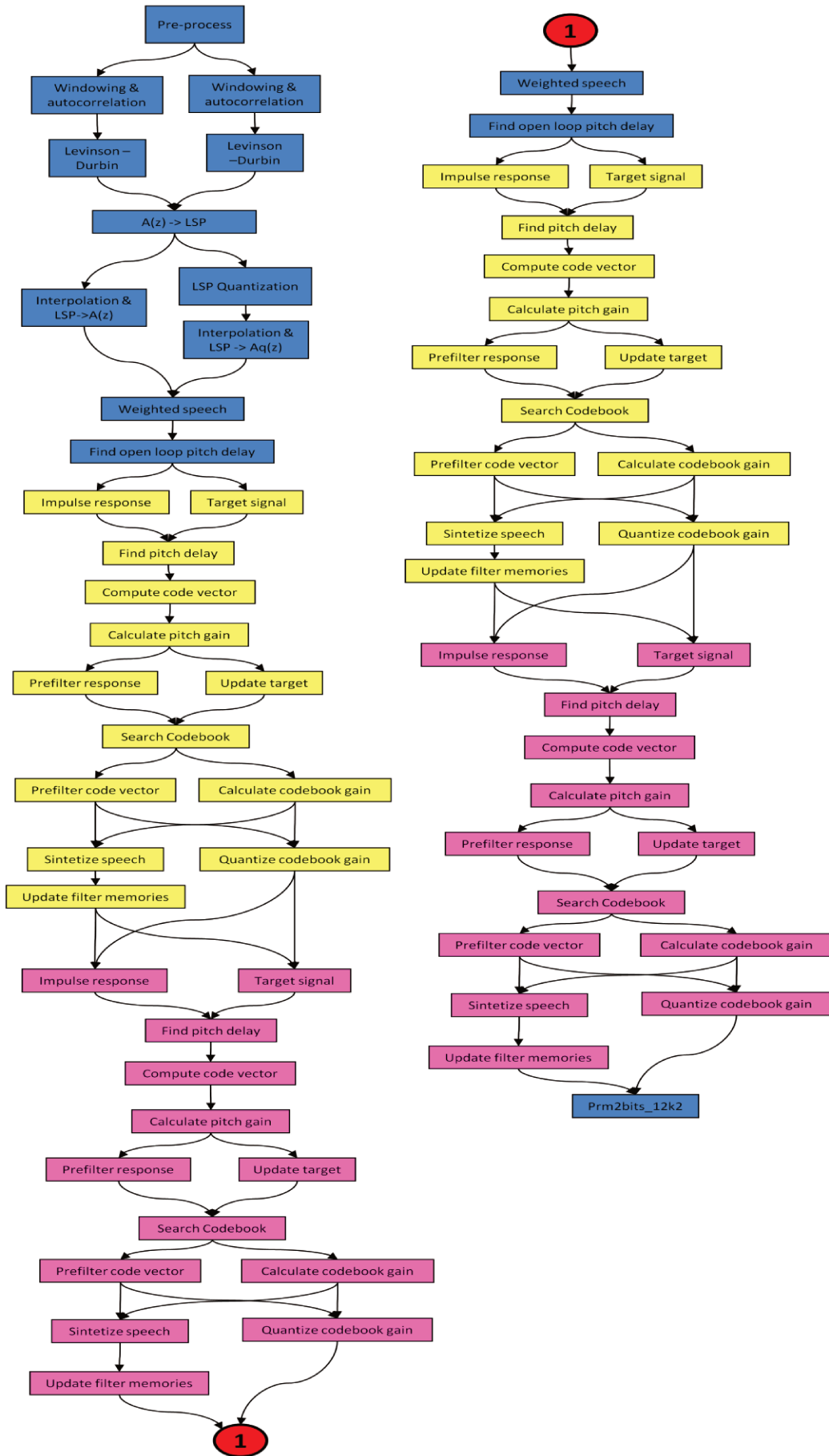


Figura A1.3: DFG do Vocoder [4]

Tabela A1.9: Ciclos de relógio e número de instruções para execução das tarefas do Vocoder [14]

Routine	calls	cycles			instructions			
		max	total	avg.	max	total	avg	
FAutocorr	18ae	6	104958	629748	104958	53938	323628	53938
FAz_lsp	16df	6	41258	246714	41119	24798	148315	24719
FChebbs	1822	628	322	202216	322	200	125600	200
FCoder_12k2	11cc	3	2715917	8142878	2714292	1561164	4680860	1560286
FConvolve	19a0	24	29702	712848	29702	16181	388344	16181
FEnc_lag6	2852	12	99	928	77	52	503	41
FG_code	2bd2	12	937	11244	937	842	10104	842
FG_pitch	28ae	12	2826	33818	2818	1625	19449	1620
FGet_lsp_pol	36c4	36	1530	55080	1530	963	34668	963
FInt_lpc	30e4	3	14879	44629	14876	9471	28409	9469
FInt_lpc2	3140	3	7618	22840	7613	4866	14591	4863
FInterpol_6	37e6	70	270	18340	262	134	9180	131
FInv_sqrt	3788	252	135	30946	122	77	17668	70
FLag_max	25b0	18	86950	916735	50929	60707	641995	35666
FLag_window	2fd2	6	443	2658	443	318	1908	318
FLevinson	2cb7	6	18650	111788	18631	10373	62168	10361
FLsf_lsp	358b	6	404	2421	403	239	1434	239
FLsf_wt	34af	6	419	2490	415	293	1758	293
FLsp_Az	363b	18	3628	65195	3621	2300	41347	2297
FLsp_lsf	35bc	6	1960	11616	1936	945	5606	934
FNorm_Corr	271b	12	64233	733169	61097	39419	448217	37351
FPitch_fr6	2669	12	66638	757923	63160	40623	460564	38380
FPitch_col	248b	6	158656	946007	157667	111212	664359	110726
FPre_Process	383c	3	15248	45744	15248	10268	30804	10268
FPred_lt_6	2c34	12	21995	263828	21985	11516	138150	11512
FPrm2bits_12k2	3013	3	9599	28592	9530	4731	14152	4717
FQ_plsf_5	318f	3	116512	348864	116288	75269	225521	75173
FReorder_lsf	361c	6	207	1239	206	126	756	126
FResidu	29d0	36	12528	451008	12528	8063	290268	8063
FSet_zero	34f7	9	1299	3119	346	1289	3029	336
FSyn_filt	306f	72	11124	792192	11002	5885	418248	5809
FVq_subvec	3331	12	22498	184014	15334	14900	122334	10194
FVq_subvec_s	33be	3	48033	143853	47951	30655	91860	30620
FWeight_Ai	2a20	48	132	6336	132	110	5280	110
Fbuild_code	2363	12	5730	68159	5679	3190	37976	3164
Fcode_10i40_35bits	19f7	12	246805	2957399	246449	132574	1589235	132436
Fcor_h	1c25	12	80454	965331	80444	44758	537033	44752
Fcor_h_x	1a74	12	31074	371601	30966	17096	204580	17048
Fencoder_reset	47a3	1	3553	3553	3553	3322	3322	3322
Fq_gain_code	2a7d	12	2471	28688	2390	1415	16661	1388
Fq_gain_pitch	2a40	12	584	6450	537	299	3402	283
Fq-p	1bf7	120	60	6360	53	31	3120	26
Fsearch_10i40	1d21	12	124153	1485073	123756	64517	772537	64378
Fset_sign	1b17	12	4633	55127	4593	2673	31697	2641

A relação de tempos de execução de cada tarefa, para cada um dos quatro níveis de operação, foi obtida dividindo o número máximo de ciclos de execução da tarefa (coluna *cycles* da Tabela A1.9) pela velocidade do nível de operação do processador. A Tabela A1.10 (dividida em 6 conjuntos de tarefas) apresenta estes valores de tempo em μs , sendo que estes valores foram atribuídos ao tempo de execução da tarefa quando tratando 100% dos dados de entrada.

Tabela A1.10 (Continuação)

Tarefa	Prob	V4 F4		V3 F3		V2 F2		V1 F1	
		Tempo	Energ	Tempo	Energ	Tempo	Energ	Tempo	Energ
46	0,69 0,91 1	0,001 0,001 0,001	1 1 1	0,003 0,003 0,003	1 1 1	0,010 0,010 0,010	1 1 1	0,017 0,017 0,017	1 1 1
47	0,45 0,68 1	1,776 2,311 3,089	252 1 1	3,553 4,623 6,178	84 1 1	14,210 18,490 24,710	14 1 1	23,683 30,817 41,183	5 1 1
48	0,5 0,77 1	7,785 10,365 13,905	1044 1 1	15,570 20,730 27,810	348 1 1	62,280 82,920 111,240	58 1 1	103,800 138,200 185,400	21 1 1
49	0,66 0,93 1	7,091 9,535 13,905	1044 1 1	14,183 19,070 27,810	348 1 1	56,730 76,280 111,240	58 1 1	94,550 127,133 185,400	21 1 1
50	0,5 0,71 1	8,246 11,573 15,660	1440 1 1	16,493 23,145 31,320	480 1 1	65,970 92,580 125,280	80 1 1	109,950 154,300 208,800	29 1 1
51	0,73 0,94 1	5,955 9,136 13,905	1044 1 1	11,910 18,273 27,810	348 1 1	47,640 73,090 111,240	58 1 1	79,400 121,817 185,400	21 1 1
52	0,6 0,85 1	8,453 12,748 15,660	1440 1 1	16,905 25,495 31,320	480 1 1	67,620 101,980 125,280	80 1 1	112,700 169,967 208,800	29 1 1
53	0,42 0,69 1	7,095 11,401 13,905	1044 1 1	14,190 22,803 27,810	348 1 1	56,760 91,210 111,240	58 1 1	94,600 152,017 185,400	21 1 1
54	0,63 0,93 1	37,075 55,645 83,298	7308 1 1	74,150 111,290 166,595	2436 1 1	296,600 445,160 666,380	406 1 1	494,333 741,933 1110,633	145 1 1
55	0,7 0,95 1	0,071 0,099 0,124	36 1 1	0,143 0,198 0,248	12 1 1	0,570 0,790 0,990	2 1 1	0,950 1,317 1,650	1 1 1
56	0,63 0,93 1	15,891 19,974 27,494	2070 1 1	31,783 39,948 54,988	690 1 1	127,130 159,790 219,950	115 1 1	211,883 266,317 366,583	41 1 1
57	0,74 0,95 1	22,788 30,344 37,128	2898 1 1	45,575 60,688 74,255	966 1 1	182,300 242,750 297,020	161 1 1	303,833 404,583 495,033	58 1 1
58	0,61 0,88 1	2,378 2,845 3,533	288 1 1	4,755 5,690 7,065	96 1 1	19,020 22,760 28,260	16 1 1	31,700 37,933 47,100	6 1 1
59	0,55 0,85 1	0,400 0,593 0,730	54 1 1	0,800 1,185 1,460	18 1 1	3,200 4,740 5,840	3 1 1	5,333 7,900 9,733	1 1 1
60	0,52 0,81 1	0,094 0,126 0,150	36 1 1	0,188 0,253 0,300	12 1 1	0,750 1,010 1,200	2 1 1	1,250 1,683 2,000	1 1 1
61	0,49 0,75 1	0,038 0,045 0,075	36 1 1	0,075 0,090 0,150	12 1 1	0,300 0,360 0,600	2 1 1	0,500 0,600 1,000	1 1 1
62	0,59 0,94 1	22,224 26,803 38,843	3060 1 1	44,448 53,605 77,685	1020 1 1	177,790 214,420 310,740	170 1 1	296,317 357,367 517,900	61 1 1
63	0,66 0,87 1	2,921 3,949 5,791	468 1 1	5,843 7,898 11,583	156 1 1	23,370 31,590 46,330	26 1 1	38,950 52,650 77,217	9 1 1
64	0,71 0,93 1	42,630 65,843 100,568	8046 1 1	85,260 131,685 201,135	2682 1 1	341,040 526,740 804,540	447 1 1	568,400 877,900 1340,900	160 1 1
65	0,59 0,95 1	78,995 99,878 155,191	11610 1 1	157,990 199,755 310,383	3870 1 1	631,960 799,020 1241,530	645 1 1	1053,267 1331,700 2069,217	230 1 1
66	0,47 0,83 1	4,349 5,639 7,163	558 1 1	8,698 11,278 14,325	186 1 1	34,790 45,110 57,300	31 1 1	57,983 75,183 95,500	11 1 1
67	0,68 0,91 1	0,049 0,059 0,075	36 1 1	0,098 0,118 0,150	12 1 1	0,390 0,470 0,600	2 1 1	0,650 0,783 1,000	1 1 1
68	0,52 0,93 1	0,086 0,118 0,150	36 1 1	0,173 0,235 0,300	12 1 1	0,690 0,940 1,200	2 1 1	1,150 1,567 2,000	1 1 1
69	0,43 0,79 1	0,659 0,980 1,171	144 1 1	1,318 1,960 2,343	48 1 1	5,270 7,840 9,370	8 1 1	8,783 13,067 15,617	3 1 1
70	0,74 0,95 1	0,001 0,001 0,001	1 1 1	0,003 0,003 0,003	1 1 1	0,010 0,010 0,010	1 1 1	0,017 0,017 0,017	1 1 1
71	0,53 0,88 1	8,409 11,231 13,905	1044 1 1	16,818 22,463 27,810	348 1 1	67,270 89,850 111,240	58 1 1	112,117 149,750 185,400	21 1 1
72	0,67 0,92 1	0,001 0,001 0,001	1 1 1	0,003 0,003 0,003	1 1 1	0,010 0,010 0,010	1 1 1	0,017 0,017 0,017	1 1 1
73	0,58 0,91 1	1,183 1,936 3,089	252 1 1	2,365 3,873 6,178	84 1 1	9,460 15,490 24,710	14 1 1	15,767 25,817 41,183	5 1 1
74	0,64 0,95 1	0,104 0,138 0,165	36 1 1	0,208 0,275 0,330	12 1 1	0,830 1,100 1,320	2 1 1	1,383 1,833 2,200	1 1 1
75	0,44 0,84 1	8,513 11,039 15,660	1440 1 1	17,025 22,078 31,320	480 1 1	68,100 88,310 125,280	80 1 1	113,500 147,183 208,800	29 1 1
76	0,54 0,82 1	5,759 9,088 13,905	1044 1 1	11,518 18,175 27,810	348 1 1	46,070 72,700 111,240	58 1 1	76,783 121,167 185,400	21 1 1
77	0,66 0,93 1	0,075 0,115 0,165	36 1 1	0,150 0,230 0,330	12 1 1	0,600 0,920 1,320	2 1 1	1,000 1,533 2,200	1 1 1
78	0,64 0,88 1	8,875 13,050 15,660	1440 1 1	17,750 26,100 31,320	480 1 1	71,000 104,400 125,280	80 1 1	118,333 174,000 208,800	29 1 1
79	0,57 0,81 1	5,356 8,674 13,905	1044 1 1	10,713 17,348 27,810	348 1 1	42,850 69,390 111,240	58 1 1	71,417 115,650 185,400	21 1 1
80	0,67 0,92 1	115,206 159,456 198,319	20016 1 1	230,413 318,913 396,638	6672 1 1	921,650 1275,650 1586,550	1112 1 1	1536,083 2126,083 2644,250	397 1 1
81	0,62 0,93 1	6,954 10,759 13,905	1044 1 1	13,908 21,518 27,810	348 1 1	55,630 86,070 111,240	58 1 1	92,717 143,450 185,400	21 1 1
82	0,44 0,79 1	7,408 9,321 13,905	1044 1 1	14,815 18,643 27,810	348 1 1	59,260 74,570 111,240	58 1 1	98,767 124,283 185,400	21 1 1
83	0,49 0,8 1	7,720 10,178 15,660	1440 1 1	15,440 20,355 31,320	480 1 1	61,760 81,420 125,280	80 1 1	102,933 135,700 208,800	29 1 1
84	0,61 0,94 1	8,119 9,806 13,905	1044 1 1	16,238 19,613 27,810	348 1 1	64,950 78,450 111,240	58 1 1	108,250 130,750 185,400	21 1 1
85	0,7 0,94 1	9,894 13,190 15,660	1440 1 1	19,788 26,380 31,320	480 1 1	79,150 105,520 125,280	80 1 1	131,917 175,867 208,800	29 1 1
86	0,73 0,95 1	6,225 9,171 13,905	1044 1 1	12,450 18,343 27,810	348 1 1	49,800 73,370 111,240	58 1 1	83,000 122,283 185,400	21 1 1
87	0,41 0,68 1	53,549 69,896 83,298	7308 1 1	107,098 139,793 166,595	2436 1 1	428,390 559,170 666,380	406 1 1	713,983 931,950 1110,633	145 1 1
88	0,69 0,93 1	0,055 0,078 0,124	36 1 1	0,110 0,155 0,248	12 1 1	0,440 0,620 0,990	2 1 1	0,733 1,033 1,650	1 1 1
89	0,4 0,89 1	12,670 17,393 27,494	2070 1 1	25,340 34,785 54,988	690 1 1	101,360 139,140 219,950	115 1 1	168,933 231,900 366,583	41 1 1
90	0,54 0,82 1	15,791 24,526 37,128	2898 1 1	31,583 49,053 74,255	966 1 1	126,330 196,210 297,020	161 1 1	210,550 327,017 495,033	58 1 1

Tabela A1.10 (Continuação)

