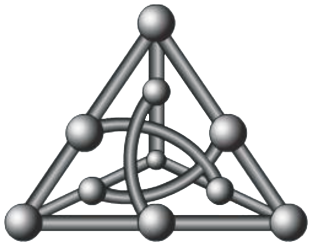
Dissertação de Mestrado

Exploração do Espaço de Projetos de Sistemas Multiprocessadores Guiado por *Dark Silicon*

Ana Caroline dos Santos Silva

Orientação: Prof. Dr. Ricardo Ribeiro dos Santos

Área de Concentração: Sistemas de Computação



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
17 de abril de 2017.

Exploração do Espaço de Projetos de Sistemas Multiprocessadores Guiado por *Dark Silicon*

Campo Grande, 17 de abril de 2017.

Banca Examinadora:

- Prof. Dr. Ricardo Ribeiro dos Santos (FACOM/UFMS) - orientador
- Profa. Dra. Sarita Mazzini Bruschi (ICMC/USP) - membro externo
- Prof. Dr. Edson Takashi Matsubara (FACOM/UFMS) - membro interno
- Profa. Dra. Liana Dessandre Duenha Garanhani - suplente

Resumo

O projeto de plataformas computacionais tem passado por mudanças significativas nos últimos anos, decorrentes, principalmente, de limitações físicas ocasionadas pelo incremento da corrente de fuga, em consequência da miniaturização dos transistores. Tais limitações geraram o que se convencionou chamar *dark silicon*, o qual se refere a porção de área do chip que deve ser desligada ou funcionar em frequência menor, por extrapolar as restrições de dissipação de potência. Para manter a viabilidade dos projetos, soluções que mitiguem o *dark silicon* de maneira eficaz têm sido estudadas, muitas das quais propõem a heterogeneidade de dispositivos de processamento. Entretanto, o aumento da diversidade de dispositivos e objetivos em um projeto o torna mais complexo, exigindo mecanismos automatizados de exploração de espaço de projeto cientes da nova realidade. Diante disto, este trabalho propõe uma solução capaz de explorar o espaço de projeto de plataformas computacionais na presença de *dark silicon* e oferecer alternativas arquiteturais para essas plataformas. Essa técnica toma como referência um modelo de otimização multiobjetivo e adota o algoritmo genético NSGA-II para resolução do problema. Esta técnica foi validada e avaliada a partir de resultados de um algoritmo de força bruta para o mesmo problema e está integrada junto ao fluxo de execução da ferramenta de projeto de plataformas *multicore* MultiExplorer. Com essa integração, um usuário da ferramenta MultiExplorer pode projetar e simular o desempenho de uma plataforma computacional, analisar parâmetros físicos e realizar a exploração do espaço de projeto buscando alternativas arquiteturais que atenuem a presença de *dark silicon*.

Abstract

The design of processor platforms comprised of multiple cores has been subject of dramatic changes in the last years. The changes are mainly by the physical constraints due to the leakage current increase in the transistor shrinking process. Currently, such constraints are mostly known as "dark silicon" that means the chip area which should be turned off or work on a reduced clock frequency to meet the power dissipation constraints. Since it is not viable to turn off a chip specific area, proposals to mitigate the dark silicon have been presented focused on using heterogeneous processing devices (different clock frequencies) on the chip. The design problem now changes to how to choose the devices (type and number) in order to meet all constraints and goals of the chip design. The goals, constraints, and the devices features make the design even more complex thus requiring electronic design automation tools focused on the dark silicon aware design space exploration problem. In this context, this work proposes a technique that performs the design space exploration aware of the dark silicon constraints and providing new architectural alternatives mitigating dark silicon in the chip. Our technique is built on the top of a multiobjective optimization model and it adopts the NSGA-II genetic algorithm to solve the problem. The technique has been validated and evaluated along with a brute force algorithm. In addition to the design, implementation and validation, we have integrated our proposal to the MultiExplorer platform design tool so that a user can design a computing platform, simulate the performance, analyze physical estimates, and perform design space exploration looking for alternative designs that mitigate the dark silicon.

Agradecimentos

Aos meus pais Edineti e Orivaldo que sempre acreditaram no poder da educação, priorizando e incentivando nossos estudos além de oferecer todo suporte e apoio para que isto fosse possível.

À minha irmã Hellen, por todo seu amor e carinho e por fazer de seu orgulho de irmã mais nova um estímulo para seguir em frente.

Ao meu namorado e futuro marido Alexandre, por todo amor, apoio e paciência, compreendendo minhas abdicações por conta dos estudos.

À minha tia Lenice por ser a minha principal referência de educadora e por me incentivar a seguir na área acadêmica, além, claro, de todo apoio e amor.

Ao Professor Ricardo que mais uma vez me honrou com a oportunidade ser sua "orientanda", compartilhando comigo parte de seu vasto saber que foi além do ensino de Ciência da Computação.

À Faculdade de Computação (FACOM) da UFMS, pela estrutura e pelo enorme acesso ao conhecimento, me ofertado por meio dos seus excelentes professores.

À Professora Sarita e ao professor Edson pelas contribuições de extrema relevância que fizeram ao projeto.

A toda a minha família, que sempre me ofereceu suporte emocional para seguir em frente.

Aos colegas do LSCAD, em especial aos do grupo de pesquisa *Dark Sicon* (Tony, Clara, Rafael, Luiz Augusto, Matheus, João e Professora Liana), por toda troca de experiências e colaboração no projeto.

Aos meus amigos que me incentivaram e ofereceram nos momentos de descontração desfrutados juntos, o equilíbrio necessário.

Aos colegas de curso, em especial a Glasiely, Edilson, Wesley, Phelipe e Thiago, que contribuíram com meu aprendizado e tornaram esta jornada mais divertida.

Aos meus colegas de trabalho que colaboraram para que fosse possível a conciliação entre o trabalho e o estudo.

À Fundect-MS pelo apoio financeiro concedido às pesquisas realizadas no LSCAD/FACOM/UFMS.

À Comissão de Aperfeiçoamento de Pessoal do Nível Superior (CAPES), pelo

financiamento do projeto.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AMP *Asymmetric Multicore Processors*

ARM *Advanced RISC Machine*

BTB *Branch Targed Buffer*

C-Cores *Conservation Cores*

CGRA *Coarse-Grained Reconfigurable Arrays*

CMOS *Complementary Metal Oxide Semiconductor*

CMP *Chip Multicore Processors*

CPU *Central Processing Unit*

C-SFU *Controlled Successive Frequency Unscaling*

DSE *Design Space Exploration*

DSU *Dark Silicon Utilization Efficiency*

DVFS *Dynamic Voltage and Frequency Scaling*

GA *Genetic Algorithms*

GPU *Graphics Processing Unit*

HPM *Hierarchical Power Management*

ILP *Instruction Level Parallelism*

IP *Intellectual Property*

ISA *Instruction Set Architecture*

ITRS *International Technology Roadmap for Semiconductors*

LLC *Last Level Cache*

LSCAD Laboratório de Sistemas Computacionais de Alto Desempenho

McPAT *Multicore Power, Area and Timing*

MIPS *Microprocessor without Interlocked Pipeline Stages*

MLP *Memory-Level Parallelism*

MOEA *Multiobejctive Evolucionary Algorithm*

MOGA *Multiobjective Genetic Algorithm*

MOOP *Multiobjective Optimization Problem*

MOSFET *Metal-Oxide-Semicondutor Field Effect Transistor*

MPSoC *Multi-Processor System on a Chip*

nMOS*negative Metal-Oxide-Semiconductor*

pMOS*positive Metal-Oxide-Semiconductor*

NoC *Network-on-a-chip*

NSGA-II *Nondominated Sorting Genetic Algorithm II*

NTV *Near-Threshould Voltage*

OoO *Out-of-Order*

QoS *Quality of Service*

ROB *Reorder Buffer*

ROI *Region of Interest*

RoO *Range of Optimality*

RSM *Response Surface Modeling*

SCE *Symbiotic Core Execution*

SFU *Successive Frequency Unscaling*

SoC *System-on-a-chip*

TDP *Thermal Dynamic Power*

TFET *Tunneling Field Effect Transistors*

THPH *Topologically Homogeneous Power-Performance Heterogeneous*

UDS U_{DS}*Utilization Dark Silicon*

ULA Unidade Lógica e Aritmética

VLSI *Very Large Scale Integration*

XML *eXtensible Markup Language*

Lista de Algoritmos

Capítulo 1

Introdução

Desde o advento do computador, a busca por incrementar a capacidade de processamento tem direcionado as pesquisas e a indústria relacionada à area. Este processamento tem sido medrado exponencialmente, norteado por décadas de aplicação da Lei de Moore [Schaller, 1997] e da escala de integração de Dennard [Dennard et al., 1974].

Não obstante a Lei de Moore manter seus preceitos até os dias atuais, para projetos de circuitos eletrônicos baseados em tecnologia *Complementary Metal Oxide Semiconductor* (CMOS) com transistores abaixo de $90nm$, a escala de integração de Dennard não pode ser aplicada [Dennard et al., 2007][Borkar, 1999][Borkar, 2009], pois, ao contrário do que se estimava, transistores abaixo dessa escala apresentam comportamento divergente quanto à corrente de fuga [Kim et al., 2003]. Devido a essa divergência, as técnicas utilizadas na área de projeto de dispositivos eletrônicos e, em particular, de processadores, têm passado por atualizações constantes, dentre essas, a utilização de novas restrições de consumo energético e dissipação térmica. Essas restrições também passaram a considerar que o processo de miniaturização de circuitos eletrônicos associado ao incremento da frequência de operação podem fazer com que parâmetros físicos do projeto do circuito como a densidade de potência dissipada não sejam mais respeitados, levando assim a considerar que parte da área do circuito eletrônico não seja utilizado em tempo de execução.

Considerando o projeto de chips processadores, este óbice ficou conhecido como *the utilization wall* (a barreira de utilização), que limita a utilização do processador em frequência máxima de operação. Nesse contexto, surgiu o termo *dark silicon* [Merritt, 2009] que se refere à área do chip que não pode funcionar na mesma frequência do restante do circuito eletrônico a fim de obedecer os limites energéticos para o projeto.

Como primeira resposta para esta objeção surgiram os projetos de processadores que utilizam múltiplos núcleos de processamento [Markoff, 2004]. A motivação para esse tipo de abordagem de projeto é explorar o paralelismo em alto nível, possibilitando assim que a divisão de tarefas simultâneas (paralelismo) de um programa possa prover desempenho mesmo utilizando um hardware com menor frequência de operação. Porém, já é bem conhecido que as restrições físicas dos transistores também limitam a evolução direta desse modelo de projeto. Diante disto, novas soluções cientes de *dark silicon* que propõem o uso de aceleradores de hardware, aumento de memória, dispositivos de processamento heterogêneo

e especialização dos dispositivos arquiteturais têm surgido com o intuito de administrar a vazão de potência e manter os limites energéticos do projeto para minimizar o *dark silicon*.

Entretanto, o aumento da diversidade de dispositivos e objetivos em um projeto de processador o torna mais complexo, exigindo mecanismos automatizados de exploração de espaço de projeto cientes de um novo regime de projeto que considera a existência do *dark silicon*. Nesse contexto, observa-se uma lacuna no que diz respeito ao emprego de metodologias, técnicas e ferramentas automatizadas para projeto de processadores modernos diante do *dark silicon*. Especialmente, verifica-se a ausência de técnicas que, em tempo de projeto, permitam explorar e avaliar a utilização de dispositivos de hardware heterogêneos objetivando a maximização do desempenho mas ciente das novas restrições físicas do projeto.

Diante do exposto, esta proposta de trabalho, em nível de mestrado, tem como objetivo o preenchimento desta lacuna por meio da proposta de algoritmos e softwares que automatizem o processo de exploração de projeto, considerando a existência de *dark silicon*, de modo a possibilitar a tomada de decisão sobre o incremento do desempenho e a utilização eficiente da área e demais parâmetros físicos do projeto.

Especificamente, este trabalho apresenta uma metodologia e infraestrutura de software capaz de aplicar técnicas para explorar esse espaço visando apontar novas alternativas arquiteturais em tempo de projeto de plataformas de processadores. Essas novas alternativas propiciam mitigar as limitações físicas existentes no projeto original (fornecido pelo usuário) impostas pelo *dark silicon*. A nova infraestrutura de software provida por este trabalho é denominada de componente DS-DSE (*Dark Silicon aware Design Space Exploration*) e foi adicionado ao fluxo de projeto da ferramenta MultiExplorer [Devigo et al., 2015]. Para validar a nova estratégia de exploração de espaço de projeto ciente de *dark silicon*, experimentos foram realizados considerando plataformas computacionais reais comercializadas pela indústria de semicondutores.

Diante do exposto, o presente trabalho encontra-se estruturado da seguinte forma: no Capítulo 2 serão abordados os conceitos que fundamentam este projeto, por meio da exposição dos antecedentes teóricos, o surgimento e os impactos do *dark silicon* e, por fim, alternativas que propõem mitigá-lo; o Capítulo 3 trata sobre o problema da exploração de espaço de projeto, apresentando seus conceitos principais, técnicas utilizadas e trabalhos relacionados ao tema; o Capítulo 4 apresenta a teoria e o desenvolvimento da proposta de exploração de espaço de projeto. O Capítulo 5 apresenta os experimentos projetados e realizados e discorre sobre os principais resultados obtidos. Finalmente, as conclusões finais do trabalho, contribuições e propostas de extensões futuras são apresentadas no Capítulo 6.

Capítulo 2

Referencial Teórico

Neste capítulo serão abordados os referenciais teóricos que contribuem para melhor compreensão dos assuntos envolvidos na proposta de trabalho. As seções a seguir apresentam as definições de transistores do tipo *Metal-Oxide-Semicondutor Field Effect Transistor* (MOSFET), antecedentes teóricos, o contexto da barreira de utilização de circuitos eletrônicos, o surgimento e impactos do *dark silicon*. Por fim, opções para mitigar o *dark silicon* a partir de alternativas arquiteturais.

2.1 Transistores

O transistor é um dispositivo utilizado em projetos de eletrônicos para funcionar, principalmente, como uma chave para habilitar ou desabilitar a passagem de corrente elétrica. O advento do transistor MOSFET [Lilienfeld, 1927], na década de 50, revolucionou o mercado de sistemas eletrônicos e, em particular, dos processadores, possibilitando que dispositivos menores e com menor consumo energético fossem criados, tornando o componente extremamente competitivo com relação aos outros (válvulas) disponíveis à época.

Para compreender a problemática apresentada no decorrer deste trabalho, faz-se necessária a compreensão dos aspectos físicos e funcionais do transistor, componente fundamental no projeto dos processadores atuais.

O transistor MOSFET é fabricado utilizando uma série de passos de processamento químico que envolve a oxidação de silício, seleção de dopantes, e corrosão de camadas de metais, isolantes e contatos. A operação dos transistores é controlada por campo elétrico, por isso são chamados de MOSFET, ou semicondutores de óxido metálico de efeito de campo, e pode possuir dois tipos: tipo p (*positive Metal-Oxide-Semiconductor* (pMOS)) e tipo n (*negative Metal-Oxide-Semiconductor* (nMOS)).

Cada transistor é constituído por uma pilha com a porta (*gate*) condutora, uma camada isolante de dióxido de silício (S_iO_2), do corpo de silício (S_i), também chamado de substrato e dois terminais dopados denominados fonte e dreno, separadas por uma distância L que consiste no comprimento do canal do transistor. A Figura 2.1 apresenta um corte transversal do transistor, onde é possível notar seus componentes principais.

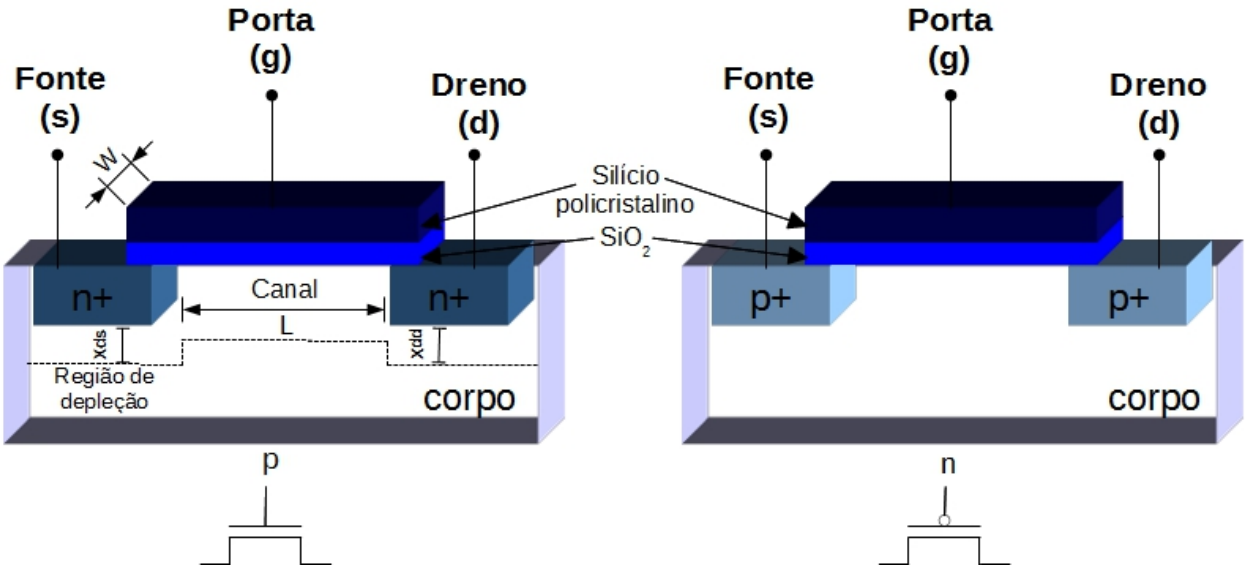


Figura 2.1: Transistor do tipo p e do tipo n.

A porta permite o fluxo de corrente entre os terminais fonte e o dreno. Em um transistor nMOS, o corpo é ligado ao terra para que as junções p-n da fonte e do dreno ao corpo sejam polarizados inversamente. Se a porta também estiver aterrada não há presença de corrente, assim o transistor está desligado. Se a tensão da porta é elevada, cria-se um campo elétrico que começa a atrair elétrons livres para a parte inferior da estrutura $Si - SiO_2$, chamada de canal. Se a tensão é elevada suficientemente, os elétrons penetram na camada de óxido e formam um canal invertido para atuar como semicondutor do tipo n. Portanto, os elétrons podem ser transportados da fonte para o dreno e a corrente flui. Nessa situação o transistor está ligado [West and Harris, 2011] e, desta forma, pode-se obter dois estados no transistor, com ou sem corrente entre a fonte e o dreno, o que na lógica binária corresponde a 0 ou 1.

Como o funcionamento do transistor depende da tensão aplicada aos seus terminais, pode-se ter três modos de operação para o transistor nMOS [Coelho, 2009, Colletta, 2015]:

Região de corte: A tensão da porta para a fonte (V_{gs}) é menor que a tensão limiar do transistor (V_{th}[1]). Neste caso, o transistor permanece desligado e não há condução entre o dreno e fonte.

Região linear: Se $V_{gs} < V_{th}$ e $V_{ds} < V_{gs} - V_{th}$, sendo V_{ds} a tensão entre o dreno e a fonte. Nesta circunstância, o transistor é ligado e o canal que é criado possibilita o fluxo de corrente entre o dreno e a fonte, controlado pela tensão na porta (Equação 2.1).

$$I_{ds} = K_{pn}\frac{W}{L}[(V_{gs} - V_{th})V_{ds} - \frac{V_{ds}^2}{2}] \tag{2.1}$$

Em que K_{pn} (Equação 2.2) é uma constante conhecida como parâmetro de transcondutância, definida pela tecnologia do processo, através do produto entre a mobilidade

[1] V_{th} corresponde à diferença de potencial mínima (tensão limiar) entre a porta e a fonte que possibilita a criação de um canal de condução entre fonte e dreno.

dos elétrons no canal (μ_n) e a capacitância por unidade de área de porta (C_{ox}). W é a largura e L é o comprimento do canal. V_{ds} é a tensão entre dreno e fonte.

$$K_{pn} = \mu_n C_{ox} \tag{2.2}$$

Região de saturação: Quando $V_{gs} > V_{th}$ e $V_{ds} > V_{gs} - V_{th}$. Nesta situação o transistor fica ligado e o canal criado possibilita o fluxo de corrente entre o dreno e a fonte, porém, como a tensão do dreno V_{ds} é maior do que a tensão na porta V_{gs} parte do canal é desligada, chamada de *pinch-off*. Assim, a corrente do dreno se torna relativamente independente da tensão do dreno e é controlada apenas pela tensão da porta, conforme Equação 2.3.

$$I_{ds} = \frac{1}{2} K_{pn} \frac{W}{L} (V_{gs} - V_{th})^2 \tag{2.3}$$

Em um transistor pMOS o funcionamento é o oposto. O corpo do tipo **n** está atrelado a um alto potencial, de modo que as junções com as fontes do tipo **p** e dreno estejam, normalmente, polarizadas inversamente. Quando a porta possui também alta diferença de potencial, nenhuma corrente flui entre dreno e fonte. Quando a tensão da porta é reduzida por um limiar de V_{th}, os elétrons são atraídos para formar um canal do tipo **p** imediatamente abaixo da porta, permitindo que a corrente possa fluir entre o dreno e a fonte [West and Harris, 2011].

A constante redução dos transistores que diminui, consequentemente, o comprimento do canal (L) e possibilita tanto o aumento da velocidade de operação do dispositivo, quanto a quantidade de dispositivos por área de um circuito integrado, também dá origem a sequelas indesejáveis denominadas efeitos de canal curto. O canal de um transistor é considerado curto quando seu comprimento (L) possui a mesma ordem de grandeza das camadas de depleção das regiões de fonte e dreno (X_{ds} e X_{dd}).

Dentre os principais efeitos indesejáveis causados por esta condição, pode-se citar a modulação de comprimento de canal, em que ocorre um estrangulamento do canal na região próxima ao dreno ΔL, fazendo com que o comprimento percorrido pela corrente não seja efetivamente L e sim $L_e = L - \Delta L$, conforme apresentado na Figura 2.2. Uma vez que ΔL expande com o aumento da tensão do dreno (V_{ds}), tem-se um consequente aumento da corrente do canal, representado pela inclusão de $(1 + \lambda V_{ds})$ (Equação 2.4).

$$I_{ds} = \frac{1}{2} K_{pn} \frac{W}{L} (V_{gs} - V_{th})^2 (1 + \lambda V_{ds}) \tag{2.4}$$

Em que λ representa parâmetro do processo de fabricação e λV_{ds} é diretamente proporcional a ΔL.

Além disso, o aumento de V_{ds} aplicado na região de dreno, provoca uma diminuição da barreira de potencial da junção canal-fonte (*DIBL - Drain Induced Barrier Lowering*), ocasionando um aumento exponencial da corrente do dreno (I_{ds}) fazendo com que uma menor tensão de porta seja necessária para ativar o dispositivo, ocasionando uma redução consequente de V_{th}.

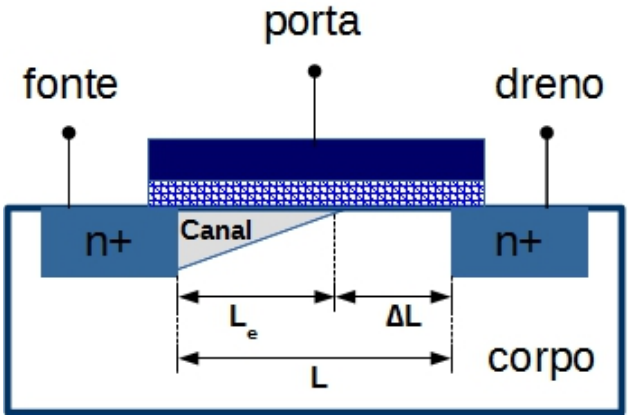


Figura 2.2: Comprimento efetivo do canal do transistor.

Outra consequência que está relacionada à redução de V_{th} é o aumento da corrente de fuga (*leakage*), que ocorre mesmo quando o transistor está desligado e pode ocorrer devido a condução subliminar entre a fonte e o dreno, vazamento de corrente da porta para o dreno e vazamento na junção da fonte para o corpo e do dreno para o corpo.

A condução sublimiar ocorre pela emissão de calor dos transportadores ao longo da barreira de potência definida pelo limites. O vazamento da porta é um efeito da mecânica quântica causada pelo tunelamento através da porta dielétrica extremamente fina. O vazamento de junção é causada pela corrente através da junção p-n entre as difusões da fonte/dreno e o corpo [West and Harris, 2011].

Nos processos tecnológicos acima de 180nm, a corrente de fuga possui valores insignificantes, exceto em aplicações de potência muito baixa. Em processos entre 90 e 65nm, a tensão limiar reduziu de forma tão significativa que os vazamentos subliminares chegaram a níveis significativos para a determinação da potência do transistor quando se trata de sistemas com milhões ou bilhões de transistores [West and Harris, 2011].

2.1.1 Potência Dinâmica e Estática

A potência instantânea consumida ou fornecida por elementos de um circuito é o produto da corrente pela tensão desse elemento. A dissipação de potência em circuitos CMOS, formados por transistores tipo p e tipo n, pode acontecer de duas formas: dinâmica e estática.

A dissipação de potência dinâmica ocorre devido a corrente gerada durante o chaveamento da porta e às capacitâncias externas e internas (cargas e descargas). Desta forma, se considerarmos uma operação dinâmica com frequência f, temos que a potência dinâmica pode ser calculada conforme a equação 2.5:

$$P_{dinamica} = \alpha C V_{DD}^2 f \tag{2.5}$$

Onde C representa a capacitância do circuito, V_{DD}^2 é a tensão de alimentação e α se refere ao fator de atividade, sendo ditado pela probabilidade de ocorrência de mudança de estado (0 ou 1). Desta forma, um ciclo de *clock* possui $\alpha = 1$ porque ocorre a transição completa

de subida e descida a cada ciclo. A maioria dos dados tem um fator de atividade de 0,5 no máximo, pois passa apenas uma vez cada ciclo. Dados aleatórios têm fator de atividade de 0,25 pois ocorre ciclo sim, ciclo não. Além disso, cumpre consignar que como a frequência está inversamente relacionada com o tempo de propagação, quanto maior a frequência, maior a dissipação de potência.

Já a dissipação de potência estática se dá quando não há variações dos sinais aplicados nas entradas. Desta forma, ela é determinada pela corrente de fuga e tensão de alimentação, logo, a potência estática pode ser obtida pela equação 2.6:

$$P_{estatica} = I_{fuga} V_{DD} \tag{2.6}$$

Por fim, a potência total de operação é dada pela soma de potência dinâmica e estática conforme equação 2.7.

$$P_{total} = P_{dinamica} + P_{estatica} \tag{2.7}$$

2.2 Transistores e Fator de Escala

Desde o advento do transistor, preceitos, tais quais a Lei de Moore, ditaram a indústria de processadores almejando o crescimento da capacidade de processamento dos transistores de silício e chips. Em 1965, Gordon Moore [Schaller, 1997] expôs que o desenvolvimento da indústria microeletrônica se daria com a integração dos circuitos eletrônicos e a miniaturização dos transistores, e estimou ainda, por meio de análise de curva de custos, que a quantidade de transistores em um chip dobraria a cada 18 meses. A estimativa de Moore só foi fundamentada na década seguinte, com a escala de integração de Dennard [Dennard et al., 1974] que demonstrou a concepção, fabricação e caracterização de transistores MOSFET com dimensões na ordem de 1μ, mostrando como estes dispositivos podem ser reduzidos, preservando suas características e, ainda, com desempenho melhorado.

Como referência para o estudo e entendimento do processo de miniaturização de transistores e circuitos eletrônicos utiliza-se como parâmetro a métrica do fator de escala tecnológica S. Essa métrica indica o fator de miniaturização entre dispositivos eletrônicos construídos com processos de fabricação (tamanho do transistor) distintos. Por exemplo, para mudança da tecnologia de $180nm$ para $90nm$ a escala S é definida por $\frac{180}{90} = 2$, portanto, $S = 2$. A Tabela 2.1 mostra o comportamento de vários parâmetros existentes em um dispositivo eletrônico constituído a partir de transistores, caracterizados em função de S seguindo os princípios da escala de integração de Dennard.

Observa-se que substituindo os valores de variáveis físicas como dimensões, concentração de dopantes e polarização em função do fator S, obtém-se que a potência dissipada $(V * I)$ sofre um escalonamento de $\frac{1}{S^2}$ e a densidade de potência não sofre escalonamento. Como as dimensões são escaladas com o fator $\frac{1}{S}$, substituindo-se na fórmula de capacitância, obtém-se que ela também é escalada pelo fator $\frac{1}{S}$. Com a redução das variáveis de capacitância, tensão e corrente elétrica pelo fator S, o tempo de atraso é escalado com $\frac{1}{S}$, fazendo com que a velocidade de chaveamento se torne maior.

Tabela 2.1: Comportamento do transistor de acordo com a escala de Dennard [Dennard et al., 1974].

