Camila de Araújo

Modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu

Natal
Julho de 2006

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação

Modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu

Dissertação submetida ao Programa de Pós-Graduação em Sistemas e Computaçãodo Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como parte dos requisitos para a obtenção do grau de Mestre em Sistemas e Computação (MSc.).

Camila de Araújo

Natal, Julho de 2006

Modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu

Camila de Araújo

'Esta Dissertação foi avaliada e considerada adequada para a obtenção do título de Mestre em Sistemas e Computação (MSc.), Área de Concentração em Teoria da Computação, e aprovada pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.'

Prof. Dr. Benjamin René Callejas Bedregal
Orientador

Profª. Drª. Thaís Vasconcelos Batista
Coordenador do Programa

Banca Examinadora:

Prof. Dr. Benjamin René Callejas Bedregal - UFRN
Presidente

Prof. Dr. Ivan Saraiva da Silva - UFRN

Prof. Dr. Regivan Nunes Santiago - UFRN

Prof. Dr. Fernando Gehm Moraes - PUC-RS

AGRADECIMENTOS

Ao Deus criador de tudo e de todos, arquiteto divino de todos os meus projetos toda a minha gratidão.

As outras quatro "meninas", que juntamente comigo integram a "Casa das cinco mulheres": minha mãe, Chagas, minha avó, Dona Antônia e as minhas tias: Cacá e Nevinha, por todo o incentivo, amor e mimos durante toda a minha vida, que foram ainda mais intensos nesses 2 anos em que estive longe. À vocês todo o meu amor, reconhecimento e agradecimento por cada vitória conquistada.

Agradeço aos "Araújo e Silva", minha tia Gercina e seu esposo Jussier, que me receberam em sua casa durante o tempo em que morei em Natal por causa do mestrado, por me acolherem em sua família e cuidarem de mim. Aos meus queridos primos Niê e Fontaine, que me doaram o seu quarto, e mais do que qualquer outra pessoa torceram pelo fim deste mestrado (risos!). Aos meus outros primos Jair e Liana, que além do incentivo me deram as fãs mirins mais fervorosas que eu poderia ter, minhas adoradas primas Gabi e Lia, que me renderam momentos de descontração e alegria. Agradeço ainda aos agregados da família Araújo e Silva: Jaíra, Geane, Ailton e Nadja pelo carinho com que sempre me trataram.

A todos os meus familiares, primos e tios, que torceram pela minha conquista, mesmo não estando perto geograficamente.

Um agradecimento especial a Demóstenes Sena, que me ajudou em todas as fases de composição desta dissertação nos papéis de revisor, debugador, suporte técnico e primeiro-leitor, além de ser um companheiro fiel durante este tempo em que morei em Natal e atender prontamente a todos os meus pedidos. Agradeço a você também, os momentos de diversão, os cinemas no meio da tarde e as partidas de "Air game", obrigada por tudo!

Aos meus professores desde o tempo do "Flautinha Mágica", passando pelos tempos de UERN e agora na pós-graduação da UFRN, a cada um dos mestres que contribuiram para a minha formação intelectual. Agradeço especialmente aos professor Benjamín Bedregal (orientador), Ivan Saraiva (co-orientador), sem os quais não teríamos o sucesso alcançado. Agradeço ainda aos professores David Deharbe, que me ajudou com os BDDs, Regivan Hugo Santiago e Fernando Gehn Moraes, que leram atentamente este texto e contribuiram para a sua melhoria.

Aos funcionários do DIMAp, Seu Gaspar, Graça, Demerson, Vamberto e Dona Selma que atenderam sempre aos meus pedidos, e me ofereceram além de préstimos, amizade e carinho.

Aos amigos do Lab PPgSC, que conseguiram através de uma boa convivência tornar aquele laboratório menos "frio e impessoal"e que foram companhias constantes na busca pelo suquinho nosso de todas as tardes. A vocês: agradecimento, amizade e saudade!

Aos que foram meus alunos nos semestres 2004.2 à 2006.1, enquanto fui professora substituta na UFRN, que foram sempre amigos-compreensivos e entenderam as vezes em que não pude atendê-los prontamente por causa das atividades do mestrado.

A todos os amigos que torceram pelo meu sucesso, e estes não são poucos! Aos amigos de casa, dos tempos de escola, da UERN, do PH, da UFRN e aos melhores amigos o meu agradecimento sincero.

Resumo

As Arquiteturas Reconfiguráveis surgiram no ambiente acadêmico como uma alternativa aos ASICs e aos GGP, mantendo um equilíbrio entre flexibilidade e performance. Este trabalho apresenta uma proposta para a modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu, descrevendo os principais assuntos relativos a esta temática.

A solução proposta consiste em uma modelagem que utiliza uma generalização dos Espaços de Chu, denominada de Chu nets, para modelar as configurações de uma Arquitetura Reconfigurável. Como forma de validar os modelos, foram desenvolvidos e implementados três algoritmos que realizam a composição de células lógicas programáveis, detecção dos vetores de controlabilidade e observabilidade em aplicações para Arquiteturas Reconfiguráveis, que estão modeladas através das Chu nets.

Área de Concentração: Teoria da Computação
Palavras-chave: Arquiteturas Reconfiguráveis; FPGAs; Espaços de Chu; Chu nets; Controlabilidade; Observabilidade.

Abstract

The Reconfigurables Architectures had appeares as an alternative to the ASICs and the GGP, keeping a balance between flexibility and performance. This work presents a proposal for the modeling of Reconfigurables with Chu Spaces, describing the subjects main about this thematic.

The solution proposal consists of a modeling that uses a generalization of the Chu Spaces, called of Chu nets, to model the configurations of a Reconfigurables Architectures. To validate the models, three algorithms had been developed and implemented to compose configurable logic blocks, detection of controllability and observability in applications for Reconfigurables Architectures modeled by Chu nets.

Area of Concentration: Computation Theory
Key words: Reconfigurable Architecture; FPGAs; Chu Spaces; Chu nets; Controllability; Observability.

Sumário

1 Introdução **1**

1.1 Motivação 2

1.2 Objetivos 2

1.3 Organização do Trabalho 2

2 Arquiteturas Reconfiguráveis **4**

2.1 Programabilidade, Configurabilidade e Reconfigurabilidade 5

2.2 FPGAs - Field-Programmable Gate Arrays 6

2.2.1 Tecnologia de Implementação das CLBs 8

2.2.2 Granularidade das CLBs 11

2.3 Software 13

2.4 Testes em Arquiteturas Reconfiguráveis 15

2.4.1 Métricas de Teste 15

3 Espaços de Chu **18**

3.1 Representações de Espaços de Chu 19

3.1.1 Matricial 19

3.1.2 Fórmulas Proposicionais 20

3.1.3 Binary Decision Diagram - BDD 21

3.1.4 Circuitos Combinacionais 21

3.2 Álgebra para Espaços de Chu 23

4 Modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu 26

4.1 Multi-Espaço de Chu 26

4.2 Chu nets 27

4.3 Chu nets como multi-Espaços de Chu 29

4.4 Multi-Chu nets 32

4.4.1 Níveis de detalhamento em multi-Chu nets 33

5 Metodologia 36

5.1 Formato Chu 36

5.2 BDD Library - BDDlib 39

5.3 Representação do formato chu através de BDD 40

5.4 Algoritmos 42

5.4.1 Composição de CLBs 42

5.4.2 Controlabilidade de Arquiteturas Reconfiguráveis 44

5.4.3 Observabilidade de Arquiteturas Reconfiguráveis 46

6 Resultados 50

6.1 Validação 50

6.2 Benchmark ISCAS85 75

7 Considerações Finais 77

7.1 Principais Contribuições 78

7.2 Trabalhos Futuros 78

A Funções da Biblioteca BDD 83

Lista de Figuras

2.1 Relação entre flexibilidade e performance nos paradigmas ASIC, Reconfigurável e GPP 5

2.2 Estrutura interna de um FPGA 7

2.3 Uma LUT de 3 entradas 9

2.4 Multiplexador para a função F com a expansão da variável A 9

2.5 Estrutura geral de um PAL 10

2.6 Um FPGA configurado como um decodificador "2 *to* 4". 11

2.7 Um FPGA configurado como um meio somador. 11

2.8 FPGA configurado como somador completo a partir de 2 meio-somadores. 12

2.9 Somador completo de 8 bits. 13

2.10 Circuito com identificação dos pontos observáveis e controláveis. 16

3.1 Representação matricial de um Espaço de Chu com $K = \{0, 1\}$ 20

3.2 Representação da função $(a \wedge b) \vee (c \wedge d)$ através de BDD 22

3.3 Representação da função $(a \wedge b) \vee (c \wedge d)$ através de BDD com 6 nós . . 22

3.4 Representação de espaços de Chu através de circuitos combinacionais . . 23

3.5 Representação matricial da operação *with* das matrizes A e B, que representam Espaços de Chu 24

3.6 Representação matricial da operação *produto* das matrizes A e B, que representam Espaços de Chu 25

4.1 Um decodificador "2 to 4", representado pelo modelo *Chu net* 28

4.2 Um meio somador, representado pelo modelo *Chu net* 29

4.3 Representação de um FPGA somador completo através de Espaços de Chu 32

4.4 Representação através de Espaços de Chu de um fpga somador de 8 bits . 34

5.1 Estrutura de um arquivo no formato **chu** 37

5.2 Arquivo no formato **chu** com o modelo de um meio somador 38

5.3 Estrutura de uma chunet, definida no arquivo **chu.h** 40

5.4 BDD que representa um evento 41

5.5 Arquitetura que representa parte de um FPGA meio-somador. 43

6.1 Arquitetura reconfigurável - meio somador 52

6.2 BDD resultante da composição da primeira saída do circuito meio somador 53

6.3 BDD resultante da segunda saída do circuito meio somador 54

6.4 Arquitetura reconfigurável - decodificardor $2to4$ 64

6.5 BDD resultante da composição da primeira saída do circuito decodificador $2to4$ 65

6.6 BDD resultante da composição da segunda saída do circuito decodificador $2to4$ 66

6.7 BDD resultante da composição da terceira saída do circuito decodificador $2to4$ 66

6.8 BDD resultante da segunda saída do circuito meio somador 66

Lista de Tabelas

2.1 Valores para obter controlabilidade e observabilidade no ponto 4 mostrado na figura 2.10 17

4.1 A completude de α 30

4.2 Multi-Espaço de Chu decodificador 31

4.3 multi-Espaço de Chu meio somador 32

4.4 Multi-Chu net Somador Completo 33

4.5 Multi-Espaço de Chu Somador Completo 33

4.6 Meio-Somador 34

4.7 Somador 1 35

4.8 Somador 6 35

4.9 Somador 7 sem saída *carry* 35

4.10 Somador de 8 bits 35

6.1 Tabela descritiva da Chu net X. 51

6.2 Tabela descritiva da Chu net meio somador. 52

6.3 Tabela descritiva da Chu net decodificador $2to4$. 64

Capítulo 1

Introdução

As arquiteturas reconfiguráveis surgiram no ambiente acadêmico como um novo dispositivo que permite a configuração de uma estrutura específica de *hardware*. Em dispositivos reconfiguráveis, como os *Field-Programmable Gate Arrays - FPGAs*, é possível atribuir diferentes configurações, e uma vez reconfigurado, o FPGA comporta-se como um circuito de propósito específico.

Um FPGA é um dispositivo constituído de três estruturas básicas: um vetor de células de I/O programáveis, uma estrutura de interconexão programável e um vetor de CLBs (*Configurable Logic Block*), células lógicas programáveis. É através das células de I/O, ou blocos de entrada e saída, que é implementada a interface do sistema, elas são usadas para alimentar os pinos físicos do FPGA. As CLBs são elementos que podem ser configurados para realizar determinadas funções lógicas. Cada CLB é projetada para que possa executar o maior número de funções que suas limitações permitem. A conexão entre as CLB's e os pinos das células de I/O é feita através de linhas rápidas de interconexão, que consiste em uma série de canais horizontais e verticais de roteamento, permitindo o roteamento global de sinais no FPGA.

Neste trabalho, foi utilizado uma extensão de um formalismo conhecido como Espaços de Chu para modelar as configurações para um FPGAs. Os Espaços de Chu são um formalismo matemático tradicionalmente útil para modelagem de concorrência e paralelismo em sistemas computacionais, no entanto, aqui foi escolhido pela facilidade na representação que pode ser feita através de matrizes, fórmulas lógicas proposicionais, BDD *Binary Decision Diagram* e circuitos lógicos, entre outras formas. Para a modelagem de arquiteturas reconfiguráveis será utilizada uma generalização dos Espaços de Chu, denominada de Chu nets, que permite uma representação matemática de uma configuração para uma arquitetura reconfigurável.

A modelagem de sistemas através de formalismos matemáticos é bastante utilizada,

pois permite que características e propriedades do sistema sejam verificadas antes da existência de sua implementação ou até mesmo de um protótipo. Descreve-se aqui um conjunto de algoritmos, que se utilizam da modelagem de arquiteturas reconfiguráveis obtidas com Chu nets para manipular arquiteturas reconfiguráveis, mais especificamente suas configurações.

1.1 Motivação

Na literatura pesquisada, não há trabalhos que utilizem-se de formalismos matemáticos para efetuar a modelagem de arquiteturas reconfiguráveis, especificamente de FPGAs, com o intuito de facilitar a verificação de propriedades do sistema, bem como a sua corretude, a partir do uso de propriedades elementares da matemática.

1.2 Objetivos

Para atender as necessidades apresentadas na seção 1.1, este trabalho apresenta os artifícios que proporcionaram a modelagem e representação de Arquiteturas Reconfiguráveis através de Chu nets, que são uma generalização do formalismo Espaço de Chu. Como forma de validar esta modelagem, objetiva-se também a descrição dos algoritmos de Composição de CLBs, detecção de vetores de controlabilidade e observabilidade de Arquiteturas Reconfiguráveis, quando estas estão modeladas com Espaços de Chu.

Também é propósito deste trabalho apresentar a teoria envolvida na elaboração do trabalho, bem como as atividades que foram desempenhadas para a obtenção dos resultados.

1.3 Organização do Trabalho

De modo a organizar esse trabalho de acordo com seus objetivos, sua estrutura é composta de 7 capítulos. No capítulo 1, fez-se uma introdução ao assunto tratado ao longo do trabalho. Nos capítulos 2 e 3, faz-se uma descrição de conceitos importantes para o entendimento do problema aqui abordado, respectivamente, os capítulos falam sobre Arquiteturas Reconfiguráveis e Espaços de Chu. No capítulo 4 descreve-se toda a teoria envolvida na modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu. Nos capítulos 5 e 6 são demonstrados, respectivamente, a metodologia envolvida na elaboração

do trabalho e os resultados obtidos. Finalmente, no capítulo 7, são feitas algumas considerações finais e apresentados alguns trabalhos futuros que daríam continuidade a este trabalho.

Capítulo 2

Arquiteturas Reconfiguráveis

As arquiteturas reconfiguráveis surgiram no ambiente acadêmico, no final da década de 80 [5] [37], como um novo dispositivo que permite a configuração de uma estrutura específica de *hardware*. Elas são uma alternativa entre os ASICs, *Application Specific Integrated Circuits*[35], e os GPPs, *General-Purpose Processors*[11]. Os ASICs são circuitos integrados destinados à execução de aplicações específicas em *hardware* e são bastante eficazes para executar a aplicação para a qual foram propostos, no entanto, não propiciam flexibilidade, uma vez que após a sua fabricação, o ASIC não pode ser alterado. Os GPPs, processadores de propósito geral, implementam um conjunto de instruções básicas, cuja combinação fornece uma infra-estrutura para a execução de *qualquer* computação sem que haja mudança no *hardware*. O custo dessa flexibilidade é o tempo necessário para que o processador execute cada instrução. A execução envolve etapas de busca, decodificação, busca dos operandos, execução da operação e atualização dos resultados. Entre o ASICs e os GPPs, estão as arquiteturas reconfiguráveis, que buscam um equilíbrio entre performance e flexibilidade, levando em consideração as necessidades do sistema. A idéia é executar computação em *hardware* para incrementar a performance, preservando a flexibilidade das soluções em *software*. A figura 2.1 mostra a relação entre flexibidade e performance nos paradigmas ASIC, Reconfigurável e GPP.

Neste capítulo será abordado o conceito de reconfigurabilidade a partir da discussão sobre programabilidade e configurabilidade, falando ainda sobre as características de *Hardware*, especificamente sobre FPGAs, *Field-Programmable Gate Arrays*, e *Software* implícitas no projeto de arquiteturas reconfiguráveis.

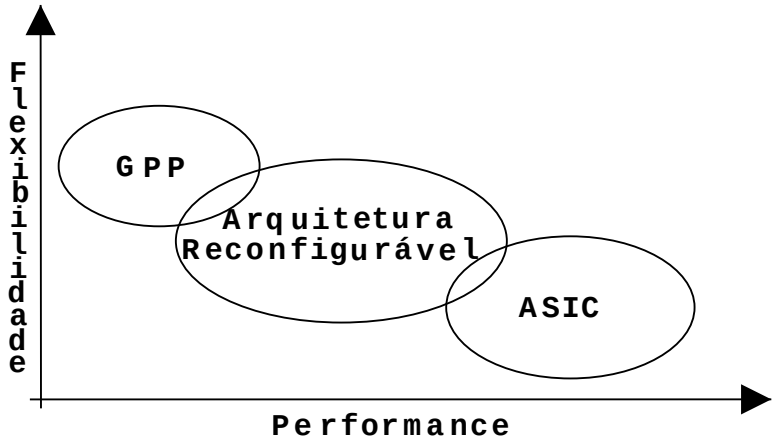

Figura 2.1: Relação entre flexibilidade e performance nos paradigmas ASIC, Reconfigurável e GPP

2.1 Programabilidade, Configurabilidade e Reconfigurabilidade

Uma característica desejável nos sistemas computacionais é que o conjunto de recursos oferecidos possa ser adaptado, através de *software* ou *hardware*, às necessidades específicas dos usuários. A adaptação de sistemas em nível de *software* é bastante disseminada, através dos paradigmas de programação, onde o *hardware* permanece inalterado.

Nos sistemas que utilizam *hardware* de propósito geral, quando uma operação não é diretamente suportada, ela pode ser implementada por uma combinação de operações existentes, o que exige facilidade de programação, ou *programabilidade*, por parte do *hardware*. Essas modificações não implicam em uma reorganização estrutural da arquitetura, são apenas a utilização de recursos que já estavam disponíveis.

A adaptação de sistemas computacionais, baseada em alterações estruturais executadas em *hardware*, está associada à utilização de arquiteturas monolíticas, cujos recursos possibilitam a execução eficiente de uma aplicação específica, como nos ASICs, ASIPs (Application Specifc Instruction-set Processor)[32], e DSPs (Digital Signal Processors)[17].

A configurabilidade de arquiteturas computacionais consiste nos mecanismos e recursos usados para dar ao sistema uma organização estrutural adaptada à aplicação a ser executada. A configurabilidade em *hardware* é recente, apesar de ser conhecida desde a década de 70, com os PALs (Programmable Array Logic)[19], PLAs (Programmable Logic Array)[10] e PMLs (Programmable Multilevel Logic Arrays)[35], popularizou-se somente na década de 80 com o surgimento dos FPGAs (Field Programmable Gate Arrays)[35], que apesar de terem o termo “programáveis” no nome, possuem recursos genéricos que permitem aos seus usuários a criação de funções em hardware.

A configurabilidade do *hardware* confere aos sistemas computacionais um maior grau de flexibilidade que a programabilidade. No entanto, tal flexibilidade não é adquirida gratuitamente, a complexidade adicional para o desenvolvimento de aplicações e a sub-utilização dos recursos e estruturas implantadas em hardware são alguns dos problemas enfrentados.

A utilização dos FPGAs despertou a idéia de definir de forma rápida e eficiente novas funções em *hardware*, como forma de atender aos requisitos de aplicações específicas. Atualmente, denomina-se como reconfigurável a classe de sistemas com capacidade de reorganizar o *hardware* de acordo com a demanda da aplicação. Mais que a configurabilidade, as arquiteturas reconfiguráveis têm a capacidade de suportar múltiplas configurações.

2.2 FPGAs - Field-Programmable Gate Arrays

Os FPGAs foram originalmente desenvolvidos como um dispositivo híbrido entre os PALs (*Programmable Array Logic*) e os MPGAs (*Mask-Programmable Gate Arrays*) [24]. Como os PALs, os FPGAs são completamente programáveis eletricamente, isto significa que não é necessário recorrer à engenharia, amortizando, desta forma, os custos de projeto e permitindo uma adaptação quase que instantânea do sistema. Como os MPGAs, os FPGAs conseguem implementar funções lógicas muito complexas, em um chip com milhões de portas lógicas, que são produzidos atualmente.

Um FPGA é descrito como um dispositivo constituído por três estruturas básicas [1], como pode ser visualizado na figura 2.2[23]

- Um vetor de células de I/O programáveis
- Uma estrutura de interconexão programável
- Um vetor de células lógicas programáveis, CLBs (*Configurable Logic Block*)

Células de I/O programáveis

É através das células de I/O, ou blocos de entrada e saída, que é implementada a interface do sistema, elas são usadas para alimentar os pinos físicos do FPGA. Em alguns

[1] Ná prática, atualmente, os FPGAs são dispositivos muito mais complexos que os desenhos e estruturas mostradas neste texto, no entanto os dispositivos aqui demonstrados são suficientes para a descrição do modelo matemático.

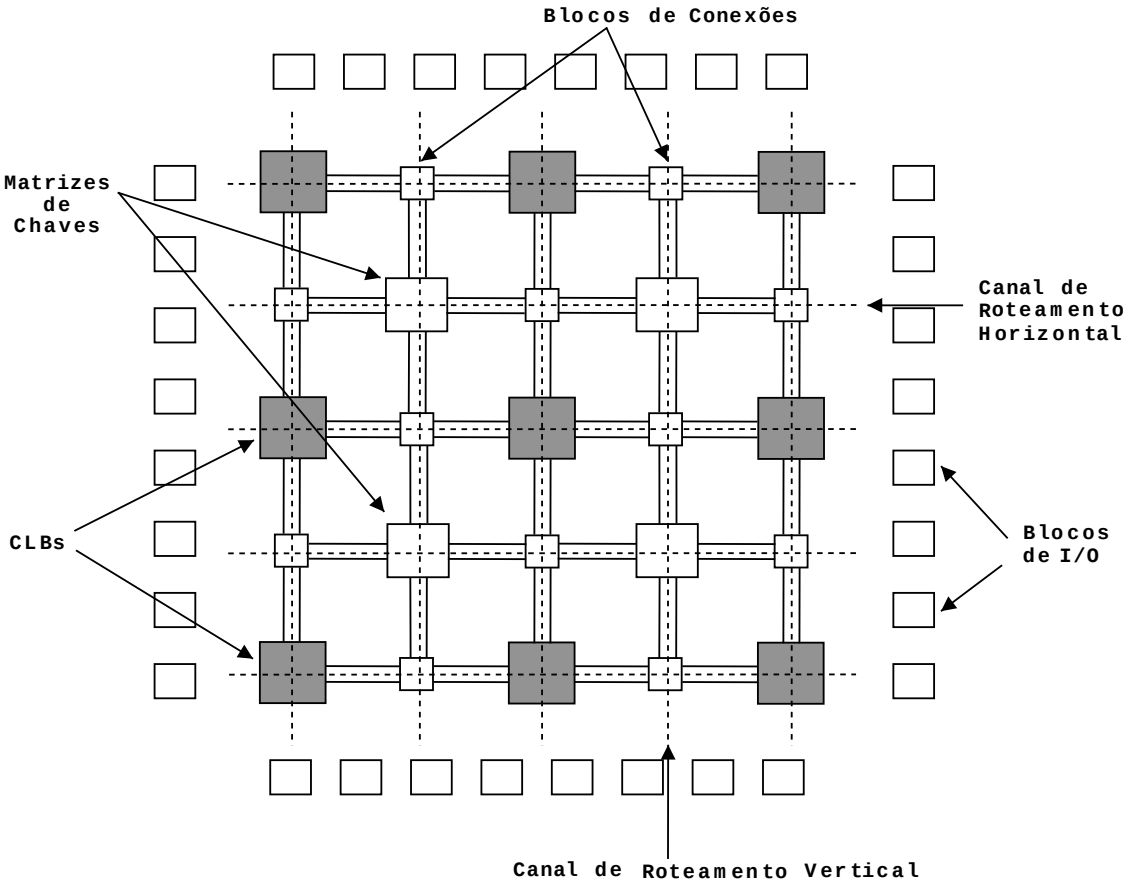

Figura 2.2: Estrutura interna de um FPGA

FPGAs, as células de I/O estão dispostas no final de cada linha e coluna da estrutura de interconexão. Cada célula de I/O pode conter um *buffer* bidirecional de I/O e um flip-flop [2], para ser utilizado como um registrador de entrada e saída, alimentando a entrada, saída, ou sinais bidirecionais.