Tarefa	Prob	V4 F4		V3 F3		V2 F2		V1 F1	
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.	Tempo	Energ.
91	0,42	1,851	288	3,703	96	14,810	16	24,683	6
	0,78	2,949		5,898		23,590		39,317	
	1	3,533		7,065		28,260		47,100	
92	0,6	0,288	54	0,575	18	2,300	3	3,833	1
	0,87	0,480		0,960		3,840		6,400	
	1	0,730		1,460		5,840		9,733	
93	0,56	0,078	36	0,155	12	0,620	2	1,033	1
	0,85	0,094		0,188		0,750		1,250	
	1	0,150		0,300		1,200		2,000	
94	0,7	0,038	36	0,075	12	0,300	2	0,500	1
	0,94	0,054		0,108		0,430		0,717	
	1	0,075		0,150		0,600		1,000	
95	0,61	15,296	3060	30,593	1020	122,370	170	203,950	61
	0,83	23,356		46,713		186,850		311,417	
	1	38,843		77,685		310,740		517,900	
96	0,67	2,790	468	5,580	156	22,320	26	37,200	9
	0,95	3,513		7,025		28,100		46,833	
	1	5,791		11,583		46,330		77,217	
97	0,41	51,421	8046	102,843	2682	411,370	447	685,617	160
	0,84	62,074		124,148		496,590		827,650	
	1	100,568		201,135		804,540		1340,900	
98	0,54	90,375	11610	180,750	3870	723,000	645	1205,000	230
	0,78	129,189		258,378		1033,510		1722,517	
	1	155,191		310,383		1241,530		2069,217	
99	0,43	2,959	558	5,918	186	23,670	31	39,450	11
	0,86	4,466		8,933		35,730		59,550	
	1	7,163		14,325		57,300		95,500	
100	0,6	0,044	36	0,088	12	0,350	2	0,583	1
	0,93	0,060		0,120		0,480		0,800	
	1	0,075		0,150		0,600		1,000	
101	0,51	0,070	36	0,140	12	0,560	2	0,933	1
	0,77	0,110		0,220		0,880		1,467	
	1	0,150		0,300		1,200		2,000	
102	0,51	0,700	144	1,400	48	5,600	8	9,333	3
	0,91	0,994		1,988		7,950		13,250	
	1	1,171		2,343		9,370		15,617	
103	0,49	0,001	1	0,003	1	0,010	1	0,017	1
	0,78	0,001		0,003		0,010		0,017	
	1	0,001		0,003		0,010		0,017	
104	0,6	6,733	1044	13,465	348	53,860	58	89,767	21
	0,87	9,543		19,085		76,340		127,233	
	1	13,905		27,810		111,240		185,400	
105	0,64	0,001	1	0,003	1	0,010	1	0,017	1
	0,88	0,001		0,003		0,010		0,017	
	1	0,001		0,003		0,010		0,017	
106	0,54	1,835	252	3,670	84	14,680	14	24,467	5
	0,81	2,228		4,455		17,820		29,700	
	1	3,089		6,178		24,710		41,183	
107	0,52	7,739	1044	15,478	348	61,910	58	103,183	21
	0,93	11,636		23,273		93,090		155,150	
	1	13,905		27,810		111,240		185,400	
108	0,6	6,248	1044	12,495	348	49,980	58	83,300	21
	0,87	8,701		17,403		69,610		116,017	
	1	13,905		27,810		111,240		185,400	
109	0,44	8,101	1440	16,203	480	64,810	80	108,017	29
	0,88	12,848		25,695		102,780		171,300	
	1	15,660		31,320		125,280		208,800	
110	0,69	9,189	1044	18,378	348	73,510	58	122,517	21
	0,95	11,361		22,723		90,890		151,483	
	1	13,905		27,810		111,240		185,400	
111	0,45	7,623	1440	15,245	480	60,980	80	101,633	29
	0,73	10,860		21,720		86,880		144,800	
	1	15,660		31,320		125,280		208,800	
112	0,7	6,153	1044	12,305	348	49,220	58	82,033	21
	0,91	9,781		19,563		78,250		130,417	
	1	13,905		27,810		111,240		185,400	
113	0,66	49,339	7308	98,678	2436	394,710	406	657,850	145
	0,9	63,033		126,065		504,260		840,433	
	1	83,298		166,595		666,380		1110,633	

Com dados de entrada diferentes, os tempos de tratamento também acabam sendo diferentes para classes de dados de entrada de uma mesma tarefa. Novamente a ferramenta de aleatoriedade foi usada para geração dos dados de tempo de tratamento, sempre com base no tempo original para cada tarefa obtido do número de ciclos definido na Tabela A1.9. O valor para cada classe $c_{i,j}$ é aleatoriamente 15 a 40% menor que o tempo de tratamento de $c_{i,j+1}$.

Para levantamento dos dados de consumo de energia, foram usadas as quantidades de instruções de cada tarefa, apresentada por Gerstlauer *et al.* na coluna *instructions* da Tabela A1.9. Vale esclarecer que não foi feita nenhuma relação de consumo com a quantidade de instruções, apenas foram aproveitados dados que estavam disponíveis para as tarefas, tornando a geração de dados mais próxima à realidade do sistema vocoder.

Para a geração dos valores de consumo de energia, o nível de operação $V_2|F_2$ (100MHz) foi utilizado como base, e os valores de consumo de energia para cada tarefa neste nível foram definidos como o resultado da quantidade máxima de instruções da coluna *instructions* dividida por 100, truncando o resultado para um valor inteiro. Ainda explorando a relação quadrática entre consumo de energia e frequência de operação, a relação entre o consumo em cada nível segue o quadrado da relação dos processadores, resultando em $Energia_{800} = 3 \times Energia_{400} = 18 \times Energia_{100} = 51 \times Energia_{60}$.

Com a tabela de distribuição probabilística das classes de dados de entrada, tempos de execução e consumo de energia para cada tarefa, foram definidos os 2 sistemas nos quais as tarefas da aplicação foram mapeadas. Os sistemas contam com 1 e 2 processadores, e a distribuição das tarefas nos processadores seguem o mapeamento da Tabela A1.11. Cada coluna indica o sistema usado e em qual processador cada tarefa está mapeada.

Tabela A1.11: Mapeamento das tarefas nos sistemas experimentais

Tarefa	Vocoder-1	Vococer-2	Tarefa	Vocoder-1	Vococer-2	Tarefa	Vocoder-1	Vococer-2	Tarefa	Vocoder-1	Vococer-2
1	1	1	40	1	1	79	1	1	118	1	1
2	1	1	41	1	1	80	1	1	119	1	1
3	1	1	42	1	1	81	1	1	120	1	2
4	1	1	43	1	2	82	1	1	121	1	1
5	1	1	44	1	1	83	1	2	122	1	1
6	1	1	45	1	1	84	1	2	123	1	1
7	1	1	46	1	1	85	1	2	124	1	1
8	1	2	47	1	2	86	1	2	125	1	1
9	1	2	48	1	1	87	1	1	126	1	1
10	1	2	49	1	1	88	1	1	127	1	1
11	1	2	50	1	2	89	1	1	128	1	2
12	1	2	51	1	2	90	1	1	129	1	1
13	1	2	52	1	2	91	1	1	130	1	1
14	1	2	53	1	2	92	1	1	131	1	1
15	1	1	54	1	1	93	1	1	132	1	2
16	1	1	55	1	1	94	1	2	133	1	2
17	1	1	56	1	1	95	1	1			
18	1	1	57	1	1	96	1	1			
19	1	1	58	1	1	97	1	1			
20	1	1	59	1	1	98	1	1			
21	1	1	60	1	1	99	1	1			
22	1	1	61	1	2	100	1	1			
23	1	1	62	1	1	101	1	1			
24	1	2	63	1	1	102	1	2			
25	1	2	64	1	1	103	1	1			
26	1	2	65	1	1	104	1	1			
27	1	2	66	1	1	105	1	1			
28	1	1	67	1	1	106	1	2			
29	1	1	68	1	1	107	1	1			
30	1	1	69	1	2	108	1	1			
31	1	1	70	1	1	109	1	2			
32	1	1	71	1	1	110	1	2			
33	1	1	72	1	1	111	1	2			
34	1	1	73	1	2	112	1	2			
35	1	2	74	1	1	113	1	1			
36	1	1	75	1	1	114	1	1			
37	1	1	76	1	1	115	1	1			
38	1	1	77	1	1	116	1	1			
39	1	1	78	1	1	117	1	1			

Após mapeamento, o DFG original foi alterado e 2 novos DFGs foram obtidos. A Tabela A1.12 faz um resumo das quantidades de arestas e caminhos de execução para a aplicação nos dois sistemas, assim como os tempos de resposta estabelecidos.

Tabela A1.12: Dados da Aplicação Vocoder

Aplicação (nós, arestas)	Nº PEs	Tempo de resposta (ms)	Após Escalonamento	
			Nº Arestas	Nº Caminhos
<i>Vocoder</i> (133, 135)	1	20,0	132	1
	2		165	131072

A1.4 TGFF70

Em seu trabalho, Dick *et al.* [12] apresentam a ferramenta TGFF, que possibilita a geração de sistemas sintéticos, sendo que dois destes sistemas são usados neste trabalho. O primeiro sistema conta com 70 tarefas e sua estrutura é definida pelo DFG da Figura A1.4. A numeração gerada pelo TGFF se inicia com a tarefa zero, incompatível com o formato de entrada dos algoritmos desenvolvidos. Assim, todas as tarefas referenciadas nas tabelas desta aplicação estão incrementadas em uma unidade em relação ao DFG da Figura A1.4.

Além do grafo de tarefas, a ferramenta fornece uma lista das arestas de dependência entre as tarefas e dados específicos para cada tarefa que, no caso deste trabalho, foram interpretados como o tempo de execução de cada tarefa. No entanto, estes dados sofreram alteração. Os valores utilizados também foram incrementados em uma unidade pois, para algumas tarefas, o TGFF gerou parâmetro ZERO.

A geração dos dados é feita através de parâmetros informados à ferramenta TGFF. Para a geração dos dados da aplicação TGFF70, os parâmetros de geração apresentados na figura A1.5 foram usados no arquivo de entrada (extensão .tgffopt).

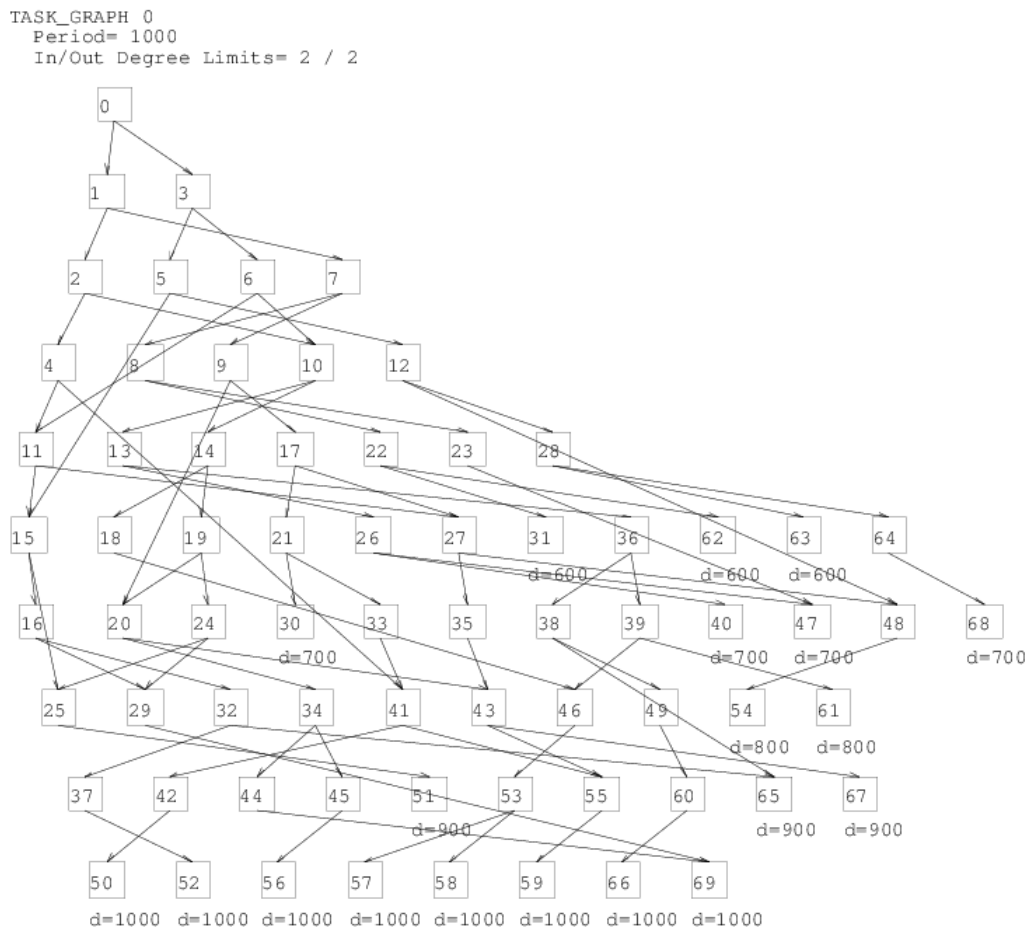


Figura A1.4: DFG do TGFF70 gerado pela ferramenta TGFF [34]


```
tg_cnt 1
task_cnt 71 1
start_node 20 1
task_degree 2 2
tg_write
eps_write
```

Figura A1.5: Parâmetros de entrada para geração da aplicação sintética TGFF70

No entanto, os dados fornecidos não são suficientes para utilização da aplicação nomeada TGFF70 neste trabalho, e outros dados de funcionamento do sistema foram criteriosamente criados a partir dos dados fornecidos pela ferramenta TGFF.

Partindo da tabela de dados fornecida pelo TGFF para cada tarefa, e aqui assumidos como tempos de execução das tarefas, foi definida uma nova tabela, com tempos de execução em três diferentes níveis de tensão/frequência, $V_1|F_1$, $V_2|F_2$ e $V_3|F_3$, além da variação de uma a três classes de dados de entrada para cada tarefa. Os dados gerados para cada tarefa estão descritos na Tabela A1.13.

O valor fornecido para cada tarefa foi definido como o tempo para tratamento da classe de dados de entrada $c_{i,1}$ ($i = 1 \dots 70$) no maior nível de operação $V_3|F_3$.

Os valores de probabilidade de descarte de classes de dados de entrada para cada tarefa foram gerados aleatoriamente, através da função ALEATÓRIO() do Microsoft ExcelTM. No entanto, esta aleatoriedade foi controlada, definindo valores coerentes (sem diferenças gritantes) para a probabilidade de ocorrência de cada classe em cada tarefa. As classes de dados de entrada $c_{i,1}$ ($i = 1 \dots 70$) foram definidas aleatoriamente dentro dos limites 0,40 a 0,75. As classes seguintes foram definidas com base no valor da classe anterior, $c_{i,j-1}$, com o critério de estarem aleatoriamente entre o valor de $c_{i,j-1} + 0,01$ e $c_{i,j-1} + 0,20$. O valor de c_{i,k_i} é sempre 1,00. Quando uma tarefa tem apenas uma classe de entrada, a tarefa sempre faz o tratamento de 100% dos dados de entrada ($c_{i,k_i} = 1$). Se a tarefa conta com apenas duas classes de dados de entrada, a aleatoriedade do valor de $c_{i,1}$ ($i = 1 \dots 70$), está definida entre 0,60 e 0,90.

Com dados de entrada diferentes, os tempos de tratamento também são diferentes para classes de dados de entrada de uma mesma tarefa. Novamente a ferramenta de aleatoriedade foi usada para geração dos dados de tempo de tratamento, sempre com base no tempo original definido pelo TGFF. O tempo de $c_{i,1}$ para o nível de operação $V_3|F_3$ foi a base de cálculo para todas as tarefas e o tempo de tratamento de cada classe $c_{i,j}$ é aleatoriamente 15 a 40% maior que o tempo de tratamento de $c_{i,j-1}$.

Definidas as probabilidades de distribuição das classes de entrada e os tempos de tratamento de cada classe para cada tarefa no nível máximo de operação $V_3|F_3$, são definidos os valores de tempo para os níveis $V_2|F_2$ e $V_1|F_1$. Sempre para a mesma classe de dados, a tarefa faz o tratamento dos dados no dobro do tempo quando operando no nível intermediário de operação $V_2|F_2$ e em quatro vezes o tempo quando operando no par tensão/frequência de menor velocidade e menor consumo $V_1|F_1$.

Tabela A1.13: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia

Tarefa	Prob	V3 F3		V2 F2		V1 F1		Tarefa	Prob	V3 F3		V2 F2		V1 F1	
		Tempo	Energia	Tempo	Energia	Tempo	Energia			Tempo	Energia	Tempo	Energia	Tempo	Energia
1	1	14	160	28	40	56	10	36	0,57	13	304	26	76	52	19
2	0,86	2	224	4	56	8	14	37	0,81	18		36		72	
	1	3		6		12		38	0,65	22	176	44	44	88	11
3	0,63	8	144	16	36	32	9	39	0,71	9	96	18	24	36	6
	0,87	12		24		48		40	0,92	17		34		68	
	1	15		30		60		41	0,68	3	48	6	12	12	3
4	1	10	144	20	36	40	9	42	0,41	5		10		20	
5	0,71	5	304	10	76	20	19	43	0,78	7		14		28	
	1	7		14		28		44	0,63	1	16	28	4	56	1
6	0,68	7	112	14	28	28	7	45	0,53	4	160	12	40	24	10
	0,89	9		18		36		46	0,84	8		16		32	
	1	12		24		48		47	0,78	1	16	32	32	64	8
7	1	6	32	12	8	24	2	48	0,71	3	224	6	56	12	14
8	0,8	4	32	8	8	16	2	49	0,92	4		8		16	
	1	5		10		20		50	0,61	6		12		24	
9	0,56	19	16	38	4	76	1	51	0,48	17	16	34	4	68	1
	0,88	25		50		100		52	0,87	13	128	26	32	52	8
	1	32		64		128		53	0,89	1	128	36	32	72	8
10	1	19	304	38	76	76	19	54	0,46	2		4	32	4	8
11	0,84	12	224	24	56	48	14	55	0,9	3		6		12	
	1	15		30		60		56	0,8	11	80	22	20	32	5
12	0,73	14	96	28	24	56	6	57	0,64	15		30		60	
	0,95	18		36		72		58	0,89	1	144	34	36	68	9
	1	22		44		88		59	0,85	1	240	44	44	88	
13	1	10	112	20	28	40	7	60	0,65	4		8	24	16	6
14	0,88	9	16	18	4	36	1	61	0,86	3	192	6	48	12	12
	1	13		26		52		62	0,71	4		8		16	
15	0,49	18	224	36	56	72	14	63	0,64	18	96	36	24	72	6
	0,76	24		48		96		64	0,8	17	144	34	36	68	9
	1	32		64		128		65	0,78	22		44		88	
16	1	12	48	24	12	48	3	66	0,64	19	48	38	12	76	3
17	0,72	12	192	24	48	48	12	67	0,89	24		48		96	
	1	17		34		68		68	0,8	29		58		116	
18	0,64	20	48	40	12	80	3	69	0,69	4	96	8	24	16	6
	0,86	27		54		108		70	0,71	3	192	6	48	12	12
	1	38		76		152			1	4		8		16	
19	1	15	80	30	20	60	5			1	80	16	20	32	5
20	0,76	13	128	26	32	52	8			1		22		44	
	1	18		36		72				1		30		60	
21	0,67	10	16	20	4	40	1			1	18	36	24	72	6
	0,94	13		26		52				1	17	34	36	68	9
	1	19		38		76				1	22	44	44	88	
22	1	16	272	32	68	64	17			1	19	38	12	76	3
23	0,88	7	176	14	44	28	11			1	19	48		96	
	1	9		18		36				1	29	58		116	
24	0,81	6	128	12	32	24	8			1	4	8	24	16	6
	0,94	8		16		32				1	3	6	48	12	12
	1	11		22		44				1	4	8		16	
25	1	3	224	6	56	12	14			1	6	12	60	24	15
26	0,79	16	48	32	12	64	3			1	7	14		28	
	1	21		42		84				1	9	18		36	
27	0,47	15	272	30	68	60	17			1	2	4	8	8	2
	0,8	18		36		72				1	7	14	8	28	2
	1	21		42		84				1	9	18		36	
28	1	17	256	34	64	68	16			1	9	18	16	36	4
29	0,6	18	304	36	76	72	19			1	18	36	60	72	15
	1	23		46		92				1	24	48		96	
30	0,67	4	112	8	28	16	7			1	30	60		120	
	0,82	6		12		24				1	9	18	16	36	4
	1	8		16		32				1	16	32	64	64	16
31	1	6	288	12	72	24	18			1	19	38		76	
32	0,68	15	240	30	60	60	15			1	19	38	24	76	6
	1	19		38		76				1	27	54		108	
33	0,6	7	240	14	60	28	15			1	37	74		148	
	0,82	10		20		40				1	15	30	12	60	3
	1	13		26		52				1	12	24	16	48	4
34	1	8	192	16	48	32	12			1	15	30		60	
35	0,71	19	176	38	44	76	11			1	1	2	44	4	11
	1	27		54		108				1	2	4		8	
										1	3	6		12	
										1	64	22	16	44	4

O TGFF fornece apenas um parâmetro para cada tarefa. Assim, foi necessário o levantamento de valores aleatórios de consumo de energia. O nível inferior $V_1|F_1$ foi usado como base para criação dos valores aleatórios de consumo de energia. Os valores de consumo para $V_2|F_2$ são quatro vezes maiores que $V_1|F_1$. Já para $V_3|F_3$, o consumo para tratamento em cada tarefa é quatro vezes maior que o consumo de $V_2|F_2$. A relação entre os valores de consumo de energia para cada nível foi baseada na relação quadrática entre o nível de operação (tempo de execução) e o consumo de energia, abordado no Capítulo 2. Aumentando o nível de operação, o tempo de execução cai pela metade, enquanto o consumo de energia aumenta em 4 vezes.

