UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RENAN ROBERTO ROGOSKI

FUNÇÕES DE ASSINATURA PARA CORRESPONDÊNCIA BOOLEANA

CHAPECÓ
2018

RENAN ROBERTO ROGOSKI

FUNÇÕES DE ASSINATURA PARA CORRESPONDÊNCIA BOOLEANA

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Emílio Wuerges

CHAPECÓ
2018

Rogoski, Renan Roberto

Funções de Assinatura para Correspondência Booleana / Renan Roberto Rogoski. – 2018.
38 f.: il.

Orientador: Prof. Dr. Emílio Wuerges
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal da Fronteira Sul, Curso de Ciência da Computação, Chapecó, SC, 2018.

1. Equivalência de Circuitos. 2. Formas Normais. 3. Bibliotecas de Células. 4. Satisfatibilidade Booleana. I. Wuerges, Emílio, orient. II. Universidade Federal da Fronteira Sul. III. Título.

E-mail: renanrogoski@gmail.com

RENAN ROBERTO ROGOSKI

FUNÇÕES DE ASSINATURA PARA CORRESPONDÊNCIA BOOLEANA

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Emílio Wuerges

Este trabalho de conclusão de curso foi defendido e aprovado pela banca em: 05/12/2018

BANCA EXAMINADORA:

Dr. Emílio Wuerges - UFFS

Me. Adriano Sanick Padilha - UFFS

Dr. Braulio Adriano de Mello - UFFS

RESUMO

Funções de Assinatura visam calcular representações compactas que caracterizam algumas propriedades de funções booleanas, chamadas de assinaturas. Cada função booleana possui uma assinatura única, mas uma assinatura pode estar relacionada a uma ou mais funções. Formas canônicas podem ser usadas para verificar se duas funções, são equivalentes, porém são muito custosas, assim o uso de funções de assinatura pode ajudar a reduzir o espaço de busca para a equivalência, desclassificando rapidamente *designs* que não são equivalentes.

Palavras-chave: Equivalência de Circuitos. Formas Normais. Bibliotecas de Células. Satisfatibilidade Booleana.

ABSTRACT

Signature Functions are intended to compute compact representations that characterize some properties of Boolean functions, called signatures. Each Boolean function has a unique signature, but a signature can be related to one or more functions. Canonical forms can be used to verify that two functions are equivalent, but they are very costly, so the use of signature functions can help shorten the search space for equivalence by quickly disqualifying designs that are not equivalent.

Keywords: Equivalence of Circuits, Normal Forms, Cell Library, Boolean Satisfiability.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Existem muitos *designs* de circuitos digitais disponíveis que as indústrias vem produzindo. Os projetos desses diferentes circuitos estão disponíveis em formato de bibliotecas de células, que são representações de diferentes circuitos que as indústrias são capazes de produzir. Através do acesso às bibliotecas de células é possível facilitar o reuso de *desings* de circuitos bem sucedidos.

As bibliotecas de células contêm o conjunto de primitivas lógicas que estão disponíveis num estilo de *design* desejado. Portanto, o processo de seleção deve permitir a busca nessa biblioteca na busca da melhor implementação possível, otimizando desempenho, consumo de energia e área [De Micheli, 1994, Garey and Johnson, 1979].

Encontrar correspondência na biblioteca de células, significa poder reconhecer se uma parte de um circuito lógico de múltiplos níveis pode ser implementada por uma determinada célula de biblioteca. A seleção significa escolher células apropriadas que otimizem a figura de interesse [Benini and De Micheli, 1997].

Conforme a lei de Moore, que previu que a capacidade de processamento dos chips aumentaria em 100% a cada 18 meses, os chips que estão sendo fabricados estão cada vez mais complexos, com uma quantidade de transistores cada vez maior, chegando a cerca de 20 bilhões de transistores em processadores de 10 nanômetros [Esmaeilzadeh et al., 2011]. Já o blog oficial da IBM divulgou que a IBM e seus parceiros de pesquisa, garantem terem desenvolvido um processo industrial de ponta para a construção de transistores de nanolâminas de silício que permitirão a produção de chips de 5 nanômetros (nm), chegando a quantidade de 30 bilhões de transistores num só chip [Bu, Huiming, 2017].

Novos projetos de circuitos são criados continuante para atender demandas nas diferentes áreas da tecnologia, mas a indústria não é capaz de atualizar constantemente seu métodos de fabricação de circuitos para implementar todos os novos projetos. Para facilitar o desenvolvimento de novos projetos eletrônicos e de automação, surge a necessidade de se encontrar *designs* de circuitos disponíveis nas bibliotecas de células, que venham a ter um comportamento equivalente aos novos projetos de circuitos. Este trabalho, através do estudo e implementação de Funções de Assinatura para correspondência booleana, visa auxiliar na seleção da biblioteca de células.

1.1 Objetivos

1.1.1 Geral

Este trabalho tem o intuito de pesquisar, estudar e testar Funções de Assinatura para reduzir o custo de verificação da equivalência de circuitos e auxiliar na seleção das saídas do design de circuito desejado em células de bibliotecas.

1.1.2 Específicos

- Implementar o cálculo de contagem da satisfação booleana das funções de saída dos casos de teste, ou seja, calcular a quantidade de valores 1 presentes nas saídas dos circuitos a partir de grupos de repetição de simulações, e implementar o cálculo da acurácia média das saídas dos casos de teste utilizado e ampliando as funcionalidades de uma ferramenta de verificação funcional de circuitos lógicos.
- Implementar o cálculo da cobertura de simulações das saídas dos casos de teste.
- Investigar se exite relação entre a acurácia média e a quantidade de saídas dos casos de teste.
- Investigar se exite relação entre a acurácia média e a cobertura de simulações das saídas dos casos de teste.
- Investigar se a contagem de valores 1 das saídas dos casos de teste é uma função de assinatura eficiente para encontrar a equivalência das saídas de diferentes tamanhos de circuitos.
- Verificar quanto tempo é necessário simular cada caso de teste para encontrar uma acurácia média satisfatória para as saídas.
- Gerar um arquivo de saída da ferramenta de simulação com os dados das simulações de cada circuito para analisar graficamente os resultados.

1.2 Justificativa

O aumento exponencial do número de transistores nos circuitos eletrônicos, confirmando a lei de Moore e chegando a cerca de 20 bilhões de transistores, pressiona a indústria de automação de projeto de eletrônicos cada vez mais a cada ano e aumenta cada vez mais a dificuldade de encontrar equivalência nos projetos de circuitos para adequar suas bibliotecas. Assim, técnicas que eram

adequadas a pouco tempo atrás se tornam difíceis de usar por causa do seu custo computacional que aumenta exponencialmente com o aumento da quantidade de transistores nos chips [Esmaeilzadeh et al., 2011].

As Funções de Assinatura podem ser uma alternativa para reduzir o custo computacional da verificação de equivalência de circuitos, permitindo que a indústria de projetos de circuitos se mantenha mais atualizada.

2 REVISÃO BIBLIOGRÁFICA

2.1 Circuitos Lógicos e Circuitos Digitais

Para representar circuitos digitais e o seu comportamento, foi criada uma abstração chamada de circuito Lógico. Circuitos lógicos podem ser denominados como conjuntos de expressões booleanas compostas por variáveis de entrada, operadores lógicos e funções de saída. Os valores presentes nas saídas dos circuitos lógicos dependem dos valores inseridos nas entradas e das operações lógicas que acontecem através dos operadores. Enquanto circuitos digitais trabalham com níveis de tensão elétrica, circuitos lógicos trabalham apenas com dois valores lógicos em seus componentes, Verdadeiro e Falso, sendo representados por 1 e 0, respectivamente. A figura 2.1 mostra um exemplo de circuito lógico.

