UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

Métricas de Testabilidade: Controlabilidade e Observabilidade em Circuitos Combinacionais

Artur Freitas Arocha

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Paulo Francisco Butzen

Rio Grande, 2019

Ficha catalográfica

A769m Arocha, Artur Freitas.
Métricas de testabilidade: controlabilidade e observabilidade em circuitos combinacionais / Artur Freitas Arocha. – 2019.
69 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2019.
Orientador: Dr. Paulo Francisco Butzen.

1. Teste de CI 2. Controlabilidade 3. Observabilidade 4. SCOAP 5. CAMELOT I. Butzen, Paulo Francisco II. Título.

CDU 004.451.25

Catalogação na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

Metricas de Testabilidade: Controlabilidade e Observabilidade em Circuitos Combinacionais

Artur Freitas Arocha

Banca examinadora:

Prof. Dr. Leomar Soares da Rosa Jr

Prof. Dr. Vagner Santos da Rosa

Prof. Dr. Paulo Francisco Butzen
Orientador(a)

Resumo

A garantia do desempenho, da acurácia e dos baixos custos na produção em larga escala de circuitos integrados (CI) é influenciada pelo teste, sendo essa uma rotina crucial no processo de desenvolvimento de CIs. As métricas de testabilidade permitem melhorias no projeto de CIs. Durante o estudo das métricas de testabilidade no método SCOAP pelo autor, foi observada a falta de uma metodologia padrão para obtenção das equações para cálculo da controlabilidade e observabilidade para uma função lógica qualquer. Deste modo, este trabalho apresenta uma metodologia para obtenção destas equações para qualquer função lógica. A comparação dos resultados obtidos com os disponíveis na literatura mostrou que os objetivos parciais foram alcançados. Além disso, foi desenvolvido um software para automatizar a obtenção das métricas de testabilidade em circuitos com os métodos CAMELOT e SCOAP. Ambos métodos se mostraram eficazes para apontar topologias de CIs melhor projetadas com foco na testabilidade.

Palavras-chave: teste de CI. controlabilidade. observabilidade. SCOAP. CAMELOT.

Abstract

Ensuring the performance, accuracy and low costs of large-scale production of integrated circuits (IC) is influenced by test, which is a crucial routine in IC developing process. Testability measures allows improvements in IC design. This work's author observed the lack of a standard methodology to obtain the equations to calculate the controllability and observability of combinational circuits for any logical function with the SCOAP method. Thus, this work presents a methodology for obtaining these equations for any logical function. The comparison of the results obtained with those available in the literature showed that the objectives were achieved. In addition, a software was developed to automate the generation of testability metrics for ICs with the CAMELOT and SCOAP methods. Both methods shown effective for pointing out better designed IC topologies focusing on testability.

Keywords: IC test. controllability. observability. SCOAP. CAMELOT.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ASIC	*Application-Specific Integrated Circuits*
ATPG	*Automatic Test Pattern Generation*
BIST	*Built-in self-test*
CAMELOT	*Computer-aided measure for logic testability*
CC0	*Combinational 0-controllability*
CC1	*Combinational 1-controllability*
CO	*Combinational observability*
CI	Circuito Integrado
CNF	*Conjunctive Normal Form*
CY	Controlabilidade no CAMELOT
CMOS	*Complementary Metal-Oxide-Semiconductor*
DFT	*Design for Testability*
DNF	*Disjunctive Normal Form*
EDA	*Electronic Design Automation*
FPGA	*Field Programmable Gate Array*
F0	*Soma dos mintermos cuja função lógica tenha saída 0*
F1	*Soma dos mintermos cuja função lógica tenha saída 1*
LATS	*IEEE Latin-American Test Symposium*
OY	Observabilidade no CAMELOT
PLD	*Programmable Logic Device*
POS	*Product of Sums*
SAT	*Propositional Satisfiability Problem*
SCOAP	*Sandia Controllability/Observability Analysis Program*

SC0	*Sequential 0-controllability*
SC1	*Sequential 1-controllability*
SO	*Sequential observability*
SOP	*Sum of Products*

Sumário

1 Introdução

A exploração da álgebra booleana no desenvolvimento da eletrônica teve grande avanço com a invenção do transistor. A ênfase na miniaturização e no uso eficiente de circuitos eletrônicos digitais permitiu a evolução da computação. A melhoria nos processos de fabricação CMOS (*Complementary Metal-Oxide-Semiconductor*) e a redução das dimensões dos transistores, acarretou no aumento na complexidade dos circuitos integrados (CI), o que aumenta a necessidade do teste dos mesmos (WESTE; ESHRAGHIAN, 1985).

Defeitos em CIs se refletem em prejuízo. O custo na descoberta de defeitos aumenta exponencialmente a medida que avança nas etapas de desenvolvimento do CI (WESTE; HARRIS, 2015). Segundo Reis et al. (1999) há três etapas de verificação de CIs: depuração de projeto, teste de produção e teste de manutenção. Portanto é melhor descobrir possíveis falhas na etapa de depuração do projeto, antes das etapas de teste de produção e teste de manutenção.

Algumas falhas que acarretam em erros podem ser detectadas estatisticamente em um determinado projeto de circuito ainda na etapa de depuração. Por conta disto, técnicas de desenvolvimento podem minimizar a ocorrência de erros ocasionados por estas falhas em CIs a partir da modificação do projeto estrutural lógico que especificará o comportamento que o CI terá (REIS et al., 1999).

Os defeitos em CIs podem ser consequência de impurezas presentes no material usado para produzir os *wafers*, de problemas ocorridos no processo de fabricação, por desgaste natural decorrente da vida útil do circuito, por influência exterior, dentre outros. Estes defeitos podem gerar falhas. As falhas podem ser permanentes, como interconexões abertas ou em curto, intermitentes, produzidas por interferência eletromagnética, ou ainda transiente, produzida por descargas atmosféricas ou outros fenômenos transitórios (REIS et al., 1999).

Por causa da diversidade de defeitos que podem ocorrer, é difícil gerar testes para todos os defeitos possíveis. Por isso é necessária a modelagem de falhas, que permite a geração de padrões de teste. A eficiência do teste de CI é garantida pela adequada modelagem das falhas testadas (WANG; CHANG; CHENG, 2009). Segundo Bushnell e Agrawal (2004), alguns dos modelos de falhas comumente usados são: *stuck-at*, que é um modelo de falha a nível de portas lógicas; modelo de falha a nível de transistor; modelo de falha *bridging*, que representa a falha de curto circuito entre um grupo de sinais; o modelo de falhas de atraso, dentre outros.

Há um conjunto de técnicas para fornecer métodos para testar CIs a procura de falhas. Algumas destas técnicas procuram testar o maior número de nós de um CI através de vetores de teste para alcançar a maior cobertura de teste do projeto. Outras destinam-se a fazer pequenas alterações no CI para inserir pontos de teste e métodos para observar o estado de registradores no intuito de aumentar, ou tornar possível, a testabilidade.

Para testar um CI, é necessário fazer o máximo para que cada nó da topologia seja controlável e observável. Controlabilidade e observabilidade podem ser considerados como dois dos princípios básicos da testabilidade, e sua medida pode contribuir para a cobertura de teste máxima possível através de um número mínimo de vetores de teste.

Este trabalho foca nas técnicas de testabilidade baseadas nas falhas do tipo *stuck-at*, que correspondem aos sinais lógicos que assumem um valor lógico fixo. A falha lógica *stack-at* pode ser modelada ao atribuir um valor fixo (0 ou 1) a um sinal no circuito (BUSHNELL; AGRAWAL, 2004). Pode ser ocasionada em função de um curto circuito ou outra falha relacionada, como a ligação acidental do nó à fonte de energia ou ao terra do circuito (UYEMURA, 2002).

O estudo do tipo de falha estrutural *stuck-at* se justifica em função da crescente complexidade dos circuitos (*scaling*), o que inviabiliza o teste a nível de dispositivo. Além disso, verificar todos os possíveis vetores de teste para um CI tornou-se impraticável em função do tempo. Assim, é necessário limitar os testes apenas aos vetores de teste necessários para verificar o máximo possível de falhas estruturais do circuito, diminuindo o tempo total do teste.

Atualmente a geração dos vetores de teste para a cobertura das falhas de CIs é feita por softwares ATPG (*Automatic Test Pattern Generation*). ATPGs podem ser classificados em estruturais e baseados em SAT (*Propositional Satisfiability Problem*). ATPGs estruturais permitem a geração automática dos vetores de teste a partir do levantamento das possíveis falhas *stack-at* do CI (WANG; WU; WEN, 2006). Segundo Cormen (2009), ATPGs baseados em SAT são de complexidade NP-completo e verificam se existe algum conjunto de valores para as variáveis de entrada que torne a expressão booleana verdadeira.

Existem propostas na literatura de implementação de ATPGs estruturais como D-algorithm, PODEM, FAN, ZALG (WESTE; ESHRAGHIAN, 1985) e TEST/80 (BREUER; FRIEDMAN, 1979). As métricas de testabilidade, assim como a informação estrutural de um circuito, são importantes para um ATPG alcançar ampla faixa de cobertura de falhas de seus vetores de teste. Alguns métodos para levantamento de métricas de testabilidade de CIs são CAMELOT (BENNETTS; MAUNDER;

ROBINSON, 1981), TMEAS (GRASON, 1979) e SCOAP (GOLDSTEIN; THIGPEN, 1980).

A partir do estudo do tema testabilidade de circuitos integrados digitais, verificou-se a importância das métricas de testabilidade para os softwares ATPG. Dentre as métricas mais citadas na literatura destacaram-se as geradas pelo método SCOAP e as geradas pelo método CAMELOT.

No estudo do método SCOAP verificou-se, a partir de todas as fontes consultadas, que não havia um método estabelecido para a geração das equações para o cálculo das métricas de testabilidade associadas ao SCOAP para qualquer função lógica. As equações disponíveis na literatura abrangiam apenas 8 funções diferentes, o que limitava o método a ser usado apenas em circuitos digitais que dispunham das portas lógicas para as quais haviam equações pré existes na literatura.

Da inexistência de um método para a geração das equações SCOAP para qualquer função lógica surgiu a necessidade em desenvolver este método, para com isso poder desenvolver um software para a geração automática destas equações. Esta necessidade se dá à vista que bibliotecas de células podem conter especificações de portas lógicas digitais com qualquer função lógica, não se limitando às funções disponíveis na literatura que aborda o método SCOAP.

O estudo do método CAMELOT possibilitou comparar os métodos SCOAP e CAMELOT. Para esta comparação surgiu a necessidade em desenvolver softwares para processar diferentes topologias de circuitos digitais combinacionais em ambas metodologias, SCOAP e CAMELOT. Os dados obtidos ainda podem proporcionar trabalhos futuros, conforme o exposto no capítulo de conclusão deste trabalho.

1.1 Objetivo

Este trabalho tem como objetivo geral dar uma visão da área de teste de CIs, com ênfase nas métricas de testabilidade de sinais lógicos para circuitos combinacionais. Durante o estudo dos métodos para obtenção de testabilidade de CIs, foi observada a falta de um método padrão para obtenção das equações para cálculo da controlabilidade e observabilidade de circuitos combinacionais para uma função lógica qualquer. Como objetivo específico, o presente trabalho se dispõem a propor uma solução genérica para obtenção das equações para o cálculo da controlabilidade e da observabilidade de funções lógicas usando o método SCOAP. Além disso irá se implementar uma ferramenta de análise de circuitos para calcular a observabilidade e a controlabilidade de CIs com base na metodologia desenvolvida neste trabalho para o método SCOAP e no método CAMELOT, assim como posterior análise dos

resultados obtidos.

1.2 Organização do texto

Este trabalho contém 6 capítulos e dois anexos. A Introdução trás uma breve explanação do tema abordado e os objetivos deste trabalho. O Capítulo 2 trata de conceitos relacionados com a proposta deste trabalho e apresenta algumas aplicações recentes do tema deste trabalho. O Capítulo 3 trata da explicação do método para criação das equações SCOAP para qualquer função lógica. O Capítulo 4 explica como foram implementados os softwares para obtenção das equações SCOAP para qualquer função, para obtenção da controlabilidade e observabilidade para o método CAMELOT e da ferramenta para cálculo das métricas de testabilidade de CIs. O Capítulo 5 aborda os resultados e a análise de escalabilidade dos softwares desenvolvidos. O Capítulo 6 trás as considerações finais e os anexos contém as imagens das topologias dos diferentes circuitos estudados neste trabalho com os valores de controlabilidade e observabilidade para o SCOAP e o CAMELOT em cada nodo do circuito.

2 Referencial teórico

O referencial teórico deste trabalho pretende dar uma visão geral sobre as várias métricas de testabilidade disponíveis. Os circuitos digitais abordados neste trabalho são formados por combinações de portas lógicas conectadas entre si para criar um arranjo lógico funcional. Circuitos digitais podem ser projetados usando a metodologia *standard cell*. Nesta metodologia, uma porta lógica é um bloco pré-definido que em geral representa uma função lógica, como NAND, NOR ou um *flip-flop*. A metodologia *standard cell* usa uma biblioteca de células para o desenvolvimento de *application-specific integrated circuits* (ASIC), que são CIs de uso específico. Apesar de existirem outros estilos de projeto de circuitos digitais como *macro cells*, *gate arrays*, PLD (*Programmable Logic Device*) e FPGA (*Field Programmable Gate Array*), este trabalho está centrado no escopo de projeto ASIC.

2.1 Representação padrão de funções lógicas

Uma das propriedades das variáveis lógicas é a fase, ou polaridade. Em termos de variáveis booleanas, o complemento de um sinal de valor 0 é 1, e o complemento de um sinal de valor 1 é 0, ou seja, complemento é a negação de uma variável e representa o valor booleano inverso. Quando uma variável lógica está em sua fase positiva significa que ela não está invertida, ou seja, que não possui o valor do seu complemento. Uma variável em sua fase negativa possui o valor do seu complemento (JUNIOR et al., 2008).

Para Wakerly (2000) a representação mais básica de uma função lógica é a tabela verdade. A tabela verdade lista todas as possíveis entradas do circuito e suas respectivas saídas, organizada em linhas com as combinações de entrada em ordem binária ascendente. A quantidade de linhas de uma tabela verdade é definida por 2^n, onde n é a quantidade de variáveis da função lógica que origina a tabela verdade.

A informação contida em uma tabela verdade pode ser convertida em uma função algébrica, onde as variáveis são representadas por letras. As expressões podem ser formadas, na sua forma canônica, por variáveis isoladas, conjunções (produto de variáveis), disjunções (soma de variáveis), soma de produtos (SOP) ou produto de somas (POS) (WAKERLY, 2000).

Um termo normal é um produto de somas ou uma soma de produtos em que nenhuma variável aparece mais de uma vez. Um mintermo é um produto de termo normal com n variáveis. Um maxtermo é uma soma de termo normal com n variáveis.

A Tabela 1 mostra os mintermos e os maxtermos para uma tabela verdade de três variáveis (WAKERLY, 2000).

Tabela 1 – Mintermos e maxtermos para uma função lógica F(X,Y,Z)

Linha	X	Y	Z	F	Mintermo	Maxtermo
0	0	0	0	F(0,0,0)	X'.Y'.Z'	X+Y+Z
1	0	0	1	F(0,0,1)	X'.Y'.Z	X+Y+Z'
2	0	1	0	F(0,1,0)	X'.Y.Z'	X+Y'+Z
3	0	1	1	F(0,1,1)	X'.Y.Z	X+Y'+Z'
4	1	0	0	F(1,0,0)	X.Y'.Z'	X'+Y+Z
5	1	0	1	F(1,0,1)	X.Y'.Z	X'+Y+Z'
6	1	1	0	F(1,1,0)	X.Y.Z'	X'+Y'+Z
7	1	1	1	F(1,1,1)	X.Y.Z	X'+Y'+Z'

Usa-se mintermo i e maxtermo i para representar o mintermo e o maxtermo de uma determinada linha da tabela verdade, onde i é a linha da tabela. Para os mintermos, uma variável específica é representada pelo seu complemento caso o *bit* correspondente na representação binária de i for 0, do contrário, a variável é representada pela sua letra correspondente. O equivalente vale para os maxtermos, mas invertido, pois uma variável específica é representada pelo seu complemento apenas caso o *bit* correspondente na representação binária de i for 1(WAKERLY, 2000).

A função booleana de uma tabela verdade pode ser escrita pela disjunção dos mintermos das linhas da tabela verdade onde a função tem valor 1, ou seja, uma soma de produtos. Esta forma de representação da função booleana é a primeira forma canônica ou forma disjuntiva (no inglês *disjunctive normal form* ou DNF) (ZUFFO, 1976). A notação para uma soma de mintermos é, considerando por exemplo um conjunto contendo os mintermos 1, 4, 5 e 6 de uma função DNF com três variáveis, $\sum_{X,Y,Z}(1,4,5,6)$ (WAKERLY, 2000).

Assim como é possível escrever uma função booleana de uma tabela verdade usando mintermos, também é possível a escrever usando a conjunção dos maxtermos das linhas da tabela verdade onde a função tem valor 0. Esta outra forma de representação da função booleana é um produto de somas, e é a segunda forma canônica ou forma conjuntiva (no inglês *conjunctive normal form* ou CNF) (ZUFFO, 1976). A notação para um produto de maxtermos é, considerando por exemplo um conjunto contendo os maxtermos 0, 2, 3, e 7 de uma função CNF com três variáveis, $\prod_{X,Y,Z}(0,2,3,7)$ (WAKERLY, 2000).

2.2 Minimização de funções lógicas

A minimização de funções lógicas permite diminuir o tamanho da área ocupada por um CI através da redução da quantidade de portas lógicas necessárias para a concepção da função lógica em um circuito eletrônico, diminuindo também o consumo de energia do circuito, sem alteração no resultado comportamental do CI (NELSON; NAGLE, 1995). Para isso podem ser usados alguns métodos, como o emprego de propriedades algébricas booleanas, utilização do método criado por Karnaugh (1953), conhecido por Mapas de Karnaugh, e a utilização do algoritmo de Quine-McCluskey (SCOTT, 1963).

A álgebra booleana compreende propriedades associativas, comutativas, absortivas, distributivas, elementos neutros e complementares, e permite a minimização de funções lógicas a partir da aplicação destas propriedades. O algoritmo de Quine-McCluskey, também chamado de Métodos dos Implicantes Primos, é geralmente usado para minimizar expressões com mais de 4 variáveis por ser mais prático que o método dos Mapas de Karnaugh (NELSON; NAGLE, 1995).

Segundo Zuffo (1976), o método dos Mapas de Karnaugh consiste em mapear funções booleanas em um sistema de coordenadas binárias. Cada elemento do mapa é posicionado respeitando a menor distância de Hamming entre eles, ou seja, de forma que elementos que diferem apenas pelo valor de uma variável fiquem vizinhos. Desta forma, levando em consideração, por exemplo, uma função com quatro variáveis (A, B, C e D), a disposição dos elementos relativos às variáveis A e B podem ficar dispostos nas colunas como 00, 01, 11 e 10, fazendo referência a estas variáveis em ordem, assim como os elementos relativos às variáveis C e D ficam dispostos nas linhas como 00, 01, 11 e 10, também fazendo referência às variáveis em ordem. A Figura 1 ilustra o exemplo explanado neste parágrafo.

Figura 1 – *Mapa de Karnaugh para* $f_{(a,b,c,d)}$

cd \ ab	00	01	11	10
00	$f_{(0,0,0,0)}$	$f_{(0,1,0,0)}$	$f_{(1,1,0,0)}$	$f_{(1,0,0,0)}$
01	$f_{(0,0,0,1)}$	$f_{(0,1,0,1)}$	$f_{(1,1,0,1)}$	$f_{(1,0,0,1)}$
11	$f_{(0,0,1,1)}$	$f_{(0,1,1,1)}$	$f_{(1,1,1,1)}$	$f_{(1,0,1,1)}$
10	$f_{(0,0,1,0)}$	$f_{(0,1,1,0)}$	$f_{(1,1,1,0)}$	$f_{(1,0,1,0)}$

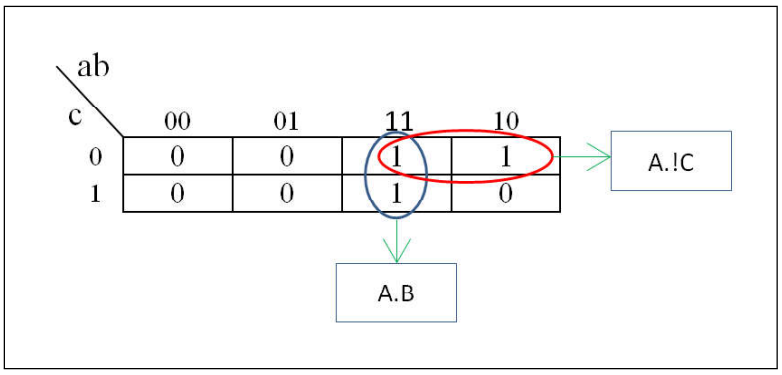
Fonte: Elaborado pelo Autor

Mapear uma função utilizando Mapas de Karnaugh consistem em colocar os valores de saída da tabela verdade em seus respectivos espaços no mapa. Em

seguida, para obter uma função simplificada é necessário agrupar os campos vizinhos com valor 1 em cubos de 2^n elementos, buscando sempre maximizar o tamanho do cubo. De cada cubo se obtém uma expressão simplificada de produto de variáveis e é somada às outras expressões formando um somatório de produtos.