Com a tabela de distribuição probabilística das classes de dados de entrada, tempos de execução e consumo de energia para cada tarefa, foram definidos os 4 sistemas nos quais as tarefas do sistema foram mapeadas. Os sistemas contam com 1, 2, 4 e 8 processadores, e a distribuição das tarefas nos processadores seguem o mapeamento da Tabela A1.14. Cada coluna indica o sistema usado e em qual processador cada tarefa está mapeada.

Tabela A1.14: Mapeamento das tarefas nos sistemas experimentais

Tarefa	TGFF70-1	TGFF70-2	TGFF70-5	TGFF70-8	Tarefa	TGFF70-1	TGFF70-2	TGFF70-5	TGFF70-8
1	1	1	3	8	36	1	2	4	2
2	1	2	1	5	37	1	1	2	7
3	1	1	1	4	38	1	2	4	2
4	1	2	4	1	39	1	1	4	2
5	1	1	4	4	40	1	2	1	3
6	1	2	2	8	41	1	1	3	7
7	1	1	4	6	42	1	2	2	8
8	1	2	3	5	43	1	1	4	6
9	1	1	3	5	44	1	2	1	6
10	1	2	3	8	45	1	1	3	8
11	1	1	2	1	46	1	2	4	6
12	1	2	4	3	47	1	1	2	2
13	1	1	1	8	48	1	2	4	6
14	1	2	3	5	49	1	1	3	1
15	1	1	2	1	50	1	2	4	5
16	1	2	2	3	51	1	1	4	6
17	1	1	2	4	52	1	2	1	8
18	1	2	4	4	53	1	1	3	4
19	1	1	1	2	54	1	2	1	4
20	1	2	4	6	55	1	1	4	3
21	1	1	4	6	56	1	2	1	7
22	1	2	3	3	57	1	1	3	8
23	1	1	2	7	58	1	2	3	5
24	1	2	4	3	59	1	1	4	5
25	1	1	1	1	60	1	2	2	5
26	1	2	2	4	61	1	1	4	5
27	1	1	1	5	62	1	2	2	3
28	1	2	3	4	63	1	1	4	7
29	1	1	4	4	64	1	2	3	1
30	1	2	2	4	65	1	1	2	6
31	1	1	3	7	66	1	2	1	6
32	1	2	1	4	67	1	1	1	4
33	1	1	4	7	68	1	2	2	4
34	1	2	3	8	69	1	1	3	8
35	1	1	1	2	70	1	2	4	7

Após mapeamento, o DFG original foi alterado e 4 novos DFGs foram obtidos. A Tabela A1.15 faz um resumo das quantidades de arestas e caminhos de execução para a aplicação em cada um dos quatro sistemas, assim como os tempos de resposta estabelecidos, citados anteriormente.

Tabela A1.15: Dados da Aplicação TGFF70

Aplicação (nós, arestas)	Nº PEs	Tempo de resposta (ms)	Após Escalonamento	
			Nº Arestas	Nº Caminhos
TGFF70 (70, 84)	1	3,0	69	1
	2	1,5	109	693
	4	1,0	129	1944
	8	0,6	129	573

A1.5 TGFF99

Novamente a ferramenta TGFF foi usada, desta vez para criar uma aplicação com 99 tarefas, que tem sua estrutura de funcionamento definida pelo DFG da Figura A1.6. Como ocorreu para o TGFF70, todas as tarefas referenciadas nas tabelas desta aplicação estão incrementadas em uma unidade em relação ao DFG da Figura A1.6, assim como os valores de tempo também foram incrementados em uma unidade pois, novamente, o TGFF gerou parâmetro ZERO para algumas tarefas.

A geração dos dados é feita através de parâmetros informados à ferramenta TGFF. Para a geração dos dados da aplicação TGFF99, os parâmetros de geração apresentados na Figura A1.7 foram usados no arquivo de entrada (extensão .tgffopt).

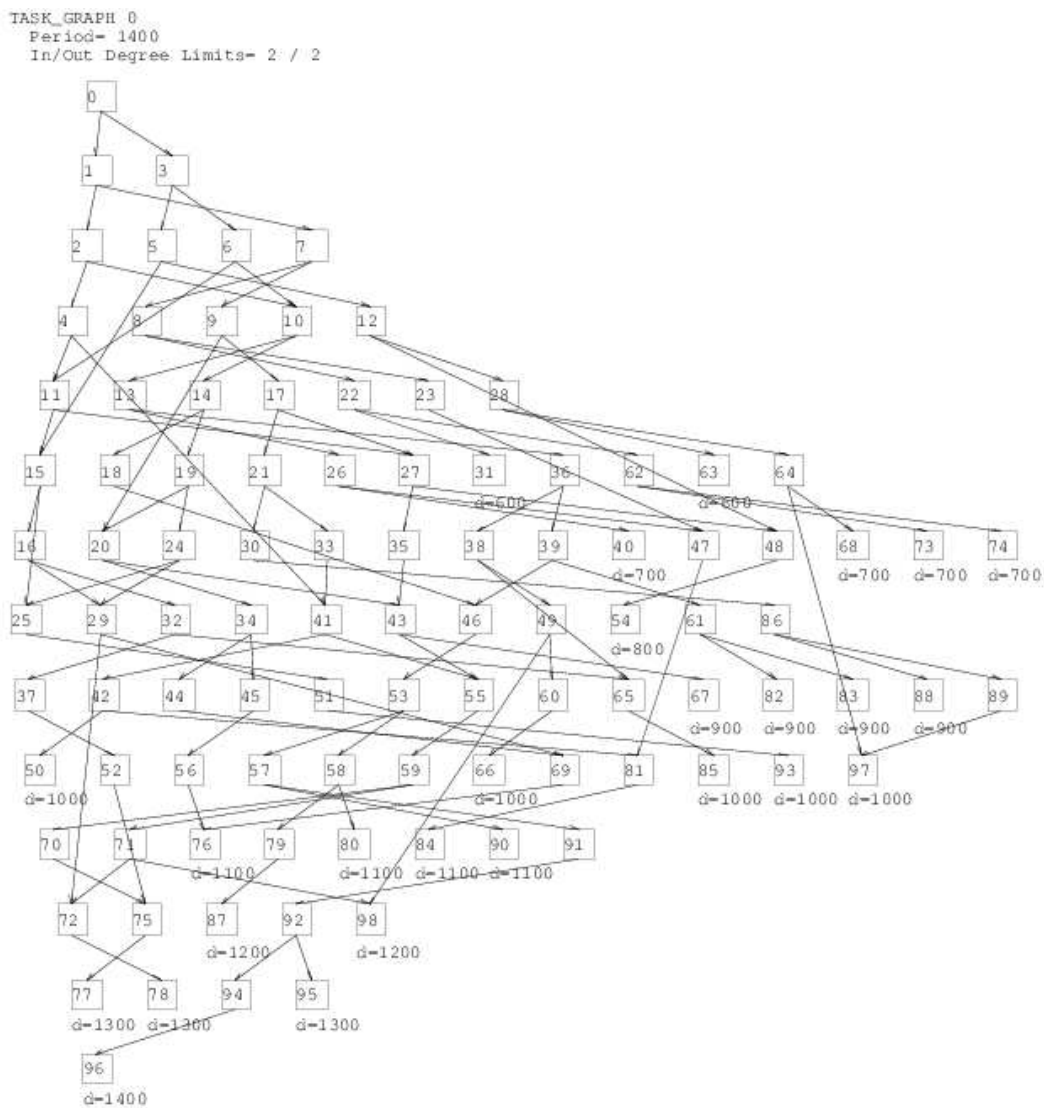


Figura A1.6: DFG do TGFF99 gerado pela ferramenta TGFF [34]

```
tg_cnt 1
task_cnt 100 1
start_node 20 1
task_degree 2 2
tg_write
eps_write
```

Figura A1.7: Parâmetros de entrada para geração da aplicação sintética TGFF99

No entanto, os dados fornecidos não são suficientes para utilização da aplicação TGFF99 neste trabalho, e outros dados de funcionamento do sistema foram criteriosamente criados a partir dos dados fornecidos pela ferramenta TGFF.

Partindo da tabela de dados fornecida pelo TGFF para cada tarefa, e aqui assumidos como tempos de execução das tarefas, foi definida uma nova tabela, com tempos de execução em três diferentes níveis de tensão/frequência, $V_1|F_1$, $V_2|F_2$ e $V_3|F_3$, além da variação de uma a cinco classes de dados de entrada para cada tarefa. Os dados gerados para cada tarefa estão descritos na Tabela A1.16 (dividida em 6 conjuntos de tarefas).

O valor fornecido para cada tarefa foi definido como o tempo para tratamento da classe de dados de entrada $c_{i,1}$ ($i = 1 \dots 99$) no maior nível de operação $V_3|F_3$.

Os valores de probabilidade de descarte de classes de dados de entrada para cada tarefa foram gerados aleatoriamente, através da função ALEATÓRIO() do Microsoft ExcelTM. No entanto, esta aleatoriedade foi controlada, definindo valores coerentes (sem diferenças gritantes) para a probabilidade de ocorrência de cada classe em cada tarefa. As classes de dados de entrada $c_{i,1}$ ($i = 1 \dots 99$) foram definidas aleatoriamente dentro dos limites 0,40 a 0,75. As classes seguintes foram definidas com base no valor da classe anterior, $c_{i,j-1}$, com o critério de estarem aleatoriamente entre o valor de $c_{i,j-1} + 0,01$ e $c_{i,j-1} + 0,20$. O valor de c_{i,k_i} é sempre 1,00. Quando uma tarefa tem apenas uma classe de entrada, a tarefa sempre faz o tratamento de 100% dos dados de entrada ($c_{i,k_i} = 1$). Se a tarefa conta com apenas duas classes de dados de entrada, a aleatoriedade do valor de $c_{i,1}$ ($i = 1 \dots 99$), está definida entre 0,60 e 0,90.

Com dados de entrada diferentes, os tempos de tratamento são diferentes para classes de dados de entrada de uma mesma tarefa. Novamente a ferramenta de aleatoriedade foi usada para geração dos dados de tempo de tratamento, sempre com base no tempo original definido pelo TGFF. O tempo de $c_{i,1}$ para o nível de operação $V_3|F_3$ foi a base de cálculo para todas as tarefas e o valor para cada classe $c_{i,j}$ é aleatoriamente 15 a 40% maior que o tempo de tratamento de $c_{i,j-1}$.

Definidas as probabilidades de distribuição das classes de entrada e os tempos de tratamento de cada classe de dados de entrada para cada tarefa no nível máximo de operação $V_3|F_3$, são definidos os valores de tempo para os níveis $V_2|F_2$ e $V_1|F_1$. Sempre para a mesma classe de dados, a tarefa faz o tratamento dos dados no dobro do tempo quando operando no nível intermediário de operação $V_2|F_2$ e quatro vezes o tempo quando operando no par tensão/frequência de menor velocidade e menor consumo.

Tabela A1.16: Distribuição probabilística de classes de entrada, tempos de execução e consumo de energia

Tarefas	Prob	V3 F3		V2 F2		V1 F1	
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.
1	1	18	288	36	72	72	18
2	0,9	2	208	4	52	8	13
	1	3		6		12	
3	0,7	11	128	22	32	44	8
	0,9	15		30		60	
	1	20		40		80	
4	0,6	10	288	20	72	40	18
	0,9	13		26		52	
	1	17		34		68	
	1	22		44		88	
5	0,4	13	80	26	20	52	5
	0,6	17		34		68	
	0,8	21		42		84	
	0,9	25		50		100	
	1	30		60		120	
6	1	19	272	38	68	76	17
7	0,9	19	96	38	24	76	6
	1	23		46		92	
8	0,7	12	32	24	8	48	2
	1	16		32		64	
	1	21		42		84	
9	0,5	9	32	18	8	36	2
	0,8	12		24		48	
	1	17		34		68	
	1	23		46		92	
10	0,6	10	272	20	68	40	17
	0,8	12		24		48	
	0,9	15		30		60	
	1	21		42		84	
	1	29		58		116	
11	1	2	48	4	12	8	3
12	0,9	6	96	12	24	24	6
	1	8		16		32	
13	0,6	15	64	30	16	60	4
	0,8	21		42		84	
	1	29		58		116	
14	0,6	17	192	34	48	68	12
	0,9	21		42		84	
	1	28		56		112	
	1	35		70		140	
15	0,6	9	224	18	56	36	14
	0,9	11		22		44	
	1	15		30		60	
	1	20		40		80	
	1	26		52		104	
16	1	5	288	10	72	20	18
17	0,9	3	96	6	24	12	6
	1	4		8		16	
18	0,6	4	112	8	28	16	7
	0,9	6		12		24	
	1	8		16		32	
19	0,6	10	208	20	52	40	13
	0,9	13		26		52	
	1	16		32		64	
	1	23		46		92	
20	0,6	4	176	8	44	16	11
	0,7	5		10		20	
	0,9	7		14		28	
	0,9	9		18		36	
	1	11		22		44	
21	1	19	288	38	72	76	18
22	0,7	6	144	12	36	24	9
	1	7		14		28	
23	0,6	7	64	14	16	28	4
	0,9	9		18		36	
	1	12		24		48	
24	0,6	3	96	6	24	12	6
	0,8	4		8		16	
	0,9	5		10		20	
	1	7		14		28	
25	0,4	6	304	12	76	24	19
	0,7	8		16		32	
	0,9	10		20		40	
	0,9	14		28		56	
	1	18		36		72	
26	1	7	160	14	40	28	10
27	0,8	11	16	22	4	44	1
	1	14		28		56	
28	0,7	14	64	28	16	56	4
	0,9	18		36		72	
	1	25		50		100	
29	0,5	10	192	20	48	40	12
	0,8	14		28		56	
	0,9	18		36		72	
	1	25		50		100	
30	0,7	8	224	16	56	32	14
	0,9	11		22		44	
	1	15		30		60	
	1	18		36		72	
	1	24		48		96	
31	1	4	176	8	44	16	11
32	0,7	11	176	22	44	44	11
	1	15		30		60	
33	0,6	3	48	6	12	12	3
	0,9	4		8		16	
	1	6		12		24	

Tabela A1.16 (Continuação)

Tarefas	Prob	V3 F3		V2 F2		V1 F1		Tarefas	Prob	V3 F3		V2 F2		V1 F1		
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.			Tempo	Energ.	Tempo	Energ.	Tempo	Energ.	
34	0,7	4	144	8	36	16	9	50	0,7	13	112	26	28	52	7	
	0,9	6		12		24			0,8	17		34		68		
	1	8		16		32			0,9	22		44		88		
	1	11		22		44			1	29		58		116		
35	0,6	8	272	16	68	32	17	51	1	11	16	22	4	44	1	
	0,7	10		20		40			52	0,8	3	48	6	12	12	3
	0,9	14		28		56				1	4		8		16	
	1	18		36		72				53	0,7	10	16	20	4	40
1	25		50		100		0,9	13			26		52			
36	1	12	128	24	32	48	8	1	18			36		72		
	37	0,8	7	304	14	76	28	19	54		0,6	6	160	12	40	24
		1	9		18		36			0,8	8		16		32	
		38	0,5	6	224	12	56	24		14	0,9	11		22		44
0,7	9			18		36		1		16		32		64		
39	0,7	13	304	26	76	52	19	55	0,7	7	304	14	76	28	19	
	0,9	17		34		68			0,9	9		18		36		
	1	21		42		84			0,9	12		24		48		
	1	25		50		100			1	16		32		64		
40	0,4	1	128	2	32	4	8	56	1	4	240	8	60	16	15	
	0,8	2		4		8			57	0,9	17	144	34	36	68	9
	0,9	3		6		12				1	20		40		80	
	1	5		10		20				58	0,5	12	96	24	24	48
1	7		14		28		0,7	16			32		64			
41	1	13	304	26	76	52	19	1	20			40		80		
	42	0,8	17	48	34	12	68	3	59		0,7	7	32	14	8	28
		1	22		44		88			1	9		18		36	
		43	0,5	6	192	12	48	24		12	1	13		26		52
0,9	8			16		32		1		18		36		72		
44	0,7	3	176	6	44	12	11	60	0,5	16	256	32	64	64	16	
	0,9	5		10		20			0,8	20		40		80		
	1	6		12		24			1	24		48		96		
	1	9		18		36			1	30		60		120		
45	0,4	17	16	34	4	68	1	61	1	8	208	16	52	32	13	
	0,7	21		42		84			62	0,7	6	96	12	24	24	6
	0,8	28		56		112				1	9		18		36	
	1	38		76		152				63	0,5	7	96	14	24	28
1	50		100		200		0,8	9			18		36			
46	1	16	96	32	24	64	6	1	11			22		44		
	47	0,8	15	96	30	24	60	6	64		0,7	4	80	8	20	16
1		20		40		80		0,9		5		10		20		
48		0,7	5	160	10	40	20	10		1	7		14		28	
	0,9	6		12		24		1		10		20		40		
49	0,5	14	96	28	24	56	6	65	0,7	3	288	6	72	12	18	
	0,8	20		40		80			0,8	4		8		16		
	0,9	26		52		104			0,9	5		10		20		
	1	31		62		124			1	7		14		28		
								1	9		18		36			