Parâmetros do dispositivo ou circuito	**Parâmetro**	**Fator de Escala**
Dimensões do dispositivo	t_{ox}, L, W	$1/S$
Concentração de dopante	N_a	S
Tensão limiar	V_{th}	$1/S$
Tensão de alimentação	$V_{DD} \approx V_{th} \times 3$	$1/S$
Corrente	I	$1/S$
Quantidade de dispositivos	Q	S^2
Frequência	F	S
Capacitância	C	$1/S$
Potência	$P = QFCV_{DD}^2$	1
Tempo de atraso por circuito	VC/I	$1/S$
Área	$A = WL$	$1/S^2$
Dissipação de potência por circuito	VI	$1/S^2$
Densidade de potência	VI/A	1
Utilização	$1/P$	1

A escala de integração de Dennard estima não apenas a redução das dimensões dos dispositivos entre processos tecnológicos, proporcionando um número maior de transistores por área, mas também preserva as características de densidade de potência, evitando problemas de temperatura, além do aumento de desempenho computacional.

Os preceitos da Lei de Moore e a escala de integração de Dennard conduziram a indústria de processadores nas últimas décadas, resultando em aumento de desempenho medrado exponencialmente. Não obstante a Lei de Moore manter seus preceitos até os dias atuais, é sabido que para projetos de processadores, com transistores abaixo de $90nm$, a escala de integração de Dennard não pode ser aplicada, pois, ao contrário do que se estimava, transistores abaixo dessa escala apresentam comportamento divergente do previsto por ele [Dennard et al., 2007].

A miniaturização dos transistores ainda é possível, porém, limites físicos, tais como o consumo energético e a dissipação de calor, restringem o projeto de processadores modernos, gerados principalmente pela corrente de fuga decorrente da miniaturização. Este óbice é conhecido como *the utilization wall* (a barreira de utilização) [Venkatesh et al., 2010]. A Tabela 2.2 mostra o comportamento das propriedades do transistor de acordo com a escala limitada por corrente de fuga [Goulding-Hotta et al., 2011], baseado no mesmo fator S, utilizado na escala de integração Dennard.

Isso significa que como a quantidade de transistores (Q) escala em S^2 e a frequência (F) escala em S, tem-se um aumento de desempenho teórico de S^3. Se, por exemplo, considerar $S = 1,4$ ($S = \frac{65}{45}$), o aumento geral de desempenho ($Q * F$) será de $2,8\times$.

Porém, acontece que, com a barreira de utilização não é possível reduzir a tensão limiar em $\frac{1}{S}$, pois gera corrente de fuga em limites inaceitáveis. Então, para não extrapolar os limites de potência ($Q * F * C * Vdd^2$), apenas a capacitância pode ser reduzida pela escala $\frac{1}{S}$, o que gera um déficit de utilização de S^2 por geração. Considerando, novamente como exemplo, $S = 1,4$, o déficit será de $2\times$, conforme Figura 2.3 [Goulding et al., 2010]. Este déficit

Tabela 2.2: Comportamento do transistor de acordo com a escala limitada por corrente de fuga [Allred, 2013].

Parâmetros do dispositivo ou circuito	**Parâmetro**	**Fator de Escala**
Tensão limiar	V_{th}	1
Tensão de alimentação	$V_{DD} \approx V_{th} \times 3$	1
Quantidade de dispositivos	Q	S^2
Frequência	F	S
Capacitância	C	$1/S$
Potência	$P = QFCV_{DD}^2$	S^2
Dissipação de potência por circuito	VI	$1/S$
Densidade de potência	VI/A	S
Utilização	$1/P$	$1/S^2$

gerado pela barreira de utilização, implica em limitação da exploração da área do chip (*die*), resultando em áreas do circuito integrado que não podem ser utilizadas na mesma frequência máxima de operação, por excederem as restrições de dissipação de potência do chip. Estas áreas são denominadas *dark silicon* [Hardavellas et al., 1999, Goulding-Hotta et al., 2011].

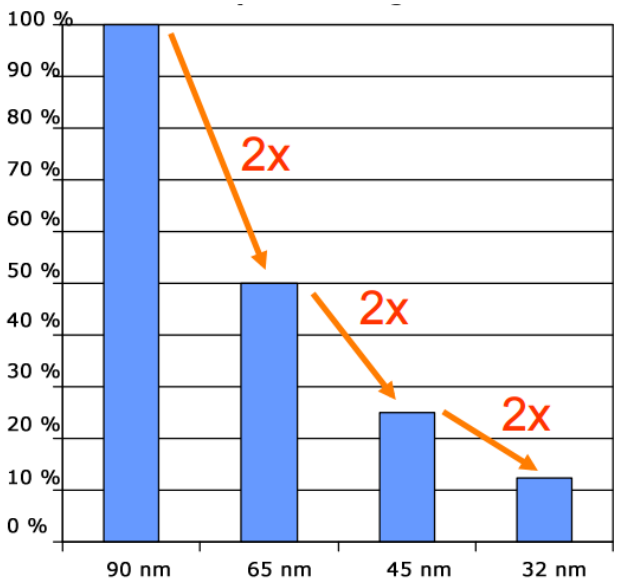


Figura 2.3: Déficit por geração [Goulding et al., 2010].

2.3 *Dark Silicon*

Devido à quebra da escala de Dennard, o percentual da área de um chip que pode funcionar em frequência integral tem caído exponencialmente a cada geração, o que força os projetos modernos de processadores a inutilizar ou utilizar em frequência menor parte considerável do chip (área em *dark silicon*). Esta fração do chip que deve ser desligado é denominado *dark silicon*. Estima-se que áreas de *dark silicon* possam chegar a 75%-85%

em $8nm$, em relação aos níveis de $45nm$ [Esmaeilzadeh et al., 2012, Borkar, 1999, 2009] (Figura 2.4).

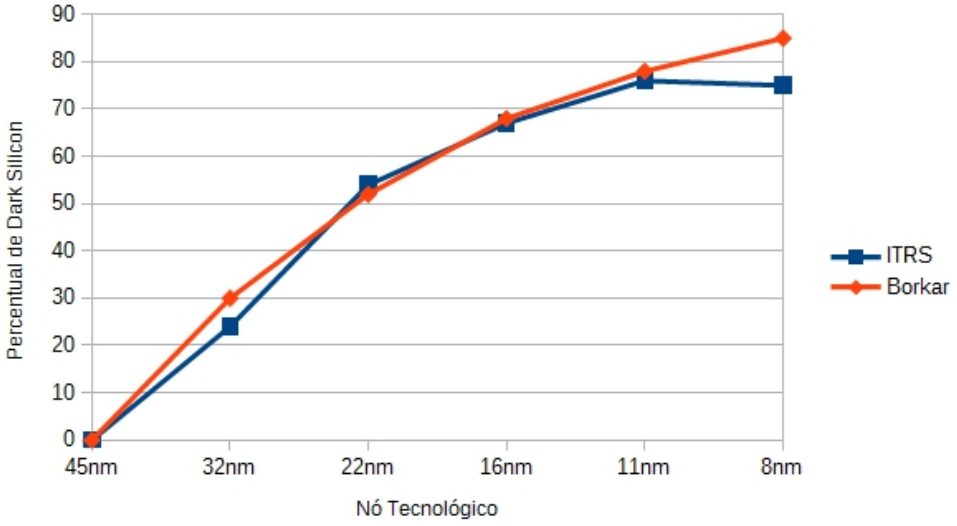


Figura 2.4: Níveis de *dark silicon* estimados com base em projeções de dimensionamento atual [Esmaeilzadeh et al., 2012].

Diante desta quebra de paradigma, o projeto de processadores modernos teve que ser modificado, e como primeira alternativa para o aumento da capacidade de processamento surgiram os processadores com múltiplos núcleos sobre um chip - *Multi-Processor System on a Chip* (MPSoC). Porém, esta abordagem não resolve os limites físicos gerados pela barreira de utilização.

Percebe-se que mesmo com grandes frações de *dark silicon*, os projetistas têm preferido optar por esta abordagem, por ser menos onerosa em relação a diminuição da potência, o que impulsiona o surgimento de técnicas que propõem utilizar eficientemente a área para obter eficiência energética.

Ante este novo paradigma de projeto, o desafio para processadores modernos consiste em explorar novas metodologias para identificar e utilizar o *dark silicon*. Com isso, novas abordagens para a utilização do *dark silicon* têm sido estudadas e projetos conscientes de *dark silicon* têm ganhado espaço [Allred, 2013].

Taylor [Taylor, 2012], intitulou essa nova era como "apocalipse do *dark silicon*" e propôs quatro abordagens para o problema:

1. **Reduzir o tamanho dos chips:** Apesar de parecer uma solução viável, a redução do tamanho dos chips é uma solução pessimista, pois com o encolhimento, áreas de *dark silicon* podem estar deixando de ser aproveitadas e seus custos não serão reduzidos exponencialmente a longo prazo, vez que custos com encapsulamento, testes, suporte, e outros não serão reduzidos, o que poderia gerar produtos com baixa lucratividade, cujas gerações posteriores não poderão se beneficiar com a Lei de Moore. Outro fator importante a ser considerado com a redução dos chips, refere-se à densidade de potência, que aumenta exponencialmente com a redução da área, e forçaria a

diminuição da *Thermal Dynamic Power* (TDP) [Huang et al., 2008], que por sua vez, limitaria a redução do tamanho dos chips.

2. **Utilizar a área em *dark silicon*:** Esta alternativa propõe o aproveitamento da área em *dark silicon* para computação de propósito geral, porém, operando em frequências menores ou sendo utilizados menos frequentemente, para isto pode se fazer uso de processadores que adotam limite de tensão reduzida - *Near-Threshould Voltage* (NTV), aumentar a área de cache, empregar arquiteturas baseadas em *arrays* reconfiguráveis de granularidade grossa - *Coarse-Grained Reconfigurable Arrays* (CGRA) ou utilizar técnicas de *sprinting* [Raghavan et al., 2012] e *turbo boost* [Naveh et al., 2011] computacionais para impulso computacional de curta duração.

3. **Especializar recursos de processamento:** Indigita o emprego do *dark silicon* para computação de propósito específico. Esta abordagem então alternaria entre a execução da computação de propósito geral e a de propósito específico, quando cada uma for mais eficiente, enquanto isso, a lógica ociosa é mantida em frequência reduzida. O problema da especialização de recursos consiste nas eventuais incompatibilidades entre hardware e software, além da superespecialização de processadores e obsolência do hardware quando da atualização do software.

4. **Resolução inverossímil:** Também podendo ser expressa pela frase em latim "Deus Ex Machina", que alude a recurso dramatúrgico que consistia originariamente na descida em cena de um deus cuja missão era dar uma solução arbitrária a um impasse vivido pelos personagens. Para o problema de *dark silicon*, esta resolução inverossímil seria a criação de um outro tipo de dispositivo eletrônico capaz de substituir os transistores ou da recriação dos mesmos com material diverso do MOSFET.

Percebe-se, que das abordagens propostas por Taylor apenas a última poderia realmente perfazer o *dark silicon*. Simplesmente reduzir a área do *die* para evitar o *dark silicon* traria consequências indesejáveis, principalmente ao que tange os benefícios da Lei de Moore. Porém, a heterogeneidade e especialização podem ser respostas eficazes para mitigar o *dark silicon*. Diante disto, um dos desafios, com as restrições apresentadas é de identificar e estimar o percentual de *dark silicon* em um determinado projeto de processador, para que, então, soluções viáveis para mitigar o *dark silicon* sejam estudadas. Desta forma, a seguir serão apresentadas propostas que buscam de diferentes maneiras atenuar o *dark silicon*.

2.4 Alternativas para Mitigar o *Dark Silicon*

Diante das restrições de projeto de processadores modernos, várias linhas de pesquisa com projetos cientes de *dark silicon* têm surgido e dentre as abordagens propostas se encontram em evidencia a heterogeneidade e especialização dos dispositivos arquiteturais, que buscam mitigar eficazmente o *dark silicon*, com foco na eficiência energética para contornar as limitações da barreira de utilização.

2.4.1 Núcleos Especializados

Ante a nova situação a ser considerada nos projetos de processadores modernos (balancear desempenho para manter orçamentos de área e dissipação de potência), uma das abordagens que têm ganhado espaço são os núcleos especializados, que divergem das unidades especializadas de hardware, pois têm seu foco na eficiência energética, ao invés de aumento de desempenho. Estas propostas são ótimas candidatas para situações de trechos de código irregulares e com pouco paralelismo.

Neste contexto, a proposta do *Conservation Cores* (C-Cores) [Venkatesh et al., 2010] é voltada para projetos de processadores especializados com foco em eficiência energética, baseado na ideia de que processadores energeticamente eficientes e especializados podem aumentar o paralelismo, pois reduzem os requisitos de energia por processamento e permitem mais processamento paralelo pelo mesmo custo energético. Os C-Cores são fortes candidatos para códigos irregulares e com pouco paralelismo.

No trabalho de Venkatesh et al. [Venkatesh et al., 2010], tanto uma arquitetura de sistema que incorpora os C-Cores, quanto ferramentas para criá-los e compilá-los automaticamente são descritas em detalhes. De maneira sucinta, na Figura 2.5, o conjunto de ferramentas extrai automaticamente os *kernels*-chave em uma determinada base de código e utiliza uma infraestrutura para gerar implementações de 45*nm* do C-Cores. O compilador converte em uma descrição dos C-Cores disponíveis em um chip e gera um código para utilizá-los.

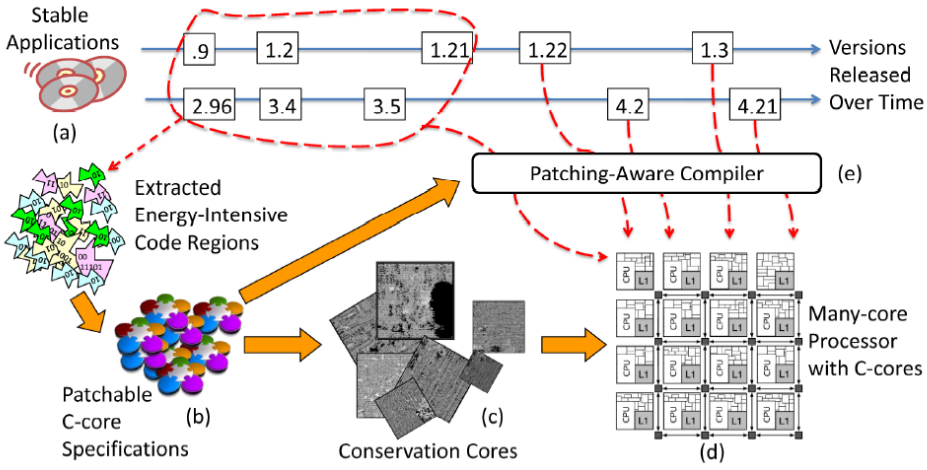


Figura 2.5: Síntese automática e compilação dos C-cores [Venkatesh et al., 2010].

A Figura 2.6 mostra a arquitetura de alto nível de um sistema C-Cores no qual: (a) mostra que a arquitetura é constituída por vários núcleos individuais; (b) cada um dos quais contém vários C-Cores, um núcleo de processamento de propósito geral e caches; (c) onde os C-Cores se comunicam com o restante do sistema através um sistema de memória coerente e uma interface simples *scan-chain-based*.

Durante a avaliação experimental dos C-Cores obteve-se redução de até 47% do consumo energético. O projeto dos C-Cores foi utilizado para geração de arquitetura para dispositivos móveis com sistema operacional Android e foi batizado como *Greendroid* [Goulding-Hotta et al., 2011]. O objetivo do projeto Greendroid é de preencher a área em *dark silicon*

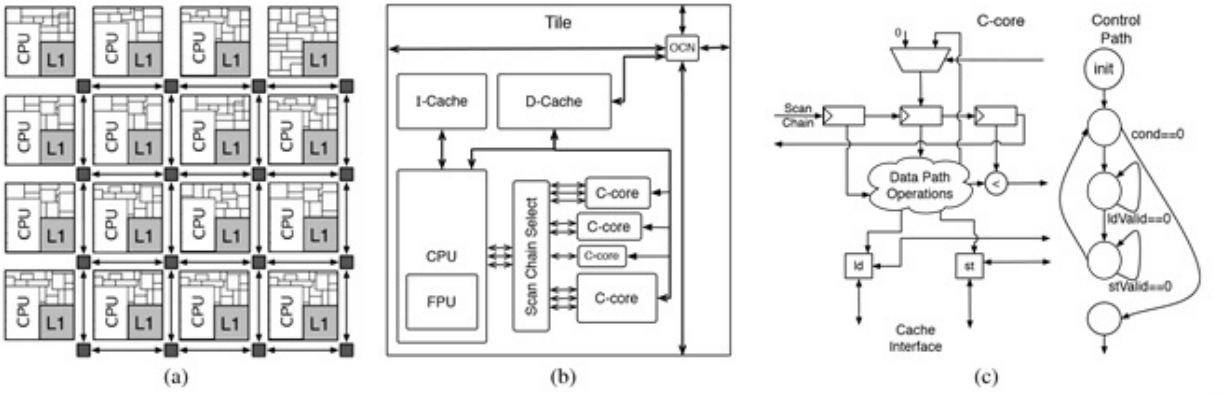


Figura 2.6: Arquitetura C-Cores [Venkatesh et al., 2010].

com centenas de núcleos especializados C-Cores com tecnologia de fabricação de 32nm, gerados automaticamente a partir da base de código que o processador pretende executar. A abordagem *Greendroid* pode economizar 11× mais energia, em comparação com um processador de propósito geral.

Além da utilização de C-Cores para construção de multiprocessadores para dispositivos móveis, no trabalho de Goulding-Hotta et al.[Goulding-Hotta et al., 2012], foram realizados uma série de experimentos para confirmar barreira de utilização. Para isto, foram sintetizados chips de $40mm^2$, com tecnologias de $90nm$ e $45nm$, preenchidos com Unidade Lógica e Aritmética (ULA)s de 32 bits. Executando o chip de $90nm$ na frequência operacional nativa destas ULAs, verificou-se que apenas 5% do chip poderia ser executado na frequência máxima em um orçamento de alimentação de $3W$, típico de dispositivos móveis. Em $45nm$, esta quantidade diminuiu para 1,8%, um fator de $2,8\times$. Usando projeções do *International Technology Roadmap for Semiconductors* (ITRS), um chip de $32nm$ reduziria esta área de utilização para 0,9%. Resultados semelhantes foram obtidos para plataformas do tipo desktop com $200mm^2$ de área e um orçamento de energia $80W$, confirmando mais uma vez que novas abordagem capazes de atenuar os efeitos da barreira de utilização são de extrema importância em projetos de processadores modernos.

2.4.2 Redução da Frequência

Sohail et al. [Sohail et al., 2011] propõem um caminho evolutivo para multiprocessadores, denominado frequência não-escalável sucessiva - *Successive Frequency Unscaling* (SFU) que continua alimentado significativamente mais núcleos, na mesma frequência de *clock*, e que são sucessivamente reduzidas a cada geração para manter o orçamento de energia. O número maior de núcleos ativos aumenta o paralelismo em nível de memória, compensando o *clock* mais lento e superando o desempenho com *dark silicon*. Este resultado é obtido através da observação analítica de que na presença de latência de memória, dado um orçamento energético, é mais viável manter um número maior de núcleos funcionando com uma frequência menor do que um número menor de núcleos com frequência maior, pois com o aumento paralelismo sobrepõe a latência de memória.

Para isso, o SFU é empregado em dois contextos com diferentes métricas de desempenho. No primeiro contexto de cargas de trabalho, no qual o tempo de execução da tarefa é a única

métrica, emprega-se o SFU completo, que diminui a frequência de todos os núcleos. Em outro contexto, com cargas de trabalho empresariais (por exemplo, processamento de transações *online*), onde tanto a produção quanto latência de resposta importam, a frequência mais lenta do SFU iria degradar o desempenho e, portanto, aumentariam a latência de resposta. Assim, é empregada a frequência não-escalável sucessiva controlada - *Controlled Successive Frequency Unscaling* (C-SFU) que moderadamente retarda o *clock* e potência de vários núcleos, mas não todos.

Como resultado dos experimentos realizados, com tecnologia de fabricação de $11nm$, obteve-se desempenho do SFU 81% melhor do que com *dark silicon*, já o C-SFU apresentou *throughput* 29% melhor do que a abordagem com *dark silicon*.

2.4.3 Heterogeneidade de Dispositivos

Devido às restrições impostas pela barreira de utilização, novos materiais e dispositivos têm sido estudados para a substituição dos tradicionais transistores CMOS. Porém, diversos dispositivos, apesar de diminuírem a corrente de fuga, apresentam longos retardos. Por isso, multiprocessadores com chips heterogêneos têm se apresentado como uma solução.

Apesar da heterogeneidade arquitetural apresentar melhoras significativas relacionadas a eficiência energética e desempenho, quando comparados com um processador homogêneo, um dos desafios desta abordagem reside na dificuldade em explorar a troca entre os núcleos implementados com diferentes tecnologias.

No trabalho de Swaminathan et. al [Swaminathan et al., 2013] é descrita uma arquitetura que utiliza núcleos compostos por dispositivos CMOS e transistores de tunelamento controlado por efeito de campo - *Tunneling Field Effect Transistors* (TFET). O dispositivo TFET utiliza efeitos de tunelamento quântico, propiciando dispositivos com inclinação sublimiar inferior a $60mV/década$ à temperatura ambiente, apresentando comportamento próximo ao de uma chave ideal, permitindo uma tensão de alimentação menor. Apesar dos dispositivos TFET não sofrerem com os problemas energéticos dos dispositivos CMOS, eles não conseguem atingir desempenho equivalente ao CMOS. Portanto, a abordagem heterogênea que compreenda núcleos CMOS (úteis para acelerar códigos sequenciais ou não escaláveis) e vários núcleos TFET (que são otimizados para funcionar de forma eficiente com baixas voltagens para atender a cargas de trabalho altamente paralelas) pode minimizar os problemas com as restrições energéticas com aumento de desempenho.

A Figura 2.7 mostra multiprocessadores CMOS homogêneos (à esquerda) e CMOS-TFET heterogêneos (à direita) operando em configurações de *dark silicon* e *dim silicon*. Os gráficos (centro) mostram a *frequência* × *número de núcleos*, e *frequência* × *potência* para os dois tipos de multiprocessadores. Em um cenário de *dark silicon* (menos núcleos com maior tensão), o multiprocessador heterogêneo pode igualar o desempenho com multiprocessadores homogêneos, desde que ele contenha núcleos CMOS suficientes (1 versus 4). Em um ambiente de *dim silicon* (mais núcleos com tensão menor), núcleos heterogêneos podem superar os homogêneos, quer usando o mesmo número de núcleos TFET a uma frequência mais elevada (2 versus 5) ou pela utilização de mais núcleos TFET na mesma frequência (2 versus 6). Além disso o *dim silicon* do multiprocessador CMOS pode habilitar mais núcleos a serem ligados,

mas obriga estes núcleos a operarem em frequências extremamente baixas, prejudicando o desempenho.

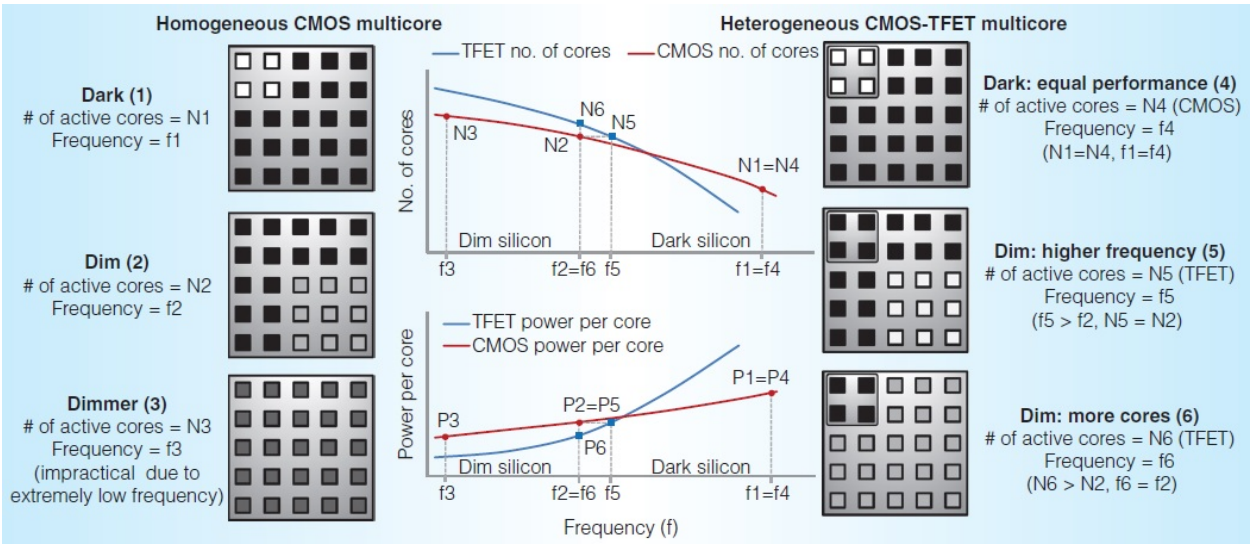


Figura 2.7: Configurações de multiprocessadores CMOS homogêneos e multiprocessadores CMOS-TFET heterogêneos [Swaminathan et al., 2013].

Nesta abordagem, dado um aplicativo para ser executado em um multiprocessador heterogêneo, podem-se considerar dois esquemas de mapeamento das *threads* por núcleo possíveis: (a) utilizando de forma homogênea apenas um tipo de núcleo (CMOS ou TFET, exclusivamente), e (b) utilizando os dois tipos de núcleos simultaneamente (mapeamento heterogêneo). No mapeamento homogêneo, núcleos que não estão sendo utilizados são mantidos em *dark silicon*, permitindo que os núcleos ativos possam usar todo o orçamento energético. No mapeamento heterogêneo todos os núcleos vão dividir o orçamento total de energia disponível, e as *threads* do aplicativo serão mapeadas para ambos tipos de núcleos, CMOS e TFET. Como núcleos de diferentes tipos de dispositivos têm características V/f distintas, esses núcleos serão executados em pontos de operação divergentes, embora cada núcleo possua orçamento energético igual. Isto irá resultar em diferentes tipos de núcleos tendo desempenho desigual. Por isso, foi utilizado um balanceamento de carga dinâmica para evitar eventuais ineficiências que poderiam surgir devido ao particionamento do trabalho igual entre segmentos de aplicação. Depois do balanceamento de carga, o tipo de núcleo que opera com maior eficiência energética completaria mais trabalho no mesmo período de tempo, pois todos os núcleos têm o mesmo orçamento energético, neste caso, reparticionar a potência disponível através de núcleos pode melhorar ainda mais o desempenho geral dos multiprocessadores.

Para a realização dos testes foi utilizado o simulador Simics [Magnusson et al., 2002], o *benchmark* SPEC-OMP [Aslot et al., 2001] modificado para incorporar agendamento de laço dinâmico e o *benchmark* PARSEC [Bienia et al., 2008]. Dentre os resultados, obteve-se, para um processador com oito núcleos CMOS e vinte e quatro núcleos TFET utilizando distribuição igualitária da carga de trabalho para os núcleos e subdivisão da potência de forma dinâmica, uma melhoria de desempenho de 27% em comparação com a abordagem homogênea.

2.4.4 Núcleos Assimétricos

A era de *dark silicon* está impulsionando o surgimento de multiprocessadores assimétricos, onde os núcleos partilham a mesma arquitetura de conjunto instrução (*Instruction Set Architecture* (ISA)), mas oferecem diversas características de potência e desempenho. Esta heterogeneidade permite melhor adequação entre a demanda de aplicações e capacidades de computação que conduzem a uma melhoria substancial da eficiência energética. Um dos exemplos desta abordagem é o multiprocessador para dispositivos móveis denominado big.LITTLE da *Advanced RISC Machine* (ARM) [ARM, 2015], onde núcleos com execução fora de ordem - *Out-of-Order* (OoO) de alto desempenho Cortex-A15 são acoplados a núcleos com eficiência energética e execução em ordem (*in-order*) Cortex-A7 no mesmo chip.

A Figura 2.8 mostra um exemplo da arquitura ARM big.LITTLE onde o grupo de núcleos Cortex e cache L2 é denominado *cluster*, desta forma, neste caso a arquitetura possui dois *clusters* sendo um deles composto por dois núcleos Cortex-A15 e o outro por três núcleos Cortex-A7.

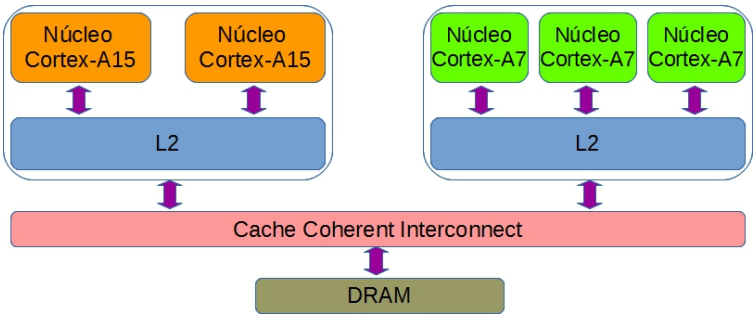


Figura 2.8: Multiprocessador assimétrico ARM big.LITTLE [Muthukaruppan et al., 2013].