2.2 Álgebra Booleana

Segundo Tocci et al. [2007], a álgebra booleana é uma ferramenta matemática relativamente simples e que nos permite descrever a relação entre a(s) saída(s) de um circuito lógico e sua(s) entrada(s) através de uma equação (expressão booleana). Como a Álgebra Booleana expressa as operações de circuitos lógicos de forma algébrica, ela apresenta-se como uma forma ideal de descrever as operações de um circuito lógico. Na Álgebra booleana existem apenas três operações básicas, sendo elas: OR (OU), AND (E) e NOT (NÃO). Abaixo é exibida a expressão em álgebra booleana representando o circuito da figura 2.1.

$S \leftarrow B \wedge ((\neg C \vee (\neg A \wedge B)))$

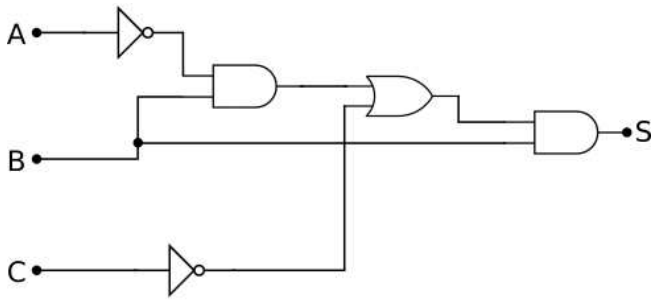

Figura 2.1 – Circuito lógico: Exemplo A

2.2.1 Constantes e Variáveis Booleanas

Na Álgebra booleana, constantes e variáveis possuem apenas dois valores permitidos, 0 ou 1. Uma variável booleana é uma quantidade que pode, em momentos diferentes, ser igual a 0 ou 1. Variáveis booleanas geralmente são utilizadas para representar o nível de tensão presente nas ligações ou terminais de entrada/saída de um circuito digital [Tocci et al., 2007].

2.2.2 Tabelas-Verdade

As operações básicas realizadas nos circuitos digitas, são representadas pela álgebra booleana através dos aperadores lógicos, or (+), and (.) e not (’). Cada operador lógico gera um resultado específico através das constantes ou variáveis aplicadas sobre ele. Para exibir os resultados de cada operação lógica, podem ser usadas as tabelas-verdade. Conforme Tocci et al. [2007], a tabela-verdade é uma maneira de descrever como a saída de um circuito lógico depende dos níveis lógicos presentes nas entradas dos circuitos. O circuito da figura 2.1, com entradas A, B, C e uma saída S, possui uma tabela-verdade, exibida abaixo como Tabela 2.1, que apresenta todas as combinações de valores lógicos para as entradas A, B, C e os valores presentes na saída S para cada combinação.

A	**B**	**C**	**S**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Tabela 2.1 – Tabela-verdade do exemplo da figura 2.1

2.2.3 Operadores lógicos

Nos circuitos lógicos existem 3 operadores básicos, que são utilizados para realizar as operações nos circuitos. Estes operadores segundo Tocci et al. [2007] são o *OR*, *AND* e *NOT*.

- ***(i) OR:*** O operador *OR* é semelhante à adição ordinária, onde ele realiza a soma entre os operandos binários, exceto quando ambos os operandos possuirem o valor 1, neste caso, a operação produz 1 + 1 = 1, ao invés de produzir 1 + 1 = 2. A operação *OR* possui característica comutativa, ou seja, a ordem dos operandos não altera o resultado da operação. Para a operação *OR*

resultar em um valor lógico alto nível (1), basta que pelo menos um valor da entrada possua o valor lógico (1).

A	**B**	**Saída**
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 2.2 – Tabela-verdade do operador *OR*

- ***(ii) AND:*** O operador *AND* é semelhante à multiplicação ordinária, onde ele realiza a multiplicação entre os operandos binários. A operação *AND* possui característica comutativa, ou seja, a ordem dos operandos não altera o resultado da operação. Para a operação *AND* resultar em um valor lógico de alto nível (1), todos os valores da entrada precisam possuir o valor 1, caso contrário, o resultado será 0.

A	**B**	**Saída**
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 2.3 – Tabela-verdade do operador *AND*

- ***(iii) NOT:*** O operador *NOT*, ao contrário das operações anteriores (*OR* e *AND*) realiza a operação sobre uma única entrada. Esta operação realiza a inversão do valor lógico de entrada, ou seja, se a porta receber o valor 1, resultará em um valor 0 como saída e vice-versa.

A	**Saída**
0	1
1	0

Tabela 2.4 – Tabela-verdade do operador *NOT*

3 REPRESENTAÇÃO CANÔNICA

3.1 Formas Normais

Uma representação canônica é uma forma de representar uma expressão booleana que possui a seguinte propiedade: se dois circuitos possuírem a mesma representação canônica, eles representam a mesma função lógica, e caso contrário, não [Benini and De Micheli, 1997].

A tabela-verdade de um circuito é um tipo de representação canônica e através dela é possível identificar que existe equivalência entre circuitos. Com o uso da tabela-verdade as expressões booleanas ou circuitos podem ser representados nas Formas Normais Conjuntiva (FNC) e Formas Normais Disjuntiva (FND), que são mais um exemplo de representação canônica. Apesar de que elas possuem fortes garantias teóricas para encontrar equivalência, na prática o custo de se gerar a tabela-verdade e as formas normais de um circuito é muito alto devido ao tamanho dos circuitos, isto é, $O(2^N)$ combinações sobre as entradas para se gerar a tabela-verdade.

Uma expressão booleana está na forma normal conjuntiva se $\theta = \bigwedge_{i=1,Ci}^{N}$, onde N > 1 e cada clausula Ci é uma disjunção de um ou mais literais, ou a expressão está na forma normal disjuntiva se $\theta = \bigvee_{i=1,Ci}^{N}$, onde N > 1 e cada cláusula Ci deve ser uma conjunção de um ou mais literais.

Para obter uma expressão na FNC, é necessário extrair informações da tabela-verdade da expressão booleana, levando em conta as linhas onde o resultado da função vale 0. Para cada uma dessas linhas devem ser escritas cláusulas com maxtermo associado às variáveis, isto é, realizar a soma (OR) das variáveis de entrada, negando cada variável que possuir nível lógico alto (1) na linha e coluna correspondente. E para finalizar a construção da expressão na forma normal conjuntiva (FNC) realizar o produto das cláusula de maxtermos. A tabela 3.1 mostra como são escritas as cláusulas de maxtermos.

A	**B**	**C**	**S**	**Max**
0	0	0	0	A+B+C
0	0	1	0	A+B+¬C
0	1	0	1	
0	1	1	1	
1	0	0	0	¬A+B+C
1	0	1	0	¬A+B+¬C
1	1	0	1	
1	1	1	0	¬A+¬B+¬C

Tabela 3.1 – Tabela-verdade com cláusulas da (FNC).

Através do produto de somas obtido da tabela anterior, segue a expressão na forma normal

conjuntiva (FNC): (A+B+C) . (A+B+¬C) . (¬A+B+C) . (¬A+B+¬C) . (¬A+¬B+¬C).

Na forma FND, assim como na FNC, é necessário extrair informações da tabela-verdade da expressão booleana. Porém levando em conta as linhas onde o resultado da função vale 1. Para cada uma dessas linhas devem ser escritas cláusulas com mintermo associado às variáveis, isto é, realizar o produto (AND) das variáveis de entrada, negando cada variável que possuir nível lógico baixo (0) na linha e coluna correspondente. E para finalizar a construção da expressão na FND, realizar a operação ou (OR) com as clausula de mintermos. A tabela 3.2 mostra como são escritas as cláusulas de mintermos.

A	**B**	**C**	**S**	**Min**
0	0	0	0	
0	0	1	0	
0	1	0	1	¬A.B¬C
0	1	1	1	¬A.B.C
1	0	0	0	
1	0	1	0	
1	1	0	1	A.B¬C
1	1	1	0	

Tabela 3.2 – Tabela-verdade com cláusulas da (FND).

Através da soma de produtos obtida da tabela anterior, segue a expressão na FND: (¬A.B.¬C) + (¬A.B.C) + (A.B.¬C).