Tabela A1.16 (Continuação)

Tarefas	Prob	V3 F3		V2 F2		V1 F1	
		Tempo	Energ.	Tempo	Energ.	Tempo	Energ.
66	1	3	192	6	48	12	12
67	0,8	11	64	22	16	44	4
	1	14		28		56	
68	0,6	1	224	2	56	4	14
	0,8	2		4		8	
	1	3		6		12	
69	0,7	13	16	26	4	52	1
	0,9	18		36		72	
	1	22		44		88	
	1	26		52		104	
70	0,6	1	32	2	8	4	2
	0,9	2		4		8	
	0,9	3		6		12	
	1	4		8		16	
	1	5		10		20	
71	1	7	256	14	64	28	16
72	0,8	17	304	34	76	68	19
	1	21		42		84	
73	0,7	11	240	22	60	44	15
	1	16		32		64	
	1	19		38		76	
74	0,7	1	224	2	56	4	14
	1	2		4		8	
	1	3		6		12	
	1	4		8		16	
75	0,5	8	240	16	60	32	15
	0,7	12		24		48	
	0,9	15		30		60	
	1	20		40		80	
	1	28		56		112	
76	1	20	256	40	64	80	16
77	0,9	2	160	4	40	8	10
	1	3		6		12	
78	0,5	1	96	2	24	4	6
	0,8	2		4		8	
	1	3		6		12	
79	0,7	1	128	2	32	4	8
	0,9	2		4		8	
	1	3		6		12	
	1	5		10		20	
80	0,5	10	288	20	72	40	18
	0,8	13		26		52	
	0,9	17		34		68	
	1	22		44		88	
	1	28		56		112	
81	1	13	304	26	76	52	19
82	0,8	13	224	26	56	52	14
	1	18		36		72	
83	0,7	11	64	22	16	44	4
	0,9	14		28		56	
	1	19		38		76	
84	0,6	2	272	4	68	8	17
	0,8	3		6		12	
	0,9	4		8		16	
	1	6		12		24	
85	0,5	12	288	24	72	48	18
	0,6	15		30		60	
	0,9	21		42		84	
	1	28		56		112	
	1	36		72		144	
86	1	3	16	6	4	12	1
87	0,9	11	48	22	12	44	3
	1	15		30		60	
88	0,6	11	224	22	56	44	14
	0,9	14		28		56	
	1	17		34		68	
89	0,6	17	160	34	40	68	10
	0,9	23		46		92	
	1	27		54		108	
	1	32		64		128	
90	0,7	2	192	4	48	8	12
	0,9	3		6		12	
	1	5		10		20	
	1	7		14		28	
	1	9		18		36	
91	1	15	64	30	16	60	4
92	0,8	11	64	22	16	44	4
	1	15		30		60	
93	0,7	15	160	30	40	60	10
	0,9	20		40		80	
	1	25		50		100	
94	0,5	6	208	12	52	24	13
	0,9	8		16		32	
	1	12		24		48	
	1	15		30		60	
95	0,5	10	304	20	76	40	19
	0,9	13		26		52	
	1	17		34		68	
	1	23		46		92	
	1	28		56		112	
96	1	20	48	40	12	80	3
97	0,9	13	176	26	44	52	11
	1	17		34		68	
98	0,5	17	256	34	64	68	16
	0,8	21		42		84	
	1	28		56		112	
99	0,7	10	64	20	16	40	4
	0,9	13		26		52	
	1	16		32		64	
	1	22		44		88	

O TGFF fornece apenas um parâmetro para cada tarefa. Assim, foi necessário o levantamento de valores aleatórios de consumo de energia para cada tarefa. O nível inferior $V_1|F_1$ foi usado como base para criação dos valores aleatórios de consumo de energia. Os valores de consumo para $V_2|F_2$ são quatro vezes maiores que $V_1|F_1$. Já para $V_3|F_3$, o consumo para tratamento em cada tarefa é quatro vezes maior que o consumo de $V_2|F_2$. A relação entre os valores de consumo de energia para cada nível foi baseada na relação quadrática entre o nível de operação (tempo de execução) e o consumo de energia, abordado no Capítulo 2. Aumentando o nível de operação, o tempo de execução cai pela metade, enquanto o consumo de energia aumenta em 4 vezes.

Com a tabela de distribuição probabilística das classes de dados de entrada, tempos de execução e consumo de energia para cada tarefa, foram definidos os 4 sistemas nos quais as tarefas do sistema foram mapeadas. Os sistemas contam com 1, 2, 5 e 8 processadores, e a distribuição das tarefas nos processadores seguem o mapeamento da Tabela A1.17. Cada coluna indica o sistema usado e em qual processador cada tarefa está mapeada.

Tabela A1.17: Mapeamento das tarefas nos sistemas experimentais

Tarefa	TGFF99-1	TGFF99-2	TGFF99-5	TGFF99-8	Tarefa	TGFF99-1	TGFF99-2	TGFF99-5	TGFF99-8	Tarefa	TGFF99-1	TGFF99-2	TGFF99-5	TGFF99-8
1	1	1	1	6	34	1	2	4	2	67	1	1	2	1
2	1	2	1	4	35	1	1	2	2	68	1	2	5	5
3	1	1	1	6	36	1	2	2	6	69	1	1	1	5
4	1	2	2	5	37	1	1	5	3	70	1	2	3	8
5	1	1	1	3	38	1	2	1	6	71	1	1	1	6
6	1	2	2	1	39	1	1	3	2	72	1	2	3	7
7	1	1	3	4	40	1	2	5	2	73	1	1	2	5
8	1	2	4	4	41	1	1	3	8	74	1	2	2	8
9	1	1	4	6	42	1	2	3	1	75	1	1	3	2
10	1	2	5	2	43	1	1	3	8	76	1	2	1	8
11	1	1	3	1	44	1	2	4	1	77	1	1	4	8
12	1	2	1	1	45	1	1	2	4	78	1	2	1	8
13	1	1	2	1	46	1	2	4	5	79	1	1	3	2
14	1	2	3	2	47	1	1	5	1	80	1	2	2	1
15	1	1	2	8	48	1	2	4	4	81	1	1	5	5
16	1	2	1	8	49	1	1	1	7	82	1	2	4	2
17	1	1	1	2	50	1	2	1	2	83	1	1	1	7
18	1	2	5	6	51	1	1	1	7	84	1	2	2	1
19	1	1	2	2	52	1	2	5	5	85	1	1	2	7
20	1	2	1	4	53	1	1	1	1	86	1	2	5	2
21	1	1	2	7	54	1	2	1	3	87	1	1	4	8
22	1	2	5	3	55	1	1	2	2	88	1	2	3	4
23	1	1	4	8	56	1	2	3	7	89	1	1	3	8
24	1	2	4	6	57	1	1	3	7	90	1	2	4	4
25	1	1	4	3	58	1	2	4	6	91	1	1	3	6
26	1	2	4	4	59	1	1	5	2	92	1	2	1	7
27	1	1	3	5	60	1	2	1	8	93	1	1	4	1
28	1	2	2	4	61	1	1	2	5	94	1	2	1	3
29	1	1	2	1	62	1	2	2	8	95	1	1	2	1
30	1	2	5	3	63	1	1	3	1	96	1	2	5	5
31	1	1	5	6	64	1	2	4	3	97	1	1	5	5
32	1	2	4	5	65	1	1	5	8	98	1	2	2	8
33	1	1	1	2	66	1	2	4	5	99	1	1	5	6

Após mapeamento, o DFG original foi alterado e 4 novos DFGs foram obtidos. A Tabela A1.18 faz um resumo das quantidades de arestas e caminhos de execução para a aplicação em cada um dos quatro sistemas, assim como os tempos de resposta estabelecidos.

Tabela A1.18: Dados da Aplicação TGFF90

Aplicação (nós, arestas)	Nº PEs	Tempo de resposta (ms)	Após Escalonamento	
			Nº Arestas	Nº Caminhos
TGFF99 (99, 100)	1	5,0	98	1
	2	2,5	156	6039
	5	1,3	170	3586
	8	1,1	197	10474

Apêndice A2: Ferramenta de Teste dos Algoritmos

Neste apêndice está descrito o funcionamento dos códigos do algoritmo e da ferramenta utilizada para levantamento dos resultados dos experimentos. Embora o código dos algoritmos e da ferramenta não tenha grandes dimensões, a quantidade de páginas para sua reprodução aqui é proibitiva. Assim, apenas algumas características do código e da ferramenta serão apresentadas. No entanto, para utilizar o código em outros trabalhos, sempre com a devida referência de origem, envie um email para pedropepe@gmail.com, e ficaremos satisfeitos em fornecer qualquer informação necessária para reprodução dos resultados apresentados neste trabalho.

Como superficialmente descrito no Capítulo 6, os algoritmos foram desenvolvidos em linguagem C, e fazem uso de duas estruturas de código separadas, sendo uma delas desenvolvida através do Software Dev-C++ [10], usando o GNU como compilador e outra desenvolvida utilizando a ferramenta NI LabWindows, da National Instruments [28]. Basicamente, a primeira estrutura faz a execução do código de *SCN3* e a segunda a execução do código de *SCN4*. Os próximos itens descrevem mais detalhadamente as estruturas e ferramentas desenvolvidas para chegar aos resultados apresentados neste trabalho.

A2.1 Entrada de Dados

Inicialmente alguns arquivos foram gerados para obtenção dos resultados aqui apresentados, considerando cada variação das aplicações usadas. E neste processo, a participação do aluno de graduação Leandro D. Pagotto foi de fundamental importância nos resultados deste trabalho. Em seu trabalho de Iniciação Científica, Leandro desenvolveu um algoritmo de geração de caminhos de dependência dentro de um DFG, partindo dos dados de dependência entre as tarefas da aplicação.

Em um primeiro momento, através de um código específico desenvolvido como parte deste trabalho, é feito o escalonamento das tarefas da aplicação original nas diversas configurações de processadores para o sistema, alterando o DFG original das aplicações e as dependências entre as tarefas. Este mesmo código gera um arquivo que descreve as dependências entre as tarefas para cada novo sistema. Utilizando este arquivo de dependências, o algoritmo desenvolvido pelo Leandro lista todos os caminhos de execução da aplicação, de acordo com o novo DFG do sistema.

Este arquivo texto, com os caminhos para cada quantidade de processadores em cada aplicação, é usado como entrada para o código do algoritmo *SCN4*.

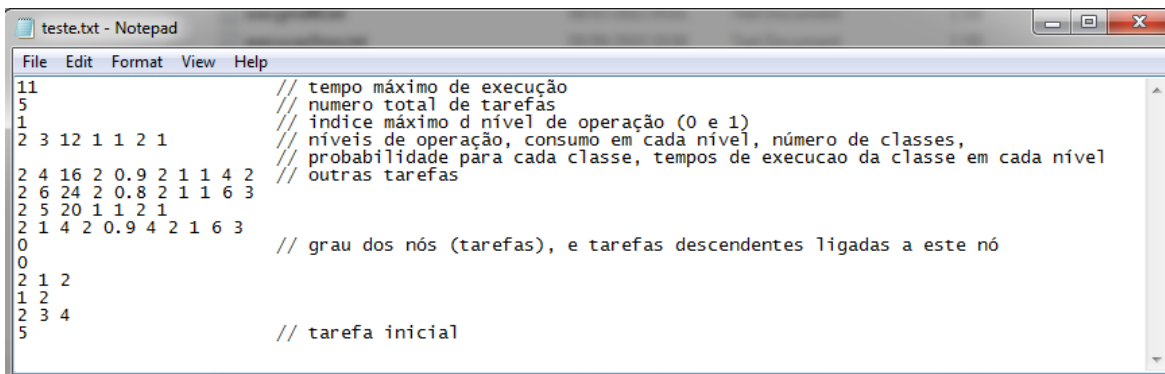
A2.2 Execução SCN1, SCN2 e SCN3

Como descrito no Capítulo 6, são necessários dois arquivos de entrada para execução do algoritmo *SCN3* em cada variação das aplicações. Um destes arquivos contém todas as informações das tabelas descritas no Apêndice A1, além dos dados de dependência entre as tarefas para o novo mapeamento nos processadores, devidamente convertidas em um arquivo texto, como exemplificado na Figura A2.1.

O segundo arquivo de entrada contém apenas os dados para geração automática de resultados, informando quantos são os processadores do sistema e quantos experimentos serão realizados para a aplicação, fazendo a combinação de valores apresentados para qualidade mínima de serviço e tempo máximo de execução. Um exemplo dos dados presentes neste arquivo segue na Figura A2.2. A primeira linha informa a quantidade de processadores do sistema, que deve ser a mesma da aplicação representada pelos dados do outro arquivo de entrada. A segunda linha mostra as *QoS* requeridas, neste caso, 5. A terceira linha mostra os tempos máximos de execução para a aplicação, sendo dois no caso deste experimento. A combinação dos dados gera 10 experimentos para a aplicação em questão.

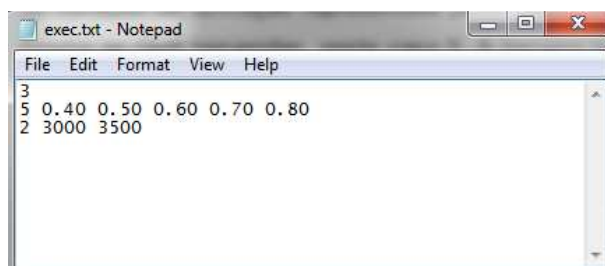
Com os arquivos montados para cada variação das aplicações (diferentes quantidades de processadores), o executável gerado é chamado via linha de comando, especificando a aplicação e os dados que serão usados na execução do experimento, gerando automaticamente o resultado para todas as combinações definidas no arquivo de entrada:

```
..\..\> main arquivo_aplicacao.txt entrada_experimento.txt
```



```
11 // tempo máximo de execução
5 // numero total de tarefas
1 // índice máximo d nível de operação (0 e 1)
2 3 12 1 1 2 1 // níveis de operação, consumo em cada nível, número de classes,
// probabilidade para cada classe, tempos de execucao da classe em cada nível
2 4 16 2 0.9 2 1 1 4 2 // outras tarefas
2 6 24 2 0.8 2 1 1 6 3
2 5 20 1 1 2 1
2 1 4 2 0.9 4 2 1 6 3
0 // grau dos nós (tarefas), e tarefas descendentes ligadas a este nó
0
2 1 2
1 2
2 3 4
5 // tarefa inicial
```

Figura A2.1: Exemplo de arquivo de dados de entrada para *SCN3* - DFG com 5 tarefas



```
3
5 0.40 0.50 0.60 0.70 0.80
2 3000 3500
```

Figura A2.2: Arquivo de dados de experimentação para os algoritmos *offline*

Tanto para os algoritmos *offline* quanto para o algoritmo *online*, a estruturação do código segue uma máquina de estados, executada dentro de um loop de código onde está colocada a rotina “main” de execução. Cada estado fica em execução até que suas condições de saída sejam satisfeitas. Embora esta estrutura não seja necessária para *SCN3*, já que a execução do código é sequencial, a estrutura de divisão em máquinas de estado foi mantida para facilitar a organização do código. A Figura A2.3 mostra um diagrama de blocos da estrutura do programa.

Inicialmente, o código copia os dados dos arquivos de entrada para as tabelas de trabalho, separando os dados de cada classe de entrada (tempo para tratamento e consumo de energia) para todas as tarefas, além dos dados de dependência entre as tarefas para a aplicação sobre a qual o algoritmo vai atuar. Embora seja necessário ao algoritmo conhecer os caminhos de execução da aplicação, este arquivo não é uma entrada para o código desenvolvido. Isso se deve à inclusão, neste código de algoritmos *offline*, do algoritmo de verificação dos caminhos desenvolvido pelo Leandro, que trabalha diretamente com a lista de dependência entre as tarefas.

Além da verificação dos caminhos, o código verifica os dados da experimentação a ser executada, verificando quantas qualidades mínimas de serviço e quantos limites de tempo de execução serão combinados para geração de resultados.

Em seguida, o algoritmo é executado. Como descrito no item 5.2, é encontrado o menor nível de operação, comum a todos os processadores, que proporcione ao sistema executar a aplicação dentro da restrição de tempo. A utilização dos processadores e o consumo de energia são calculados para a configuração encontrada.

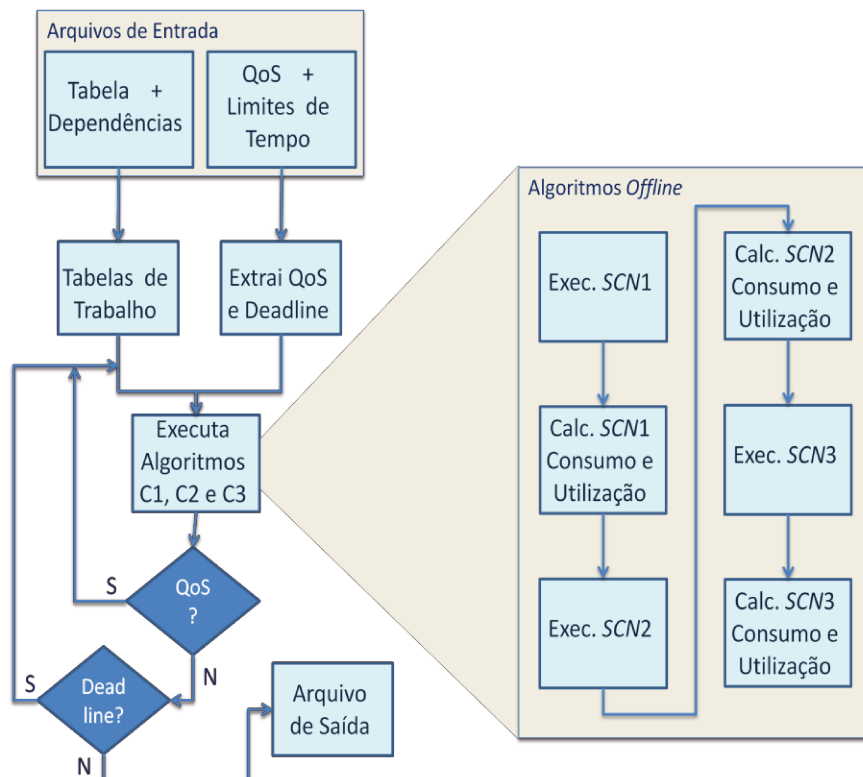


Figura A2.3: Diagrama do fluxo de execução dos algoritmos *offline*

Na sequência é executado o algoritmo *SCN2*, após o qual novamente são calculados os valores de utilização dos processadores e consumo de energia para a nova configuração.