No trabalho de Muthukaruppan et. al [Muthukaruppan et al., 2013] é apresentado um modelo de gerenciamento de energia para a arquitetura big.LITTLE da ARM no contexto de dispositivos móveis, que pode minimizar o consumo de energia dentro da restrição da TDP.

A abordagem propõe identificar o comportamento de potência e desempenho dos diferentes núcleos para que as tarefas possam ser atribuídas aos núcleos adequados em tempo real, sem alterar as propriedades do sistema operacional. Para isso deve-se certificar que núcleos com potência e frequência elevadas são utilizados somente quando for estritamente necessário, além de assegurar que a tensão dinâmica e escala de frequência - *Dynamic Voltage and Frequency Scaling* (DVFS) são aplicadas no *cluster* e não nos núcleos, necessitando, para isso, de estratégias de balanceamento de carga adequadas. Além disso, um *cluster* deve ser executado com a frequência mínima necessária sem prejudicar o desempenho, de modo a poupar energia, garantindo que o orçamento energético é alocado de forma oportunista entre os *clusters*. Por fim, o framework deve verificar se limites energéticos são extrapolados e se a qualidade de serviço - *Quality of Service* (QoS) das tarefas são degradadas.

Isto posto, foi desenvolvido um *framework* de gestão hierárquica de potência - *Hierarchical Power Management* (HPM) que se baseia na teoria de controle e integra vários controladores para alcançar a meta de troca entre energia e desempenho de forma ideal, sob orçamento de energia restrito em arquiteturas de multiprocessadores assimétricos. O HPM é integrado com Linux na plataforma ARM big.LITTLE e explora a assimetria entre os núcleos por

meio da migração seletiva e emprega DVFS para minimizar o consumo de energia, desde que satisfaçam as restrições de QoS. Nos experimentos realizados com o HPM, os autores constataram que ele pode explorar assimetria para proporcionar economias significativas de energia em comparação com multiprocessadores simétricos, além de possuir desempenho melhor do que o Linaro[2] e poder responder a emergência térmica de forma harmoniosa sem interferir nas propriedades desejadas do Linux CFS no que tange a QoS.

2.4.5 Métricas para Estimar e Utilizar o *Dark Silicon*

Para mitigar o *dark silicon*, metodologias de projeto devem adaptar-se e identificar os sistemas progressivos que efetivamente possam explorar o crescente *dark silicon*. No trabalho de Allred et al. [Allred, 2013, Allred et al., 2012] é demonstrado que confiar em métricas tradicionais do projeto pode levar a escolhas inadequadas, com o aumento da área de *dark silicon*. Para tanto, uma nova métrica para guiar sistemas cientes de *dark silicon* é proposta, através da utilização de um algoritmo de otimização estocástica para o projeto de multiprocessadores.

Para este estudo, foi utilizada uma arquitetura de multiprocessadores topologicamente homogênea com potência e desempenho heterogêneos - *Topologically Homogeneous Power-Performance Heterogeneous* (THPH) [Chakraborty and Roy, 2011], que é composta por núcleos idênticos projetados para serem ótimos em potência e desempenho para domínios de frequência e tensão diferentes. Tal abordagem de projeto foi concebida para oferecer maior eficiência energética do que a técnica de tensão dinâmica e escala de frequência (DVFS).

No sistema THPH, *threads* de frequências de utilização mais baixas são atribuídas aos núcleos mais lentos, de baixo consumo energético e *threads* de frequências de utilização mais altas são atribuídas aos núcleos mais rápidos, com maior consumo energético. No entanto, quando as exigências de carga de trabalho não são consistentes com a configuração do sistema, *threads* de alta frequência, por vezes, são atribuídas a núcleos de baixo desempenho, prejudicando a performance geral do sistema. Da mesma forma, de vez em quando, *threads* de baixa frequência serão atribuídas a núcleos de alto desempenho, degradando a eficiência energética do sistema. O aumento de *dark silicon* oferece uma oportunidade para um sistema THPH otimizar tal cenário, utilizando de forma eficaz os recursos de hardware adicionais para maior flexibilidade no tempo de execução. Eventualmente, ele pode se aproximar de uma configuração ideal em que cada segmento pode ser atribuído a um núcleo projetado de maneira ideal para ele.

Para o estudo da metodologia foi utilizado um sistema com 16 ULAs de 32 bits do *benchmark* ISCAS [Hansen et al., 1999], em que cada ULA pertence a um núcleo independente do conjunto de núcleos. O nível de *dark silicon* nos projetos variou entre 0% e 62,5%, representando várias gerações tecnológicas futuras. As configurações exibidas na Tabela 2.3 foram sintetizadas usando o *Synopsys Design Compiler* e uma biblioteca de tecnologia de $45nm$ para medir a sua potência e desempenho. Em cada projeto foram simuladas 1000 cargas de trabalho, cada uma composta por um conjunto de processos de software com

[2]Linaro é um projeto lançado em 2010, tendo como base uma organização sem fins lucrativos, fundada por um consórcio de empresas de hardware e software, incluindo ARM, AMD, IBM, Samsung, Canonical, entre outros. Seus objetivos incluem o desenvolvimento de softwares de código aberto especialmente para *System-on-a-chip* (SoC)s ARM. Mais informações sobre o projeto estão disponíveis em: https://www.linaro.org.

demandas de frequência escolhidos a partir de uma distribuição de Gauss, com uma média de 75% da frequência nominal, que capta a diversidade de aplicação em cargas de trabalho típico. O número de *threads* em cada conjunto é igual ao número total de núcleos ativos (ULA).

Tabela 2.3: Configurações dos processadores avaliados [Allred, 2013].

	Convencional	**Est. A**	**Est. B**	**Est C**	**Est. D**	**Est. E**
Núcleos em 100% freq	16	8	4	6	3	5
Núcleos em 75% freq	0	0	4	6	5	5
Núcleos em 50% freq	0	8	4	2	5	3
Núcleos em 25% freq	0	0	4	2	3	3

Nos resultados das simulações (Figura 2.9) foi possível observar que, em primeiro lugar, a eficiência energética num sistema convencional melhora com níveis mais elevados de *dark silicon*. Isso ocorre porque os níveis crescentes de *dark silicon* fornecem ao sistema maior flexibilidade na escolha de um conjunto de núcleos que melhor correspondam às exigências de software. Em segundo lugar, observa-se que um estilo de projeto que é ideal para 0% *dark silicon* (Estilo A) pode não ser eficiente para 62,5% de *dark silicon* com relação a objetivos de projeto tradicionais. Por outro lado, o Estilo B mostra que a promessa limitada em baixos níveis de *dark silicon*, pode ser superior com níveis crescentes de *dark silicon*.

A eficiência energética é inversamente proporcional a $Energia \times Atraso^{\gamma}$ e sua medida ideal é feita através da simulação de um sistema onde a *thread* pode ser atribuída ao núcleo de sua escolha independente de conflito com outra *thread* ou se há ou não aquele núcleo disponível no projeto atual.

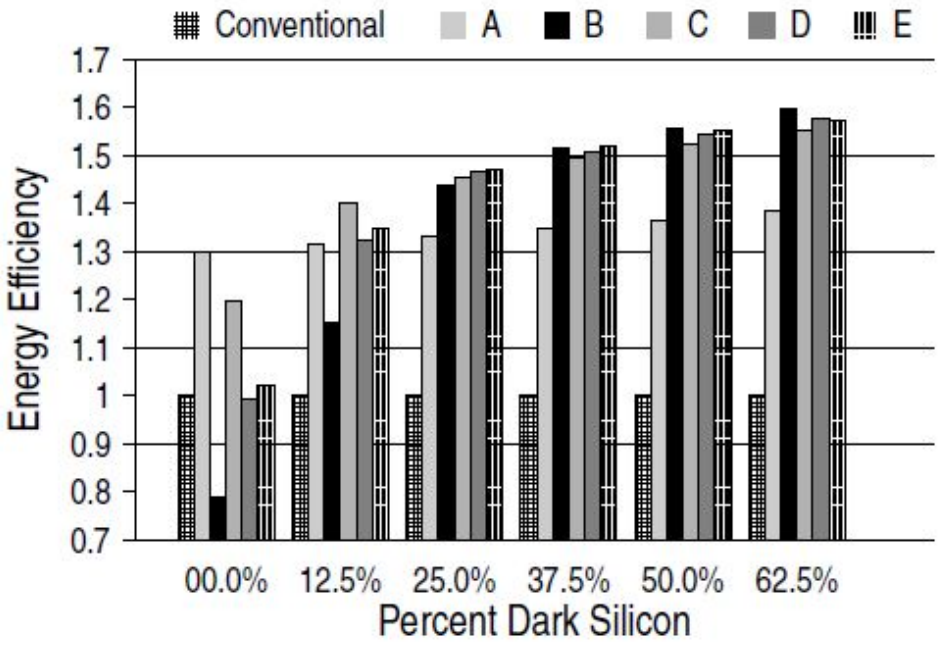


Figura 2.9: Eficiência energética nos diferentes estilos de projeto [Allred et al., 2012].

Diante dos resultados obtidos, Allred et al. [Allred, 2013] propuseram uma nova metodologia de projeto, na qual inicialmente estabeleceram métricas para mensurar níveis de *dark silicon* em sistemas heterogêneos, baseando-se na ideia de que quando um núcleo é projetado de maneira ideal para um conjunto de *threads*, ele pode operar de maneira mais eficiente para *threads* dentro de uma faixa de frequência próxima da ideal, este intervalo é denominado Faixa de otimização - *Range of Optimality* (RoO) de um núcleo específico da gama de processos de software para o qual o núcleo é especializado.

Quanto mais especializado for um núcleo de processamento, melhor a eficiência energética para uma série decrescente de cargas de trabalho de destino. Essa é a ideia explorada pela métrica de eficiência de utilização do *dark silicon* - *Dark Silicon Utilization Efficiency* (DSU) que é medida como o produto de dois sub-indicadores que dimensionam um ao outro para fornecer uma troca entre o nível de especialização de um núcleo e a gama de *threads* para o qual foi concebido, conforme Equação 2.8.

$$DSU = U_{DS} \times E_{cores} \tag{2.8}$$

A métrica de utilização do *dark silicon* - *Utilization Dark Silicon* (U_{DS}) (Equação 2.9) se refere a utilização do *dark silicon*, onde a chave para a exploração ideal está na escolha dos núcleos a serem ativados em um sistema com múltiplos núcleos com uma determinada combinação de núcleos especializados, tal que as cargas de trabalho do software escolham os núcleos especializados de maneira a utilizá-los uniformemente. Tendo a porção de *dark silicon* como γ, e n o número de núcleos totais no chip, a média de utilização do núcleo será igual a fração de silício ativo, $(1 - \gamma)$.

$$U_{DS} = 1 - \frac{\sum_{i}^{|O|}(utilization(O_i) - (1 - \gamma))}{\gamma \times (n \times (1 - \gamma))} \tag{2.9}$$

Onde O é o conjunto de núcleos mais utilizados, ou seja, com uma taxa média de utilização superior a $(1 - \gamma)$.

Desta maneira, o valor de U_{DS} pode variar de 0 a 1, sendo 0 quando a carga de trabalho sempre prefere o mesmo conjunto de núcleos, subutilizando os núcleos inativos; e 1 quando todos os núcleos são utilizados de maneira uniforme. Com o *dark silicon* em nível de γ (que pode variar de 0 a 1) e um multiprocessador com n *cores*, haverá $\gamma \times n$ *cores* inativos em qualquer tempo. Como em níveis mais baixos de cache (L2 ou L3) existem substancialmente menos densidade de potência, Allred et al. consideram que o *dark silicon* é aplicado aos núcleos mais utilizados.

A superutilização de determinados núcleos indica que eles podem melhorar sua eficiência, aumentando seu nível de especialização e diminuindo sua RoO, atendendo a um número menor de *threads*. Os núcleos subutilizados, por outro lado, devem aumentar sua utilização através do aumento da sua RoO, no geral há uma troca entre a especialização de um núcleo e sua utilização, sendo fundamental entendê-la para a concepção de sistemas de *dark silicon*.

A métrica E_{cores} (Equação 2.10) é uma medida de eficiência do sistema em comparação com a eficiência de um sistema ideal. Permite que uma *thread* seja sempre executada em um núcleo especializado para ela. Nesse estudo, a eficiência energética é a métrica alvo, resultando na seguinte equação:

$$E_{cores} = \frac{[Energy\ Efficiency]_{atual}}{[Energy\ Efficiency]_{ideal}} \tag{2.10}$$

O valor da métrica E_{cores} pode variar de 0 a 1, sendo mais alto para atribuições "*thread*-núcleo" que fornecem um melhor "software-hardware". A métrica E_{cores} também pode ser utilizada para outros objetivos do projeto, tais quais, a temperatura média ou a confiabilidade do chip, baseado em metas de projeto. Através dos experimentos realizados utilizando esta abordagem, no geral, a técnica demonstrou melhoria de 7-18% para tecnologias de longo prazo e 9-23% em benefícios de tecnologia de curto prazo.

2.5 Considerações Finais do Capítulo

Nesse capítulo foram introduzidos os principais conceitos para compreensão e norteamento do desenvolvimento da proposta de mestrado. Apresentou-se os conceitos principais sobre o funcionamento e projeto de transistores e a relação da evolução desses dispositivos com o fator de escala de Dennard. Também foi apresentado o problema da barreira de utilização de chips com transistores fabricados a partir de $90nm$, o surgimento do *dark silicon* e o impacto desses problemas no projeto de chips processadores mais avançados. Abordagens que se propõem a mitigar o *dark silicon* a partir de soluções arquiteturais foram apresentadas e discutidas visando apontar o estado da arte na área e lacunas ainda não abordadas. O próximo capítulo apresenta conceitos de exploração de espaço de projeto, bem como ferramentas e algoritmos para projetos de multiprocessadores que utilizam esta técnica.

Capítulo 3

Problema de Exploração de Espaço de Projeto

Devido à crescente complexidade do projeto de sistemas compostos por múltiplos processadores e à diversidade de parâmetros arquiteturais que devem ser explorados para encontrar a melhor combinação de objetivos múltiplos concorrentes (minimização do consumo energético, maximizar largura de banda, entre outros), a exploração do espaço de projetos de plataformas computacionais deve considerar todas as combinações possíveis de cada parâmetro (número de processadores, largura de emissão do processador, tamanhos de cache L1 e L2, etc). Diante deste cenário, basear-se somente na experiência de projetistas e/ou no desempenho teórico dos sistemas se torna inviável, sendo necessária uma metodologia automatizada para apoiar sistematicamente a exploração e comparação quantitativa das alternativas de projeto em termos de objetivos múltiplos concorrentes [Silvano et al., 2011, Zaccaria et al., 2010].

Neste capítulo serão apresentados os principais conceitos acerca do problema de exploração de espaço de projeto, bem como técnicas de otimização utilizadas e trabalhos relacionados ao tema.

3.1 Considerações Iniciais

A exploração de espaço de projeto - do inglês *Design Space Exploration* (DSE) - refere-se à atividade de explorar parâmetros alternativos para o projeto antes de sua implementação. A capacidade de operar no espaço de potenciais projetos candidatos torna essa atividade útil para muitas tarefas de engenharia, tais como [Kang et al., 2011]:

Prototipagem rápida: utilizada para gerar um conjunto de protótipos antes da implementação de maneira que a simulação e criação de perfis desses protótipos podem aumentar a compreensão do impacto das decisões de projeto, levando em conta a complexidade do sistema.

Otimização: empregada quando métricas estão disponíveis para a comparação de um projeto para o outro, sendo a exploração de espaço de projeto aplicada para executar a otimização, eliminando projetos inferiores e reunindo um conjunto de candidatos finais.

Integração de sistemas: exige a montagem, configuração e integração de vários componentes. Desse modo, DSE pode ser usado para encontrar conjuntos válidos e configurações que satisfaçam as restrições globais de projeto.

Segundo Ascia et al. [Ascia et al., 2007] o fluxo de projeto de um SoC apresenta o uso combinado de técnicas heterogêneas, metodologias e ferramentas com as quais um modelo arquitetônico é gradualmente refinado com base em especificações funcionais e requisitos do sistema. Cada etapa do fluxo de projeto pode ser considerada como um problema de otimização, que é resolvido pela definição de alguns parâmetros do sistema de tal forma a otimizar certos indicadores de desempenho. Estes problemas de otimização geralmente são abordados por meio de processos com base em sucessivos refinamentos cíclicos, em que a partir de uma configuração inicial do sistema, introduzem transformações em cada iteração a fim de melhorar a sua qualidade.

Nesse sentido, qualquer técnica de exploração de espaço de projeto pode ser esquematicamente representada como na Figura 3.1. Começando com uma configuração inicial e o processo de exploração que consiste num refinamento iterativo, compreendendo duas fases principais: avaliação e ajuste dos parâmetros de configuração. A fase de avaliação muitas vezes se resume a uma simulação em nível de sistema que constitui um gargalo no processo de exploração. A fase de ajustes utiliza os resultados da fase de avaliação para modificar os parâmetros de configuração do sistema, de modo a otimizar certos índices de desempenho. O ciclo termina quando uma configuração de sistema que atenda as restrições de projeto é obtida, ou, mais frequentemente, quando um conjunto de configurações para os índices a serem otimizados foram acumuladas.

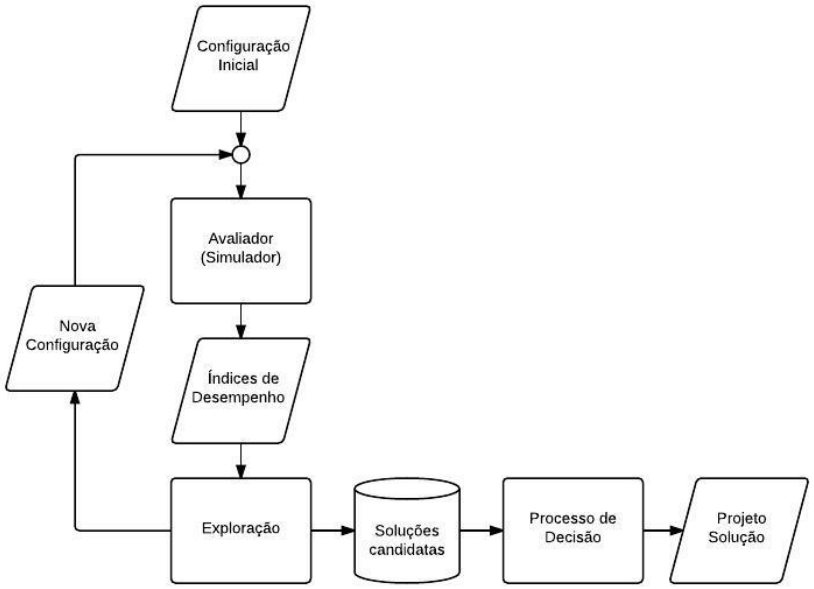


Figura 3.1: Fluxo geral da exploração de espaço de projeto [Ascia et al., 2007].

Em projetos de processadores, a etapa de exploração consiste em um problema de otimização multiobjetivo e frequentemente utiliza métodos heurísticos para sua resolução, conforme explicado em maior nível de detalhes na subseção 3.2. Assim, é um processo de otimização que explora um grande espaço de projeto composto por vários parâmetros microarquiteturais e de sistema que envolvem a minimização ou maximização de múltiplos objetivos [Hwang and Masud, 1979].

Após a obtenção das soluções candidatas, o projetista se depara com um novo desafio, a escolha da solução mais adequada para o projeto em questão. Como muitas vezes as soluções candidatas compreendem um número elevado de soluções dispersas ao longo da fronteira de Pareto [1], métodos que auxiliam o processo de decisão de acordo com o interesse do projetista - *Region of Interest* (ROI) têm sido propostos. De acordo com o trabalho de Miettinen [Miettinen, 1998], esta abordagem pode ser dividida em: sem preferência, quando as preferências não são consideradas; *a priori*, em que as preferências são determinadas antes da otimização e normalmente incluem a alteração da formulação dos objetivos; *a posteriori*, com as preferências sendo definidas após o processo de otimização e; interativas, com as preferências sendo definidas durante o processo de otimização. Algumas técnicas de resolução de problemas multiobjetivo, por exemplo, algoritmos evolucionários multiobjetivos - *Multiobejctive Evolucionary Algorithm* (MOEA) pertencem a categoria *a posteriori* e exemplos desta técnica podem ser encontrados nos trabalhos de Ishibuchi et al. [Ishibuchi et al., 2009] e Parreiras et al. [Parreiras et al., 2006].

As técnicas adotadas na resolução de um problema de exploração de espaço de projetos, devem ser aplicadas com cautela, por conta do grande número de alternativas de projeto a serem exploradas. Um sistema pode admitir milhões, senão bilhões de alternativas de projeto. Além disso, a avaliação de uma única configuração quase sempre requer o uso de simuladores ou modelos analíticos que são, muitas vezes, altamente complexos. Por isso, a abordagem manual de exploração do espaço de projeto é tediosa e propensa a erros. Da mesma forma, é computacionalmente inviável utilizar uma estratégia de exploração exaustiva. Portanto, para obter resultados subótimos em tempo viável, técnicas heurísticas são utilizadas para busca de melhores alternativas de projeto [Briao, 2008].

3.2 Otimização Multiobjetivo

Problemas de otimização simples intentam obter uma solução ótima que maximize ou minimize uma função objetivo. Contudo, muitos problemas reais, incluindo a exploração de espaço de projeto para multiprocessadores, envolvem mais de uma função objetivo, não possuindo, portanto, uma única solução exata. O problema da otimização multiobjetivo - *Multiobjective Optimization Problem* (MOOP) consiste na busca da resolução de um problema envolvendo propósitos que devem ser atingidos simultaneamente. Tais propósitos normalmente são conflitantes entre si, não existindo, por isto, uma solução única que otimize todos os parâmetros ao mesmo tempo. Logo, envolvem a minimização ou maximização de parâmetros simultâneos de uma coleção de propósitos, satisfazendo aos conjuntos de restrições do projeto. Para problemas multiobjetivos, otimizar significa encontrar uma

[1] O trabalho original de Pareto é "Cours d'economie politique, F. Rouge, Lausanne, 1896"

solução que forneça os valores de todas as funções objetivo aceitáveis para o projeto [Zaccaria et al., 2010].

Como exemplo, considere que o problema a ser otimizado possui N_{Obj} funções objetivo que formam o vetor $f(x) = [f_1(x), f_2(x), ..., f_{N_{Obj}}(x)]^T$, ele pode ser formulado como:

$$\begin{array}{ll} \text{Min/Max} & f(x) \\ \text{Sujeito a:} & g(x) = [g_1(x), g_2(x), ..., g_K(x)]^T \leq 0 \\ & h(x) = [h_1(x), h_2(x), ..., h_P(x)]^T = 0 \\ & {x_i}^{inf} \leq x_i \leq {x_i}^{sup} \end{array}$$

em que x é um vetor de variáveis de decisão com dimensão N_{var}, $g(x)$ e $h(x)$ representam o conjunto das restrições de desigualdade e igualdade de tamanho K e P, respectivamente. Os limites inferiores e superiores para cada variável x_i, que definem o espaço das variáveis, são representados por x^{inf} e x^{sup}.

Como as funções objetivo aplicadas aos MOOPs são, geralmente, conflitantes entre si, não se pode melhorar o valor de uma determinada função sem piorar o valor de outra. Para comparar as soluções possíveis para um determinado problema de forma a eleger as soluções preferíveis, utiliza-se comumente na literatura o conceito de dominância de Pareto [Zaccaria et al., 2010].

Na dominância de Pareto, diz-se que $f(x_a)$ domina $f(x_b)$ ($f(x_a) \prec f(x_b)$) quando a solução de $f(x_a)$ é superior a $f(x_b)$ em pelo menos uma função objetivo e quando a solução de $f(x_a)$ é pelo menos igual $f(x_b)$ em todas as funções objetivo. Uma solução é preferível se não existir solução que melhore pelo menos uma das funções objetivos sem piorar outra. Assim, a fronteira de Pareto é formada pela coleção de soluções não dominadas. Um dos problemas da dominância de Pareto é o número elevado de soluções dominantes para problemas com mais de três objetivos [Silvano et al., 2011].

A Figura 3.2 apresenta um exemplo da dominância de Pareto com um conjunto viável de soluções x_a, x_b, x_c. O ponto x_c é dominado pelo ponto x_b uma vez que ambos $f_1(x_c)$ e $f_2(x_c)$ são maiores do que $f_1(x_b)$ e $f_2(x_b)$. Neste caso, tem-se se que a fronteira de Pareto aproximada para o exemplo são os pontos x_a, x_b.

Por conseguinte, outras formas de dominância, baseadas na de Pareto, para identificação de soluções preferíveis têm sido propostas na literatura. Uma das primeiras técnicas propostas neste sentido foi a α-dominância [Ikeda et al., 2001], que estabelece limites superiores e inferiores de relações de compromisso entre pares de objetivos, através do parâmetro α, de modo a permitir a ampliação do hipervolume dominado para cada solução candidata.

No trabalho de Laumanns et al. [Laumanns et al., 2002] foi proposta a ε-dominância em que hipercubos, com tamanho definido pelo parâmetro ε, especificado pelo usuário, são distribuídos ao longo do espaço de objetivos, compostos por uma única solução não dominada. A definição adequada do parâmetro ε é essencial para o bom funcionamento da técnica, uma vez que para valores elevados de ε, não obstante aumentar a velocidade de convergência das soluções, pode prejudicar a qualidade da fronteira de Pareto e, para valores reduzidos

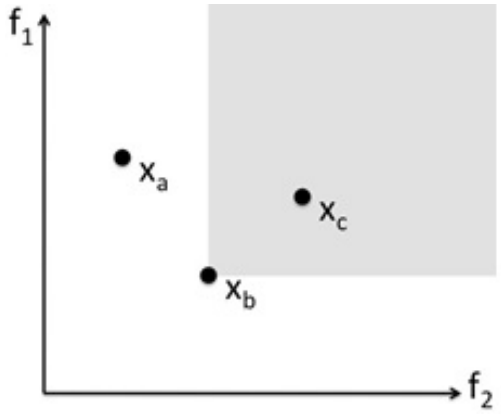


Figura 3.2: O conceito de dominância de Pareto para x_a, x_b, x_c, em que o ponto x_c é dominado pelo ponto x_b [Silvano et al., 2011].

de ε, aumentam a qualidade da fronteira de Pareto mas, também, o tempo de execução. Esta técnica de dominância tem sido empregada em algoritmos evolucionários tais quais ε-MOEA [Deb et al., 2003, 2005], ε-MyDE [Santana-Quintero and Coello, 2005] e ε-DEMO [Cai et al., 2007].

3.3 Exploração do Espaço de Projeto Utilizando Algoritmos Genéticos

Em geral, quando o espaço de configuração é demasiado grande para a exploração exaustiva, a utilização de técnicas evolutivas representa uma solução viável. Algoritmos genéticos - *Genetic Algorithms* (GA) têm sido aplicados em vários ambientes de projeto *Very Large Scale Integration* (VLSI) [Mazumder and Rudnick, 1999], como solução para problemas de roteamento [Lienig and Thulasiraman, 1993], em problemas de projeto como a estimativa de energia [Jiang et al., 1997], em testes de chips por meio da geração de vetores de testes eficientes [Saab et al., 1996], entre outros. Normalmente esses problemas são intratáveis no sentido de que nenhum algoritmo de tempo polinomial pode garantir uma solução ótima, pertencendo, portanto, às categorias NP-completos e NP-difíceis [Ascia et al., 2007].

A abordagem baseada em algoritmos genéticos mostra-se adequada para muitos cenários pois, em geral, encontra uma solução eficaz para tais problemas. É uma abordagem geral, requerendo a definição de uma representação da configuração, os operadores e as funções genéticas objetivas para serem otimizadas. [Ascia et al., 2007].

3.3.1 Conceitos Gerais Sobre Algoritmos Genéticos

Na biologia, todos os organismos vivos consistem em células e cada célula contém o mesmo conjunto de um ou mais cromossomos (cadeia de DNA) que servem como um "mapa" para o organismo. Um cromossomo pode ser dividido em genes, cada um dos quais codifica uma proteína particular. A grosso modo, pode-se pensar que um gene codifica um traço, como a cor dos olhos. As diferentes "configurações" possíveis para uma característica (por exemplo,

azul, marrom, avelã) são chamados alelos. Cada gene está localizado num lócus particular (posição) no cromossomo [Mitchell, 1998].