3.2 Mapas de Karnaugh

O Mapa de Karnaugh é um método gráfico utilizado para realizar a simplificação de equações lógicas ou para converter uma tabela-verdade no seu circuito lógico correspondente, de um modo simples e ordenado. Embora um mapa de Karnaugh, possa ser usado em problemas que envolvam qualquer número de variáveis, sua aplicação prática limita-se ao uso com seis variáveis [Tocci et al., 2007].

Para construir o mapa de Karnaugh, serão utilizadas as cláusulas de mintermos das linhas da tabela-verdade de uma expressão booleana. Dada a tabela-verdade mostrada na tabela 3.2, é construída a tabela de adjacências, que possui dispostos por linha quatro mintermos, formando assim duas linhas na tabela.

Com a tabela de adjacências construída é possível iniciar as simplificações. A simplificação pode ser realizada nos mintermos adjacentes em sentido horizontal, vertical e extremidades, como se as laterais, topo e chão da tabela fossem unidas. No entanto, é possível realizar simplificações apenas

¬A.¬B.¬C	¬A.¬B.C	¬A.B.¬C	¬A.B.C
A.¬B.¬C	A.¬B.C	A.B.¬C	A.B.C

Tabela 3.3 – Tabela de adjacências para uma função de 3 variáveis.

nos mintermos adjacentes que diferem em apenas uma variável e seu valor verdade na função valer 1. Os nintermos que possuem valor verdade 0, são desconsiderados. Na tabela 3.4 estão dispostos os valores verdade da função correspondentes aos mintermos da tabela 3.3.

0	0	1	1
0	0	1	0

Tabela 3.4 – Tabela com os valores verdade da função.

Para realizar a simplificação, observa-se na tabela de adjacência os mintermos adjacentes e utiliza-se a parte da expressão que é idêntica, e desconsidera-se a variável que difere de um mintermo adjacente para o outro. Por exemplo, a simplificação do terceiro e quarto mintermo, gera a expressão ¬A.B e a simplificação do terceiro e sétimo mintermo gera a expressão B.¬C. Caso um nintermo que possui valor verdade 1, não possuir mintermos adjacentes, ele será incluído na expressão final sem simplificação. Finalmente para se obter a equação final é realizada a soma de produtos dos mintermos simplificados, e então é obtida a seguinte equação: ¬A.B + B.¬C.

4 FUNÇÕES DE ASSINATURA

A assinatura de uma função booleana é uma representação compacta que caracteriza algumas das propriedades da própria função. Cada função booleana possui uma assinatura única. Por outro lado, uma assinatura pode estar relacionada a duas ou mais funções. Este é um problema chamado de *aliasing*, o qual distingue as assinaturas das formas canônicas [Benini and De Micheli, 1997].

Para exemplificar os problemas de *aliasing*, notamos que os circuitos da figura 4.1 e da figura 4.2 são diferentes, mas possuem funções lógicas equivalentes e assinaturas iguais, como pode ser visto na tabela 4.1. Já o circuito da figura 4.3 não é equivalente com os circuitos da figura 4.1 e da figura 4.2, mas possui uma assinatura idêntica caracterizando um erro de *aliasing*.

Uma condição necessária para uma correspondência booleana é que as assinaturas correspondentes sejam iguais. Quando as assinaturas são compactas, compará-las é um método eficiente para determinar quando duas funções não coincidem, e portanto, reduzir o espaço de busca para uma correspondência. Por causa de erros de *aliasing*, as assinaturas não representam condições suficientes para inferir a correspondência. Assim, elas são peculiarmente menos poderosas do que formas canônicas. As assinaturas foram usadas antes da introdução de formas canônicas e, posteriormente, em casos que as formas canônicas são custosas para serem calculadas ou seu tamanho é muito grande [Mohnke and Malik, 1993].

Assinaturas podem ser obtidas considerando a contagem de satisfação de uma função, que é o número de mintermos presentes na saída da função. A contagem de satisfação de uma função f é denotada por $|f|$ [Bryant, 1986]. A contagem de satisfação é invariante para a permutação de entradas e complementação. Assim, ela pode ser usada como uma assinatura para determinar P-equivalência e PN-equivalência. Caso seja realizada a complementação da saída, a contagem de satisfação de uma função f de n entradas muda de $|f|$ para $2^n - |f|$ [Benini and De Micheli, 1997].

Benini and De Micheli [1997] apresentam funções de assinatura com o objetivo de encontrar

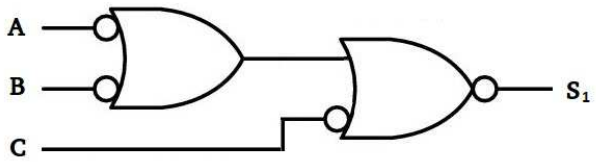

Figura 4.1 – Circuito lógico: Exemplo B

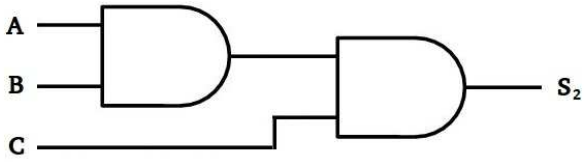

Figura 4.2 – Circuito lógico: Exemplo C

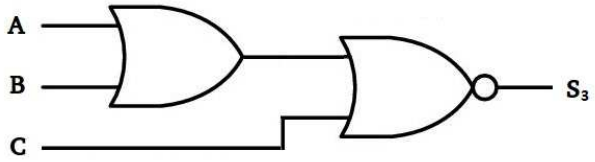

Figura 4.3 – Circuito lógico: Exemplo D

A	B	C	S1	S2	S3
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	0

Tabela 4.1 – Tabela-verdade com as assinaturas dos circuitos B, C e D.

equivalência entre classes de funções booleanas com assinaturas idênticas, e funções combinacionais que modelam células de bibliotecas de células. Então para simplificar a conceitualização dos itens a seguir, devem ser consideradas funções booleanas que modelam classes de circuitos, como funções de *cluster*. E devem ser consideradas funções combinacionais que modelam uma célula de biblioteca de células, como funções de padrão.

Os autores Mailhot and Di Micheli [1993] usaram assinaturas para reduzir o número de verificações de tautologia necessárias para determinar a equivalência de P-equivalência e NPN-equivalência. Estas assinaturas são baseadas nos seguintes fatos:

- Qualquer atribuição de variável deve associar uma variável *unate* (*binate*) na função de *cluster* com uma variável *unate* (*binate*) na função de padrão.

- Variáveis ou grupos de variáveis intercambiáveis na função de $cluster$, devem ser intercambiá-

veis na função de padrão.

A primeira condição implica que as funções de *cluster* e padrão devem ter o mesmo número de variáveis *unate* (*binate*) para existir uma correspondência [Benini and De Micheli, 1997]. Uma variável *unate* significa que se a variável existir na função booleana de forma negada, todas as ocorrências dessa variável na função booleana devem ser na forma negada e vice-versa, e para uma variável ser considerada *binate* significa que deve existir pelo menos uma ocorrência dessa variável na função booleana na forma negada e uma ocorrência na forma não negada.

Se denotarmos por b o número de variáveis *binate*, então b é uma assinatura da função. Portanto, o número de variáveis *unate* $(n-b)$ é também uma assinatura. Além disso, no máximo $b!(n-b)!$ permutações de variáveis precisam ser consideradas na busca de uma correspondência no pior caso [Benini and De Micheli, 1997].

Exemplo 1: Uma função booleana de padrão de biblioteca de células $g = s1s2a + s1s2'b + s1's3c + s1's3'd$ com $n = 7$ variáveis, possui 4 variáveis $unate$ (a, b, c, d) e 3 variáveis $binate$ $(s1, s2, s3)$. Considerando uma função de cluster f com $n = 7$ variáveis, uma condição necessária para existir equivalência entre f e g, é que a função f também possua 4 variáveis $unate$ e 3 variáveis $binate$. Se este for o caso, apenas $3!4! = 144$ ordens de variáveis devem ser considerados no pior dos casos [Benini and De Micheli, 1997].