Seguindo a sequência descrita na Figura A2.3, a execução do algoritmo *SCN3* encontra a configuração para funcionamento da aplicação. Novamente são feitos os cálculos de utilização dos processadores e consumo de energia, desta vez comparando-os aos resultados obtidos para os cenários C1 e C2. Todos os resultados de execução dos algoritmos estão agora disponíveis para montagem do arquivo de saída.

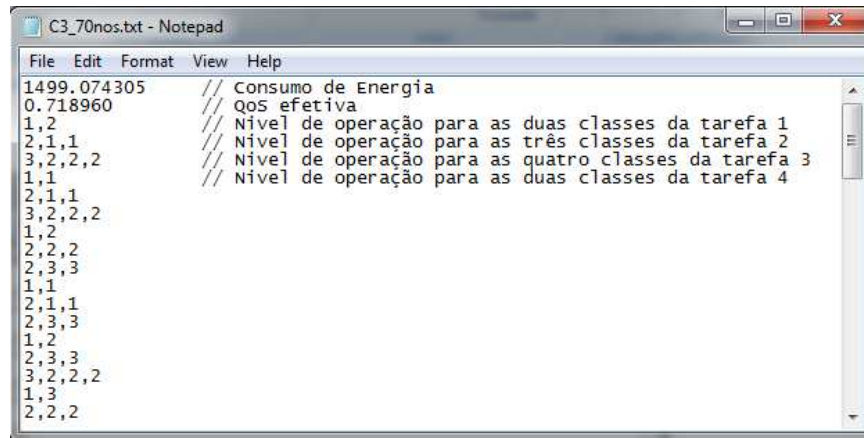
O arquivo de saída pode ser configurado em vários formatos, considerando que o código gera um arquivo texto com os resultados da execução do algoritmo. Durante todo o levantamento de resultados dos experimentos, este arquivo assumiu muitas formas, desde apenas dados de energia e utilização para cada cenário, até qualidade final de serviço e dados de nível de operação de cada classe em cada tarefa da aplicação. Um exemplo de um arquivo de saída é mostrado na Figura A2.4.

Em uma de suas diversas formas, este arquivo de saída é usado como tabela de configuração a ser consultada pela aplicação antes do início de sua execução. Em outro formato, este arquivo é usado como entrada para o código do algoritmo de *SCN4*, desenvolvido com o uso do NI LabWindows. Quando utilizado como entrada para o código de *SCN4*, este arquivo contém os valores de consumo de energia e qualidade efetiva de serviço, além dos níveis de operação de cada classe de entrada, como descrito na Figura A2.5.

O cálculo do consumo de energia final da configuração encontrada por *SCN3*, assim como a taxa de utilização dos processadores, não faz parte do escopo do algoritmo, mas seus valores são levantados para que se tenha um parâmetro de avaliação do algoritmo apresentado.

QoS	Deadline	EnergiaC2C1	EnergiaC3C1	Caminho Máximo	System QoS	UR2UR1	UR3UR1	
0.400000	3500	0.281526	0.239320	3334.000000	0.411167	1.274029		1.345395
0.500000	3500	0.360311	0.302495	3334.000000	0.512550	1.275274		1.346640
0.600000	3500	0.449956	0.364969	3283.000000	0.617530	1.246408		1.337888
0.700000	3500	0.457894	0.372012	3283.000000	0.706880	1.251957		1.343012
0.800000	3500	0.644003	0.436065	3499.000000	0.800000	1.093528		1.323775
0.900000	3500	0.785886	0.510779	3237.000000	0.902400	1.068962		1.334393
1.000000	3500	0.841837	0.533295	3378.000000	1.000000	1.072279		1.351281

Figura A2.4: Arquivo de saída com resultados de experimento para diferentes QoS



```
C3_70nos.txt - Notepad
File Edit Format View Help
1499.074305 // Consumo de Energia
0.718960 // QoS efetiva
1,2 // Nivel de operação para as duas classes da tarefa 1
2,1,1 // Nivel de operação para as três classes da tarefa 2
3,2,2,2 // Nivel de operação para as quatro classes da tarefa 3
1,1 // Nivel de operação para as duas classes da tarefa 4
2,1,1
3,2,2,2
1,2
2,2,2
2,3,3
1,1
2,1,1
2,3,3
1,2
2,3,3
3,2,2,2
1,3
2,2,2
```

Figura A2.5: Arquivo de saída para uso de *SCN4*

O capítulo 4 abordou os conceitos para cálculo do consumo de energia do sistema e taxa de utilização dos processadores. Caso o modelo de cálculo do consumo de energia abordado neste trabalho seja de interesse em aplicações futuras, ficaremos gratos em fornecer os códigos de cálculo, não reproduzidos aqui pela quantidade proibitiva de páginas necessárias.

A2.3 Execução *SCN4*

No início dos trabalhos para o desenvolvimento dos algoritmos aqui apresentados, o aplicativo desenvolvido com o NI LabWindows foi usado para o desenvolvimento dos algoritmos para os cenários C1, C2 e C3. No entanto, em sua última versão, foi utilizado apenas para o desenvolvimento e experimentação do algoritmo *SCN4*. A Figura A2.6 mostra a tela de trabalho da aplicação. Devido ao desenvolvimento anterior dos algoritmos para os outros cenários, alguns campos relacionados ainda permanecem, mas não são mais usados em sua totalidade.

A escolha de uma ferramenta de interface visual para a experimentação do algoritmo foi feita de modo a facilitar a automatização dos testes, considerando a enorme quantidade de experimentos executados para obtenção dos resultados apresentados neste trabalho. Deixamos claro aqui que o intuito deste aplicativo nunca foi o de ser uma ferramenta de automação de experimentos, e sim um meio mais direto de levantar resultados experimentais para *SCN4*. Sendo assim, a interface do aplicativo está longe de ser uma interface amigável e corretamente criada para o uso de outros pesquisadores.

Como descrito no item anterior, este aplicativo usa o arquivo de saída gerado no código desenvolvido para os algoritmos *offline*, exemplificado na Figura A2.6, como um arquivo de entrada para execução do algoritmo *SCN4*. Além deste arquivo, são usados outros quatro arquivos para a experimentação do algoritmo *online*.

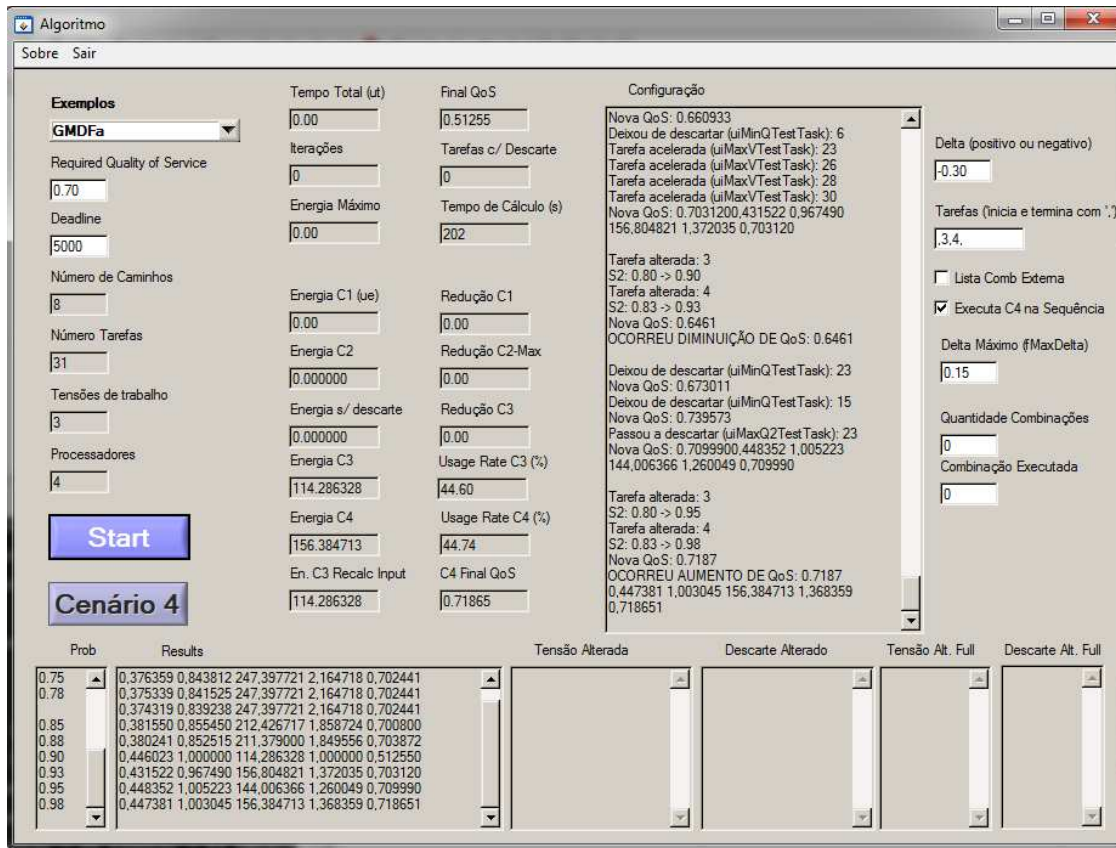


Figura A2.6: Tela do aplicativo para o algoritmo *SCN4*

Novamente os caminhos de execução de cada variação das aplicações são usados como entrada para a experimentação do algoritmo. Estes caminhos são definidos por parte do código desenvolvido em Dev C++, e seguem o formato do arquivo mostrado na Figura A2.7. O valor '250' foi usado para definir o final do caminho dentro do arquivo, já que o formato do arquivo não informa a quantidade de tarefas que compõem o caminho, apenas a quantidade total de caminhos, expressa na primeira linha do arquivo.

Assim como para os algoritmos *offline*, a execução de *SCN4* faz uso das tabelas apresentadas no apêndice anterior, alteradas para refletir também o mapeamento das tarefas de cada aplicação nas diferentes variações de quantidade de processadores. Um exemplo deste arquivo segue na Figura A2.8.

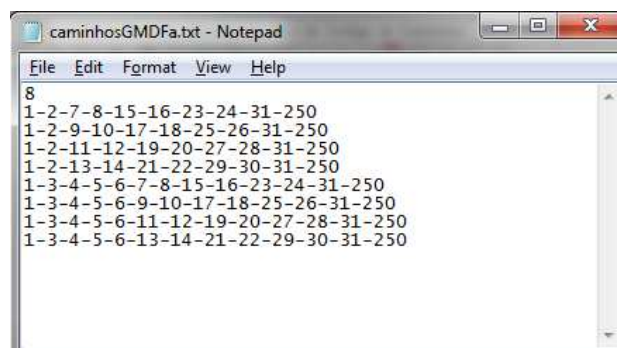


Figura A2.7: Arquivo de caminhos para a execução do experimento

```

tabelaExemplo.txt - Notepad
File Edit Format View Help
5,11,2,2,0.8 tarefas, deadline, níveis de operação, processadores, QoS mínima
1,2,3,12 processador tarefa 1, níveis de operação tarefa 1, consumo nível 1, consumo nível 2
1 número de classes
1.0,2,1 probabilidade da classe 1, tempo execução classe 1 no nível 1, tempo execução classe 1 no nível 2
2,2,4,16 processador tarefa 2, níveis de operação tarefa 2, consumo nível 1, consumo nível 2
2 número de classes
0.9,2,1 probabilidade da classe 1, tempo execução classe 1 no nível 1, tempo execução classe 1 no nível 2
1.0,4,2 probabilidade da classe 2, tempo execução classe 2 no nível 1, tempo execução classe 2 no nível 2
2,2,6,24 tarefa 3
2
0.8,2,1
1.0,6,3
2,2,5,20 tarefa 4
1
1.0,2,1
1,2,1,4 tarefa 5
2
0.9,4,2
1.0,6,3

```

Figura A2.8: Tabela de dados para a aplicação do experimento

As alterações na distribuição das classes de entrada são tratadas fazendo a comparação da tabela exemplificada na Figura A2.8 e uma nova tabela, que descreve a nova distribuição probabilística das classes de dados de entrada para as tarefas que fazem descarte. Esta variação está exemplificada na Figura A2.9, que mostra a mudança na distribuição, comparando a tabela original e a tabela com a nova distribuição. O uso deste arquivo, no entanto, é necessário apenas quando o aplicativo não está fazendo a variação automática do *delta*, e as novas distribuições de classes de dados de entrada estão sendo informados manualmente. Quando trabalhando em modo automatizado, o aplicativo desenvolvido faz a variação do *delta* de acordo com a configuração nos campos “*Delta* (positivo ou negativo)” e “*Delta* Máximo (fMaxDelta)”, definindo os valores mínimo e máximo de variação, respectivamente. As tarefas que vão sofrer o *delta* de variação neste experimento são definidas no campo “Tarefas (inicia e termina com ‘,’)”.

É necessário citar novamente que o algoritmo trabalha com a somatória de probabilidades de distribuição das classes de entrada. Assim, como descrito para as tabelas do Apêndice A1, os valores de distribuição probabilística das classes de entrada mostrados na Figura A2.9 são a soma acumulada da distribuição das classes para as tarefas. No exemplo, a classe $c_{3,2}$, destacada na figura, passa de 0,27 para 0,52, enquanto $c_{3,3}$ (descartada para esta aplicação) passa de 0,20 para 0,05.

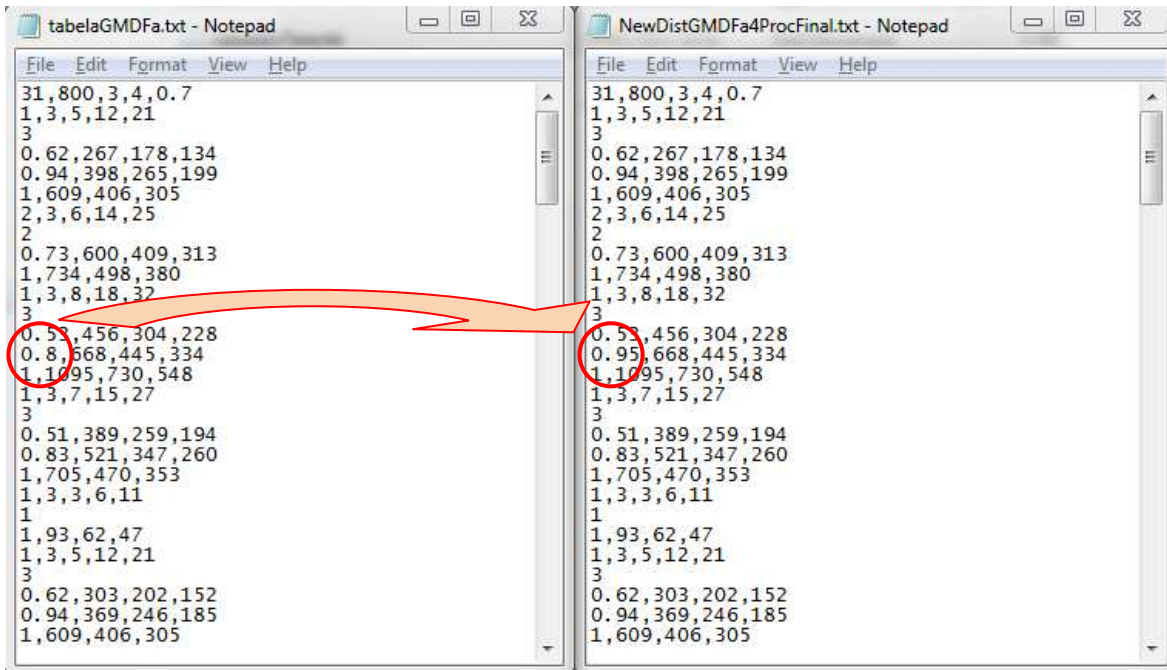


Figura A2.9: Nova distribuição das classes de entrada da tarefa 3, GMDFa-4, $Q = 0,7$

O último arquivo de entrada para a execução do algoritmo *SCN4* não conta com dados que sejam usados pelo algoritmo, e não precisa necessariamente ser usado. Este arquivo define parte da automação dos experimentos, de acordo com as combinações de tarefas que descartam classes listadas no arquivo. O intuito deste arquivo é substituir a entrada manual de combinações de tarefas com descarte no campo “Tarefas (inicia e termina com ‘,’)”, possibilitando a execução de várias combinações de descarte simultâneo em uma ou mais tarefas. A Figura A2.10 exemplifica um arquivo em que 13 combinações de descarte entre as tarefas que podem descartar classes de entrada. A primeira linha define a quantidade de combinações que serão experimentadas, no caso do exemplo, parte das combinações possíveis entre as 6 tarefas que descartam classes de entrada. O primeiro valor de cada linha define a quantidade de tarefas que fazem parte da combinação. Para forçar o uso do arquivo externo, a caixa “Lista Comb Externa” deve estar marcada na tela do aplicativo.

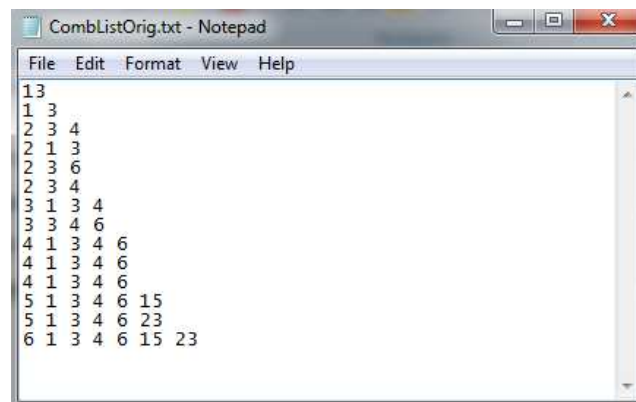


Figura A2.10: Lista de combinações das tarefas que descartam classes de entrada

Uma vez definidos os arquivos de entrada e como serão feitas as combinações (uso do arquivo externo com combinações de tarefas ou entrada manual), o aplicativo pode ser executado para experimentação em uma das várias configurações de processadores definidas para cada aplicação. Inicialmente, através do botão “Start”, as tabelas de trabalho são preenchidas com os dados dos arquivos de entrada e as listas de combinações, assim como outros dados informados via interface de tela. A Figura A2.11 traz um esboço do funcionamento do aplicativo para experimentação das aplicações com *SCN4*.