O método para obtenção da expressão simplificada de cada cubo de 2^n elementos pode ser explicado utilizando a Figura 2, conforme segue: no cubo dos elementos 110 e 100, A repete seu sinal nos dois elementos e tem o valor de 1, por isso deve ser usado na expressão simplificada, B muda seu sinal e por isso não é usado na expressão simplificada, C tem apenas um valor, o de 0, e como não mudou seu sinal é usado para integrar a expressão simplificada, mas como seu sinal é 0 será representado pelo seu complemento e a expressão simplificada para o cubo dos elementos 110 e 100 é A.!C. No cubo dos elementos 110 e 111, tanto A quanto B mantém o valor do seu sinal, que é 1, e por isso são usados na expressão simplificada sendo representados em sua fase positiva, C muda seu sinal e por isso não é colocado na expressão simplificada e a expressão simplificada para o cubo dos elementos 110 e 111 é A.B. Com as expressões simplificadas de cada cubo de 2^n elementos do Mapa de Karnaugh se monta a função mapeada no mapa somando-se as expressões, que para a função apresentada na Figura 2 é A.!C + A.B.

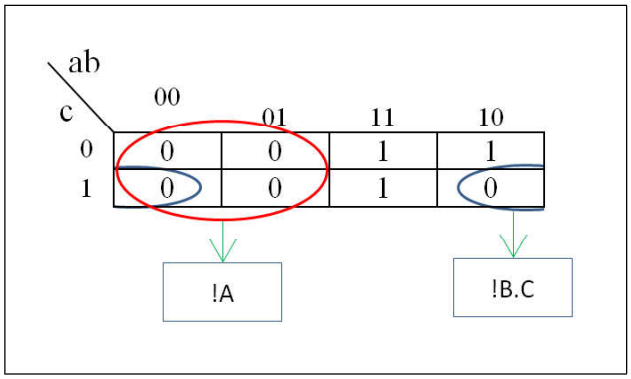
Figura 2 – *Mapa de Karnaugh para obtenção da função simplificada*



Fonte: Elaborado pelo Autor

Para a obtenção da expressão minimizada do complemento de uma função, ao invés de utilizar os valores de saída 1 da tabela verdade, são utilizados os valores de saída 0. Para a função mapeada na Figura 3, seu complemento a partir da soma de produtos dos mintermos da tabela verdade é !A.!B.!C+!A.!B.C+!A.B.!C+!A.B.C+A.!B.C, e a minimização utilizando Mapa de Karnaugh é !A+!B.C.

Figura 3 – *Mapa de Karnaugh para obtenção da função simplificada complementar*



Fonte: Elaborado pelo Autor

2.3 *Design for Testability*

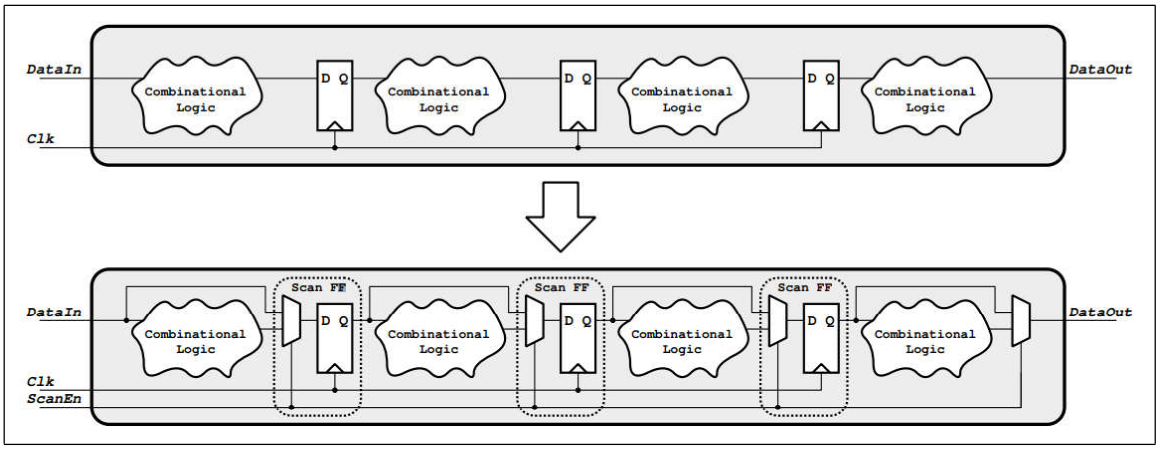
Design for Testability (DFT) trata de técnicas para a inserção de modificações nos circuitos com o intuito de melhorar, ou permitir, a testabilidade dos mesmos. Nesta seção abordaremos as técnicas *scan chain*, pontos de teste e BIST (*built-in self-test*).

Scan chains é uma forma de controlar e observar o estado de circuitos sequenciais em um CI. Segundo Wang, Chang e Cheng (2009), é uma das técnicas de DFT mais significantes, pois seu processo de implementação é automatizado e facilmente inserido no desenvolvimento de CIs por ferramentas EDA (*Electronic Design Automation*).

Scan chains podem ser implementados em circuitos com elementos de retenção de estado, ou registradores, como *flip-flops* e *latchs*. Esta técnica trata de prover uma entrada ao circuito para ativar ou desativar o modo de teste, que controla multiplexadores de duas entradas e uma saída (MX2X1) para contornar, quando no modo de teste, os circuitos combinacionais que fornecem as entradas de dados aos registradores. Quando em modo de teste, os registradores ficam ligados em configuração *shift register* e é setado um estado inicial para eles. Em seguida é mudado para o estado normal de operação do circuito e o padrão de teste presente nos registradores é aplicado. A seguir volta para o estado de teste, é propagado para a saída o estado final presente nos registradores e é configurado o estado inicial para o próximo teste (WILLIAMS; ANGELL, 1973). A Figura 4 mostra um exemplo da aplicação de *scan chain*.

Outra técnica de DFT, que melhora a cobertura de testes de um circuito, é a inserção de pontos de teste. Esta técnica trata de ideias no sentido de reduzir a complexidade da tarefa de teste. Basicamente, se refere a colocar registradores em determinadas posições de baixa observabilidade do CI, para registrar o sinal

Figura 4 – *Scan chain*



Fonte: (KAESLIN; FELBER, 2016)

destas posições, assim como colocar registradores para setar sinais, a partir da seleção do sinal por um multiplexador (MX2X1), em determinadas posições de baixa controlabilidade do CI (WESTE; HARRIS, 2015).

A técnica BIST é um mecanismo que permite que o CI tenha como testar partes do seu próprio circuito (JHA; GUPTA, 2003). Não há uma metodologia padrão para descrever o teste BIST, que pode ser um procedimento para teste de memória que automaticamente verifica se todas as posições de memória podem ser gravas e lidas corretamente, por exemplo. Há diversos testes que podem ser classificados como BIST e cada teste é desenvolvido de acordo com as características do CI em análise.

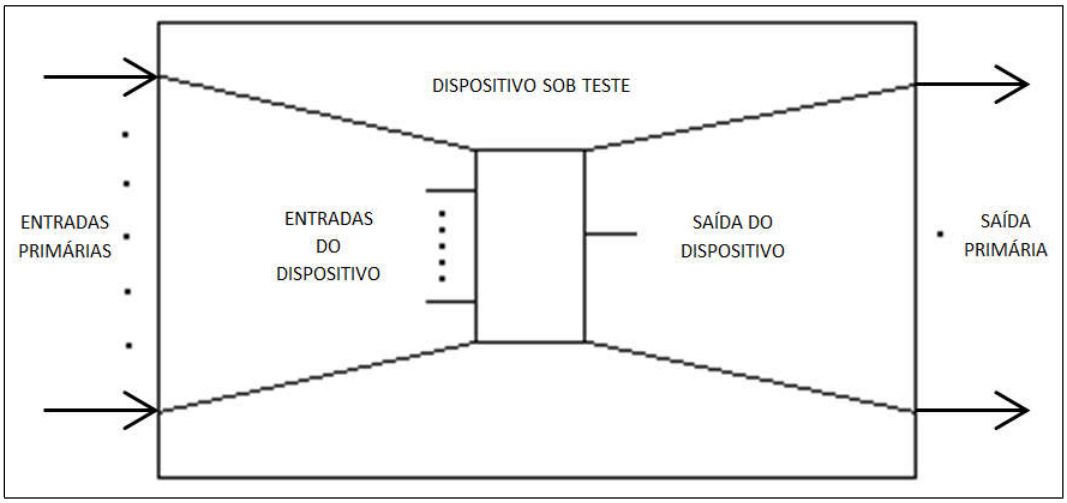
A importância da controlabilidade e da observabilidade para o DFT reside na questão de que algumas decisões de projeto tem implicação na testabilidade, e o teste de cobertura de falhas pode ser melhorado com mudanças no projeto. Portanto, é importante avaliar a testabilidade no início do projeto e, se necessário, fazer mudanças topológicas para melhorar, ou viabilizar, o processo de testes (KAESLIN; FELBER, 2016).

2.4 Métodos para obtenção da testabilidade

Há diferentes definições de testabilidade, mas basicamente ela refere-se a custo, como o custo de gerar padrões de teste para atender a certos critérios e o custo de assegurar um correto diagnóstico. Segundo Kantipudi (2010), a testabilidade de um CI é definida pela sua controlabilidade e observabilidade. A controlabilidade para um CI é definida como a dificuldade em estabelecer um sinal lógico particular para 0 ou 1. A observabilidade de um CI é definida como a dificuldade em observar o estado de um sinal lógico. A Figura 5 mostra a abstração de um dispositivo sob teste,

as entradas primárias e a saída primária. Um nó é controlado a partir da entrada primária de um dispositivo sob teste e observado a partir da saída primária. Desta forma, medidas de testabilidade baseadas em controlabilidade e observabilidade são medidas do quão fácil, ou difícil, é gerar padrões de teste (BENNETTS, 1982).

Figura 5 – *Dispositivo sob teste*



Fonte: adaptado de Bennetts (1982)

Para Bennetts (1982) as métricas de testabilidade podem ser usadas como uma ajuda interativa no projeto da topologia de circuitos, e cita algum usos para elas:

- Informa qual a melhor dentre duas topologias de circuito
- Permite uma escolha mais sensata de pontos de teste
- Identifica nós potencialmente difíceis de testar (baixa controlabilidade e observabilidade)

Segundo Bushnell e Agrawal (2004), as noções de controlabilidade e observabilidade de sinais em um circuito originaram-se da teoria do controle automático. Conforme Wang, Chang e Cheng (2009), a análise de testabilidade de um circuito também é importante para a tomada de decisão durante a geração de vetores de teste em ATPGs.

Segundo Bennetts, Maunder e Robinson (1981), basicamente há duas formas de quantificar a testabilidade do design de um circuito digital: os métodos baseados em pontuação e os baseados em algoritmo. Os métodos baseados em pontuação fundamentam-se na identificação de características dos circuitos que contribuem ou prejudicam a testabilidade. Para cada característica é dada uma pontuação

que representa a magnitude dos efeitos que sua presença provoca na testabilidade. Exemplos de métricas de testabilidade baseadas em pontuação são dados por Dejka (DEJKA, 1977), Wood (WOOD, 1979) e Danner e Consolla (DANNER; CONSOLLA, 1979).

Métodos baseados em pontuação são fáceis de implementar, permitem a comparação entre diferentes circuitos e não requerem o auxílio de computação, no entanto não são adequados para o auxílio ao projeto de circuitos digitais pois não permitem estabelecer a causa precisa da baixa testabilidade de um circuito (BENNETTS; MAUNDER; ROBINSON, 1981).

Métodos baseados em algoritmo, por outro lado, estimam a testabilidade de um circuito digital a partir da análise de sua descrição topológica. Pela quantificação da testabilidade de cada nó do circuito, consegue expor áreas em que a testabilidade seja mais difícil (BENNETTS; MAUNDER; ROBINSON, 1981).

Um dos métodos para quantificação da testabilidade baseado em algoritmo mais estudado é o SCOAP (*Sandia Controllability/Observability Analysis Program*), e suas variantes (KANTIPUDI, 2010). Apesar de algumas limitações relacionadas a análise ser feita apenas sobre a descrição estrutural do circuito, e também de algumas simplificações para garantir a complexidade linear do algoritmo, é justamente sua complexidade algorítmica que garante sua popularidade.

Outro método de testabilidade baseado em algoritmo é o CAMELOT (*computer-aided measure for logic testability*). O CAMELOT gera valores numéricos que medem a facilidade com que um determinado nó de um circuito pode ser ajustado para um nível lógico desejado e a facilidade com que seu estado pode ser observado (BENNETTS; MAUNDER; ROBINSON, 1981).

2.4.1 SCOAP

Em Goldstein (1979) é apresentado um método para analisar circuitos digitais em termos de seis funções que caracterizam controlabilidade e observabilidade para circuitos combinacionais e sequenciais. O método permite o cálculo de uma medida quantitativa de dificuldade em controlar e observar os valores lógicos dos sinais em todos os nós de interligação entre portas lógicas da topologia do circuito. No ano seguinte, Goldstein e Thigpen (1980) descrevem a implementação do software SCOAP, desenvolvido no *Sandia National Laboratories*, para a análise da testabilidade de circuitos digitais e baseado no método criado por Goldstein no ano anterior.

Os seis valores numéricos calculados pelo SCOAP, para um sinal s, são:

- CC0(s): *Combinational 0-controllability* de s

- CC1(s): *Combinational 1-controllability* de s
- CO(s): *Combinational observability* de s
- SC0(s): *Sequential 0-controllability* de s
- SC1(s): *Sequential 1-controllability* de s
- SO(s): *Sequential observability* de s

Como neste trabalho vamos focar em circuitos combinacional, iremos abordar apenas as medidas CC0, CC1 e CO de testabilidade. Segundo Goldstein (1979), os valores de CC0, CC1 e CO podem ser vistos como a função de custo em medir a dificuldade, em um sentido espacial, de um conjunto completo de nós necessários para controlar ou observar um determinado nó do circuito.

O cálculo da testabilidade, segundo Goldstein (1979), segue a seguir. Os valores de CC0 e CC1 variam de 1 até o infinito, e de CO de 0 até o infinito. O cálculo da controlabilidade e da observabilidade se dá para cada porta lógica do circuito. Cada porta lógica tem uma equação associada a sua controlabilidade e a sua observabilidade. As entradas primárias do circuito recebem o valor 1 para CC0 e CC1. De posse destes valores nas entradas primárias são calculados os valores de controlabilidade da saída das portas lógicas ligadas às entradas primárias do circuito, que são usados para o cálculo das portas lógicas subsequentes até alcançar as saídas primárias do circuito.

O cálculo da observabilidade depende dos valores de controlabilidade. É atribuído o valor 0 à observabilidade das saídas primárias do circuito e calculada a observabilidade das entradas de cada porta lógica até chegar às entradas primárias do circuito (GOLDSTEIN, 1979).

Segundo Goldstein (1979), a equação para cálculo da observabilidade de um nó de entrada é definida pela observabilidade da saída sensibilizada, que é mais fácil de observar, mais a soma das controlabilidades de mínimo custo que sensibilizam a entrada, mais a profundidade da célula.

Wang, Chang e Cheng (2009), assim como Wang, Wu e Wen (2006) e Kantipudi (2010), fornecem equações para o cálculo da controlabilidade e da observabilidade para as portas lógicas AND, NAND, OR, NOR, XOR, XNOR, NOT e BUFFER usando o SCOAP, conforme as Tabelas 2 e 3, mas não fornecem uma metodologia para a criação destas equações.

Tabela 2 – Regras para cálculo de controlabilidade combinacional para uso no SCOAP

Portas	CC0	CC1
Entrada primária	1	1
AND	min {CC0 das entradas} + 1	∑(CC1 das entradas) + 1
OR	∑(CC0 das entradas) + 1	min {CC1 das entradas} + 1
NOT	CC1 da entrada + 1	CC0 da entrada + 1
NAND	∑(CC1 das entradas) + 1	min {CC0 das entradas} + 1
NOR	min {CC1 das entradas} + 1	∑(CC0 das entradas) + 1
BUFFER	{CC0 da entrada} + 1	{CC1 da entrada} + 1
XOR	min {CC1(a) + CC1(b), CC0(a) + CC0(b)} + 1	min {CC1(a) + CC0(b), CC0(a) + CC1(b)} + 1
XNOR	min {CC1(a) + CC0(b), CC0(a) + CC1(b)} + 1	min {CC1(a) + CC1(b), CC0(a) + CC0(b)} + 1
Ramo	CC0 da haste	CC1 da haste

Fonte: (WANG; CHANG; CHENG, 2009)

Tabela 3 – Regras para cálculo de observabilidade combinacional para uso no SCOAP

Portas	CO
Saída primária	0
AND/NAND	∑(CO da saída, CC1 das outras entradas) + 1
OR/NOR	∑(CO da saída, CC0 das outras entradas) + 1
NOT/BUFFER	CO da saída + 1
XOR/XNOR	entrada a: ∑(CO da saída, min{CC0(b), CC1(b)}) + 1 entrada b: ∑(CO da saída, min{CC0(a), CC1(a)}) + 1
Haste	*min* {CO dos ramos}

Fonte: (WANG; CHANG; CHENG, 2009)

2.4.2 CAMELOT

No CAMELOT, os valores de controlabilidade são representados por CY e variam de 0 a 1. O valor 0 denota uma impossibilidade em controlar o sinal do nó do circuito para qualquer um dos dois estados lógicos. O valor 1 é o máximo valor de CY que um nó pode ter, e implica na máxima facilidade em estabelecer um valor lógico ao nó. Um exemplo de CY com valor 1 são para as entradas primárias do circuito (BENNETTS; MAUNDER; ROBINSON, 1981).

A equação da CY é dada por:

$$CY(\text{nó de saída}) = CTF \text{ x } f\{CY\text{s (nós de entrada)}\}$$

onde CTF (*controllability transfer factor*) é o fator de transferência de controlabilidade do dispositivo cuja saída se busca a CY, e a função f é a média aritmética

das *CY*s das entradas (para circuitos combinacionais) (BENNETTS; MAUNDER; ROBINSON, 1981).

O *CTF* é uma medida relativa à função lógica do dispositivo para o qual se está calculando a *CY* e é dado pela seguinte equação:

$$CTF = 1 - |\frac{N_{(0)} - N_{(1)}}{N_{(0)} + N_{(1)}}|$$

onde $N(0)$ é o número de sinais 0 na saída da tabela verdade do dispositivo e $N(1)$ é o respectivo número de sinais 1.

Já a observabilidade no CAMELOT, segundo Bennetts, Maunder e Robinson (1981), é definida como a medida da facilidade em observar o estado das saídas primárias do circuito. É representada por *OY*. Os valores de *OY* variam de 0 a 1, sendo 1 a *OY* das saídas primárias, assim como a *OY* de um nó do circuito medida no próprio nó.

Segundo Bennetts (1984) a equação da *OY* é dada por:

$$OY(\text{entrada->saída}) = OY(\text{entrada->entrada}) \text{ x } OTF \text{ x } g\{CYs \text{ (entradas de apoio)}\}$$

onde OY(entrada->entrada) é a observabilidade da entrada primária do dispositivo medido na própria entrada primária, cujo valor é 1, o *OTF* (*observability transfer factor*) é um fator que reflete a facilidade em propagar uma mudança de valor de uma entrada específica do dispositivo para a saída, e a função g é a média aritmética das *CY*s das entradas de apoio. As entradas de apoio são todas as entradas primárias do dispositivo menos a entrada para a qual se está calculando a *OY*.

A equação do *OTF* é dada por:

$$OTF = \frac{N(SP)}{N(SP)+N(IP)}$$

onde *N(SP)* é o número de caminhos sensíveis distintos da entrada para a saída e *N(IP)* é o número de caminhos insensíveis da entrada para a saída. *SP* (*sensitive path*) é o caminho percorrido pelo sinal no circuito, de uma entrada até uma saída primária. É possível identificar um *SP* para uma determinada entrada se a mudança no sinal desta entrada analisada seja a única a acarretar em uma mudança no sinal da saída. Caso esta mudança no sinal da entrada analisada não acarrete em uma mudança no sinal da saída, temos um caminho insensível (*IP*).