Estrutura de interconexão programável

A conexão entre as CLB's e os pinos das células de I/O é feito através de linhas rápidas de interconexão, que consiste em uma série de canais horizontais e verticais de roteamento, permitindo o roteamento global de sinais no FPGA. Essa forma de roteamento necessita de *matrizes de chaves* [23], para rotear e conectar um número variável de caminhos, o que aumenta o retardo na transmissão entre as CLBs, reduzindo, então, o desempenho do FPGA. Para fazer a conexão dos canais entre as CLBs e a matriz de chaves, existem os blocos de conexões que permitem que os sinais mudem a direção da rota em qualquer ponto.

As características e complexidade da estrutura de interconexão programável variam de acordo com a tecnologia de programação utilizada. A estrutura de roteamento para

FPGAs é chamada de layout em estilo de ilhas [8], uma vez que os CLBs são arrodeados por canais de roteamento.

Vetor de CLBs

As CLBs são elementos que podem ser configurados para realizar determinadas funções lógicas. Cada CLB é projetada para que possa executar o maior número de funções. Existem diversas tecnologias, que podem ser escolhidas para o desenvolvimento de um bloco lógico, a escolha dessa estratégia influencia em quesitos como complexidade das funções implementáveis, área do bloco e custo. As estratégias para implementação das células lógicas serão abordadas na seção 2.2.1

2.2.1 Tecnologia de Implementação das CLBs

Existem três tecnologias de implementação de blocos lógicos mais comumente encontradas em FPGAs [35]: baseada em *Look-up-tables*, multiplexadores e vetores lógicos programáveis. Cada uma destas estruturas é apresentada nos itens a seguir.

- LUTs (*Look-up-tables*)

As LUTs, ou tabelas de conversão, são memórias pequenas, desenvolvidas para a computação de funções lógicas, que são geralmente utilizadas, como estruturas computacionais em uma FPGA. Uma LUT pode computar qualquer função com N entradas [8], onde N é o número de sinais de controle para o multiplexador da LUT, através da computação da tabela-verdade da função desejada, com 2^N combinações. A figura 2.3 [8] representa uma LUT de 3 entradas, dessa forma, se todos os bits de programação (P_N, com N variando de 1 a 8) estivessem setados com valor 0, exceto o correspondente às entradas $E1$,$E2$ e $E3$, quando estas possuem valor 1, a LUT se comportaria como uma AND de 3 entradas.

- Multiplexadores programáveis

A implementação de CLBs com multiplexadores programáveis se baseia no *teorema da expansão de Shannon* [34]. O teorema da expansão de Shannon estabelece que qualquer função booleana, pode ser expandida em relação as suas variáveis da seguinte forma:

Seja $F(V_1, V_2, ..., V_n)$ uma função booleana com n variáveis. A expansão de Shannon de F com relação à variável V_i pode ser escrita como sendo $F'(V_1, V_2, ..., V_n) = V_i \cdot F(V_i =$

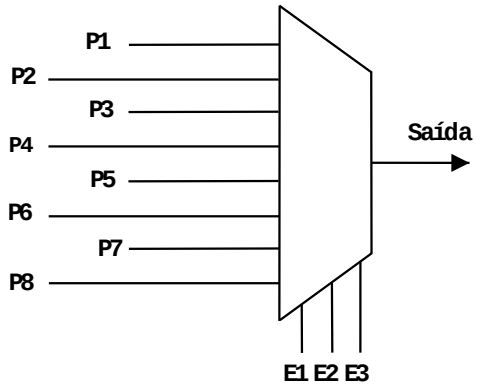

Figura 2.3: Uma LUT de 3 entradas

$1) + V_i' \cdot F(V_i = 0)$. Onde $F(V_i = 1)$ representa a função F com a variável V_i definida com o valor lógico 1. Assim, $F(V_i = 1) = F(V_1, \ldots, V_{i-1}, 1, V_{i+1}, \ldots, V_n)$.

Por exemplo, seja a função $F(A, B, C) = A' \cdot B + A \cdot B \cdot C' + A' \cdot B' \cdot C$, podemos expandir a F em relação a variável A:

$$F'(A, B, C) = A \cdot F(A = 1) + A' \cdot F(A = 0)$$
$$F'(A, B, C) = A \cdot (0 \cdot B + 1 \cdot B \cdot C' + 0 \cdot B' \cdot C') + A' \cdot (1 \cdot B + 0 \cdot B \cdot C' + 1 \cdot B' \cdot C)$$
$$F'(A, B, C) = A \cdot (B \cdot C') + A' \cdot (B + B' \cdot C')$$

Com a expansão de Shannon em relação à variável A, a função F decompôs-se em duas funções menores, $F(A = 1) = B \cdot C'$ e $F(A = 0) = B + B' \cdot C$. A função F descreve um multiplexador de duas entradas com o valor de saída definido em função do valor da variável A. A figura 2.4 mostra o multiplexador para a função F. Funções complexas podem ser implementadas em multiplexadores maiores ou por combinação de multiplexadores simples.

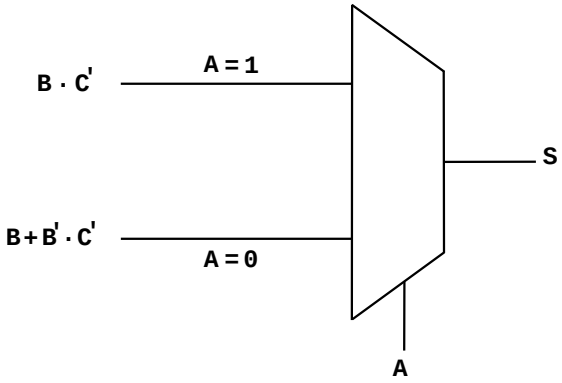

Figura 2.4: Multiplexador para a função F com a expansão da variável A

- PALs (*Programmable Array Logic*)

Um PAL é um dispositivo lógico que consiste em um *array* programável de portas AND conectado a um *array* fixo de portas OR, através do qual podem ser implementadas funções booleanas, representadas em FND[2] . A saída do conjunto de portas OR fica, comumente, armazenada em *flip-flops*. A figura 2.5 mostra uma matriz PAL.

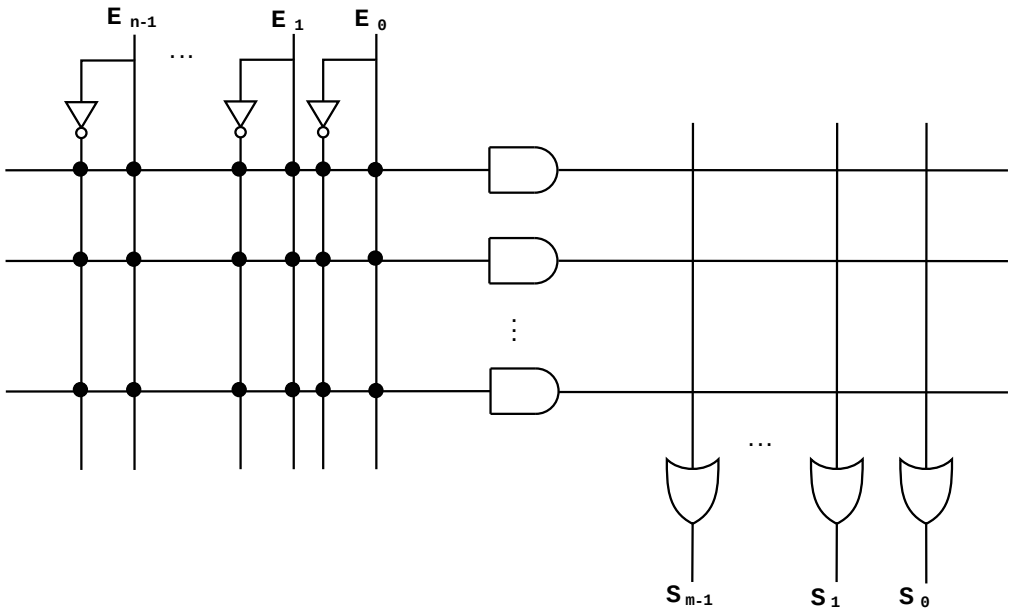

Figura 2.5: Estrutura geral de um PAL

Exemplos de configurações para um FPGA

As figuras 2.6 e 2.7 mostram duas possíveis configurações para o FPGA mostrado na figura 2.2. Em 2.6 está implementado um decodificador binário do tipo "2 *to* 4", que possui 2 entradas, cujas possíveis combinações ativam distintamente as 4 saídas. Reutilizando a mesma arquitetura, é possível implementar um meio-somador, como pode ser observado na figura 2.7.

Na figura 2.8, é mostrado um FPGA configurado com um somador completo, onde as CLBs funcionam como meio somadores. Observe que na figura 2.7 a função meio somador é configurada em um FPGA, enquanto que na figura 2.8 o meio somador é calculado em um CLB. Na figura 2.9, existem CLB's que implementam meios-somadores e outras que implementam somadores completos, para que o FPGA funcione como um somador de $8bits$. Essas diferenças estão implicitamente ligadas à complexidade e à granularidade das CLBs.

[2]Forma Normal Disjuntiva, onde as funções booleanas são somas de produtos.

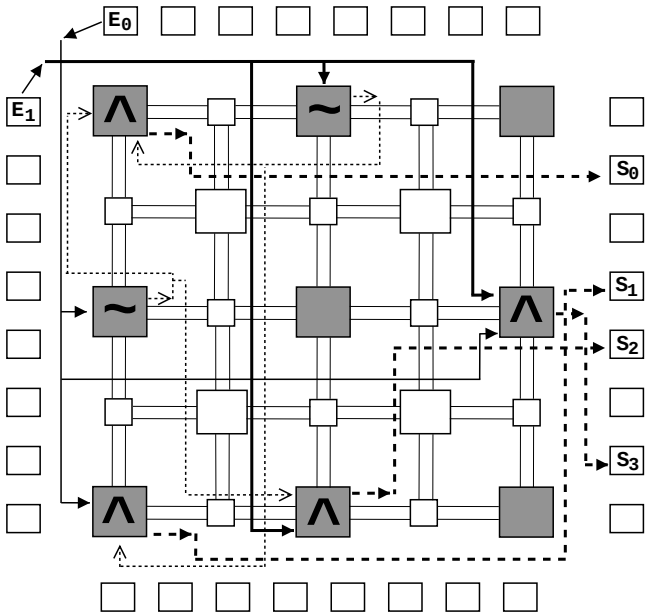

Figura 2.6: Um FPGA configurado como um decodificador "2 *to* 4".

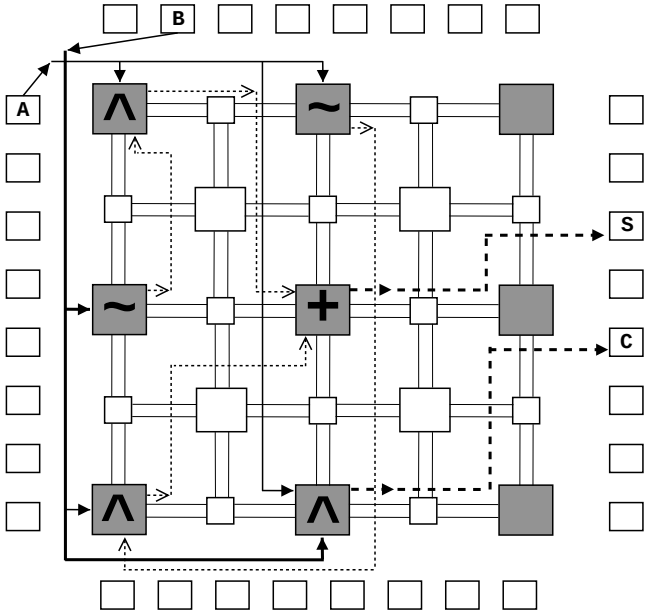

Figura 2.7: Um FPGA configurado como um meio somador.

2.2.2 Granularidade das CLBs

Os blocos lógicos podem ser diferenciados entre si por vários aspectos como objetivo da reconfigurabilidade [31], integração [31], modelo de execução [26], programabilidade [1] e granularidade [15], entre outros. O critério da granularidade, que leva em consideração o tamanho das unidades reconfiguráveis [13] ou o tamanho da palavra tratada pelas unidades [15], é o mais amplamente adotado. Quanto a granularidade, a arquitetura de um circuito reconfigurável pode ser classificada como:

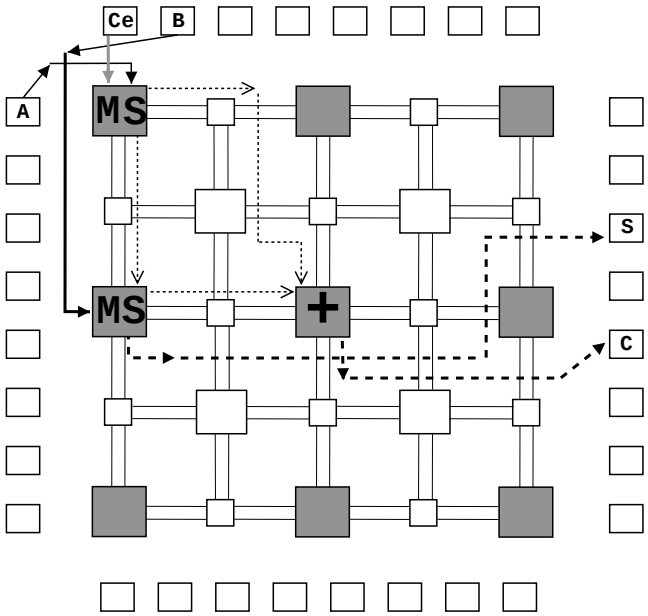

Figura 2.8: FPGA configurado como somador completo a partir de 2 meio-somadores.

- topologia baseada em granularidade fina, onde os blocos lógicos representam funções que trabalham com entrada de 1 bit apenas.
- topologia baseada em granularidade grossa, onde os blocos lógicos representam funções que trabalham com palavras de tamanhos maiores, como 8 bits, por exemplo.

As topologias baseadas em granularidade fina apresentam uma grande flexibilidade lógica, atribuída ao fato de trabalharem com portas lógicas simples, possibilitando a criação de estruturas de tamanho arbitrário. No entanto, é necessário um número grande de unidades lógicas de granularidade fina para implementar funções complexas, necessitando, portanto uma grande quantidade de blocos de conexão, gerando *overhead* no roteamento dos sinais. Exemplos de FPGAs que utilizam esta topologia são as famílias AT6000 da Atmel [9] e XC6200 da Xilinx [20].

Em uma topologia baseada em granularidade grossa, um único bloco lógico pode implementar muitas funções lógicas, seja uma porta lógica ou funções lógicas complexas. No entanto, blocos lógicos compostos por um grande número de portas lógicas não servem para implementar estruturas regulares, como memória e *datapaths*[3]. Como exemplos deste tipo de topologia podemos citar as máquinas RAW, *Reconfigurable Architecture Workstation* [38] e arquitetura Garp [16].

A granularidade é um critério importante, pois determina, de um certo modo, a eficiência do projeto da arquitetura, que pode ser medida em termos de área ocupada pelas

[3]Caminho de dados, utilizado aqui para referenciar operações de transferência de dados, como mover o conteúdo de um registrador para outro.

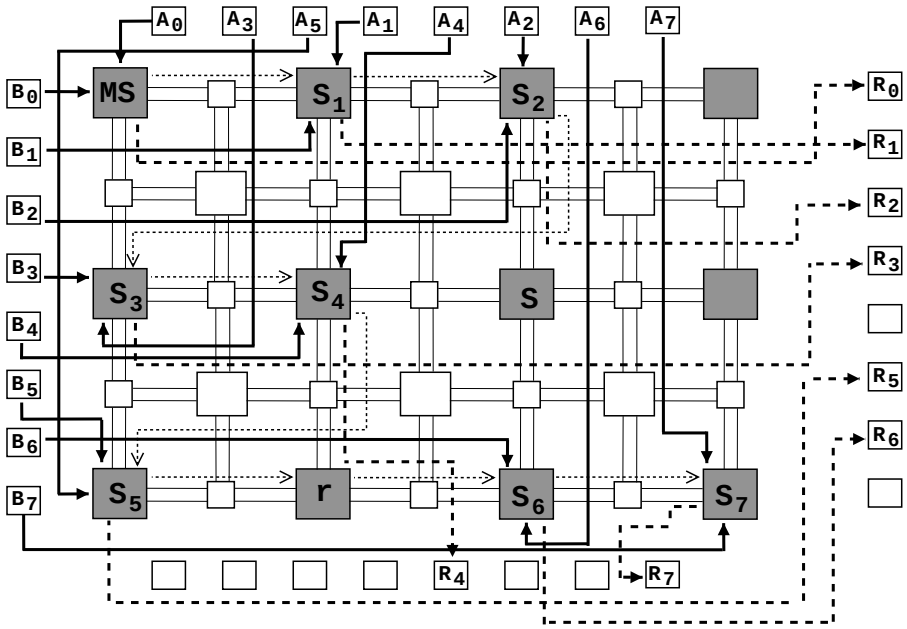

Figura 2.9: Somador completo de 8 bits.

estruturas de interconexão [15] ou em termos de complexidade das operações implementadas por uma unidade reconfigurável [13], [31].

2.3 Software

O *hardware* reconfigurável tem mostrado um desempenho significativo que beneficia a execução de programas, no entanto, esse argumento pode ser desconsiderado pelos programadores de aplicação se não for possível incorporar facilmente o uso do *hardware* reconfigurável em seus sistemas. Isto requer um ambiente de projeto de *software* que auxilie na criação das configurações para um *hardware* reconfigurável. Este *software* pode prover assistência para a criação manual do circuito, ou até mesmo servir de ambiente para o desenvolvimento automático de projeto de circuitos. A descrição manual de circuitos é um método eficiente para a criação dos projetos de circuito de alta qualidade, entretanto, requer o conhecimento profundo do sistema, bem como uma quantidade significativa de tempo de projeto. Um sistema automático de desenvolvimento fornece uma maneira rápida e fácil de programar sistemas reconfiguráveis, e conseqüentemente, faz o uso do *hardware* reconfigurável mais acessível aos programadores da aplicação.

Para ambos, criação manual e automática do circuito, o processo do projeto deve prosseguir com um número de fases distintas. Segundo Compton e Hauck [8], o projeto de circuitos deve ser dividido em cinco fases:

- Especificação do circuito,

- Descrição em nível de porta,
- Mapeamento entre tecnologias,
- Distribuição dos blocos,
- Roteamento.

A fase de *especificação do circuito* é o processo de descrever as funções que serão habilitadas no hardware reconfigurável. Isto pode ser feito de forma tão simples quanto escrever um programa em C que representa a funcionalidade do algoritmo a ser executado no *hardware*. Por outro lado, isto pode também ser tão complexo quanto especificar as entradas, as saídas, e a operação de cada bloco básico no sistema reconfigurável. Entre estes dois métodos está a especificação do circuito usando componentes genéricos complexos, tais como os adicionadores e os multiplicadores, que serão mapeados ao *hardware* real em outra fase no processo do projeto.

Quando os sistemas incluem um *hardware* reconfigurável e um microprocessador tradicional, os programas devem, ser divididos em "seções a serem executadas no *hardware* reconfigurável" e "seções a serem executadas em *software* no microprocessador". Seqüências de controle complexas, como os laços variáveis, são executadas mais eficientemente em *software*, enquanto as operações de *datapath* fixo podem ser mais eficientemente executadas em *hardware*. Isto pode ser realizado manualmente, delimitando seções do programa dentro do código fonte a ser executado em *hardware*, ou alternativamente, uma ferramenta automática de compilação pode determinar essas áreas do código.

Para descrições em uma linguagem de alto nível, como C/C++ ou Java, ou uma que utiliza blocos de construção complexos, o código destinado ao hardware reconfigurável deve ser compilado para uma rede de componentes em nível de portas lógicas. Para as execuções em linguagem de alto nível, isto envolve geração de componentes computacionais que executem as operações aritmética e lógica dentro do programa, e a criação de estruturas que assegurem o controle do programa, tal como iterações em laços e operações condicionais. Dada uma descrição estrutural, gerada a partir de uma linguagem de alto nível ou especificada pelo usuário, cada estrutura complexa é substituída com uma rede das portas básicas que implementa a função executada pela estrutura. Essa tarefa é realizada durante a fase de *descrição em nível de porta*.

Uma vez que a descrição detalhada do circuito em nível de porta foi criada, estas estruturas deve ser traduzida para os elementos reais de lógica do *hardware* reconfigurável. Este estágio é conhecido como *tecnologia de mapeamento*, e é dependente da arquitetura escolhida para o projeto.

Depois que o circuito foi mapeado, os blocos resultantes devem ser colocados no *hardware* reconfigurável. É atribuído a cada bloco uma posição específica dentro do *hardware*, se possível perto dos outros blocos lógico com os quais se comunica, esta fase é chamada de *Distribuição dos blocos*.

Finalmente, os diferentes componentes reconfiguráveis que compreendem o circuito da aplicação são conectados durante o estágio de *roteamento*. Isto pode tornar-se difícil, se na fase de distribuição dos blocos componentes conectados sejam colocados distantes entre si, pois os sinais transmitidos por distâncias maiores utilizam mais recursos de roteamento do que os que viajam por caminhos mais curtos. Uma boa distribuição é essencial ao processo de roteamento, uma vez que os recursos disponíveis para roteamento são limitados. No projeto geral da hardware, o objetivo é minimizar o número de trilhas de roteamento usadas em uma linha entre as filas de blocos lógicos programáveis.

2.4 Testes em Arquiteturas Reconfiguráveis

Devido ao aumento da complexidade de desenvolvimento de arquiteturas reconfiguráveis, a presença dos testes já na fase inicial da concepção tornou-se essencial. Os testes são necessários para detectar problemas de funcionamento que podem comprometer a confiabilidade de um sistema.

O principal requisito para classificar um sistema como confiável é determinar se ele está livre de erros. A confiabilidade é imprescindível na maior partes das áreas da computação, por exemplo na computação médica, uma falha no sistema pode levar o médico a um diagnóstico errado.

Motivado pela complexidade do processo de teste em sistemas, pode-se optar por um modelo de concepção que objetive facilitar a etapa de teste após a fabricação, abordagem chamada de *projeto visando a testabilidade* [33].

Outra abordagem para o acréscimo de confiabilidade é através de técnicas de *tolerância a falhas* [3], que visa permitir que o sistema continue funcionando na presença de falhas, utilizando-se da redundância em hardware ou software.

2.4.1 Métricas de Teste

Um projeto que visa a testabilidade, geralmente, consegue um aumento qualitativo nas métricas de teste: observabilidade e controlabilidade [33]. A observabilidade é representada pelo esforço necessário para observar um valor lógico de um ponto interno através

das saídas do circuito. A controlabilidade é o esforço necessário para atribuir um determinado valor lógico a um ponto interno. Deve ser entendido por ponto interno, os pontos do sistema que não são entradas, nem saídas. O aumento no número de pontos internos de um circuito afeta negativamente a observabilidade e a controlabilidade do sistema.