A função do botão “Start” é apenas preparar os dados *offline* de *SCN4* para a execução da etapa *online* do algoritmo. O Botão “Cenário 4” inicia os experimentos com a etapa *online* do algoritmo *SCN4*. Caso esteja em uso o arquivo de entrada em que são definidas as combinações de tarefas que fazem descarte de classes de dados de entrada, o aplicativo executa um ciclo de repetição da etapa *online*, de acordo com a quantidade de combinações requeridas. Para cada combinação são experimentadas diversas variações de *delta*, seguindo os dados definidos para tal fim na tela de trabalho. Também é possível que o botão “Start” inicie a experimentação das etapas *offline* e *online* do algoritmo, bastando a caixa “Executa C4 na Sequência” estar marcada.

Para evitar o cálculo dos dados *offline* antes de toda experimentação *online*, estes dados são calculados uma vez (para cada variação das aplicações) e guardados em tabelas de trabalho que são resgatadas antes da experimentação da etapa *online* para cada combinação de tarefas.

Os dados de consumo de energia e utilização dos processadores são novamente calculados para fim de comparação com os resultados obtidos por *SCN3*. Os métodos de cálculo de consumo e utilização dos processadores são os mesmos utilizados nos algoritmos *offline*.

Como mostrado para os algoritmos *offline* *SCN1*, *SCN2* e *SCN3*, o arquivo de saída para *SCN4* também pode assumir diversos formatos. Para montagem das tabelas e gráficos apresentados neste trabalho o aplicativo gerou, para cada combinação de tarefas, os dados de qualidade efetiva, consumo de energia e utilização dos processadores, além da relação entre os valores

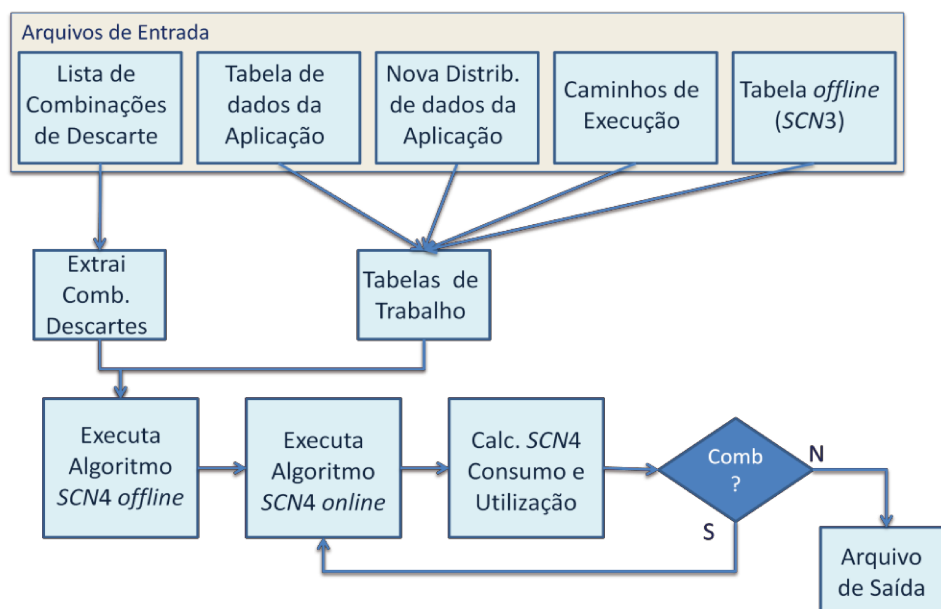


Figura A2.11: Fluxo de execução do algoritmo *SCN4*

iniciais calculados para o cenário C3 e os resultados obtidos para o cenário C4. Um exemplo do arquivo de saída é mostrado na Figura A2.12. A linha abaixo de cada bloco de resultados identifica a combinação de tarefas utilizada no experimento.

```

Resultados.txt - Notepad
File Edit Format View Help
Tarefas com descarte: 1, 3, 4, 6, 15, 23,
QoS C3: 0.712550
Original Energy C3: 114.286328
Energy C3:114.286328

0,437663 0,981257 165,824409 1,450956 0,704052
0,436950 0,979659 165,824409 1,450956 0,704052
0,436238 0,978062 165,824409 1,450956 0,704052
0,435525 0,976464 165,824409 1,450956 0,704052
0,434813 0,974867 165,824409 1,450956 0,704052
0,379650 0,851190 211,822506 1,853437 0,704052
0,446023 1,000000 114,286328 1,000000 0,712550
0,406266 0,910865 180,703656 1,581148 0,705500
0,431536 0,967521 157,879887 1,381442 0,702180
0,431191 0,966748 165,436861 1,447565 0,711542
Tarefas da Combinação: 3,

0,385479 0,864258 220,221901 1,926931 0,709982
0,384471 0,861999 220,221901 1,926931 0,709982
0,383464 0,859740 220,221901 1,926931 0,709982
0,382456 0,857482 220,221901 1,926931 0,709982
0,381449 0,855223 220,221901 1,926931 0,709982
0,380441 0,852964 220,221901 1,926931 0,709982
0,446023 1,000000 114,286328 1,000000 0,712550
0,378236 0,848021 219,435013 1,920046 0,711842
0,378236 0,848021 219,435013 1,920046 0,711842
0,378236 0,848021 219,435013 1,920046 0,711842
Tarefas da Combinação: 6, 15,

0,389754 0,873843 220,221901 1,926931 0,709982
0,388034 0,869987 220,221901 1,926931 0,709982
0,386314 0,866130 220,221901 1,926931 0,709982
0,384594 0,862274 220,221901 1,926931 0,709982
0,382874 0,858418 220,221901 1,926931 0,709982
0,381154 0,854561 220,221901 1,926931 0,709982
0,446023 1,000000 114,286328 1,000000 0,712550
0,405056 0,908152 180,703656 1,581148 0,705500
0,430326 0,964808 157,879887 1,381442 0,702180
0,447876 1,004155 143,750596 1,257811 0,704427
Tarefas da Combinação: 3, 6, 15,

```

Figura A2.12: Arquivo de saída do algoritmo SCN4

O uso do NI LabWindows é feito sob licença. No entanto, o código utilizado pode ser migrado para outra aplicação de licença livre, com pequenas modificações de acesso às interfaces.

Apêndice A3: Tabelas de Resultados

Neste apêndice são relacionadas as tabelas dos resultados obtidas durante os experimentos, que foram usadas para montar os gráficos apresentados neste trabalho.

A3.1 Resultados SCN2 e SCN3

A Tabela A3.1 mostra os resultados de cada aplicação quando exigida uma qualidade mínima de serviço $Q_{MIN} = 0,50$. O código dos algoritmos já fornece como resultado a relação entre o valor de consumo de energia obtido nos cenários C2 e C3 em relação ao cenário de referência C1. Estas mesmas relações de valores também são apresentadas para a utilização dos processadores.

A Tabela A3.2 mostra as mesmas relações de valores de consumo de energia e ocupação dos processadores exigindo, no entanto, uma qualidade mínima de serviço $Q_{MIN} = 0,70$ durante a execução dos algoritmos.

As Tabelas A3.1 e A3.2 definem os valores apresentados em forma de gráfico nas Figuras 6.2 e 6.3 do capítulo 6.

Tabela A3.1: Relações de consumo de energia e ocupação dos processadores quando qualidade de serviço mínima $Q_{MIN} = 0,50$

$Q_{MIN} = 0,5$	EC2/EC1	EC3/EC1	UR2/UR1	UR3/UR1
GMDFa-1	0,462864	0,306201	1,069624	1,385277
GMDFa-2	0,380851	0,30846	1,256066	1,369612
GMDFa-4	0,316325	0,291547	1,356264	1,389537
GMDFa-8	0,360311	0,302495	1,275274	1,346640
TGFF70-1	0,237554	0,237554	1,531822	1,531822
TGFF70-2	0,273231	0,273231	1,430500	1,430500
TGFF70-4	0,282996	0,281969	1,451905	1,454532
TGFF70-8	0,478933	0,477906	1,209831	1,212458
TGFF99-1	0,212827	0,203929	1,580802	1,621152
TGFF99-2	0,297582	0,269094	1,437350	1,478304
TGFF99-5	0,407677	0,360928	1,287441	1,369967
TGFF99-8	0,269817	0,252184	1,558024	1,598105
Sonar-1	0,389533	0,356723	1,082276	1,282622
Vocoder-1	0,346961	0,346961	3,052778	3,052778
Vocoder-2	0,306338	0,306338	3,435958	3,435958

Tabela A3.2: Relações de consumo de energia e ocupação dos processadores quando qualidade de serviço mínima $Q_{MIN} = 0,70$

$Q_{MIN} = 0,7$	EC2/EC1	EC3/EC1	UR2/UR1	UR3/UR1
GMDFa-1	0,599414	0,375981	1,065107	1,405515
GMDFa-2	0,472321	0,37416	1,265132	1,389039
GMDFa-4	0,429925	0,360266	1,313887	1,391523
GMDFa-8	0,457894	0,372012	1,251957	1,343012
TGFF70-1	0,319115	0,319115	1,526736	1,526736
TGFF70-2	0,370028	0,368552	1,430855	1,433482
TGFF70-4	0,391963	0,390486	1,423178	1,425805
TGFF70-8	0,602067	0,600591	1,214072	1,216699
TGFF99-1	0,29597	0,285092	1,548829	1,584077
TGFF99-2	0,409492	0,373475	1,384502	1,427212
TGFF99-5	0,509572	0,442937	1,279987	1,372661
TGFF99-8	0,372301	0,33054	1,479529	1,541323
Sonar-1	0,559756	0,495334	1,065996	1,290378
Vocoder-1	0,476028	0,476028	3,060480	3,060480
Vocoder-2	0,417868	0,417868	3,443660	3,443660

A3.2 Resultados SCN4

Os gráficos de resultados apresentados para o cenário C4 foram montados a partir de vários dados de experimentação do algoritmo *SCN4*. Este apêndice apresenta, na forma de tabelas e alguns gráficos, os resultados usados para montagem dos gráficos para cada variação das aplicações apresentados no capítulo 6, partindo dos resultados obtidos para o cenário C3 por *SCN3*.

A3.2.1 Vocoder

Por ser a aplicação com maior número de tarefas de execução e maior número de níveis de operação dos processadores, a aplicação Vocoder foi a mais explorada para obtenção de resultados no Cenário 4, fazendo parte dos experimentos na forma de 3 aplicações diferentes, Vocoder-2, $Q = 0,5$, Vocoder-2, $Q = 0,7$ e Vocoder-1, $Q=0,7$.

Vocoder-2, $Q = 0,5$

Os resultados apresentados na forma de gráficos nas Figuras 6.4 a 6.6 do capítulo 6 para a aplicação Vocoder-2, $Q = 0,5$ tiveram como origem os dados que serão apresentados aqui. Todas as tabelas foram montadas com os valores fornecidos pelo experimento, através do programa feito em LabWindows para o cenário C4.

Como resultado da configuração final para o cenário C3 obtido por *SCN3* para a aplicação Vocoder-2, $Q = 0,5$, até 6 tarefas podem fazer descarte de classes de dados de entrada. Assim, são possíveis 63 combinações de descarte de tarefas, desde uma até 6 tarefas descartando classes de dados de entrada ao mesmo tempo.

Devido à grande quantidade de combinações possíveis, foram consideradas até 7 combinações para cada quantidade de tarefas com descarte, totalizando 34 experimentos. As combinações usadas em cada situação seguem na Tabela A3.3. As tarefas que podem descartar classes de entrada são as tarefas 2, 8, 13, 21, 39 e 65.

Tabela A3.3: Combinações de descartes usadas no Cenário 4 para Vocoder-2, $Q = 0,5$

Descarte	Combinações usadas
1 Tarefa	2, 8, 13, 21, 39, 65
2 Tarefas	2/8, 2/13, 2/65, 13/65, 21/39, 8/39, 13/21
3 Tarefas	2/8/13, 8/13/65, 2/13/21, 21/39/65, 2/21/39, 8/21/39, 2/8/65
4 Tarefas	2/8/13/65, 8/13/21/65, 2/21/39/65, 2/8/39/65, 8/13/39/65, 2/8/21/39, 2/13/21/39
5 Tarefas	2/8/13/21/39, 2/8/13/21/65, 2/8/13/39/65, 2/8/21/39/65, 2/13/21/39/65, 8/13/21/39/65
6 Tarefas	2/8/13/21/39/65

Para os 6 casos de descarte de apenas uma tarefa, foram montados os gráficos de resultados para consumo de energia, ocupação dos processadores e qualidade efetiva após a execução de *SCN4*, expostos na Figura A3.1.

As linhas contínuas nos gráficos da Figura A3.1 mostram a média de valores calculada através dos valores obtidos para os seis experimentos de *SCN4* em que apenas uma tarefa descarta classes de dados de entrada. Os pontos em dispersão são os resultados obtidos quando cada uma das tarefas fez descarte de classes de entrada, e foram usados no cálculo das médias. Os valores apresentados nos gráficos estão na Tabela A3.4.

Para as outras quantidades de tarefas com descarte de classes, não foram gerados os gráficos com dados em dispersão, apenas as médias, que foram montadas em um mesmo gráfico, apresentados nas Figuras 6.3 a 6.5. A Tabela A3.5 mostra os resultados de *SCN4* combinando descartes em duas tarefas. As Tabelas A3.6 a A3.9 mostram, respectivamente, os resultados para descartes simultâneos de classes de dados de entrada em 3, 4, 5 e 6 tarefas. A indicação *No Result* indica onde a degradação dos dados de entrada foi tão grande a ponto de o algoritmo *SCN4* não encontrar uma configuração para que o sistema cumpra as restrições de funcionamento.

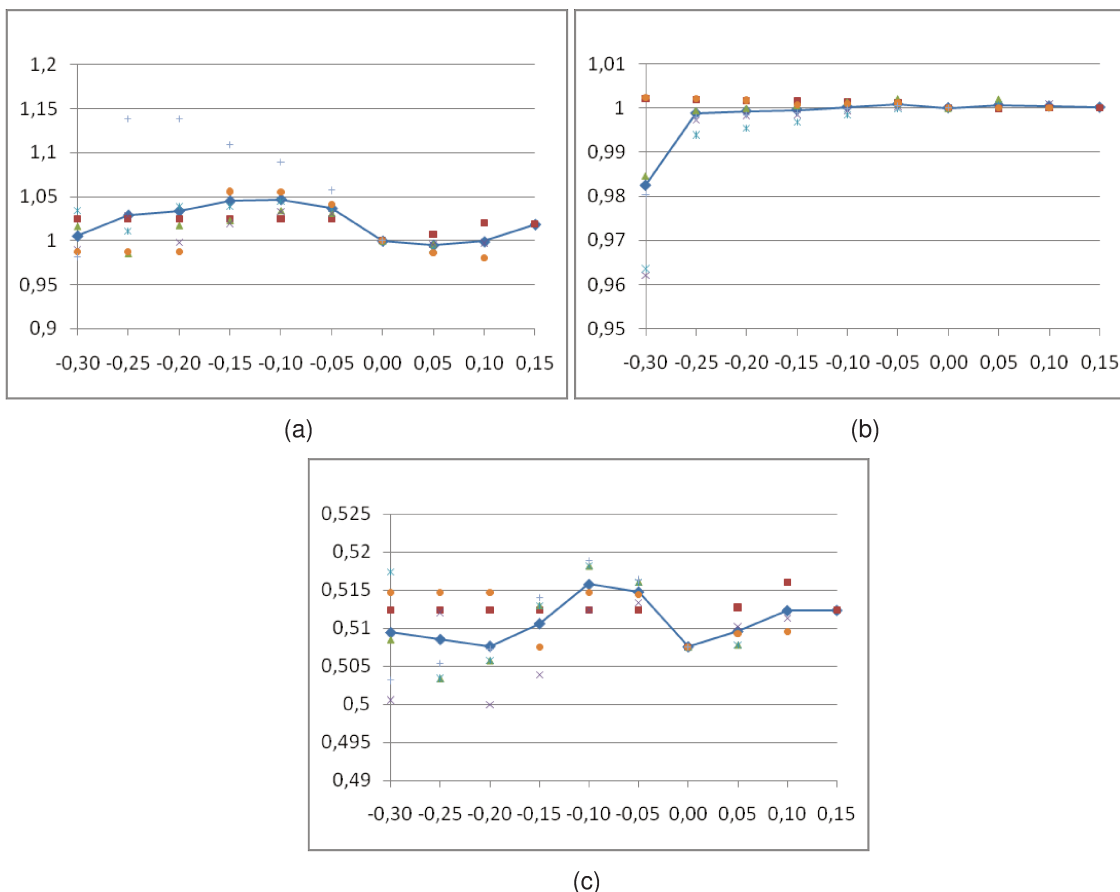


Figura A3.1: Gráficos das médias de (a) consumo de energia, (b) ocupação dos processadores e (c) qualidade efetiva quando apenas uma tarefa descarta classes de dados de entrada.