A equação para o cálculo da *OY* proposta por Bennetts (1984) não possui uma referência do valor de observabilidade do nodo anterior, gerando apenas o valor

do fator de transferência de dificuldade de propagação da observabilidade entre um nodo e outro, não refletindo o aumento da dificuldade em observar um sinal em diferentes pontos do circuito. Por este motivo, foi entendido que para Bennetts (1984), ao trabalhar com a observabilidade de um sinal, a entrada a qual ele se refere é a saída da porta lógica para a qual se está calculando a observabilidade, visto que o cálculo da observabilidade é feito a partir das saídas primárias de um circuíto, indo em direção às entradas primárias. Portanto, quando Bennetts (1984) faz referência à OY(entrada->entrada) deve ser usado o valor da observabilidade já calculada no nodo anterior, pois esta entrada da equação de observabilidade corresponde à saída da porta lógica para a qual se está calculando a observabilidade. Ao adotar esta estratégia obteve-se resultados para os valores de observabilidade condizentes com o aumento da dificuldade em observar um sinal de um CI a medida que se percorre sua topologia. Quanto à média aritmética das CY das entradas de apoio, estas são relativas às entradas da porta lógica para a qual se está calculando a observabilidade. O *software* para o cálculo da OY foi desenvolvido seguindo esta metodologia explanada no presente parágrafo.

Para o cálculo da observabilidade de hastes com dois ou mais ramos, como mostrado na Figura 6, a equação para o cálculo da observabilidade da haste é dada por:

$$Oy(\text{a}) = 1 - [1 - Oy(\text{a - S1})] \text{ x } [1 - Oy(\text{a - S2})] \text{ x } [1 - Oy(\text{a - Sn})]$$

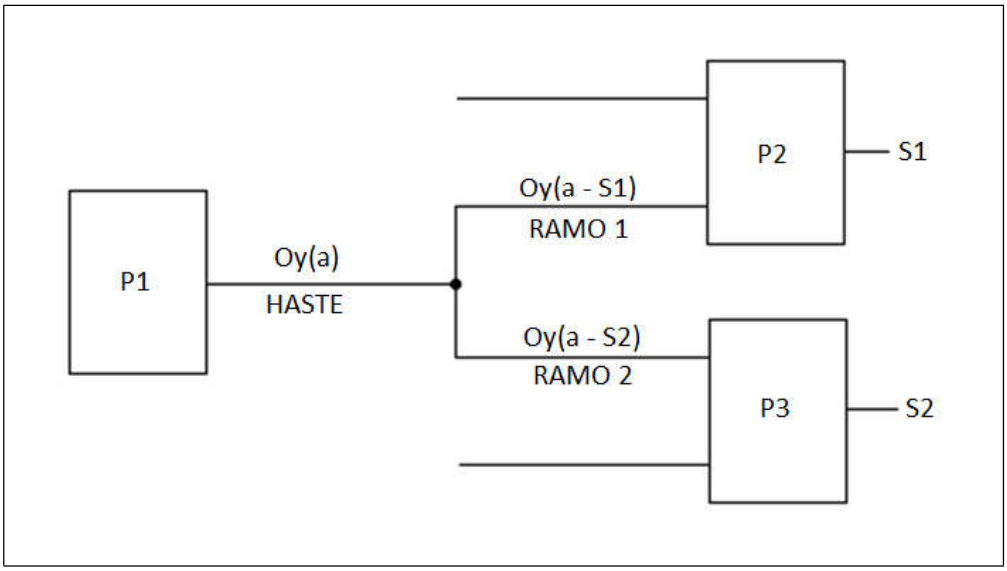
onde Oy(a) é a observabilidade para a haste *a*, Oy(a - S1) é a observabilidade no primeiro ramo, Oy(a - S2) é a observabilidade no segundo ramo, e Oy(a - Sn) é a observabilidade nos ramos subsequentes até o ramo n.

2.5 Aplicações das métricas

A crescente complexidade dos circuitos integrados requer constante melhoria na área de diagnóstico de falhas. Jamil e Mohammed (2015) aborda o tema ao propor um software de análise de circuitos com o algoritmo VICTOR, capaz de analisar a controlabilidade e a observabilidade de um circuito para gerar vetores de teste. Kantipudi (2010) também aborda o tema dissertando sobre diversos métodos para obtenção de métricas de testabilidade, e mostra que a automação da geração das métricas de testabilidade é importante na criação de circuitos voltados à testabilidade (DFT).

Tille et al. (2009) usa métricas de testabilidade na ajuda da tomada de decisão da heurística do algoritmo de satisfabilidade usado para a criação de um software ATPG. Karimi, Riyahi e Navabi (2003) aborda a testabilidade nos níveis de

Figura 6 – *Cálculo da observabilidade de hastes com ramos para o CAMELOT*



Fonte: Elaborado pelo Autor

abstração de portas lógicas, RTL e comportamental, e conclui que análises a nível de portas lógica são precisas mas demandam tempo, e análises a nível RTL e a nível comportamental são menos precisas.

Ainda que muitos dos artigos abordando métricas de testabilidade sejam focados em teste, nos últimos anos estão surgindo diversos artigos abordando esta temática na área de segurança. Como Rajski et al. (2017), que discorre sobre o tema *Design-for-Security* e propõem o uso dos pontos de teste presentes no CI para a procura de circuitos misturados (*scrambled*). Métricas de testabilidade, como o SCOAP, são usadas para identificar os melhores pontos de teste, evitando pontos de difícil controle e observabilidade e, por consequência, evitar simulações demoradas.

Salmani e Tehranipoor (2016) busca identificar possíveis *hardware trojan* maliciosos, inseridos por fundições não confiáveis, a partir das métricas de testabilidade. Assim como Xie et al. (2017), que propõem um método de detecção de *hardware* malicioso a partir de uma análise estática do *netlist* no nível de portas lógicas, baseada na controlabilidade e observabilidade do circuito.

Ainda há muito a explorar quanto às métricas de testabilidade na área de segurança de CIs. Um exemplo é a recente obra de Salmani (2018), na qual várias métricas são discutidas para quantificar as vulnerabilidades dos circuitos à *trojans* de *hardware* em diferentes níveis. Salmani (2017) também aborda o tema à procura de hardware malicioso usando métricas de testabilidade, analisando *netlists* a nível de portas lógicas.

No entanto há outras abordagens para as métricas de testabilidade, como

Mehta e Dhare (2017) que discorre sobre uma futura era "Além do CMOS" (*transistor less computation paradigm*), em que a análise de defeitos e métodos de teste poderão se beneficiar com o legado das técnicas de testabilidade, o que justifica o contínuo estudo destas técnicas.

3 Métrica do SCOAP para qualquer função

Portas lógicas podem ser criadas para assumir qualquer função lógica. Isso cria a necessidade de métricas de testabilidade aptas a trabalhar com qualquer função lógica. O SCOAP não possuía uma solução genérica para a obtenção das equações para o cálculo das métricas de testabilidade de qualquer função lógica. Em função disso se desenvolveu este trabalho, buscando uma metodologia para a geração das equações para testabilidade de qualquer função lógica para o método SCOAP. Neste capítulo é apresentada a explicação do método proposto de criação das equações de controlabilidade e de observabilidade para uma função lógica qualquer usando o SCOAP.

3.1 Controlabilidade para o SCOAP

Para a condução deste trabalho, é necessário determinar alguns conceitos. A soma dos mintermos cuja função lógica tenha saída 1 (DNF), assim como o produto dos maxtermos cuja função lógica tenha saída 0 (CNF), serão representados pela sigla *F1*. A soma dos mintermos cuja função lógica tenha saída 0, assim como o produto dos maxtermos cuja função lógica tenha saída 1, serão representados pela sigla *F0*. A tabela verdade gerada pela função lógica criada a partir do *F0* é exatamente complementar à gerada pela função lógica criada a partir do *F1*.

A criação das equações para o cálculo da controlabilidade para o SCOAP levou em consideração a análise das equações propostas por Wang, Chang e Cheng (2009). Tendo em consideração que a controlabilidade é definida como a dificuldade em estabelecer um sinal lógico particular para um 1 ou um 0, observou-se que as equações para cálculo da controlabilidade de 1 tem relação com o DNF de uma função lógica (*F1*), e a controlabilidade de 0 tem relação com o DNF complementar desta função lógica (*F0*).

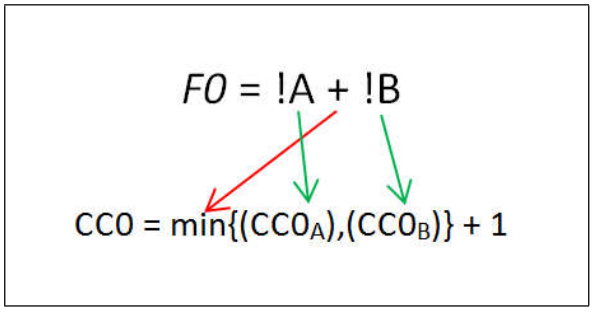
Observou-se que uma soma (disjunção) em uma equação booleana é relacionada nas equações de controlabilidade pelo mínimo valor de controlabilidade entre os sinais representados pelas variáveis, ou termos normais, que são somados na equação. Quanto à multiplicação (conjunção) em uma equação booleana, observou-se que nas equações de controlabilidade ocorre a soma de cada uma das controlabilidades dos sinais representados pelas variáveis ou termos normais da multiplicação da equação

booleana. As equações de controlabilidade não comportam o formato de termos normais das equações booleanas, portanto todos os termos normais, quando houverem, devem ser transformados para o formato das equações de controlabilidade.

Outra característica observada é em relação às variáveis dos termos normais do *F1* e do *F0*. Se a variável estiver na fase negativa, é representada por CC0 na equação de controlabilidade, e se estiver na fase positiva é representada por CC1.

Por exemplo, para uma porta lógica AND, cuja equação booleana é A.B, Wang, Chang e Cheng (2009) propõem que a CC0 do seu sinal de saída seja o menor valor das CC0 dos sinais de entrada mais 1. O valor 1 somado é relativo ao incremento do nível. O *F0* da equação booleana A.B é !A+!B, obtido através do Mapa de Karnaugh. Como há uma soma entre as variáveis A e B, a equação de controlabilidade deve tratar do mínimo, e como no *F0* ambas as variáveis estão na fase negativa, é o mínimo entre as CC0 de A e de B, mais 1. Este exemplo é mostrado na Figura 7.

Figura 7 – *Método para criação da CC0*



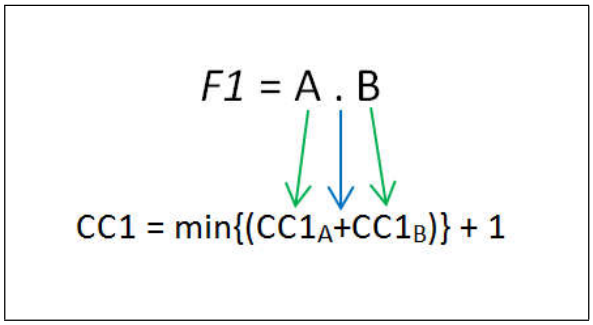
Fonte: Elaborado pelo Autor

Já para a CC1 do sinal de saída de uma porta lógica AND, Wang, Chang e Cheng (2009) propõem que seja o somatório das CC1 dos sinais de entrada da porta lógica, mais 1. O *F1* da função A.B é A.B. Como há uma multiplicação entre as variáveis A e B, a equação da controlabilidade deve tratar de uma soma das controlabilidades dos sinais representados pelas variáveis, e como as variáveis estão na sua fase positiva, é uma soma das CC1 de cada sinal representado pelas variáveis, mais 1. Este exemplo é mostrado na Figura 8.

3.2 Observabilidade para o SCOAP

Quando Goldstein (1979) se refere à soma das controlabilidades de mínimo custo que sensibilizam a entrada, isso significa identificar todas as entradas da tabela verdade referentes à função analisada em que a variável que representa o sinal ao qual se está calculando a observabilidade é o único sinal responsável por mudar o

Figura 8 – *Método para criação da CC1*

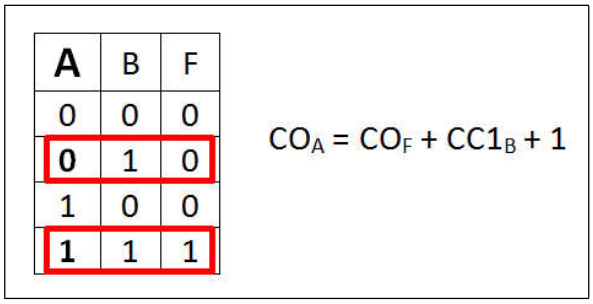


Fonte: Elaborado pelo Autor

valor de saída, somar as controlabilidades dos sinais representados pelas variáveis que permitem que o sinal para o qual se está calculando a observabilidade controle a saída e, por último, escolher a soma cujo resultado seja o menor. De posse deste valor, somar à observabilidade da saída sensibilizada, mais 1 para a profundidade da célula.

Para exemplificar este conceito, conforme mostra a Figura 9, o cálculo da observabilidade da entrada A de uma porta lógica AND, cuja função booleana é A.B, é dada pelo valor da controlabilidade da saída sensibilizada, mais o valor mínimo das controlabilidades que sensibilizam a entrada, que no caso é apenas o sinal representado pela variável B e com o valor 1, portanto $CC1_B$, pois quando B é 1, A pode controlar a saída ao alternar seu valor entre 0 e 1, somado de 1 para a profundidade da célula. Esta equação é a mesma disponível em Wang, Chang e Cheng (2009).

Figura 9 – *Método para a criação da equação de CO de uma função A.B*



Fonte: Elaborado pelo Autor

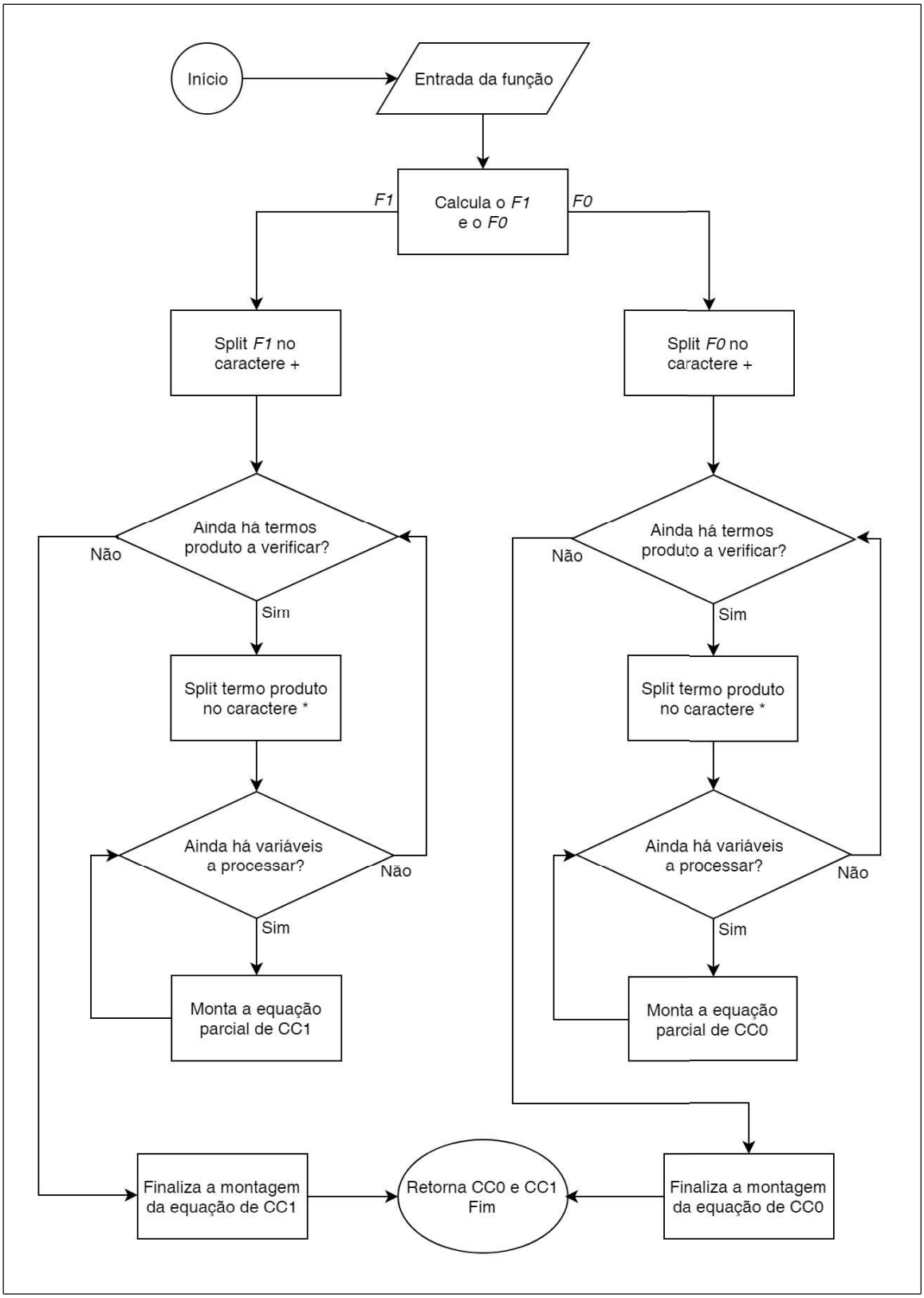
4 Implementação dos métodos SCOAP e CAMELOT

4.1 SCOAP

A Figura 10 mostra o fluxograma do software desenvolvido para a criação das equações de controlabilidade, de acordo com os detalhes apresentados no capítulo anterior. As ações especificadas no fluxograma são explicadas a seguir.

- Calcula o *F1* e o *F0* : Gera a tabela verdade da função e a processa para obtenção do *F1* a partir do método Quine-McCluskey. Para obter o *F0* da função, inverte as saídas da tabela verdade, a submete ao método Quine-McCluskey e a função resultante é o *F0* da função para a qual se está calculando o CC0.
- Split *F1* no caractere + : Separa o *F1* em seus termo produtos.
- Split *F0* no caractere + : Separa o *F0* em seus termo produtos.
- Split termo produto no caractere * : Separa o termo produto nas suas variáveis.
- Monta a equação parcial de CC1 : A equação parcial de CC1 da saída é montada para cada termo produto da função de *F1*. Para cada variável do termo produto, caso esteja precedida pelo símbolo de negação, é representada na equação parcial pelo valor de CC0 desta variável de entrada, e caso a variável do termo produto seja seu valor direto, é representada na equação parcial pelo valor de CC1 desta variável de entrada. O conectivo usado entre cada parcela de um termo produto é o caractere *, e o conectivo usado entre cada parcela da equação parcial é o caractere +. Com isso, o valor calculado com a equação parcial é um número inteiro, sendo a soma das CC0 e/ou CC1 das variáveis de entrada, dependendo das variáveis do termo produto que deu origem à equação parcial da CC1 da saída.
- Monta a equação parcial de CC0 : A equação parcial de CC0 da saída é montada para cada termo produto da função de *F0*. Para cada variável do termo produto, caso esteja precedida pelo símbolo de negação, é representada na equação parcial pelo valor de CC0 desta variável de entrada, e caso a variável do termo produto seja seu valor direto, é representada na equação parcial pelo valor de CC1 desta variável de entrada. O conectivo usado entre cada parcela de um termo produto é o caractere *, e o conectivo usado entre cada parcela

Figura 10 – *Fluxograma para criação das equações de controlabilidade para o SCOAP*



Fonte: Elaborado pelo Autor

da equação parcial é o caractere +. Com isso, o valor calculado com a equação parcial é um número inteiro, sendo a soma das CC0 e/ou CC1 das variáveis de entrada, dependendo das variáveis do termo produto que deu origem à equação parcial da CC0 da saída.

- Finaliza a montagem da equação de CC1 : O resultado da equação final de CC1 utiliza apenas o menor valor inteiro obtido dentre cada uma das suas equações parciais, somada do número inteiro 1 que representa o nível da porta lógica para a qual se está calculando o CC1 de sua saída. Por isso é necessário representar na equação final de CC1 cada equação parcial e usar apenas o resultado da que obtiver o menor valor inteiro para somar ao valor do nível.

- Finaliza a montagem da equação de CC0 : O resultado da equação final de CC0 utiliza apenas o menor valor inteiro obtido dentre cada uma das suas equações parciais, somada do número inteiro 1 que representa o nível da porta lógica para a qual se está calculando o CC0 de sua saída. Por isso é necessário representar na equação final de CC0 cada equação parcial e usar apenas o resultado da que obtiver o menor valor inteiro para somar ao valor do nível.

A Figura 11 mostra o fluxograma do software desenvolvido para a criação das equações de observabilidade com base nas regras propostas no capítulo anterior. As ações especificadas no fluxograma são explicadas a seguir.