A segunda condição permite explorar propriedades de simetria para simplificar a busca de uma correspondência [Mailhot and Di Micheli, 1993, Morrison et al., 1989]. Considerando o conjunto de suporte de uma função $f(x)$, um conjunto de simetria é um conjunto de variáveis que são intercambiáveis por pares, isto é, são cláusulas de uma função booleana onde uma pode ser trocada pela outra, sem afetar a funcionalidade lógica. Uma classe de simetria é um conjunto de conjuntos de simetria com a mesma cardinalidade. Denotamos uma classe de simetria por Ci quando seus elementos têm cardinalidade $i, i = 1, 2, ..., n$. Portanto, as classes podem ser vazias. As classes de simetria das funções de padrão podem ser calculadas de antemão e fornecem uma assinatura para os próprios padrões. Na verdade, uma condição necessária para a correspondência é ter classes de simetria da mesma cardinalidade para cada $i = 1, 2, ..., n$ [Benini and De Micheli, 1997].

Exemplo 2: A função $f = x1x2x3 + x4x5 + x6x7$ possui variáveis de suporte que podem ser divididas em três conjuntos de simetria: $\{x1x2x3\}, \{x4x5\}, \{x6x7\}$. Existem duas classes de simetria não-vazias para a função f, sendo elas: $C2 = \{\{x4, x5\}, \{x6, x7\}\}$ e $C3 = \{\{x1, x2, x3\}\}$. Assim, uma assinatura para f é $[0, 2, 1, 0, 0, 0, 0]$, onde cada valor da lista representa uma classe Ci. Agora considerando as funções de padrão $g1 = y1 + y2y3 + y4y5 + y6y7$ e $g2 = (y1' + y2').(y3 +$

$y4).(y5 + y6 + y7)$. As assinaturas das funções $g1$ e $g2$ são, respectivamente, $[1, 3, 0, 0, 0, 0, 0]$ e $[0, 2, 1, 0, 0, 0, 0]$. Agora é possível observar que as assinaturas de f e $g2$ são iguais, e de fato $g2$ é NPN-equivalente à f. Porém é importante observar que a associação geral de assinaturas, é apenas uma condição necessária para a correspondência booleana [Benini and De Micheli, 1997].

5 CORRESPONDÊNCIA DE SAÍDAS POR SIMULAÇÃO

Através de simulações, este trabalho propõe uma técnica para encontrar correspondência entre as saídas de circuitos digitais. A técnica consiste em realizar grupos de simulações, onde cada grupo é formado por 1000 simulações consecutivas de um mesmo caso de teste, com entradas aleatórias diferentes em cada grupo. Para tentar validar essa técnica, a estratégia utilizada, foi simular várias vezes um mesmo circuito. Em cada simulação de um caso de teste, são alimentadas aleatoriamente as entradas do circuito e o processo de simulação dos componentes lógicos é realizado até serem gerados os valores das saídas.

Após cada simulação, os valores lógicos das saídas precisam ser contabilizados num somatório para cada saída. Esse somatório por saída, soma 1 quando o valor lógico da saída for verdadeiro e soma 0 caso seja falso. O objetivo da contagem dos valores das saídas é saber a proporção de quanto dos valores das saídas é 0 ou quanto é 1 do total das simulações.

O somatório de cada saída representa a contagem de satisfação de uma função booleana, ou seja, é a contagem de valores 1 presentes na saída da função. Através de grupos de simulações de um caso de teste, são obtidas as contagens de valores 1 das saídas. Os valores dessas contagens são usados para identificar as saídas com proporção de valores 1 mais próxima entre os diferentes grupos de simulação, através da ordenação dos valores de cada grupo.

A ordenação das contagens de valores 1 das saídas é uma forma de identificar as saídas independentemente de permutação e associar as possíveis saídas com correspondência.

Por exemplo, na tabela 5.1 um certo caso de teste possui listado nas colunas as contagens de valores 1 das saídas dos dois diferentes grupos de simulação de cada linha da tabela. Na linha do grupo 1, do conjunto total de simulações do grupo que neste caso é 1000 simulações, a saída f obteve 261 simulações que geraram o valor 1 na saída, a saída g obteve 459 simulações que geraram o valor 1 na saída e a saída h obteve 720 simulações que geraram o valor 1 na saída.

Ao ser realizada a ordenação dos valores de contagem dos grupos 1 e 2 da tabela 5.1, é verificado que a saída f do grupo 1 com valor 261, e a saída f do grupo 2 com valor 244 de contagem das simulações são saídas que foram identificadas pela posição após a ordenação, através da proporção de contagem de valores 1 mais próxima. Da mesma maneira, a saída g do grupo 1 com a saída g do grupo 2 possuem identificadas as proporções de contagem mais próximas e por último também, são identificadas as saídas h nos dois grupos.

A tabela 5.2 mostra os grupos de simulação 3 e 4 do caso de teste após a ordenação dos valores

Grupo 1	[f] 261	[g] 459	[h] 720
Grupo 2	[f] 244	[g] 519	[h] 763

Tabela 5.1 – Contagem de mintermos-1 das saídas.

de proporção das saídas, onde na linha do grupo 3, a saída f obteve 243 valores 1 e na linha do grupo 4 a saída g obteve 123 valores 1, sendo identificadas pela posição as saídas com as proporções mais próximas uma com a outra. Já para a saída g do grupo 3 com 383 valores 1 foi identificada com proporção mais próxima para com a saída f do grupo 4 com 490 valores 1 e por fim a saída h do grupo 3 é identificada com proporção de valores 1 mais próxima para com a saída h do grupo 4.

Grupo 3	[f] 243	[g] 383	[h] 573
Grupo 4	[g] 123	[f] 490	[h] 663

Tabela 5.2 – Contagem ordenada de mintermos-1 das saídas.

Verificando a posição das saídas $[f, g, h]$ dos grupos 1 e 2 da tabela 5.1, é identificado que entre os esses dois grupos exite uma acurácia de 100% das saídas, pois após a ordenação dos valores das proporções, as saídas corretas ficaram na mesma posição. Na tabela 5.2, os grupos 3 e 4, possuem uma acurácia de aproximadamente 33,33%, pois apenas as proporções de valores 1 da saída h dos dois grupos, ficaram na mesma posição, o que gerou o acerto para a saída. Portanto, com diferentes grupos de simulações como os das tabelas 5.1 e 5.2 de um mesmo circuito, este trabalho se propõe a testar a técnica de checagem de equivalência das saídas a fim de encontrar uma acurácia média satisfatória entre grupos de simulações gerados em um tempo finito estabelecido.

O objetivo das simulações é validar a hipótese que, dados os circuitos A e B, com suas saídas permutadas através da ordenação pela proporção seja possível identificar as saídas correspondentes. Então, para isso são realizados grupos de simulações com as contagens das saídas de um mesmo caso de teste com conjuntos diferentes de valores aleatórios para as entradas, como se cada grupo de simulações fosse pertencer a outro circuito, porém pertencendo ao mesmo, a ideia é que as contagens desses grupos gerem uma acurácia satisfatória para o caso de teste.

6 METODOLOGIA

A técnica de checagem de equivalência das saídas de circuitos, desenvolvida neste trabalho, propõe simular muitas vezes um mesmo circuito para gerar as linhas com a contagem dos valores 1 das saídas de cada grupo de simulações. Para validar essa técnica é necessário simular conjuntos de valores aleatórios para as entradas e verificar os valores presentes nas saídas para cada simulação de um circuito.

6.1 Casos de teste

Os casos de teste consistem em diferentes arquivos em *verilog* que representam os projetos lógicos de circuitos digitais. Os casos de teste foram obtidos do ICCAD 2015 e ICCAD 2016, um dos principais concursos na área de eletrônica e automação, o qual disponibiliza desafios de programação para solucionar problemas envolvidos com a indústria de circuitos integrados, sistemas embarcados e eletrônica.

6.2 Simulação dos casos de teste

Para simular os casos de teste foi usado como base uma ferramenta desenvolvida no projeto de pesquisa de circuitos digitais da UFFS, a qual encontra-se disponível no *Github*[1] . Em seu funcionamento padrão a ferramenta tem métodos para carregar um circuito em *verilog*, gerar valores aleatórios para as entradas e simular os valores presentes nas saídas a partir dos valores gerados para as entradas.