Tabela A3.4: Resultados para descarte de classes nas tarefas (a) 2, (b) 8, (c) 13, (d) 21, (e) 39, (f) 65 e (g) a média dos valores

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,00214	1,024971	0,512355
-0,25	1,00195	1,024971	0,512355
-0,20	1,001759	1,024971	0,512355
-0,15	1,001568	1,024971	0,512355
-0,10	1,001377	1,024971	0,512355
-0,05	1,001187	1,024971	0,512355
0,00	1	1,000000	0,507569
0,05	0,999876	1,007506	0,512709
0,10	1,000008	1,020340	0,516066
0,15	1,000122	1,01874	0,512355

(a)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,984638	1,016601	0,508497
-0,25	0,999459	0,985290	0,503411
-0,20	0,999951	1,017505	0,505792
-0,15	1,000613	1,023233	0,512969
-0,10	1,001436	1,034630	0,518163
-0,05	1,002163	1,031450	0,516066
0,00	1	1,000000	0,507569
0,05	1,002076	0,995731	0,507839

(b)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,962128	0,989719	0,500570
-0,25	0,997337	1,028555	0,512046
-0,20	0,998321	0,997796	0,500001
-0,15	0,998554	1,019807	0,503908
-0,10	0,999497	1,032817	0,512425
-0,05	1,00032	1,030613	0,513369
0,00	1	1,000000	0,507569
0,05	1,000373	0,995853	0,510225
0,10	1,000952	0,997374	0,511346

(c)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,963563	1,034255	0,517376
-0,25	0,9939	1,011023	0,503411
-0,20	0,995356	1,039161	0,505792
-0,15	0,99686	1,038652	0,512969
-0,10	0,998425	1,044821	0,518163
-0,05	0,99981	1,036259	0,516066
0,00	1	1,000000	0,507569
0,05	1,000839	0,990732	0,507839

(d)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,00235	0,987073	0,514698
-0,25	1,002081	0,987073	0,514698
-0,20	1,001812	0,987073	0,514698
-0,15	1,00077	1,055973	0,507497
-0,10	1,000943	1,055638	0,514698
-0,05	1,001091	1,041323	0,514437
0,00	1	1,000000	0,507569
0,05	1,000018	0,986747	0,509280
0,10	1,000123	0,980041	0,509551

(e)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,980361	0,981723	0,503144
-0,25	0,997958	1,138426	0,505331
-0,20	0,99838	1,138070	0,507231
-0,15	0,998982	1,109135	0,513994
-0,10	0,999755	1,089468	0,518813
-0,05	1,000437	1,057246	0,516371
0,00	1	1,000000	0,507569

(f)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,98253	1,0057237	0,50944
-0,25	0,99878083	1,029223	0,508542
-0,20	0,99926317	1,034096	0,5076448
-0,15	0,99955783	1,0452952	0,5106153
-0,10	1,00023883	1,0470575	0,5157695
-0,05	1,00083467	1,036977	0,5147773
0,00	1	1	0,507569
0,05	1,0006364	0,9953138	0,5095784
0,10	1,000361	0,9992517	0,512321
0,15	1,000122	1,018738	0,512355

(g)

Tabela A3.5: Resultados para descarte simultâneo das combinações (a) 2/8, (b) 2/13, (c) 2/65, (d) 13/65, (e) 21/39, (f) 8/39, (g) 13/21 e (h) a média dos valores.5

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,985783	1,016601	0,508497
-0,25	1,000412	0,985290	0,503411
-0,20	1,000714	1,017505	0,505792
-0,15	1,001185	1,023233	0,512969
-0,10	1,001818	1,034630	0,518163
-0,05	1,002354	1,031450	0,516066
0,00	1	1,000000	0,507569
0,05	1,002226	1,011273	0,512982
0,10	1,002355	1,011826	0,510905
0,15	1,002419	1,009796	0,507231

(a)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,963272	0,989719	0,500570
-0,25	0,99829	1,028555	0,512046
-0,20	0,999084	0,997796	0,500001
-0,15	0,999126	1,019807	0,503908
-0,10	0,999879	1,032817	0,512425
-0,05	1,000511	1,030613	0,513369
0,00	1	1,000000	0,507569
0,05	1,000523	1,011434	0,515392
0,10	1,001179	1,001244	0,509019
0,15	1,001303	1,012773	0,510734

(b)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,981505	0,981723	0,503144
-0,25	0,998911	1,138426	0,505331
-0,20	0,999143	1,138070	0,507231
-0,15	0,999555	1,109135	0,513994
-0,10	1,000136	1,089468	0,518813
-0,05	1,000627	1,057246	0,516371
0,00	1	1,000000	0,507569
0,05	0,999876	1,007506	0,512709
0,10	1,000008	1,02034	0,516066
0,15	1,000619	0,986728	0,506962

(c)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,970726	0,989719	0,500570
-0,25	0,975963	1,006658	0,512046
-0,20	0,975795	0,987060	0,500001
-0,15	0,997613	1,080523	0,501885
-0,10	0,998474	1,082146	0,508017
-0,05	0,999711	1,062663	0,517393
0,00	1	1,000000	0,507569
0,05	1,000373	0,995853	0,510225
0,10	1,000952	0,997374	0,511346

(d)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,965332	1,034255	0,517376
-0,25	0,995401	1,011023	0,503411
-0,20	0,997169	1,023708	0,512896
-0,15	0,998674	1,021235	0,514698
-0,10	0,99859	1,061295	0,509632
-0,05	0,999854	1,052498	0,518163
0,00	1	1,000000	0,507569
0,05	1,001131	0,985603	0,509551
0,10	1,001233	0,967058	0,504455

(e)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,986407	1,016601	0,508497
-0,25	1,000959	0,985290	0,503411
-0,20	1,001763	1,003296	0,512896
-0,15	1,002427	1,006771	0,514698
-0,10	1,001601	1,050485	0,509632
-0,05	1,002208	1,047412	0,518163
0,00	1	1,000000	0,507569
0,05	1,002368	0,990573	0,509551
0,10	1,00247	0,971762	0,504455

(f)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,952004	0,996091	0,500080
-0,25	0,951278	1,022633	0,517494
-0,20	0,950318	0,991165	0,500001
-0,15	0,995491	1,014080	0,500884
-0,10	0,997144	1,038429	0,507381
-0,05	0,999084	1,041634	0,517087
0,00	1	1,000000	0,507569
0,05	1,001486	0,994681	0,510496
0,10	1,002062	0,984086	0,506233

(g)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,972147	1,00353	0,505533
-0,25	0,98874486	1,025411	0,508164
-0,20	0,98914086	1,022657	0,505545
-0,15	0,999153	1,039255	0,509005
-0,10	0,99966314	1,05561	0,512009
-0,05	1,00062129	1,046217	0,516659
0,00	1	1	0,507569
0,05	1,00114043	0,99956	0,511558
0,10	1,00146557	0,993384	0,508926
0,15	1,001447	1,003099	0,508309

(h)

Tabela A3.6: Resultados para descarte simultâneo das combinações (a) 2/8/13, (b) 8/13/65, (c) 2/13/21, (d) 21/39/65, (e) 2/21/39, (f) 8/21/39, (g) 2/8/65 e (h) a média dos valores.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,981595	1,026109	0,500080
-0,25	0,953244	1,030083	0,517494
-0,20	0,953193	0,998800	0,500001
-0,15	0,999816	0,998600	0,500884
-0,10	1,000537	1,027953	0,507381
-0,05	1,001628	1,036625	0,517087
0,00	1	1,000000	0,507569
0,05	1,002870	1,002924	0,51024
0,10	1,003536	1,00569	0,509289
0,15	1,001818	1,028854	0,520915

(a)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,960514	1,003643	0,500080
-0,25	0,959455	1,030083	0,517494
-0,20	0,958162	0,998800	0,500001
-0,15	0,975004	0,977555	0,500884
-0,10	0,999714	1,060829	0,510082
-0,05	1,000826	1,056766	0,515655
0,00	1	1,000000	0,507569
0,05	1,002723	0,99972	0,510496
0,10	1,003299	0,988922	0,506233

(b)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,953148	0,996091	0,500080
-0,25	0,952231	1,022633	0,517494
-0,20	0,951081	0,991165	0,500001
-0,15	0,996063	1,014080	0,500884
-0,10	0,997526	1,038429	0,507381
-0,05	0,999275	1,041634	0,517087
0,00	1	1,000000	0,507569
0,05	1,001633	0,998121	0,510238
0,10	1,002299	1,001102	0,509289
0,15	1,000581	1,024472	0,520915

(c)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,97393	1,034255	0,517376
-0,25	0,9743	0,998705	0,503411
-0,20	0,974632	0,999690	0,507497
-0,15	0,996882	1,100486	0,510909
-0,10	0,998676	1,078396	0,520921
-0,05	0,999242	1,073161	0,516728
0,00	1	1,000000	0,507569
0,05	1,001131	0,985603	0,509551
0,10	1,001233	0,967058	0,504455

(d)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,966477	1,034255	0,517376
-0,25	0,996354	1,011023	0,503411
-0,20	0,997931	1,023708	0,512896
-0,15	0,999246	1,021235	0,514698
-0,10	0,998971	1,061295	0,509632
-0,05	1,000045	1,052498	0,518163
0,00	1	1,000000	0,507569
0,05	1,001278	0,989089	0,509293
0,10	1,00147	0,984061	0,507501
0,15	0,999752	1,007173	0,519086

(e)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,978207	1,071686	0,522880
-0,25	0,950625	1,010445	0,503411
-0,20	0,951218	1,010893	0,507497
-0,15	0,998512	1,016618	0,509890
-0,10	1,000358	1,024929	0,520269
-0,05	1,000969	1,046603	0,516423
0,00	1	1,000000	0,507569
0,05	1,003478	0,977268	0,504455
0,10	1,00359	0,971672	0,504724

(f)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,979977	1,001730	0,517376
-0,25	0,979312	0,973177	0,503411
-0,20	0,978758	0,980394	0,507497
-0,15	1,000245	1,084256	0,510909
-0,10	1,000795	1,083983	0,513706
-0,05	1,001745	1,063451	0,520111
0,00	1	1,000000	0,507569
0,05	1,002226	1,011273	0,512982
0,10	1,002355	1,011826	0,510905
0,15	1,002976	0,991094	0,507231

(g)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,97054971	1,023967	0,51075
-0,25	0,966503	1,010878	0,509447
-0,20	0,966425	1,000493	0,505056
-0,15	0,99510971	1,030404	0,507008
-0,10	0,999511	1,053688	0,512767
-0,05	1,00053286	1,052963	0,517322
0,00	1	1	0,507569
0,05	1,00219129	0,994857	0,509608
0,10	1,00254029	0,990047	0,507485
0,15	1,00128175	1,012898	0,517037

(h)

Tabela A3.7: Resultados para descarte simultâneo das combinações (a) 2/8/13/65, (b) 8/13/21/65, (c) 2/21/39/65, (d) 2/8/39/65, (e) 8/13/39/65, (f) 2/8/21/39, (g) 2/13/21/39 e (h) a média dos valores.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,961658	1,003643	0,500080
-0,25	0,960409	1,030083	0,517494
-0,20	0,958925	0,998800	0,500001
-0,15	0,975576	0,977555	0,500884
-0,10	1,000095	1,060829	0,510082
-0,05	1,001016	1,056766	0,515655
0,00	1	1,000000	0,507569
0,05	1,00287	1,002924	0,510238
0,10	1,003536	1,00569	0,509289
0,15	1,002375	1,009648	0,520915

(a)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	0,952499	1,015591	0,506213
-0,10	0,972711	0,996912	0,509443
-0,05	0,99986	1,065415	0,513923
0,00	1	1,000000	0,507569
0,05	1,003833	0,986255	0,505391
0,10	1,004359	0,975235	0,50117
0,15	1,004917	0,956786	0,50117

(b)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,975074	1,034255	0,517376
-0,25	0,975254	0,998705	0,503411
-0,20	0,975395	0,999690	0,507497
-0,15	0,997454	1,100486	0,510909
-0,10	0,999057	1,078396	0,520921
-0,05	0,999433	1,073161	0,516728
0,00	1	1	0,507569
0,05	1,001278	0,989089	0,509293
0,10	1,001470	0,984061	0,507501
0,15	1,000310	0,988723	0,519086

(c)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,981746	1,001730	0,517376
-0,25	0,980812	0,973177	0,503411
-0,20	0,97999	0,980394	0,507497
-0,15	1,001207	1,084256	0,510909
-0,10	1,002069	1,068205	0,520921
-0,05	1,001787	1,068037	0,516728
0,00	1	1,000000	0,507569
0,05	1,002515	0,993826	0,509293
0,10	1,002707	0,988522	0,507501
0,15	1,001546	0,992955	0,519086

(d)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,962283	1,003643	0,500080
-0,25	0,960955	1,030083	0,517494
-0,20	0,959394	0,998800	0,500001
-0,15	0,975966	0,977555	0,500884
-0,10	1,000407	1,060829	0,510082
-0,05	1,001141	1,070816	0,512301
0,00	1	1,000000	0,507569
0,05	1,003013	0,98229	0,506825
0,10	1,003651	0,9657	0,50286
0,15	1,002366	0,958584	0,512611

(e)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,979351	1,071686	0,522880
-0,25	0,951579	1,010445	0,503411
-0,20	0,951981	1,010893	0,507497
-0,15	0,999085	1,016618	0,509890
-0,10	1,000739	1,024929	0,520269
-0,05	1,00116	1,046603	0,516423
0,00	1	1,000000	0,507569
0,05	1,003576	0,980271	0,5042
0,10	1,001985	0,99981	0,517617
0,15	1,00211	1,011339	0,519362

(f)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,954917	0,996091	0,50008
-0,25	0,953732	1,022633	0,51749
-0,20	0,952312	0,991165	0,50000
-0,15	0,997025	1,014080	0,50088
-0,10	0,998800	1,023224	0,51451
-0,05	0,999317	1,045910	0,51372
0,00	1,000000	1,000000	0,50757
0,05	1,001873	0,980729	0,50657
0,10	1,000869	1,002919	0,52119
0,15	1,000747	0,98823	0,50644

(g)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,9691715	1,018508	0,509645
-0,25	0,9637902	1,010854	0,510453
-0,20	0,9629995	0,996624	0,503749
-0,15	0,9855446	1,026592	0,505796
-0,10	0,9962683	1,044761	0,515175
-0,05	1,0005306	1,060958	0,515069
0,00	1	1	0,507569
0,05	1,0027083	0,987912	0,507401
0,10	1,0026539	0,988848	0,50959
0,15	1,002053	0,986609	0,514095

(h)

Tabela A3.8: Resultados para descarte simultâneo das combinações (a) 2/8/13/21/39, (b) 2/8/13/21/65, (c) 2/8/13/39/65, (d) 2/8/21/39/65, (e) 2/13/21/39/65, (f) 8/13/21/39/65 e (g) a média dos valores.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	0,949735	1,015591	0,506213
-0,10	0,999459	1,018454	0,509443
-0,05	1,000705	1,049567	0,511998
0,00	1	1,000000	0,507569
0,05	1,004230	0,985134	0,50684
0,10	1,00298	0,980989	0,505002
0,15	1,003104	0,992813	0,506704

(a)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	0,953071	1,015591	0,506213
-0,10	0,973092	0,996912	0,509443
-0,05	1,000051	1,065415	0,513923
0,00	1	1	0,507569
0,05	1,00393	0,989213	0,505135
0,10	1,002814	1,017125	0,519441
0,15	1,003249	0,98356	0,504733

(b)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,963428	1,003643	0,500080
-0,25	0,961909	1,030083	0,517494
-0,20	0,960156	0,998800	0,500001
-0,15	0,976538	0,977555	0,500884
-0,10	1,000789	1,060829	0,510082
-0,05	1,001332	1,070816	0,512301
0,00	1	1,000000	0,507569
0,05	1,00311	0,985249	0,506569
0,10	1,002106	1,007243	0,521192
0,15	1,002541	0,974877	0,506435

(c)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,959415	1,048257	0,522880
-0,25	0,958744	1,010445	0,503411
-0,20	0,957713	1,010893	0,507497
-0,15	0,974844	0,995193	0,509890
-0,10	0,999716	1,073020	0,515794
-0,05	1,000822	1,076726	0,514992
0,00	1	1,000000	0,507569
0,05	1,003576	0,980271	0,5042
0,10	1,001985	0,99981	0,517617
0,15	1,00242	0,967404	0,502961

(d)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,963515	0,996091	0,500080
-0,25	0,960896	1,022633	0,517494
-0,20	0,958044	0,991165	0,500001
-0,15	0,972785	0,992262	0,500884
-0,10	0,997777	1,071305	0,510082
-0,05	0,998979	1,076127	0,512301
0,00	1	1,000000	0,507569
0,05	1,001873	0,980729	0,506569
0,10	1,000869	1,002919	0,521192
0,15	1,001304	0,970577	0,506435

(e)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	0,953461	1,015591	0,506213
-0,10	0,973404	0,996912	0,509443
-0,05	1,000866	1,049934	0,521141
0,00	1	1,000000	0,507569
0,05	1,004073	0,968736	0,501757
0,10	1,002929	0,976739	0,512883

(f)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,9621193	1,015997	0,50768
-0,25	0,9605163	1,021054	0,5128
-0,20	0,9586377	1,000286	0,5025
-0,15	0,9634057	1,001964	0,50505
-0,10	0,9907062	1,036239	0,510715
-0,05	1,0004592	1,064764	0,514443
0,00	1	1	0,507569
0,05	1,0034653	0,981555	0,505178
0,10	1,0022805	0,997471	0,516221
0,15	1,0025236	0,977846	0,505454

(g)

Tabela A3.9: Resultados para descarte simultâneo de todas as tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	0,954033	1,015591	0,506213
-0,10	0,973786	0,996912	0,509443
-0,05	1,001057	1,049934	0,521141
0,00	1	1	0,507569
0,05	1,00423	0,985134	0,506838
0,10	1,00298	0,980989	0,505002
0,15	1,003662	0,978919	0,506704

Vocoder-2, Q = 0,7

Como resultado da configuração final para o cenário C3 obtido por *SCN3* para a aplicação Vocoder-2, Q = 0,7, até 4 tarefas podem fazer descarte de classes de dados de entrada. Assim, são possíveis 15 combinações de descarte de tarefas, desde uma até 4 tarefas descartando classes de dados de entrada ao mesmo tempo.

Para esta variação da aplicação Vocoder, uma menor quantidade de combinações foi encontrada, possibilitando a experimentação de todas as combinações de tarefas com descarte de classes de dados de entrada. As combinações usadas em cada situação seguem na Tabela A3.10. As tarefas que podem descartar classes de entrada são as tarefas 1, 8, 13 e 21.

Para os 4 casos de descarte de apenas uma tarefa, também foram montados os gráficos de resultados para consumo de energia, ocupação dos processadores e qualidade efetiva após a execução de *SCN4*, expostos na Figura A3.2.