- Gera tabela verdade: Recebe a função e gera um vetor, onde a primeira posição possui os nomes das variáveis separados por vírgula e as demais posições todas as combinações possíveis de entrada para as variáveis e seu respectivo resultado lógico para a função dada.

- Verifica se a variável em processamento é a única a sensibilizar a saída : Compara a linha da tabela a verificar com outra linha da tabela a procura de alguma linha em que há mudança no valor da variável em processamento, as outras variáveis não sofrem mudança no seu valor, há mudança no sinal do valor de saída da tabela verdade e esta linha ainda não foi processada em uma comparação anterior.

- Montagem parcial da equação: Armazena em um vetor o nome das variáveis que não estão sendo processadas no momento, com seus respectivos sinais correspondentes, para montagem do trecho de equação referente à soma das controlabilidades que permitem que a variável em processamento seja a única a sensibilizar a saída. A equação parcial é criada observando-se o sinal lógico de entrada da tabela verdade para a variável presente no vetor de variáveis não

processadas, sendo este valor lógico 0, é usado o valor da controlabilidade 0 desta variável, e caso o valor lógico seja 1 é usado o valor de controlabilidade 1 desta variável. O conectivo entre estes valores de controlabilidade 0 e/ou controlabilidade 1 é a soma. Com isso o valor numérico de uma equação parcial é um número inteiro gerado a partir da soma das controlabilidades 1 e/ou controlabilidades 0 das entradas representadas nestas equações parciais.

- Montagem final da equação: Monta a equação de observabilidade para a variável em processamento. A equação final usa o menor valor numérico inteiro dentre as equações parciais geradas, somado do valor da observabilidade da saída, somado do nível, que é o valor 1. Por isso, na geração da equação final é necessário representar todas as equações parciais, para que seja escolhida apenas o valor da equação parcial que retornar menor valor inteiro quando do cálculo da observabilidade para uma porta específica.

4.2 CAMELOT

O método CAMELOT não necessita da criação de equações para o cálculo de sua controlabilidade e de sua observabilidade como no método SCOAP. O fluxograma do algoritmo desenvolvido para o cálculo da controlabilidade com o método CAMELOT é apresentado na Figura 12. As ações especificadas no fluxograma são explicadas a seguir.

- Gera tabela verdade: Recebe a função e gera um vetor, onde a primeira posição possui os nomes das variáveis separados por vírgula e as demais posições todas as combinações possíveis de entrada para as variáveis e seu respectivo resultado lógico para a função dada.
- Calcula o CTF: O CTF é dado pela diferença de 1 com o módulo da razão da diferença e a soma de N(0) e N(1), onde N(0) é o número de sinais 0 na saída da tabela verdade do dispositivo e N(1) é o número de sinais 1 na saída da tabela verdade do dispositivo.
- Calcula a controlabilidade: o cálculo da controlabilidade combinacional para o método CAMELOT é dado pela multiplicação do CTF pela média aritmética das controlabilidades das entradas.

Já para o cálculo da observabilidade para o método CAMELOT, é necessário calcular o valor do OTF para depois calcular a observabilidade. O fluxograma para geração das equações de observabilidade segundo o método CAMELOT está disponível na Figura 13.As ações especificadas no fluxograma são explicadas a seguir.

- Gera tabela verdade: Recebe a função e gera um vetor, onde a primeira posição possui os nomes das variáveis separados por vírgula e as demais posições todas as combinações possíveis de entrada para as variáveis e seu respectivo resultado lógico para a função dada.

- Incrementa a variável SP de 1: A variável SP (*Sensitive Paths*) armazena o número de caminhos sensíveis distintos da entrada para a saída. Seu valor inicial é 0 e é incrementada de um sempre que é encontrado um novo caminho sensível distinto da entrada para a saída.

- Incrementa a variável IP de 1: A variável IP (*Insensitive Paths*) armazena o número de caminhos insensíveis distintos da entrada para a saída. Seu valor inicial é 0 e é incrementada de um sempre que é encontrado um novo caminho insensível distinto da entrada para a saída.

- Calcula o OTF para a entrada da variável processada: O OTF é dado pela razão entre o número de caminhos sensíveis distintos da entrada para a saída e a soma do número de caminhos sensíveis distintos da entrada para a saída com o número de caminhos insensíveis distintos da entrada para a saída.

- Calcula a observabilidade para a entrada processada: O cálculo da observabilidade é dado pela multiplicação do OTF pela média aritmética das controlabilidades das entradas.

4.3 Ferramenta para análise de circuitos

Em relação à ferramenta para análise de circuitos para processar as métricas de testabilidade de CIs, seu fluxograma está disponível na Figura 14. Esta ferramenta foi usada para calcular a testabilidade de circuitos usando o método SCOAP. O método CAMELOT também usou esta ferramenta, adaptada para as características do CAMELOT, visto que o SCOAP faz uso de equações para processar a testabilidade de circuitos, já o CAMELOT não necessita destas equações. As ações especificadas no fluxograma são explicadas a seguir.

- Gera grafo com a estrutura do circuito: Carrega arquivo com a *netlist* do CI a analisar em uma estrutura de dados de grafo, colocando as portas lógicas nos nós e às interligando com as arestas

- Calcula os níveis das portas lógicas: Atribui nível a cada porta lógica, a partir das entradas

- Calcula a Controlabilidade: Calcula a controlabilidade de cada porta lógica neste nível e salva o valor no objeto desta porta lógica para posterior cálculo da controlabilidade dos próximos níveis, até chegar ao último nível

- Calcula a Observabilidade: Calcula a observabilidade de cada porta lógica neste nível e salva o valor no objeto desta porta lógica para posterior cálculo da observabilidade das portas lógicas dos próximos níveis, até chegar às entradas do circuito representado no grafo

Figura 11 – *Fluxograma para criação das equações de observabilidade para o SCOAP*

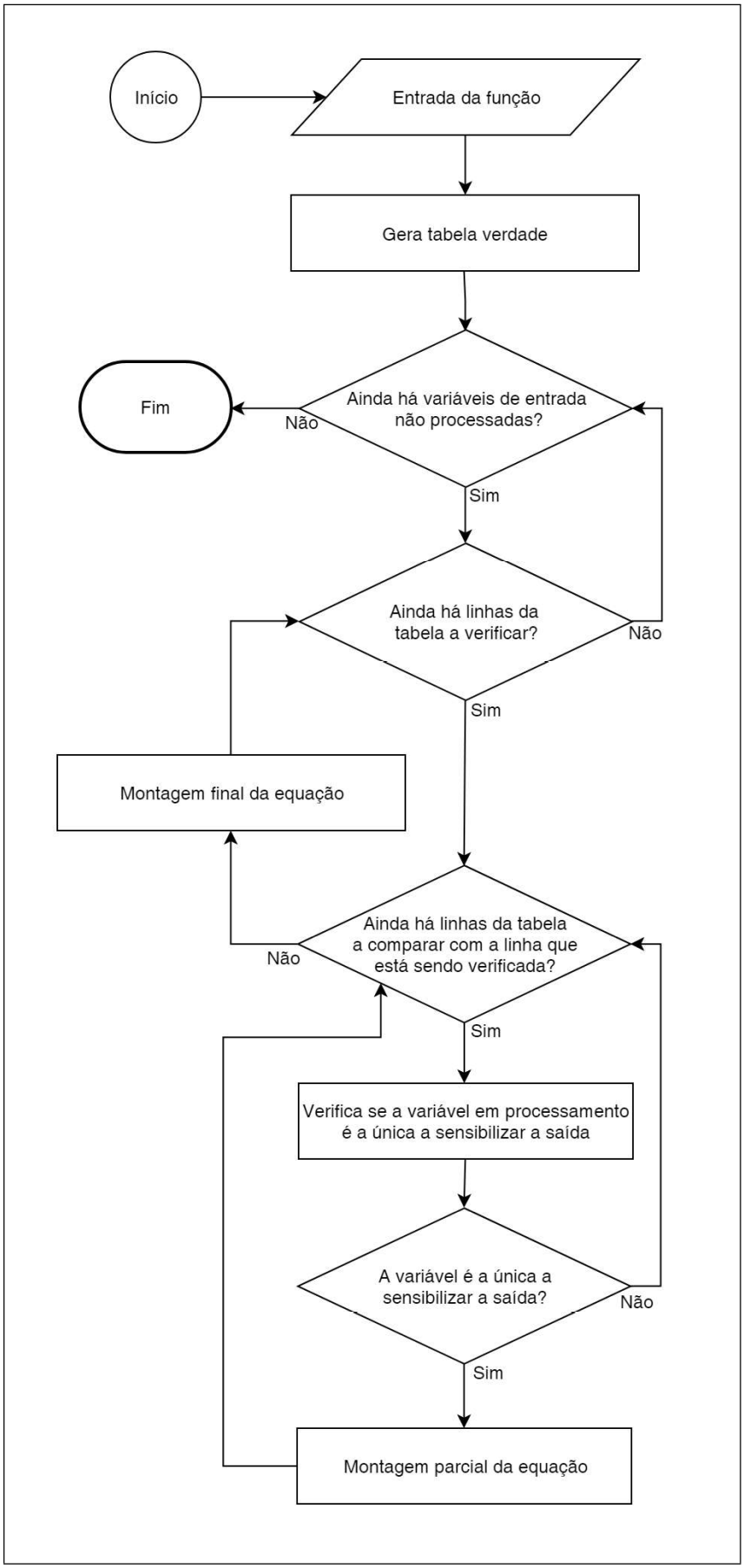
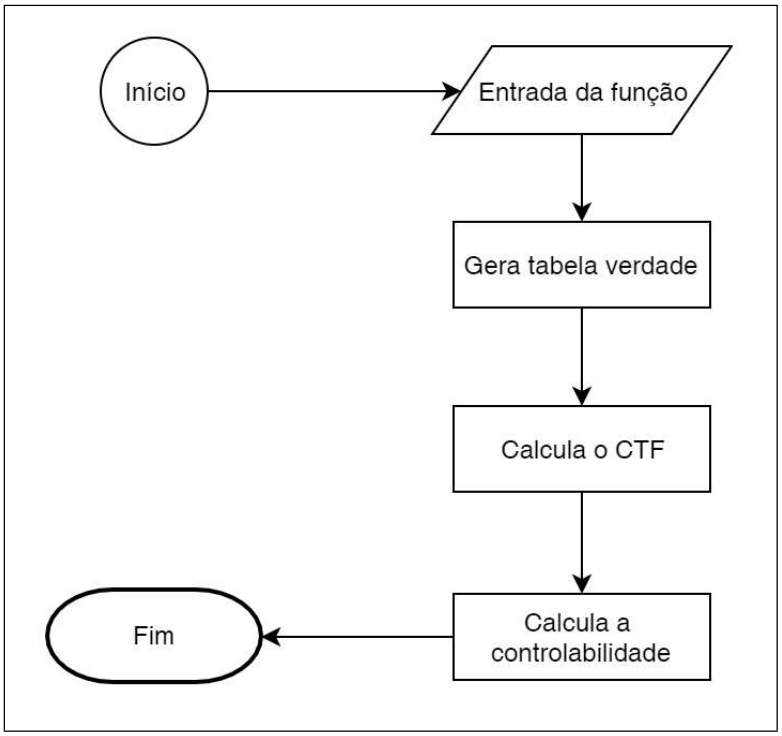
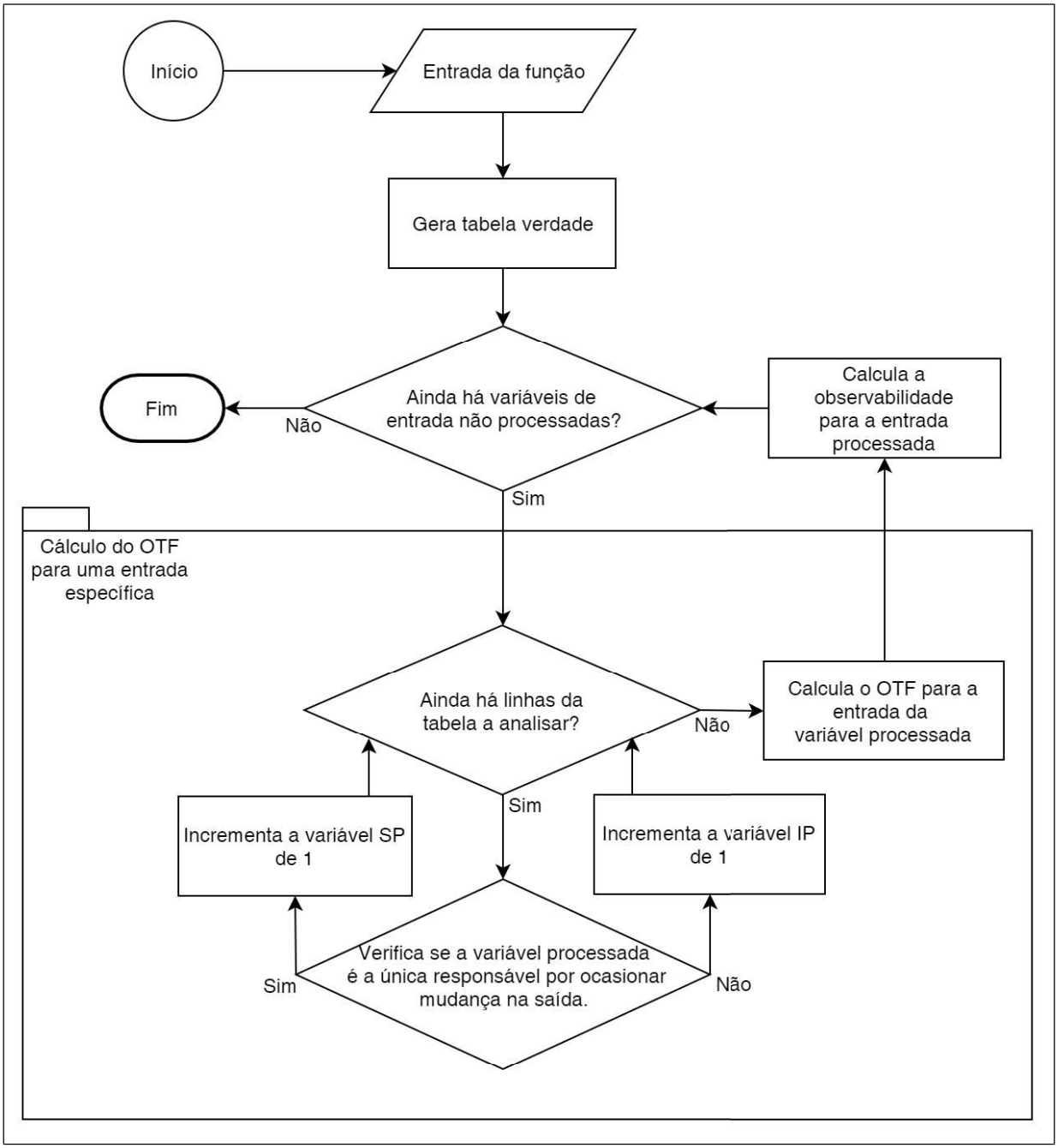


Figura 12 – *Fluxograma para cálculo da controlabilidade para o CAMELOT*



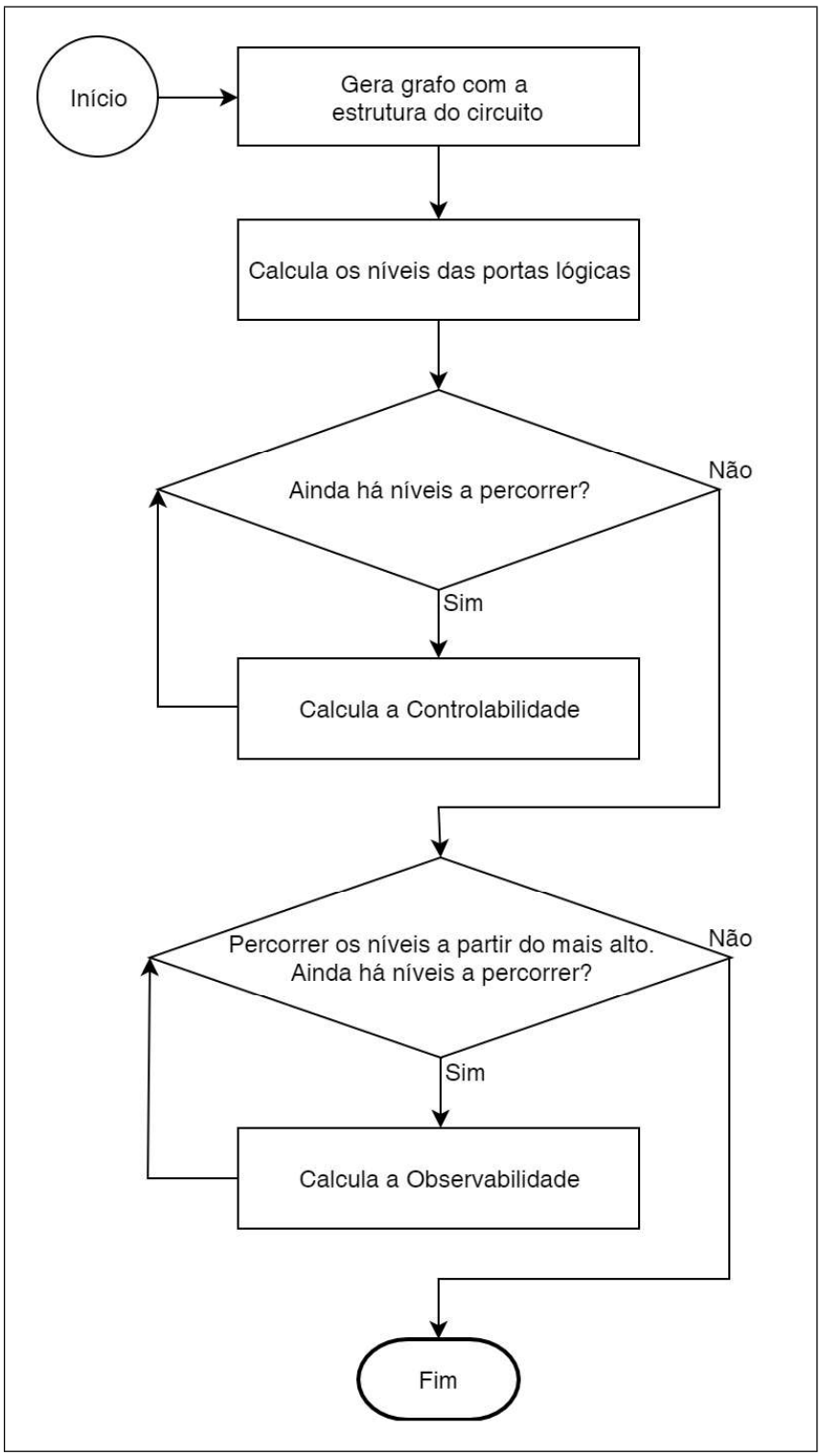
Fonte: Elaborado pelo Autor

Figura 13 – *Fluxograma para o cálculo da observabilidade para o CAMELOT*



Fonte: Elaborado pelo Autor

Figura 14 – *Fluxograma do software de análise de circuitos*



Fonte: Elaborado pelo Autor

5 Resultados

5.1 Gerador das equações SCOAP

A Tabela 4 mostra as funções das portas lógicas que são abordadas nesta sessão. A validação das equações de controlabilidade e de observabilidade geradas pelo software desenvolvido para o método SCOAP foram feitas manualmente a partir do método genérico para obtenção das equações de controlabilidade e de observabilidade para qualquer função desenvolvido neste trabalho. A Tabela 5 mostra as equações de controlabilidade criadas para algumas portas lógicas a partir das regras propostas neste trabalho. A Tabela 6 mostra as equações de observabilidade criadas para algumas portas lógicas a partir das regras propostas neste trabalho. As Tabelas 5 e 6 tem funções genéricas não presentes na proposta original do SCOAP e largamente utilizadas no projeto de CIs atuais.

Tabela 4 – Funções lógicas

Porta	Função Lógica
AND2X1	Y=A*B
OR2X1	Y=A+B
NOT	Y=!A
NAND2X1	Y=!(A*B)
NOR2X1	Y=!(A+B)
XOR2X1	Y=(A*!B)+(!A*B)
XNOR2X1	Y=!((A*!B)+(!A*B))
OAI21X1	Y=!((A0+A1)*B0)
OAI22X1	Y=!((A0+A1)*(B0+B1))
AOI21X1	Y=!(A0*A1+B0)
AOI22X1	Y=!(A0*A1+B0*B1)
MX2X1	Y=(S0*B)+(!S0*A)

Fonte: Elaborado pelo Autor

5.1.1 Análise de escalabilidade para o software gerador das equações SCOAP

O computador usado nas análises de escalabilidade deste trabalho possui processador Intel i5-3320M com dois núcleos de 2,60GHz de velocidade, suporta até quatro *threads* simultâneas, 8Gb de memória RAM e sistema operacional Windows 7 de 64 bits. O software especificado neste trabalho foi desenvolvido na versão 8 da linguagem de programação Java.