Muitos organismos têm vários cromossomos em cada célula. A coleção completa de material genético (o conjunto de cromossomos) é chamada genoma do organismo. O termo genótipo refere-se ao conjunto específico de genes contidos em um genoma. Dois indivíduos que têm genomas idênticos são ditos ter o mesmo genótipo. O genótipo dá origem, em fase de desenvolvimento fetal e, mais tarde, às características físicas e mentais, fenótipos do organismo, tais como a cor dos olhos, altura, tamanho do cérebro e inteligência.

Organismos cujos cromossomos estão dispostos em pares são chamados diplóide; organismos cujos cromossomos são desemparelhados (haploides). Na natureza, a maioria das espécies de reprodução sexuada são diploides, incluindo os seres humanos, contendo 23 pares de cromossomos em cada célula somática no corpo. Durante a reprodução sexual, a recombinação (ou cruzamento) ocorre: em cada um dos pais, os genes são trocados entre cada par de cromossomos para formar um gameta (um único cromossomo) e, em seguida gametas dos dois pais emparelham-se para criar um conjunto completo de cromossomos diploides. Na reprodução sexual haploide, genes são trocados entre os cromossomos de cadeia simples dos dois pais. Filhos estão sujeitos a mutação, que ocorre quando nucleotídeos únicos (bits elementares do DNA) são modificados de pais para filhos, resultantes muitas vezes de erros aleatórios de cópia de DNA. A aptidão de um organismo é tipicamente definida como a probabilidade do organismo vivo para reproduzir (viabilidade) ou como uma função do número de descendentes que possui (fertilidade) [Mitchell, 1998].

Algoritmos genéticos são uma subárea da computação evolutiva, e utilizam conceitos análogos a teoria da evolução natural proposta por Charles Darwin e aos fundamentos da genética. Foram introduzidos por John Holland [Holland, 1975] na década de 1960 e disseminados mais tarde por Goldberg [Goldberg, 1989]. Em algoritmos genéticos, os indivíduos são formados por genes, onde cada gene representa um parâmetro de configuração. Os genes podem ser pedaços individuais ou blocos curtos de bits adjacentes que codificam um elemento particular candidato a solução. Por exemplo, no contexto de função de otimização com múltiplos parâmetros, os bits que codificam um determinado parâmetro podem ser considerados como sendo um gene. Um alelo de uma sequência de bits é qualquer 0 ou 1; para alfabetos maiores são possíveis mais alelos em cada lócus.

Após a criação de indivíduos, uma coleção deles, denominada população, é formada, a qual caracteriza a exploração paralela do espaço de busca. Cada iteração realizada sobre a população é chamada de geração, sendo o número de gerações pode ser utilizado como critério de parada do algoritmo.

Como cada indivíduo da população pode ser uma solução candidata, um mecanismo de avaliação para discriminar os melhores e piores indivíduos para a solução é necessário. A seleção é realizada por uma função de aptidão (*fitness*), a qual imita o processo de seleção natural que conduz a evolução das espécies.

O cruzamento (*crossover*) consiste na troca de material genético entre os indivíduos (pais) mais aptos para a geração de novos indivíduos. O tipo de cruzamento pode variar de acordo com as características do problema, sendo as principais técnicas de cruzamento descritas a seguir:

Ponto único: Um ponto de cruzamento é escolhido aleatoriamente e a partir dele é realizado o cruzamento entre os pais, sendo que o primeiro filho herda do início do cromossomo até o ponto determinado do primeiro pai e o restante do segundo pai, e o segundo filho de maneira inversa [Holland, 1975], conforme Figura 3.3.

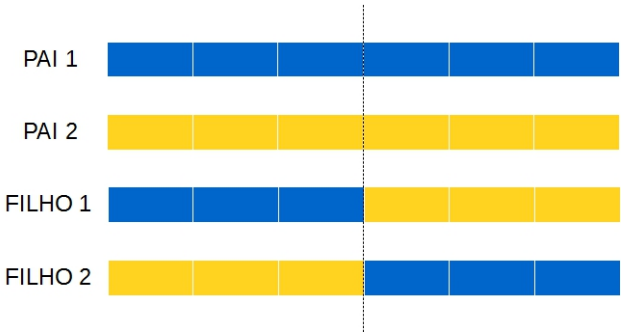


Figura 3.3: Cruzamento com um ponto.

Multiponto: Trata-se de uma generalização do cruzamento de ponto único, onde múltiplos pontos de cortes podem ser selecionados [Cavicchio, 1970] [DeJong, 1975](Figura 3.4).

Figura 3.4: Cruzamento com dois pontos.

Cruzamento uniforme: cada gene do cromossomo é escolhido aleatoriamente de qual pai deve ser herdado, mantendo-se, normalmente, um percentual de herança de cinquenta por cento de cada pai [Ackley, 2012], conforme ilustrado na Figura 3.5.

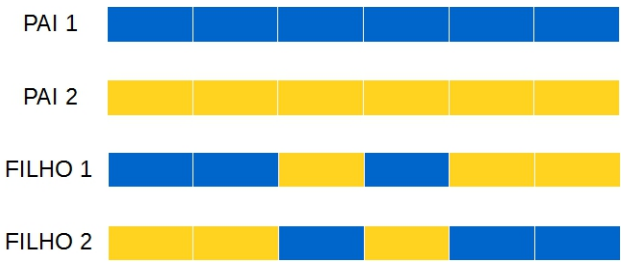


Figura 3.5: Cruzamento uniforme.

Apesar do cruzamento garantir certa variabilidade entre os indivíduos, pode ocorrer estagnação de soluções em uma determinada região. Para evitar este problema é empregado

o operador de mutação, que realiza modificações nas soluções com uma probabilidade pré-determinada. A mutação pode ser uniforme, na qual o gene do indivíduo que sofrerá mutação é escolhido aleatoriamente, sendo substituído por outro dentro de um limite inferior e superior pré-estabelecido para o gene em questão. Outros tipos somam um valor aleatório ao gene conforme uma distribuição e podem variar de acordo com o problema em questão [Bäck et al., 2000].

Cumpre consignar que a seleção, cruzamento, mutação e substituição dos indivíduos são realizadas recursivamente, com o objetivo de aumentar a qualidade dos indivíduos a cada geração, de forma que a solução ótima seja encontrada.

Dado um problema claramente definido a ser resolvido e uma representação de sequência de bits para soluções candidatas, um algoritmo genético simples funciona da seguinte forma [Mitchell, 1998]:

1. Comece com uma população gerada aleatoriamente de N indivíduos (soluções candidatas para um problema).
2. Calcule a função de aptidão *(fitness)* $f(x)$ de cada indivíduo x da população.
3. Repita as seguintes etapas até que N filhos sejam criados:
   (a) Selecione um par de indivíduos pais da população atual, a probabilidade da seleção é uma função crescente de aptidão. A seleção é feita com a substituição, o que significa que o mesmo indivíduo pode ser selecionado mais de uma vez para se tornar um pai.
   (b) Com probabilidade P_c (probabilidade ou taxa de cruzamento), cruzar o par de cada ponto escolhido aleatoriamente (elegido com probabilidade uniforme) para formar dois filhos. Se nenhum cruzamento ocorre, dois filhos que são cópias exatas de seus respectivos pais são gerados. Note-se que aqui a taxa de cruzamento é definida como a probabilidade de que dois pais vão cruzar em um ponto único. Existem também versões de cruzamento multiponto, em que a taxa para um par de pais é o número de pontos em que ocorre um cruzamento.
   (c) Mutar os dois filhos em cada lócus com probabilidade P_m (probabilidade ou taxa de mutação), e coloque os indivíduos resultantes na nova população.
   Se N é ímpar, um novo indivíduo da população pode ser descartado de forma aleatória.
4. Substitua a população atual pela nova população.
5. Retorne ao passo 2.

Esse procedimento é a base para a maioria das aplicações de algoritmos genéticos. Há um número de detalhes que precisam ser definidos, tais como o tamanho da população e as probabilidades de cruzamento e mutação, e o sucesso do algoritmo depende muito desses detalhes.

Os pontos fortes desta abordagem são sua generalidade, não requerendo conhecimento detalhado do sistema, embora seja necessário definir o mapeamento de uma configuração do

sistema para um indivíduo. A maneira mais simples de fazer isto é a utilização de um gene para cada um dos parâmetros do sistema e limitar a sua gama de variação para que um dos parâmetros o representem [Ascia et al., 2007].

Como exemplo de algoritmo genético multiobjetivo que utiliza como base a sequência de passos descrita anteriormente pode-se mencionar o *Multiobjective Genetic Algorithm* (MOGA) [Fonseca and Fleming, 1998] que foi o pioneiro a propor classificação baseada no conceito de dominância de Pareto, combinando domínio com informações de preferências para produzir uma estratégia de atribuição de função de aptidão adequada.

Não obstante as técnicas apresentadas serem a base para o projeto de algoritmos genéticos, melhorias no mecanismo de seleção baseada em elitismo propõem a inclusão de melhores indivíduos da população atual na próxima população, de maneira a preservar as melhores soluções encontradas. Neste sentido, várias pesquisas comprovam a importância do elitismo na estratégia de seleção [Lee and El-Sharkawi, 2008, Zitzler, 1999]. Na subseção 3.3.2 será apresentado o algoritmo *Nondominated Sorting Genetic Algorithm II* (NSGA-II) que utiliza seleção baseada em elitismo e foi utilizado para resolução do problema de exploração de espaço de projeto ciente de *dark silicon* abordado neste trabalho.

3.3.2 Algoritmo NSGA-II

O algoritmo genético com ordenação não dominada II (NSGA-II) [Deb et al., 2002, 2000], é uma variação do algoritmo NSGA [Srinivas and Deb, 1994] e é baseado em uma ordenação elitista por dominância (Pareto *ranking*), em que combina a população atual com a população anterior para preservar os melhores indivíduos. O algoritmo é dividido em três componentes: seleção rápida não dominada (*fast nondominated sorting*), distância da multidão (*crowding distance*) e o laço principal.

O algoritmo utiliza uma abordagem de seleção rápida não dominada em que para cada solução i da população P calcula-se a contagem de domínio, o número de soluções que a dominam (n_i), e o conjunto de soluções que são dominadas (S_i) por i. Todas as soluções da primeira fronteira não dominada terão zero como contagem de dominação. Este processo é realizado para cada solução e, a cada iteração são retiradas as soluções que não são dominadas, diminuindo-se do contador das soluções dominadas. A cada repetição uma nova fronteira é criada com as soluções retiradas do conjunto. Os indivíduos localizados nas primeiras fronteiras (F_t) são os que apresentam as melhores soluções para a geração.

Utilizando o critério de dominância, o algoritmo integra o conceito de elitismo que classifica a população em diferentes níveis de qualidade, permitindo a priorização dos indivíduos melhores classificados.

A etapa de cálculo do número de soluções que dominam uma determinada solução i e do conjunto de soluções que ela domina requer $O(MN^2)$ comparações, onde M é o número de objetivos e N o tamanho da população, o pseudo-código desta etapa é apresentado no Algoritmo 1 . Por sua vez, o laço para a criação das fronteiras possui complexidade $O(N^2)$ para o pior caso, quando cada indivíduo ocupa uma fronteira, conforme detalhado no

Algoritmo 2. Logo, a complexidade do componente (*fast nondominated sorting*) é $O(MN^2)+O(N^2)$, ou apenas, $O(MN^2)$.

Algoritmo 1: Cálculo do número de soluções.

```
para todo i ∈ P faça
    para todo j ∈ P faça
        se i ≺ j então
            S_i = S_i ∪ {j}
        senão se j ≺ i então
            n_i = n_i + 1
se n_i = 0 então
    F_1 = F_1 ∪ {i}
```

Algoritmo 2: Criação da fronteira de Pareto.

```
t=0;
enquanto F_t ≠ 0 faça
    H = 0
    para todo i ∈ F_t faça
        para todo j ∈ S_i faça
            n_j=n_j-1
            se D_j=0 então
                H = H ∪ {j}
                i = i + 1
            F_t = H
retorna F
```

Após a criação das fronteiras, um cálculo de distância entre as soluções, distância da multidão, da mesma fronteira é realizado de modo a classificar a população de acordo com os valores de cada função objetivo em ordem crescente de magnitude. Em seguida, para cada função objetivo, soluções com valores mais longínquos recebem infinito para a distância. Para as outras soluções é atribuído um valor de distância igual a diferença absoluta normalizada com os valores de função de duas soluções adjacentes, formando um cubóide em relação ao ponto central, cujos vértices são seus vizinhos mais próximos. O valor geral da distância é calculado como a soma dos valores de distâncias individuais correspondentes a cada objetivo. A ideia é distribuir os resultados ao longo da fronteira de Pareto, conforme ilustrado na Figura 3.6. Os círculos representam as soluções não dominadas da mesma fronteira [Deb et al., 2002].

Utilizando as equações 3.1 e 3.2 é possível calcular de forma algorítmica o valor da distância da multidão de uma solução.

$$d(x_i) = d(x_i) + d_j(x_i) \tag{3.1}$$

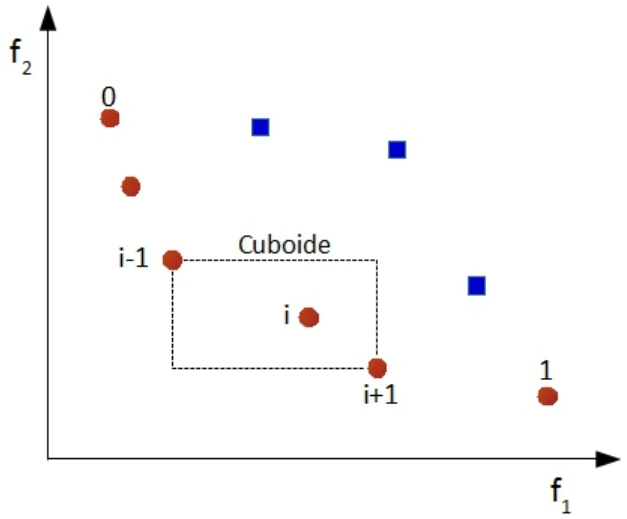


Figura 3.6: Cálculo de *crowding distance*.

$$d_j(x_i) = \frac{f_j(x_{i+1}) - f_j(x_{i-1})}{f_j^{max} - f_j^{min}}, i \in F \text{ e } 1 \leq j \leq M \tag{3.2}$$

Em que:

$d(x_i)$: Distância da multidão da solução i;
$d_j(x_i)$: distancia da solução i e suas soluções mais próximas na fronteira F;
$f_j(x_{i+1})$: valor da função objetivo j para o vizinho $i+1$;
$f_j(x_{i-1})$: valor da função objetivo j para o vizinho $i-1$;
f_j^{max} : valor máximo da função objetivo j na fronteira F;
f_j^{min} : valor mínimo da função objetivo j na fronteira F.

A complexidade deste procedimento é determinada pelo algoritmo de ordenação que, no pior caso, quando todas as soluções estão em uma frente, requer $O(MN\ log\ N)$.

Após a criação de uma população aleatória P_0, ordenada com base em não-dominação, classificação de acordo com seu grau de dominação, então o algoritmo utiliza um processo de seleção por torneio, cruzamento e mutação para obter a população filha Q_0, ambas de tamanho N. As populações P_0 e Q_0 são unidas para gerar uma população R_0.

Após esse procecimento, um novo processo para as próximas gerações é considerado para a criação da *t-ésima* geração, R_t. Agora, as soluções pertencentes ao melhor conjunto não dominado F_1, devem ser enfatizadas com relação a todas as outras soluções combinadas da população. Se o tamanho de F_1 for menor do que N, soluções da fronteira seguinte são escolhidos sucessivamente até que a população esteja completa (tamanho N). A nova população $P_t + 1$ de tamanho N é agora usada para seleção, cruzamento e mutação para a criação da população $Q_t + 1$, e assim sucessivamente.

Para escolher uma solução em detrimento de outra será comparado se a mesma possui uma menor contagem de dominação, e quando ambas pertencerem à mesma fronteira, o

critério de desempate será a que possuir a maior distância da multidão. Nesta etapa, a seleção possui complexidade $O(2Nlog(2N))$. O procedimento pode ser acompanhado na Figura 3.7.

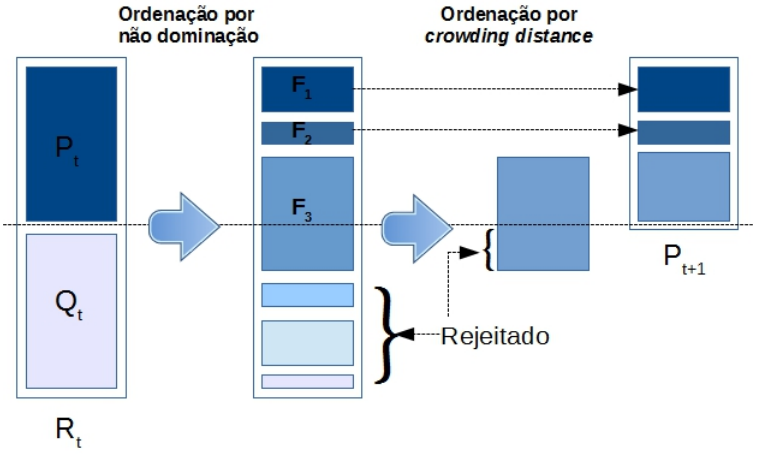


Figura 3.7: Procedimento de seleção utilizado no NSGA-II [Deb et al., 2002].

Nota-se que, considerando todos os componentes do algoritmo NSGA-II, a complexidade de pior caso do algoritmo ainda continua sendo $O(MN^2)$. A principal vantagem do algoritmo está na manutenção da diversidade entre as soluções não dominadas, no método de comparação por multidão e na dispensabilidade da criação de nichos. Suas desvantagens são para uma fronteira F_1 maior que N, uma vez que soluções são perdidas e o algoritmo entra em um ciclo de soluções Pareto-ótimas e soluções não Pareto-ótimas até confluir para um conjunto de soluções Pareto-ótimas [Deb, 2001]. O pseudo código do NSGA-II é apresentado no Algoritmo 3 [Cordeiro and Silva-Filho, 2010] e seu fluxograma na Figura 3.8.

Algoritmo 3: Algoritmo NSGA-II.

gerar P_0 de tamanho N
$Q_0 = \varnothing$
$t = 0$
$R_t = P_t \cup P_t$
F= seleção rápida não dominada (R_t)
$P_{t+1} = \varnothing$
$i = 0$
enquanto $|P_t + 1| + |F| \leq N$ **faça**
    cálculo distância da multidão (F_t)
    $P_{t+1} = P_{t+1} \cup F_t$
    $i = i + 1$
    ordenar (F_t)
    $P_{t+1} = P_{t+1} \cup F_t[1 : (N - |P_{t+1}|)]$
    Q_{t+1} = criar nova população (P_{t+1}) $t = t + 1$
retorna população P_t;

O algoritmo NSGA-II também permite a implementação de restrições para um determinado problema, apenas com uma modificação da função dominância para tratar três situações: 1) quando ambas as soluções são viáveis; 2) uma solução é viável e outra não; e 3)

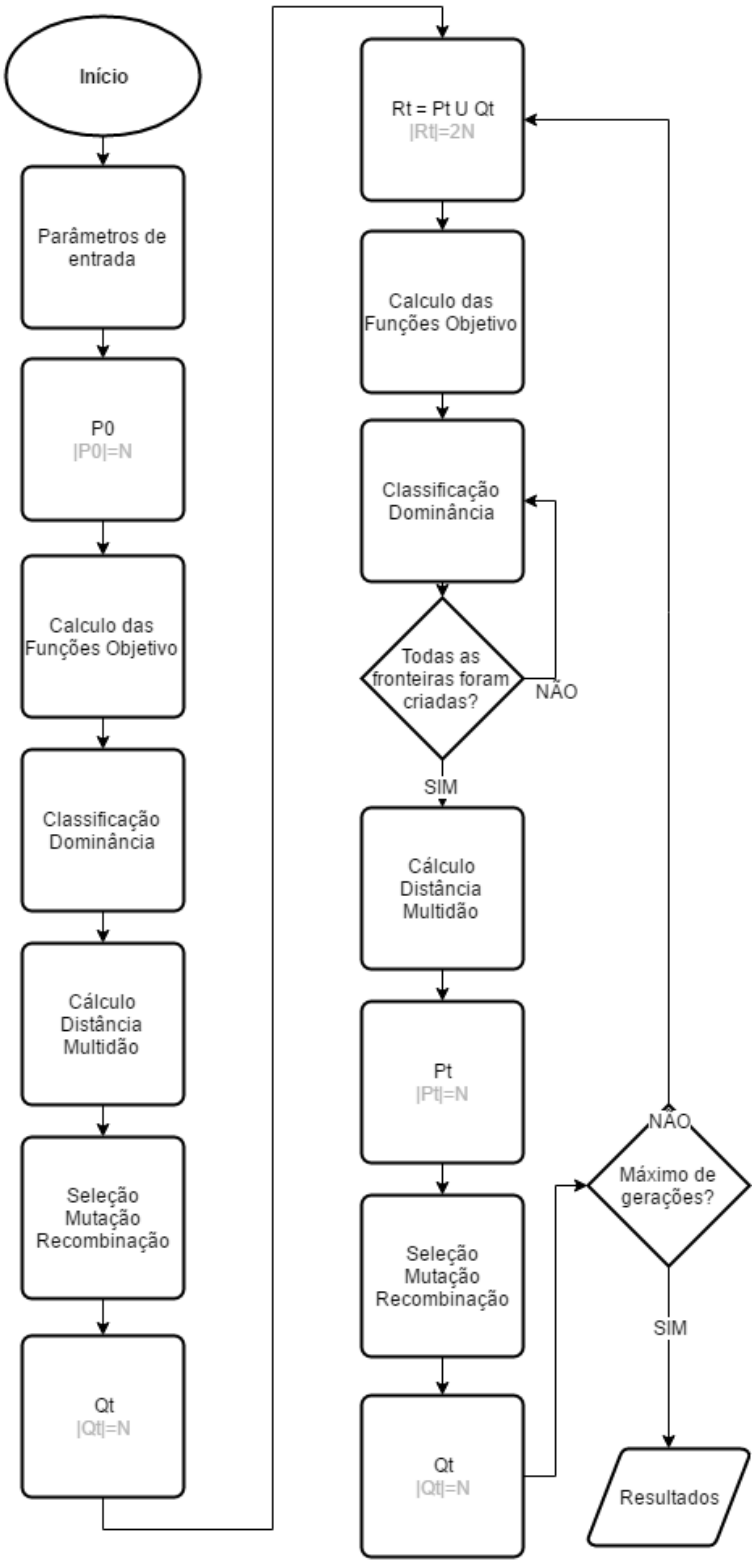


Figura 3.8: Fluxograma do NSGA-II.

ambas são inviáveis. Para isto, diz-se que uma solução i domina com limitações uma solução j, se qualquer uma das seguintes condições é verdadeira.

1. A solução i é viável e a solução j não é;

2. As soluções i e j são ambas inviáveis, mas solução i viola menos restrições;

3. Ambas soluções são viáveis e solução i solução domina j.

O efeito do uso da dominância restrita é que qualquer solução viável tem uma melhor classificação de não-dominação do que qualquer solução inviável. Todas as soluções viáveis são classificadas de acordo com seu nível de não-dominação com base nos valores das funções objetivo. No entanto, entre duas soluções inviáveis, a solução com uma violação de restrição menor tem um ranking melhor. Além disso, esta modificação, em princípio, não muda a complexidade computacional do algoritmo. O restante dos procedimentos descritos do NSGA-II permanecem inalterados.

3.4 Exemplos de Aplicações

Heurísticas de otimização multiobjetivo e técnicas de simulação de alto nível diminuem drasticamente o tempo de exploração, garantindo simultaneamente um certo nível de proximidade à fronteira real de Pareto.

Desta forma, a exploração de espaço de projeto se inicia a partir de uma definição de espaço representando o conjunto de configurações arquitetônicas viáveis. Uma ferramenta para DSE utiliza critérios numéricos e objetivos combinados com heurísticas eficientes para conduzir a exploração. O pressuposto básico é que o modelo deve ser configurável podendo ser manipulado para gerar automaticamente qualquer instância do espaço de projeto destino. Dado o modelo configurável, a ferramenta DSE vai mudar sistematicamente todos os parâmetros em cada etapa e avaliar o melhor resultado com base em heurísticas de otimização robustas, dentre as quais pode-se citar o algoritmo NSGA-II [Deb et al., 2002]. O resultado final do fluxo automático DSE é um conjunto de configurações dominantes no espaço de projeto com valores muito próximos da fronteira de Pareto. Além disso, todos os dados relativos a avaliações anteriores são armazenadas em um banco de dados estruturado que podem ser analisadas usando ferramentas matemáticas e estatísticas para obter informações agregadas.

Para diminuir o tempo de exploração do espaço de projeto, técnicas de modelagem de resposta de superfície - *Response Surface Modeling* (RSM) têm sido utilizadas para reduzir o número de simulações, definindo um modelo de resposta analítica das métricas de sistema, baseado em um subconjunto de simulações utilizadas como conjunto de treino. Um conjunto de técnicas analíticas (interpolação de Shepard, funções de base radial [Joseph et al., 2006b], regressão linear [Joseph et al., 2006a, Lee and Brooks, 2006], redes neurais artificiais [Bishop, 1995] e regressão *Spline-based*) foram implementados como módulos de software e integrado ao *framework*.

3.4.1 Exploração de Espaço de Projeto Utilizando NSGA-II

Em Cordeiro et al. [Cordeiro and Silva-Filho, 2010] é proposta uma metodologia para ajuste da arquitetura de memória com dois níveis, sendo o segundo nível unificado, utilizando o algoritmo NSGA-II. O foco deste trabalho está na compensação entre desempenho e

energia, uma vez que o consumo energético de hierarquias de memória pode chegar a 50% da energia consumida por um microprocessador [Verma and Marwedel, 2007, Kandemir and Choudhary, 2002] e ajustes adequados de parâmetros de memória cache podem reduzir em até 60% o consumo energético [Zhang and Vahid, 2003].

A arquitetura adotada é composta por um processador *Microprocessor without Interlocked Pipeline Stages* (MIPS) com os componentes de cache IC1, DM1 e L2, e a memória principal (MEM). Utiliza-se também uma tensão de entrada de $1,7V$, com política de escrita *write-through* e tecnologia de transistor de $70nm$. Para o espaço de exploração foram utilizadas configurações de cache comerciais normalmente utilizadas em sistemas embarcados (Tabela 3.1).

Tabela 3.1: Configurações de memória utilizadas

Parâmetro	**Cache Nível 1**	**Cache Nível 2**
Tamanho de cache	2KB, 4KB, 8KB	16KB, 32KB, 64KB
Tamanho de linha	8B, 16B, 36B	8B, 16B, 32B
Associatividade	1, 2, 4	1, 2, 4

Cada configuração de memória do espaço de exploração é representada por um indivíduo da população do NSGA-II. Para o mapeamento da cache foi definido que o indivíduo (cromossomo) que representa a solução possui 9 parâmetros: tamanho de cache, tamanho de linha e associatividade para cache de instrução (IC1) e dados (DM1) do primeiro nível e cache unificada do segundo nível (L2). No processo de cruzamento são esses parâmetros que são trocados entre os indivíduos, e modificados no processo de mutação.

Os objetivos a serem otimizados pelo NSGA-II são o consumo energético e o total de ciclos para a execução, que refletem no desempenho da arquitetura. No término da execução do algoritmo serão obtidas as soluções candidatas na fronteira de Pareto. O fluxo de execução do algoritmo é mostrado na Figura 3.9.

Conforme o fluxograma, após o início do algoritmo a população inicial é criada. A cada nova solução criada são definidos novos parâmetros de cache e esses parâmetros são submetidos às ferramentas Simplescalar [Burger and Austin, 1997] e eCACTI [Mamidipaka and Dutt, 2004] de modo a obter os valores de energia e ciclos correspondentes. Posteriormente, as informações obtidas são atualizadas no conjunto de dados que constituem a solução. O mesmo processo de simulação para obter valores de energia e ciclos é necessário nas etapas de cruzamento e mutação. Após a nova população ter sido criada, caso o critério de parada seja satisfeito o algoritmo termina, caso contrário retorna para a execução que foi descrita no Algoritmo 3.

Os parâmetros do NSGA-II utilizados, tais como probabilidade de mutação, cruzamento e critério de parada são apresentados na Tabela 3.2. O critério de parada utilizado é baseado na comparação dos cinco melhores indivíduos em relação à população anterior. Caso não haja diferença então o algoritmo termina.