Tabela A3.10: Combinações de descartes usadas no Cenário 4 para Vocoder-2, Q = 0,7

Descarte	Combinações usadas
1 Tarefa	1, 8, 13, 21
2 Tarefas	1/8, 1/13, 1/21, 8/13, 8/21, 13/21
3 Tarefas	1/8/13, 1/8/21, 1/13/21, 8/13/21
4 Tarefas	1/8/13/21

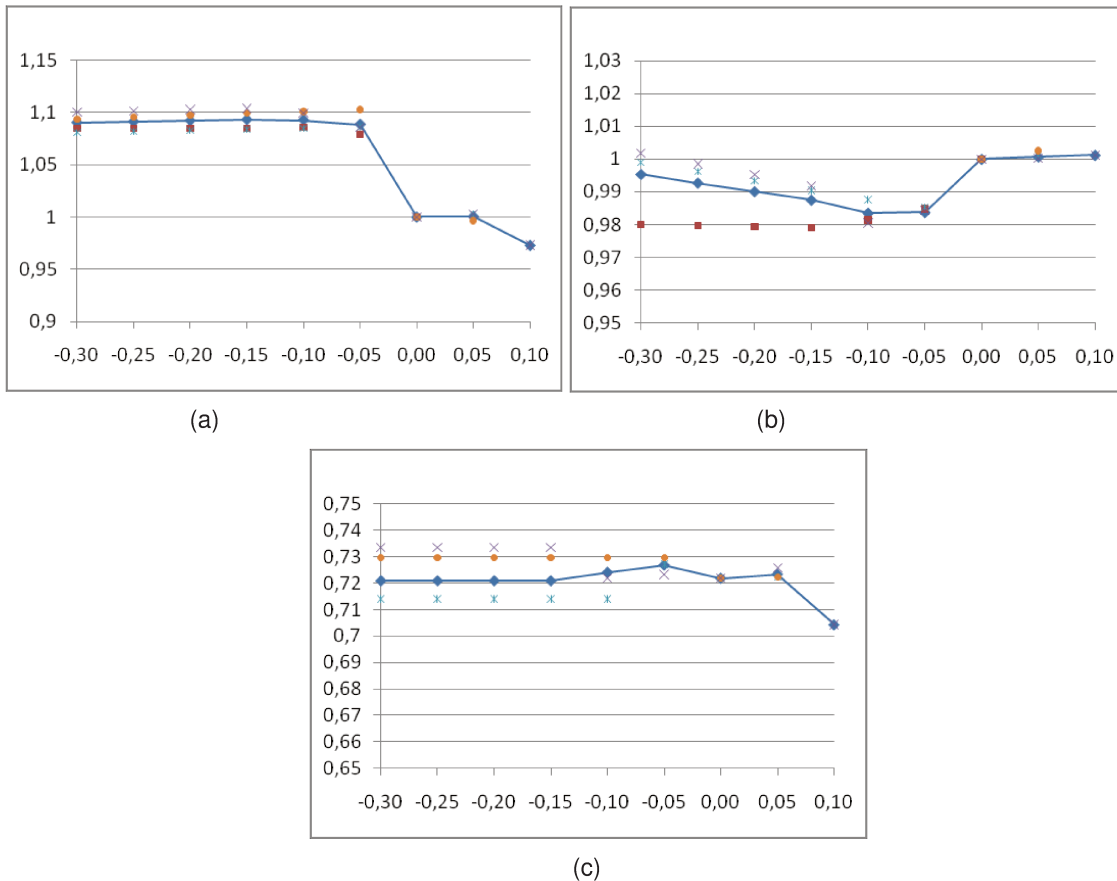


Figura A3.2: Gráficos das médias para (a) consumo de energia, (b) ocupação dos processadores e (c) qualidade efetiva quando apenas uma tarefa descarta classes de dados de entrada.

As linhas contínuas nos gráficos da Figura A3.2 mostram a média de valores calculada através dos valores obtidos para os seis experimentos de *SCN4* em que apenas uma tarefa descarta classes de dados de entrada. Os pontos em dispersão são os resultados obtidos quando cada uma das tarefas fez descarte de classes de entrada, e foram usados no cálculo das médias. Os valores apresentados nos gráficos estão na Tabela A3.11.

Tabela A3.11: Resultados para descarte de classes nas tarefas (a) 1, (b) 8, (c) 13, (d) 21 e (e) a média dos valores

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,980013	1,084564	0,706541
-0,25	0,979655	1,084710	0,706541
-0,20	0,979298	1,084860	0,706541
-0,15	0,97894	1,085014	0,706541
-0,10	0,981348	1,085446	0,731000
-0,05	0,984788	1,078719	0,727560
0,00	1	1,000000	0,721901

(a)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,998969	1,081118	0,714058
-0,25	0,996128	1,081894	0,714058
-0,20	0,993286	1,082708	0,714058
-0,15	0,990445	1,083564	0,714058
-0,10	0,987604	1,084466	0,714058
-0,05	0,985211	1,086126	0,727130
0,00	1	1,000000	0,721901
0,05	1,001196	1,002573	0,722285

(b)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,001881	1,100613	0,733653
-0,25	0,998559	1,101620	0,733653
-0,20	0,995237	1,102685	0,733653
-0,15	0,991916	1,103812	0,733653
-0,10	0,980488	1,099114	0,722000
-0,05	0,984283	1,085639	0,723330
0,00	1	1,000000	0,721901
0,05	1,000339	1,002860	0,725678
0,10	1,00121	0,973099	0,704307

(c)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,000636	1,093913	0,729581
-0,25	0,996645	1,095571	0,729581
-0,20	0,992655	1,097292	0,729581
-0,15	0,988664	1,099080	0,729581
-0,10	0,984674	1,100937	0,729581
-0,05	0,980683	1,102870	0,729581
0,00	1	1,000000	0,721901
0,05	1,000573	0,996629	0,722285

(d)

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,99537475	1,090052	0,720958
-0,25	0,99274675	1,090949	0,720958
-0,20	0,990119	1,091886	0,720958
-0,15	0,98749125	1,092868	0,720958
-0,10	0,9835285	1,092491	0,72416
-0,05	0,98374125	1,088339	0,7269
0,00	1	1	0,721901
0,05	1,00070267	1,000687	0,723416
0,10	1,00121	0,973099	0,704307

(e)

Para as outras quantidades de tarefas com descarte de classes, não foram gerados os gráficos com dados em dispersão, apenas as médias, que foram montadas em um mesmo gráfico, já apresentados nas Figuras 6.7 a 6.9. A fim de evitar uma apresentação exaustiva de resultados, a Tabela A3.12 mostra apenas a média dos resultados de *SCN4* combinando descartes em duas tarefas. A Tabela A3.13 mostra as médias dos resultados para descartes simultâneos de classes de dados de entrada em 3 tarefas simultaneamente, enquanto a Tabela A3.14 mostra os resultados para descarte simultâneo de classes de dados de entrada em todas as tarefas.

Tabela A3.12: Média dos resultados para descarte simultâneo em duas tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,0087263	1,072874	0,708063
-0,25	1,0042118	1,074417	0,708063
-0,20	0,999038	1,063246	0,709441
-0,15	0,9926612	1,076691	0,708248
-0,10	0,9889055	1,087435	0,72241
-0,05	0,9859055	1,091852	0,728891
0,00	1	1	0,721901
0,05	1,0013467	0,994964	0,719071
0,10	1,0014453	0,980696	0,707028

Tabela A3.13: Média dos resultados para descarte simultâneo em três tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,025068	1,07402	0,706541
-0,25	1,017879	1,076539	0,706541
-0,20	1,01069	1,079161	0,706541
-0,15	0,9964125	1,090159	0,708003
-0,10	0,992886	1,091159	0,72235
-0,05	0,9864668	1,0987	0,731081
0,00	1	1	0,721901
0,05	1,0019955	0,978694	0,706211
0,10	1,0031103	0,969068	0,705168

Tabela A3.14: Resultados para descarte simultâneo de todas as tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	1,002243	1,090432	0,71
-0,10	0,995533	1,105154	0,722
-0,05	0,988708	1,092212	0,729
0,00	1	1	0,721901
0,05	1,002233	0,97872	0,70351
0,10	1,006217	0,948475	0,706142

Vocoder-1, Q = 0,7

A configuração final para o cenário C3 obtido por *SCN3* para a aplicação Vocoder-1, Q = 0,7 definiu que até 4 tarefas podem fazer descarte de classes de dados de entrada. Assim, são possíveis 15 combinações de descarte de tarefas, desde uma até 4 tarefas descartando classes de dados de entrada ao mesmo tempo.

Para esta variação da aplicação Vocoder, serão experimentadas todas as combinações de tarefas com descarte de dados de entrada. As combinações usadas em cada situação seguem na Tabela A3.15. As tarefas que podem descartar classes de entrada são as tarefas 1, 8, 13 e 21.

Novamente, a fim de evitar uma apresentação exaustiva dos resultados obtidos, apresentaremos apenas as tabelas com as médias dos resultados sobre os quais foram traçados os gráficos apresentados na Figura 6.10 do capítulo 6. As Tabelas A3.16 a A3.18 mostram, respectivamente, as médias dos resultados de *SCN4* quando ocorrem descartes de classes de entrada em uma, duas e três tarefas. A Tabela A3.19 mostra os resultados para descarte simultâneo de classes de dados de entrada em todas as tarefas.

Tabela A3.15: Combinações de descartes usadas no Cenário 4 para Vocoder-1, Q = 0,7

Descarte	Combinações usadas
1 Tarefa	1, 8, 13, 21
2 Tarefas	1/8, 1/13, 1/21, 8/13, 8/21, 13/21
3 Tarefas	1/8/13, 1/8/21, 1/13/21, 8/13/21
4 Tarefas	1/8/13/21

Tabela A3.16: Média dos resultados para descarte em apenas uma tarefa.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	1,01457375	0,994999	0,721633
-0,25	1,01108525	0,996631	0,721633
-0,20	1,007597	0,998341	0,721633
-0,15	1,0041085	1,000137	0,721633
-0,10	1,00131075	1,018127	0,722071
-0,05	0,997749	1,011172	0,719142
0,00	1	1	0,721901
0,05	1,000376	1,005784	0,723982
0,10	1,001414	0,978082	0,704307

Tabela A3.17: Média dos resultados para descarte simultâneo em duas tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	1,00949	0,966377	0,709302
-0,20	1,01014	0,97004	0,703667
-0,15	1,00345117	0,973186	0,711362
-0,10	1,00373383	0,996024	0,716945
-0,05	1,00116733	1,009466	0,719148
0,00	1	1	0,721901
0,05	1,00082783	0,993044	0,71337
0,10	1,00134167	0,986434	0,707028

Tabela A3.18: Média dos resultados para descarte simultâneo em três tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	No Result		
-0,10	0,9984335	0,979169	0,7139
-0,05	1,001709	1,011365	0,720659
0,00	1	1	0,721901
0,05	1,001721	0,990167	0,70795
0,10	1,0017213	0,967457	0,705116

Tabela A3.19: Resultados para descarte simultâneo de todas as tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	No Result		
-0,25	No Result		
-0,20	No Result		
-0,15	No Result		
-0,10	0,995503	0,956924	0,706581
-0,05	0,995425	0,991713	0,71289
0,00	1	1	0,721901
0,05	1,001623	0,98827	0,703432
0,10	1,002468	0,940529	0,713575

A3.2.2 Cancelador de Eco Acústico

GMDFa-4, Q = 0,5

Foram também levantados resultados experimentais para a aplicação de cancelamento de eco acústico, em sua variação com maior quantidade de classes com possibilidade de descarte, GMDFA-4, Q = 0,5. A configuração final para o cenário C3 obtido por *SCN3* definiu que até 6 tarefas podem fazer descarte de classes de dados de entrada. Assim, são possíveis 34 combinações de descarte de tarefas, desde uma até 6 tarefas descartando classes de dados de entrada ao mesmo tempo.

Devido à grande quantidade de combinações possíveis, foram consideradas até 7 combinações para cada quantidade de tarefas com descarte, totalizando 34 experimentos. As combinações usadas em cada situação seguem na Tabela A3.20. As tarefas que podem descartar classes de entrada são as tarefas 1, 3, 4, 6, 15 e 23.

Tabela A3.20: Combinações de descartes usadas no Cenário 4 para GMDFa-4, Q = 0,5

Descarte	Combinações usadas
1 Tarefa	1, 3, 4, 6, 15, 23
2 Tarefas	1/3, 1/6, 3/4, 6/15, 15/23, 3/6, 4/23
3 Tarefas	1/3/4, 3/4/6, 1/6/23, 4/15/23, 1/15/23, 1/4/15, 3/6/23
4 Tarefas	1/3/4/6, 4/6/15/23, 1/3/15/23, 3/4/6/15, 3/4/15/23, 1/3/6/15, 1/4/6/23
5 Tarefas	1/3/4/6/15, 1/3/4/6/23, 1/3/4/15/23, 1/3/6/15/23, 1/4/6/15/23, 3/4/6/15/23
6 Tarefas	1/3/4/6/15/23

Como na aplicação anterior, serão apresentadas apenas as tabelas com as médias dos resultados sobre os quais foram traçados os gráficos apresentados na Figura 6.11 do capítulo 6. As Tabelas A3.21 a A3.25 mostram, respectivamente, as médias dos resultados de *SCN4* quando ocorrem descartes de classes de entrada em uma, duas, três, quatro e cinco tarefas. A Tabela A3.26 mostra a tabela com os resultados para descarte simultâneo de classes de dados de entrada em todas as tarefas.

Tabela A3.21: Média dos resultados para descarte em apenas uma tarefa.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,9645035	1,140557	0,513799
-0,25	0,96307467	1,139272	0,514213
-0,20	0,97207067	1,108237	0,512799
-0,15	0,9804935	1,085624	0,514025
-0,10	1,00468783	0,936244	0,51019
-0,05	1,00329517	0,966019	0,505069
0,00	1	1	0,51255
0,05	0,999704	1,03325	0,511647
0,10	0,997937	1,090679	0,502723
0,15	0,9961915	1,136689	0,502588

Tabela A3.22: Média dos resultados para descarte simultâneo em duas tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,9164573	1,274333	0,512269
-0,25	0,9332864	1,235906	0,513294
-0,20	0,9384337	1,230008	0,507472
-0,15	0,976704	1,055764	0,508931
-0,10	0,9870154	1,038389	0,507953
-0,05	1,0064437	0,933752	0,510744
0,00	1	1	0,51255
0,05	1,0011206	1,061754	0,508858
0,10	1,00056	1,144156	0,504479
0,15	1,0021388	1,150656	0,504401

Tabela A3.23: Média dos resultados para descarte simultâneo em três tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,89354329	1,355474	0,511247
-0,25	0,898946	1,34921	0,509447
-0,20	0,91279771	1,320625	0,509644
-0,15	0,94986514	1,10696	0,508604
-0,10	0,96849786	1,064274	0,507243
-0,05	0,99803657	0,96797	0,504989
0,00	1	1	0,51255
0,05	1,00239157	1,088041	0,50728
0,10	1,0031852	1,171954	0,508163
0,15	1,003551	1,143782	0,50526

Tabela A3.24: Média dos resultados para descarte simultâneo em quatro tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,88066114	1,319581	0,51065
-0,25	0,89638071	1,306926	0,507788
-0,20	0,90959214	1,286579	0,507843
-0,15	0,92931771	1,18895	0,508906
-0,10	0,944682	1,178673	0,508365
-0,05	0,98903843	0,99158	0,509921
0,00	1	1	0,51255
0,05	1,00455271	1,122934	0,504789
0,10	1,00642586	1,132579	0,50677
0,15	1,00525971	1,08778	0,509401

Tabela A3.25: Média dos resultados para descarte simultâneo em cinco tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,882062	1,345617	0,509992
-0,25	0,8877705	1,35259	0,505871
-0,20	0,89315983	1,360999	0,511219
-0,15	0,8959865	1,26878	0,510105
-0,10	0,92178067	1,214323	0,509049
-0,05	0,97507583	1,05368	0,50861
0,00	1	1	0,51255
0,05	1,00982933	1,157764	0,50629
0,10	1,00734217	1,087226	0,505329
0,15	1,00664	0,98958	0,509407

Tabela A3.26: Resultados para descarte simultâneo de todas as tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,30	0,890435	1,3543	0,51
-0,25	0,884757	1,354683	0,51
-0,20	0,879079	1,355079	0,51
-0,15	0,868783	1,35267	0,515712
-0,10	0,924865	1,155055	0,511
-0,05	0,978921	1,020708	0,52065
0,00	1	1	0,51255
0,05	1,01123	1,191624	0,512404
0,10	1,006823	1,057451	0,50223
0,15	1,008353	0,931649	0,50558

A3.2.3 TGFF

TGFF70-4, Q = 0,5

Assim como foi feito para aplicações reais, a aplicação TGFF70, em sua variação TGFF70, Q = 0,5, também foi experimentada no cenário C4. A configuração final para o cenário C3 obtido por *SCN3* definiu que até 5 tarefas podem fazer descarte de classes de dados de entrada. Assim, são possíveis 31 combinações de descarte de tarefas, desde uma até 5 tarefas descartando classes de dados de entrada ao mesmo tempo.

Para esta aplicação foi experimentada uma menor quantidade de combinações entre as tarefas que descartam classes de dados de entrada. As combinações usadas em cada situação seguem na Tabela A3.27. As tarefas que podem descartar classes de entrada são as tarefas 2, 12, 18, 20 e 24.

Serão apresentadas apenas as tabelas com as médias dos resultados sobre os quais foram traçados os gráficos apresentados na Figura 6.12 do capítulo 6. As Tabelas A3.28 a A3.30 mostram, respectivamente, as médias dos resultados de *SCN4* quando ocorrem descartes de classes de entrada em uma, duas ou três tarefas.

Tabela A3.27: Combinações de descartes usadas no Cenário 4 para TGFF70-4, Q = 0,5

Descarte	Combinações usadas
1 Tarefa	2, 12, 18, 20, 24
2 Tarefas	2/12, 18/20, 12/24, 2/18, 12/20
3 Tarefas	2/12/18, 2/20/24, 18/20/24, 12/18/20, 2/12/24

Tabela A3.28: Média dos resultados para descarte em apenas uma tarefa.

Delta	URC4/URC3	EC4/EC3	QoS
-0,26	0,9943294	1,009605	0,50384
-0,21	0,9936232	1,0098732	0,50384
-0,16	0,992561	1,0234388	0,504466
-0,11	0,992631	1,030096	0,502545
-0,06	0,9909664	1,029045	0,511039
0,00	1	1	0,500396
0,04	1,0009074	1,023404	0,51979

Tabela A3.29: Média dos resultados para descarte simultâneo em duas tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,26	0,9971182	0,955986	0,5013342
-0,21	0,9961418	0,957961	0,5063878
-0,16	0,9920048	1,017551	0,5091604
-0,11	0,9924242	1,002109	0,5057658
-0,06	0,990288	1,018038	0,5086916
0,00	1	1	0,500396
0,04	0,9999582	1,03215	0,515213

Tabela A3.30: Média dos resultados para descarte simultâneo em três tarefas.

Delta	URC4/URC3	EC4/EC3	QoS
-0,26	1,0004795	0,986313	0,501178
-0,21	0,9934313	0,932476	0,508858
-0,16	0,990932	0,991925	0,504486
-0,11	0,9914102	0,977393	0,50191
-0,06	0,991296	1,045947	0,509858
0,00	1	1	0,500396
0,04	1,0036888	0,9669	0,502149