Tabela 5 – Equações de controlabilidade criadas pela proposta do presente trabalho

Porta	*F0* e *F1*	Equações de controlabilidade
AND2X1	*F0*=!A+!B *F1*=A*B	$CC0_Y$=min{$CC0_A$,$CC0_B$}+1 $CC1_Y$=$CC1_A$+$CC1_B$+1
OR2X1	*F0*=!A*!B *F1*=A+B	$CC0_Y$=$CC0_A$+$CC0_B$+1 $CC1_Y$=min{$CC1_A$,$CC1_B$}+1
NOT	*F0*=A *F1*=!A	$CC0_Y$=$CC1_A$+1 $CC1_Y$=$CC0_A$+1
NAND2X1	*F0*=A*B *F1*=!A+!B	$CC0_Y$=$CC1_A$+$CC1_B$+1 $CC1_Y$=min{$CC0_A$,$CC0_B$}+1
NOR2X1	*F0*=A+B *F1*=!A*!B	$CC0_Y$=min{$CC1_A$,$CC1_B$}+1 $CC1_Y$=$CC0_A$+$CC0_B$+1
XOR2X1	*F0*=(A*B)+(!A*!B) *F1*=(A*!B)+(!A*B)	$CC0_Y$=min{($CC1_A$+$CC1_B$), ($CC0_A$+$CC0_B$)}+1 $CC1_Y$=min{($CC1_A$+$CC0_B$), ($CC0_A$+$CC1_B$)}+1
XNOR2X1	*F0*=(A*!B)+(!A*B) *F1*=(!A*!B)+(A*B)	$CC0_Y$=min{($CC1_A$+$CC0_B$), ($CC0_A$+$CC1_B$)}+1 $CC1_Y$=min{($CC0_A$+$CC0_B$), ($CC1_A$+$CC1_B$)}+1
OAI21X1	*F0*=(A*C)+(B*C) *F1*=(!A*!B)+!C	$CC0_Y$=min{($CC1_A$+$CC1_C$), ($CC1_B$+$CC1_C$)}+1 $CC1_Y$=min{($CC0_A$+$CC0_B$), ($CC0_C$}+1
OAI22X1	*F0*=A*C+A*D+B*C+B*D *F1*=(!A*!B)+(!C*!D)	$CC0_Y$=min{($CC1_A$+$CC1_C$), ($CC1_A$+$CC1_D$),($CC1_B$+$CC1_C$), ($CC1_B$+$CC1_D$)}+1 $CC1_Y$=min{($CC0_A$+$CC0_B$), ($CC0_C$+$CC0_D$}+1
AOI21X1	*F0*=(A*B)+C *F1*=(!A*!C)+(!B*!C)	$CC0_Y$=min{($CC1_A$+$CC1_B$), ($CC1_C$)}+1 $CC1_Y$=min{($CC0_A$+$CC0_C$), ($CC0_B$+$CC0_C$)}+1
AOI22X1	*F0*=(A*B)+(C*D) *F1*=(!A*!C)+(!B*!C)+ (A*!B*!D)+(!A*!D)	$CC0_Y$=min{($CC1_A$+$CC1_B$), ($CC1_C$+$CC1_D$)}+1 $CC1_Y$=min{($CC0_A$+$CC0_C$), ($CC0_B$+$CC0_C$), ($CC1_A$+$CC0_B$+$CC0_D$), ($CC0_A$+$CC0_D$)}+1
MX2X1	*F0*=(!A*!C)+(!B*C) *F1*=(A*!C)+(B*C)	$CC0_Y$=min{($CC0_A$+$CC0_C$), ($CC0_B$+$CC1_C$)}+1 $CC1_Y$=min{($CC1_A$+$CC0_C$), ($CC1_B$+$CC1_C$)}+1

Fonte: Elaborado pelo Autor

Tabela 6 – Equações de observabilidade criadas pela proposta do presente trabalho

Porta	Equações de observabilidade
AND2X1	$CO_A=CO_Y+CC1_B+1$ $CO_B=CO_Y+CC1_A+1$
OR2X1	$CO_A=CO_Y+CC0_B+1$ $CO_B=CO_Y+CC0_A+1$
NOT	$CO_A=CO_Y+1$
NAND2X1	$CO_A=CO_Y+CC1_B+1$ $CO_B=CO_Y+CC1_A+1$
NOR2X1	$CO_A=CO_Y+CC0_B+1$ $CO_B=CO_Y+CC0_A+1$
XOR2X1	$CO_A=CO_Y+\min\{CC0_B,CC1_B\}+1$ $CO_B=CO_Y+\min\{CC1_A,CC0_A\}+1$
XNOR2X1	$CO_A=CO_Y+\min\{CC0_B,CC1_B\}+1$ $CO_B=CO_Y+\min\{CC1_A,CC0_A\}+1$
OAI21X1	$CO_A=CO_Y+CC0_B+CC1_C+1$ $CO_B=CO_Y+CC0_A+CC1_C+1$ $CO_C=CO_Y+\min\{(CC1_A+CC0_B), (CC0_A+CC1_B), (CC1_A+CC1_B)\}+1$
OAI22X1	$CO_A=CO_Y+\min\{(CC0_B+CC1_C+CC1_D),(CC0_B+CC1_C+CC0_D),$ $(CC0_B+CC0_C+CC1_D)\}+1$ $CO_B=CO_Y+\min\{(CC0_A+CC1_C+CC1_D),(CC0_A+CC1_C+CC0_D),$ $(CC0_A+CC0_C+CC1_D)\}+1$ $CO_C=CO_Y+\min\{(CC0_A+CC1_B+CC0_D),(CC1_A+CC0_B+CC0_D),$ $(CC1_A+CC1_B+CC0_D)\}+1$ $CO_D=CO_Y+\min\{(CC0_A+CC1_B+CC0_C),(CC1_A+CC0_B+CC0_C),$ $(CC1_A+CC1_B+CC0_C)\}+1$
AOI21X1	$CO_A=CO_Y+CC1_B+CC0_C+1$ $CO_B=CO_Y+CC1_A+CC0_C+1$ $CO_C=CO_Y+\min\{(CC0_A+CC0_B), (CC1_A+CC0_B),(CC0_A+CC1_B)\}+1$
AOI22X1	$CO_A=CO_Y+\min\{(CC1_B+CC0_C+CC0_D),(CC1_B+CC1_C+CC0_D),$ $(CC1_B+CC0_C+CC1_D)\}+1$ $CO_B=CO_Y+\min\{(CC1_A+CC0_C+CC0_D),(CC1_A+CC1_C+CC0_D),$ $(CC1_A+CC0_C+CC1_D)\}+1$ $CO_C=CO_Y+\min\{(CC1_A+CC0_B+CC1_D),(CC0_A+CC1_B+CC1_D),$ $(CC0_A+CC0_B+CC1_D)\}+1$ $CO_D=CO_Y+\min\{(CC1_A+CC0_B+CC1_C),(CC0_A+CC1_B+CC1_C),$ $(CC0_A+CC0_B+CC1_C)\}+1$
MX2X1	$CO_A=CO_Y+CC0_C+1$ $CO_B=CO_Y+CC1_C+1$ $CO_C=CO_Y+\min\{(CC0_A+CC1_B),(CC1_A+CC0_B)\}+1$

Fonte: Elaborado pelo Autor

A Tabela 7 mostra a média de tempo de processamento em milissegundos para diferentes ações do algoritmo de criação das equações SCOAP para qualquer função. O tempo de processamento foi obtido a partir da média aritmética do tempo de 5 execuções dos algoritmos. Estas ações são: criação da tabela verdade, algoritmo Quine-McCluskey para o *F1*, algoritmo Quine-McCluskey para o *F0*, algoritmo da controlabilidade e o algoritmo da observabilidade.

Tabela 7 – Análise de escalabilidade - geração das equações SCOAP

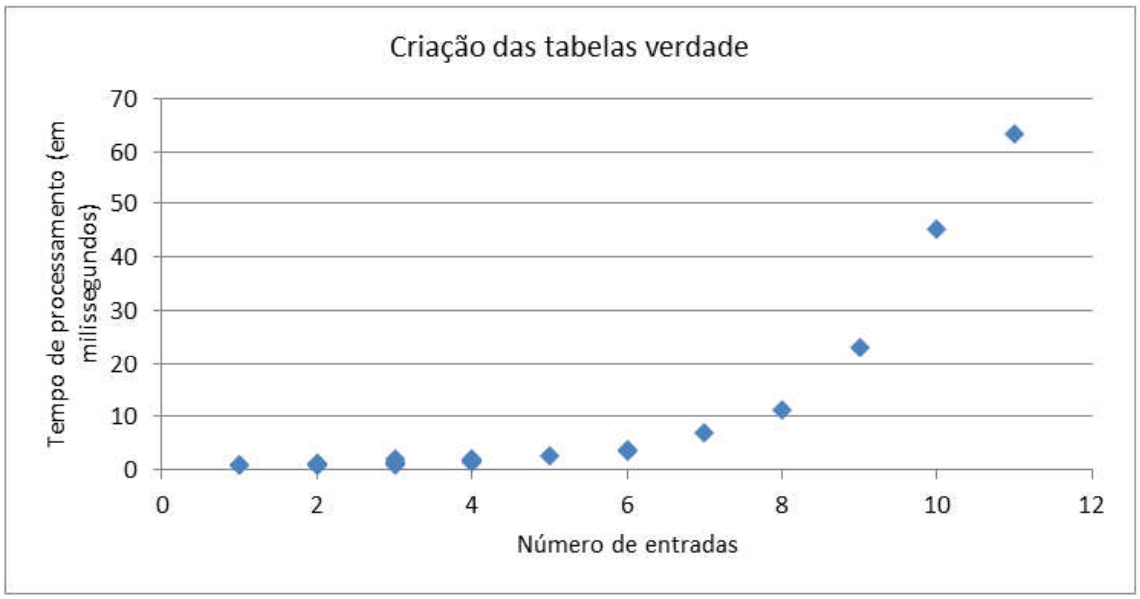
Porta	Função	Qnt. Ent.	Média de tempo de processamento (em milissegundos)				
			Criação da tabela verdade	*Quine-McCluskey F1*	*Quine-McCluskey F0*	Algoritmo controlabilidade	Algoritmo observabilidade
INVX1	!A	1	1	1	0,01	2,6	1
BUFX1	A	1	1	0,01	0,01	0,166	1
NOR2X1	!(A+B)	2	1	0,074	0,094	0,34	1
NOR3X1	!(A+B+C)	3	1,2	0,064	0,3	0,5	2,6
NOR4X1	!(A+B+C+D)	4	1,8	0,1	2	3,6	6,8
NAND2X1	!(A*B)	2	1,2	0,062	0,03	0,4	1
NAND3X1	!(A*B*C)	3	1,2	0,32	0,058	0,7	2,6
NAND4X1	!(A*B*C*D)	4	1,8	2	0,1	2	6
OR2X1	A+B	2	1	0,064	0,032	0,4	1,2
OR4X1	A+B+C+D	4	1,4	1,8	0,12	2	7
AND2X1	A*B	2	1	0,03	0,068	0,42	1,2
OAI21X1	!((A0+A1)*B0)	3	1	0,12	0,09	0,46	2
OAI22X1	!((A0+A1)*(B0+B1))	4	1,6	0,34	0,52	1,4	6,2
OAI32X1	!((A0+A1+A2)* (B0+B1))	5	2,6	0,88	3,8	5	24,6
OAI33X1	!((A0+A1+A2)* (B0+B1+B2))	6	3,6	3,6	56,2	60,6	77,6
AOI21X1	!(A0*A1+B0)	3	1,4	0,1	0,16	0,56	2,4
AOI22X1	!(A0*A1+B0*B1)	4	2,2	0,6	0,36	1,2	7,4
MX2X1	(A*B)+(S0*B)+(!S0*A)	3	2	0,14	0,2	0,54	2,8
XOR2X1	(A*!B)+(!A*B)	2	1,2	0,06	0,058	0,28	1
	(A0*(A1+A2* (A3+A4*A5)))	6	3,8	2,6	11,2	14	82,2
	(A0*(A1+A2* (A3+A4*(A5+A6))))	7	7	12	85,6	97,8	174,6
	(A0*(A1+A2*(A3+A4* (A5+A6*(A7)))))	8	11,4	83,2	237,8	321	403,6
	(A0*(A1+A2*(A3+A4* (A5+A6*(A7+A8)))))	9	23	242,4	1733,6	1977,4	982
	(A0*(A1+A2*(A3+A4* (A5+A6*(A7+A8* (A9))))))	10	45,2	1988,4	20258,6	22251,2	3346
	(A0*(A1+A2*(A3+A4* (A5+A6*(A7+A8* (A9*A10))))))	11	63,4	27299,8	304385,2	331689	11625,6

Fonte: Elaborado pelo Autor

A Figura 15 mostra o gráfico com os tempos de execução para a criação das tabelas verdade. A análise de complexidade assintótica é um método para descrever o comportamento de limites para saber como que a função de complexidade de tempo se comporta quando n (a variável para a qual se está fazendo a análise assintótica) tende ao infinito. A análise do tempo de processamento em relação ao número de variáveis das funções para o algoritmo de criação das tabelas verdade exposto no gráfico da Figura 15 mostra que o crescimento de n depende do próprio valor de

n, por isso esta é uma função de crescimento com comportamento de crescimento assintótico exponencial O(a^n).

Figura 15 – *Gráfico de tempo para a criação das tabelas verdade*



Fonte: Elaborado pelo Autor

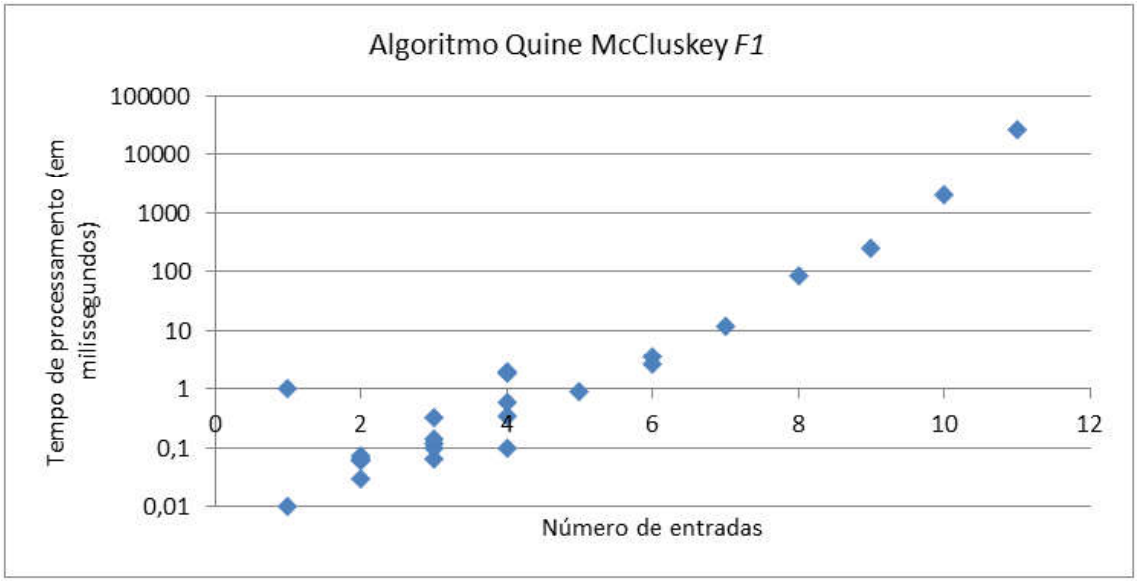
A Figura 16 mostra o gráfico de tempo de processamento para o algoritmo de processamento do *F1* com o método Quine-McCluskey, assim como a Figura 17 para o *F0*. O método Quine-McCluskey é usado para gerar o *F1* e o *F0* a partir das tabelas verdade das funções, e se mostrou o principal consumidor de tempo de execução dentro do algoritmo para criação das equações de controlabilidade, cujo gráfico de tempo de processamento é mostrado na Figura **??**. Todos os três gráficos dos tempos de processamento em relação ao número de entradas das portas lógicas para os algoritmos que fazem uso do método Quine-McCluskey (*F1*, *F0* e controlabilidade) mostram um crescimento assintótico da curva do gráfico que podem ser representados por uma função de complexidade de tempo de algoritmo da ordem de O(a^n).

O gráfico do algoritmo de processamento da observabilidade, mostrado na Figura 19, tem seu tempo de processamento dependente do número de variáveis das funções processadas e também das próprias funções. A análise dos dados denota um custo de execução com comportamento de crescimento assintótico O(a^n).

5.2 Ferramenta de análise de circuitos

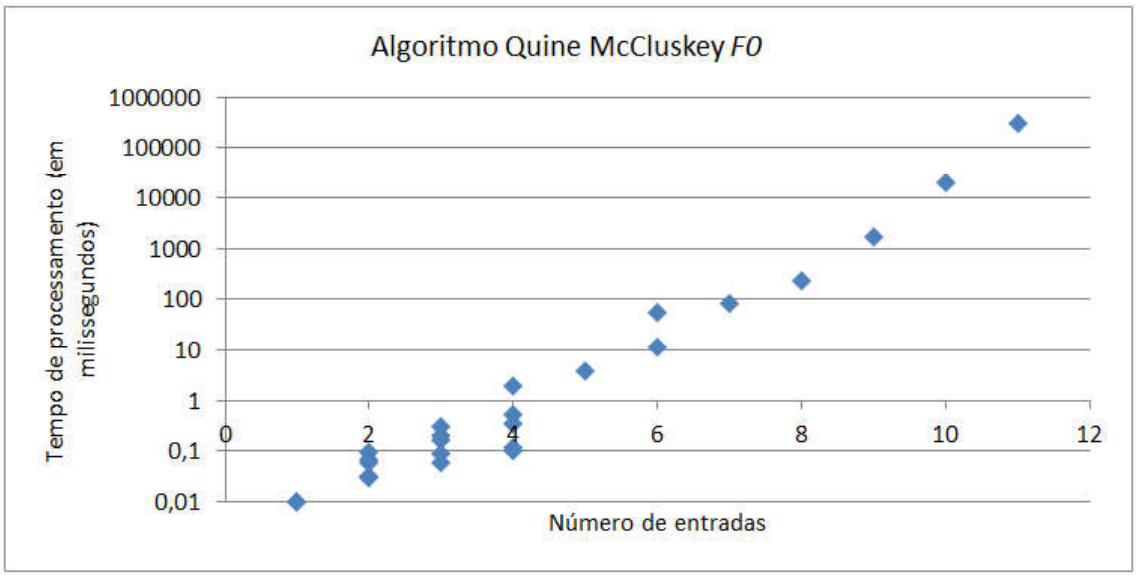
A validação da ferramenta de análise de circuitos digitais desenvolvida neste trabalho foi feita a partir de cinco topologias diferentes do circuito de *benchmark*

Figura 16 – *Gráfico de tempo para obter o F1 com o Quine-McCluskey*



Fonte: Elaborado pelo Autor

Figura 17 – *Gráfico de tempo para obter o F0 com o Quine-McCluskey*

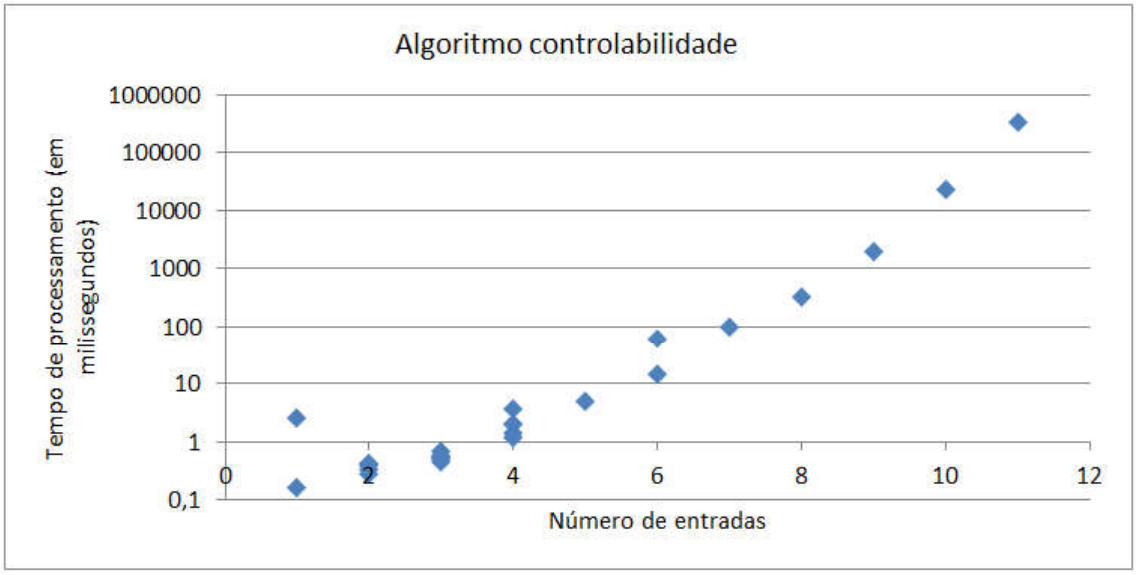


Fonte: Elaborado pelo Autor

c17. Este circuito é um dos circuitos combinacionais propostos no *International Symposium on Computer Architecture* do ano de 1985 (ISCAS'85) para a realização de *benchmarks* de softwares e ajudar na comparação de ferramentas ATPG.