Em experimentos realizados com 18 aplicações foi possível concluir que as soluções obtidas estão próximas ao Pareto-ótimo. Os autores observaram, por meio de análises comparativas com a heurística TECH-CYCLES, que o NSGA-II apresentou melhores resultados em todas as aplicações. Em relação ao número de simulações, o NSGA-II obteve redução do espaço de

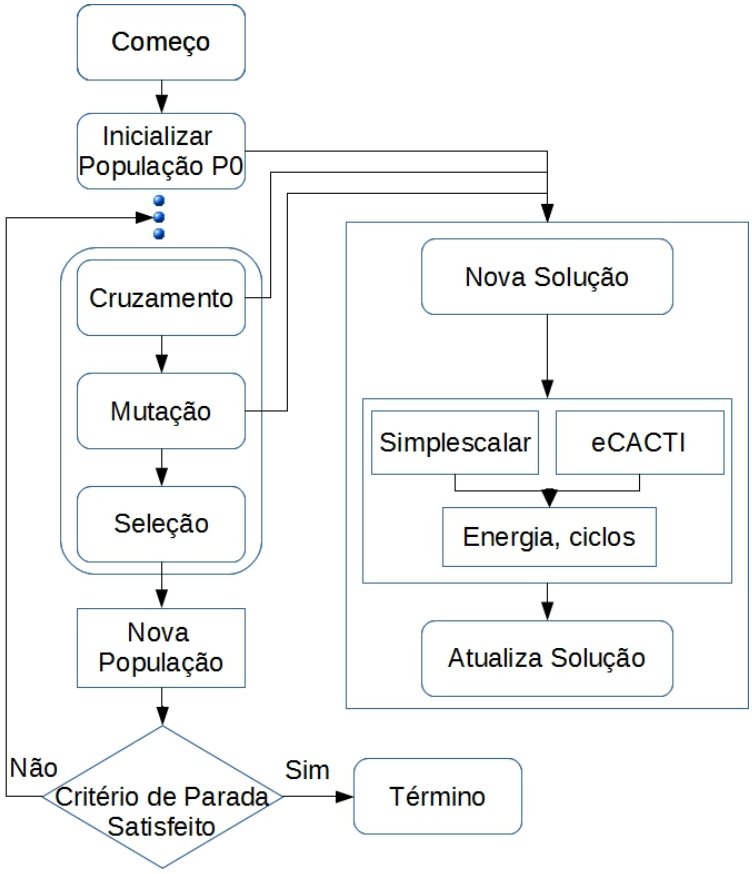


Figura 3.9: Fluxo de execução do algoritmo [Cordeiro and Silva-Filho, 2010].

Tabela 3.2: Parâmetros para o NSGA-II [Cordeiro and Silva-Filho, 2010].

Parâmetros	**Valores**
Taxa de Cruzamento	50%
Taxa de Mutação do Indivíduo	20%
Taxa de Mutação do Gene	20%
Tamanho da População	10

busca, sendo necessária em média 104 simulações, o que corresponde a 1% do espaço total de exploração.

3.4.2 Ferramenta de Exploração de Espaço de Projeto Multicube

O Multicube [Silvano et al., 2011, Zaccaria et al., 2010] é um *framework open source* para apoiar a exploração de espaço de projeto multiobjetivo de arquiteturas SoC com múltiplos núcleos de maneira automatizada. O *framework* permite o ajuste de parâmetros arquiteturais para minimizar diversas métricas (como energia e latência) enquanto atende as restrições de nível de sistema (tais como o *throughput*, largura de banda e QoS). O projeto centra-se na definição de metodologias de modelagem e otimização automáticas para melhorar o fluxo de projeto de um SoC convencional.

A ferramenta explora de maneira eficiente e automática arquiteturas embarcadas paralelas em termos de vários parâmetros de projeto, tais como paralelismo disponível (por exemplo, número de núcleos e largura de emissão do processador), subsistema de cache (por exemplo, tamanho de cache e associatividade) e parâmetros relacionados com a *Network-on-a-chip* (NoC) (por exemplo, o tamanho do *buffer* de canal). Ao lidar com arquiteturas de múltiplos núcleos complexos, surge a necessidade em adotar técnicas automatizadas de exploração com base em heurísticas de otimização que devem ser utilizadas para descobrir um conjunto aproximado de Pareto em um tempo razoável.

O fluxo de projeto do Multicube é composto por dois *frameworks*, conforme a Figura 3.10.

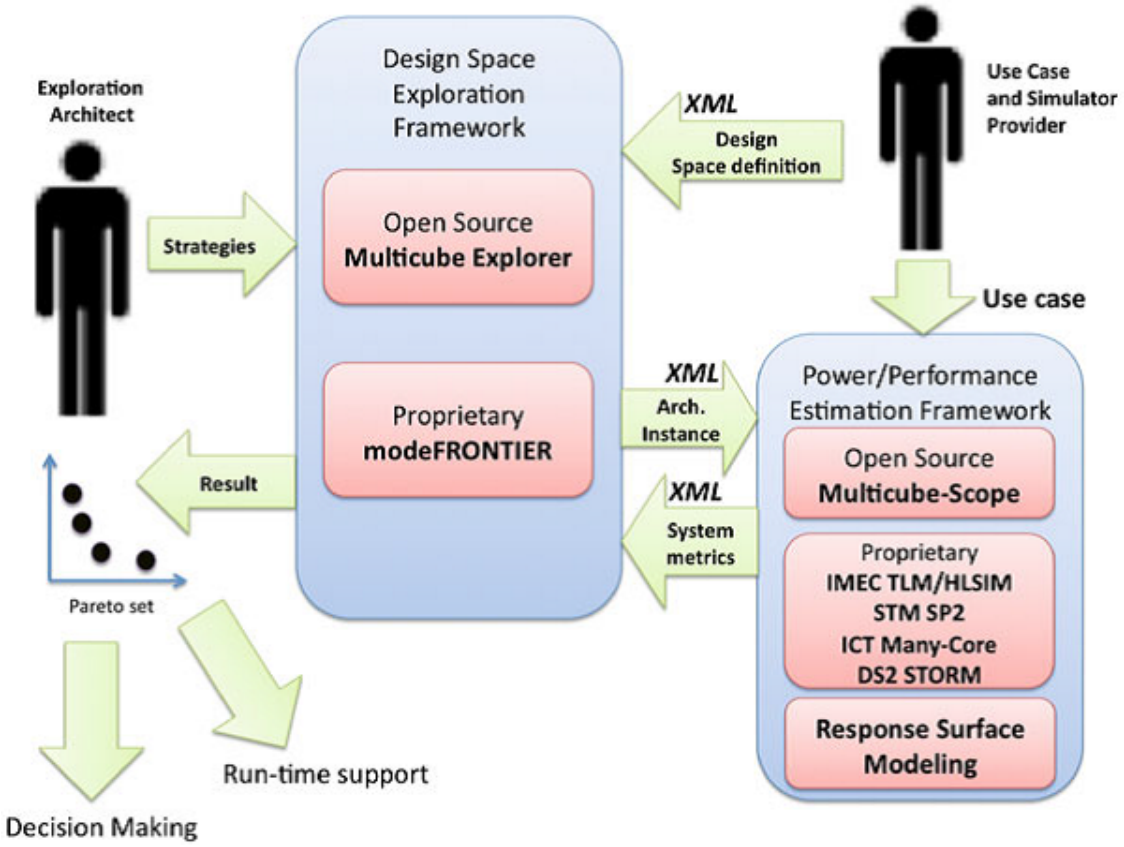


Figura 3.10: Fluxo de projeto Multicube [Zaccaria et al., 2010].

O *framework* de estimativa de potência/desempenho inclui ferramentas de simulação para potência e desempenho das arquiteturas a serem exploradas. Cada modelo de simulação configurável recebe como entrada uma configuração arquitetônica, gerando no final da simulação as métricas da aplicação de referência. Os simuladores utilizados no âmbito do projeto são o MULTICUBESCOPE, IMEC TLM e o STMicroelectronics SP2. A interface da ferramenta é padronizada com a definição da interface *eXtensible Markup Language* (XML) MULTICUBE, sendo aberta para a inclusão de outros simuladores.

O *framework* de exploração de espaço de projeto contém ferramentas para DSE automatizado e interage com o *framework* de estimativa de potência/desempenho gerando diferentes instâncias do espaço de projeto e realizando a leitura dos resultados de avaliação de potência e desempenho (métricas do sistema). Este *framework* inclui várias heurísticas de otimização

para identificação de compensação entre potência e desempenho, gerando a fronteira próxima da de Pareto.

O provedor de caso de uso e simulador é um usuário do sistema que fornece ao *framework* um conjunto de parâmetros configuráveis e modelos de estimativas de desempenho e potência para o caso de uso de destino.

Arquiteto de exploração é o responsável por otimizar a arquitetura configurável através do *framework* de DSE, podendo interagir diretamente com o núcleo de DSE para delinear estratégias de exploração, métricas e restrições.

Os resultados das explorações finais (fronteira de Pareto) podem ser ainda otimizados por um conjunto de mecanismos de tomada de decisão, e as soluções podem ser ordenadas de acordo com alguns critérios de classificação para obter a solução candidata final para a execução.

Um dos experimentos realizados com a ferramenta consistiu na customização de uma arquitetura *Chip Multicore Processors* (CMP) para a execução de um aplicativo decodificador do formato MPEG2. A arquitetura alvo consiste em um multiprocessador de memória compartilhada e cache L2 privada, sendo que os parâmetros explorados e sua faixas de valores são os listados na Tabela 3.3.

Tabela 3.3: Espaço de projeto para a plataforma multiprocessador com memória compartilhada [Zaccaria et al., 2010].

Parâmetros	**Mínimo**	**Máximo**
Número de processadores	2	16
Largura de emissão do processador	1	8
Tamanho da cache L1 de instruções	2K	16K
Tamanho da cache L1 de dados	2K	16K
Tamanho da cache L2 privada	32K	256K
Associatividade da cache L1 de instruções	1	8
Associatividade da cache L1 de dados	1	8
Associatividade da cache L2 privada	1	8
Tamanho do bloco das caches	16	32

Para a avaliação das métricas do sistema foi utilizado a ferramenta de simulação SESC [Renau et al., 2005] que tem suporte para a ferramenta CACTI [Li et al., 2011, Wilton and Jouppi, 1996] e WATTCH [Brooks et al., 2000] para os cálculos de consumo de energia da memória e núcleos, respectivamente. Os objetivos considerados no experimento foram ciclos, que refere-se ao número real de ciclos de execução da aplicação alvo na arquitetura, e energia, que é o consumo de energia (em Joule) associado com a arquitetura quando executa a aplicação alvo. A exploração de espaço de projeto foi composta por 131.072 configurações alternativas. Segundo os autores, o Multicube foi capaz de encontrar a curva de Pareto que corresponde aos objetivos definidos no experimento.

3.4.3 Exploração de Espaço de Projeto com *Dark Silicon*

No trabalho de Turakhia et al. [Turakhia et al., 2013] é proposta uma otimização iterativa por meio da síntese arquitetural em MPSoCs com *dark silicon* que permite determinar o número ideal de núcleos de cada tipo para fornecer CMPs heterogêneos. Essa abordagem iterativa possibilita que decisões possam ser realizadas em tempo de execução, incluindo o grau de paralelismo para cada aplicação e mapeamento de *threads* para os núcleos. A técnica proposta foi implementada em um *framework* denominado Hades.

Nota-se que neste trabalho o objetivo não é atenuar o *dark silicon*, e sim melhorar o desempenho. A potência de pico e área em *dark silicon* são limitadas baseando-se apenas nas restrições de potência de pico e área ($P_{orçamento}$ e $A_{orçamento}$), informadas pelo projetista. Os parâmetros de exploração de espaço de projeto são a quantidade de núcleos, tamanho do *Reorder Buffer* (ROB), tamanho da cache L1, tamanho da cache L2 e frequência (Tabela 3.4) e os objetivos são maximizar a quantidade de núcleos de cada tipo e o desempenho, tendo como restrição um orçamento de potência de pico e área.

Além disso foi proposto um modelo de desempenho para aplicações *multi-threaded* em núcleos homogêneos e CMPs heterogêneos. O tempo de execução E_{ij} para uma aplicação $i(i \in [1, N])$ executando em um CMP homogêneo com núcleos do tipo $j(j \in [1, M])$ é expresso pela equação 3.3:

$$E_{ij} = t^s_{ij} + \frac{t^p_{ij}}{D_i} + D_i K_{ij} \quad i \in [1, N], j \in [1, M] \tag{3.3}$$

Em que t^s_{ij} é o tempo de execução na fase sequencial, t^p_{ij} o tempo de execução na fase paralela, D_i representa o grau de paralelismo da aplicação e K_{ij} simboliza o aumento da latência na execução paralela causada pela contenção de recursos.

Já o tempo de execução para CMPs heterogêneos é diferente, pois a latência é determinada pelo pior caso de todos os segmentos paralelos. Logo, o núcleo mais lento determina o tempo de execução de uma aplicação. Com base nesta informação é apresentada a equação 3.4 para obtenção do tempo de execução em um CMP heterogêneo.

$$E_i = t^s_{im^s} + \max_{v \in [1, D_i]} \left(\frac{t^p_{im^p(v)}}{D_i} + D_i K_{im^p(v)}\right) \tag{3.4}$$

Em que m^s representa o tipo do núcleo que executa o segmento sequencial e $m^p(v)$ representa o tipo de núcleo que executa a v^{th} *thread* paralela.

Agregando o cálculo do tempo de execução a proposta, formalmente tem-se um modelo de otimização multiobjetivo:

$$
\begin{aligned}
&\text{Min} && \Sigma E_{ij} \\
&\text{Max} && Q_j \\
&\text{Restrito a:} && \\
& && Q_j \geq s_{ij} + r_{ij} \quad \forall i \in [1, N] \\
& && \sum_{j=1}^{M} r_{ij} P_{ij} + (Q_j - r_{ij}) P_j^{ocioso} \leq P_{orçamento} \\
& && \sum_{j=1}^{M} Q_j A_j \leq A_{orçamento}
\end{aligned}
$$

Em que Q_j representa a quantidade de núcleos do tipo j e deve ser pelo menos maior ou igual ao número de núcleos desse tipo usados pelas *threads* sequenciais e paralelas de qualquer aplicação, s_{ij} e r_{ij} respectivamente. A dissipação de potência de pico do núcleo j ($j \in [1, M]$) executando uma das *threads* paralelas da aplicação i ($i \in [1, N]$) é denotada por P_{ij} e inclui potência dinâmica e de fuga. Quando um núcleo está ocioso, isto é, sempre que nenhum segmento é mapeado para ele, sua dissipação de corrente de fuga é indicada por P_j^{ocioso}. A área do núcleo do tipo j é A_j, incluindo o pipeline e caches privados. Como objetivo é maximizar o desempenho, assume-se que cada núcleo é executado em seu maior nível de tensão e frequência.

Na Figura 3.11 é possível observar uma visão geral da ferramenta. A infraestrutura do Hades é baseada na definição do problema da síntese arquitetural de MPSoCs heterogêneos com *dark silicon*, passa pela biblioteca de núcleos de propósito geral, pelo conjunto de aplicações *multi-threaded* e pelo orçamento de energia e área.

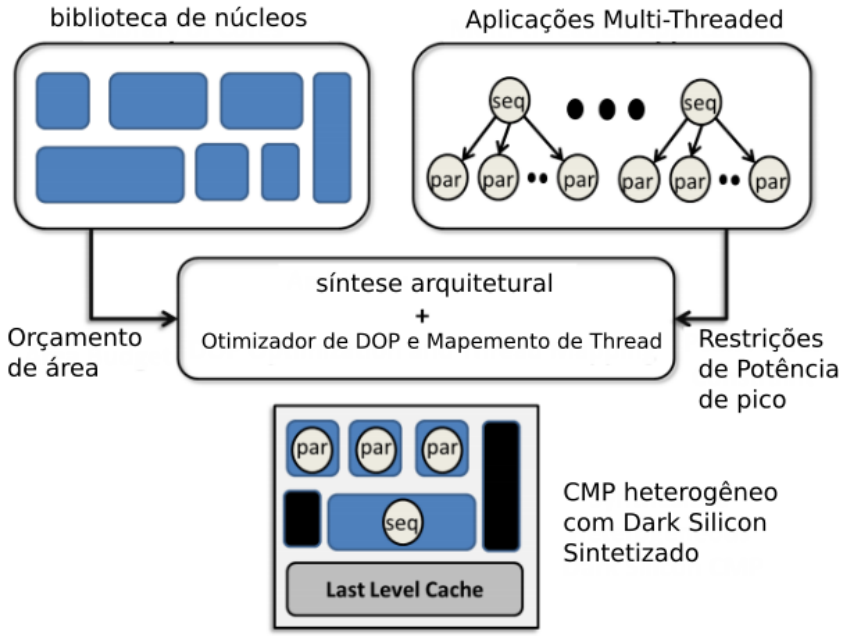


Figura 3.11: Visão geral da *ferramenta* Hades [Turakhia et al., 2013].

Inicialmente a ferramenta se concentrou em otimizar apenas os núcleos e, para isto, foi estabelecido que todos os núcleos compartilham uma cache global única de último nível *Last Level Cache* (LLC)), com uma penalidade de acesso uniforme e com mesmo tamanho para todos os projetos. Além disso, foi fixado valores para componentes de fora do núcleo, por exemplo, largura de banda, quantidade de controladores de memória, entre outros.

Os experimentos foram realizados com base na biblioteca de núcleos da Tabela 3.4, e foram executados com as ferramenta Sniper e *Multicore Power, Area and Timing* (McPAT), com nó tecnológico de $22nm$, tensão de alimentação de $1.0V$, cache L3 com $32MB$ e latência de acesso uniforme, memória principal *Dynamic Random Access Memory* (DRAM) de $1GB$ que é acessada usando um controlador de memória com largura de banda de $7,6Gbps$ e grau de paralelismo de 32 para cada aplicação. Como restrições para os experimentos considerou-se $180mm^2$ de área para potências de pico de $40W$, $60W$ e $80W$.

Tabela 3.4: Parâmetros microarquiteturais da biblioteca de núcleos [Turakhia et al., 2013].

Parâmetros da plataforma	**Valores**
Quantidade de núcleos de	1,2,4
Tamanho Janela ROB	8,16,32,64
Cache L1 I/D	64,128,256 KB
Cache L2 (privada)	256 KB
Frequência	2.5,3.5,4.5GHz

Os resultados dos experimentos foram comparados com projetos de CMPs homogêneos e com grau de paralelismo fixo em 16, e observou-se que o CMP heterogêneo obteve um ganho de até 60% em desempenho nas soluções com restrições de 40W (Figura 3.12).

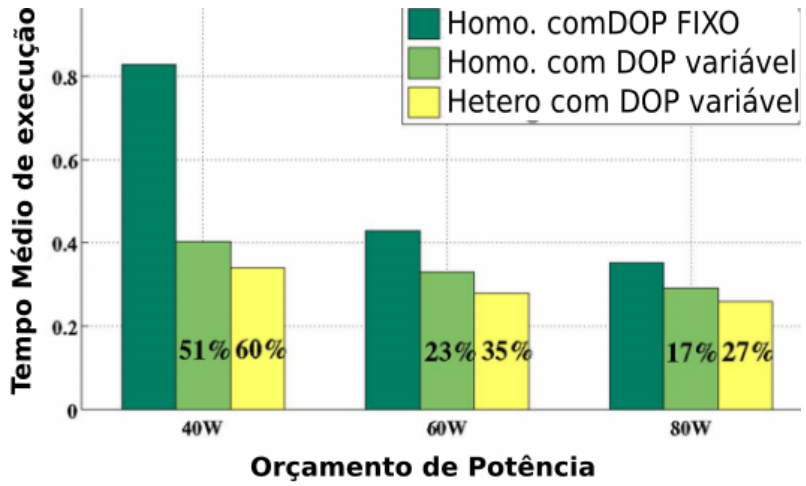


Figura 3.12: Tempo de execução para as diferentes configurações arquiteturais e orçamento de energia [Turakhia et al., 2013].

3.5 Considerações Finais do Capítulo

Neste capítulo foram expostos os conceitos sobre o problema de exploração de espaço de projeto. Técnicas, algoritmos e ferramentas que norteiam soluções para para esse problema também foram apresentadas objetivando delimitar o estado da arte de soluções e, também, identificar possibilidades ainda abertas para desenvolvimento de novos trabalhos. Ressalta-se que este trabalho visa a exploração arquitetural do espaço de projeto de projetos de plataformas computacionais, considerando a presença (e área) de *dark silicon*. A solução proposta é baseada na exploração de núcleos de processamento da plataforma e possui fluxo de execução parecido com a proposta do *framework* Hades, possibilitando assim tanto minimizar a densidade de potência de um chip quanto maximizar o desempenho. Entretanto, ressalta-se que a solução proposta está centrada na redução da densidade de potência de forma a eliminar ou minimizar a área em *dark silicon* de uma plataforma. A proposta será apresentada com maior nível de detalhes no próximo capítulo.

Capítulo 4

Exploração do Espaço de Projeto Ciente de *Dark Silicon*

Diante da necessidade de incremento do poder de processamento nos processadores modernos e devido às restrições de projetos com tecnologia abaixo de $90nm$, torna-se essencial que dispositivos eletrônicos atuais sejam projetados considerando *dark silicon*, viabilizando assim a possibilidade de explorar o espaço de projeto para utilização eficiente da área do circuito.

Isto posto, este capítulo discorre sobre o desenvolvimento do trabalho com enfoque no projeto de técnicas e algoritmos que culminaram em uma ferramenta computacional para exploração do espaço de projeto de sistemas *multicore* cientes de *dark silicon*. Cumpre consignar que em busca literária não foram encontradas ferramentas de exploração de espaço de projeto que considerem e otimizem explicitamente a área do chip em *dark silicon*.

4.1 Considerações Iniciais

No passado, os projetos de processadores que seguiam os princípios da escala de integração de Dennard tinham como foco o aumento de processamento disponível em um chip. Porém, com a escala dos transistores limitada pela corrente de fuga, o projeto de processadores modernos passou por atualizações que envolvem a tomada de decisão considerando múltiplos objetivos como área, potência consumida e desempenho, tornando os projetos de sistemas *multicore* e MPSoC mais complexos.

Neste sentido, é essencial, nos projetos de processadores modernos, mecanismos automáticos que possibilitem a associação de múltiplos parâmetros que atendam a múltiplos objetivos e ainda, estimem *dark silicon* e apoiem a exploração de espaço de projeto de modo a mitigá-lo eficazmente. Diante deste desafio, inicialmente foi desenvolvido no Laboratório de Sistemas Computacionais de Alto Desempenho (LSCAD) da Universidade Federal de Mato Grosso do Sul a ferramenta MultiExplorer [Devigo et al., 2015] para projeto de sistemas *multicore* e MPSoCs. Essa ferramenta permite a definição de parâmetros do projeto em alto nível para a obtenção de estimativas de frequência, potência e área do chip, possibilitando o desenvolvimento e avaliação de *benchmarks*, metodologias e componentes de hardware.

Para atender aos novos requisitos de projeto cientes das restrições energéticas dos transistores e por consequência do *dark silicon*, trabalhos publicados pelo grupo de pesquisadores do LSCAD [Silva et al., 2015] [Santos et al., 2016] [Santos, 2016] estenderam a ferramenta MultiExplorer para, a partir de estimativas de parâmetros físicos identificar a densidade de potência e estimar o percentual de *dark silicon* sobre a área de um chip.

Considerando também a necessidade de alternativas para mitigar o *dark silicon*, apresenta-se aqui uma nova extensão na ferramenta Multiexplorer que realiza a exploração de espaço de projeto para a busca de novas configurações que intentem minimizar o *dark silicon* sem perder desempenho e mantendo a área da configuração inicial proposta pelo projetista. Desta forma, nas seções seguintes serão apresentadas as características da ferramenta MultiExplorer, a metodologia para estimativa de *dark silicon*, as técnicas e algoritmos adotados para a exploração de espaço de projeto e o novo fluxo de execução da ferramenta.

4.2 MultiExplorer

A ferramenta MultiExplorer consiste em um ambiente para automatização e experimentação de projetos de plataformas computacionais em que um projetista pode definir, em alto nível, parâmetros arquiteturais e obter, a partir das ferramentas incluídas nesse ambiente, valores para parâmetros físicos e de desempenho, possibilitando a análise do projeto sob diversas perspectivas.

Inicialmente, o MultiExplorer [Devigo et al., 2015] utilizava as ferramentas MPSoC-Bench [Duenha et al., 2014], um conjunto de ferramentas de simulação composto por MPSoCs escaláveis que permite o desenvolvimento e a avaliação de novas ferramentas, metodologias, software e componente de hardware em alto nível e McPAT [Li et al., 2009, 2013], uma ferramenta para estimativa de parâmetros físicos (área, potência e *clock*).

Porém, objetivando oferecer ao projetista uma gama maior de ferramentas com relação às plataformas passíveis de experimentação, estimativa do *dark silicon* e a exploração de espaço de projeto, foram implementadas interfaces para a integração de novas ferramentas e algoritmos de simulação. Com essas interfaces foram adicionados suporte para utilização integrada das ferramentas de simulação Multi2Sim (que suporta a simulação da arquitetura x86) e Sniper (que oferece a simulação de núcleos heterogêneos), além da adição da estimativa de *dark silicon* e algoritmos para a exploração de espaço de projeto.

A Figura 4.1 apresenta o fluxo completo de modelagem e simulação do MultiExplorer, e os passos compreendidos nesse fluxo serão detalhados nas subseções a seguir.

4.2.1 Descrição da Plataforma

O usuário do MultiExplorer especifica parâmetros iniciais para a execução da ferramenta, porém, como o MultiExplorer é uma ferramenta para definição de parâmetros em alto nível, alguns deles podem ser omitidos e serão definidos automaticamente com base nas características arquiteturais do projeto. As referências arquiteturais pré-definidas melhoram o nível de abstração para o usuário, permitindo que ele escolha, por exemplo, apenas o conjunto de instruções do projeto (MIPS, x86, entre outros) e as características detalhadas

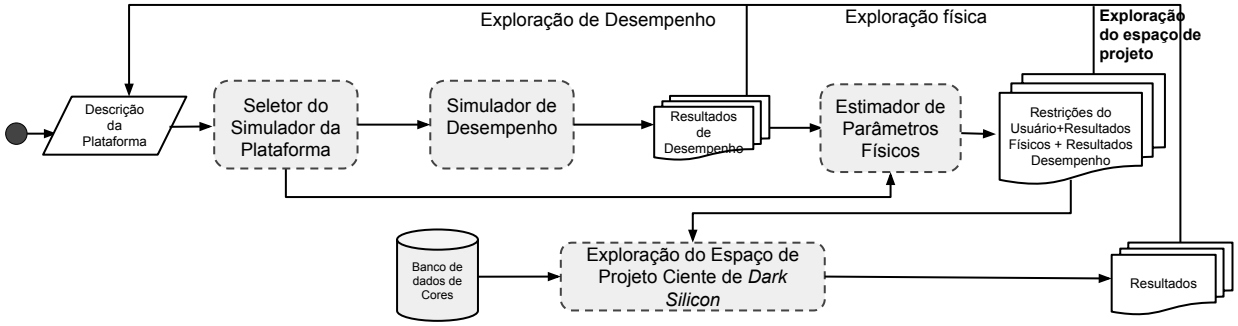


Figura 4.1: Fluxo para definição do projeto utilizando MultiExplorer.

da arquitetura definida serão buscadas em uma base de dados de parâmetros arquiteturais. Dentre os parâmetros de alto nível utilizados na execução da plataforma podemos citar:

1. Informações sobre a arquitetura do sistema como modelo do processador, estágios de *pipeline*, número de núcleos, arquitetura de *caches* e políticas;
2. Arquitetura do conjunto de instruções;
3. Processo tecnológico;
4. Restrições de projeto: área, potência e *clock*;

4.2.2 Seleção da Plataforma e Simulação de Desempenho

Devido à complexidade, para o usuário, em escolher uma ferramenta de simulação adequada às características do seu projeto, o MultiExplorer possui um módulo seletor que levará em conta características especificadas pelo usuário no projeto da plataforma computacional para definir qual a ferramenta de simulação mais adequada. Após a seleção do simulador, os parâmetros definidos pelo usuário, bem como as configurações padrões da ferramenta, são repassados ao simulador selecionado que executa o projeto. Em seguida, os resultados são exibidos ao projetista e alguns são repassados para a ferramenta de estimativa física, por exemplo, número de ciclos, estatísticas de acesso à *cache*, quantidade total de instruções por categoria (inteiros, pontos flutuantes e saltos), estatísticas de predição de saltos.

O fluxo de descrição e seleção da plataforma, bem como da simulação de desempenho é especificado na Figura 4.2 e as ferramentas de simulação suportadas pelo MultiExplorer, atualmente, são detalhadas a seguir.