A ferramenta de simulação de circuitos foi adaptada e ampliada para simular lotes de casos de teste e gerar a cada 1000 simulações de um circuito, uma linha com a contagem de valores 1 que apareceram em cada saída, sendo cada linha de resultados correspondente a um grupo de simulações. Para cada uma das 1000 simulações, novas entradas aleatórias foram geradas e para as simulações de cada caso de teste foi delimitado o tempo de 1800 segundos. O Algoritmo 1 mostra a ordem dos eventos de simulações de cada caso de teste para criação das linhas de resultados com as contagens de valores 1 das saídas.

[1] *https://github.com/wuerges/iccad2016uffs*

6.2.1 Contagem dos valores 1 das saídas

Junto ao módulo de simulação de circuitos, foi desenvolvido um método que realiza a contagem de valores 1 das saídas dos circuitos. Ao final de cada simulação são identificados os valores das saídas e gravados em uma estrutura de contagem para cada saída, até a conclusão das 1000 simulações, quando é gerada uma linha identificando a contagem de satisfação de cada saída, ou seja, a contagem de valores 1. O momento da criação da linha de contagem de cada grupo pode ser observado no Algoritmo 1 abaixo.

Algoritmo 1: Simulação e contagem dos valores 1 das saídas

```
Entrada: circuito caso de teste
Saída: linhas de grupos de simulações
início
    inicialização;
    repita
        repita
            gera um novo conjunto de entradas aleatórias;
            simulação e contagem de valores 1 das saídas;
        até chegar a 1000 simulações;
        criação da linha de contagem de valores 1 do grupo;
    até 1800 segundos;
    sobre cada linha das linhas de grupos, ordenação da
     contagem de valores 1 das saídas;
fim
```

6.2.2 Cálculo da cobertura das saídas

Para tentar encontrar uma relação entre o quanto foi simulado cada caso de teste e a acurácia média das saídas, foi desenvolvido o cálculo da cobertura. A cobertura consiste na quantidade de simulações realizadas em relação a quantidade de simulações necessárias para contemplar todas as combinações de valores das saídas.

Conforme a equação 6.1, o cálculo da cobertura é realizado através da divisão da quantidade de grupos de simulações realizadas para cada caso de teste multiplicado por 1000, pela quantidade de combinações de valores possíveis no circuito. A variável G representa a quantidade de grupos de simulações realizadas para as saídas, que multiplicando por 1000 simulações por grupo, dá o total de simulações por caso de teste e $2^{\in}$ significa as possibilidades de combinações de valores para as entradas. Assim os casos de teste que obtiveram os valores de cobertura entre 0 e 1, estão variando de 0% a 100% de cobertura, e os casos de teste com o valor da cobertura maior que 1, são casos de teste que foram simulados além do valor máximo de possibilidades de valores para as entradas,

considerados com cobertura maior que 100%.

$$C = \frac{G * 1000}{2^{\in}} \tag{6.1}$$

6.3 Verificação da acurácia média das saídas

Neste trabalho, a acurácia média das saídas se refere ao percentual de acerto das saídas de um mesmo circuito, através de simulações. Foi usada a quantidade de valores 1 das saídas de cada grupo de simulações para realizar uma aproximação da contagem dos valores 1 das saídas entre as contagens dos diferentes grupos de simulações. Sobre cada par de grupos o algoritmo tenta identificar as saídas correspondentes a partir da quantidade de valores 1 de cada saída. Cada grupo de simulações é representado por uma linha de resultados após 1000 simulações. A acurácia das saídas é calculada após a ordenação das contagens de valores 1 das saídas de cada linha de contagens. Através da ordenação dos valores de contagem das linhas é esperado que as saídas correspondentes fiquem posicionadas nas mesmas posições dentro de cada linha de resultados.

Com o resultado das simulações de cada caso de teste, ou seja, as linhas com a contagem de valores 1 das saídas, foi verificada a acurácia média das saídas com base nas equações 6.2 e 6.3. A equação 6.2, determina acurácia entre os pares de linhas gerados, de modo que a cada acerto de uma saída no par de linhas testado, é somado 1 (um) ao totalizador de acertos, e ao final da verificação o total de acertos é dividido pela quantidade de saídas. A equação 6.3, determina a acurácia média das saídas, considerando todas[2] linhas geradas para o circuito. Assim, o a acurácia média é calculada através da soma da acurácia de todas as possíveis combinações de pares de tuplas diferentes, dividido pelo número de pares de tuplas testadas.

$$a(y, \bar{y}) = \frac{\sum_{i=1}^{n} \begin{cases} 1 & \text{se } y_i = \bar{y}_i \\ 0 & \text{senão} \end{cases}}{n} \tag{6.2}$$

$$\|Y\| = \frac{\sum_{i \neq j} a(y_i, \bar{y}_j)}{n(n-1)} \tag{6.3}$$

Para um melhor entendimento do cálculo da acurácia média de cada caso de teste, o Algoritmo 2 mostra a sequência de eventos para a realização dos cálculos. Após ter sido carregado o arquivo com os grupos de simulações, as linhas dos grupos foram percorridas de uma em uma, para que com

[2] Nos casos de teste que a cobertura é muito maior que 100%, foi gerada uma quantidade de grupos de resultados superior ao necessário para cobrir as possibilidades de valores das saídas, portanto, foi limitado o uso de até 200 grupos de simulações nesses casos de teste para o cálculo da acurácia, evitando um esforço desnecessário.

cada uma das linhas fosse testada a acurácia com todas as outras. Depois do teste da acurácia entre as combinações de linhas foi realizado o cálculo da média das acurácias dos pares de linhas, ou seja, o cálculo da acurácia média do caso de teste.

Algoritmo 2: Cálculo da acurácia média do caso de teste

```
Entrada: linhas de grupos de simulações
Saída: acurácia média do caso de teste
início
    inicialização;
    repita
        repita
            if linha grupo i diferente de linha grupo j then
                verifica acurácia entre saídas do grupo i e grupo
                 j;
        até grupo i percorrer todo linhas de grupos;
    até grupo j percorrer todo linhas de grupos;
    a partir da acurácia entre os pares de grupos, cálculo da
     acurácia média do caso de teste;
fim
```

7 RESULTADOS

Os resultados calculados através das simulações são a cobertura de simulações das saídas e a acurácia média das saídas dos casos de teste. Foram criados dois gráficos para relacionar as variáveis calculadas, um que possui os dados dos casos de teste com alta cobertura e outro com os dados dos casos de teste de baixa cobertura. Esta separação foi necessária, visto que, os casos de teste de baixa cobertura geraram valores de cobertura muito próximos de 0 (zero), dificultando a visualização dos dados dos circuitos de alta cobertura.

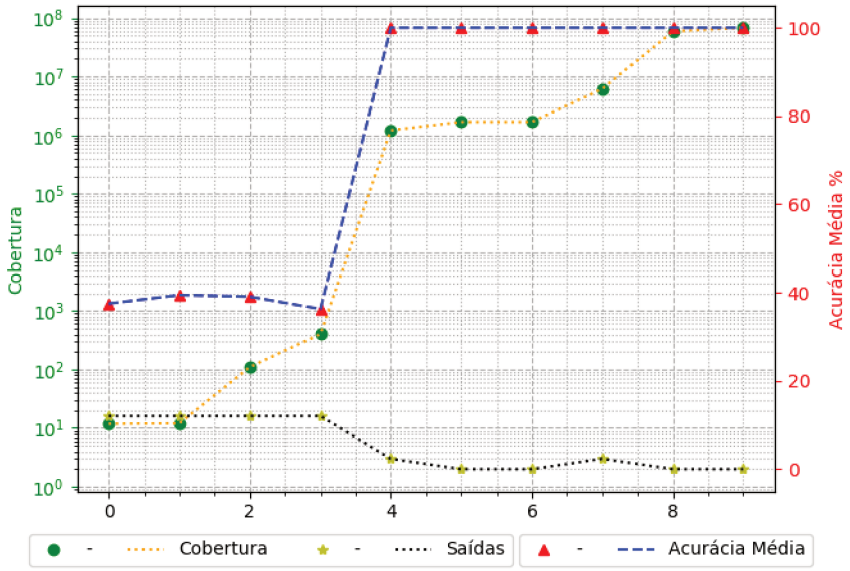

Figura 7.1 – Alta Cobertura / Acurácia Média

A figura 7.1 mostra os dados dos casos de teste que possuem alta cobertura. Além da cobertura, estão relacionadas na figura 7.1 o número de saídas e a acurácia média. É possível visualizar na figura 7.1 que apesar da alta cobertura de simulações, exitem casos de teste que possuem acurácia média muito baixa. Como todos os casos de teste da figura 7.1 possuem cobertura maior que 100%, não é seguro afirmar que a acurácia média esta relacionada com a alta cobertura.