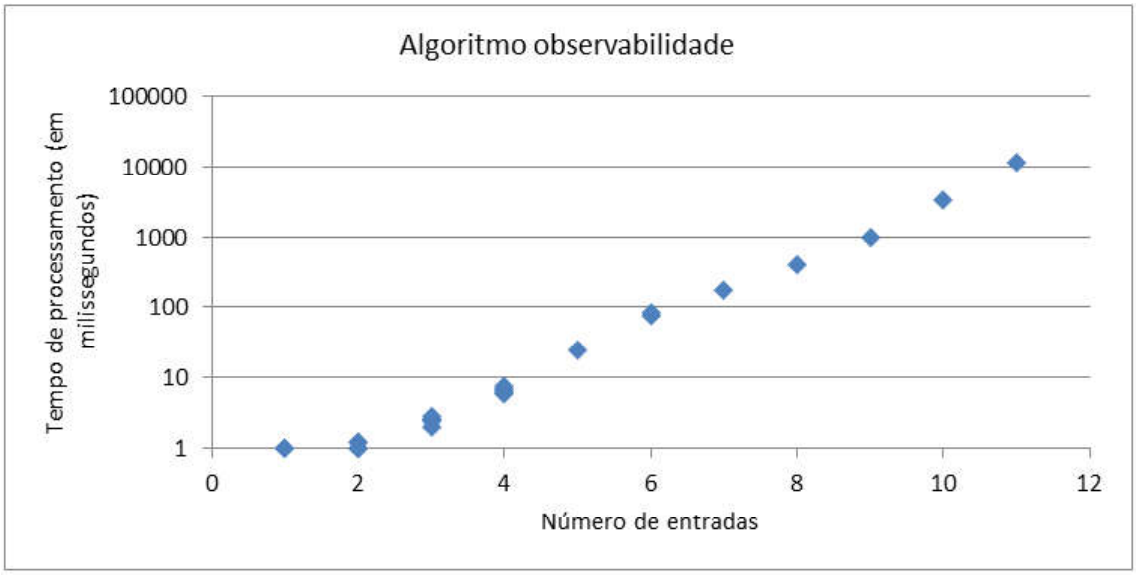
As Tabelas 8, 9, 10, 11 e 12 possuem os valores de CC0, CC1 e CO para o SCOAP e de *CY* e *OY* para o CAMELOT. O Anexo A trás as Figuras 25, 26, 27, 28 e 29 que mostram o cálculo de CC0, CC1 e CO para o SCOAP nas cinco diferentes topologias do circuito c17. Cada nodo do circuito apresenta seus valores

Figura 18 – *Gráfico de tempo para a criação das equações de controlabilidade*



Fonte: Elaborado pelo Autor

Figura 19 – *Gráfico de tempo para a criação das equações de observabilidade*



Fonte: Elaborado pelo Autor

de testabilidade, organizado da seguinte forma: três números inteiros separados por barras e entre parênteses, sendo o primeiro número relativo à CC0, o segundo à CC1 e o terceiro número à CO; os nomes das entradas primárias e das saídas primárias estão nelas, enquanto que os nomes dos nodos internos dos circuitos estão especificados a esquerda dos parênteses que contém os valores de testabilidade do nodo.

O Anexo B trás as Figuras 30, 31, 32, 33 e 34 que mostram o cálculo de CY e OY para o CAMELOT em cinco diferentes topologias do circuito c17. Cada nodo do

Tabela 8 – Cálculo da testabilidade para o circuito c17-v1

Nodo	SCOAP CC0	SCOAP CC1	SCOAP CO	CAMELOT CY	CAMELOT OY
p_1gat_0_	1	1	5	1,0	0,09375
p_6gat_3_	1	1	7	1,0	0,1796875
p_7gat_4_	1	1	6	1,0	0,125
p_2gat_1_	1	1	6	1,0	0,1796875
p_3gat_2_	1	1	7	1,0	0,256591796875
p_22gat_10_	5	4	0	0,21875	1,0
p_23gat_9_	4	6	0	0,25	1,0
n1	3	2	3	0,5	0,1875
n2	3	2	5	0,5	0,125
n3	2	2	6	1,0	0,125
n4	4	2	3	0,375	0,25
n5	2	2	6	1,0	0,125
n6	2	3	4	0,5	0,25
n7	3	5	3	0,5	0,25

Fonte: Elaborado pelo Autor

Tabela 9 – Cálculo da testabilidade para o circuito c17-v2

Nodo	SCOAP CC0	SCOAP CC1	SCOAP CO	CAMELOT CY	CAMELOT OY
p_1gat_0_	1	1	5	1,0	0,09375
p_6gat_3_	1	1	5	1,0	0,171875
p_7gat_4_	1	1	5	1,0	0,125
p_2gat_1_	1	1	5	1,0	0,1796875
p_3gat_2_	1	1	5	1,0	0,24951171875
p_22gat_10_	5	4	0	0,21875	1,0
p_23gat_9_	4	5	0	0,25	1,0
n1	3	2	3	0,5	0,34375
n2	4	2	3	0,375	0,25
n3	3	2	3	0,5	0,1875
n4	3	2	3	0,5	0,25

Fonte: Elaborado pelo Autor

circuito apresenta seus valores de testabilidade, organizado da seguinte forma: dois números reais separados por uma barra e entre parênteses, sendo o primeiro número relativo à CY e o segundo número à OY; os nomes das entradas primárias e das saídas primárias estão nelas, enquanto que os nomes dos nodos internos dos circuitos estão especificados a esquerda dos parênteses que contém os valores de testabilidade do nodo.

Tabela 10 – Cálculo da testabilidade para o circuito c17-v3

Nodo	SCOAP CC0	SCOAP CC1	SCOAP CO	CAMELOT CY	CAMELOT OY
p_1gat_0_	1	1	5	1,0	0,09375
p_6gat_3_	1	1	7	1,0	0,1353759765625
p_7gat_4_	1	1	6	1,0	0,046875
p_2gat_1_	1	1	6	1,0	0,09765625
p_3gat_2_	1	1	5	1,0	0,21643447875976562
p_22gat_10_	5	4	0	0,21875	1,0
p_23gat_9_	5	5	0	0,1875	1,0
n1	3	2	5	0,5	0,270751953125
n2	4	2	3	0,375	0,390625
n3	3	2	3	0,5	0,1875
n4	4	2	3	0,375	0,1875

Fonte: Elaborado pelo Autor

Tabela 11 – Cálculo da testabilidade para o circuito c17-v4

Nodo	SCOAP CC0	SCOAP CC1	SCOAP CO	CAMELOT CY	CAMELOT OY
p_1gat_0_	1	1	6	1,0	0,09375
p_6gat_3_	1	1	5	1,0	0,171875
p_7gat_4_	1	1	5	1,0	0,125
p_2gat_1_	1	1	5	1,0	0,1796875
p_3gat_2_	1	1	5	1,0	0,24951171875
p_22gat_10_	6	4	0	0,21875	1,0
p_23gat_9_	4	5	0	0,25	1,0
n1	2	2	6	1,0	0,0625
n2	2	3	3	0,5	0,34375
n3	3	5	3	0,375	0,25
n4	2	3	4	0,5	0,1875
n5	2	3	3	0,5	0,25

Fonte: Elaborado pelo Autor

5.2.1 Análise de escalabilidade para o software de análise de circuitos para o SCOAP

A análise de escalabilidade para o software de análise de circuitos foi feita com os circuitos de teste de *benchmark* propostos no ISCAS'85. A análise de tempo de execução foi dividida em três etapas, listadas a seguir: cálculo da controlabilidade, cálculo da observabilidade e pós-processamento.

A Tabela 13 mostra os tempos de execução para a etapa do cálculo da controlabilidade. A Figura 20 trás o gráfico mostrando a curva que relaciona o tempo de

Tabela 12 – Cálculo da testabilidade para o circuito c17 da Cadence

Nodo	SCOAP CC0	SCOAP CC1	SCOAP CO	CAMELOT CY	CAMELOT OY
\1GAT	1	1	7	1,0	0,28125
\2GAT	1	1	6	1,0	0,2890625
\3GAT	1	1	7	1,0	0,567626953125
\6GAT	1	1	7	1,0	0,3984375
\7GAT	1	1	6	1,0	0,1875
\22GAT	5	4	0	0,5	1,0
\23GAT	4	5	0	0,625	1,0
n7	2	2	5	1,0	0,2890625
n8	3	2	5	0,5	0,5625
n9	2	3	5	0,5	0,796875
n11	2	2	5	1,0	0,1875

Fonte: Elaborado pelo Autor

execução do algoritmo com a quantidade de portas, para o cálculo da controlabilidade, que mostra um crescimento linear do tempo de execução do algoritmo relacionado ao número de portas do circuito. Isso está de acordo com uma complexidade algorítmica da ordem de O(n).

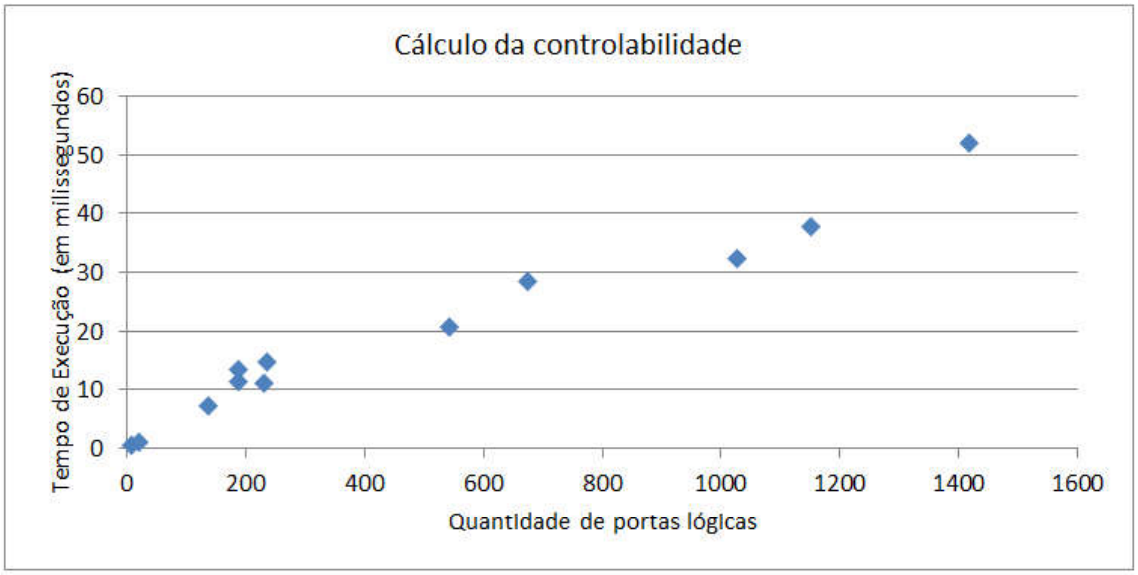
Tabela 13 – Análise de escalabilidade - cálculo da controlabilidade

Porta	Qnt. Portas	Cálculo da controlabilidade					
		Tempo de Execução (em milissegundos)					
		t1	t2	t3	t4	t5	Média
c17	6	0,5	0,4	0,4	0,5	0,5	0,46
c20	20	2	1	1	1	1	1,2
c432	136	6	6	7	11	6	7,2
c499	188	18	8	9	14	8	11,4
c1355	188	20	8	13	11	15	13,4
c880	229	9	11	17	9	9	11
c1908	234	13	17	13	16	14	14,6
c2670	543	16	24	23	20	21	20,8
c3540	673	25	33	26	25	33	28,4
c5315	1026	31	30	33	37	30	32,2
c7552	1151	32	32	50	33	41	37,6
c6288	1418	54	34	54	64	54	52

Fonte: Elaborado pelo Autor

A Tabela 14 mostra os tempos de execução para o cálculo da observabilidade. A Figura 21 com o gráfico mostrando os dados de quantidade de portas lógicas de um circuito em relação ao tempo de execução do algoritmo para o cálculo

Figura 20 – *Gráfico de tempo para o cálculo da controlabilidade*



Fonte: Elaborado pelo Autor

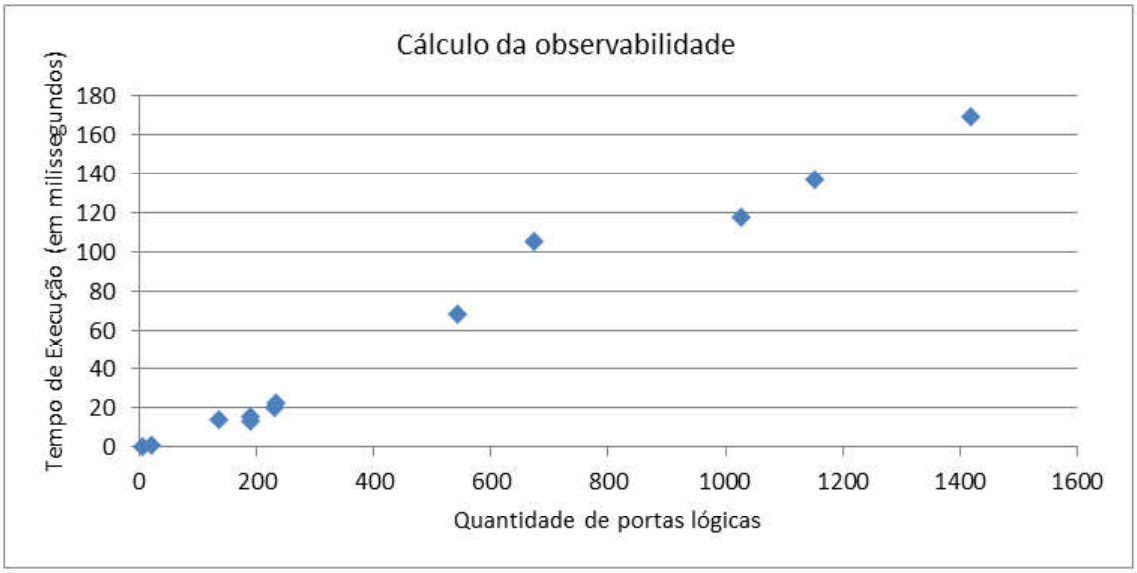
da observabilidade mostra um crescimento de tempo de execução linear relativo ao crescimento do número de portas lógicas. Isso denota um comportamento de complexidade algorítmica da ordem de O(n).

Tabela 14 – Análise de escalabilidade - cálculo da observabilidade

Porta	Qnt. Portas	Cálculo da observabilidade					
		Tempo de Execução (em milissegundos)					
		t1	t2	t3	t4	t5	Média
c17	6	0,3	0,2	0,3	0,3	0,3	0,28
c20	20	1	0,7	0,6	0,7	1	0,8
c432	136	11	20	14	11	15	14,2
c499	188	15	17	15	14	15	15,2
c1355	188	15	14	11	13	14	13,4
c880	229	21	22	17	19	22	20,2
c1908	234	25	21	20	23	23	22,4
c2670	543	67	69	65	70	72	68,6
c3540	673	111	100	101	108	106	105,2
c5315	1026	104	121	136	115	111	117,4
c7552	1151	133	132	136	131	152	136,8
c6288	1418	161	214	139	161	170	169

A Tabela 15 mostra os tempos de execução da etapa de pós-processamento, que compreende percorrer o grafo e listar os valores de controlabilidade e observabilidade. A Figura 22 trás o gráfico da relação de tempo de processamento pela quantidade de portas do circuito, para o pós-processamento, que mostra um crescimento da curva

Figura 21 – *Gráfico de tempo para o cálculo da observabilidade*



Fonte: Elaborado pelo Autor

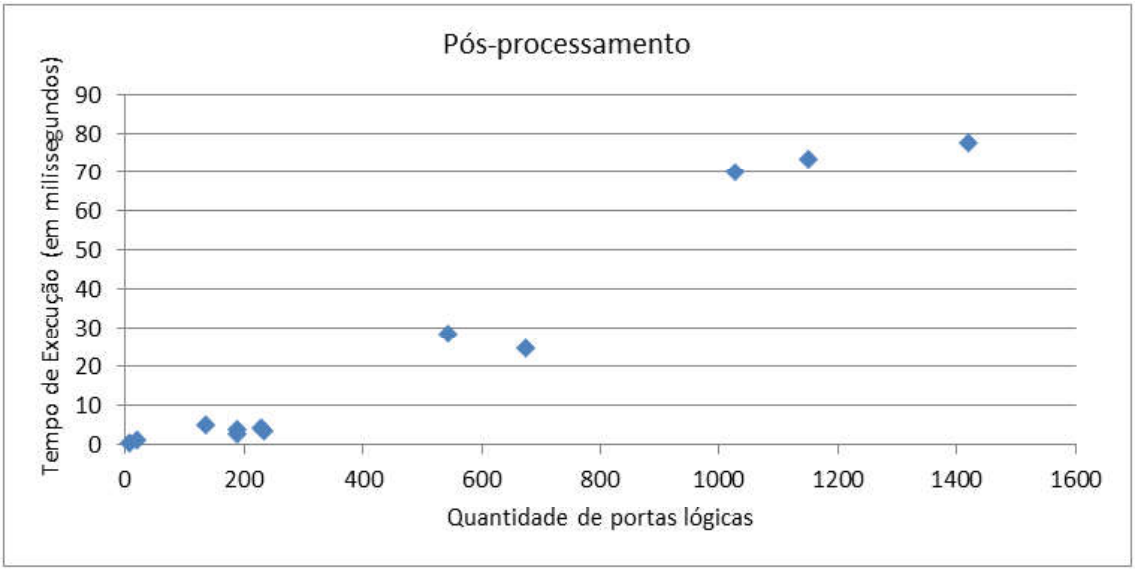
do gráfico da ordem de complexidade algorítmica de O(n).

Tabela 15 – Análise de escalabilidade - pós-processamento

Porta	Qnt. Portas	Pós-processamento					
		Tempo de Execução (em milissegundos)					
		t1	t2	t3	t4	t5	Média
c17	6	0,8	0,3	0,4	0,4	0,5	0,48
c20	20	2	1	1	1	2	1,4
c432	136	6	8	3	5	3	5
c499	188	5	3	4	4	3	3,8
c1355	188	3	3	3	3	2	2,8
c880	229	4	5	3	3	7	4,4
c1908	234	4	4	3	4	3	3,6
c2670	543	25	25	28	35	30	28,6
c3540	673	27	17	18	30	31	24,6
c5315	1026	93	84	47	78	48	70
c7552	1151	69	59	65	102	72	73,4
c6288	1418	87	69	79	79	75	77,8

Fonte: Elaborado pelo Autor

Figura 22 – *Gráfico de tempo para o pós-processamento*



Fonte: Elaborado pelo Autor

5.2.2 Análise de escalabilidade para o software de análise de circuitos para o CAMELOT

A Tabela 16 mostra os tempos de execução para o cálculo da controlabilidade e da observabilidade para o CAMELOT. O método CAMELOT não gera diferentes equações para cada função lógica como no método SCOAP, mas é fortemente dependente da geração da tabela verdade para a função para a qual se está calculando a controlabilidade e a observabilidade. Um circuito é composto por diversas portas lógicas, cada uma representando uma função. Portanto, o tempo de processamento do CAMELOT para um circuito é dependente do número de portas lógicas deste circuito e da quantidade de variáveis de cada função representada pelas portas lógicas. O gráfico de geração de tabelas verdade para o processamento das equações SCOAP mostrou um comportamento de crescimento assintótico linear O(n), sendo este o mesmo método usado para a geração das tabelas verdade usadas pelo CAMELOT.

O cálculo do CTF de uma função é baseado em dados do resultado da tabela verdade desta função. Estes dados são o número de saídas com valor lógico 1, representado pela variável N(1), e o número de saídas com valor lógico 0, representada pela variável N(0), de uma determinada tabela verdade. Estes dados são usados em uma expressão algébrica dada pela diferença de 1 com o módulo da razão da diferença e a soma de N(0) e N(1). A complexidade do algoritmo para esta expressão algébrica é linear, dada por O(n).