Na figura 2.10 é mostrado um circuito digital que implementa a função booleana $A + (B \cdot C)$. Foram assinalados com círculos preenchidos alguns pontos do sistema. Inicialmente os pontos $1, 2$ e 3 são controláveis - pois são entradas do sistema, onde se tem o controle de atribuir um determinado valor - e o ponto 5 é observável - pois é saída do sistema e portanto é possível observar o seu valor sem nenhum esforço adicional. Já a saída da porta *and*, no ponto 4, não é diretamente observável, bem como a entrada da porta *or*, que também está assinalada pelo ponto 4, não é diretamente controlável, no entanto, com a atribuição de determinados valores às entradas, pode-se conseguir observabilidade e controlabilidade no ponto 4.

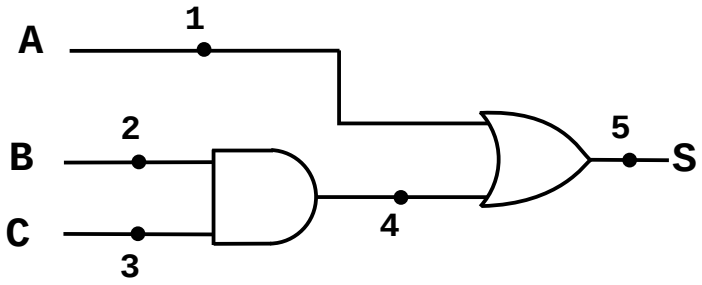

Figura 2.10: Circuito com identificação dos pontos observáveis e controláveis.

Na tabela 2.1, são mostradas combinações de valores para obter-se observabilidade e controlabilidade no ponto 4 representado na figura 2.10. Uma combinação de valores como as mostradas na tabela 2.1, são denominadas de vetores de testes [12]. No entanto, para elaborar vetores de testes, é importante especificar que tipo de falha objetiva-se detectar, por exemplo, ainda utilizando o circuito da figura 2.10, deseja-se verificar se existe uma colagem[4][18] em 0 no ponto 4 do circuito. Nesse caso, pode-se utilizar o vetor $0, 1, 1$ para "forçar"uma saída 1 nos ponto 4 e 5. No entanto, mesmo que ocorra o previsto, não há como saber, com esse teste, se existe uma colagem em 1 no ponto 5 do circuito, demonstrando que, muitas vezes, um único vetor de teste não é suficiente para atestar ausência de falhas.

[4]Uma colagem, ou *stuck-at* é um defeito físico, que modifi ca o comportamento de um bloco lógico, de forma que o valor computado fi xa-se em um dos valores lógicos, independente da variação das entradas do bloco

	A	B	C
$control = 1$	$0/1$	1	1
$control = 0$	$0/1$	1	0
$control = 0$	$0/1$	0	1
$control = 0$	$0/1$	0	0
$observabilidade$	0	$0/1$	$0/1$

Tabela 2.1: Valores para obter controlabilidade e observabilidade no ponto 4 mostrado na figura 2.10

Capítulo 3

Espaços de Chu

Espaços de Chu [14] [27] foram definidos inicialmente, como resultado das construções de Chu[1] na categoria dos conjuntos.

As construções de Chu são uma tentativa de formalizar uma noção apropriada de reticulado distributivo parcial [14] como uma estrutura de eventos para modelar computação concorrente. O modelo desenvolvido por Chu caracteriza-se pela sua simplicidade em virtude da sua representação matricial, que facilita a descrição de um sistema concorrente como um conjunto de estados.

Definição 3.1 *Um* **Espaço de Chu** *em um conjunto K é uma tripla $\mathcal{A} = (X, A, R)$, onde X e A são conjuntos e $R : X \times A \longrightarrow K$ é uma função. Os elementos de X e A são vistos, geralmente, com estados e eventos, respectivamente. Um Espaço de Chu será finito se X e A são conjuntos finitos [6].*

Exemplo 3.1 *O Espaço de Chu $\mathcal{A}$, é denotado pela tripla (X, A, R), onde:*
$X = \{x, y, z\}$,
$A = \{a, b, c, d\}$ *e*
$R = \{ ((x,a),0), ((x,b),1), ((x,c),0), ((x,d),1),$
$((y,a),1), ((y,b),0), ((y,c),0), ((y,d),0),$
$((z,a),0), ((z,b),1), ((z,c),1), ((z,d),0)\}$

A partir da definição de um Espaço de Chu $\mathcal{A} = (X, A, R)$, pode-se obter ainda duas funções μ e η. Seja uma função $\mu : X \longrightarrow \mathbb{P}(A)$, onde $\mu(x) = \{a \in A \| R(x, a) = 1\}$. Seja função $\eta : A \longrightarrow \mathbb{P}(X)$, onde $\eta(a) = \{x \in X \| R(x, a) = 1\}$.

[1] O nome espaços de Chu foi sugerido a Vaughan Pratt por Michael Barr, por terem sido construídos por este e Po-Hsiang Chu

Quando aplicadas ao Espaço de Chu representado no exemplo 3.1, as funções μ e η produzem os seguintes resultados:

$\mu(x) = \{b, d\}$

$\mu(y) = \{a\}$

$\mu(z) = \{b, c\}$

$\eta(a) = \{y\}$

$\eta(b) = \{x, z\}$

$\eta(c) = \{z\}$

$\eta(d) = \{x\}$

Tratando-se da relação de estados e eventos, as funções $\mu(x)$ e $\eta(a)$ representam, respectivamente, que eventos ocorrem em um estado x e em quais estados ocorre o evento a.

O Espaço de Chu $\mathcal{A}$ é denominado de *extensional* [28] quando a função μ é injetora, ou seja, estados distintos associam-se sempre a subconjuntos distintos de eventos, o que na representação matricial, vista na seção 3.1.1, significa dizer que não há linhas iguais. Quando η é uma função injetora, o Espaço de Chu $\mathcal{A}$ é chamado de *separável* [28], eventos distintos associam-se a subconjuntos distintos de estados, e em sua representação matricial não encontra-se colunas iguais. Um Espaço de Chu que é *extensional* e *separável*, é dito como *biextensional*[28].

3.1 Representações de Espaços de Chu

Quando $K = 2$ é possível representar espaços de Chu sobre vários aspectos[14], como Representação Pictorial, Pdlats, representação matricial, lógica e circuitos combinacionais, estes três últimos são de maior importância para este trabalho, portanto serão abordados nas seções seguintes.

3.1.1 Matricial

Para uma representação trivial de um Espaço de Chu é utilizada uma matriz K-valorada de dimensão $|X| \times |A|$, onde as linhas e colunas são indexadas pelos elementos de X e

A, respectivamente, e para cada $x \in X$ e $a \in A$ o valor da posição (x, a) na matriz é dado por $R(x, a)$. A matriz

$\mathcal{A}$	a	b	c	d
x	0	2	0	1
y	1	2	0	0
z	2	1	1	2

é a representação do Espaço de Chu $\mathcal{A}$ em $K = \{0, 1, 2\}$, onde $X = \{x, y, z\}$, $A = \{a, b, c, d\}$, $R(x, a) = R(x, c) = R(y, c) = R(y, d) = 0$, $R(x, d) = R(y, a) = R(z, b) = R(z, c) = 1$ e $R(x, b) = R(y, b) = R(z, a) = R(z, d) = 2$.

3.1.2 Fórmulas Proposicionais

Um Espaço de Chu finito[2], onde $K = \{0, 1\}$, pode ser representado através de fórmulas proposicionais na Forma Normal Disjuntiva Completa, FNDC[3]. Seja $\mathcal{A} = (X, A, R)$ um Espaço de Chu e ϱ uma função que atribui um símbolo proposicional a cada evento $a \in A$, tem-se que para cada estado $x \in X$ existe uma fórmula $\phi(x) = \bigwedge_{a \in A} \imath_a$, onde $\imath_a = \varrho(a)$, se $R(x, a) = 1$ e $\imath_a = \neg\varrho(a)$, se $R(x, a) = 0$. A fórmula proposicional

$$\varphi(\mathcal{A}) = \bigvee_{x \in X} \phi(x)$$

representa $\mathcal{A}$.

$\mathcal{A}$	a	b	c	d
x	0	1	0	1
y	1	0	0	0
z	0	1	1	0

Figura 3.1: Representação matricial de um Espaço de Chu com $K = \{0, 1\}$

A matriz na figura 3.1, representa o Espaço de Chu $\mathcal{A}$, onde $\phi(x) = \neg\varrho(a) \wedge \varrho(b) \wedge \neg\varrho(c) \wedge \varrho(d)$, $\phi(y) = \varrho(a) \wedge \neg\varrho(b) \wedge \neg\varrho(c) \wedge \neg\varrho(d)$ e $\phi(z) = \neg\varrho(a) \wedge \varrho(b) \wedge \varrho(c) \wedge \neg\varrho(d)$. Para simplificar a notação, serão utilizados a, $\bar{a}$ e ab ao invés de $\varrho(a)$, $\neg\varrho(a)$ e $\varrho(a) \wedge \varrho(b)$, respectivamente. Dessa forma, a equação

$$\varphi(\mathcal{A}) = \bar{a}b\bar{c}d \vee a\bar{b}\bar{c}\bar{d} \vee \bar{a}bc\bar{d}$$

[2]A partir deste ponto, os Espaços de Chu finitos, isto é, onde os conjuntos de estados e eventos são finitos, para $K = \{0, 1\}$ serão chamados simplesmente de Espaços de Chu.

[3]Um fórmula na Forma Normal Disjuntiva chama-se Completa quando cada cláusula depende de todas as variáveis proposicionais do domínio da função booleana.

é a fórmula proposicional na FNDC, associada ao Espaço de Chu $\mathcal{A}$.

Cada fórmula proposicional α e portanto um Espaço de Chu, descreve uma função booleana de $\mathbb{B}^n \longrightarrow \mathbb{B}$, onde n é a quantidade de diferentes símbolos proposicionais em α e $\mathbb{B}$ é o conjunto $\{0, 1\}$. Pode-se associar cada Espaço de Chu $\mathcal{A}$ a uma função booleana $\chi_{\mathcal{A}} : \mathbb{B}^n \longrightarrow \mathbb{B}$ com $n = |A|$. Seja uma função bijetora $\tau_{\mathcal{A}} : A \longrightarrow \{1, \ldots, n\}$, que associa um índice a cada elemento de A, a função $\chi_{\mathcal{A}}$ é formalmente definida por:

$$\chi_{\mathcal{A}}(\vec{b}) = \nu_{\rho}(\varphi(\mathcal{A})),$$

onde ν_{ρ} é a função que avalia uma fórmula proposicional para a associação $\rho : \varrho(A) \longrightarrow \mathbb{B}$, definida por $\rho(\varrho(a)) = \pi_{\tau(a)}(\vec{b})$ com $\pi_i(b_1, \ldots, b_n) = b_i$.

3.1.3 Binary Decision Diagram - BDD

Um diagrama de decisão binária, BDD, é uma forma de representar e manipular expressões lógicas booleanas. Os BDDs são estruturas de dados acíclicas, direcionados, com raiz, onde existem dois tipos de nós: os terminais e os não terminais. Os nós terminais são sempre dois e representam o resultado da função lógica representada, podendo assumir valor verdadeiro (1) ou falso (0). Os nós não terminais estão associados a uma variável de entrada e dele partem sempre duas arestas, uma para cada possível valor lógico binário. O resultado da expressão é encontrado percorrendo os nós do grafo, até um nó terminal, que é o valor da função. A vantagem na utilização de BDDs, na representação de expressões booleanas, é que geralmente os BDDs ocupam menos espaço que representações tradicionais, como tabelas verdades, árvores de decisão ou mapas de Karnaugh. Na figura 3.2 está a representação de uma função booleana utilizando BDD.

Apesar dos BDDs ocuparem menos espaço para representar funções booleanas do que outros métodos, como árvores de decisão e tabelas verdade, a ordem em que as variáveis estão dispostas pode alterar o tamanho do BDD [7]. Na figura 3.3 mostra um outro BDD para a mesma função do BDD da figura 3.2, no entanto, há diferença no tamanho dos BDDs, na figura 3.2, quando a ordem das variáveis é $a < b < c < d$ o BDD resultante tem 4 nós, enquanto na figura 3.3, cuja a ordem das variáveis é $a < c < b < d$ o número de nós sobe para 6.

3.1.4 Circuitos Combinacionais

A possibilidade de representar Espaços de Chu através de fórmulas lógicas proposicionais induz a idéia de utilizar circuitos combinacionais para representar Espaços de Chu, uma

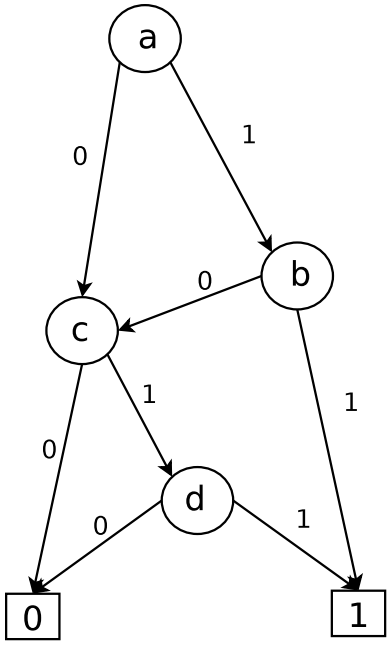

Figura 3.2: Representação da função $(a \wedge b) \vee (c \wedge d)$ através de BDD

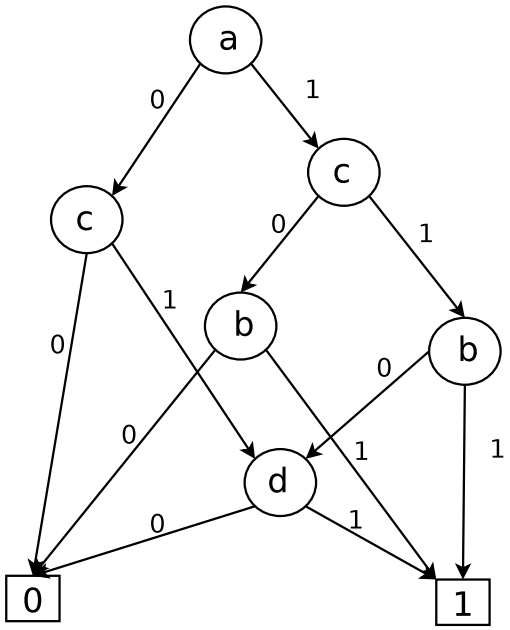

Figura 3.3: Representação da função $(a \wedge b) \vee (c \wedge d)$ através de BDD com 6 nós

vez que estes circuitos computam funções lógicas booleanas. Em um circuito combinacional que representa o espaço de Chu $\mathcal{A}$, descrito pela a tripla (X, A, R), as entradas são os elementos de A, e a saída é o cálculo da expressão booleana que representa o espaço de Chu $\mathcal{A}$. A figura 3.4 ilustra a representação, por meio de circuitos combinacionais, do Espaço de Chu representado na figura 3.1.

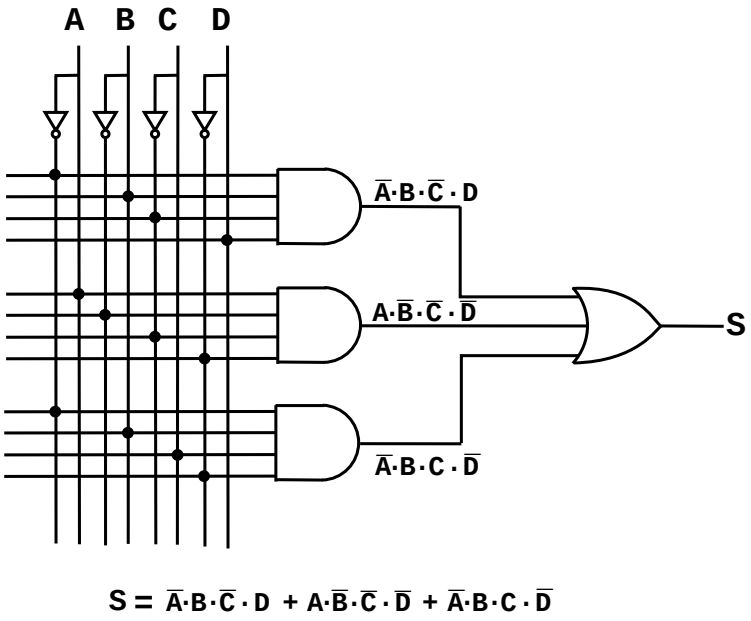

Figura 3.4: Representação de espaços de Chu através de circuitos combinacionais

3.2 Álgebra para Espaços de Chu

Em [14] e em [27] é dado uma interpretação em álgebra de processos aos conectivos da lógica linear, mostrando que os Espaços de Chu também pode ser um modelo para a Lógica Linear [30].

Desde a sua concepção, a lógica linear tem sido interpretada sob vários pontos de vista, e em várias áreas da Ciência da Computação, principalmente aquelas que lidam com processamento paralelo e concorrência entre processos. Os operadores da lógica linear podem ser agrupados em três categorias [27]: os fragmentos multiplicativo, aditivo e exponencial, além da negação linear em conjunto com a implicação linear.

As operações do fragmento multiplicativo, como *produto tensor* $(\otimes)$ e *par* $(\wp)$ são derivadas da operação Produto Cartesiano. As operações *plus* $(\oplus)$ e *with* $(\&)$, do fragmento aditivo são derivadas de operações sobre conjuntos. No fragmento exponencial, as operações *É claro* $(!)$ e *por que não* $(?)$ se baseiam nas operações com conjuntos das partes. No *site Chu Spaces Live* [29] encontra-se uma calculadora para essas operações.

Uma das operações definidas é *with*, que é conhecido como produto direto ou categórico. O resultado da operação *with* é a concatenação de todos os eventos de um par de Espaço de Chu. Sejam $\mathcal{A} = (X_1, A_1, R_1)$ e $\mathcal{B} = (X_2, A_2, R_2)$ Espaços de Chu, $\mathcal{A}\&\mathcal{B} = (X, A, R)$, onde $X = X_1 \uplus X_2$[4] e $A = A_1 \times A_2$, logo $|A| = |A_1| \times |A_2|$ e

[4] $\uplus$ é a operação união disjunta, onde $A \uplus B = (A \times \{0\}) \cup (B \times \{1\})$

$$R(x,e) = \begin{cases} R_1(x_1,e_1) & \text{se } x_1 \in X_1,\ e_1 \in A_1 \text{ e } \exists e_2 \in A_2 \| e = (e_1,e_2) \text{ e } x = (x_1,0) \\ R_2(x_2,e_2) & \text{se } x_2 \in X_2,\ e_2 \in A_2 \text{ e } \exists e_1 \in A_1 \| e = (e_1,e_2) \text{ e } x = (x_1,1) \end{cases}$$

A figura 3.5 mostra a produto de dois Espaços de Chu representados pelas matrizes $\mathcal{A}$ e $\mathcal{B}$.

$\mathcal{A}$	a_1	a_2
x_1	0	1
x_2	1	0

&

$\mathcal{B}$	a'_1	a'_2
x'_1	1	1
x'_2	0	1

=

$\mathcal{A}\&\mathcal{B}$	(a_1,a'_1)	(a_1,a'_2)	(a_2,a'_1)	(a_2,a'_2)
$(x_1,0)$	0	0	1	1
$(x_2,0)$	1	1	0	0
$(x'_1,1)$	1	1	1	1
$(x'_2,1)$	0	1	0	1

Figura 3.5: Representação matricial da operação *with* das matrizes A e B, que representam Espaços de Chu

Os Espaços de Chu são comumente utilizado para modelar sistemas onde há concorrência e paralelismo, por isso operações da lógica linear parecem ser apropriadas para transformar Espaços de Chu. No entanto, neste trabalho, os Espaços de Chu são utilizados para representar circuitos lógicos, e operações baseadas na lógica booleana clássica são mais pertinentes. Dessa forma, são definidas aqui, as operações de *Produto* $(\times)$, *Inclusão de um evento* e *Composição* $(\gtrdot)$, definida na seção 4.3.

O operador *produto* concatena todos os estados de um par de Espaço de Chu. Sejam $\mathcal{A} = (X_1, A_1, R_1)$ e $\mathcal{B} = (X_2, A_2, R_2)$ Espaços de Chu, $\mathcal{A} \times \mathcal{B} = (X, A, R)$, onde $X = X_1 \times X_2$, logo $|X| = |X_1| \times |X_2|$, $A = A_1 \uplus A_2$ e

$$R(x,e) = \begin{cases} R_1(x_1,e_1) & \text{se } e_1 \in A_1,\ x_1 \in X_1 \text{ e } \exists x_2 \in X_2 \| x = (x_1,x_2) \text{ e } e = (e_1,0) \\ R_2(x_2,e_2) & \text{se } e_2 \in A_2,\ x_2 \in X_2 \text{ e } \exists x_1 \in X_1 \| x = (x_1,x_2) \text{ e } e = (e_1,1) \end{cases} \tag{3.1}$$

A figura 3.6 mostra a operação *produto* de dois Espaços de Chu representados pelas matrizes $\mathcal{A}$ e $\mathcal{B}$.

Utilizando fórmulas proposicionais, tem-se que $\varphi(\mathcal{A}) = \bar{a_1}a_2 + a_1\bar{a_2}$ e $\varphi(\mathcal{B}) = a'_1a'_2 + \bar{a'_1}a'_2$, dessa forma $\varphi(\mathcal{A} \times \mathcal{B}) = \varphi(\mathcal{A}) \times \varphi(\mathcal{B})$:

$$(\bar{a_1}a_2 + a_1\bar{a_2}) \times (a'_1a'_2 + \bar{a'_1}a'_2) = \bar{a_1}a_2a'_1a'_2 + \bar{a_1}a_2\bar{a'_1}a'_2 + a_1\bar{a_2}a'_1a'_2 + a_1\bar{a_2}\bar{a'_1}a'_2$$

$\mathcal{A}$	a_1	a_2
x_1	0	1
x_2	1	0

$\times$

$\mathcal{B}$	a'_1	a'_2
x'_1	1	1
x'_2	0	1

$=$

$\mathcal{A} \times \mathcal{B}$	$(a_1, 0)$	$(a_2, 0)$	$(a'_1, 1)$	$(a'_2, 1)$
(x_1, x'_1)	0	1	1	1
(x_1, x'_2)	0	1	0	1
(x_2, x'_1)	1	0	1	1
(x_2, x'_2)	1	0	0	1

Figura 3.6: Representação matricial da operação *produto* das matrizes A e B, que representam Espaços de Chu

Sempre é possível fazer a *Adição de um evento em um Espaço de Chu* $\mathcal{A}$, sem modificar o comportamento de $\chi_{\mathcal{A}}$. Dado um Espaço de Chu $\mathcal{A}$ e um evento a, onde $a \notin A$, define-se o Espaço de Chu $\mathcal{A}_{+a}$ como $(A \cup \{a\}, X', R')$.

- $X' = X \uplus X$
- Para cada $x \in X$ e $b \in A$, $R'((x, 0), b) = R'((x, 1), b) = R(x, b)$, e
- Para cada $x \in X$, $R((x, 0), a) = 0$ e $R((x, 1), a) = 1$.

Capítulo 4

Modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu

Para permitir a modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu, são necessários alguns conceitos adicionais, como Multi-Espaço de Chu e Chu nets que estão definidos neste capítulo, baseado no artigo "Chu Spaces as a Formal Model for Reconfigurable Architectures" [6].