De certo modo, está visível uma maior acurácia média nos casos de teste de cobertura maior que 100%, mas não é o suficiente para encontrar a correspondência das saídas de todos os casos de teste. Por exemplo, os casos de teste do 0 ao 3 da figura 7.1, possuem uma cobertura muito maior que 100%, porém sua acurácia média é inferior a 50%.

Com um número de saídas muito maior nos casos de teste, a figura 7.2 possui os casos de

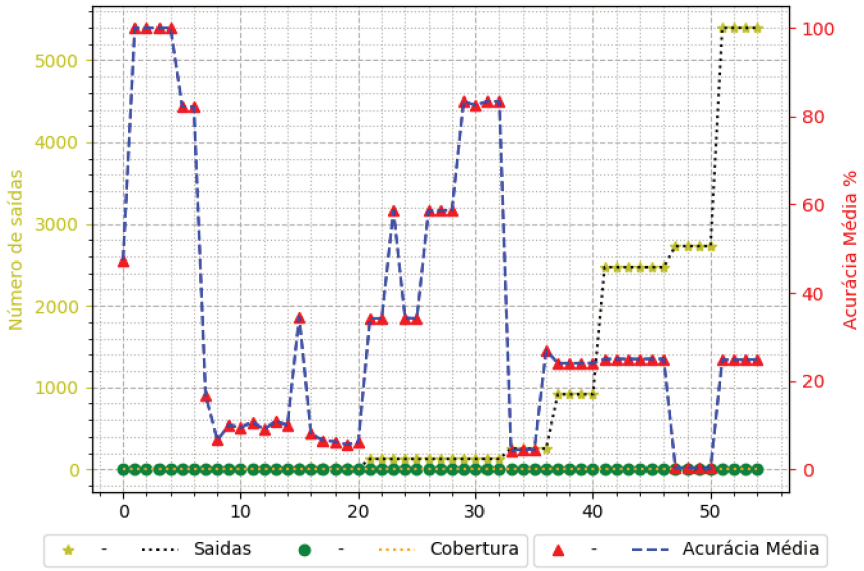

Figura 7.2 – Baixa Cobertura / Acurácia Média

teste com cobertura muito próxima de 0 (zero), sendo valores que não puderam ser calculados com precisão por não serem suportados por uma variável do tipo *double*. Isto significa que foi possível simular uma quantidade muito pequena de possíveis combinações de valores para as saídas, ou seja, uma cobertura muito baixa foi alcançada.

Ordenados pela quantidade de saídas, os casos de teste de baixa cobertura mostrados no gráfico da figura 7.2 possuem diferentes valores de acurácia média. Por exemplo, os casos de teste 19 e 20 da figura 7.2 possuem acurácia média menor que 10%, enquanto os casos do 1 ao 4 possuem uma acurácia média bem melhor, chegando a 100%. Esses dois conjuntos de casos de teste com quantidades de saídas muito próximas, sendo menores que 200 saídas, possuem uma diferença muito grande nos resultado de acurácia média.

Outra observação interessante é que em comparação aos casos de teste 41 ao 46, que apesar de terem a quantidade de saídas muito superior com cerca de 2500 saídas e com acurácia média maior que 20%, os casos de teste do 37 ao 40 possuem pouca acurácia média levando em conta sua quantidade de saídas relativamente pequena.

7.0.1 Resultados com cálculo de cobertura a partir das saídas

Além dos resultados da cobertura dos circuitos a partir das possibilidades de valores para as entradas, foi gerado um conjunto de dados da cobertura usando as possibilidades de valores para as

saídas como base para o cálculo da cobertura, onde a quantidade de simulações dos casos de teste é dividida pelas possibilidades de valores para as saídas, conforme equação 7.1. O Apêndice B mostra as figuras da alta e baixa cobertura das saídas, exibindo o número de saídas e usando para ordenação dos casos de teste da baixa cobertura a quantidade de saídas.

$$C = \frac{G * 1000}{2^S} \tag{7.1}$$

7.0.2 Resultados com 3000 simulações por grupo

Para comparar e validar os resultados da acurácia das saídas dos casos de teste com definição de 1000 simulações por grupo, foi simulado um novo conjunto de resultados com 3000 simulações para cada grupo de simulações de um caso de teste e o tempo de simulação foi aumentado de 1800 segundos para 5400 segundos por caso de teste.

Apesar de ter sido aumentado significativamente o esforço de simulação, o conjunto de resultados obtidos foi muito semelhante ao do conjunto de resultados obtido com cálculo de cobertura a partir das saídas exibido no Apêndice B. Para a visualização dos detalhes, o conjunto de dados adicional gerado, encontra-se exibido graficamente no Apêndice D, nas figuras de alta e baixa cobertura, que foram calculadas com base nas possibilidades de valores para as saídas, assim como os resultados adicionais citados na subseção 7.0.1 deste trabalho.

8 CONCLUSÃO

No presente trabalho, através de simulações de casos de teste, foram obtidos os resultados da cobertura e acurácia das saídas dos circuitos lógicos, onde os resultados estão separados em conjuntos de alta e baixa cobertura de simulações, os quais estão respectivamente identificados nos gráficos das figuras 7.1 e 7.2.

As linhas do gráfico da alta cobertura mostram que apesar de ter sido alcançado uma cobertura maior que 100% em todos casos de teste, não foi possível conseguir uma boa acurácia em todos casos de teste. Com isso, para os casos de teste da alta cobertura é difícil relacionar diretamente a cobertura de simulações das saídas com a quantidade de acurácia das saídas.

No gráfico da baixa cobertura, os valores da cobertura de simulações das saídas são muito pequenos em todos os casos de teste, com valores muito próximos de 0, por este motivo não foi identificada qualquer relação com a acurácia média das saídas dos circuitos.

Os casos de teste mostrados na figura da alta cobertura, possuem o número de saídas muito pequeno em relação aos casos de teste da baixa cobertura, mas isso não se mostrou o suficiente para a melhoria da acurácia média em todos casos de teste. Na figura da baixa cobertura é visível que mesmo com muita diferença na quantidade de saídas, vários casos de teste com quantidades de saídas muito diferentes possuem acurácia muito semelhante, impossibilitando a identificação de uma relação entre a cobertura e o número de saídas.

A forma como foram implementados os cálculos da contagem de satisfação booleana das saídas e da permutação das saídas dos circuitos, se mostrou uma função de assinatura pouco eficiente para a maioria dos casos de teste. Apenas alguns casos de teste específicos e com poucas saídas, obtiveram o resultado de 100% de acurácia média das saídas.

Através dessas caraterísticas dos resultados foi concluído que existem mais fatores e variáveis sobre as características dos circuitos a serem levados em conta para alcançar resultados melhores, além da cobertura, da quantidade de saídas, e do tempo de simulação de cada caso de teste.

REFERÊNCIAS

L. Benini and G. De Micheli. A survey of boolean matching techniques for library binding. *ACM Transactions on Design Automation of Electronic Systems*, 2(3):193–226, 1997.