A modelagem da plataforma e a simulação de desempenho são avaliados pelo MultiExplorer que pode fazer uso de uma das três ferramentas para simulação de plataformas:

1. **MPSoCBench:** Desenvolvido por Duenha et al. [Duenha et al., 2014, Duenha and Azevedo, 2013] é um conjunto de ferramentas de simulação composto por MPSoCs escaláveis que permite o desenvolvimento e a avaliação de novas ferramentas, metodologias, software e componentes de hardware e suporta quatro diferentes arquiteturas

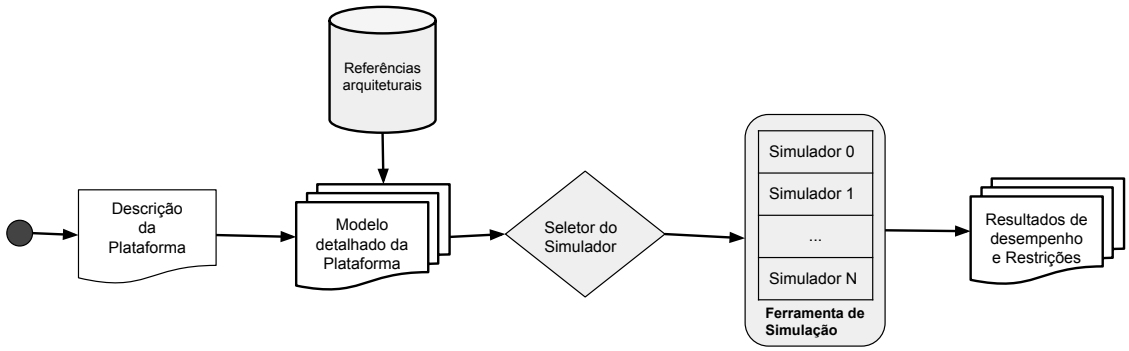


Figura 4.2: Fluxo de simulação de desempenho do MultiExplorer.

de conjunto de instruções (ISAs) (PowerPC, MIPS, *Scalable Processor Architecture* (SPARC) e ARM) em diversas plataformas MPSoCs configuráveis e escaláveis com 1, 2, 4, 8, 16, 32 ou 64 núcleos, com diferentes interconexões (*routers* ou NoC). A ferramenta possibilita ao projetista definir parâmetros em alto nível que serão considerados na avaliação de desempenho, retornando como resultado de execução as estatísticas de simulação que incluem número de instruções, tempo de simulação, velocidade de simulação, número de acessos à memória, número de acessos a *locks* e acessos aos dispositivos de interconexão.

2. **Multi2Sim:** O Multi2Sim [Ubal et al., 2012] é um *Framework* de código fonte aberto, modulares e altamente configuráveis de simulação para *Central Processing Unit* (CPU) e *Graphics Processing Unit* (GPU) que permite avaliação funcional de novas arquiteturas apresentando suas estimativas de desempenho. Ele inclui modelos para o projeto de plataformas computacionais baseadas em processadores superscalares, *multithreadeds*, além de CPUs integradas com GPUs. A ferramenta suporta três tipos de processadores (x86, ARM e MIPS) e dois tipos de GPUs (AMD Evergreen e NVIDIA Fermi). A ferramenta consiste em dois componentes principais de software que interagem entre si, o simulador funcional assemelha-se à execução de um programa em processador x86, interpretando o programa binário e reproduzindo dinamicamente seu comportamento no nível do conjunto de instruções, já o simulador arquitetural mapeia as instruções x86 do simulador funcional e acompanha a execução das estruturas de *hardware* do processador com suporte ciclo-a-ciclo.

3. **Sniper:** Criado no trabalho de Trevor et al. [Carlson et al., 2011], como resultado da extensão do simulador paralelo *Graphite* [Miller et al., 2010], trata-se de um simulador *multicore* escalável e paralelizável que utiliza simulação intercalada, que permite simular sistemas com múltiplos núcleos e múltiplos processadores em um nível de abstração mais alto do que a simulação com precisão de ciclo (*cycle-accurate*), oferecendo um melhor custo-benefício em termos de precisão e tempo de execução. Esta ferramenta suporta apenas projetos de processados que utilizam o conjunto de instruções x86.

4.2.3 Estimativa Física

As estatísticas de atividades dinâmicas (resultados das simulações dos projetos) são essenciais para a geração de parâmetros físicos adequados. Assim, a integração de ferramentas de alto nível (simulação) e baixo nível (estimativas físicas) para projetos de plataformas computacionais é necessária, pois, enquanto uma ferramenta fornece a simulação funcional (MPSoCBench, Multi2Sim e Sniper), a ferramenta de estimativa física (no caso do Multi-Explorer utiliza-se a ferramenta McPAT) estima e retorna informações sobre o projeto físico (área dos componentes de hardware, temporização, potência, tecnologia de implementação) da plataforma.

O McPAT é um *framework* de modelagem de arquiteturas de processadores com foco em potência, área e *clock* (simultaneamente) considerando projetos *multicores* e *manycores* de tecnologias que variam de $90nm$ a $22nm$. O *framework* pode especificar parâmetros de projeto de baixo nível, como interconexões e caches, deixando para o usuário a especificação de métricas de alto nível como taxa de *clock* alvo de otimização (utilizado como restrição de projeto), permitindo, assim, que detalhes de baixo nível sejam ignorados pelo usuário.

Além disso, o McPAT modela além de potência dinâmica, que é crítico em novas tecnologias, todos os três tipos de dissipação de potência: dinâmica, estática e curto-circuito para fornecer uma visão completa envolvendo energia dos processadores *multicores*, através de projeções de tecnologia do ITRS.

Para o cálculo de área, o McPAT utiliza a metodologia analítica do *Cache Access and Cycle Time Model* (CACTI) [Wilton and Jouppi, 1996, Li et al., 2011] para modelar área de portas lógicas e as estruturas regulares, incluindo matrizes de memória (por exemplo, *Random Access Memory* (RAM) e *Content-Addressable Memory* (CAM)) e interligações. Para estruturas mais complexas como unidades de execução (ULA, *Floating Point Unit* (FPU) e etc.) utiliza modelagem empírica [Gupta et al., 2000], que usa dados gerados pela análise de vários núcleos comerciais e padronizando os tamanhos para qualquer nó tecnológico abordado. A ferramenta calcula a área de cada componente individualmente e a área final do processador consiste na soma delas.

Os principais componentes do McPAT são: os modelos hierárquicos de potência, área e sincronização; o otimizador para determinar as implementações no nível do circuito, no caso de não serem definidas pelo usuário e otimizar componentes baseados em estruturas de memória (interconexões e *arrays*); e a representação de chip interno que impulsiona a análise da potência, área e tempo. A maioria dos parâmetros da representação interna de chips, como a capacidade de cache e largura de emissão do núcleo estão diretamente definidos pelos parâmetros de entrada.

O McPAT integra modelos de potência, área e tempo organizados em três níveis hierárquicos: a) Nível de arquitetura, com modelos de processadores *multicores/manycores*, composto de componentes principais como núcleos, NoCs, caches, controladores de memória e *clock*; b) No nível de circuito, fios hierárquicos, *arrays* lógicos complexos e redes de *clock*, suportando modelagens de temporização, área dinâmica, curto circuito e corrente de fuga para cada um dos dispositivos previstos no ITRS – CMOS, *Silicon-On-Insulator* (SOI) e transistores *double-gate*; c) No nível de tecnologia, os dados do ITRS obtidos através da ferramenta

MASTAR [ITRS, 2007] são utilizados para calcular os parâmetros físicos de dispositivos e fios, tais como a resistência, capacitância e densidades de corrente.

A Figura 4.3 ilustra o fluxo da estimativa física do MultiExplorer, em que inicialmente recebe informações da descrição da plataforma e os valores de desempenho, executa e apresenta como resultados finais, de maneira hierárquica, para cada componente, suas partes principais e constituintes somadas à suas respectivas áreas e potência. Em uma das extensões realizadas no MultiExplorer, foi agregado ao McPAT a estimativa de *dark silicon* [Silva et al., 2015] [Santos et al., 2016] [Santos, 2016], que utiliza os valores de área e potência de um circuito de referência e do *die* obtidos pela ferramenta, conforme detalhado na subseção seguinte.

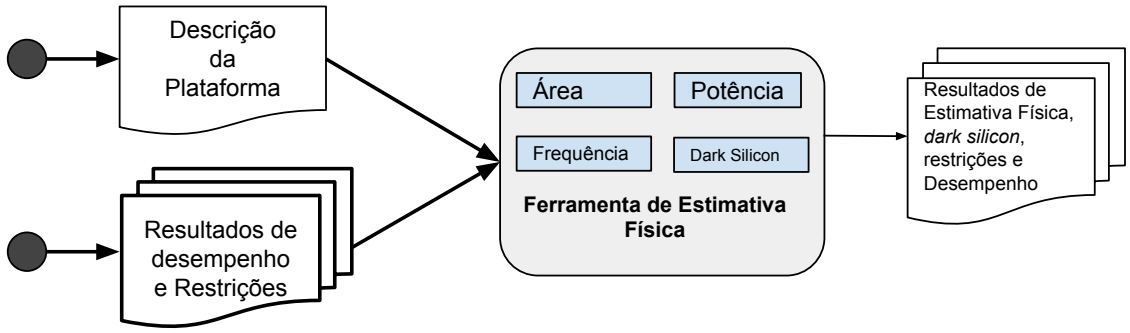


Figura 4.3: Fluxo de estimativa física do MultiExplorer.

A maior contribuição da integração entre as ferramentas de simulação e estimativas físicas no MultiExplorer é a de permitir a associação rápida e automática das plataformas virtuais aos parâmetros físicos de projeto oferecidos pelo McPAT, de maneira a disponibilizar uma estrutura de software que, a partir de diferentes níveis de abstração, possibilite a modelagem, simulação e estimativas das características de componentes físicos para uma plataforma MPSoC [Devigo et al., 2015], bem como a estimativa de *dark silicon* no projeto [Silva et al., 2015, Santos et al., 2016, Santos, 2016].

4.2.4 Estimativa de *Dark Silicon*

Conforme já abordado no Capítulo 2, o *dark silicon* ocorre devido às limitações físicas nos transistores abaixo de $90nm$. Nesses transistores, há um aumento da corrente de fuga superior ao observado em transistores fabricados em processos acima de $90nm$. Como consequência desse aumento, a densidade de potência sobre o chip (quantidade de potência consumida por mm^2 de área) sofre um incremento que extrapola os limites de dissipação energética do projeto.

Entretanto, nota-se na literatura que a maioria dos trabalhos são conservadores e mantêm a densidade de potência de todo o chip, inclusive o *dark silicon*. Essas abordagens

desconsideram a possibilidade de substituir esta área em *dark silicon* por um *hardware* com densidade de potência menor, viabilizando a diminuição da área comprometida do *chip*.

Isto posto, a abordagem adotada pelo MultiExplorer estima a área em *dark silicon* e possibilita a utilização de sua área. Desta forma, a estimativa de *dark silicon* em projetos de processadores consiste em identificar a densidade de potência sobre um chip e compará-lo com a densidade de potência de um projeto de referência (processador projetado com tecnologia de $90nm$). Se houver aumento da densidade de potência em projetos baseados nas mesmas características físicas e de desempenho, mas com diferentes processos tecnológicos, então, seguindo a escala Pós-Dennard (Tabela 2.2), isso gerará aumento da potência total sobre o *chip* e, considerando que os projetos possuem a mesma área, um incremento da densidade de potência indicando, então, a presença de *dark silicon* no projeto.

As equações a seguir mostram como a estimativa de *dark silicon* é realizada no MultiExplorer. A equação 4.1 utiliza características de área e potência e representa a diferença de densidade de potência atual (dp_{atual} em $Watts/mm^2$) e a densidade de potência base (dp_{base}), logo Δ_{dp} representa a potência excedente por mm_2.

$$\Delta_{dp} = (dp_{atual} - dp_{base}) \tag{4.1}$$

Caso $\Delta_{dp} > 0$ então há presença de *dark silicon* no projeto. Para obter a potência total excedente na área do chip (t_{pe}), multiplica-se $\Delta_{dp} > 0$ pela área do chip (a), conforme equação 4.2. t_{pe} é o que pretende-se suprimir para mitigar o *dark silicon*.

$$t_{pe} = \begin{cases} \Delta_{dp} \times a & \text{if } \Delta_{dp} > 0 \\ 0 & \text{senão} \end{cases} \tag{4.2}$$

Para estabelecer a área do *chip* que representa o *dark silicon* é preciso converter o total de potência excedente t_{pe}, para isto, se faz necessário a utilização de um elemento fundamental do *chip* (por exemplo, um transistor) que será usado como referência de área (a_c) e potência (p_c). A equação 4.3 indica como obter a_{ds}.

$$a_{ds} = \frac{t_{pe}}{p_c} \times a_c \tag{4.3}$$

Para validar e avaliar a metodologia de estimativa de *dark silicon* foram realizados experimentos baseados em processadores reais (microarquiteturas Intel Pentium e AMD K8/K10). Nos projetos com processadores AMD K8/K10 com nós tecnológicos de $90nm$ até $32nm$ observou-se até 8,26% de *dark silicon* sobre a área do *chip*, e para experimentos utilizando o aumento de frequência segundo a escala de Dennard obteve-se até 16,65% de *dark silicon* em projetos baseados no processador Intel Smithfield e 9,52% utilizando o processador AMD Santa Rosa [Silva et al., 2015, Santos et al., 2016, Santos, 2016].

Se compararmos a metodologia de estimativa de *dark silicon* apresentada, utilizando a densidade de potência como parâmetro de estimativa, com a estimativa Pós Dennard para *dark silicon* em nós tecnológicos de 45nm a 8nm, obtemos as curvas apresentadas na figura 4.4.

Para efeitos de comparação utilizou-se circuitos de referência com 10% e 30% mais densidade de potência do que a média do *chip*. As curvas DP-10% E DP-30% foram construídas adotando a escala Pós-Dennard para a evolução da potência (S^2), Quantidade de Transistores (S^2), Frequência (S) e capacitância ($\frac{1}{S}$) entre os nós tecnológicos.

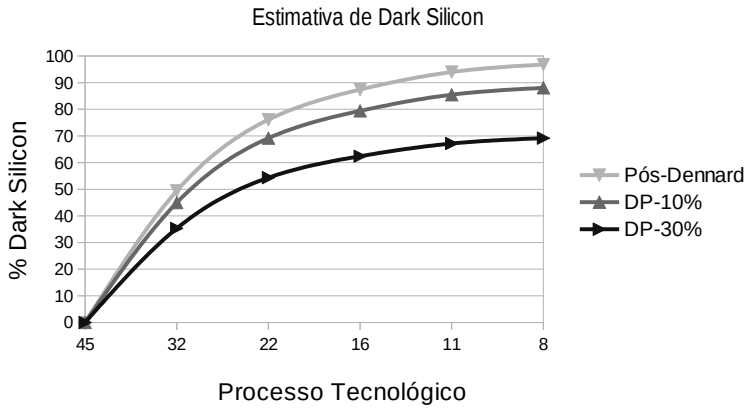


Figura 4.4: Comparação entre a proposta de estimativa e a escala Pós-Dennard para estimativa de *dark silicon*.

Cumpre consignar que na estimativa de *dark silicon*, a ferramenta MultiExplorer calcula a potência excedente de um projeto atual com base em um projeto de referência de 90nm livre de *dark silicon*. Outrossim, como o fluxo de estimativa de parâmetros físicos utiliza a ferramenta McPAT, que não fornece em seus resultados a área estimada de um transistor, foi necessário a adoção de um circuito de referência para estimar a área de *dark silicon*.

Esses experimentos possibilitaram avaliar que mesmo a abordagem de estimativa de *dark silicon* proposta ser considerada otimista com relação a outras da literatura, ainda há uma lacuna no que se refere a ferramentas automatizadas capazes de estimar e utilizar a área de *dark silicon* de maneira eficiente.

4.3 Exploração de Espaço de Projeto Ciente de *Dark Silicon*

Considerando a existência de *dark silicon* em processos tecnológicos abaixo de 90nm, torna-se obrigatório controlar o aumento de potência ao longo desses processos, a fim de manter os custos de projeto de acordo com o orçamento e o aumento do desempenho. Uma forma de controlar esse aumento de potência é identificar a área em *dark silicon* e fornecer novas soluções arquiteturais.

Ao observar a densidade de potência dos componentes do chip, observa-se que os dispositivos com maior densidade de potência são os núcleos. Assim, a base da exploração do espaço de projetos consiste em otimizar o chip de modo que núcleos com alta densidade de potência sejam substituídos por núcleos com menor densidade e, preferencialmente, com desempenho

equivalente. A Figura 4.5 apresenta o fluxo de execução dessa estratégia de exploração de projeto.

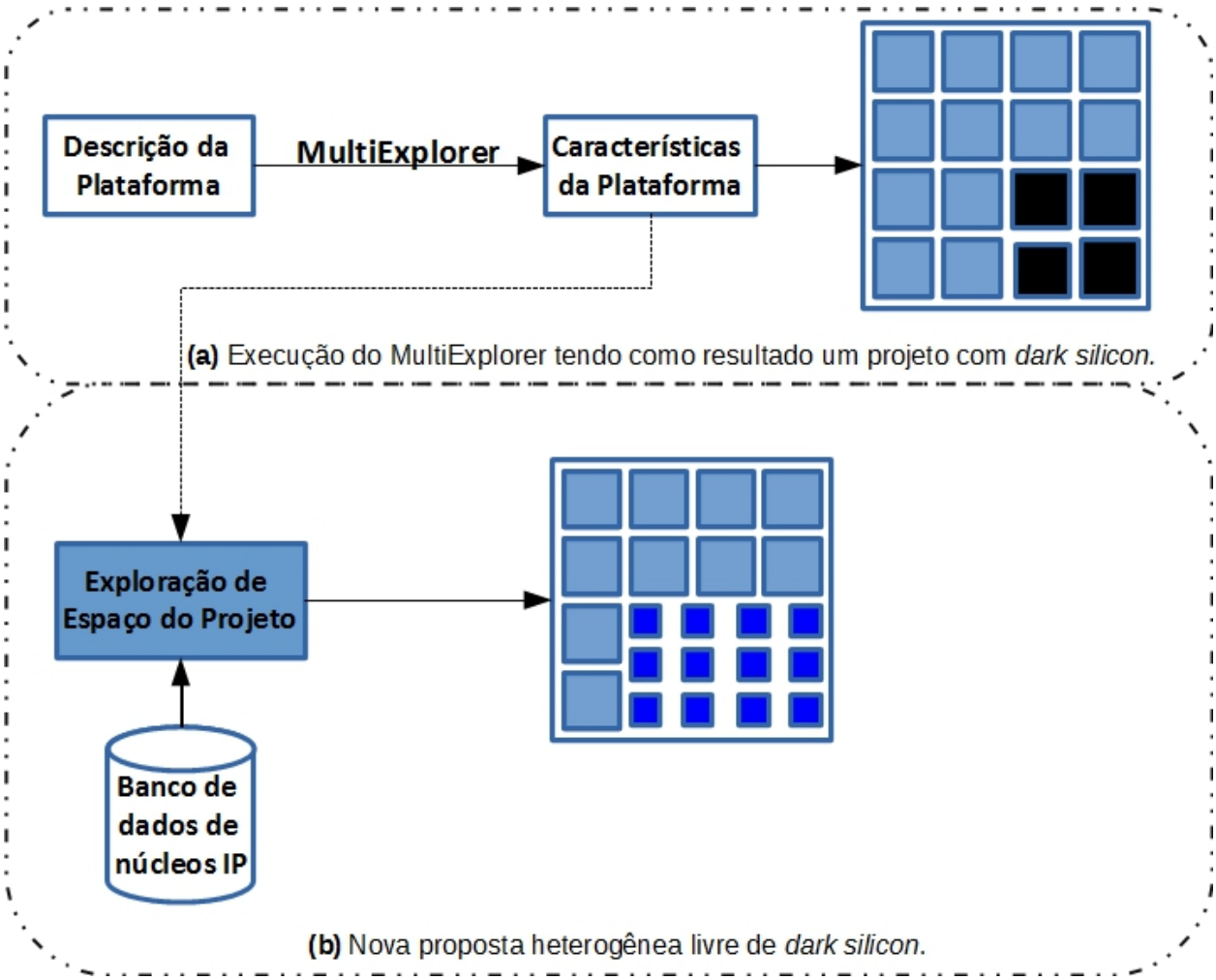


Figura 4.5: Fluxograma do módulo de exploração de espaço de projeto ciente de *dark silicon*.

Nota-se que o problema a ser resolvido consiste em determinar uma plataforma computacional viável que minimize o *dark silicon* (plataforma heterogênea livre de *dark silicon*). Logo, observa-se que a modelagem do problema apresenta características multiobjetivo, uma vez que há dois objetivos conflitantes entre si, minimização da densidade de potência e maximização do desempenho. As restrições a serem consideradas são físicas (área, potência, tipo de núcleo, entre outros) e do usuário (quantidade de núcleos originais, quantidade máxima de núcleos, entre outros). Outro ponto a ser considerado é a origem dos núcleos que substituirão os núcleos com alta densidade de potência. Para isto, criou-se um banco de dados de núcleos (banco de núcleos de propriedade intelectual (*Intellectual Property* (IP)) que serão utilizados como alternativas arquiteturais durante a exploração de espaço de projeto.

4.3.1 Definição do Problema e Modelo

A fim de modelar o problema multiobjetivo, sujeito às restrições de área, potência total do chip, tipos de núcleos e outras, o usuário do ambiente MultiExplorer pode informar algumas dessas restrições a fim de guiar o processo de exploração de espaço do projeto. Além disso, os dados utilizados para elaboração do modelo de otimização consistem em saídas dos módulos de simulação e estimativas físicas do MultiExplorer. Para padronizar os termos utilizados neste trabalho, os núcleos do projeto inicial definido pelo usuário serão indicados como "núcleos originais". O termo "núcleos IP" diz respeito aos núcleos de propriedade intelectual do banco de dados, que servirão como alternativas para exploração do projeto.

Assim, as informações oriundas do fluxo do MultiExplorer para o módulo de exploração de espaço de projeto são:

- Quantidade de núcleos(n): representa o número de núcleos de processamento do projeto original;
- Área do núcleo original (a_o): representa a área de um núcleo de processamento;
- Potência do núcleo original (p_o): corresponde a potência de pico de um núcleo do projeto original;
- Desempenho do projeto original (d_o): representa o desempenho de todo o projeto original e é obtido pela equação 4.7.

A caracterização dos núcleos IP que consistem em oportunidades para exploração do projeto, é composta por :

1. Tipo do núcleo IP (T_{ip}): refere-se a identificação de cada núcleo constante no banco de dados de núcleos IP;
2. Área do núcleo IP (a_{ip}): diz respeito a área de cada núcleo de um determinado tipo do banco de dados de núcleos;
3. Potência do núcleo IP (p_{ip}): representa a potência de pico do núcleo IP de cada tipo;
4. Desempenho do núcleo IP (d_{ip}): retrata o desempenho (equação 4.7) para um projeto com um núcleo do referido tipo de núcleo.

Como a intenção é criar novos projetos heterogêneos com base no projeto original, deseja-se que possíveis soluções sejam capazes de possuir em sua configuração tanto núcleos originais como núcleos IP. Uma possível solução para o problema é identificada com três tipos de informações:

1. Quantidade de núcleos originais (n_o);
2. Tipo do núcleo IP (T_{ip});
3. Quantidade de núcleos IP(n_{ip}).

Para confrontar os resultados obtidos na exploração entre si e com o projeto original é necessário calcular a potência (p_t), a área (a_t), a densidade de potência (dp_t) e o desempenho (d_t) total dos núcleos:

$$p_t = (p_o \times n_o) + (p_{ip} \times n_{ip}) \tag{4.4}$$

$$a_t = a_o \times a_o + a_{ip} \times a_{ip} \tag{4.5}$$

$$dp_t = \frac{p_t}{a_t} \tag{4.6}$$

Para comparar se um projeto tem o desempenho melhor que outro, necessita-se minimizar seu tempo de execução, uma vez que o desempenho é dado por 4.7.

$$Desempenho = \frac{1}{Tempo\ de\ execução} \tag{4.7}$$

Desta forma, obtêm-se o desempenho total de um projeto (d_t) através da equação 4.8.

$$d_t = (d_o \times n_o) + (d_{ip} \times n_{ip}) \tag{4.8}$$

A partir dessas equações, tem-se bem definido os dois objetivos para o problema de exploração de espaço de projeto ciente de *dark silicon*: a minimização da densidade de potência (dp_t) e a maximização do desempenho (d_t).

Considerando a delimitação do espaço de projeto como restrição, tem-se então:

$$\begin{aligned} \text{Min} \quad & dp_t \\ \text{Max} \quad & d_t \\ \text{Sujeito a:} \quad & \\ & a_t^{inf} \leq a_t \leq a_t^{sup} \\ & 1 \leq n_o \leq n - 1 \\ & 1 \leq n_{ip} \leq 2n_{ip}^{max} \end{aligned}$$

Em que a_t^{sup} e a_t^{inf} representam, respectivamente, o limite superior e inferior da restrição de área total do projeto e podem ser definidos pelo usuário. A plataforma resultante deve possuir pelo menos um núcleo original. A quantidade de núcleos IP da plataforma resultante deve ser pelo menos um (para diferenciar da plataforma original) e está limitada pela quantidade total de núcleos IP máxima definidos pelo usuário (n_{ip}^{max}).

4.3.2 Estratégia de Solução

No Capítulo 3 os problemas multiobjetivos foram apresentados com possibilidade de diversas abordagens para sua solução, dentre elas os algoritmos genéticos têm se mostrado uma excelente alternativa por possuir tempo de execução reduzido se comparado com a busca exaustiva.

Este trabalho adotou o algoritmo genético NSGA-II [Deb et al., 2002, 2000] como estratégia para busca de soluções arquiteturais viáveis para resolução do modelo apresentado na seção anterior. Algumas das motivações para utilização dessa estratégia se devem ao fato que é um algoritmo já aplicado em problemas multiobjetivo, com alcance de resultados próximos da solução ótima e com desempenho pouco influenciado pelo problema a ser tratado, se comparado com outras técnicas.

O NSGA-II, assim como outros algoritmos genéticos, fornece apenas o mecanismo geral de busca. Entretanto, para cada problema particular é necessário definir a representação computacional das soluções candidatas (indivíduos), como a população inicial que será gerada, o operador de cruzamento, o operador de mutação e o critério de dominância entre as soluções. A utilização do algoritmo NSGA-II como técnica para resolução de um problema multiobjetivo permite impor restrições que privilegiem determinadas soluções em detrimento de outras, e esta escolha pode ser realizada *a priori* ou *a posteriori*.

Para o problema de exploração de espaço de projeto ciente de *dark silicon*, optou-se pelo uso das duas abordagens, sendo que uma das escolhas *a priori* é implementada usando a dominância restrita proposta na subseção 3.3.2 em que a área do chip é utilizada como restrição. Como resultado da aplicação dessa técnica para resolução do problema de exploração de espaço de projeto, tem-se uma descrição de plataforma que atende às restrições de área do chip, potência dissipada, desempenho da plataforma computacional e mitigação de *dark silicon*.

Especificamente, a restrição à área total a_t é que seus limites superior e inferior (a_t^{sup}, a_t^{inf}) podem variar de acordo com um determinado percentual da área original. Assim, no contexto das soluções preferenciais para o problema, diz-se que uma solução i domina uma solução j se:

1. **Se** $(a_t^{inf} \leq a_t(i) \leq a_t^{sup})$ **e** $(a_t(j) < a_t^{inf}$ ou $a_t(j) > a_t^{sup})$: i **é factível e** j **é infactível**;
2. **Senão se** $(a_t(i) < a_t^{inf}$ ou $a_t(i) > a_t^{sup})$ **e** $(a_t(j) < a_t^{inf}$ ou $a_t(j) > a_t^{sup})$ **e** $((|a_t^{inf} - a_t(i)| + |a_t^{sup} - a_t(i)|) < (|(a_t^{inf} - a_t(j)| + |a_t^{sup} - a_t(j)|))$: i **e** j **são infactíveis, mas** i **possui área** a_t **mais próxima do limite inferior ou superior de área**;
3. **Senão se** $(a_t^{inf} \leq a_t(i) \leq a_t^{sup})$ **e** $(a_t^{inf} \leq a_t(j) \leq a_t^{sup})$ **e** $(dp_t(i) \leq dp_t(j)$ **e** $d_t(i) \geq d_t(j))$ **e** $(dp_t(i) < dp_t(j)$ **ou** $d_t(i) > d_t(j))$: i **e** j **são factíveis e** i **domina** j;

Pode-se observar que o algoritmo NSGA-II não tem seu modelo de execução e funcionamento alterado com essas configurações. Para alcançar essas configurações, altera-se apenas a dominância padrão para a dominância restrita.

Considerando o problema de exploração de espaço de projeto apresentado e a estratégia de solução baseada no algoritmo NSGA-II, considerou-se que a representação de um indivíduo (definição dos genes) é composta por:

1. Quantidade de núcleos originais (n_o);
2. Tipo do núcleo IP (T_{ip});
3. Quantidade de núcleos IP(n_{ip}).

A população inicial é gerada de maneira randômica e indivíduos repetidos são descartados. Para delimitar o espaço de projeto e manter parte da originalidade do projeto inicial foi definida uma faixa de valores na qual o algoritmo randômico pode operar:

- $1 \leq n_o \leq n - 1$;
- $\forall n \in T_{ip}$

Outras configurações definidas para o algoritmo NSGA-II aplicado no problema aqui abordado foram:

- O tipo de cruzamento e mutação escolhidos para este problema são do tipo uniforme, conforme apresentado na subseção 3.3. Esta escolha se deve ao fato de serem estratégias independentes do problema e se apresentam adequadas quando os genes são representados por valores inteiros.
- Como a fronteira de Pareto pode apresentar diversas soluções para encontrar a solução mais adequada para o problema, o usuário pode inserir suas preferências *a posteriori*, informando, por exemplo, que deseja a solução com menor densidade de potência.