4.1 Multi-Espaço de Chu

Definição 4.1 *Seja $\mathbb{M}$ um conjunto finito e não vazio de Espaços de Chu e $\gamma : \{1, \ldots, |\mathbb{M}|\} \longrightarrow \mathbb{M}$ uma bijeção. $\mathcal{M} = (\mathbb{M}, \gamma)$ é chamado de multi-Espaço de Chu, se para cada $\mathcal{A}, \mathcal{A}' \in \mathbb{M}$, $A = A'$. $|\mathbb{M}|$ é a dimensão do multi-Espaço de Chu [6].*

Desde que o conjunto de eventos no multi-Espaço de Chu seja o mesmo, é possível denotá-lo de outra forma: Uma notação alternativa para um multi-Espaço de Chu $(\mathbb{M}, \gamma)$ de dimensão n é $(\vec{X}, A, \vec{R})_n$, onde $\vec{X}$ e $\vec{R}$ são vetores n-dimensionais, tal que

$$\vec{X}(i) = s(\gamma(i)) \text{ e } \vec{R}(i) = r(\gamma(i)),$$

com $s(\mathcal{A})$ e $r(\mathcal{A})$ são o conjunto de estados e a função do Espaço Chu $\mathcal{A}$, respectivamente, e $\vec{B}(i)$ é o i-ésimo componente do vetor. Então, $\gamma(i) = (\vec{X}(i), A, \vec{R}(i))$.

Um Espaço de Chu pode ser visto como um multi-Espaço de Chu de uma dimensão, dessa forma a definição de multi-Espaço de Chu generaliza a noção de Espaço de Chu. Analogamente, a função booleana que específica um Espaço de Chu, um multi-Espaço de

Chu $\mathcal{M}$ de dimensão n é representado pela função booleana $\chi_{\mathcal{M}} : \mathbb{B}^{|A|} \longrightarrow \mathbb{B}^n$, definida por

$$\chi_{\mathcal{M}}(\vec{a}) = (\chi_{\gamma(1)}(\vec{a}), \ldots, \chi_{\gamma(n)}(\vec{a}))$$

Por outro lado, cada fórmula proposicional α pode representar um Espaço de Chu, para isto, é necessário apenas que α esteja na FNDC. A partir deste ponto é fácil perceber que ao aplicar φ^{-1} será obtido o Espaço de Chu representado pela $FNDC(\alpha)$.

4.2 Chu nets

Definição 4.2 *Seja $\boldsymbol{C}$ um conjunto finito de Espaços de Chu, $\boldsymbol{A} = \bigcup_{\mathcal{A} \in \boldsymbol{C}} A$, $F : \boldsymbol{C} \longrightarrow \boldsymbol{A}$ seja uma função injetora parcial, m e n inteiros positivos, e $\alpha : \{1, \ldots, m\} \longrightarrow \boldsymbol{A}$ e $\beta : \{1, \ldots, n\} \longrightarrow \boldsymbol{C}$ funções injetoras. $(\boldsymbol{C}, F, \alpha, \beta)$ é uma* **Chu net** *de m entradas e n saídas se as seguintes condições forem satisfeitas [6]:*

1. $imag(\alpha) \cap imag(F) = \emptyset$
2. $imag(\alpha) \cup imag(F) = \boldsymbol{A}$
3. $imag(\beta) \cup dom(F) = \boldsymbol{C}$

Nenhuma restrição é imposta aos estados nem às relações dos Espaços de Chu na Chu net, isso permite que o comportamento interno de um Espaço de Chu seja independente dos outros que compõem a Chu net.

Definição 4.3 *Uma* **Chu net reconfigurável** *é uma tripla $(r, \vec{p}, k)$, onde r e k são números inteiros positivos e $\vec{p}$ é um vetor de números inteiros positivos de dimensão r.*

Uma Chu net reconfigurável modela uma arquitetura reconfigurável sem configuração específica. Neste sentido, r, k e $\vec{p}$ indicam, respectivamente, o número de células lógicas programáveis, a quantidade de células de I/O e a quantidade de portas disponíveis nas CLB's [1], na arquitetura reconfigurável. A arquitetura reconfigurável da figura 2.2 é modelada pela Chu net reconfigurável $\mathfrak{R} = (9, (6, 6, 6, 6, 6, 6, 6, 6, 6), 32)$, assumindo que cada CLB dispõe de 6 portas para a reconfiguração.

[1] O valor do p_i, o i-ésimo componente de $\vec{p}$, indica o máximo de portas que podem ser utilizadas na configuração da i-ésima célula programável

Definição 4.4 *Uma Chu net* $(\boldsymbol{C}, F, \alpha, \beta)$ *com* m *entradas e* n *saídas é uma configuração para uma Chu net* $(r, \vec{p}, k)$*, se* $|\boldsymbol{C}| \leqslant r$*,* $m+n \leqslant k$ *e há uma enumeração de* $\boldsymbol{C}$*, de forma que para cada* $i = 1, \ldots, r, |A_i| + 1 \leqslant p_i$*.*

Assim, duas configurações de uma Chu net reconfigurável podem ser diferentes. Uma possível configuração de $\mathfrak{R}$ que pode ser visualizada na figura 4.1, é uma Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, onde $\boldsymbol{C} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$, com

$\mathcal{A}_1$	b_3	a_5	$\mathcal{A}_2$	a_1	$\mathcal{A}_3$	a_3	$\mathcal{A}_4$	a_1	a_3	$\mathcal{A}_5$	a_3	a_5	$\mathcal{A}_6$	a_1	b_3
x_1	1	1	x_2	0	x_3	0	x_4	1	1	x_5	1	1	x_6	1	1

$F(\mathcal{A}_1) =\uparrow$[2], $F(\mathcal{A}_2) = a_5$, $F(\mathcal{A}_3) = b_3$, $F(\mathcal{A}_4) =\uparrow$, $F(\mathcal{A}_5) =\uparrow$, $F(\mathcal{A}_6) =\uparrow$, $\alpha(1) = a_3$[3], $\alpha(2) = a_1$, $\beta(1) = \mathcal{A}_1$, $\beta(2) = \mathcal{A}_5$, $\beta(3) = \mathcal{A}_6$, $\beta(4) = \mathcal{A}_4$.

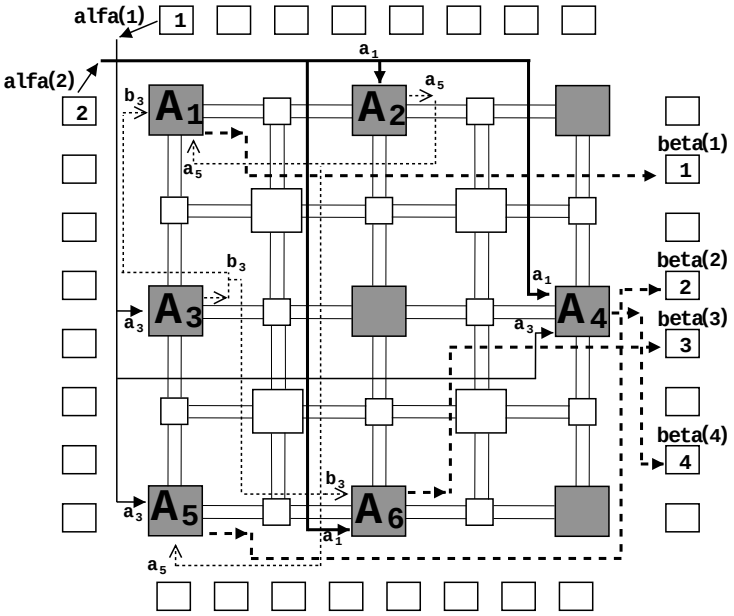

Figura 4.1: Um decodificador "2 to 4", representado pelo modelo *Chu net*

Uma outra configuração de $\mathfrak{R}$, mostrada na figura 4.2, é a Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, onde $\boldsymbol{C} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$, com

$\mathcal{A}_1$	a_1	b_1	$\mathcal{A}_2$	a_1	$\mathcal{A}_3$	a_3	$\mathcal{A}_4$	a_4	b_4	$\mathcal{A}_5$	a_5	a_3	$\mathcal{A}_6$	a_1	a_3
x_1	1	1	x_2	0	x_3	0	x_4	1	1	x_5	1	1	x_6	1	1
							y_4	1	0						
							z_4	0	1						

[2]A função F é indefinida para $\mathcal{A}_1$

[3]Na figura 4.1, as funções α e β são representadas, respectivamente por "alfa"e "beta"

$F(\mathcal{A}_1) = b_4$, $F(\mathcal{A}_2) = a_5$, $F(\mathcal{A}_3) = b_1$, $F(\mathcal{A}_4) = \uparrow$, $F(\mathcal{A}_5) = a_4$, $F(\mathcal{A}_6) = \uparrow$, $\alpha(1) = a_1$, $\alpha(2) = a_3$, $\beta(1) = \mathcal{A}_4$ e $\beta(2) = \mathcal{A}_6$.

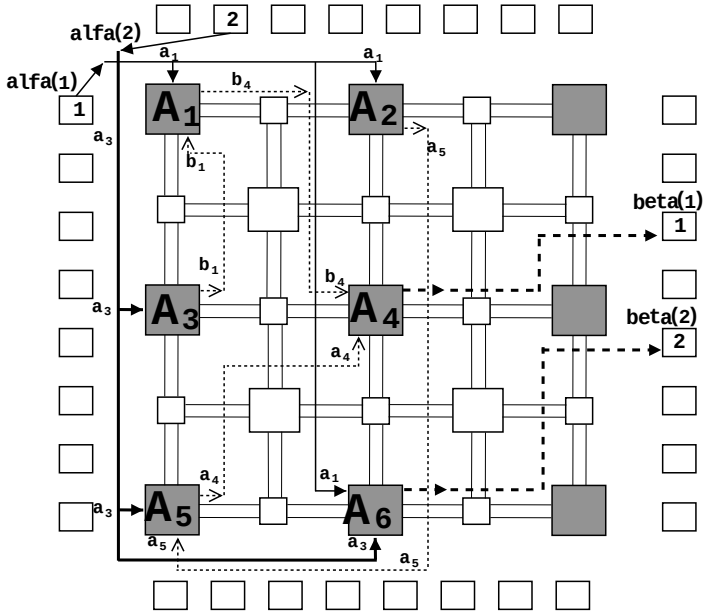

Figura 4.2: Um meio somador, representado pelo modelo *Chu net*

Assim, uma Chu net $(\boldsymbol{C}, F, \alpha, \beta)$ pode ser um modelo de uma configuração de uma Arquitetura Reconfigurável, onde os Espaços de Chu em $\boldsymbol{C}$ representam as unidades reconfiguráveis, o conjunto $\{1, \ldots, m\}$ modela as m portas utilizadas como entrada e $\{1, \ldots, n\}$ modela as n portas das unidades reconfiguráveis, F modela a conexão entre as portas de saídas e entradas das unidades reconfiguráveis

4.3 Chu nets como multi-Espaços de Chu

É possível associar um multi-Espaço de Chu a uma Chu net, e consequentemente a uma função booleana. Dessa forma, se cada Espaço de Chu modela uma função booleana, então pode-se pensar na composição de Espaços de Chu.

Definição 4.5 *A composição dos Espaços de Chu $\mathcal{A}_1, \ldots, \mathcal{A}_n$ com o Espaço de Chu $\mathcal{A}$, e A é o conjunto de eventos de $\mathcal{A}$, com respeito à bijeção $\tau\{1, \ldots, n\} \longrightarrow A$, onde $n = |A|$, é denotado pelo Espaço de Chu $\langle \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle \succ \mathcal{A}$, que é equivalente a:*

$$\varphi^{-1}(FNDC(\varphi(\mathcal{A})[(\varphi(\mathcal{A}_1), \varrho(\tau_{\mathcal{A}}(1))), \ldots, (\varphi(\mathcal{A}_n), \varrho(\tau_{\mathcal{A}}(n)))])),$$

onde $\alpha[(\beta_1, p_1), \ldots, (\beta_n, pn)]$ é a substituição de todas as ocorrências do símbolo proposicional p_i em α pela fórmula β_i, com $i = 1, \ldots, n$ [6].

$\varphi^{-1}(FNDC(\alpha))$	a_1	b_1	a_2	b_2
x_1	1	1	1	1
x_2	1	1	1	0
x_3	1	1	0	1
x_4	1	1	0	0
$x_5 = x_1$	1	1	1	1
x_6	1	0	1	1
x_7	0	1	1	1
x_8	0	0	1	1

Tabela 4.1: A completude de α

Exemplo 4.1 *Dados os* **Espaços de Chu** $\mathcal{A}_1$, $\mathcal{A}_2$ *e* $\mathcal{A}$

$\mathcal{A}_1$	a_1	b_1
x_1	*1*	*1*

$\mathcal{A}_2$	a_2	b_2
x_2	*1*	*1*

$\mathcal{A}$	a	b
x	*1*	*1*
y	*1*	*0*
z	*0*	*1*

representados pelas fórmulas proposicionais

- $\varphi(\mathcal{A}_1) = a_1 b_1$
- $\varphi(\mathcal{A}_2) = a_2 b_2$
- $\varphi(\mathcal{A}) = ab \vee a\bar{b} \vee \bar{a}b$

Dessa forma, considerando que $\tau_{\mathcal{A}}(1) = a$ e $\tau_{\mathcal{A}}(2) = b$ é a enumeração dos eventos de $\mathcal{A}$, tem-se que $\varphi(\mathcal{A})[(\varphi(\mathcal{A}_1, a), (\varphi(\mathcal{A}_2, b)]$ é a fórmula

$$(a_1 b_1)(a_2 b_2) \vee (a_1 b_1)\overline{(a_2 b_2)} \vee \overline{(a_1 b_1)}(a_2 b_2)$$

Mas essa fórmula não está na FNDC, sua FND, Forma Normal Disjuntiva é:

$$\alpha = a_1 b_1 \vee a_2 b_2$$

Para torná-la completa são necessários 7 fórmulas conjuntivas de 4 literais cada um. O Multi-Espaço de Chu associado $(\varphi^{-1}(FNDC(\alpha)))$ é o mostrado na tabela 4.1.

Definição 4.6 *Seja* $\mathcal{C} = (\boldsymbol{C}, F, \alpha, \beta)$*, obtemos* $\mathcal{M}_C = (\mathbb{M}, \gamma)$ *onde* $\mathbb{M}$ *e* γ *através do algoritmo:*

```
Begin
   𝕄 := ∅
   γ := ∅
   For i = 1 to n Do
   Begin
      𝒜 := β(i)
      While A ⊈ imag(α) Do
      Begin
         k := |A|
         For j = 1 to k Do
         Begin
            If A[j] ∈ imag(α) Then
               𝒜_j = ({A[j]}, {x}, R(x, A[j]) = 1)
            Else
               𝒜_j := F^{-1}(A[j])
         End
         𝒜 := ⟨𝒜_1, ..., 𝒜_k⟩ ⋗ 𝒜
      End
      B := imag(α) − A
      𝕄 := 𝕄 ∪ 𝒜_{+B}
      γ := γ ∪ {(i, 𝒜_{+B}}
   End
End
```

onde n é a quantidade de portas de saída, ou seja $|dom(\beta)|$ e $A[j]$, em uma enumeração de A, representa o j-ésimo elemento de A. Dessa forma, $\mathcal{M}_C$ representa a Chu net $\mathcal{C}$.

Exemplo 4.2 *Aplicando o algoritmo da definição 4.6 à Chu net na figura 4.1, tem-se que o multi-Espaço de Chu da tabela 4.2 representa-o:*

		a_1	a_3
$\gamma(1)$	x_1	0	0
$\gamma(2)$	x_2	0	1
$\gamma(3)$	x_3	1	0
$\gamma(4)$	x_4	1	1

Tabela 4.2: Multi-Espaço de Chu decodificador

Exemplo 4.3 *Aplicando o algoritmo à Chu net na figura 4.2, tem-se que o multi-Espaço de Chu da tabela 4.3 representa-o:*

		a_1	a_3
$\gamma(1)$	x_1	0	1
	x_2	1	0
$\gamma(2)$	x_3	1	1

Tabela 4.3: multi-Espaço de Chu meio somador

4.4 Multi-Chu nets

Definição 4.7 *Seja $\boldsymbol{C}$ um conjunto finito de multi-Espaços de Chu, $\mathcal{C} = \bigcup_{\mathcal{M}\in\boldsymbol{C}} \mathbb{M}$, $\boldsymbol{A} = \bigcup_{\mathcal{M}\in\boldsymbol{C}} A$, $F : \mathcal{C} \longrightarrow \boldsymbol{A}$ uma função injetora parcial, m e n inteiros positivos, e $\alpha : \{1,\ldots,m\} \longrightarrow \boldsymbol{A}$ e $\beta : \{1,\ldots,n\} \longrightarrow \mathcal{C}$ funções injetoras. $(\boldsymbol{C}, F, \alpha, \beta)$ é uma* **multi-Chu net** *de m entradas e n saídas se as seguintes condições forem satisfeitas [6]:*

1. $imag(\alpha) \cap imag(F) = \emptyset$
2. $imag(\alpha) \cup imag(F) = \boldsymbol{A}$
3. $imag(\beta) \cup dom(F) = \mathcal{C}$

A dimensão de uma multi-Chu net é a dimensão máxima de seus multi-Espaços de Chu. Naturalmente, cada Chu net pode ser vista como uma multi-Chu net de dimensão 1.

Uma multi-Chu net $\mathcal{M}$, m por n, é uma configuração de uma Chu net (multi-Chu net reconfigurável) $(r, \vec{p}, k)$ se $|\boldsymbol{C}| \leq r$,$m+n \leq k$ e existe uma enumeração $\mathcal{M}_1, \ldots, \mathcal{M}_r$ de $\boldsymbol{C}$ de forma que $|\boldsymbol{A}_i| + n_i \leq p_i$, onde n_i é a dimensão do multi-Espaço de Chu $\mathcal{M}_i$.

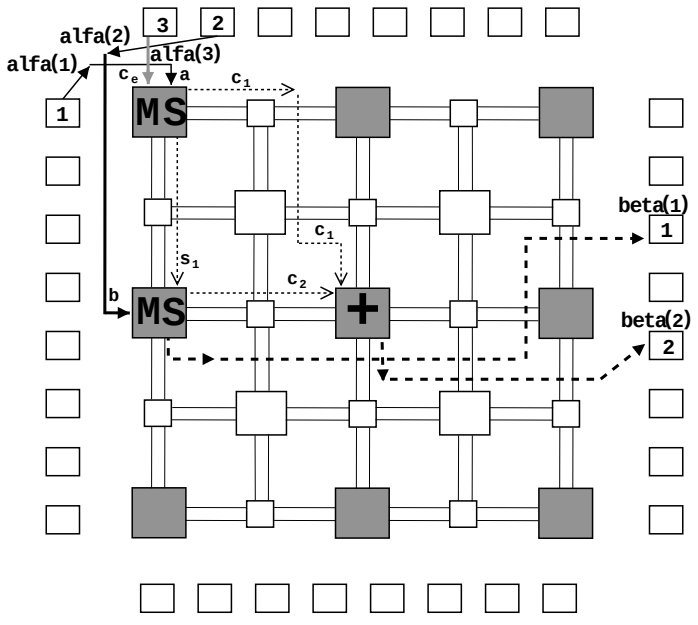

Figura 4.3: Representação de um FPGA somador completo através de Espaços de Chu

Exemplo 4.4 *Seja $\mathcal{MS}$ o multi-Espaço de Chu do exemplo 4.3. Então a multi-Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, na figura 4.3, é uma configuração para a multi-Chu net reconfigurável $(9, (6,6,6,6,6,6,6,6,6), 32)$, onde $\boldsymbol{C} = \{\mathcal{MS}_1, \mathcal{MS}_2, \mathcal{A}\}$, com*

$\mathcal{MS}_1$		a	Ce
$\gamma(1)$	x_1	0	1
	x_2	1	0
$\gamma(2)$	x_3	1	1

$\mathcal{MS}_2$		b	s_1
$\gamma(1)$	x_1	0	1
	x_2	1	0
$\gamma(2)$	x_3	1	1

$\mathcal{A}$		c_1	c_2
$\gamma(1)$	x_1	1	1
	x_2	1	0
	x_3	0	1

Tabela 4.4: Multi-Chu net Somador Completo

e $F(\gamma_{\mathcal{MS}_1}(1)) = s_1$, $F(\gamma_{\mathcal{MS}_1}(2)) = c_1$, $F(\gamma_{\mathcal{MS}_2}(1)) = \uparrow$, $F(\gamma_{\mathcal{MS}_2}(2)) = c_2$, $F(\gamma_{\mathcal{A}(1)}) = \uparrow$, $\alpha(1) = a$, $\alpha(2) = b$, $\alpha(3) = Ce$, $\beta(1) = \gamma_{\mathcal{MS}_2}(1)$ *e* $\beta(1) = \gamma_{\mathcal{A}(1)}$

Este multi-Chu net modela a configuração na figura 2.8 onde as unidades reconfiguráveis $\mathcal{MS}$s são modeladas pelos multi-Espaços de Chu $\mathcal{MS}_1$ e $\mathcal{MS}_2$ e a unidade reconfigurável da disjunção $(+)$ é modelada pelo multi-Espaço de Chu $\mathcal{A}$.

Cada multi-Chu net está associada a um multi-Espaço de Chu e a função associada a ele tem o mesmo comportamento da multi-Chu net.

Exemplo 4.5 *Para o caso da multi-Chu net na figura 2.8, o multi-Espaço de Chu associado é:*

		a	b	Ce
$\gamma(1)$	x_1	1	1	1
	x_2	1	0	0
	x_3	0	1	0
	x_4	0	0	1
$\gamma(2)$	x_1	1	1	1
	x_2	1	1	0
	x_3	1	0	1
	x_4	0	1	1

Tabela 4.5: Multi-Espaço de Chu Somador Completo

$\gamma(1)$ *e* $\gamma(2)$ *são, respectivamente, as saídas S e C mostradas na figura.*

4.4.1 Níveis de detalhamento em multi-Chu nets

Multi-Chu nets podem ser representada por multi-Espaços de Chu, e por outro lado, pode-se ter multi-Chu nets que representam unidades reconfiguráveis em uma multi-Chu net, gerando vários níveis de detalhamento.

Por exemplo, analogamente à figura 2.8, onde a configuração na 4.4 é usada como uma unidade reconfigurável simples, pode-se usar o multi-Espaço de Chu do exemplo 4.4, que é identificado na multi-Chu net 4.6, como parte de uma multi-Chu net.