R. E. Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

Bu, Huiming. 5 nanometer transistors inching their way into chips, June 2017. URL <https://www.ibm.com/blogs/think/2017/06/5-nanometer-transistors/>. Access date: June 5, 2017.

G. De Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education, 1994.

H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 365–376. IEEE, 2011.

M. R. Garey and D. S. Johnson. Computers and intractability, 40 wh freeman and co. *New York*, 41, 1979.

F. Mailhot and G. Di Micheli. Algorithms for technology mapping based on binary decision diagrams and on boolean operations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):599–620, 1993.

J. Mohnke and S. Malik. Permutation and phase independent boolean comparison. *INTEGRATION, the VLSI journal*, 16(2):109–129, 1993.

C. Morrison, R. Jacoby, and G. Hachtel. Techmap: Technology mapping with delay and area optimization. *Logic and Architecture Synthesis for Silicon Compilers*, pages 53–64, 1989.

R. Tocci, N. Widmer, and G. Moss. São Paulo, 10^a edition, 2007.

APÊNDICES

APÊNDICE A – Informações sobre os casos de teste

Casos de Teste	Número de Entradas	Número de Saídas	Grupos gerados
testcases/trivial/not1.v.csv	1	2	118904
testcases/trivial/buf1.v.csv	1	2	136569
testcases/cad16_np3_case2/case10/cir1.v.csv	4	2	26902
testcases/cad16_np3_case2/case10/cir2.v.csv	4	2	26913
testcases/cad16_np3_case2/case6/cir1.v.csv	512	2	56
testcases/cad16_np3_case2/case8/cir1.v.csv	2048	2	40
testcases/cad16_np3_case2/case8/cir2.v.csv	2048	2	48
testcases/cad16_np3_case/case0/cir1.v.csv	3	3	49160
testcases/cad16_np3_case/case0/cir2.v.csv	5	3	38843
testcases/cad16_np3_case2/case12/cir1.v.csv	512	5	121
testcases/cad16_np3_case2/case12/cir2.v.csv	512	5	190
testcases/cad16_np3_case2/case14/cir1.v.csv	36	7	1841
testcases/cad16_np3_case2/case14/cir2.v.csv	36	7	1533
testcases/cad16_np3_case2/case13/cir1.v.csv	192	10	4
testcases/cad16_np3_case2/case13/cir2.v.csv	192	10	4
testcases/cad16_np3_case2/case7/cir1.v.csv	2184	10	23
testcases/cad16_np3_case2/case7/cir2.v.csv	2186	10	19
testcases/cad16_np3_case2/case11/cir1.v.csv	193	10	109
testcases/cad16_np3_case2/case11/cir2.v.csv	192	10	245
testcases/cad16_np3_case2/case9/cir1.v.csv	2151	10	24
testcases/cad16_np3_case2/case9/cir2.v.csv	2151	10	21
testcases/cad16_np3_case/case3/cir1.v.csv	249	15	99
testcases/cad16_np3_case/case3/cir2.v.csv	250	15	284
testcases/cad16_np3_case/case1/cir1.v.csv	13	16	900
testcases/cad16_np3_case/case1/cir2.v.csv	16	16	788
testcases/cad16_np3_case/case4/cir1.v.csv	11	16	825
testcases/cad16_np3_case/case4/cir2.v.csv	16	16	770
testcases/cad16_np3_case/case2/cir1.v.csv	42	16	1181
testcases/cad16_np3_case/case2/cir2.v.csv	43	16	1154
testcases/cad16_np3_case/case5/cir1.v.csv	80	16	36
testcases/cad16_np3_case/case5/cir2.v.csv	80	16	35
testcases/unit13/in_1.v.csv	99	128	14
testcases/unit13/in_2.v.csv	99	128	15
testcases/unit15/in_1.v.csv	99	128	46
testcases/unit15/in_2.v.csv	99	128	28
testcases/unit12/in_1.v.csv	99	128	18
testcases/unit12/in_2.v.csv	99	128	14
testcases/unit14/in_1.v.csv	99	128	46
testcases/unit14/in_2.v.csv	99	128	29
testcases/unit10/in_1.v.csv	56	129	32
testcases/unit10/in_2.v.csv	56	129	38
testcases/unit11/in_1.v.csv	56	129	37
testcases/unit11/in_2.v.csv	56	129	37
testcases/unit16/in_1.v.csv	128	256	8
testcases/unit16/in_2.v.csv	128	256	6
testcases/unit17/in_1.v.csv	128	256	6
testcases/unit17/in_2.v.csv	128	256	11
testcases/unit2/in_1.v.csv	249	914	33
testcases/unit2/in_2.v.csv	249	914	52
testcases/unit1/in_1.v.csv	249	914	33
testcases/unit1/in_2.v.csv	249	914	53
testcases/unit9/in_1.v.csv	650	2474	8
testcases/unit9/in_2.v.csv	650	2474	11
testcases/unit6_removed/in_1.v.csv	650	2474	9
testcases/unit6_removed/in_2.v.csv	650	2474	13
testcases/unit5/in_1.v.csv	650	2474	9
testcases/unit5/in_2.v.csv	650	2474	13
testcases/unit7/in_1.v.csv	2282	2726	7
testcases/unit7/in_2.v.csv	2282	2726	6
testcases/unit8/in_1.v.csv	2282	2726	7
testcases/unit8/in_2.v.csv	2282	2726	6
testcases/unit4/in_1.v.csv	1364	5397	6
testcases/unit4/in_2.v.csv	1364	5397	10
testcases/unit3/in_1.v.csv	1364	5397	6
testcases/unit3/in_2.v.csv	1364	5397	10

Tabela A.1 – Lista dos casos de teste

APÊNDICE B – Resultados com cálculo da cobertura a partir das saídas, ordenados pelo número de saídas na baixa cobertura

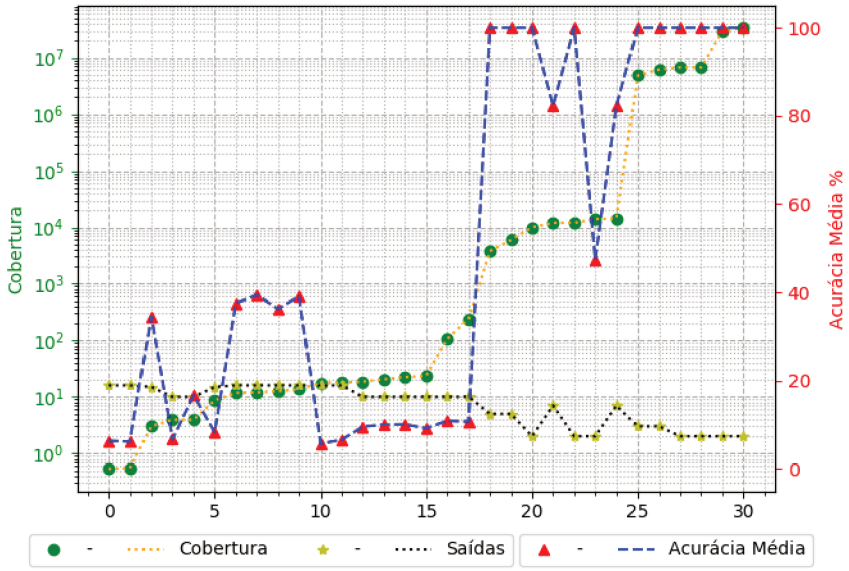

Figura B.1 – Alta Cobertura / Acurácia Média

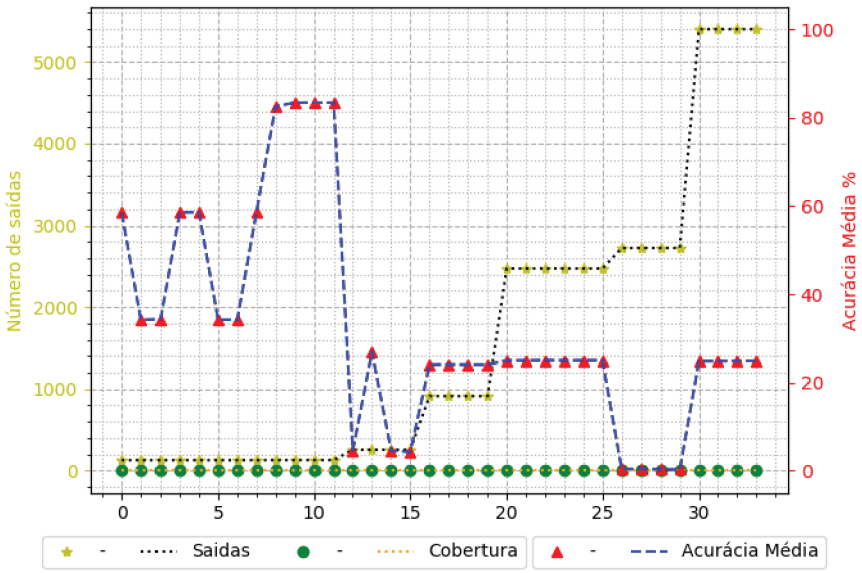

Figura B.2 – Baixa Cobertura / Acurácia Média

APÊNDICE C – Resultados com cálculo da cobertura a partir das entradas, ordenados pelo número de entradas na baixa cobertura