A Figura 4.6 apresenta o fluxo desse novo módulo integrado no MultiExplorer. O componente de exploração de espaço de projeto representa as restrições do usuário (da especificação do usuário, desempenho e resultados físicos) como um modelo multiobjetivo (funções objetivas e restrições de projeto). O modelo multiobjetivo e um banco de dados de núcleos IP são usados por um *kernel* de otimização responsável por compor um novo projeto de plataforma que atenda às restrições e objetivos.

Além disso, conforme já apresentado na Subseção 3.3.2, que aborda sobre funcionamento do algoritmo NSGA-II, P_0 refere-se a população inicial que é gerada de maneira aleatória, ordenada com base em não-dominação, classificada de acordo com seu grau de dominação. O algoritmo utiliza um processo de seleção por torneio, cruzamento e mutação para obter a população filha Q_0, ambas de tamanho N. As populações P_0 e Q_0 são unidas para gerar uma população R_0. Após esse procedimento, um novo processo para as próximas gerações é considerado para a criação da *t-ésima* geração, R_t. Agora, as soluções pertencentes ao melhor conjunto não dominado F_1, devem ser enfatizadas com relação a todas as outras soluções combinadas da população. Se o tamanho de F_1 for menor do que N, soluções da fronteira seguinte são escolhidas sucessivamente até que a população esteja completa (tamanho N). A nova população $P_t + 1$ de tamanho N é agora usada para seleção, cruzamento e mutação para a criação da população $Q_t + 1$.

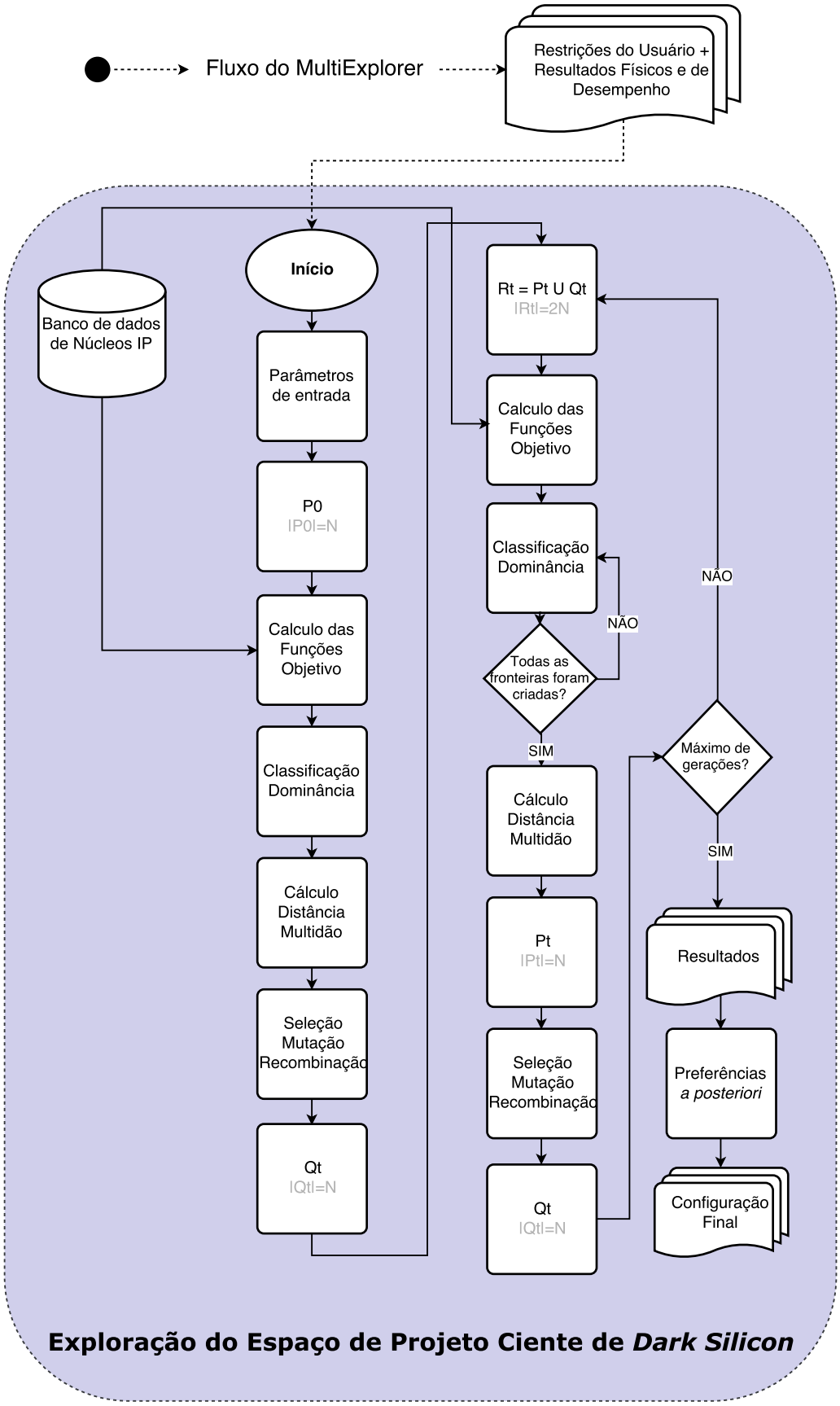


Figura 4.6: Algoritmo NSGA-II integrado ao módulo de exploração de espaço de projeto.

4.4 Considerações Finais do Capítulo

Neste capítulo foi apresentado o desenvolvimento do módulo de exploração de espaço de projeto ciente de *dark silicon* integrado à ferramenta MultiExplorer. A ferramenta MultiExplorer e o modelo de estimativa de *dark silicon* foram apresentados e discutidos. Essa ferramenta e o modelo de estimativa serviram de base para a extensão de funcionalidades, visando a exploração do espaço de projeto ciente de *dark silicon* em plataformas compostas por múltiplos núcleos de processamento. Em pesquisas realizadas na literatura da área, ressalta-se que não foram encontradas ferramentas para projeto de plataformas computacionais que sejam cientes e explorem de maneira automática o espaço de projeto de modo a mitigar eficazmente o *dark silicon*. Essa ausência, aliada a necessidade de utilizar eficientemente a área do chip, reforçam a importância do desenvolvimento deste trabalho. O próximo capítulo apresenta os experimentos e discute os resultados obtidos com projetos de plataformas computacionais que foram submetidas a estratégia proposta de exploração de espaço de projeto.

Capítulo 5

Experimentos e Resultados

Este capítulo apresenta os experimentos realizados para avaliar e validar a metodologia de exploração de espaço do projeto, bem como, o novo ambiente MultiExplorer, ambos propostos e apresentados no Capítulo 4. Os experimentos são baseados em projetos de plataformas computacionais com múltiplos núcleos de processamento. Resultados e discussões são apresentados sobre as estimativas de *dark silicon* e a estratégia proposta para exploração do espaço do projeto que busca minimizar a área em *dark silicon* e a maximização do desempenho da plataforma computacional em tempo de projeto.

5.1 Descrição dos Experimentos

Nos experimentos apresentados a seguir, o processador Intel^{TM} Smithfield 820 foi adotado como referência. Entretanto, diferentes projetos de processadores baseados nas microarquiteturas Intel^{TM}, AMD^{TM} e ARM^{TM} foram utilizados no banco de dados de núcleos empregados pela estratégia implementada de exploração de espaço de projeto. O conjunto de processadores do banco de dados foi escolhido considerando características físicas e arquiteturais específicas como: área, potência dissipada, frequência de *clock*, recursos de paralelismo em nível de instrução, processo tecnológico, conjunto de instruções, organização e hierarquia de memória cache, dentre outros recursos.

Os processadores utilizados no banco de dados foram modelados com apenas um núcleo e seguiram as informações presentes na Tabela 5.1. Entretanto, como uma plataforma computacional composta por mais de um núcleo utiliza outras características arquiteturais como memória RAM, NoC, entre outros, essas foram consideradas a partir das características do processador Smithfield 820.

A modelagem das plataformas seguiu o fluxo de execução do MultiExplorer. Adicionalmente, todas as plataformas foram simuladas adotando o Sniper como simulador de desempenho com as aplicações barnes, cholesky, fft, fmm, lu.cont, ocena.cont, radix e water.nsq do benchmark Splash-2 [Woo et al., 1995]. Importante salientar, também, que o MultiExplorer possui um erro médio de 10% a 20% na estimativa de área, devido ao modelo de estimativa de área e potência adotado na ferramenta McPAT [Li et al., 2009].

Tabela 5.1: Configurações das plataformas utilizadas nos experimentos.

Processador	**Tecnologia** (nm)	**Área Die** (mm^2)	**Frequência** (GHz)	**Total cores**	**Threads**	**Caches**
1 - Smithfield 820	90	206	2.8	2	1	L1-2-16KB, L2-2-1MB
2 - Allendale E2140	65	77	1.6	2	1	L1-4-32KB, L2-1-1MB
3 - Wolfdale E5200	45	82	2.5	2	1	L1-4-32KB, L2-2-2MB
4 - Clarkdale G6950	32	81	2.8	2	1	L1-4-32KB, L2-2-256KB, L3-1-3MB
5 - Santa Rosa 8224 SE	90	230	3.2	2	1	L1-2-64KB, L2-2-1MB
6 - Barcelona 8360 SE	65	285	2.5	4	1	L1-4-64KB, L2-4-512KB, L3-1-2MB
7 - Shanghai 8393 SE	45	258	3.1	4	1	L1-8-64KB, L2-4-512KB, L3-1-6MB
8 - Interlagos 6204	32	316	3.3	4	1	L1-6-64KB, L2-2-2MB, L3-2-8MB
9 - Quark x1000	32	6,5	0.4	1	1	L1-1-16KB
10 - Atom Silvermont	22	5,5	0.5	1	1	L1-1-56KB, L2-1-1MB
11 - ARM Cortex-A53	22	7,2	1.6	1	1	L1-1-80KB, L2-1-512MB
12 - ARM Cortex-A57	22	13	2.0	1	1	L1-1-80KB, L2-1-512MB

As informações de área dos núcleos de processadores 9-12 foram obtidas a partir da modelagem realizada utilizando a ferramenta McPAT. Os fabricantes desses modelos de processadores não apresentaram essas informações nos documentos técnicos (*datasheets*) disponibilizados para os usuários.

A partir das plataformas apresentadas na Tabela 5.1, os seguintes experimentos foram projetados e realizados:

1. Estimativa de *dark silicon* em uma plataforma real, considerando sua evolução tecnológica, incremento de frequência de *clock* de acordo com a escala de Dennard e evolução da quantidade de núcleos por um fator de 2, e outro com a frequência de *clock* constante entre diferentes processos tecnológicos;

2. Exploração do espaço de projeto que resulta em configurações alternativas heterogêneas utilizando núcleos originais e núcleos fornecidos pelo banco de dados. O objetivo deste experimento é identificar e apresentar plataformas de processadores alternativas e eficientes que diminuam a densidade de potência de um projeto de maneira a mitigar o *dark silicon* mantendo (ou melhorando) o desempenho.

A exploração de espaço do projeto foi realizada com o modelo proposto no Subseção 4.3 já integrado ao MultiExplorer, adotando o algoritmo NSGA-II com as seguintes configurações:

Tabela 5.2: Parâmetros do NSGA-II.

Parâmetros	**Valores**
Taxa de Cruzamento	50%
Taxa de Mutação	10%
Tamanho da População	12
Quantidade de Gerações	50000

A taxa de cruzamento utilizada foi de 50%, pois garante que os filhos herdem metade das características de cada pai [Ackley, 2012]. A taxa de mutação para a maioria dos problemas [Cordeiro and Silva-Filho, 2010] é um valor baixo, porém, como o indivíduo modelado neste problema possui poucos genes, uma taxa de mutação tão baixa não garante a

variabilidade da população. O tamanho da população e a quantidade de gerações se basearam exclusivamente em experimentos empíricos realizados neste trabalho.

Para comparar e validar os resultados obtidos com os experimentos, utilizando a estratégia de exploração do espaço de projeto, implementou-se um algoritmo de força bruta (Algoritmo 4). Esse algoritmo realiza todas as combinações possíveis dos parâmetros do indivíduo e retorna as configurações que atendem às restrições de área e densidade de potência, retornando como resposta o indivíduo com maior desempenho.

Para melhor compreensão do algoritmo de força bruta (FB) faz-se necessário retomar algumas convenções apresentadas na subseção 4.3.1, bem como, outras definições expostas a seguir:

- n_o: quantidade de núcleos originais;
- n_o^{min}: quantidade mínima de núcleos originais;
- n_o^{max}: quantidade máxima de núcleos originais;
- n_{ip}: quantidade de núcleos IP;
- n_{ip}^{min}: quantidade mínima de núcleos IP;
- n_{ip}^{max}: quantidade máxima de núcleos IP;
- T_{ip}: tipo do núcleo IP;
- a_t: área total (equação 4.5);
- a_o: área do núcleo original;
- a_{ip}: área do núcleo IP;
- p_t: potência total (equação 4.4) ;
- p_o: potência de pico do núcleo original;
- p_{ip}: potência de pico do núcleo IP;
- dp_t: densidade de potência total (equação **??**);
- d_t: desempenho total do projeto (equação 4.8);
- d_o: desempenho do núcleo original;
- d_{ip}: desempenho do núcleo IP;
- a_l: área limite;
- dp_l: densidade de potência limite;
- S: conjunto de soluções viáveis.

Algoritmo 4: Algoritmo força bruta.

```
1  para todo n_o^min ≤ n_o ≤ n_o^max faça
2      para todo n_ip^min ≤ n_ip ≤ n_ip^max faça
3          para todo i ∈ T_ip faça
4              a_t = a_o × n_o + a_ip × n_ip;
5              p_t = (p_o × n_o) + (p_ip × n_ip);
6              dp_t = p_t / a_t;
7              se a_t≤a_l E dp_t≤dp_l então
8                  d_t = (d_o × n_o) + (d_ip × n_ip);
9                  S = S ∪ {n_o, n_ip, T_ip, a_t, dp_t, d_t}
10 Buscar indivíduo com maior desempenho d_t e sujeito às restrições a_t e dp_t no conjunto
   S de soluções viáveis.
```

A análise do Algoritmo 4 revela a construção de um conjunto S de indivíduos (plataformas viáveis) a partir da busca exaustiva por todo o espaço de soluções. Ao final, busca-se, nesse conjunto, a solução (plataforma) com melhor (maior) desempenho. Observa-se que a complexidade de pior caso para esse algoritmo reside na construção do conjunto S. Esse conjunto terá, no máximo, tamanho $(n_o^{max} \times n_{ip}^{max} \times T_{ip})$. O procedimento de construção requer $(n_o^{max} \times n_{ip}^{max} \times T_{ip})$ passos. Adicionalmente, a busca pelo indivíduo de maior desempenho irá requerer que uma comparação com todos os $(n_o^{max} \times n_{ip}^{max} \times T_{ip})$ elementos do conjunto seja realizada. Finalmente, tem-se então que o algoritmo de força bruta realizará, no pior caso, um total de $2 \times (n_o^{max} \times n_{ip}^{max} \times T_{ip})$ passos. Considerando a notação assintótica de pior caso, esse algoritmo possui complexidade $O(n_o^{max} \times n_{ip}^{max} \times T_{ip})$.

5.2 Experimento 1 - Estimativas de *Dark Silicon* no Processador Smithfield 820

O primeiro experimento se baseou na modelagem do processador Smithfield 820 (Processador 1) em vários processos tecnológicos, incrementando a frequência de *clock* pelo fator de escala entre as gerações de transistores ($S = 1,4$), conforme a escala de Dennard (Seção 2.2), enquanto que a quantidade de núcleos presentes em cada plataforma do novo processo tecnológico aumentou de acordo com 2^p, em que p indica o passo do avanço tecnológico. A Tabela 5.3 apresenta os resultados dessas estimativas sobre o processador Smithfield 820. O objetivo deste experimento consiste em analisar os efeitos de dark silicon em projetos que seguem a escala de integração de Dennard e estimar a área em *dark silicon*, a ser explorada no próximo experimento, em plataformas que seguem a escala de integração Pós-Dennard.

Importante atentar que, conforme já mencionado no Capítulo 4, a estratégia de estimativa de *dark silicon* implementada no MultiExplorer utiliza como referência uma plataforma de processador modelada e avaliada com tecnologia de $90nm$ e freqüência de *clock* original. Como circuito de referência, para as estimativas de área em *dark silicon*, utiliza-se os valores de área e potência da ULA.

Na Tabela 5.3, pode-se verificar que o incremento da frequência de *clock* (linha 4) é determinante para o aumento dos parâmetros físicos de potência e, principalmente, para o

Tabela 5.3: Estimativas físicas com frequência de *clock* incrementada: processador Smithfield.

Avanço tecnológico p	1	2	3	4
Quantidade de núcleos	2	4	8	16
Tecnologia (nm)	90	65	45	32
Frequência (GHz)	2,8	3,9	5,4	7,6
Área do *chip* (mm^2)	203,14	181,69	165,89	160,43
Potência de pico do *chip* (W)	74,28	147,38	248,63	376,296
Potência de fuga do *chip* (W)	4,93	20,46	50,46	25,06
Densidade de potência do *chip* $(\frac{W}{mm^2})$	0,37	0,81	1,50	2,35
Área do núcleo (mm^2)	64,71	35,33	18,25	9,47
Potência de pico do núcleo (W)	73,13	145,83	246,31	375,60
Potência de fuga do núcleo (W)	3,79	18,94	48,17	24,40
Densidade de potência do núcleo $(\frac{W}{mm^2})$	0,57	1,03	1,69	2,48
Área da ULA (mm^2)	0,160	0,112	0,078	0,054
Potência de pico da ULA (W)	0,85	0,88	0,87	0,79
Percentual de *dark silicon* no *chip* $(\%)$	-	5,65	10,13	13,61

aumento relevante no percentual de *dark silicon* do projeto (5,65% - 13,61%). O incremento desse parâmetro é resultado do impacto da dissipação de potência total que foi incrementada pelo aumento da potência de fuga e, como consequência, da potência de pico e da densidade de potência. Os resultados obtidos corroboram propostas encontradas na literatura que apontam que o incremento da frequência de *clock* é um dos fatores determinantes e devem ser afrontados visando mitigar *dark silicon* no projeto.

Deve-se notar que o percentual de *dark silicon* dos projetos (linha 9 das colunas 3-5) implica em uma área do chip que deve ser mantida desligada ou com dissipação de potência reduzida em relação ao restante do chip. Essa área foi calculada a partir do procedimento para estimativa de *dark silicon* apresentado no Capítulo 4. Assim, tomando como exemplo a plataforma com 16 núcleos (coluna 5) e que o circuito de referência (ULA) possui área (a_c) de 0,054 mm^2 (linha 14 da coluna 5) e potência (p_c) de 0,79 W (linha 15 da coluna 5), aplicando a equação 4.3 nota-se que esta plataforma possui $21{,}83mm^2$ de área em *dark silicon*. Considerando que nesta plataforma cada núcleo possui $9{,}47mm^2$ (linha 10 da coluna 5), então, a área em *dark silicon* representa, neste caso, uma região do chip (área do chip) maior que dois núcleos de processamento.

Observa-se que essa estimativa de área do chip em *dark silicon* é otimista, uma vez que, se for considerado a densidade de potência do chip $0,37$ como referência para estimativa de área, será necessário manter em *dark silicon* uma área de $134{,}46mm^2$, o que equivale a $\approx 83\%$ da área do chip desligada em tempo de execução.

A Tabela 5.4 apresenta novos resultados baseando-se no mesmo experimento, mas mantendo constante a frequência de *clock* ao longo dos diferentes processos tecnológicos. Com a frequência inalterada conforme a evolução dos processos tecnológicos, o percentual de *dark silicon* é minimizado comparado com os valores obtidos com a frequência escalada (Tabela 5.3).

Tabela 5.4: Estimativas físicas e frequência de *clock* constante: processador Smithfield.

Avanço tecnológico p	1	2	3	4
Quantidade de núcleos	2	4	8	16
Tecnologia (nm)	90	65	45	32
Frequência (GHz)	2,8	2,8	2,8	2,8
Área do *chip* (mm^2)	203,14	179,23	154,09	157,32
Potência de pico (W)	74,28	100,26	151,51	143,47
Potência de fuga (W)	4,93	20,41	49,03	24,81
Densidade de potência $(\frac{W}{mm^2})$	0,37	0,56	0,98	0,92
Área do núcleo (mm^2)	64,71	35,16	16,93	9,32
Potência de pico do núcleo (W)	73,13	98,70	149,19	142,78
Potência de fuga do núcleo (W)	3,79	18,90	46,73	24,15
Densidade de potência do núcleo $(\frac{W}{mm^2})$	0,57	0,70	1,10	0,96
Área da ULA (mm^2)	0,160	0,112	0,078	0,054
Potência de pico da ULA (W)	0,85	0,88	0,87	0,79
Percentual de *dark silicon* $(\%)$	-	2,46	5,52	3,76

Mesmo mantendo a frequência de *clock* constante ao longo do avanço dos processos tecnológicos, observa-se o aumento da densidade de potência gerada pelo aumento da potência de pico. Este aumento é devido, por sua vez, ao incremento da potência de fuga. O projeto com processo tecnológico de $45nm$ alcançou 5,52% de *dark silicon* o que representa uma área de $8,5mm^2$.

5.3 Experimento 2 - Exploração de Espaço do Projeto Visando Mitigar o Dark Silicon no Processador Smithfield 820

Nos experimentos apresentados a seguir, realiza-se a exploração de espaço do projeto, tomando como referência a plataforma Smithfield 820 modelada com tecnologia de $32nm$ e frequência de $2{,}8GHz$ (coluna 5 da Tabela 5.4). Para isto, utilizou-se o banco de dados de núcleos que armazena a descrição das plataformas apresentadas na Tabela 5.1, também configuradas com a tecnologia de fabricação de $32nm$, gerando as configurações apresentadas na Tabela 5.5.

Como a estratégia proposta para exploração do espaço de projeto ciente de *dark silicon* (DS-DSE) visa diminuir a densidade de potência do chip a partir da combinação de núcleos heterogêneos, a modelagem baseada no algoritmo NSGA-II define um indivíduo composto por núcleos originais e núcleos IP (presentes no banco de dados). Assim, além dos valores do chip apresentados na Tabela 5.1, faz-se necessário, também, utilizar características físicas dos núcleos, conforme apresentado nas linhas 10-13 da Tabela 5.4.

Para mitigar o *dark silicon*, uma vez que esta abordagem visa apenas núcleos, é desejável obter no projeto proposto uma densidade de potência dos núcleos inferior ao do projeto

Tabela 5.5: Configurações das plataformas utilizadas nos experimentos.

Processador	Área do Núcleo a_{ip} (mm^2)	Potência $p_{ip}(W)$	Densidade de Potência $dp_{ip}(\frac{W}{mm^2})$	Desempenho d_{ip}
1	9,32	8,91	0,96	1903
2	13,04	12,83	0,98	1301
3	11,70	13,82	1,18	1800
4	12,00	16,46	1,37	2030
5	12,40	20,65	1,67	2555
6	11,83	10,74	0,91	1805
7	12,23	15,78	1,29	2248
8	12,50	12,60	1,01	2403
9	6,42	1,06	0,17	176
10	5,43	2,51	0,46	355
11	7,18	5,50	0,77	1117
12	1,89	12,13	0,94	1528

original, neste caso $dp_t < 0,96$ (linha 13 e coluna 5 da Tabela 5.4), por outro lado, se o objetivo for eliminar totalmente o *dark silicon*, então a densidade de potência dos núcleos no projeto proposto deve ser $dp_t \leq 0,57$ (linha 13 e coluna 2 da Tabela 5.4). Além disso, deseja-se que o desempenho obtido seja maior que o do projeto original com 16 núcleos ($d_t > 16188$) e que a plataforma respeite a restrição de área total dos núcleos $a_t \leq 149,2$ ($a_t = a_o \times n_o$). Deve-se atentar para o fato que a área total dos núcleos ($a_t = 149,2$) é menor do que a área do chip apresentada na Tabela 5.4 (linha 5 e coluna 5).

A Tabela 5.7 apresenta resultados do experimento com a exploração do espaço de projeto utilizando preferência *a posteriori* de: $dp_t < 0,57$. Por sua vez, na Tabela 5.6 são exibidos os resultados do algoritmo de força bruta com densidade de potência limite de 0,57 (dp_l). Interessante notar que, claramente, não houve qualquer solução livre de *dark silicon* que gerou desempenho superior ($d_t > 16188$) em comparação com a plataforma original. Mesmo assim, optou-se por apresentar os resultados de ambas as técnicas (a estratégia proposta baseada em NSGA-II e o algoritmo de força bruta) para comparar a qualidade das soluções livres de *dark silicon* com relação à proximidade do desempenho requerido.

Pode-se verificar que as soluções apresentadas eliminam totalmente *dark silicon* das plataformas, pois possuem densidade de potência dos núcleos menor que 0,57. Nota-se que a estratégia proposta apresentou 2 soluções similares aos melhores resultados do algoritmo de força-bruta. A diferença de resultados entre as duas abordagens se deve ao fato de que a estratégia baseada no algoritmo NSGA-II procura resolver um problema multiobjetivo, oferecendo soluções que ofereçam a melhor relação entre os objetivos (desempenho e densidade de potência), enquanto que o algoritmo força bruta apresenta os resultados com maior desempenho, considerando as restrições de área e densidade de potência. Considerando o desempenho como objetivo de maior peso para determinar solução ótima, observa-se que o algoritmo de força bruta obteve solução com desempenho 21% superior comparado a melhor solução obtida pelo estratégia baseada em NSGA-II.

Tabela 5.6: Soluções do algoritmo força bruta com restrição $dp \leq 0,57$.

Solução	n_o	n_{ip}	T_{ip}	**Área** (mm^2)	**Densidade de Potência** ($\frac{W}{mm^2}$)	**Desempenho**
FB1	7	13	9	148,73	0,51	15609
FB2	7	12	9	142,31	0,53	15433
FB3	7	11	9	135,89	0,55	15257
FB4	7	10	9	129,47	0,56	15081
FB5	6	14	9	145,83	0,47	13882
FB6	6	13	9	139,41	0,48	13706
FB7	6	12	9	132,99	0,50	13530
FB8	3	22	10	147,43	0,56	13519
FB9	6	11	9	126,57	0,52	13354
FB10	6	10	9	120,15	0,53	13178
FB11	3	21	10	142,00	0,56	13164
FB12	6	9	9	113,73	0,55	13002
FB13	3	20	10	136,57	0,56	12809
FB14	3	19	10	131,14	0,57	12454
FB15	2	24	10	148,97	0,52	12326
FB16	5	15	9	142,92	0,42	12155

Tabela 5.7: Soluções da proposta com restrição $dp \leq 0,57$.

Solução	n_o	n_{ip}	T_{ip}	**Área** a_t (mm^2)	**Densidade de Potência** dp_t ($\frac{W}{mm^2}$)	**Desempenho** d_t
DS-DSE1	1	21	9	144,14	0,22	5599
DS-DSE2	2	20	9	147,04	0,27	7326
DS-DSE3	5	15	9	142,90	0,42	12155
DS-DSE4	1	25	10	145,07	0,49	10778
DS-DSE5	2	24	10	148,96	0,52	12326

A Tabela 5.9 apresenta os resultados da abordagem proposta, definindo como preferência a *posteriori* densidade de potência menor do que 0,96. A Tabela 5.8 mostra os resultados do algoritmo força bruta com densidade de potência limite (dp_l) igual a 0,96.

Tabela 5.8: Soluções do algoritmo força bruta com restrição $dp < 0,96$.

Solução	n_o	n_{ip}	T_{ip}	**Área** a_t (mm^2)	**Densidade de Potência** dp_t $(\frac{W}{mm^2})$	**Desempenho** d_t
FB1	15	1	11	147,05	0,95	29662
FB2	15	1	10	145,30	0,94	28900
FB3	14	2	11	144,91	0,94	28876
FB4	15	1	9	146,29	0,92	28721
FB5	14	1	6	142,38	0,95	28447
FB6	12	5	11	147,80	0,91	28421
FB7	13	2	6	144,88	0,95	28349
FB8	12	3	6	147,39	0,95	28251
FB9	14	1	12	143,44	0,96	28170
FB10	13	3	11	142,76	0,93	28090
FB11	14	1	2	143,59	0,96	27943
FB12	13	2	12	147,00	0,95	27795
FB13	14	1	11	137,73	0,95	27759
FB14	14	3	10	146,84	0,90	27707

Tabela 5.9: Soluções da proposta com restrição $dp < 0,96$.

Solução	n_o	n_{ip}	T_{ip}	**Área** (mm^2)	**Densidade de Potência** $(\frac{W}{mm^2})$	**Desempenho**
DS-DSE1	4	20	10	145,88	0,58	14712
DS-DSE2	9	10	9	148,08	0,61	18887
DS-DSE3	11	7	9	147,46	0,72	22165
DS-DSE4	10	10	10	147,50	0,77	22580
DS-DSE5	12	5	9	143,94	0,78	23716
DS-DSE6	6	12	11	142,08	0,84	24822
DS-DSE7	8	10	11	146,36	0,86	26394
DS-DSE8	10	7	11	143,46	0,89	26849
DS-DSE9	14	2	9	143,32	0,89	26994
DS-DSE10	14	3	10	146,77	0,90	27707
DS-DSE11	12	5	11	147,74	0,91	28421
DS-DSE12	15	1	9	146,22	0,92	28721
DS-DSE13	14	2	11	144,84	0,94	28876
DS-DSE14	15	1	10	145,23	0,94	28900

Observa-se que todos os resultados apresentados na Tabela 5.9 mitigam *dark silicon*, sendo que do resultado DS-DSE2 ao DS-DSE14, além de mitigar, melhoram o desempenho, se comparado com a plataforma original ($d_t > 16188$). Para ratificar que os resultados

obtidos nos experimentos foram satisfatórios quanto a mitigação do *dark silicon*, tomando-se como base a densidade de potência da solução II da Tabela 5.9, considerando apenas a alteração dos núcleos proposta nesta abordagem, tem-se uma redução de cerca de 2,5 vezes o percentual de *dark silicon* no projeto, obtendo apenas 1,45% do chip em *dark silicon*. Importante notar também que a diferença entre a solução ótima de desempenho do algoritmo de força-bruta e a melhor solução obtida pelo algoritmo genético foi de $2,5\%$.

Visando comparar a qualidade das soluções obtidas entre as duas abordagens (força bruta e o algoritmo NSGA-II), assim como o desempenho, o banco de dados de núcleos foi estendido, a partir das configurações dos 12 núcleos iniciais, para conter 100, 1000, 2000, 5000 e 10000 núcleos. A Figura 5.1 apresenta os resultados de desempenho e densidade de potência das melhores soluções (maximizam desempenho e minimizam densidade de potência) obtidas pelo algoritmo de força bruta e do NSGA-II utilizando o banco de dados com 10000 núcleos e tendo como restrições $d_t > 16188$ e $dp < 0,57$.

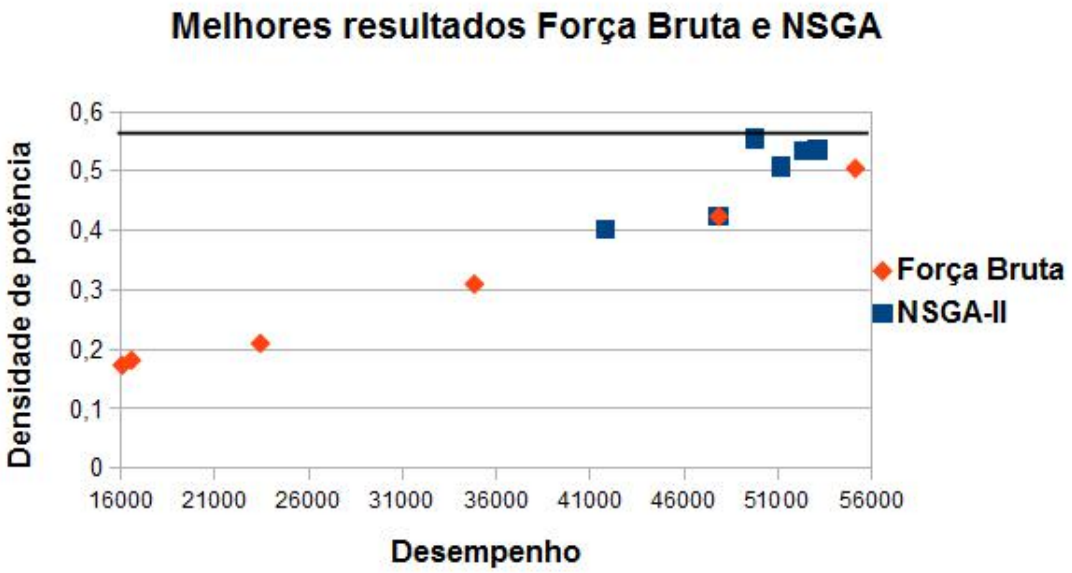


Figura 5.1: Comparação entre soluções obtidas pelo algoritmo de força bruta e NSGA-II.

Pode-se observar que as soluções obtidas pelo algoritmo força bruta constituem em soluções ótimas (para cada ponto de densidade de potência) por maximizar o desempenho e minimizar a densidade de potência. A melhor solução do algoritmo força bruta (o último ponto da Figura 5.1) foi comparada com a melhor solução do NSGA-II e observou-se que as diferenças de desempenho e densidade de potência são pequenas. O melhor desempenho da plataforma NSGA-II é apenas 3,7% menor do que o melhor desempenho da plataforma obtida no força bruta e a melhor densidade de potência atingida no NSGA-II é 6,2% menor do que a melhor solução obtida no algoritmo força bruta.

omo o algoritmo NSGA-II executa um conjunto de etapas de acordo com o modelo de otimização, o tempo de execução depende do tamanho da população e do número de objetivos. Por outro lado, o algoritmo força bruta avalia todos os núcleos do banco de dados de núcleos, de modo que, se o banco de dados aumentar, o desempenho do algoritmo força bruta irá diminuir. Para exemplificar esta projeção, os tempos de execução dos algoritmos,

em cada uma das configurações do banco de dados de núcleos, são apresentados na Figura 5.2. Nota-se que o NSGA-II supera o desempenho do algoritmo força bruta a partir de um banco de dados com 4000 núcleos. Cabe mencionar que para estes experimentos utilizou-se um computador com núcleo Intel Core I3 -3110M dual core, 2,40 GHz e 4GB de memória RAM.

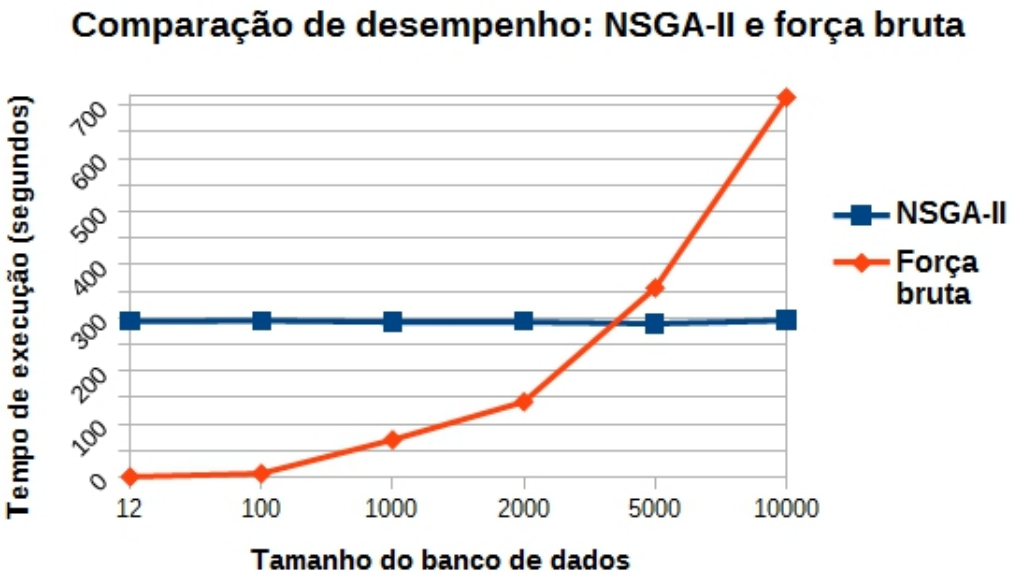


Figura 5.2: Melhores resultados obtidos com o algoritmo NSGA-II

5.4 Considerações Finais do Capítulo

Os resultados obtidos permitem observar que a utilização da abordagem proposta é apta para encontrar configurações viáveis que atendem ao compromisso entre os diferentes objetivos. As configurações alternativas obtidas como resultados dos experimentos foram avaliados empiricamente e ratificaram as estimativas físicas e de desempenho estimadas pela estratégia aqui proposta. Finalmente, ressalta-se que, a partir da pesquisa na literatura técnica da área, a estratégia aqui proposta é a primeira proposição de solução para o problema de exploração de espaço de projeto em plataformas computacionais que considera explicitamente o *dark silicon* como variável do modelo de otimização. Essa abordagem mostra-se promissora com vistas ao projeto automatizado das futuras plataformas computacionais que, inevitavelmente, considerarão a heterogeneidade dos núcleos para manter os parâmetros físicos e de desempenho atendidos de acordo com as demandas dos usuários. O próximo capítulo apresenta as conclusões finais do trabalho, as contribuições principais e sugestões para desenvolvimento de trabalhos futuros.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho propôs uma metodologia e técnica para solucionar o problema de exploração do espaço de projeto em plataformas computacionais, compostas por núcleos de processamento, ciente de *dark silicon*. Como resultado direto, tem-se uma infraestrutura de software testada e validada capaz de, em tempo de projeto da plataforma, identificar a existência de *dark silicon*, estimar a área do projeto a ser considerada na exploração e aplicar técnicas para explorar essa área visando oferecer novas alternativas arquiteturais.

Ressalta-se que a modelagem teórica que sustenta a proposição das técnicas para solucionar o problema, baseia-se em um modelo de otimização multiobjetivo proposto para explorar o espaço de projeto, objetivando minimizar o *dark silicon* (minimizar a densidade de potência) e maximizando o desempenho, enquanto obedece às restrições de área, potência dissipada e demais restrições impostas pelo usuário da plataforma. Para resolver o modelo multiobjetivo se adotou uma técnica de resolução baseada no algoritmo genético NSGA-II. Este algoritmo foi projetado e implementado para se adequar às características do problema apresentado neste trabalho e esta nova infraestrutura de software foi integrada ao fluxo da ferramenta MultiExplorer e denominada de componente DS-DSE (*Dark Silicon aware Design Space Exploration*).

Para validar a nova estratégia de exploração de espaço de projeto ciente de *dark silicon*, experimentos foram realizados considerando plataformas computacionais reais comercializadas pela indústria de semicondutores. A validação e avaliação dos resultados da técnica proposta neste trabalho foram realizadas comparando com os resultados obtidos por um algoritmo de força bruta para o mesmo problema. A análise e comparação desses resultados permitiram concluir que a abordagem proposta é apta para encontrar soluções viáveis que atendem ao compromisso entre os diferentes objetivos e obedeça às restrições do problema. Os resultados também permitiram concluir que a estratégia de buscar alternativas arquiteturais tomando como referência o núcleo de processamento, resultando assim em plataformas heterogêneas, mostra-se promissora para mitigar *dark silicon* e ainda maximizar o desempenho computacional.

Diante do exposto, este trabalho cumpriu os objetivos inicialmente apresentados ao propor uma infraestrutura de software automatizada que possibilita a exploração de espaço de projeto ciente de *dark silicon*, promovendo novas alternativas arquiteturais formadas por

núcleos heterogêneos para mitigar o *dark silicon*. Alguns resultados e contribuições obtidas com o desenvolvimento deste trabalho são:

- Proposição de um modelo de otimização para exploração de espaço de projeto ciente de *dark silicon* em plataformas computacionais;
- Proposição de uma técnica, baseada no algoritmo genético NSGA-II, para resolução do problema. A partir das diversas pesquisas realizadas em trabalhos relacionados encontrados na literatura da área, entende-se que este trabalho é o primeiro a propor um modelo de otimização e técnica para resolução do problema de exploração de espaço de projeto ciente de *dark silicon*.
- Integração da técnica proposta junto ao fluxo de execução da ferramenta Multiexplorer, oferecendo assim uma ferramenta para o projeto de plataformas computacionais que possibilita estimar e quantificar *dark silicon* e prover mecanismos para explorar o projeto;
- Publicação de artigos em conferência nacional e internacional:
  - SILVA, A. C. S. ; BIGNARDI, T. ; PALMA, E. ; ALVES, R. ; HAYASHI, C. ; SANTOS, R. Identificação Automática de Dark Silicon em Processadores Multicore. In: Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD), 2015, Florianópolis.
  - BIGNARDI, T., SANTOS, A. C. S., SANTOS, R., DUENHA, L., AZEVEDO, R., ORDONEZ, D. On the Dark Silicon Automatic Evaluation on Multicore Processors. In: 28th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2016, Los Angeles-USA.
- Ressalta-se também que está sendo realizada a escrita de artigo científico para ser submetido em Abril/2017 para a revista *International Journal of Parallel and Distributed Computing*. Este novo artigo será submetido a partir do convite realizado pelos editores da revista que tiveram como referência o artigo publicado e apresentado no evento SBAC-PAD 2016.

Como propostas para desenvolvimento de trabalhos futuros são sugeridas:

- Projeto e implementação de novas técnicas e algoritmos para exploração de espaço de projeto e comparação dos resultados com o algoritmo implementado neste trabalho;
- Projeto e extensão da base de dados de núcleos de forma a abranger a exploração de outros componentes da plataforma computacional;
- Projeto, avaliação e utilização de modelos de desempenho mais precisos considerando plataformas com núcleos heterogêneos;
- Estender o modelo de otimização para contemplar mais restrições físicas (temperatura de operação, consumo energético, entre outros), arquiteturais (conjunto de instruções, unidades funcionais específicas, quantidade de memória, interconexões) e do usuário (faixa de desempenho, tecnologia).

Referências Bibliográficas

David Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer Science & Business Media, 2012.

Jason Allred. *Designing, optimizing, and sustaining heterogeneous chip multiprocessors to systematically exploit dark silicon*. PhD thesis, UTAH STATE UNIVERSITY, 2013.

Jason Allred, Sanghamitra Roy, and Koushik Chakraborty. Designing for dark silicon: a methodological perspective on energy efficient systems. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 255–260. ACM, 2012.

ARM. big.LITTLE Technology. `http://www.arm.com/products/processors/technologies/biglittleprocessing.php`, 2015. Acessado: 01-09-2015.

Giuseppe Ascia, Vincenzo Catania, Alessandro Di Nuovo, Maurizio Palesi, and Davide Patti. Efficient design space exploration for application specific systems-on-a-chip. *Journal of Systems Architecture*, 53(10):733–750, 2007.

Vishal Aslot, Max Domeika, Rudolf Eigenmann, Greg Gaertner, Wesley Jones, and Bodo Parady. SPEComp: A new benchmark suite for measuring parallel computer performance. In *OpenMP Shared Memory Parallel Programming*, pages 1–10. Springer, 2001.

Thomas Bäck, David Fogel, and Zbigniew Michalewicz. *Evolutionary computation 1: basic algorithms and operators*, volume 1. CRC Press, 2000.

Christian Bienia, Sanjeev Kumar, Jaswinder Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

Christopher Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

Shekhar Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, 1999.

Shekhar Borkar. Design perspectives on 22nm CMOS and beyond. In *Proceedings of the 46th Annual Design Automation Conference*, pages 93–94. ACM, 2009.

Eduardo Briao. Métodos de exploração de espaço de projeto em tempo de execução em sistemas embarcados de tempo real soft baseados em redes-em-chip. Master's thesis, Universidade Federal do Rio Grande do Sul, 2008.

David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the International Symposium on Computer Architecture*. ACM, 2000.

Doug Burger and Todd Austin. The simplescalar tool set, version 2.0. *ACM SIGARCH Computer Architecture News*, 25(3):13–25, 1997.

Zhihua Cai, Wenyin Gong, and Yongqin Huang. A novel differential evolution algorithm based on ε-domination and orthogonal design method for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 286–301. Springer, 2007.

Trevor E Carlson, Wim Heirman, and Lieven Eeckhout. Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 52. ACM, 2011.

Daniel Cavicchio. Adaptive search using simulated evolution. Technical report, University of Michigan, 1970.

Koushik Chakraborty and Sanghamitra Roy. Topologically homogeneous power-performance heterogeneous multicore systems. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2011.

Pedro Coelho. A influência do efeito de modulação de canal na distorção harmônica em espelhos de corrente CMOS. Technical report, Universidade Federal do Rio de Janeiro, 2009.

Gustavo Colletta. *Uma arquitetura de modulação sigma-delta assíncrona em ultra-baixa potência para aplicações biomédicas.* PhD thesis, Universidade Federal de Itajubá, 2015.

Felipe Cordeiro and Abel Silva-Filho. NSGAII applied to unified second level cache memory hierarchy tuning aiming energy and performance optimization. In *11th Symposium on Computing Systems (WSCAD-SCC)*, pages 64–71. IEEE, 2010.

Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving From Nature*, pages 849–858. Springer, 2000.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Towards a quick computation of well-spread pareto-optimal solutions. In *Evolutionary multi-criterion optimization*, pages 222–236. Springer, 2003.

Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary computation*, 13(4):501–525, 2005.

Kenneth DeJong. An analysis of the behavior of a class of genetic adaptive systems. *Ph. D. Thesis, University of Michigan*, 1975.

R. H. Dennard, J. Cai, and A. Kumar. A perspective on today's scaling challenges and possible future directions. *Solid-State Electronics*, 5(4):518–525, 2007.

Robert Dennard, Fritz Gaensslen, Hwa-Nien Yu, Leo Rideout, Ernest Bassous, and Andre Leblanc. Design of ion-implanted mosfets with very small physical dimensions. *IEEE Journal of Solid-Circuits*, pages 256–267, 1974.

Rodrigo Devigo, Liana Duenha, Rodolfo Azevedo, and Ricardo Santos. Multiexplorer: A tool set for multicore system-on-chip design exploration. In *IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 160–161. IEEE, 2015.

Liana Duenha and Rodolfo Azevedo. Mpsocbench: A benchmark suite for evaluating multiprocessor system-on-chip tools and methodologies. Technical Report IC-13-19, Institute of Computing - State University of Campinas, 2013.

Liana Duenha, Marcelo Guedes, Hyggo Almeida, Matheus Boy, and Rodolfo Azevedo. MPSoCBench: A toolset for MPSoC system level evaluation. In *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, pages 164–171. IEEE, 2014.

Hadi Esmaeilzadeh, Emily Blem, Renee St Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. *IEEE Micro*, 39(3):122–134, 2012.

Carlos Fonseca and Peter Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28(1):26–37, 1998.

David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

Nathan Goulding, Jack Sampson, Ganesh Venkatesh, Saturnino Garcia, Joe Auricchio, Jonathan Babb, Michael B Taylor, and Steven Swanson. Greendroid: A mobile application processor for a future of dark silicon. In *Hot Chips 22 Symposium (HCS), 2010 IEEE*, pages 1–39. IEEE, 2010.

Nathan Goulding-Hotta, Jack Sampson, Ganesh Vekatesh, Saturnino Garcia, Joe Auricchio, Po-Chao Huang, Manish Arora, Siddhartha Nath, Vikram Bhatt, Jonathan Babb, Steven Swanson, and Michael Bedford. The greendroid mobile application processor: An architecture for silicon's dark future. *IEEE Micro*, 31(2):86–95, 2011.

Nathan Goulding-Hotta, Jack Sampson, Qiaoshi Zheng, Vikram Bhatt, Joe Auricchio, Steven Swanson, and Michael Bedford Taylor. Greendroid: An architecture for the dark silicon age. *IEEE Micro*, pages 100–105, 2012.

Shashank Gupta, Stephen Keckler, and Doug Burger. Technology independent area and delay estimates for microprocessor building blocks. *Citeseer*, 2000.

Mark Hansen, Hakan Yalcin, and John Hayes. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test of Computers*, 16(3):72–80, 1999.

Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 30(4):4–15, 1999.

John Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

Wei Huang, Mircea Stant, Karthik Sankaranarayanan, Robert Ribando, and Kevin Skadron. Many-core design from a thermal perspective. In *Proceedings of the 45th annual Design Automation Conference*, pages 746–749. ACM, 2008.

Ching Hwang and Abu Masud. *Multiple objective decision making-methods and applications: a state-of-the-art survey*, volume 164. Berlin: Springer, 1979.

Ken-ichi Ikeda, Hajime Kita, and Shigenobu Kobayashi. Failure of pareto-based moeas: does non-dominated really mean near to optimal? In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 957–962. IEEE, 2001.

Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima. Selecting a small number of representative non-dominated solutions by a hypervolume-based solution selection approach. In *IEEE International Conference on Fuzzy Systems*, pages 1609–1614. IEEE, 2009.

ITRS. Semiconductor industries association. model for assessment of CMOS technologies and roadmaps (MASTAR). `http://www.itrs.net/models.html`, 2007.

Yi-Min Jiang, Kwang-Ting Cheng, and Ante Kristic. Estimation of maximum power and instantaneous current using a genetic algorithm. In *Proceedings of the IEEE on Custom Integrated Circuits Conference*, pages 135–138. IEEE, 1997.

PJ Joseph, Kapil Vaswani, and Matthew Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *The Twelfth International Symposium on High-Performance Computer Architecture*, pages 99–108. IEEE, 2006a.

PJ Joseph, Kapil Vaswani, and Matthew Thazhuthaveetil. A predictive performance model for superscalar processors. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 161–170. IEEE Computer Society, 2006b.

Mahmut Kandemir and Alok Choudhary. Compiler-directed scratch pad memory hierarchy design and management. In *Proceedings of the 39th annual Design Automation Conference*, pages 628–633. ACM, 2002.

Eunsuk Kang, Ethan Jackson, and Wolfram Schulte. An approach for effective design space exploration. In *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, pages 33–54. Springer, 2011.

Nam Sung Kim, Todd Austin, David Baauw, Trevor Mudge, Krisztián Flautner, Jie S Hu, Mary Jane Irwin, Mahmut Kandemir, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *computer*, 36(12):68–75, 2003.

Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

Benjamin Lee and David Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. *ACM SIGPLAN Notices*, 41(11):185–194, 2006.

Kwang Lee and Mohamed El-Sharkawi. *Modern heuristic optimization techniques: theory and applications to power systems*, volume 39. John Wiley & Sons, 2008.

Sheng Li, Jung Ahn, Richard Strong, Jay Brockman, Dean Tullsen, and Norman Jouppi. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 469–480. IEEE, 2009.

Sheng Li, Ke Chen, Jung Ahn, Jay Brockman, and Norman Jouppi. CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 694–701. IEEE, 2011.

Sheng Li, Jung Ahn, Richard Strong, Jay Brockman, Dean Tullsen, and Norman Jouppi. The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(1):5, 2013.

Jens Lienig and Krishnaiyan Thulasiraman. A genetic algorithm for channel routing in vlsi circuits. *Evolutionary Computation*, 1(4):293–311, 1993.

Julius Lilienfeld. Electric current control mechanism, July 19 1927. URL `http://www.google.com/patents/CA272437A?cl=pt-BR`. CA Patent 272,437.

Peter Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hallberg, Johan Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002.

Mahesh Mamidipaka and Nikil Dutt. eCACTI: An enhanced power estimation model for on-chip caches. *Technical Report of the Center for Embedded Computer Systems*, pages 04–28, 2004.

John Markoff. Intel's big shift after hitting technical wall. *New York Times*, 17, 2004.

Pinaki Mazumder and Elizabeth Rudnick. *Genetic algorithms for VLSI design, layout & test automation*. Prentice Hall PTR, 1999.

Rick Merritt. Arm cto: power surge could create'dark silicon'. *EE Times*, October 2009.

Kaisa Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Springer, 1998.

Jason Miller, Harshad Kasture, George Kurian, Charles Gruenwald, Nathan Beckmann, Christopher Celio, Jonathan Eastep, and Anant Agarwal. Graphite: A distributed parallel simulator for multicores. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–12. IEEE, 2010.

Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT press, 1998.

Thannirmalai Muthukaruppan, Mihai Pricopi, Vanchinathan Venkataramani, Tulika Mitra, and Sanjay Vishin. Hierarchical power management for asymmetric multi-core in dark silicon era. In *Proceedings of the 50th Annual Design Automation Conference*, page 174. ACM, 2013.

Alon Naveh, Doron Rajwan, Avinash Ananthakrishnan, and Eli Weissmann. Power management architecture of the 2nd generation intel® core™ microarchitecture, formerly codenamed sandy bridge. In *Hot Chips*, volume 23, page 0, 2011.

Roberta Parreiras, João Maciel, and João Vasconcelos. The a posteriori decision in multiobjective optimization problems with smarts, promethee II, and a fuzzy algorithm. *IEEE Transactions on Magnetics*, 42(4):1139–1142, 2006.

Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin Pipe, Thomas Wenisch, and Milo Martin. Computational sprinting. In *Proceedings of IEEE 18th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12. IEEE, 2012.

Jose Renau, Basilio Fraguela, James Tuck, Wei Liu, Milos Prvulovic, Luis Ceze, Smruti Sarangi, Paul Sack, Karin Strauss, and Pablo Montesinos. Sesc simulator, 2005.

Daniel Saab, Youssef Saab, and Jacob Abraham. Automatic test vector cultivation for sequential vlsi circuits using genetic algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(10):1278–1285, 1996.

Luis Santana-Quintero and Carlos Coello. An algorithm based on differential evolution for multi-objective problems. *International Journal of Computational Intelligence Research*, 1(1):151–169, 2005.

Tony Santos. Estimativas de dark silicon em projetos de sistemas multicore. Master's thesis, Universidade Federal de Mato Grosso do Sul, 2016.

Tony Santos, Ana Silva, Liana Duenha, Ricardo Santos, Edward Moreno, and Rodolfo Azevedo. On the dark silicon automatic evaluation on multicore processors. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2016 28th International Symposium on*, pages 166–173. IEEE, 2016.

Robert Schaller. Moore's law: Past, present and future. *IEEE Spectrum*, 34(6):52–59, 1997.

Ana Silva, Tony Bignardi, Edilson de Palma, Rafael Alves, Clara Hayashi, and Ricardo Santos. Identificação automatica de dark silicon em processadores multicore. In *Anais do Simposio de Sistemas Computacionais de Alto Desempenho-WSCAD*, 2015.

Cristina Silvano, William Fornaciari, Gianluca Palermo, Vittorio Zaccaria, Fabrizio Castro, Marcos Martinez, Sara Bocchio, Roberto Zafalon, Prabhat Avasare, Geert Vanmeerbeeck, et al. Multicube: Multi-objective design space exploration of multi-core architectures. In *VLSI 2010 Annual Symposium*, pages 47–63. Springer, 2011.

Hamza Bin Sohail, Mithuna Thottethodi, and TN Vijaykumar. Dark silicon is sub-optimal and avoidable. *Technical Report, Purdue University*, 2011.

Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

Karthik Swaminathan, Emre Kultursay, Vinay Saripalli, Vijaykrishnan Narayanan, Mahmut Kandemir, and Soupayan Datta. Steep-slope devices: From dark to dim silicon. *IEEE Micro*, 33(5):50–59, 2013.

Michael Taylor. Is dark silicon useful?: harnessing the four horsemen of the coming dark silicon apocalypse. In *Proceedings of the 49th Annual Design Automation Conference*, pages 1131–1136. ACM, 2012.

Yatish Turakhia, Bharathwaj Raghunathan, Siddharth Garg, and Diana Marculescu. Hades: Architectural synthesis for heterogeneous dark silicon chip multi-processors. In *Proceedings of the 50th Annual Design Automation Conference*, page 173. ACM, 2013.

Rafael Ubal, Byunghyun Jang, Perhaad Mistry, Dana Schaa, and David Kaeli. Multi2Sim: a simulation framework for CPU-GPU computing. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, pages 335–344. ACM, 2012.

Ganesh Venkatesh, Jack Sampson, Nathan Goulding, Saturnino Garcia, Vladyslav Bryksin, Jose Lugo-Martinez, Steven Swanson, and Michael Bedford Taylor. Conservation cores: reducing the energy of mature computations. *SIGARCH Computuer Architeture News*, 38 (1):205–218, 2010.

Manish Verma and Peter Marwedel. *Advanced Memory Optimization Techniques for Low Power Embedded Processors*, volume 1. Springer, 2007.

Neil West and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspectives*, volume 1. Addison Wesley, 4 edition, 2011.

Steven Wilton and Norman Jouppi. CACTI: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, 1996.

Steven Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Singh, and Anoop Gupta. The splash-2 programs: Characterization and methodological considerations. *ACM SIGARCH Computer Architecture News*, 23(2):24–36, 1995.

Vittorio Zaccaria, Gianluca Palermo, Fabrizio Castro, Cristina Silvano, and Giovanni Mariani. Multicube explorer: An open source framework for design space exploration of chip multi-processors. In *23rd International Conference on Architecture of Computing Systems (ARCS)*, pages 1–7. VDE, 2010.

Chuanjun Zhang and Frank Vahid. Cache configuration exploration on prototyping platforms. In *Proceedings of the 14th IEEE International Workshop on Rapid Systems Prototyping*, pages 164–170. IEEE, 2003.

Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer, 1999.