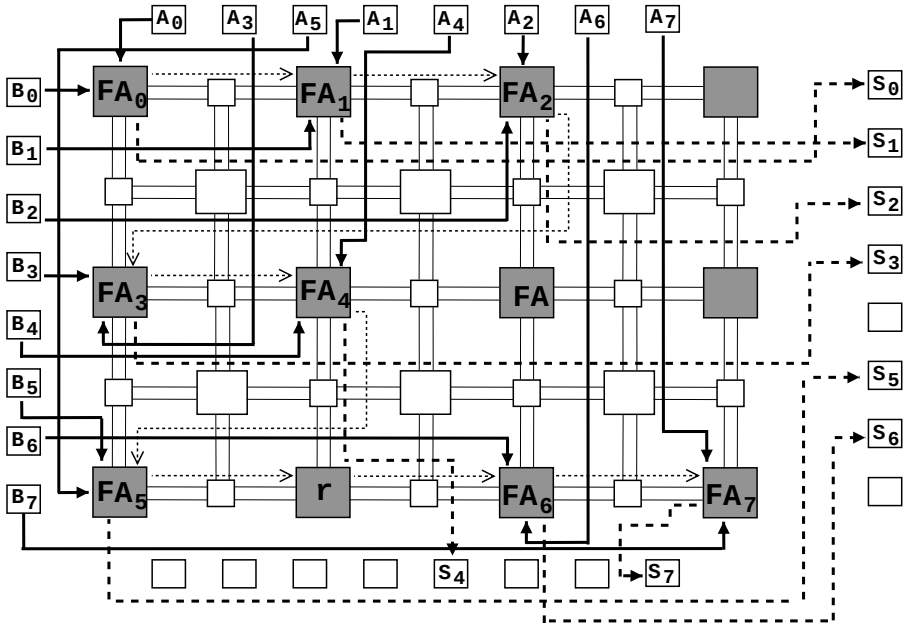

Figura 4.4: Representação através de Espaços de Chu de um fpga somador de 8 bits

Exemplo 4.6 *A multi-Chu net* $(\boldsymbol{C}, F, \alpha, \beta)$*, onde* $\boldsymbol{C} = \{\mathcal{FA}_0, \ldots, \mathcal{FA}_7, \mathcal{R}\}$*, com os Espaços de Chu representados nas tabelas 4.6, 4.7, 4.8, 4.9, 4.10 e* $F(\gamma\mathcal{FA}_0(1)) = \uparrow$*,* $F(\gamma\mathcal{FA}_0(2)) = c_1$*,* $F(\gamma\mathcal{FA}_1(1)) = \uparrow$*,* $F(\gamma\mathcal{FA}_1(2)) = c_2$*, . . .,* $F(\gamma\mathcal{FA}_4(1)) = \uparrow$*,* $F(\gamma\mathcal{FA}_4(2)) = c_5$*,* $F(\gamma\mathcal{FA}_5(1)) = \uparrow$*,* $F(\gamma\mathcal{FA}_5(2)) = r$*,* $F(\gamma\mathcal{R}(1)) = c_6$*,* $F(\gamma\mathcal{FA}_6(1)) = \uparrow$*,* $F(\gamma\mathcal{FA}_6(2)) = c_7$*,* $F(\gamma\mathcal{FA}_7(1)) = \uparrow$*,* $\alpha(1) = a_0$*,* $\alpha(2) = b_0$*, . . .,* $\alpha(15) = a_7$*,* $\alpha(16) = b_7$*,* $\beta(1) = \gamma\mathcal{FA}_0(1), \ldots, \beta(8) = \gamma\mathcal{FA}_7(1)$*.*

Esse modelo multi-Chu net de configuração na figura 2.9, onde as unidades reconfiguráveis FA's são modeladas pelos multi-Espaços de Chu $\mathcal{FA}_0, \ldots, \mathcal{FA}_7$ *e a unidade reconfigurável* r *é modelada pelo multi-Espaço de Chu* $\mathcal{R}$

$\mathcal{FA}_0$		a_0	b_0
$\gamma(1)$	x_1	0	1
	x_2	1	0
$\gamma(2)$	x_3	1	1

Tabela 4.6: Meio-Somador

⋮

$\mathcal{FA}_1$		a_1	b_1	C_1
$\gamma(1)$	x_1	1	1	1
	x_2	1	0	0
	x_3	0	1	0
	x_4	0	0	1
$\gamma(2)$	x_1	1	1	1
	x_2	1	1	0
	x_3	1	0	1
	x_4	0	1	1

Tabela 4.7: Somador 1

$\mathcal{FA}_6$		a_6	b_6	C_6
$\gamma(1)$	x_1	1	1	1
	x_2	1	0	0
	x_3	0	1	0
	x_4	0	0	1
$\gamma(2)$	x_1	1	1	1
	x_2	1	1	0
	x_3	1	0	1
	x_4	0	1	1

Tabela 4.8: Somador 6

$\mathcal{FA}_7$		a_7	b_7	C_7
$\gamma(1)$	x_1	1	1	1
	x_2	1	0	0
	x_3	0	1	0
	x_4	0	0	1

Tabela 4.9: Somador 7 sem saída *carry*

$\mathcal{R}$		r
$\gamma(1)$	x_1	1

Tabela 4.10: Somador de 8 bits

Capítulo 5

Metodologia

Neste capítulo, serão apresentados os artifícios desenvolvidos para a representação de Arquiteturas Reconfiguráveis com chu nets, num nível mais próximo a implementação, isto é, a representação em formato persistente de uma configuração de uma Arquitetura Reconfigurável. Será descrito ainda a metodologia de criação e implementação dos algoritmos de Composição de CLBs, detecção dos vetores de controlabilidade e observabilidade de Arquiteturas Reconfiguráveis, bem como o modo como esses algoritmos manipulam os modelos chu.

5.1 Formato Chu

Na seção 4.2, foi mostrado as Chu nets, que são um modelo matemático para representar configurações de Arquiteturas Reconfiguráveis, no entanto, para a manipulação de uma chu net através de um algoritmo é necessário a descrição das chu nets em nível mais próximo a implementação. Para atender essa necessidade, desenvolveu-se o formato de arquivo denominado de **chu**, que contém a descrição de uma configuração para uma arquitetura reconfigurável, e como será mostrado na seção 5.4, é utilizado como entrada para os algoritmos de composição de CLBs, controlabilidade e observabilidade de arquiteturas reconfiguráveis. A estrutura de um arquivo **chu** pode ser visualizada na figura 5.1.

Um arquivo no formato **chu**, como está mostrado na figura 5.1[1], deve conter na linha 1 a informação de quantos eventos a chu net possui, nesse valor estão inclusos os eventos de entrada e os eventos internos, ou seja, que não são saídas da chu net. Na linha 2 do arquivo, deve ser expressa a quantidade de Espaços de Chu, que compõe a chu net,

[1] As linhas foram numeradas para facilitar a compreensão, no entanto, o arquivo real não deve conter esse artifício

```
1  qtde_eventos
2  qtde_espchu
3  qtde_entradas
4  qtde_saidas
5
6  qes qev
7  indices_eventos
8  matriz_espchu
9
10 id_espchu qtde_ev_saida indices_ev_saida
11
12 indices_eventos_entradas
13
14 indices_espchus_saidas
```

Figura 5.1: Estrutura de um arquivo no formato **chu**

enquanto que as linhas 3 e 4, possuem, respectivamente, as informações de quantas são as entradas e as saídas da chu net. As linhas 5, 9, 11 e 13 são linhas em branco, que devem necessariamente existir no arquivo **chu**, para separar informações que não estão relacionadas. Nas linhas 6, 7 e 8 temos informações relativas a um Espaço de Chu: na linha 6 tem-se a informação de quantos estados e quantos eventos, necessariamente nessa ordem, estão presentes naquele Espaço de Chu; na linha 7, estão separados por espaço em branco os índices dos eventos que compõem no Espaço de Chu; finalmente, na linha 8 deve estar descrita a matriz que representa o Espaço de Chu, cuja a quantidade de colunas e linhas são expressas, respectivamente, pela quantidade de eventos e estados do Espaço de Chu - espaços em branco devem separar os valores em uma mesma linha da matriz, e uma quebra de linha deve separar as linhas; Todos os Espaços de Chu da chu net devem ser decritos da forma especificada nas linhas 6, 7 e 8, uma quebra de linha deve separá-los entre si, e os mesmos devem aparecer na ordem com a qual foram identificados. Na linha 10, estão separadas por espaço em branco o indentificador de um Espaço de Chu, a quantidade de eventos que são saídas deste Espaço de Chu e os índices dos mesmos; deve existir uma linha, com as informações contidas na linha 10 para cada Espaço de Chu da chu net, inclusive para os Espaços de Chu que são saídas do sistemas, onde o identificador de evento será substituido pelo número 0, como forma de indicar essa situação especial. Na linha 12, tem-se os identificadores dos eventos de entrada da chu net, que estão separados por quebras de linha, e ordenados de acordo com o índice da entrada representada pelo evento. Finalmente, na linha 13, tem-se os identificadores dos espaços de chu, cujas saídas são saídas diretas da arquitetura reconfigurável, separados por quebras de linha, e ordenados de acordo com o índice da saída representada pelo Espaço de Chu.

Na figura 4.2, localizada da seção 4.2, está representada uma arquitetura reconfigurá-

vel cujo o arquivo **chu** correspondente está identificado na figura 5.2.

```
6
6
2
2

1 2
1 2
1 1

1 1
1
0

1 1
3
0

3 2
4 5
1 0
0 1
1 1

1 2
6 3
1 1

1 2
1 3
1 1

1 1 5
2 1 6
3 1 2
4 1 0
5 1 4
6 1 0

1
3

4
6
```

Figura 5.2: Arquivo no formato **chu** com o modelo de um meio somador

Na representação em arquivo, os eventos e espaços de chu possuem um número associado, e não um nome, como na figura 4.2. Dessa forma, os eventos estão representados no arquivo mostrado na figura 5.2 com a seguinte equivalência: $a_1 = 1$,$b_1 = 2$, $a_3 = 3$, $a_4 = 4$, $b_4 = 5$ e $a_5 = 6$; já os espaços de chu obedecem a seguinte equivalência: $\mathcal{A}_1 = 1$, $\mathcal{A}_2 = 2$, $\mathcal{A}_3 = 3$, $\mathcal{A}_4 = 4$, $\mathcal{A}_5 = 5$ e $\mathcal{A}_6 = 6$; os espaços de chu 1, 5 e 6 representam a função lógica And, os espaços de chu 2 e 3 representam a função Not, enquanto o espaço de chu 4 representa um Or, essa informação é percebida ao se observar o arquivo, pois a matriz que representa o Espaço de Chu, possui todas as combinações de eventos de entradas que resultam no valor lógico verdadeiro (1).

5.2 BDD Library - BDDlib

Como demonstrado na seção 3.1.3, uma das possibilidades para representar Espaços de Chu é através de BDDs. Dessa forma, optou-se por utilizar BDDs para representar os Espaços de Chu no momento da implementação dos algoritmos, especificamente na representação em memória das chu nets, uma vez que na representação persistente, utilizou-se matrizes. A decisão de utilizar BDDs foi motivada pela existência de uma biblioteca, denominada de BDD Library [36], que disponibiliza funções que facilitam a manipulação de funções lógicas booleanas, representadas através de BDDs.

A biblioteca BDD Library foi desenvolvida em C pelo *Model Checking Group* que integra o *Specification and Verification Center* da *Carnegie Mellon University*, visando a sua utilização em verificação sequência de sistemas. A BDDlib tem como principais características:

- estrutura para a verificação seqüêncial de sistemas;
- rotinas para prover informações sobre nós, emitindo assim informações como número de nós utilizados e analise do tamanho do BDD;
- rotinas para armazenamento e recuperação de BDDs em arquivos;
- coleta de lixo através de contagem à referência;
- alteração no limite de nós do BDD;
- suporte para reordenação dinâmica de variáveis;

A maioria das rotinas fornecidas pela biblioteca BDD requerem um gerenciador de BDD como um de seus argumentos. Um gerenciador de BDD é uma estrutura que armazena as variáveis dos BDDs. Dessa forma, ao se utilizar BDDs é necessário a presença de uma variável do tipo **bdd_manager**, que fará o papel do gerenciador de BDDs.

As variáveis booleanas presentes em um BDD são mantidas segundo uma ordenamento global, de acordo com a ordem que aparecem no BDD. Cada variável tem um índice, que representa a sua posição de ordenação, e um número de identificação (ID) que é invariável, dessa forma, variáveis podem ser criadas em qualquer posição dentro da ordem, o mesmo não ocorre para o seu índice. A biblioteca suporta reordenamento dinâmico de variáveis, como forma de diminuir a quantidade de nós de um BDD.

No apêndice A há uma listagem de todas as funções da biblioteca BDD Library utilizada na implementação dos algoritmos de composição de CLBs, controlabilidade e observabilidade de arquiteturas reconfiguráveis.

5.3 Representação do formato chu através de BDD

Na seção 5.1, foi definido o formato **chu** utilizado pelos arquivos que funcionam como entrada para os algoritmos de composição, controlabilidade e observabilidade. No formato **chu**, os Espaço de Chu são representados através de matrizes, no entanto, na seção 5.2, mencionou-se que na representação em memória dos algoritmos, os Espaços de Chu são representados através de BDD, e não por matrizes, uma vez que essa medida simplifica a implementação. Então por que não inserir no arquivo **chu** a representação BDD dos Espaços de Chu? Não decidiu-se por fazer a representação através de BDD, por acreditar-se que o BDD não é fácil de ser escrito, principalmente comparado a representação matricial que apresenta-se muito mais intuitiva, simples e "amigável" para os usuários.

Uma vez que fez-se opção por uma representação em memória utilizando BDDs, tornou-se necessário representar a chunet através de BDD's, foi definido então a estrutura mostrada na figura 5.3, que contém as mesmas informações contidas no modelo matemático de uma Chu net, mostrado na seção 4.2.

```
1 /* Definindo as estruturas de dados */

2  //Evento
3  typedef bdd event;

4  //Espaco de Chu
5  typedef bdd chu;

6  //Elemento do conjunto F (F: C -> A)
7  typedef struct{
8    chu fdom;
9    event * fran;
10   int sfran;
11 }felem;

12 //Chu net
13 typedef struct{
14   chu * cset;
15   felem * fset;
16   event * alpha;
17   chu * beta;
18   int sev;
19   int size;
20   int m;
21   int n;
22 }chunet;
```

Figura 5.3: Estrutura de uma chunet, definida no arquivo **chu.h**

As linhas demonstradas na figura 5.3 é parte do arquivo de implementação chu.h.

Na linha 3 da figura 5.3, está definido o tipo `event`, a partir do tipo `bdd`, isto é possível pelo fato do evento ser uma variável lógica que pode assumir valor `verdadeiro` ou `falso`, dessa forma, pode ser representado por um BDD composto apenas pelo nó raiz e suas folhas, como demonstrado na figura 5.4.

Na linha 3 define-se o tipo `event`, a partir do tipo `bdd`, pois como pode ser observado na figura 5.4 um evento é uma variável que pode assumir valor *verdadeiro* ou *falso*, logo pode ser representado como um BDD composto apenas pelo nó raiz e folhas.

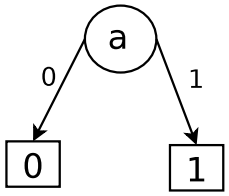

Figura 5.4: BDD que representa um evento

Na linha 5, está definido o tipo `chu`, também a partir de `bdd`, como já foi mostrado anteriormente, na seção 3.1.3, é possivel essa representação. Entre as linhas 13 e 22 é definida a estrutura `chunet`, que contém todas as informações contidas no arquivo **chu** representadas através de BDD: na linha 14 define-se um vetor de `chu`, denominado de `cset`, que contém os Espaços de Chu da chu net; Na linha 15, define-se um vetor de `felem`, estrutura que está definida entre as linhas 7 e 11, para representar os elementos do vetor `fset`, que associa cada `bdd` espaço de chu aos seus eventos de saída; Nas linhas 16 e 17 estão especificados, respectivamente, um vetor de eventos, que armazenará os eventos de entrada da chu net, e um vetor de chu, que guardará os espaços de chu, cuja a saída pertence as saídas do sistema. Nas linhas 18, 19, 20 e 21 define-se as variáveis que guardarão, nessa ordem, quantidade de eventos da `chunet`, quantidade de espaços de chu, quantidade de eventos de entrada e quantidade de saídas da `chunet`.

Uma vez definida a estrutura `chunet` que será manipulada pelos algoritmos, é necessário conhecer o mecanismo de "tradução" do arquivo para a estrutura, que tem seus passos listados a seguir:

- Todos os `event` são criados utilizando-se a função `bdd_new_var_last` da biblioteca BDDlib, listada no apêndice A, e armazenados em um vetor, de acordo com a sua ordem de criação.
- É realizada a construção do conjunto `cset`, a partir da construção de cada Espaço de chu, que segue as seguintes etapas:

    1. Os eventos que são entradas do Espaço de Chu a ser construído são armazenados em um vetor;

2. É criada em memória uma matriz com o Espaço de Chu que está descrito no arquivo **chu**;
3. O vetor e a matriz resultantes das etapas 1 e 2 são enviados para a função `makechu`, que utiliza as funções `bdd_not`, `bdd_and` e `bdd_or` - listadas no apêndice A- para construir um `bdd` que é uma função lógica booleana representada na FNDC.

- São criados e armazenados os elementos do conjunto `fset` apenas associando os Espaços de Chu com os seus eventos de saída, informações estas contidas no arquivo **chu**
- Finalmente, são criados os conjuntos `alpha`, com eventos que já foram criados, a partir das informações do arquivo **chu** e `beta`, com os `bdd` que já foram criados.

5.4 Algoritmos

Nesta seção será descrita a metodologia de construção e implementação dos algoritmos de composição de CLBs , controlabilidade e observabilidade de Arquiteturas Reconfiguráveis, como modo de demonstrar através dessas aplicações, a eficiência da utilização de Espaços de Chu na modelagem de Arquiteturas Reconfiguráveis. Os algoritmos descritos nessa seção, foram implementados utilizando a linguagem de programção C e a biblioteca BDD Library.

5.4.1 Composição de CLBs

O algoritmo de composição foi desenvolvido com o objetivo de efetuar a composição de células lógicas programáveis, com base na operação definida na seção 4.3 e no algoritmo da definição 4.6.

A entrada para o algoritmo da Composição é um arquivo no formato **chu** que contém a descrição da Chu net sobre a qual a composição atuará, e como saída, será gerado um arquivo, contendo os BDDs que representam as saídas da Chu net do arquivo **chu** compostas em função das entradas, na seção 7.2, será demonstrada a intenção de desenvolver-se uma função para traduzir os BDDs resultantes da Composição em um arquivo **chu**.

Na implementação do algoritmo da Composição, a estratégia utilizada é a mesma empregada no algoritmo da definição 4.6, ou seja, a partir dos Espaços de Chu de saída, constrói-se as funções dos Espaços de Chu em função das suas entradas, até que a função

inteira esteja descrita em função das variáveis de entrada da Arquitetura reconfigurável. Na figura 5.5 tem-se a representação apenas da parte do circuito que está relacionada com a primeira saída.

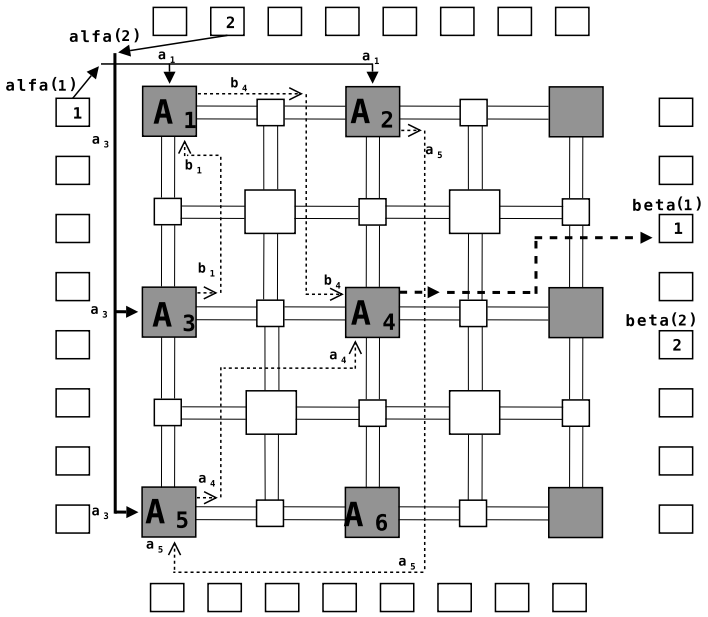

Figura 5.5: Arquitetura que representa parte de um FPGA meio-somador.

Um arquivo **chu** que representasse a configuração na figura 5.5, ao ser submetido ao algoritmo de composição, seguiria a seguinte sequência de passos até o resultado final:

1. Inicia a avaliação da Chu net a partir do Espaço de Chu A_4 (Saída);

2. Verifica-se se os eventos de entrada de A_4, a_4 e b_4, são entradas do sistema, o que não é verdade para nenhum dos dois eventos. Neste ponto, tem-se que a função computada na saída 1 é $a_4 \vee b_4$[2];

3. Um dos eventos é escolhido, a_4, por exemplo, e através da composição, ele é substituido pela função da CLB (Espaço de Chu) do qual é saída, A_5. Neste ponto, sabe-se que a função computada na saída 1 é $(a_3 \wedge a_5) \vee b_4$;

4. Agora, ocorre a substituição do outro evento de entrada do Espaço de Chu A_5, b_4, pela função da CLB do qual é saída, A_1. Neste ponto, a função computada na saída 1 é $(a_3 \wedge a_5) \vee (a_1 \wedge b_1)$;

5. Verifica-se se os eventos que compõem a equação pertencem aos eventos de entrada da Chu net, os que não pertencem, vão sendo substituídos, da mesma forma que os eventos a_4 e b_4, pela função computada pela CLB da qual são saídas. Neste ponto tem-se a função $(a_3 \wedge \neg a_1) \vee (a_1 \wedge \neg a_3)$

[2]Sabe-se que a função computada é um OR por causa das outras informações contidas na Chu net, como o conjunto de Espaços de Chu.

6. O algoritmo pára de compor CLBs quando é verificado que todos os eventos que compõem a fórmula resultante da composição, pertencem aos eventos de entrada da Chu net.

7. Finalmente, é impresso no arquivo de saída, o BDD resultante da composição para cada saída da Chu net.

5.4.2 Controlabilidade de Arquiteturas Reconfiguráveis

Como visto na seção 2.4.1, a controlabilidade em Arquiteturas Reconfiguráveis é uma característica desejável aos projetos de arquiteturas que visam a testabilidade [25]. O uso da controlabilidade, aliada a observabilidade é bastante utilizado para a detecção de falhas em projetos [22], a idéia é simples: controlar pontos e observá-los de forma a atestar a existência, ou não, de problemas.

O algoritmo de controlabilidade foi desenvolvido com o intuito de detectar para cada ponto do sistema (eventos), o conjunto de valores de entrada do sistema que originam valores lógicos, 0, para a controlabilidade em zero, e 1 para a controlabilidade em 1, naquele ponto. O algoritmo desenvolvido está especificado na definição 5.1.

Definição 5.1 *Seja* $\mathcal{C} = (\boldsymbol{C}, F, \alpha, \beta)$*, e* $\mathcal{M}_C = (\mathbb{M}, \gamma)$*, obtem-se* $RcC = \{rC_1, rC_2, \ldots, rC_m\}$*, onde* $rC_i = (id, A_z, A0)$*, com* $A_z = \{V_1, V2, \ldots, V_k\}$ *e* $A_o = \{V_1, V2, \ldots, V_w\}$ *através do algoritmo:*

$\mathcal{M}_C :=$ `compose`$(\mathcal{C})$
`controllability(`$\mathcal{C}$`){`
  $A := \bigcup_{\mathcal{A} \in C} A$
  $RcC := \emptyset$
  `For all` $e \in A$ `do{`
    $RcC := RcC \cup$ `control(`e`)`
  `}`
  `For all` $o \in \mathbb{M}$ `do{`
    $RcC := RcC \cup$ `scontrol(`o`)`
  `}`
`}`

```
//Função control
```
`control(`e`){`
  $rC.id :=$ `identifier(`e`)`
  $rC.A_z := \emptyset$
  $rC.A_o := \emptyset$
  `If` $e \in imag(\alpha)$ `then {`

$rC.A_z := \{\{id_1 := x, id_2 := x, \ldots, id_{rC.id} := 0, \ldots, id_{|\alpha|} := x\}\}$
$rC.A_o := \{\{id_1 := x, id_2 := x, \ldots, id_{rC.id} := 1, \ldots, id_{|\alpha|} := x\}\}$ }
Else{
$\mathcal{C}_e := (C, F <+ \{F^{-1}(e), \uparrow\}, \alpha, \beta := \{(1, F^{-1}(e))\})$
$\mathcal{M}_e :=$ compose($\mathcal{C}_e$)
For all $V := \{\{id_1 := v_1, id_2 := v_2, \ldots, id_n := v_n, \}\}/\chi_{\mathcal{M}_e[1]}(V) = 0$ do{
$rC.A_z := rC.A_z \cup V$
}
For all $V := \{\{id_1 := v_1, id_2 := v_2, \ldots, id_n := v_n, \}\}/\chi_{\mathcal{M}_e[1]}(V) = 1$ do{
$rC.A_o := rC.A_o \cup V$
}
}
retorne(rC)
}

```
//Função scontrol
```
scontrol(o){
For all $V := \{\{id_1 := v_1, id_2 := v_2, \ldots, id_n := v_n, \}\}/\chi_o(V) = 0$ do{
$rC.A_z := rC.A_z \cup V$
}
For all $V := \{\{id_1 := v_1, id_2 := v_2, \ldots, id_n := v_n, \}\}/\chi_o(V) = 1$ do{
$rC.A_o := rC.A_o \cup V$
}
retorne(rC)
}

onde m é a quantidade de eventos de Chu net, incluindo-se as saídas do sistema, RcC é o vetor de Controlabilidade da mesma Chu net, que para cada evento, contém uma estrutura rC, que contém: id do evento; A_z - matriz de controlabilidade em zero; A_o - matriz de controlabilidade em um; k e w são as quantidades de combinações dos eventos de entrada que levam o evento id, respectivamente, aos valores 1 e 0.