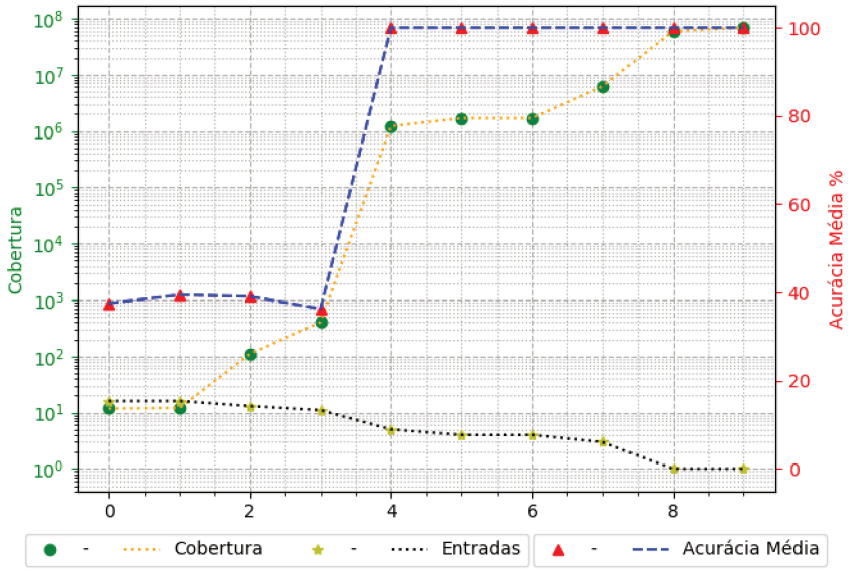

Figura C.1 – Alta Cobertura / Acurácia Média

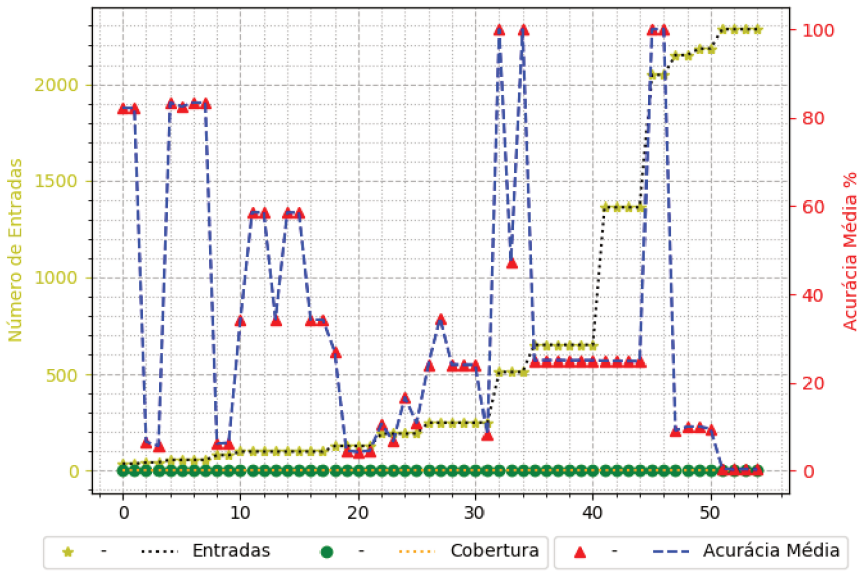

Figura C.2 – Baixa Cobertura / Acurácia Média

APÊNDICE D – Resultados com 3000 simulações por grupo, com cálculo de cobertura a partir das saídas, ordenados pelo número de saídas na baixa cobertura

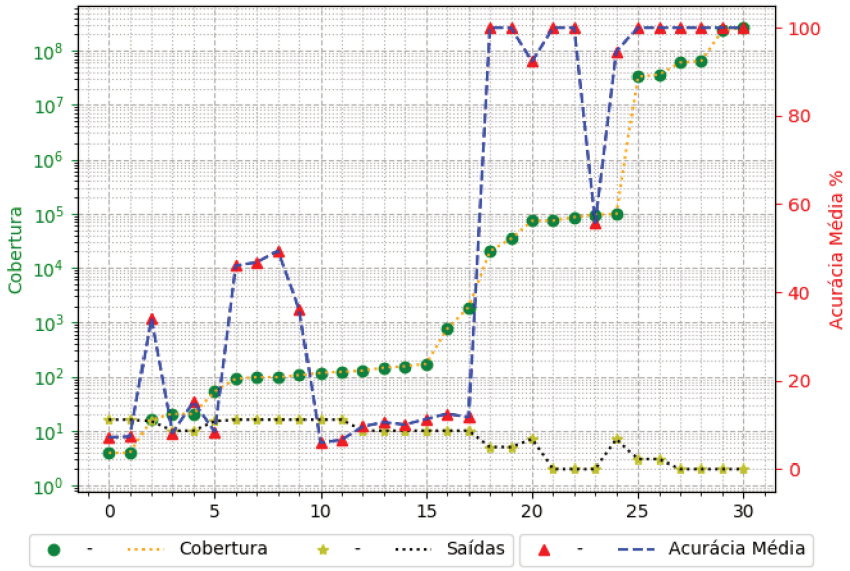

Figura D.1 – Alta Cobertura / Acurácia Média

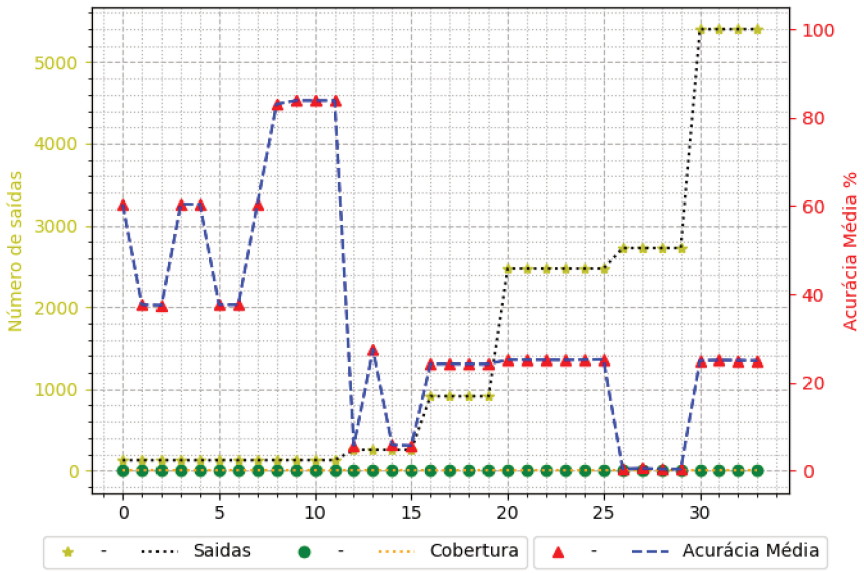

Figura D.2 – Baixa Cobertura / Acurácia Média