O cálculo da controlabilidade da saída de uma determinada porta lógica é dado por outra expressão algébrica com complexidade algorítmica linear O(n), dada pela

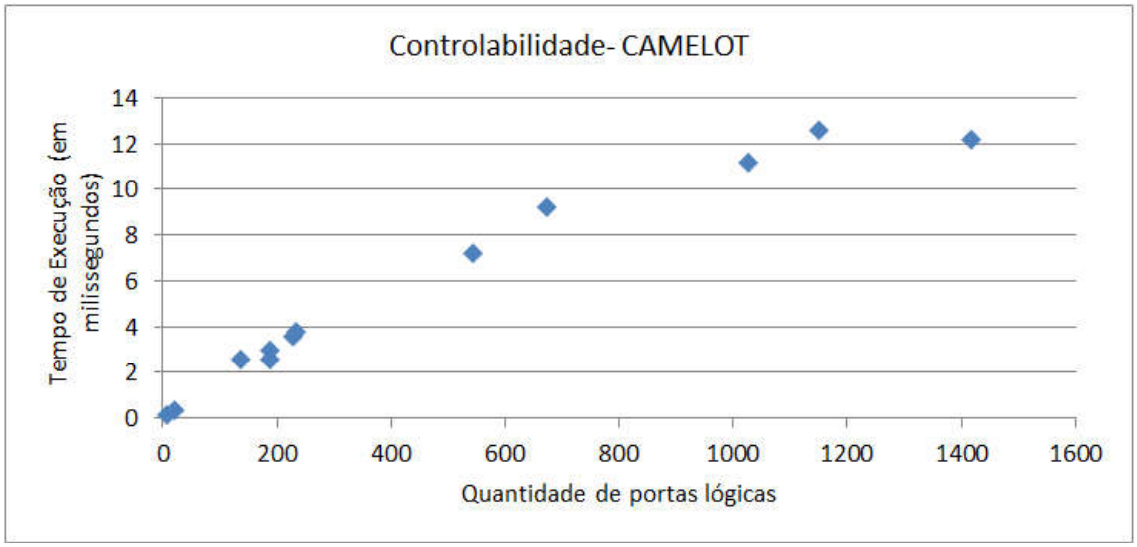
multiplicação do CTF desta porta lógica pela média aritmética das controlabilidades das estradas desta mesma porta lógica. Com base nisso, e na análise da Figura 23 com o gráfico do tempo de execução para o cálculo da controlabilidade no CAMELOT, conclui-se que a complexidade algorítmica para o cálculo da controlabilidade usando o método CAMELOT é O(n).

Tabela 16 – Análise de escalabilidade - tempos de execução para o CAMELOT

Bench-mark	Qnt. Portas	Tempo de Execução (em milissegundos)										Média	
		T1 *CY*	T1 *OY*	T2 *CY*	T2 *OY*	T3 *CY*	T3 *OY*	T4 *CY*	T4 *OY*	T5 *CY*	T5 *OY*	T *CY*	T *OY*
c17	6	0,2	4	0,1	5	0,2	4	0,2	4	0,2	5	0,18	4,4
c20	20	0,3	8	0,4	13	0,4	8	0,3	10	0,4	12	0,36	10,2
c432	136	2	55	3	65	3	55	2	59	3	54	2,6	57,6
c499	188	3	72	4	76	2	86	3	74	3	82	3	78
c1355	188	2	56	2	76	3	70	3	67	3	81	2,6	70
c880	229	4	82	3	75	4	79	3	107	4	69	3,6	82,4
c1908	234	3	63	4	66	4	83	4	73	4	74	3,8	71,8
c2670	543	7	130	7	124	8	134	6	143	8	137	7,2	133,6
c3540	673	8	178	10	172	10	202	8	219	10	196	9,2	193,4
c5315	1026	11	180	11	190	12	175	10	189	12	213	11,2	189,4
c7552	1151	14	220	13	235	13	326	13	243	10	221	12,6	249
c6288	1418	10	366	12	265	13	211	14	255	12	277	12,2	274,8

Fonte: Elaborado pelo Autor

Figura 23 – *Gráfico de tempo para processamento da controlabilidade no CAMELOT*



Fonte: Elaborado pelo Autor

O algoritmo da observabilidade do método CAMELOT calcula o valor de observabilidade para cada uma das entradas da função. Neste processo cada entrada é avaliada para saber se é a única a sensibilizar a saída da função, para cada um dos possíveis vetores de entrada da função, e o resultado desta avaliação gera o número de caminhos sensíveis distintos (SP) e o número de caminhos insensíveis distintos (IP). Este algoritmo possui comportamento assintótico O(n).

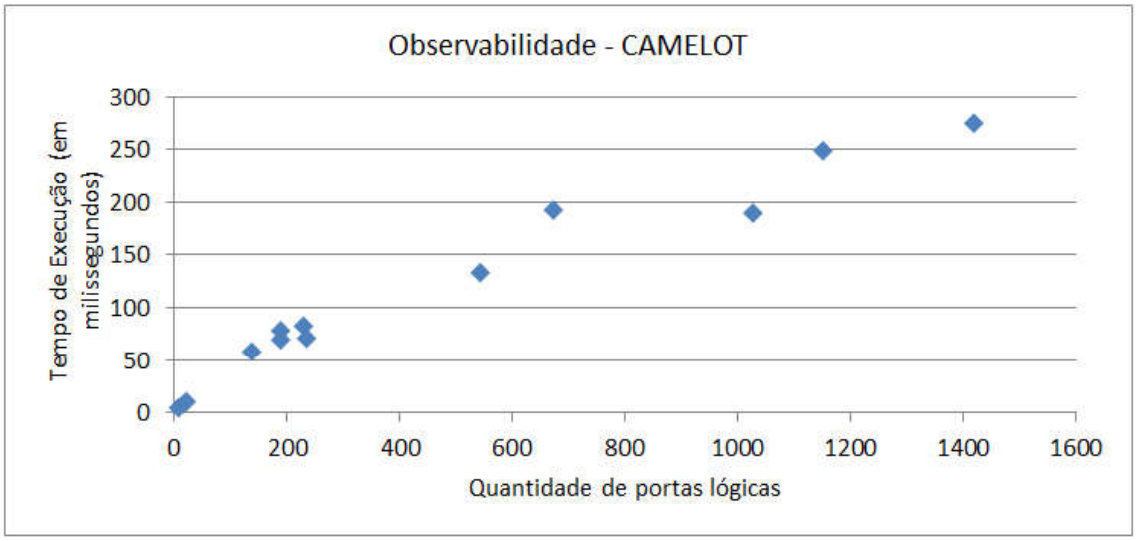
O número de caminhos sensíveis e insensíveis distintos para cada entrada é usado no cálculo do OTF pela expressão algébrica da razão do número de caminhos sensíveis pela soma do número de caminhos sensíveis e o número de caminhos insensíveis. Esta expressão algébrica tem comportamento algorítmico linear O(n).

O cálculo da observabilidade é dado pelos produtos do OTF pela observabilidade da saída e pela média aritmética das controlabilidades das entradas de apoio (todas as entradas distintas da entrada para a qual se está calculando a observabilidade). Este cálculo é linear e possui complexidade algorítmica O(n). Com base nisso, e na análise da Figura 24 com o gráfico do tempo de execução para o cálculo da observabilidade no CAMELOT, conclui-se que a complexidade algorítmica para o processamento da observabilidade de um circuito no CAMELOT é de O(n).

5.3 Análise dos resultados obtidos

De posse dos dados obtidos neste trabalho, presentes nas Tabelas 8, 9, 10, 11 e 12, é possível fazer inúmeras análises. Nesta análise foi considerado o maior valor de controlabilidade e de observabilidade obtido em cada circuito, tanto para os

Figura 24 – *Gráfico de tempo para processamento da observabilidade no CAMELOT*



Fonte: Elaborado pelo Autor

métodos SCOAP quanto para o CAMELOT, desconsiderando se o valor obtido é de CC0 ou de CC1 e quantas vezes ele aparece repetidamente. No circuito c17-v1, o maior valor de CC1 pode ser observado no nodo p_23gat_9_ com o valor de 6 e o menor valor de CY é 0,218 no nodo p_22gat_10_. No circuito c17-v2, o maior valor de CC1 pode ser observado no nodo p_23gat_9_ com o valor de 5 e o menor valor de CY é 0,218 no nodo p_22gat_10_. No circuito c17-v3, o maior valor de CC0 pode ser observado no nodo p_23gat_9_ com o valor de 5 e o menor valor de CY é 0,187 no nodo p_23gat_9_. No circuito c17-v4, o maior valor de CC0 pode ser observado no nodo p_22gat_10_ com o valor de 6 e o menor valor de CY é 0,218 no nodo p_22gat_10_. No circuito c17 da Cadence, o maior valor de CC0 pode ser observado no nodo \22GAT com o valor de 5 e o menor valor de CY é 0,5 no nodo \22GAT.

No circuito c17-v1, o maior valor de CO pode ser observado no nodo p_6gat_3_ com o valor de 7 e o menor valor de OY é 0,093 no nodo p_1gat_0_. No circuito c17-v2, o maior valor de CO pode ser observado no nodo p_1gat_0_ com o valor de 5 e o menor valor de OY é 0,093 no nodo p_1gat_0_. No circuito c17-v3, o maior valor de CO pode ser observado no nodo p_6gat_3_ com o valor de 7 e o menor valor de OY é 0,046 no nodo p_6gat_3_. No circuito c17-v4, o maior valor de CO pode ser observado no nodo p_1gat_0_ com o valor de 6 e o menor valor de OY é 0,093 no nodo p_1gat_0_. No circuito c17 da Cadence, o maior valor de CO pode ser observado no nodo \1GAT com o valor de 7 e o menor valor de OY é 0,187 no nodo \7GAT.

Com base nestes dados, verifica-se que o circuito c17-v2 possui melhor testabi-

lidade no método SCOAP e o circuito c17 da Cadence possui melhor testabilidade no método CAMELOT, pois ambos mostraram os melhores valores de controlabilidade e observabilidade. Considerando que o valor de CC0 no método SCOAP para o circuito c17 da Cadence é o mesmo valor de CC0 para o circuito c17-v2, desconsiderando o valor de CO maior do c17 da Cadence e sua melhor testabilidade para o método CAMELOT, conclui-se que o circuito c17 da Cadence é o que possui melhor testabilidade dentre as cinco diferentes topologias do circuito c17 estudadas neste trabalho.

6 Considerações finais

O presente trabalho teve o propósito de dar uma breve visão da área de teste de CIs focada na testabilidade, apresentando uma solução genérica para obtenção das equações para o cálculo da controlabilidade e da observabilidade para quaisquer funções lógicas no método SCOAP, assim como a comparação do método SCOAP com o método CAMELOT.

Ao abordar o método SCOAP neste trabalho, verificou-se a relação entre o *F1* e o *F0* de uma função lógica e as equações para o cálculo da controlabilidade. Esta relação deu origem a um procedimento para a criação destas equações e a um software para a automação da criação destas equações. Também foi analisada e explanada a definição de Goldstein (1979) para a geração das equações para o cálculo da observabilidade, e a partir dela sistematizado um processo para a geração destas equações, assim como foi desenvolvido um software para a automação da geração destas equações.

As equações de controlabilidade e de observabilidade criadas com o método SCOAP para as portas lógicas AND, NAND, OR, NOR, XOR, XNOR e NOT geradas a partir da proposta deste trabalho são as mesmas apresentadas por Wang, Chang e Cheng (2009), Wang, Wu e Wen (2006) e Kantipudi (2010). Além disso, as equações para as portas lógicas OAI21X1, OAI22X1, AOI21X1, AOI22X1 e MX2X1 também foram geradas usando a proposta do presente trabalho, com o intuito de ilustrar seu potencial. Esta contribuição permite a expansão do uso do método SCOAP para além da proposta na literatura da área.

Foi criado um software para o cálculo da controlabilidade e da observabilidade de circuitos combinacionais adotando os métodos SCOAP e CAMELOT. Foram processadas cinco topologias diferentes do CI c17 e os resultados mostraram que tanto o método SCOAP, como o CAMELOT, possibilitam a escolha de uma topologia de CI com as melhores métricas de testabilidade.

Este trabalho mostrou a importância das métricas de testabilidade de CIs para o projeto de CIs com foco na testabilidade, podendo ser usadas como auxílio interativo no projeto de CIs e possibilitando a redução do custo do teste de CIs. Como trabalhos futuros, as métricas de testabilidades podem ser exploradas na procura de falhas do tipo *easy-to-test* (fáceis de testar) e *hard-to-test* (difíceis de testar), possibilitando a automação da análise e comparação de diferentes topologias de CIs e o estabelecimento da melhor topologia de acordo com critérios de escolha da topologia com menor quantidade de falhas difíceis de testar.

Referências

BENNETTS, R. *Design of testable logic circuits*. Addison-Wesley Pub. Co., 1984. (Microelectronics systems design series). ISBN 9780201144031. Disponível em: <https://books.google.com.br/books?id=O_5SAAAAMAAJ>. Citado 2 vezes nas páginas 25 e 26.

BENNETTS, R.; MAUNDER, C.; ROBINSON, G. Comelot: a computer-aided measure for logic testability. *IEE Proceedings E (Computers and Digital Techniques)*, IET, v. 128, n. 5, p. 177–189, 1981. Citado 5 vezes nas páginas 13, 21, 22, 24 e 25.

BENNETTS, R. G. *Introduction to digital board testing*. [S.l.]: Russak, 1982. Citado na página 21.

BREUER, M. A.; FRIEDMAN, A. D. Test/80- a proposal for an advanced automatic test generation system. *AUTOTESTCON'79*, p. 305–312, 1979. Citado na página 12.

BUSHNELL, M.; AGRAWAL, V. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. [S.l.]: Springer Science & Business Media, 2004. v. 17. Citado 3 vezes nas páginas 11, 12 e 21.

CORMEN, T. H. *Introduction to algorithms*. [S.l.]: MIT press, 2009. Citado na página 12.

DANNER, F.; CONSOLLA, W. An objective pcb testability rating system'. In: *Proceedings IEEE semiconductor test conference*. [S.l.: s.n.], 1979. p. 23–28. Citado na página 22.

DEJKA, W. J. Measure of testability in device and system design. In: *Proc. 20th Midwest Symposium on Circuits and Systems*. [S.l.: s.n.], 1977. p. 39–52. Citado na página 22.

GOLDSTEIN, L. Controllability/observability analysis of digital circuits. *IEEE Transactions on Circuits and Systems*, IEEE, v. 26, n. 9, p. 685–693, 1979. Citado 4 vezes nas páginas 22, 23, 30 e 60.

GOLDSTEIN, L. H.; THIGPEN, E. L. Scoap: Sandia controllability/observability analysis program. In: ACM. *Proceedings of the 17th Design Automation Conference*. [S.l.], 1980. p. 190–196. Citado 2 vezes nas páginas 13 e 22.

GRASON, J. Tmeas, a testability measurement program. In: IEEE PRESS. *Proceedings of the 16th Design Automation Conference*. [S.l.], 1979. p. 156–161. Citado na página 13.

JAMIL, T.; MOHAMMED, I. Simulation of victor algorithm for fault-diagnosis of digital circuits. *International Journal of Computer Theory and Engineering*, IACSIT Press, v. 7, n. 2, p. 103, 2015. Citado na página 26.

JHA, N. K.; GUPTA, S. *Testing of digital systems*. [S.l.]: Cambridge University Press, 2003. Citado na página 20.

JUNIOR, R. et al. Automatic generation and evaluation of transistor networks in different logic styles. 2008. Citado na página 15.

KAESLIN, H.; FELBER, N. *VLSI II: Design of Very Large Scale Integration Circuits*. [S.l.], 2016. Citado na página 20.

KANTIPUDI, K. R. Controllability and observability. *ELEC7250-001 VLSI Testing (Spring 2005), Instructor: Professor Vishwani D. Agrawal*, 2010. Citado 5 vezes nas páginas 20, 22, 23, 26 e 60.

KARIMI, N.; RIYAHI, P.; NAVABI, Z. A survey of testability measurements at various abstraction levels. In: CITESEER. *Proc. NATW conf. Citeseer*. [S.l.], 2003. Citado na página 26.

KARNAUGH, M. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, IEEE, v. 72, n. 5, p. 593–599, 1953. Citado na página 17.

MEHTA, U.; DHARE, V. Quantum-dot cellular automata (qca): A survey. *arXiv preprint arXiv:1711.08153*, 2017. Citado na página 28.

NELSON, V. P.; NAGLE. *Digital logic circuit analysis and design*. [S.l.]: Prentice Hall, 1995. Citado na página 17.

RAJSKI, J. et al. *Test Point-Enhanced Hardware Security*. [S.l.]: Google Patents, 2017. US Patent App. 15/353,412. Citado na página 27.

REIS, R. A. et al. *LIVRO TEXTO: I ESCOLA DE MICROELETRÔNICA DA SBC-SUL*. [S.l.]: Editora da UFRGS - Porto Alegre, 1999. v. 1. Citado na página 11.

SALMANI, H. Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. *IEEE Transactions on Information Forensics and Security*, IEEE, v. 12, n. 2, p. 338–350, 2017. Citado na página 27.

SALMANI, H. *Trusted Digital Circuits: Hardware Trojan Vulnerabilities, Prevention and Detection*. [S.l.]: Springer, 2018. Citado na página 27.

SALMANI, H.; TEHRANIPOOR, M. M. Vulnerability analysis of a circuit layout to hardware trojan insertion. *IEEE Transactions on Information Forensics and Security*, IEEE, v. 11, n. 6, p. 1214–1225, 2016. Citado na página 27.

SCOTT, N. A new approach to the design of switching circuits. *Proceedings of the IEEE*, IEEE, v. 51, n. 2, p. 413–413, 1963. Citado na página 17.

TILLE, D. et al. Structural heuristics for sat-based atpg. In: IEEE. *Very Large Scale Integration (VLSI-SoC), 2009 17th IFIP International Conference on*. [S.l.], 2009. p. 77–82. Citado na página 26.

UYEMURA, J. P. Introduction to vlsi circuits and systems. Wiley India, 2002. Citado na página 12.

WAKERLY, J. F. *Digital design: Principles and practices*. [S.l.]: Prentice-Hall, Inc. USA, 2000. Citado 2 vezes nas páginas 15 e 16.

WANG, L.-T.; CHANG, Y.-W.; CHENG, K.-T. T. *Electronic design automation: synthesis, verification, and test*. [S.l.]: Morgan Kaufmann, 2009. Citado 9 vezes nas páginas 11, 19, 21, 23, 24, 29, 30, 31 e 60.

WANG, L.-T.; WU, C.-W.; WEN, X. *VLSI test principles and architectures: design for testability*. [S.l.]: Academic Press, 2006. Citado 3 vezes nas páginas 12, 23 e 60.

WESTE, N. H.; ESHRAGHIAN, K. *Principles of CMOS VLSI design*. [S.l.]: Addison-Wesley New York, 1985. v. 188. Citado 2 vezes nas páginas 11 e 12.

WESTE, N. H.; HARRIS, D. *CMOS VLSI design: a circuits and systems perspective*. [S.l.]: Pearson Education India, 2015. Citado 2 vezes nas páginas 11 e 20.

WILLIAMS, M. J. Y.; ANGELL, J. B. Enhancing testability of large-scale integrated circuits via test points and additional logic. *IEEE Transactions on Computers*, IEEE, v. 100, n. 1, p. 46–60, 1973. Citado na página 19.

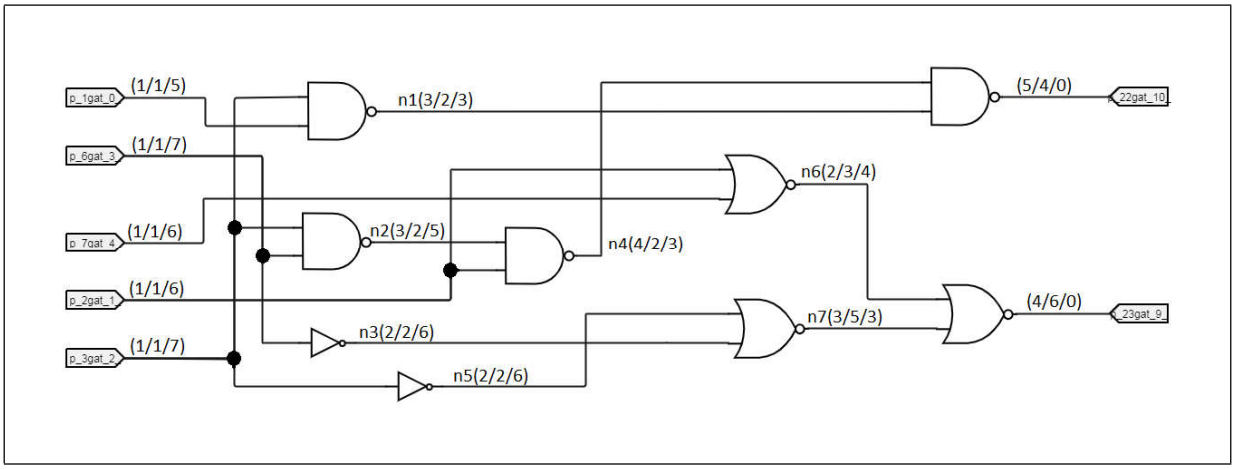
WOOD, C. T. The quantitative measure of testability. In: *AUTOTESTCON'79*. [S.l.: s.n.], 1979. p. 286–291. Citado na página 22.

XIE, X. et al. Hardware trojans classification based on controllability and observability in gate-level netlist. *IEICE Electronics Express*, The Institute of Electronics, Information and Communication Engineers, v. 14, n. 18, p. 20170682–20170682, 2017. Citado na página 27.

ZUFFO, J. A. *Sistemas eletronicos digitais; organizacao interna e projeto*. [S.l.]: E. Blucher, 1976. Citado 2 vezes nas páginas 16 e 17.

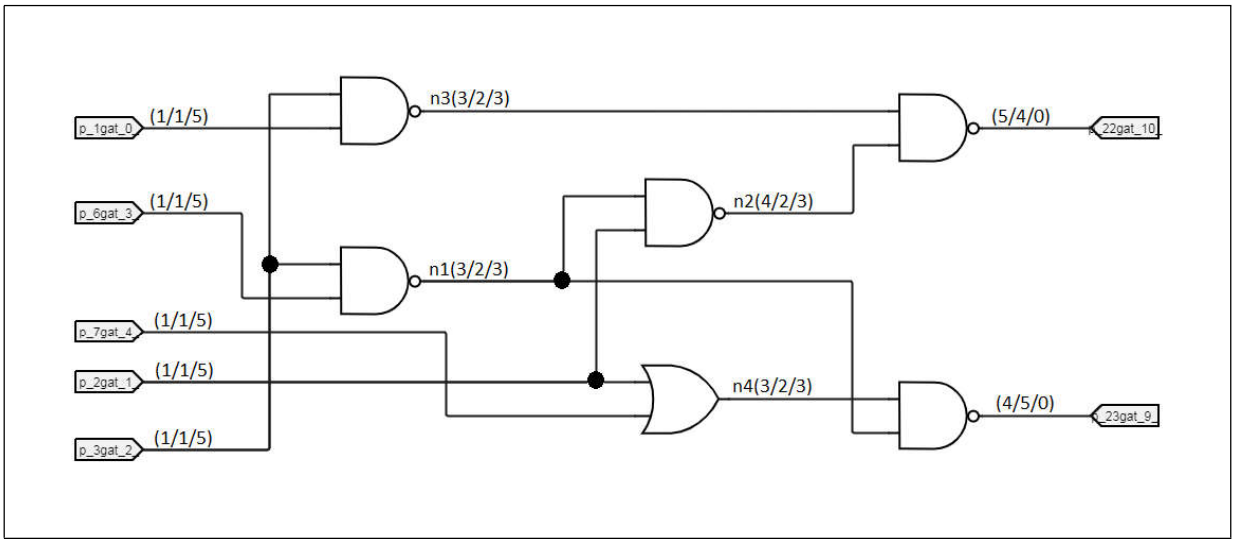
ANEXO A – Circuitos com o cálculo de CC0, CC1 e CO para o SCOAP

Figura 25 – *Cálculo da testabilidade do c17-v1 para o SCOAP*



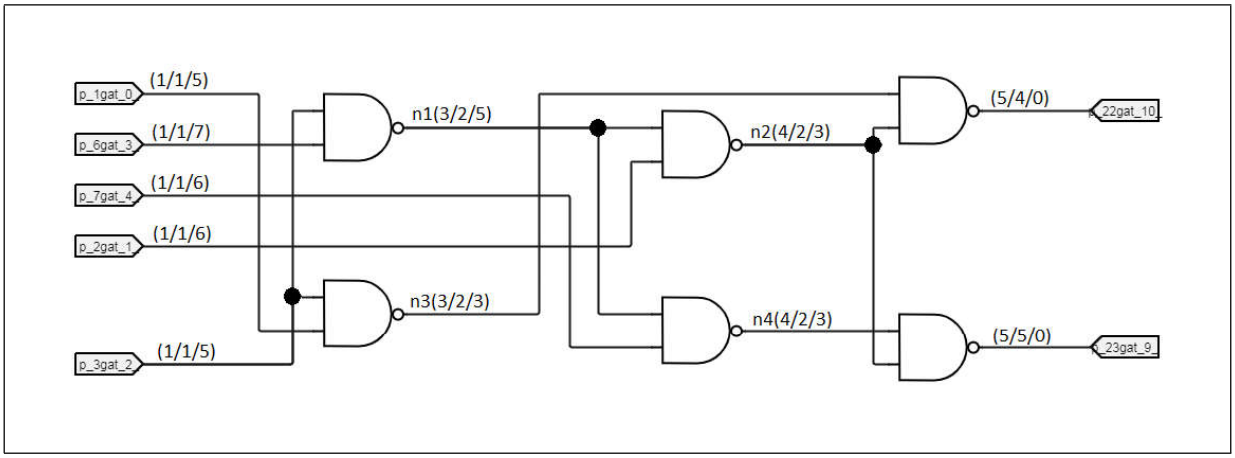
Fonte: Elaborado pelo Autor

Figura 26 – *Cálculo da testabilidade do c17-v2 para o SCOAP*



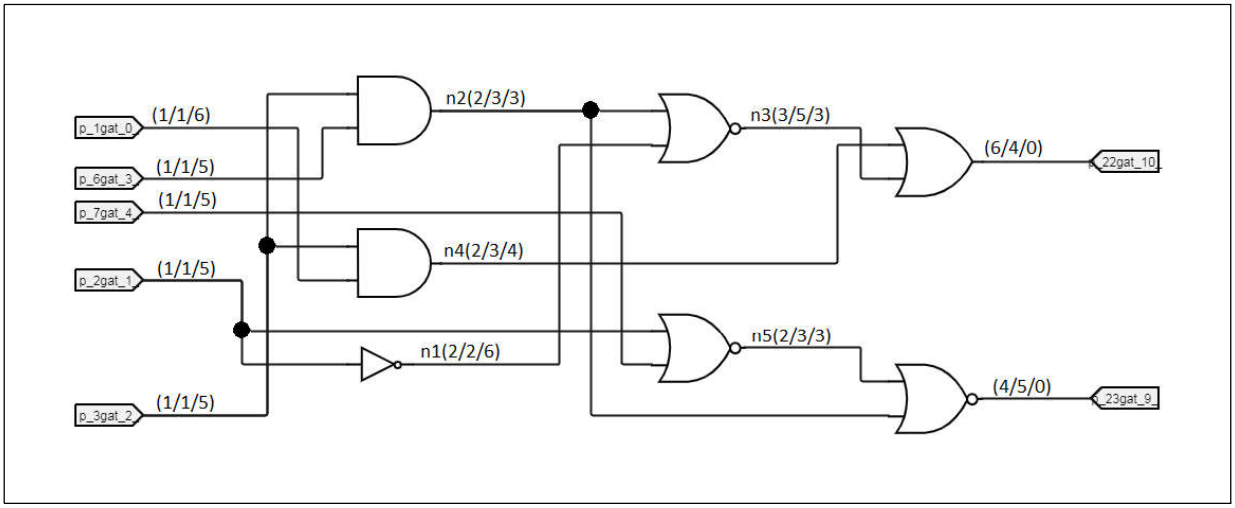
Fonte: Elaborado pelo Autor

Figura 27 – *Cálculo da testabilidade do c17-v3 para o SCOAP*



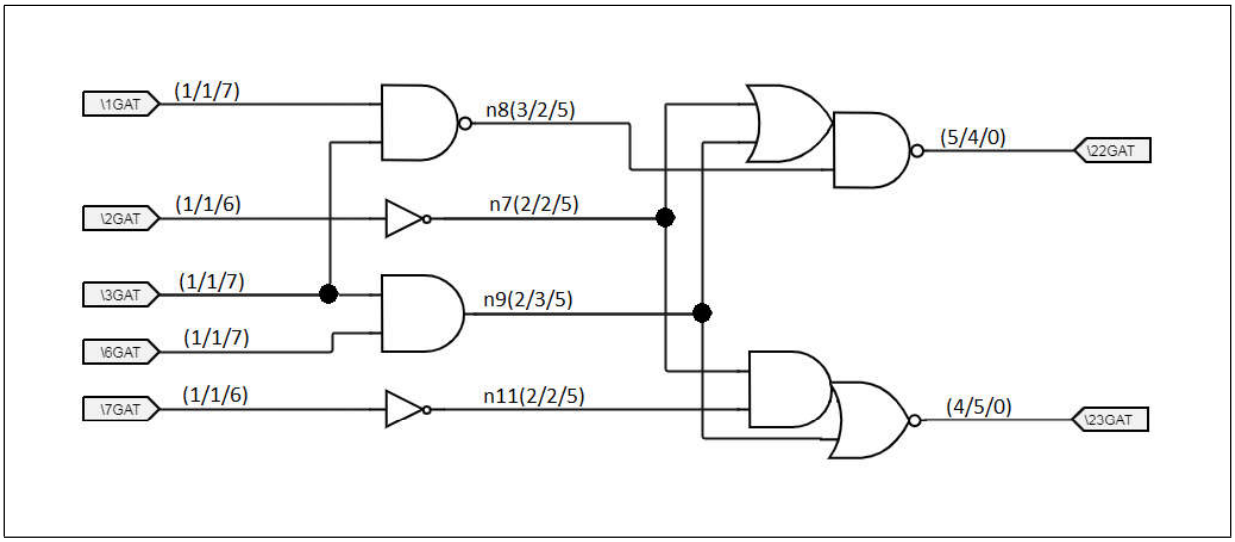
Fonte: Elaborado pelo Autor

Figura 28 – *Cálculo da testabilidade do c17-v4 para o SCOAP*



Fonte: Elaborado pelo Autor

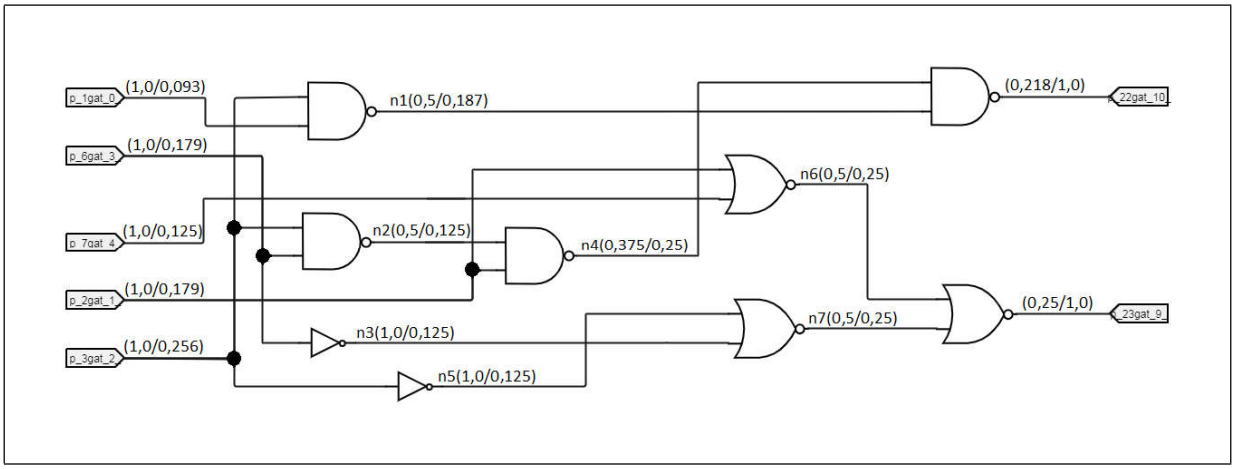
Figura 29 – *Cálculo da testabilidade do c17 da Cadence para o SCOAP*



Fonte: Elaborado pelo Autor

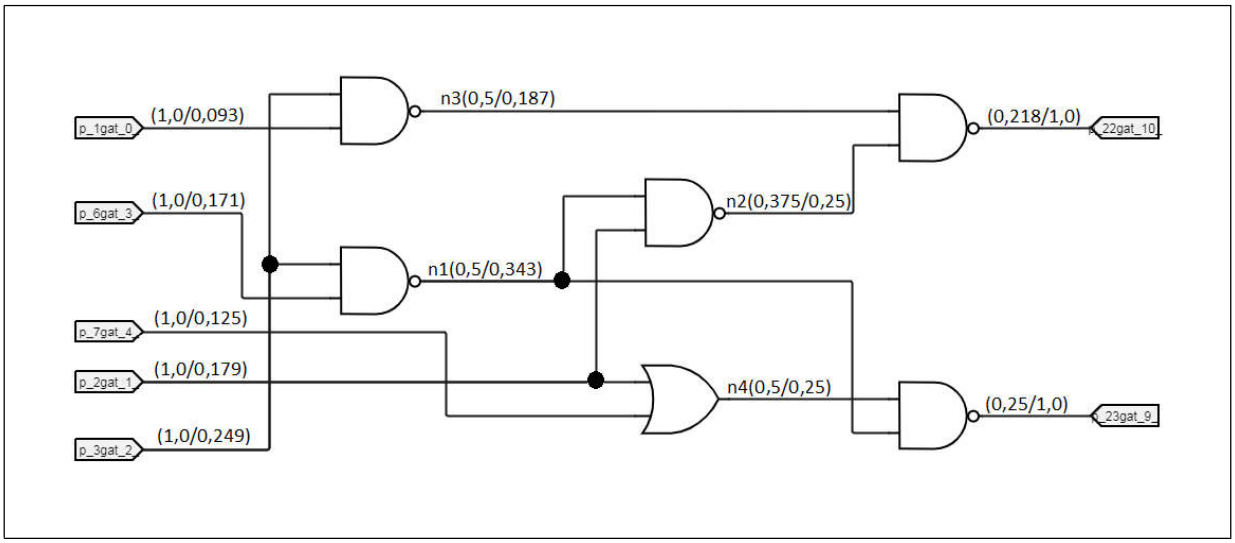
ANEXO B – Circuitos com o cálculo de *CY* e *OY* para o CAMELOT

Figura 30 – *Cálculo da testabilidade do c17-v1 para o CAMELOT*



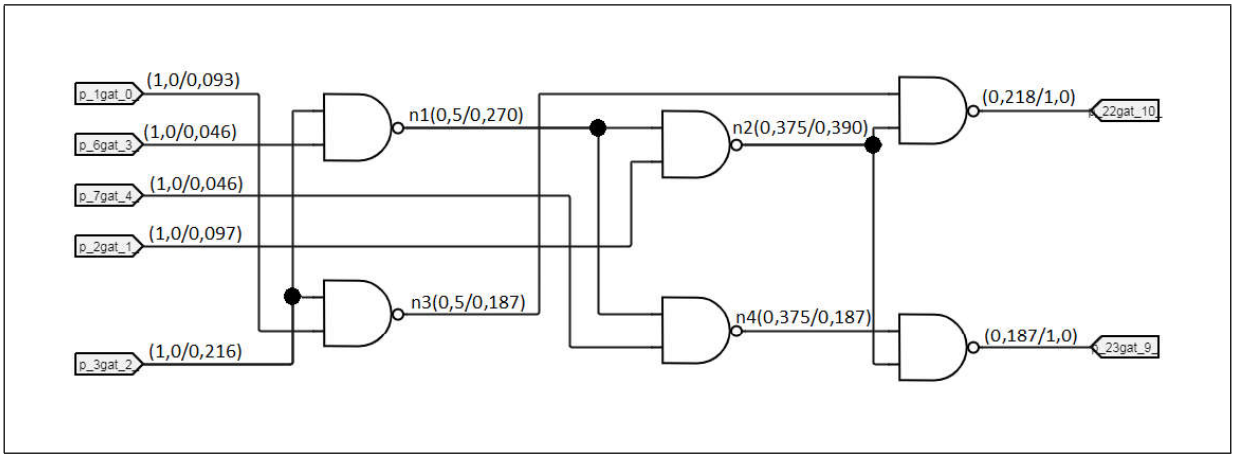
Fonte: Elaborado pelo Autor

Figura 31 – *Cálculo da testabilidade do c17-v2 para o CAMELOT*



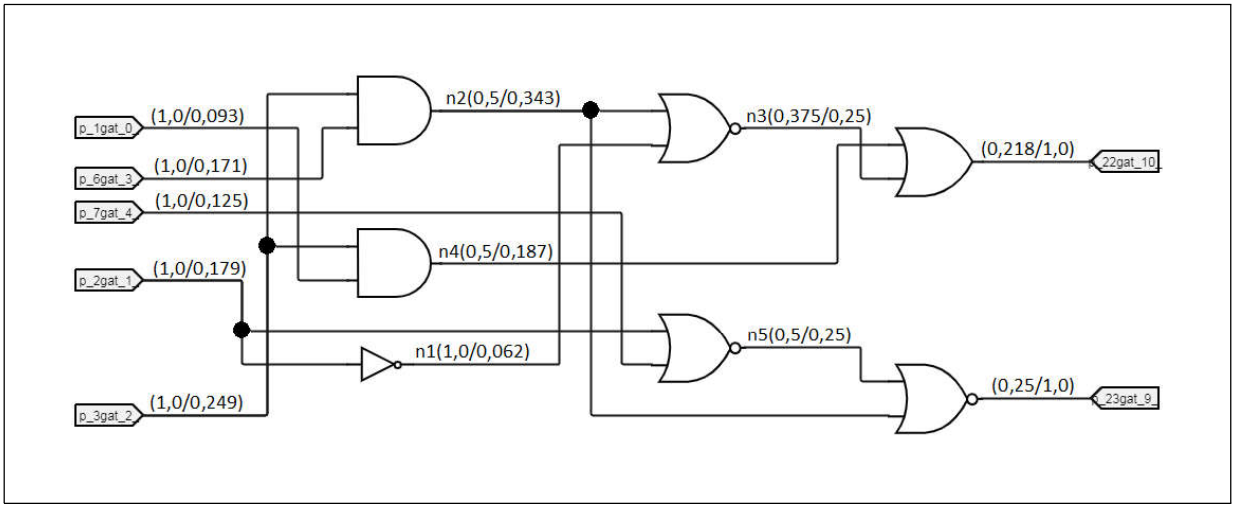
Fonte: Elaborado pelo Autor

Figura 32 – *Cálculo da testabilidade do c17-v3 para o CAMELOT*



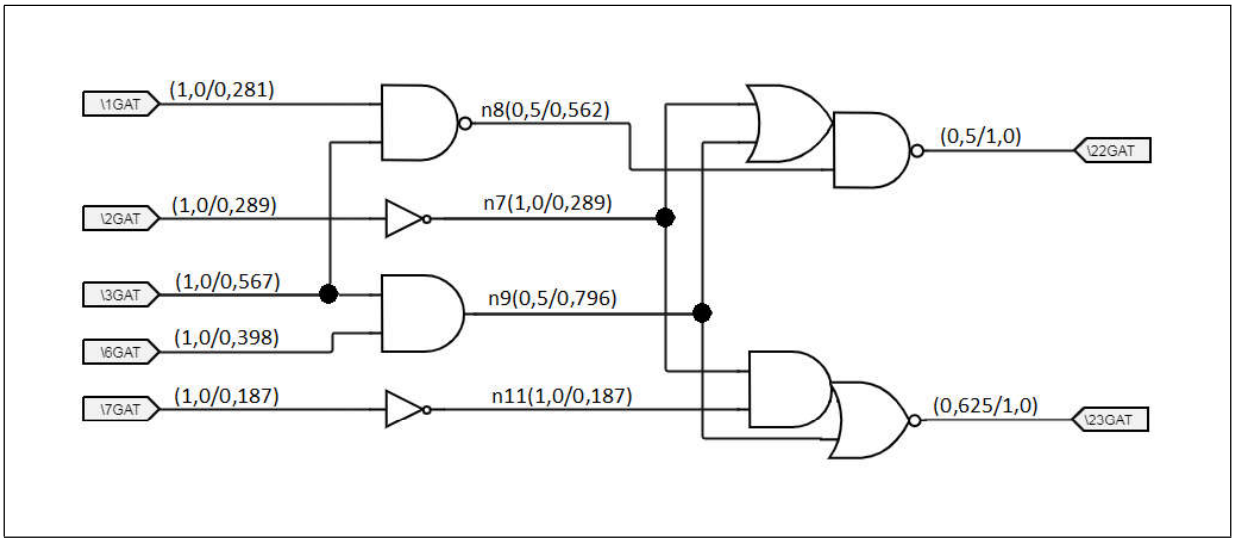
Fonte: Elaborado pelo Autor

Figura 33 – *Cálculo da testabilidade do c17-v4 para o CAMELOT*



Fonte: Elaborado pelo Autor

Figura 34 – *Cálculo da testabilidade do c17 da Cadence para o CAMELOT*



Fonte: Elaborado pelo Autor