A entrada para o algoritmo da Controlabilidade é um arquivo no formato **chu** que contém a descrição da Chu net sobre a qual serão detectados os vetores de controlabilidade 0 e 1 para seus eventos. Como saída do processamento do algoritmo tem-se um arquivo, contendo para cada evento, e saída da Chu net, os vetores de controlabilidade 0 e 1.

Na implementação do algoritmo de Controlabilidade, utilizou-se a lógica empregada no algoritmo da definição 5.1, ou seja, é criada uma Chu net modificada cujo o evento para o qual deseja-se encontrar os vetores de controlabilidade é inserido com um evento de saída, é então feita a composição desta Chu net modificada, obtendo-se como saída a função computada naquela "saída", em função das entradas do sistema. Posteriomente,

cria-se a matriz de controlabilidade simplesmente gerando vetores de todas as combinações possíveis das entradas, e verificando-se qual o valor resultante daquela combinação, os vetores que resultam em 1 irão compor a matriz de controlabilidade em 1, enquanto os vetores que resultam em 0 irão compor a matriz de controlabilidade em 0. As entradas de uma Chu net são diretamente controláveis.

Quando executa-se o algoritmo da Controlabilidade, tendo como entrada a Arquitetura Reconfigurável na figura 5.5, expressa através de um arquivo **chu**, segue-se a seguinte sequência de passos, para detectar os vetores de controlabilidade para o evento b_4:

1. Cria-se uma Chu net modifica cujos elementos são (C, F, α, A_1), ou seja, o Espaço de Chu cuja a saída é o evento b_4, é o único elemento do conjunto β;

2. Utiliza-se a função `compose` para efetuar a composição da Chu net modificada. Dessa forma é obtida a função calculada pela única saída do sistema, que é denotada pela fórmula $a_1 \wedge \neg a_3$;

3. São geradas todas as combinações possíveis a partir das entradas da Chu net, como são 2 entradas, tem-se 2_2 combinações:

a_1	a_3	$a_1 \wedge \neg a_3$
0	0	0
0	1	0
1	0	1
1	1	0

4. É feita a avaliação dos vetores gerados no item 3, se a combinação resulta em 0 na saída da Chu net modificada, o vetor é inserido na matriz de controlabilidade em 0, caso contrário é inserido na matriz de controlabilidade em 1;

$\mathcal{A}_z$	a_1	a_3
	0	0
	0	1
	1	1

$\mathcal{A}_o$	a_1	a_3
	1	0

5. Finalmente, as matrizes são impressas no arquivo de saída.

5.4.3 Observabilidade de Arquiteturas Reconfiguráveis

Além da controlabilidade, a observabilidade é uma característica importante em Arquiteturas Reconfiguráveism conforme explicado na seção 5.4.2.

O algoritmo de observabilidade foi desenvolvido para detectar o conjunto de valores de entradas que permitem que cada evento seja diretamente visualizado nas saídas do sistema. O algoritmo desenvolvido está especificado na definição 5.2.

Definição 5.2 *Seja* $\mathcal{C} = (\boldsymbol{C}, F, \alpha, \beta)$*,* $\mathcal{M}_C = (\mathbb{M}, \gamma)$*, obtem-se* $RoC = \{rC_1, rC_2, \ldots, rC_m\}$*, onde* $rC_k = (id_e, id_o, A)$*, com* $A = \{V_1, V2, \ldots, V_w\}$*, através do algoritmo:*

$\mathcal{M}_C :=$ `compose`$(\mathcal{C})$
`observability(`$\mathcal{C}$`){`
  $A := \bigcup_{\mathcal{A} \in C} A$
  $RoC := \emptyset$
  `For all` $e \in A$ `do{`
    `If` $e \notin imag(\alpha)$ `then {`
      $\mathcal{C}_p := (C, F, \alpha \cup \{(|\alpha| + 1, e)\}, \beta)\})$
    `}`
    `Else {`
      $\mathcal{C}_p := \mathcal{C}$
    `}`
    $\mathcal{MC}_p :=$ `compose(`$\mathcal{C}_p$`)`
    $RoC := RoC \cup$ `observout(`e`)`

  `}`
  `For all` $o \in \mathbb{M}$ `do{`
    $RoC := RoC \cup$ `observout(`o`)`
  `}`
`}`

`//Função observ`
`observ(`e`){`
  $rC := \emptyset$ `For all` $m \in \mathbb{M}_p$ `do{`
    `If dependsOn(`e, m`) then {`
      $A_z := \emptyset$
      $A_o := \emptyset$
    `For all` $V := \{\{id_1 := v_1, id_2 := v_2, \ldots, id_n := v_n, \}\}/\chi_{\mathcal{M}_e[1]}(V) = 0$ `do{`
      $A_z := A_z \cup V$
    `}`
    `For all` $V := \{\{id_1 := v_1, id_2 := v_2, \ldots, id_n := v_n, \}\}/\chi_{\mathcal{M}_e[1]}(V) = 1$ `do{`
      $A_o := A_o \cup V$
    `}`
    $controlC :=$ `control(`e`)`
    $A_z := A_z \cap controlC.A_z$
    $A_o := A_o \cap controlC.A_o$
    $rC := rC \cup \{($`identifier(`e`),identifier(`m`),` $\{A_z \cup A_o\})\}$
    `{`
  `{`

{

//Função observout
observout(o){
$rC := \emptyset$ For all $m \in \mathbb{M}_p$ do{
$rC := rC \cup \{$(identifier(m), indentifier(m), $\emptyset$)$\}$
}
$rC := rC < +\{$(identifier(o), indentifier(o),
$\{id_1 := x, id_2 := x, \ldots, id_w := x\})\}$
{

onde m é a quantidade de eventos de Chu net, incluindo-se as saídas do sistema, RoC é o vetor de Observabilidade da mesma Chu net, que para cada evento, contém uma estrutura rC, que contém: id do evento; ido, o identificador da saída onde o evento é observado;A, matriz de observabilidade; k e w são, respectivamente, a quantidade de registros no conjunto de observabilidade e a quantidade de vetores em um registro de observabilidade.

A entrada para o algoritmo da Observabilidade é um arquivo no formato **chu** que contém a descrição da Chu net sobre a qual serão detectados os vetores de observabilidade para seus eventos. Como saída do processamento do algoritmo tem-se um arquivo, contendo para cada evento, os vetores de observabilidade, identificando inclusive em que saída o evento pode ser observado a partir daquele vetor.

Na implementação do algoritmo de Controlabilidade, utilizou-se a lógica empregada no algoritmo da definição 5.2, onde é criada uma Chu net modificada cujo o evento para o qual deseja-se encontrar os vetores de controlabilidade é inserido como um evento de entrada do sistema, é feita então a composição da Chu net modificada, obtendo-se como saída a função de cada saída, em função das entradas do sistema (inclusive o evento). É então verificado se a função obtida pela composição da Chu net modificada depende do evento, caso a afirmação seja verdadeira, significa que aquele evento pode ser verificado naquela saída, caso contrário, é criado um matriz de observabilidade vazia para o par evento-saida. Caso o evento possa ser observado em uma saída, é criada uma matriz $zero$ com o conjunto de combinações das entradas que resultam em 0, e uma matriz um com o conjunto de combinações das entradas que resultam em 1, em seguida, é realizada a controlabilidade do evento cuja a observabilidade está sendo detectada. Faz-se a interseção entre as matriz $zero$ e a matriz controlabilidade em 0, a fim de detectar se há linhas iguais nas duas matrizes, pois isto significa dizer que há um conjunto de valores de entradas, que produzem o mesmo valor na saída e no evento a ser observado, essas linhas irão compor a matriz de observabilidade. A mesma função interseção é aplicada sobre as

matrizes um e controlabilidade em 1. Já as saídas da Chu net são diretamente observáveis exclusivamente na saída correspondente ao seu identificador.

Quando executa-se o algoritmo da Observabilidade, tendo como entrada a Arquitetura Reconfigurável na figura 5.5, expressa através de um arquivo **chu**, segue-se a seguinte sequência de passos, para detectar os vetores de controlabilidade para o evento b_4:

1. Cria-se uma Chu net modifica cujos elementos são ($C, F, \alpha \cup \{b_4\}, \beta$), ou seja, o evento b_4 também passa a ser entrada do sistema;

2. Utiliza-se a função `compose` para efetuar a composição da Chu net modificada. Dessa forma é obtida a função calculada na saída 1, que é denotada pela fórmula $b_4 \vee a3 \wedge \neg a_3$;

3. Verifica-se, se a fórmula, resultado da composição, depende do evento b_4, o que é afirmativo e leva a execução do passo seguinte;

4. São geradas duas matrizes $zero$ e um que mostram todas as combinações a partir dos eventos de entrada, a_1, a_3 e b_4 que levam a função a $zero$ ou a um:

$zero$	a_1	a_3	b_4
	0	0	0
	1	0	0
	1	1	0

um	a_1	a_3	b_4
	0	0	1
	0	1	0
	0	1	1
	1	0	1
	1	1	1

5. É feita a controlabilidade do evento b_4, como já foi calculada na seção 5.4.2, sabe-se que resulta nas matrizes:

$\mathcal{A}_z$	a_1	a_3
	0	0
	0	1
	1	1

$\mathcal{A}_o$	a_1	a_3
	1	0

6. Após a obtenção das matrizes é efetuada a interseção entre: $zero$ e A_z;um e A_o, resultando nas matrizes O_z e O_o, sobre as quais é realizada a união

O_z	a_1	a_3
	0	0
	1	0

O_o	a_1	a_3
	1	0

$O_z \cup O_o$	a_1	a_3
	0	0
	1	0
	1	1

7. A matriz $O_z \cup O_o$ é então a matriz de observabilidade para o evento b_4, na saída 1 do sistema.

Capítulo 6

Resultados

Após a apresentação da modelagem de Arquiteturas Reconfiguráveis através das Chu nets, e de descrever os algoritmos que utilizam essa modelagem, este capítulo apresenta os realizados para a validação dos algoritmos de composição, controlabilidade e observabilidade. Os testes aqui apresentados têm caráter de validação, e não de avaliação de desempenho.

Um total de 2 aplicações são apresentadas neste capítulo. Elas estão divididas em pequenos relatórios que seguem um mesmo *template*, que juntamente com as aplicações está apresentado na seção 6.1.

6.1 Validação

Nesta seção são mostradas 2 modelos de arquiteturas reconfiguráveis, no intuito de validar os algoritmos de composição, controlabilidade e observabilidade, que foram mostrados no capítulo 5. A primeira modelagem é de um circuito meio-somador e segunda modelagem é de um decodificador 2 *to* 4. Antes de iniciar as validações, será demonstrado o *template* utilizado.

Validação x

Função: fonte.chu

Descrição: Breve descrição sobre a funcionalidade da aplicação configurada na Arquitetura Reconfigurável.

Informações gerais: Informações gerais sobre o sistema, como quantidade de variáveis, funções utilizadas, e Arquitetura Reconfigurável.

Tabela Descritiva: Tabela que contém a descrição da Chu net, como no modelo abaixo:

Espaço de Chu	Função	Eventos de entrada	Eventos de saída
Identifi cador do Espaço de chu	Função computada	Lista de eventos de entrada	Lista de eventos de saída

Tabela 6.1: Tabela descritiva da Chu net X.

Arquivo chu: Arquivo chu onde está modelada a Chu net correspondente a aplicação.

Composição matemática: Demonstração da operação de Composição sobre a Chu net através de funções lógicas, conjuntos e BDDS.

Resultado do Compose: Arquivo resultante da execução do Compose.

Controlabilidade matemática: Demonstração da operação de Composição sobre a Chu net através de funções lógicas e conjuntos.

Resultado do Controllability: Arquivo resultante da execução do Controllability.

Observabilidade matemática: Demonstração da operação de Observabilidade sobre a Chu net através de funções lógicas e conjuntos.

Resultado do Observability: Arquivo resultante da execução do Observability.

Validação 1

Função: meiosomador.chu

Descrição: O meio somador é um circuito combinacional destinado a somar dois bits. O circuito meio somador possui duas entradas, os bits a serem somados, e duas saídas, a soma dos dois bits e o “vai 1”.

Informações gerais: A Arquitetura Reconfiguravel que implementa o meio somador está demonstrada na figura 6.1, com 2 variáveis de entrada (a_0, a_2), 2 saídas $(beta(1), beta(2))$, 6 variáveis no sistema $(a_0, a_1, a_2, a_3, a_4 e a_5)$ e 6 Células Lógicas Programáveis.

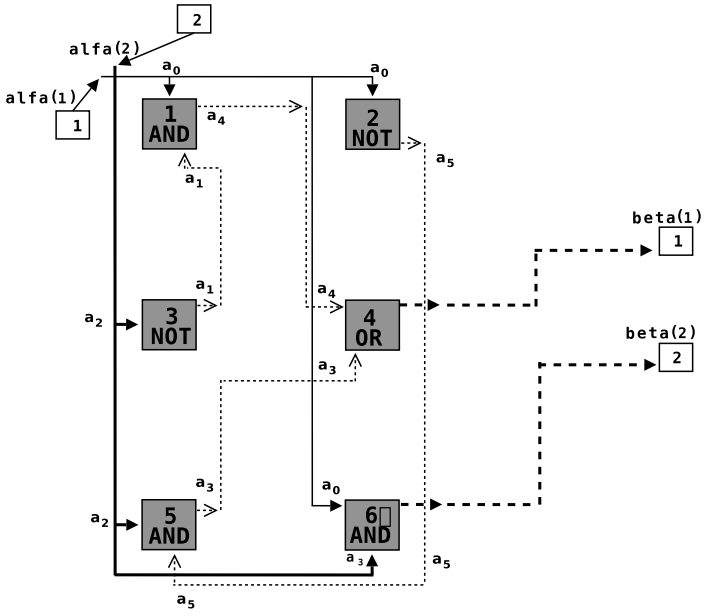

Figura 6.1: Arquitetura reconfigurável - meio somador

Tabela Descritiva: Tabela que contém a descrição da Chu net, como no modelo abaixo:

Espaço de Chu	Função	Eventos de entrada	Eventos de saída
1	AND	a_0, a_1	a_4
2	NOT	a_0	a_5
3	NOT	a_2	a_1
4	OR	a_3, a_4	0
5	AND	a_5, a_2	a_3
6	AND	a_0, a_2	0

Tabela 6.2: Tabela descritiva da Chu net meio somador.

Arquivo chu: Arquivo chu onde está modelada a Chu net.

Composição matemática: Segundo a definição 4.5, a composição dos Espaços de Chu, sejam dadas as seguintes fórmulas proposicionais dos Espaços de Chu representados na tabela 6.1:
$\varphi(\mathcal{A}_1) = a_0 a_1$
$\varphi(\mathcal{A}_2) = \bar{a_0}$
$\varphi(\mathcal{A}_3) = \bar{a_2}$
$\varphi(\mathcal{A}_4) = a_3 \vee a_4$
$\varphi(\mathcal{A}_5) = a_5 b_2$
$\varphi(\mathcal{A}_6) = a_0 b_2$
E os Espaços de Chu indentidade para os eventos de saída:
$(\mathcal{C}_{a_0}, \{a_0\}, R(x_1, a_0) = 1)$
$(\mathcal{C}_{a_2}, \{a_2\}, R(x_1, a_2) = 1)$
Podemos efetuar a composição para a 1^a saída da Chu net:
$\langle \mathcal{A}_5, \mathcal{A}_1 \rangle \gg \mathcal{A}_4$, considerando que $\tau_{\mathcal{A}_4}(1) = a_3$ e $\tau_{\mathcal{A}_4}(2) = a_4$, tem-se que $\varphi(\mathcal{A}_4)[(\varphi(\mathcal{A}_5, a_3), (\varphi(\mathcal{A}_1, a_4)]$ resulta na fórmula

$$\varphi(\mathcal{A}_4) = a_5 a_2 \vee a_0 a_1$$

Como em $\varphi(\mathcal{A}_4)$, ainda possui variáveis que não pertencem ao conjunto de eventos de entrada, compõem-se: $\langle \mathcal{A}_2, \mathcal{C}_{a_2}, \mathcal{C}_{a_0}, \mathcal{A}_3, \rangle \gg \mathcal{A}_4$, considerando que $\tau_{\mathcal{A}_4}(1) = a_5$, $\tau_{\mathcal{A}_4}(2) = a_2$, $\tau_{\mathcal{A}_4}(3) = a_0$ e $\tau_{\mathcal{A}_4}(4) = a_1$, tem-se que $\varphi(\mathcal{A}_4)[(\varphi(\mathcal{A}_2, a_5), (\varphi(\mathcal{C}_{a_2}, a_2), , (\varphi(\mathcal{C}_{a_0}, a_0), (\varphi(\mathcal{A}_4, a_1)]$ resulta na fórmula:

$$\varphi(\mathcal{A}_4) = \bar{a_0} a_2 \vee a_0 \bar{a_2}$$

A fórmula proposicional resultante da composição de $\varphi(\mathcal{A}_4)$ pode ser representada pelo BDD mostrado na figura 6.2.

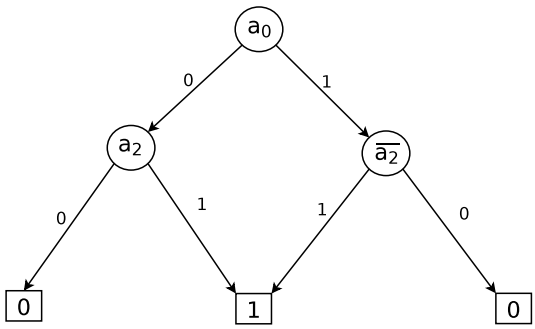

Figura 6.2: BDD resultante da composição da primeira saída do circuito meio somador

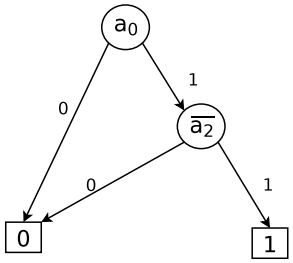

Figura 6.3: BDD resultante da segunda saída do circuito meio somador

Na figura 6.3, está desenhado o BDD que representa a função de $\varphi(\mathcal{A}_6)$, que não precisou ser composta, pois seus eventos já são eventos de saída.

Resultado do Compose: Arquivo resultante da execução do Compose para esta Chu net.

Controlabilidade matemática: Segundo a definição na seção 5.4.2, para detectar-se a controlabiliade dos eventos em uma Chu net, seja dada a Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, onde:
$\mathbf{C} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$
$F = \{(\mathcal{A}_1, a_4), (\mathcal{A}_2, a_2), (\mathcal{A}_3, a_1), (\mathcal{A}_4, \uparrow), (\mathcal{A}_5, a_3), (\mathcal{A}_6, \uparrow)\}$
$\alpha = \{(1, a_0), (2, a_2)\}$
$\beta = \{(1, \mathcal{A}_4), (2, \mathcal{A}_6)\}$

Sabendo ainda que:
$A := \bigcup_{\mathcal{A} \in C} A$
$A := \{a_0, a_1, a_2, a_3, a_4, a_5\}$

Tem-se $\forall e \in A \wedge e \in imag(\alpha) \bullet control(e)$:

- Control(a_0):

$\mathcal{A}_z(a_0)$	a_0	a_2
	0	2

$\mathcal{A}_o(a_0)$	a_1	a_3
	1	2

- Control(a_2):

$\mathcal{A}_z(a_2)$	a_0	a_2
	2	0

$\mathcal{A}_o(a_2)$	a_1	a_3
	2	1

$\forall e \notin A \wedge e \in imag(\alpha) \bullet control(e)$:

- Control(a_1):

  O Espaço de chu, cuja a saída é a_1 é definido como única saída da Chu net modificada, $\mathcal{C}_e$: $\mathcal{C}_e(C, F < +(\mathcal{A}_3, \uparrow), \alpha, \beta := \{(1, \mathcal{A}_3)\})$. Executa-se a composição da Chu net modificada:

  $\mathcal{M}_e := Compose(\mathcal{C}_e)\ \mathcal{M}_e := \bar{a_2}$

  Cuja a tabela verdade é:

$\mathcal{M}_e$	a_0	a_2	$\varphi(\mathcal{M}_e)$
	0	0	1
	0	1	0
	1	0	1
	1	1	0

  Divide-se a matriz em duas $\mathcal{A}_z$ e $\mathcal{A}_o$, que contém, respectivamente, as linhas cuja $\varphi(\mathcal{M}_e) = 0$ e $\varphi(\mathcal{M}_e) = 1$

$\mathcal{A}_z(a_1)$	a_0	a_2
	2	1

$\mathcal{A}_o(a_1)$	a_1	a_3
	2	0

- Control(a_3):

  O Espaço de chu, cuja a saída é a_3 é definido como única saída da Chu net modificada, $\mathcal{C}_e$: $\mathcal{C}_e(C, F < +(\mathcal{A}_5, \uparrow), \alpha, \beta := \{(1, \mathcal{A}_5)\})$. Executa-se a composição da Chu net modificada:

  $\mathcal{M}_e := Compose(\mathcal{C}_e)\ \mathcal{M}_e := \bar{a_0}a_2$

  Cuja a tabela verdade é:

$\mathcal{M}_e$	a_0	a_2	$\varphi(\mathcal{M}_e)$
	0	0	0
	0	1	1
	1	0	0
	1	1	0

  Divide-se a matriz em duas $\mathcal{A}_z$ e $\mathcal{A}_o$, que contém, respectivamente, as linhas cuja $\varphi(\mathcal{M}_e) = 0$ e $\varphi(\mathcal{M}_e) = 1$

$\mathcal{A}_z(a_3)$	a_0	a_2
	0	0
	1	0
	1	1

$\mathcal{A}_o(a_3)$	a_1	a_3
	0	1

- Control(a_4):

  O Espaço de chu, cuja a saída é a_4 é definido como única saída da Chu net modificada, $\mathcal{C}_e$: $\mathcal{C}_e(C, F < +(\mathcal{A}_1, \uparrow), \alpha, \beta := \{(1, \mathcal{A}_1)\})$. Executa-se a composição da Chu net modificada:

  $\mathcal{M}_e := Compose(\mathcal{C}_e)\ \mathcal{M}_e := a_0\bar{a_2}$

  Cuja a tabela verdade é:

$\mathcal{M}_e$	a_0	a_2	$\varphi(\mathcal{M}_e)$
	0	0	0
	0	1	0
	1	0	1
	1	1	0

  Divide-se a matriz em duas $\mathcal{A}_z$ e $\mathcal{A}_o$, que contém, respectivamente, as linhas cuja $\varphi(\mathcal{M}_e) = 0$ e $\varphi(\mathcal{M}_e) = 1$

$\mathcal{A}_z(a_4)$	a_0	a_2
	0	0
	0	1
	1	1

$\mathcal{A}_o(a_4)$	a_1	a_3
	1	0

- Control(a_5):

  O Espaço de chu, cuja a saída é a_5 é definido como única saída da Chu net modificada, $\mathcal{C}_e$: $\mathcal{C}_e(C, F < +(\mathcal{A}_2, \uparrow), \alpha, \beta := \{(1, \mathcal{A}_2)\})$. Executa-se a composição da Chu net modificada:

  $\mathcal{M}_e := Compose(\mathcal{C}_e)\ \mathcal{M}_e := \bar{a_0}$

  Cuja a tabela verdade é:

$\mathcal{M}_e$	a_0	a_2	$\varphi(\mathcal{M}_e)$
	0	0	1
	0	1	1
	1	0	0
	1	1	0

  Divide-se a matriz em duas $\mathcal{A}_z$ e $\mathcal{A}_o$, que contém, respectivamente, as linhas cuja $\varphi(\mathcal{M}_e) = 0$ e $\varphi(\mathcal{M}_e) = 1$

$\mathcal{A}_z(a_5)$	a_0	a_2
	1	0
	1	1

$\mathcal{A}_o(a_5)$	a_1	a_3
	0	0
	0	1

Resultado do Controllability: Arquivo resultante da execução do Controllability para esta Chu net.

Observabilidade matemática: Segundo a definição na seção 5.4.2, para detectar-se a controlabiliade dos eventos em uma Chu net, seja dada a Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, onde:
$\mathbf{C} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$
$F = \{(\mathcal{A}_1, a_4), (\mathcal{A}_2, a_2), (\mathcal{A}_3, a_1), (\mathcal{A}_4, \uparrow), (\mathcal{A}_5, a_3), (\mathcal{A}_6, \uparrow)\}$
$\alpha = \{(1, a_0), (2, a_2)\}$
$\beta = \{(1, \mathcal{A}_4), (2, \mathcal{A}_6)\}$
Sabendo ainda que:
$A := \bigcup_{\mathcal{A} \in C} A$
$A := \{a_0, a_1, a_2, a_3, a_4, a_5\}$

Tem-se $\forall e \in A \wedge e \in imag(\alpha) \bullet control(e)$:

$\mathcal{M}_{\mathcal{C}p} := Compose(C)$,
$\mathbb{M}_{\mathbb{C}p}[1] := \bar{a_0}a_2 \vee a_0\bar{a_2}$,
$\mathbb{M}_{\mathbb{C}p}[2] := a_0 a_2$

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1 é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_0	a_2
	0	0
	1	1

$\mathcal{A}_{o1}$	a_1	a_3
	0	1
	1	0

As matrizes A_{z2} e A_{o2} que mostram a combinação de valores em que o resultado da função computada na saída 2 é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o2}$	a_1	a_3
	1	1

- observ(a_0)

  São geradas as matrizes de controlabilidade do evento a_0, que são:

$\mathcal{A}_z(a_0)$	a_0	a_2
	0	2

$\mathcal{A}_o(a_0)$	a_1	a_3
	1	2

  A observabilidade do evento a_0 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_0)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_0))$$

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_0)$	a_0	a_2
	0	0

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_0)$	a_0	a_2
	1	0

Observ1	a_0	a_2
	0	0
	1	0

A observabilidade do evento a_0 na saída 2 é dado pela fórmula:

$$(\mathcal{A}_{z2} \cap \mathcal{A}_z(a_0)) \cup (\mathcal{A}_{o2} \cap \mathcal{A}_o(a_0))$$

$\mathcal{A}_{z2} \cap \mathcal{A}_z(a_0)$	a_0	a_2
	0	0
	0	1

$\mathcal{A}_{o2} \cap \mathcal{A}_o(a_0)$	a_0	a_2
	1	1

Observ2	a_0	a_2
	0	0
	0	1
	1	1

- observ(a_2)

  São geradas as matrizes de controlabilidade do evento a_2, que são:

$\mathcal{A}_z(a_0)$	a_0	a_2
	2	0

$\mathcal{A}_o(a_0)$	a_1	a_3
	2	1

  A observabilidade do evento a_2 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_2)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_2))$$

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_2)$	a_0	a_2
	0	0

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_2)$	a_0	a_2
	0	1

Observ1	a_0	a_2
	0	0
	0	1

  A observabilidade do evento a_2 na saída 2 é dado pela fórmula:

$$(\mathcal{A}_{z2} \cap \mathcal{A}_z(a_2)) \cup (\mathcal{A}_{o2} \cap \mathcal{A}_o(a_2))$$

$\mathcal{A}_{z2} \cap \mathcal{A}_z(a_2)$	a_0	a_2
	0	0
	1	0

$\mathcal{A}_{o2} \cap \mathcal{A}_o(a_2)$	a_0	a_2
	1	1

Observ2	a_0	a_2
	0	0
	1	0
	1	1

  Tem-se $\forall e \in A \wedge e \notin imag(\alpha) \bullet control(e)$:

- observ(a_1)

  Para detectar a observabilidade de a_1 é necessário criar uma Chu net modificada, onde a_1 fará parte do conjunto de entradas: $\mathcal{C}_e := (C, F, \alpha \cup \{(|\alpha| + 1, e)\}, \beta\}$, dessa forma:

$\mathcal{M}_{\mathcal{C}e} := Compose(\mathcal{C}_e)$,
$\mathbb{M}_{\mathbb{C}e}[1] := \bar{a_0}a_2 \vee a_0a_1$,
$\mathbb{M}_{\mathbb{C}e}[2] := a_0a_2$

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_0	a_2	a_1
	0	0	0
	0	0	1
	1	0	0
	1	1	0
	1	1	1

$\mathcal{A}_{o1}$	a_1	a_3	a_1
	0	1	0
	0	1	1
	1	0	1

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o2}$	a_1	a_3
	1	1

São geradas as matrizes de controlabilidade do evento a_1, que são:

$\mathcal{A}_z(a_1)$	a_0	a_2
	0	1
	1	1

$\mathcal{A}_o(a_1)$	a_1	a_2
	0	0
	1	0

A observabilidade do evento a_1 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_1)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_1))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_1)$	a_0	a_2
	1	1

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_1)$	a_0	a_2
	1	0

Observ1	a_0	a_2
	1	1
	1	0

A função computada na saída 2 não depende do evento a_1, logo não é possível observá-lo nesta saída.

- observ(a_3)

Para detectar a observabilidade de a_3 é necessário criar uma Chu net modificada, onde a_3 fará parte do conjunto de entradas: $\mathcal{C}_e := (C, F, \alpha \cup \{(|\alpha| + 1, a_3)\}, \beta\}$, dessa forma:

$\mathcal{M}_{\mathcal{C}e} := Compose(\mathcal{C}_e)$,
$\mathbb{M}_{\mathbb{C}e}[1] := a_3 \vee a_0\bar{a_2}$,
$\mathbb{M}_{\mathbb{C}e}[2] := a_0a_2$

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_0	a_2	a_3
	0	0	0
	0	1	0
	1	1	0

$\mathcal{A}_{o1}$	a_0	a_2	a_3
	0	0	1
	0	1	1
	1	0	0
	1	0	1
	1	1	1

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 2, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o2}$	a_0	a_2
	1	1

São geradas as matrizes de controlabilidade do evento a_3, que são:

$\mathcal{A}_z(a_1)$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_o(a_1)$	a_0	a_2
	1	1

A observabilidade do evento a_3 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_3)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_3))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_1)$	a_0	a_2
	0	0
	0	1

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_1)$	a_0	a_2
	1	1

Observ1	a_0	a_2
	0	0
	0	1
	1	1

A função computada na saída 2 não depende do evento a_3, logo não é possível observá-lo nesta saída.

- observ(a_4)

Para detectar a observabilidade de a_4 é necessário criar uma Chu net modificada, onde a_4 fará parte do conjunto de entradas: $\mathcal{C}_e := (C, F, \alpha \cup \{(|\alpha|+1, a_4)\}, \beta\}$, dessa forma:

$\mathcal{M}_{\mathcal{C}e} := Compose(\mathcal{C}_e)$,
$\mathbb{M}_{\mathbb{C}e}[1] := \bar{a_0}a_2 \vee a_4$,
$\mathbb{M}_{\mathbb{C}e}[2] := a_0 a_2$

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_0	a_2	a_4
	0	0	0
	1	0	0
	1	1	0

$\mathcal{A}_{o1}$	a_0	a_2	a_4
	0	0	1
	0	1	0
	0	1	1
	1	0	1
	1	1	1

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 2, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o2}$	a_0	a_2
	1	1

São geradas as matrizes de controlabilidade do evento a_4, que são:

$\mathcal{A}_z(a_4)$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_o(a_4)$	a_0	a_2
	1	1

A observabilidade do evento a_4 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_4)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_4))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_4)$	a_0	a_2
	0	0
	1	0

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_4)$	a_0	a_2
	1	1

Observ1	a_0	a_2
	0	0
	1	0
	1	1

A função computada na saída 2 não depende do evento a_4, logo não é possível observá-lo nesta saída.

- observ(a_5)

  Para detectar a observabilidade de a_5 é necessário criar uma Chu net modificada, onde a_5 fará parte do conjunto de entradas: $\mathcal{C}_e := (C, F, \alpha \cup \{(|\alpha| + 1, a_5)\}, \beta\}$, dessa forma:

  $\mathcal{M}_{\mathcal{C}_e} := Compose(\mathcal{C}_e)$,
  $\mathbb{M}_{\mathbb{C}_e}[1] := a_5 a_2 \vee a_0 \bar{a_2}$,
  $\mathbb{M}_{\mathbb{C}_e}[2] := a_0 a_2$

  As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_0	a_2	a_4
	0	0	0
	0	0	1
	0	1	0
	1	1	0

$\mathcal{A}_{o1}$	a_0	a_2	a_4
	0	1	1
	1	1	0
	1	0	1
	1	1	1

  As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 2, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_0	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o2}$	a_0	a_2
	1	1

  São geradas as matrizes de controlabilidade do evento a_5, que são:

$\mathcal{A}_z(a_5)$	a_0	a_2
	1	0
	1	1

$\mathcal{A}_o(a_5)$	a_0	a_2
	0	0
	0	1

  A observabilidade do evento a_5 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_5)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_5))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_5)$	a_0	a_2
	1	1

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_5)$	a_0	a_2
	0	1

Observ1	a_0	a_2
	0	1
	1	1

A função computada na saída 2 não depende do evento a_5, logo não é possível observá-lo nesta saída.

Resultado do Observability: Arquivo resultante da execução do Observability para esta Chu net.

Validação 2

Função: decod2to4.chu

Descrição: Um decodificador é um circuito combinacional usado para ativar ou habilitar um (e somente um) dentre n componentes. No decodificador aqui representado, tem-se 2 entradas e 4 saídas.

Informações gerais: A Arquitetura Reconfiguravel que implementa o decodificador $2to4$ está demonstrada na figura 6.4, com 2 variáveis de entrada, 4 saídas, 6 variáveis no sistema e 6 Células Lógicas Programáveis.

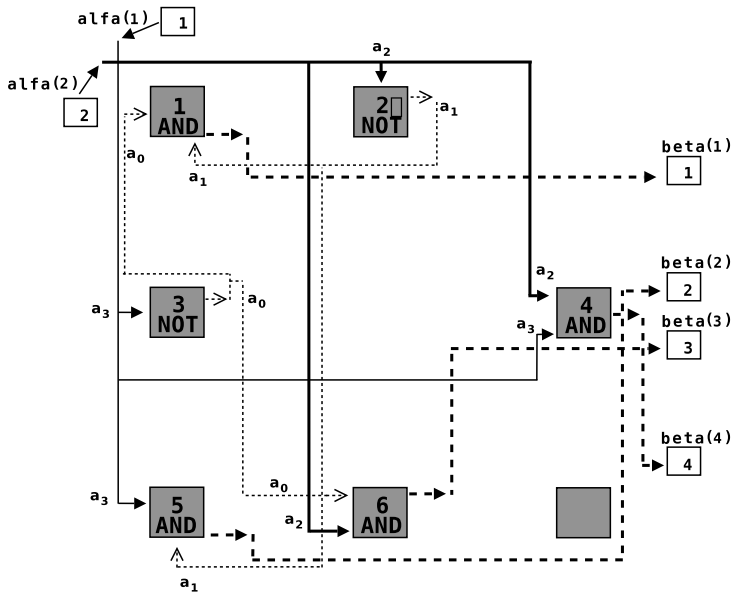

Figura 6.4: Arquitetura reconfigurável - decodificardor $2to4$

Tabela Descritiva: Tabela que contém a descrição da Chu net, como no modelo abaixo:

Espaço de Chu	Função	Eventos de entrada	Eventos de saída
1	AND	a_0, a_1	0
2	NOT	a_2	a_1
3	NOT	a_3	a_0
4	AND	a_2, a_3	0
5	AND	a_3, a_1	0
6	AND	a_2, a_0	0

Tabela 6.3: Tabela descritiva da Chu net decodificador $2to4$.

Arquivo chu: Arquivo chu onde está modelada a Chu net.

Composição matemática: Segundo a definição 4.5, a composição dos Espaços de Chu, sejam dadas as seguintes fórmulas proposicionais dos Espaços de Chu representados na tabela 6.1:
$\varphi(\mathcal{A}_1) = a_0 a_1$
$\varphi(\mathcal{A}_2) = \bar{a_2}$
$\varphi(\mathcal{A}_3) = \bar{a_3}$
$\varphi(\mathcal{A}_4) = a_2 a_3$
$\varphi(\mathcal{A}_5) = a_3 a_1$
$\varphi(\mathcal{A}_6) = a_2 a_0$
E os Espaços de Chu indentidade para os eventos de entrada:
$(\mathcal{C}_{a_3}, \{a_3\}, R(x_1, a_3) = 1)$
$(\mathcal{C}_{a_2}, \{a_2\}, R(x_1, a_2) = 1)$
Podemos efetuar a composição para a 1ª saída da Chu net:
$\langle \mathcal{A}_3, \mathcal{A}_2 \rangle \gg \mathcal{A}_1$, considerando que $\tau_{\mathcal{A}_1}(1) = a_0$ e $\tau_{\mathcal{A}_1}(2) = a_1$, tem-se que $\varphi(\mathcal{A}_1)[(\varphi(\mathcal{A}_3, a_0)), (\varphi(\mathcal{A}_2, a_1))]$ resulta na fórmula

$$\varphi(\mathcal{A}_1) = \bar{a_3}\bar{a_2}$$

Como em $\varphi(\mathcal{A}_1)$, todas as variáveis pertencem aos eventos de entrada, não há outras composições. A fórmula proposicional resultante da composição de $\varphi(\mathcal{A}_1)$ pode ser representada pelo BDD mostrado na figura 6.5.

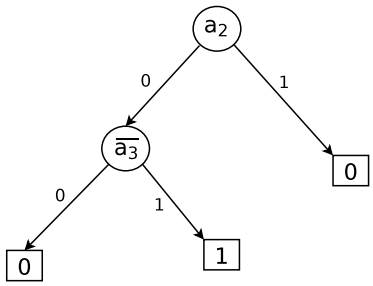

Figura 6.5: BDD resultante da composição da primeira saída do circuito decodificador $2to4$

Para a 2ª saída da Chu net, podemos efetuar a seguinte composição:
$\langle \mathcal{C}_{a_3}, \mathcal{A}_2 \rangle \gg \mathcal{A}_5$, considerando que $\tau_{\mathcal{A}_5}(1) = a_3$ e $\tau_{\mathcal{A}_5}(2) = a_1$, tem-se que $\varphi(\mathcal{A}_5)[(\varphi(\mathcal{C}_{a_3}, a_3)), (\varphi(\mathcal{A}_2, a_1))]$ resulta na fórmula

$$\varphi(\mathcal{A}_5) = a_3\bar{a_2}$$

Como em $\varphi(\mathcal{A}_5)$, todas as variáveis pertencem aos eventos de entrada, não há outras composições. A fórmula proposicional resultante da composição de $\varphi(\mathcal{A}_5)$ pode ser representada pelo BDD mostrado na figura 6.6.

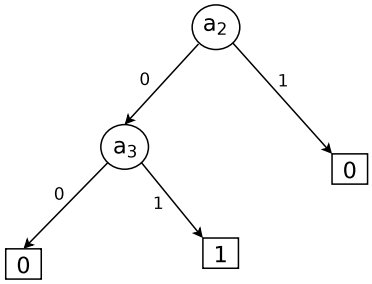

Figura 6.6: BDD resultante da composição da segunda saída do circuito decodificador $2to4$

Para a 3ª saída da Chu net, podemos efetuar a seguinte composição:
$\langle \mathcal{C}_{a_2}, \mathcal{A}_3 \rangle \gg \mathcal{A}_6$, considerando que $\tau_{\mathcal{A}_6}(1) = a_2$ e $\tau_{\mathcal{A}_6}(2) = a_0$, tem-se que $\varphi(\mathcal{A}_6)[(\varphi(\mathcal{C}_{a_2}, a_2), (\varphi(\mathcal{A}_3, a_0)]$ resulta na fórmula

$$\varphi(\mathcal{A}_6) = a_2\bar{a_3}$$

Como em $\varphi(\mathcal{A}_6)$, todas as variáveis pertencem aos eventos de entrada, não há outras composições. A fórmula proposicional resultante da composição de $\varphi(\mathcal{A}_6)$ pode ser representada pelo BDD mostrado na figura 6.7.

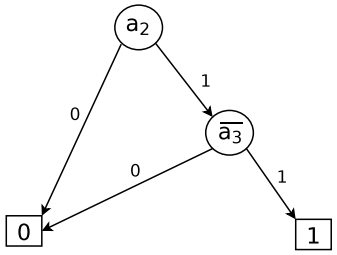

Figura 6.7: BDD resultante da composição da terceira saída do circuito decodificador $2to4$

Na figura 6.8, está desenhado o BDD que representa a função de $\varphi(\mathcal{A}_4)$, que não precisou ser composta, pois seus eventos já são eventos de saída.

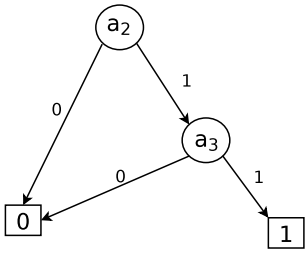

Figura 6.8: BDD resultante da segunda saída do circuito meio somador

Resultado do Compose: Arquivo resultante da execução do Compose para esta Chu net.

Controlabilidade matemática: Segundo a definição na seção 5.4.2, para detectar-se a controlabiliade dos eventos em uma Chu net, seja dada a Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, onde:
$\mathbf{C} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$
$F = \{(\mathcal{A}_1, \uparrow), (\mathcal{A}_2, a_1), (\mathcal{A}_3, a_0), (\mathcal{A}_4, \uparrow), (\mathcal{A}_5, \uparrow), (\mathcal{A}_6, \uparrow)\}$
$\alpha = \{(1, a_3), (2, a_2)\}$
$\beta = \{(1, \mathcal{A}_1), (2, \mathcal{A}_5), (3, \mathcal{A}_6), (4, \mathcal{A}_4)\}$

Sabendo ainda que:
$A := \bigcup_{\mathcal{A} \in C} A$
$A := \{a_0, a_1, a_2, a_3\}$

Tem-se $\forall e \in A \wedge e \in imag(\alpha) \bullet control(e)$:

- Control(a_3):

$\mathcal{A}_z(a_3)$	a_3	a_2
	0	2

$\mathcal{A}_o(a_3)$	a_3	a_2
	1	2

- Control(a_2):

$\mathcal{A}_z(a_2)$	a_3	a_2
	2	0

$\mathcal{A}_o(a_2)$	a_3	a_2
	2	1

$\forall e \notin A \wedge e \in imag(\alpha) \bullet control(e)$:

- Control(a_1):

  O Espaço de chu, cuja a saída é a_1 é definido como única saída da Chu net modificada, $\mathcal{C}_e$: $\mathcal{C}_e(C, F < +(\mathcal{A}_2, \uparrow), \alpha, \beta := \{(1, \mathcal{A}_2)\})$. Executa-se a composição da Chu net modificada:

  $\mathcal{M}_e := Compose(\mathcal{C}_e)\ \mathcal{M}_e := \bar{a_2}$

  Cuja a tabela verdade é:

$\mathcal{M}_e$	a_3	a_2	$\varphi(\mathcal{M}_e)$
	0	0	1
	0	1	0
	1	0	1
	1	1	0

Divide-se a matriz em duas $\mathcal{A}_z$ e $\mathcal{A}_o$, que contém, respectivamente, as linhas cuja $\varphi(\mathcal{M}_e) = 0$ e $\varphi(\mathcal{M}_e) = 1$

$\mathcal{A}_z(a_1)$	a_3	a_2
	2	1

$\mathcal{A}_o(a_1)$	a_3	a_2
	2	0

- Control(a_0):

O Espaço de chu, cuja a saída é a_0 é definido como única saída da Chu net modificada, $\mathcal{C}_e$: $\mathcal{C}_e(C, F < +(\mathcal{A}_3, \uparrow), \alpha, \beta := \{(1, \mathcal{A}_3)\})$. Executa-se a composição da Chu net modificada:

$\mathcal{M}_e := Compose(\mathcal{C}_e)\ \mathcal{M}_e := \bar{a_3}$

Cuja a tabela verdade é:

$\mathcal{M}_e$	a_2	a_3	$\varphi(\mathcal{M}_e)$
	0	0	1
	0	1	1
	1	0	0
	1	1	0

Divide-se a matriz em duas $\mathcal{A}_z$ e $\mathcal{A}_o$, que contém, respectivamente, as linhas cuja $\varphi(\mathcal{M}_e) = 0$ e $\varphi(\mathcal{M}_e) = 1$

$\mathcal{A}_z(a_0)$	a_3	a_2
	1	2

$\mathcal{A}_o(a_0)$	a_3	a_2
	0	2

Resultado do Controllability: Arquivo resultante da execução do Controllability para esta Chu net.

Observabilidade matemática: Segundo a definição na seção 5.4.3, para detectar-se a controlabiliade dos eventos em uma Chu net, seja dada a Chu net $(\boldsymbol{C}, F, \alpha, \beta)$, onde:
$\mathbf{C} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$
$F = \{(\mathcal{A}_1, \uparrow), (\mathcal{A}_2, a_1), (\mathcal{A}_3, a_0), (\mathcal{A}_4, \uparrow), (\mathcal{A}_5, \uparrow), (\mathcal{A}_6, \uparrow)\}$
$\alpha = \{(1, a_3), (2, a_2)\}$
$\beta = \{(1, \mathcal{A}_1), (2, \mathcal{A}_5), (3, \mathcal{A}_6), (4, \mathcal{A}_4)\}$
Sabendo ainda que:
$A := \bigcup_{\mathcal{A} \in C} A$
$A := \{a_0, a_1, a_2, a_3\}$

Tem-se $\forall e \in A \wedge e \in imag(\alpha) \bullet control(e)$:

$\mathcal{M}_{\mathcal{C}p} := Compose(C)$,
$\mathbb{M}_{\mathbb{C}p}[1] := \bar{a_3}\bar{a_2}$,
$\mathbb{M}_{\mathbb{C}p}[2] := a_3\bar{a_2}$
$\mathbb{M}_{\mathbb{C}p}[3] := \bar{a_3}a_2$
$\mathbb{M}_{\mathbb{C}p}[4] := a_3a_2$

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1 é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_3	a_2
	0	1
	1	0
	1	1

$\mathcal{A}_{o1}$	a_3	a_2
	0	0

As matrizes A_{z2} e A_{o2} que mostram a combinação de valores em que o resultado da função computada na saída 2 é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_3	a_2
	0	0
	0	1
	1	1

$\mathcal{A}_{o2}$	a_3	a_2
	1	0

As matrizes A_{z3} e A_{o3} que mostram a combinação de valores em que o resultado da função computada na saída 3 é igual a zero e a um, respectivamente:

$\mathcal{A}_{z3}$	a_3	a_2
	0	0
	1	0
	1	1

$\mathcal{A}_{o3}$	a_3	a_2
	0	1

As matrizes A_{z4} e A_{o4} que mostram a combinação de valores em que o resultado da função computada na saída 4 é igual a zero e a um, respectivamente:

$\mathcal{A}_{z4}$	a_3	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o4}$	a_3	a_2
	1	1

- observ(a_3)

  São geradas as matrizes de controlabilidade do evento a_3, que são:

$\mathcal{A}_z(a_3)$	a_3	a_2
	0	2

$\mathcal{A}_o(a_3)$	a_3	a_2
	1	2

A observabilidade do evento a_3 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_3)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_3))$$

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_3)$	a_3	a_2
	0	1

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_3)$	a_3	a_2

Observ1	a_3	a_2
	0	1

A observabilidade do evento a_3 na saída 2 é dado pela fórmula:

$$(\mathcal{A}_{z2} \cap \mathcal{A}_z(a_3)) \cup (\mathcal{A}_{o2} \cap \mathcal{A}_o(a_3))$$

$\mathcal{A}_{z2} \cap \mathcal{A}_z(a_3)$	a_3	a_2
	0	0
	0	1

$\mathcal{A}_{o2} \cap \mathcal{A}_o(a_3)$	a_3	a_2
	1	0

Observ2	a_3	a_2
	0	0
	0	1
	1	0

A observabilidade do evento a_3 na saída 3 é dado pela fórmula:

$$(\mathcal{A}_{z3} \cap \mathcal{A}_z(a_3)) \cup (\mathcal{A}_{o3} \cap \mathcal{A}_o(a_3))$$

$\mathcal{A}_{z3} \cap \mathcal{A}_z(a_3)$	a_3	a_2
	0	0

$\mathcal{A}_{o3} \cap \mathcal{A}_o(a_3)$	a_3	a_2

Observ3	a_3	a_2
	0	0

A observabilidade do evento a_3 na saída 4 é dado pela fórmula:

$$(\mathcal{A}_{z4} \cap \mathcal{A}_z(a_3)) \cup (\mathcal{A}_{o4} \cap \mathcal{A}_o(a_3))$$

$\mathcal{A}_{z4} \cap \mathcal{A}_z(a_3)$	a_3	a_2
	0	0
	0	1

$\mathcal{A}_{o4} \cap \mathcal{A}_o(a_3)$	a_3	a_2
	1	1

Observ4	a_3	a_2
	0	0
	0	1
	1	1

- observ(a_2)

  São geradas as matrizes de controlabilidade do evento a_2, que são:

$\mathcal{A}_z(a_2)$	a_3	a_2
	2	0

$\mathcal{A}_o(a_2)$	a_3	a_2
	2	1

  A observabilidade do evento a_2 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_2)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_2))$$

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_2)$	a_3	a_2
	1	0

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_2)$	a_3	a_2

Observ1	a_3	a_2
	1	0

A observabilidade do evento a_2 na saída 2 é dado pela fórmula:

$$(\mathcal{A}_{z2} \cap \mathcal{A}_z(a_2)) \cup (\mathcal{A}_{o2} \cap \mathcal{A}_o(a_2))$$

$\mathcal{A}_{z2} \cap \mathcal{A}_z(a_2)$	a_3	a_2
	0	0

$\mathcal{A}_{o2} \cap \mathcal{A}_o(a_2)$	a_3	a_2

Observ2	a_3	a_2
	0	0

A observabilidade do evento a_2 na saída 3 é dado pela fórmula:

$$(\mathcal{A}_{z3} \cap \mathcal{A}_z(a_2)) \cup (\mathcal{A}_{o3} \cap \mathcal{A}_o(a_2))$$

$\mathcal{A}_{z3} \cap \mathcal{A}_z(a_2)$	a_3	a_2
	0	0
	1	0

$\mathcal{A}_{o3} \cap \mathcal{A}_o(a_2)$	a_3	a_2
	0	1

Observ3	a_3	a_2
	0	0
	0	1
	1	0

A observabilidade do evento a_2 na saída 4 é dado pela fórmula:

$$(\mathcal{A}_{z4} \cap \mathcal{A}_z(a_2)) \cup (\mathcal{A}_{o4} \cap \mathcal{A}_o(a_2))$$

$\mathcal{A}_{z4} \cap \mathcal{A}_z(a_2)$	a_3	a_2
	0	0
	1	0

$\mathcal{A}_{o4} \cap \mathcal{A}_o(a_2)$	a_3	a_2
	1	1

Observ4	a_3	a_2
	0	0
	1	0
	1	1

Tem-se $\forall e \in A \wedge e \notin imag(\alpha) \bullet control(e)$:

- observ(a_1)

  Para detectar a observabilidade de a_1 é necessário criar uma Chu net modificada, onde a_1 fará parte do conjunto de entradas: $\mathcal{C}_e := (C, F, \alpha \cup \{(|\alpha| + 1, a_1)\}, \beta\}$, dessa forma:

  $\mathcal{M}_{\mathcal{C}e} := Compose(\mathcal{C}_e)$,
  $\mathbb{M}_{\mathbb{C}e}[1] := \bar{a_3} a_1$
  $\mathbb{M}_{\mathbb{C}e}[2] := a_3 a_1$
  $\mathbb{M}_{\mathbb{C}e}[3] := a_2 \bar{a_3}$
  $\mathbb{M}_{\mathbb{C}e}[4] := a_2 a_3$

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_3	a_2	a_1
	0	0	0
	0	1	0
	1	0	0
	1	0	1
	1	1	0
	1	1	1

$\mathcal{A}_{o1}$	a_3	a_2	a_1
	0	0	1
	0	1	1

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 2, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_3	a_2	a_1
	0	0	0
	0	0	1
	0	1	0
	0	1	1
	1	0	0
	1	1	0

$\mathcal{A}_{o2}$	a_3	a_2	a_1
	1	0	1
	1	1	1

As matrizes A_{z3} e A_{o3} que mostram a combinação de valores em que o resultado da função computada na saída 3, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z3}$	a_3	a_2
	0	0
	1	0
	1	1

$\mathcal{A}_{o3}$	a_3	a_2
	0	1

As matrizes A_{z4} e A_{o4} que mostram a combinação de valores em que o resultado da função computada na saída 4, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z4}$	a_3	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o4}$	a_3	a_2
	1	1

São geradas as matrizes de controlabilidade do evento a_1, que são:

$\mathcal{A}_z(a_1)$	a_3	a_2
	0	1
	1	1

$\mathcal{A}_o(a_1)$	a_3	a_2
	0	0
	1	0

A observabilidade do evento a_1 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_1)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_1))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_1)$	a_3	a_2
	0	1
	1	1

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_1)$	a_3	a_2
	0	0

Observ1	a_3	a_2
	0	0
	0	1
	1	1

A observabilidade do evento a_1 na saída 2 é dado pela fórmula:

$$(\mathcal{A}_{z2} \cap \mathcal{A}_z(a_1)) \cup (\mathcal{A}_{o2} \cap \mathcal{A}_o(a_1))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z2} \cap \mathcal{A}_z(a_1)$	a_3	a_2
	0	1
	1	1

$\mathcal{A}_{o2} \cap \mathcal{A}_o(a_1)$	a_3	a_2
	1	0

Observ2	a_3	a_2
	0	1
	1	0
	1	1

As funções computadas nas saídas 3 e 4 não dependem do evento a_1, logo não é possível observá-lo nestas saídas.

- observ(a_0)

  Para detectar a observabilidade de a_0 é necessário criar uma Chu net modificada, onde a_0 fará parte do conjunto de entradas: $\mathcal{C}_e := (C, F, \alpha \cup \{(|\alpha| + 1, a_0)\}, \beta\}$, dessa forma:

  $\mathcal{M}_{\mathcal{C}e} := Compose(\mathcal{C}_e)$,
  $\mathbb{M}_{\mathbb{C}e}[1] := a_0\bar{a_2}$
  $\mathbb{M}_{\mathbb{C}e}[2] := a_3\bar{a_2}$
  $\mathbb{M}_{\mathbb{C}e}[3] := a_2a_0$
  $\mathbb{M}_{\mathbb{C}e}[4] := a_2a_3$

  As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 1, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z1}$	a_0	a_2	a_1
	0	0	0
	0	1	0
	0	1	1
	1	0	0
	1	0	1
	1	1	0
	1	1	1

$\mathcal{A}_{o1}$	a_0	a_2	a_1
	0	0	1
	1	0	1

As matrizes A_{z1} e A_{o1} que mostram a combinação de valores em que o resultado da função computada na saída 2, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z2}$	a_3	a_2
	0	0
	0	1
	1	1

$\mathcal{A}_{o2}$	a_3	a_2
	1	0

As matrizes A_{z3} e A_{o3} que mostram a combinação de valores em que o resultado da função computada na saída 3, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z3}$	a_3	a_2	a_0
	0	0	0
	0	0	1
	0	1	0
	1	0	0
	1	0	1
	1	1	0

$\mathcal{A}_{o3}$	a_3	a_2	a_0
	0	1	1
	1	1	1

As matrizes A_{z4} e A_{o4} que mostram a combinação de valores em que o resultado da função computada na saída 4, da Chu net modificada, é igual a zero e a um, respectivamente:

$\mathcal{A}_{z4}$	a_3	a_2
	0	0
	0	1
	1	0

$\mathcal{A}_{o4}$	a_0	a_2
	1	1

São geradas as matrizes de controlabilidade do evento a_0, que são:

$\mathcal{A}_z(a_0)$	a_3	a_2
	1	0
	1	1

$\mathcal{A}_o(a_0)$	a_3	a_2
	0	0
	0	1

A observabilidade do evento a_0 na saída 1 é dado pela fórmula:

$$(\mathcal{A}_{z1} \cap \mathcal{A}_z(a_0)) \cup (\mathcal{A}_{o1} \cap \mathcal{A}_o(a_0))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z1} \cap \mathcal{A}_z(a_0)$	a_3	a_2
	1	0
	1	1

$\mathcal{A}_{o1} \cap \mathcal{A}_o(a_0)$	a_3	a_2
	0	0

Observ1	a_3	a_2
	0	0
	1	0
	1	1

A observabilidade do evento a_0 na saída 3 é dado pela fórmula:

$$(\mathcal{A}_{z3} \cap \mathcal{A}_z(a_0)) \cup (\mathcal{A}_{o3} \cap \mathcal{A}_o(a_0))$$

Onde na Interseção, a coluna correspondente ao evento que foi inserido na Chu net é retirada:

$\mathcal{A}_{z3} \cap \mathcal{A}_z(a_0)$	a_3	a_2
	1	0
	1	1

$\mathcal{A}_{o3} \cap \mathcal{A}_o(a_0)$	a_3	a_2
	0	1

Observ3	a_3	a_2
	0	1
	1	0
	1	1

As funções computadas nas saídas 2 e 4 não dependem do evento a_1, logo não é possível observá-lo nestas saídas.

Resultado do Observability: Arquivo resultante da execução do Observability para esta Chu net.

6.2 Benchmark ISCAS85

Os *benckmarks* são conjuntos de testes que podem ser utilizados para avaliar aspectos de performance de sistemas computacionais. Um *benckmark* é composto por vários arquivos no formato **bench**, que contém especificações de circuitos seguindo um certo padrão. O formato **bench** é bastante conhecido para teste de circuitos combinacionais e sequênciais.

Para realização dos testes dos algoritmos de composição, controlabilidade e observabilidade, utilizou-se o benchmark [21], composto por 11 arquivos que contém a descrição de circuitos combinacionais. Para utilizar esse benchmark, era necessário realizar a conversão de **bench** para **chu**, com este objetivo, foi desenvolvido o *parser* bench2chu. O

bench2chu recebe como entrada um arquivo com extensão .bench, realiza a conversão, e devolve um arquivo contendo o mesmo circuito descrito no .bench, só que no formato **chu**.

Por uma questão de “extensão” dos circuitos, os testes com o benchmark Iscas85 não foram validados, no entanto, é possível acessar os resultados gerados através dos algoritmos Compose, Controllability e Observability desses circuitos em [4].

Capítulo 7

Considerações Finais

Este trabalho, apresentou uma proposta para a modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu. Para isso, muitas pesquisas foram realizadas e os resultados foram concretizados na escrita desta dissertação. No primeiro capítulo, fez-se uma breve introdução, com o intuito de expor as tarefas a serem desenvolvidas nesta perquisa. Em seguida, nos segundo e terceiro capítulo, apresentou-se a fundamentação teórica, respectivamente, sobre Arquiteturas Reconfiguráveis e Espaços de Chu. No quarto capítulo, descreveu-se a modelagem de arquiteturas reconfiguráveis com Espaços de Chu, definindo a teoria sobre Chu nets e Multi-Espaços de Chu. No quinto capítulo, foi mostrada a metodologia de trabalho para o desenvolvimento do trabalho e para a implementação dos algoritmos de composição, controlabilidade e observabilidade, que utilizam a arquiteturas reconfiguráveis, modeladas através de Espaços de Chu. O capítulo sexto, mostram os resultados obtidos através dos algoritmos implementados, e finalmente, neste sétimo e último capítulo, apresenta-se algumas considerações finais sobre este trabalho.

Em sistemas reconfiguráveis é necessário, assim como em outros tipos de sistemas computacionais, mecanismos de verificação e validação, como forma de atestar se as características essenciais ao sistema estão sendo mantidas. Este trabalho, apresentou o conceito de modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu, que são um formalismo matemático, e permite que características e funções do sistema sejam verificadas e validadas.

Neste trabalho foi mostrada a metodologia sob a qual se desenvolveu a modelagem, bem como a implementação dos algoritmos Composição de CLBs, Controlabilidade e Observabilidade de Arquiteturas Reconfiguráveis, que se encontram modeladas através de Chu net, a generalização de Espaços de Chu que permite a modelagem de arquiteturas reconfiguráveis através de Espaços de Chu.

Como objetivo de validar os algoritmos que foram implementados, foram realizados

testes com circuitos combinacionais que integram o *benchmark* Iscas85

7.1 Principais Contribuições

O uso de Arquiteturas Reconfiguráveis surgiu como uma nova opção na computação em *hardware* que consegue equilibrar as características de flexibilidade e eficiência, efetivando-se com uma alternativa entre os processadores de propósito geral e os ASICs. Apresentou-se neste estudo, uma forma de modelar Arquiteturas Reconfiguráveis, através do uso da generalização de um formalismo matemático conhecido como Espaços de Chu. Assim, a realização deste estudo, possibilitou ao longo da pesquisa, o desenvolvimento de alguns artefatos, que contribuem com a temática nele abordada. Dentre essas contribuições destacam-se:

- Servir como fonte de consulta sobre Arquiteturas Reconfiguráveis, Espaços de Chu e modelagem de Arquiteturas Reconfiguráveis utilizando Espaços de Chu;
- Criação do formato persistente **chu**, que representa Arquiteturas Reconfiguráveis modeladas através de Espaços de Chu, e servindo dessa forma, como entrada para algoritmos que manipulem Arquiteturas Reconfiguráveis.
- Desenvolvimento e implementação dos algoritmos para a composição de CLBs, controlabilidade e observabilidade de Arquiteturas Reconfiguráveis.
- A implementação do *parser* **bench2chu**, que transforma arquivos com circuitos combinacionais descritos no formato bench, para o formato chu, permitindo dessa forma. Como o formato bench é bastante difundido, permite que usuários que não conhecem o formato chu, possam traduzir seus circuitos bench.

É importante ressaltar que tais contribuições servirão posteriormente como subsídios para realização de novas pesquisas, além de constituir com um documento que oferecerá suporte técnico para futuras aplicações desenvolvidas que se utilizam de modelos em Chu para Arquiteturas Reconfiguráveis.

7.2 Trabalhos Futuros

A pesquisa em modelagem de Arquiteturas Reconfiguráveis com Espaços de Chu está apenas começando, por isto, vários pontos ainda são insuficientes e necessitam da realização de novas pesquisas, nesta seção, apresenta-se os trabalhos futuros, com objetivo de dar-se continuidade à pesquisa, ou complementar a pesquisa reallizada até aqui:

- Desenvolver o módulo para ser adicionado ao algoritmo de composição, para converter o BDD resultado, em Chu net novamente.

- Uma vez que já existe os algoritmo de composição de CLBs, deseja-se fazer uma adapatação, para criar arquiteturas de granularidade grossa, criando funções mais complexas e redistribuindo as CLBs.

- A controlabilidade, juntamente com a observabilidade, é utilizada para detectar vetores de colagens em Arquiteturas Reconfiguráveis, a idéia é controlar e observar pontos do sistema afim de verificar a anormalidade de algum bloco. Dessa forma, é interessante a implementação de um algoritmo que forneça os vetores de colagens, gerados a partir dos algoritmos de controlabilidade e observabilidade. Alguns estudos já foram realizados, no que diz respeito à representação de CLBs "coladas" através de Espaços de Chu, inclusive, foi desenvolvido e implementado um algoritmo que insere, aleatoriamente, falhas do tipo colagem em arquivos ".chu".

Referências Bibliográficas

[1] A. M. S. Adário. *Arquiteturas Reconfiguráveis*. CPGCC-UFGRS, Porto Alegre, Agosto 1997.

[2] Altera. Flex 10k: Embedded programable logic device family. Technical report, Altera Corporation, Março 2001.

[3] A. M. Amory. Integração e avaliação de técnicas de teste baseado em software no fluxo de projeto de socs. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2003.

[4] C. Araújo. Modelagem de arquiteturas reconfiguráveis com espaços de chu. http://www.ppgsc.ufrn.br/ camila/, Julho 2006.

[5] Peter M. Athanas and Harvey F. Silverman. Processor reconfiguration through instruction-set metamorphosis. *IEEE Computer*, 26(3):11–18, 1993.

[6] B. R. C. Bedregal, I. S. Silva, and R. H. N. Santiago. Chu spaces as a formal model for reconfigurable arhitectures, Maio 2005.

[7] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, 1986.

[8] K. Compton and S. Hauck. An introduction to reconfigurable computing. *IEEE Computer*, Abril 2000.

[9] Atmel Corporation. http://www.atmel.com/.

[10] A. Dewey. Analysis and design of digital systems with vhdl, 1997.

[11] J. F. Eastman and D. R. Wooten. A general purpose, expandable processor for real-time computer graphics. In *SIGGRAPH '74: Proceedings of the 1st annual conference on Computer graphics and interactive techniques*, pages 30–30, New York, NY, USA, 1974. ACM Press.

[12] M. G. O. Gericota. *Metodologias de testes para FPGAs integradas em sistemas reconfiguráveis.* PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2003.

[13] S. A. Guccione and M. Gonzalez. Classification and performance for reconfigurable architectures. In *15th International Workshop on Field Programmable Logic Application*, Oxford, 1995.

[14] V. Gupta. *Chu Spaces: A Model of Concurrency*. PhD thesis, Standford University, 1994.

[15] R. Hartenstein. The microprocessor is no more general purpose. In *ISIS 97*, Texas, USA, Outubro 1997.

[16] J. R. Hauser and J. Wawrzynek. Garp: A mips processor with a reconfigurable coprocessor. In Kenneth L. Pocek and Jeffrey Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 12–21, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[17] W. Hinrichs, J. P. Wittenburg, H. Lieske, H. Kloos, M. Ohmacht, and P. Pirsch. A 1.3-gops parallel dsp for high-performance image-processing applications. *IEEE Journal of Solid-State Circuits*, 35:946–952, Julho 2000.

[18] M. C. Hsueh, T. K. Tsai, and R. K. Iyer. Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82, 1997.

[19] Advanced Micro Devices Inc. http://www.amd.com.

[20] Xilinx Inc. http://www.xilinx.com/.

[21] Iscas85 combinational benchmark circuits. http://www.visc.vt.edu/ mhsiao/iscas85.html, 1985.

[22] Wei-Cheng Lai and Kwang-Ting Cheng. Instruction-level DFT for testing processor and IP cores in system-on-a-chip. In *Design Automation Conference*, pages 59–64, 2001.

[23] F. G. Lima. Projeto com matrizes de células lógicas programáveis. Master's thesis, UFRGS, 1999.

[24] D. Marple and L. Cooke. An mpga compatible fpga architecture. In *ACM International Symposium on Field-Programmable Gate Arrays*, pages 39–44, Monterey, CA, Fevereiro 1992.

[25] S. Mourad and Y. Zorian. *Principles of Testing Electronic Systems*. John Wiley and Sons, 2000.

[26] I. Page. Microprocessors and Microsystems, Maio 1996.

[27] V. R. Pratt. Chu spaces:complementarity and uncertainty in rational mechanics, Julho 1994.

[28] V. R. Pratt. Chu spaces: Notes for the school on category theory and applications, Julho 1999.

[29] V. R. Pratt. http://chu.stanford.edu/live/, Setembro 2005.

[30] Vaughan Pratt. Modeling concurrency with partial orders. *Int. J. Parallel Program.*, 15(1):33–71, 1986.

[31] B. Radunovic and V. Milutinovic. A survey of reconfigurable computing architectures. In *32nd Hawaii International Conference on System Sciences*, 1999.

[32] L. Raffo, S. P. Sabatini, M. Mantelli, A. Gloria, and G. M. Bisio. Design of an (asip) architecture for low-level visual elaborations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5:145–153, Março 1997.

[33] R. A. L. Reis. *Concepção de Circuitos Integrados*. Sagra Luzzatto, 2000.

[34] C. E. Shannon. The synthesis of two-terminal switching circuits. *Bell System Tech. Journal*, 28, Janeiro 1949.

[35] M. J. S. Smith. *Application-Specific Integrated Circuits*. Addison-Wesley, 1997.

[36] Carnegie Mellon University. The bdd library. http://www.cs.cmu.edu/ modelcheck/bdd.html, Junho 2006.

[37] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Boucard. Programmable active memories: Reconfigurable systems come of age. *IEEE Transactions on VLSI Systems*, 4(1):56–69, 1996.

[38] E. et al. Waingolg. Baring it all to software: The raw machine. MIT Laboratory for Computer Science, Março 1997.

Apêndice A

Funções da Biblioteca BDD

bdd_init()

A função bdd_init cria e inicializa um novo gerenciador de BDD e vários gerenciadores podem existir em um mesmo momento. A função não recebe nenhum parâmetro de entrada e retorna o bdd_manager que foi criado e inicializado.

bdd_quit(manager)

Libera o gerenciador de BDD dado por `manager` e todas as informações armazenadas sobre ele. Recebe como parâmetro o gerenciador de BDD e não há retorno.

bdd_new_var_last(manager)

Cria uma nova variável no final da ordenação do BDD. A função bdd_new_var_last recebe como parâmetro de entrada o gerenciador de BDD que armazena informações sobre o BDD no qual a variável será inserida e retorna o BDD atualizado.

bdd_var_with_index (manager, i)

A função bdd_var_with_index recebe como parâmetros de entrada o gerenciador de BDD e um inteiro longo, `i`. Se a variável com o índice `i` foi criada, a função retorna o BDD para a variável, caso contrário, a variável não existe e a função retorna “null”

bdd_one(manager)

Retorna o BBD para a contante lógica VERDADEIRO.

bdd_zero(manager)

Retorna o BBD para a contante lógica FALSO.

bdd_and(manager, f, g)

Retorna o BDD para o resultado da função lógica AND entre os BDDs `f` e `g`.

bdd_or(manager, f, g)

Retorna o BDD para o resultado da função lógica OR entre os BDDs `f` e `g`.

bdd_not(manager, f)

Retorna o BDD para o resultado da função lógica NOT para o BDD `f`.

bdd_if(manager, f)

Retorna o BDD para a variável cujo o nó raiz é o BDD `f`

bdd_then(manager, f)

Retorna o BDD dado pelo "ramo" do BDD gerenciado por `manager`, determinado pelo valor VERDADEIRO da variável raiz `f`.

bdd_else(manager, f)

Retorna o BDD dado pelo "ramo" do BDD gerenciado por `manager`, determinado pelo valor FALSO da variável raiz `f`.

bdd_if_index(manager, f)

Retorna o índice da variável, cujo o rótulo do nó raiz do BDD é dado por `f`.

bdd_compose(manager, f, g, h)

A função bdd_compose retorna o BDD resultante da substituição de `h` para a variável `g` em `f`. Quando `h` não depende de `g`, a operação pode ser vista como uma composição de funções lógicas. Se `h` depende de `g`, corresponde a uma simples substituição na fórmula booleana.

bdd_depends_on(manager, f, g)

Retorna 1 se o BDD `f` depende da variável dada pelo BDD `g`, em caso contrário, retorna 0.

bdd_size(manager, f, negout)

Retorna o número de nós no BDD `f`. O parâmetro `negout` é um indicador que diz se variáveis que são negação de outras, devem ser considerados.

bdd_print_bdd(manager, f, naming_fn, terminal_id_fn, null, fp)

Imprime uma representação legível do BDD `f` no arquivo descrito por `fp`. Os outros parâmetros são padrões para a impressão de BDDs.

bdd_vars(manager)

Retorna o número de variáveis gerenciadas pelo gerenciador de BDDs.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo