

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

**Desenvolvimento de Ferramenta para Cálculo da
Confiabilidade de Circuitos Combinacionais Utilizando o
Método PTM**

Rafaél Ígor Fritz

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. Denis Teixeira Franco
Coorientador: Prof. Dr. Paulo Francisco Butzen

Rio Grande, 2017

Dados de catalogação na fonte:

colocar NOME DO BIBLIOTECÁRIO – CRB-colocar número do crb do bibliotecário
Biblioteca Central – FURG

A999a Fritz, Rafaél Ígor

Desenvolvimento de Ferramenta para Cálculo da Confiabilidade de Circuitos Combinacionais Utilizando o Método PTM / Rafaél Ígor Fritz. – Rio Grande, 2017. – 67 f: gráf. – Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande. Centro de Ciências Computacionais. Rio Grande, 2017. – Orientador Denis Teixeira Franco; Coorientador Paulo Francisco Butzen.

1. Confiabilidade. 2. Circuitos Combinacionais. 3. Método PTM. I. Franco, Denis Teixeira. II. Butzen, Paulo Francisco. III. Título.

CDD: 999.9



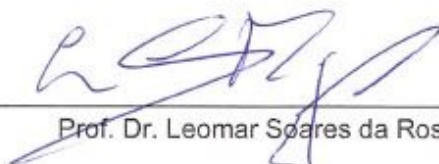
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

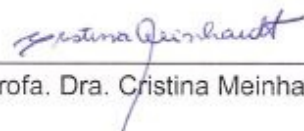
Desenvolvimento de Ferramenta para Cálculo da Confiabilidade de Circuitos Combinacionais Utilizando o Método PTM

Rafaél Igor Fritz

Banca examinadora:



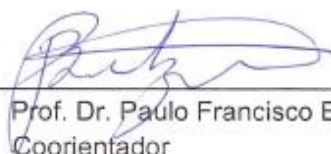
Prof. Dr. Leomar Soares da Rosa Junior



Profa. Dra. Cristina Meinhardt



Prof. Dr. Denis Teixeira Franco
Orientador



Prof. Dr. Paulo Francisco Butzen
Coorientador

*Dedico este trabalho à minha mãe Renita Renata Fritz
e à minha esposa Daiana Schons.*

AGRADECIMENTOS

Gostaria de agradecer em primeiro lugar a Deus por permitir que concluísse esse trabalho. À minha esposa Daiana que esteve sempre ao meu lado apoiando e incentivando nos momentos difíceis. À minha mãe que sempre me incentivou a continuar estudando e não mediu esforços permitindo eu ter chegado a esse momento tão especial.

Em especial o meu agradecimento ao meu Orientador, Denis Teixeira Franco e ao Coorientador Paulo Francisco Butzen, pela brilhante forma que orientaram esse trabalho e estiveram sempre prontos para esclarecer dúvidas referente às diferentes etapas do desenvolvimento da ferramenta, e que me mostraram uma forma diferente de se desenvolver um *software* e de ver as coisas com outro olhar.

Meus agradecimentos ao Fabian, diretor do IFSul - campus Jaguarão, que permitiu a minha redução de carga horária para que pudesse realizar as disciplinas e demais atividades do mestrado, sempre muito compreensível. Aos amigos e colegas de trabalho que acompanharam de alguma forma esse trabalho.

RESUMO

FRITZ, Rafaél Ígor. **Desenvolvimento de Ferramenta para Cálculo da Confiabilidade de Circuitos Combinacionais Utilizando o Método PTM**. 2017. 67 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande, Rio Grande.

A evolução dos sistemas computacionais embarcados tem tornado os chips mais complexos e com elevado número de funcionalidades em um mesmo circuito integrado (CI). Em contrapartida, o custo final de mercado tem diminuído significativamente em relação à maior complexidade do CI. À medida que as dimensões e as tensões de operação da microeletrônica de computadores são reduzidas para satisfazer a demanda insaciável do consumidor, a sensibilidade à radiação desses circuitos aumentam. As partículas ionizantes, ao incidirem em uma região sensível de um semicondutor, como as áreas ativas dos transistores, podem causar um pulso transiente, que, por sua vez, pode alterar o estado lógico do circuito. O desenvolvimento da indústria de semicondutores possibilitou o desenvolvimento de sistemas eletrônicos complexos. Hoje em dia, sistemas como satélites e sistemas aviônicos requerem alto grau de confiabilidade. Com a existência de sistemas que necessitam de um alto grau de confiabilidade, faz-se necessário ferramentas de projeto que possibilitem determinar a confiabilidade dos circuitos. Nesse trabalho, foi implementado o método de Matriz de Transferência Probabilística (PTM), que possibilita determinar a confiabilidade de circuitos. A ferramenta desenvolvida possibilitou a realização do cálculo da confiabilidade de circuitos lógicos combinacionais a partir da leitura do arquivo de descrição de hardware (Netlist) de um circuito.

Palavras-chave: Confiabilidade, Circuitos Combinacionais, Método PTM.

ABSTRACT

FRITZ, Rafaél Ígor. **Tool Development to Calculate the Reliability of Combinational Circuits Using the PTM Method**. 2017. 67 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande, Rio Grande.

The evolution of embedded computing systems has made the chips more complex and with a high number of functionalities in the same integrated circuit (IC). In contrast, the final market cost has decreased significantly in relation to the increased IC complexity. As the dimensions and operating voltages of computer electronics are reduced to satisfy insatiable consumer demand, the radiation sensitivity of such circuits increase. The ionizing particles, when they occur in a sensitive region of a semiconductor, as the active areas of transistors, can cause a transient pulse, which, in turn, can change the logical state of the circuit. The development of the semiconductor industry has enabled the development of complex electronic systems. Nowadays, systems such as satellites and avionics systems require a high degree of reliability. With the existence of systems that require a high degree of reliability, there's a need for design tools that allow to determine the reliability of the circuits. In this work, the Probabilistic Transfer Matrix (PTM) method was implemented, which makes possible to determine the circuit reliability. The developed tool made possible to perform the computation of the reliability of combinational logic circuits from the reading of the hardware description file (Netlist) of a circuit.

Keywords: Reliability, Combinational Circuits, PTM method.

LISTA DE FIGURAS

Figura 1	Transistores em microprocessadores Intel (WESTE; HARRIS, 2005).	16
Figura 2	Representação dos planos <i>pull-up</i> e <i>pull-down</i> .	20
Figura 3	Estrutura física dos transistores NMOS e PMOS (BUTZEN et al., 2012).	20
Figura 4	Simbologia dos transistores NMOS e PMOS (BUTZEN et al., 2012).	20
Figura 5	Fluxo de Síntese RTL (WESTE; HARRIS, 2005).	22
Figura 6	Comparação dos últimos ciclos solares (SOLEN, 2017).	24
Figura 7	Número de manchas solares ao longo dos anos (SOLEN, 2017).	25
Figura 8	Radiação induzida no óxido do gate, em (a) operação normal, em (b) após a irradiação (OLDHAM; MCLEAN, 2003).	26
Figura 9	Modelo de três universos: falha, erro e defeito	27
Figura 10	Mascaramento lógico (KARNIK; HAZUCHA, 2004)	28
Figura 11	Mascaramento elétrico (KARNIK; HAZUCHA, 2004)	28
Figura 12	Mascaramento temporal (KARNIK; HAZUCHA, 2004)	28
Figura 13	Matriz ITM e PTM da porta lógica OR	30
Figura 14	Matriz ITM da divisão de sinal, cruzamento de fios e interconexões	30
Figura 15	Matrizes PTM das portas NOT, AND e OR	31
Figura 16	Divisão do circuito em níveis	31
Figura 17	Representação de matriz com ADDs (BAHAR et al., 1993).	33
Figura 18	Um exemplo de circuito constituído por quatro <i>macro-gates</i> (XIAO et al., 2011).	34
Figura 19	Circuito genérico de lógica combinacional.	34
Figura 20	Representação matricial das probabilidades de sinal de quatro estados.	36
Figura 21	Exemplo de propagação da probabilidade do sinal em uma porta XOR.	36
Figura 22	Exemplo de propagação da probabilidade do sinal em um circuito (FRANCO et al., 2008).	37
Figura 23	Visão geral da ferramenta	40
Figura 24	Circuito exemplo.	41
Figura 25	Descrição do circuito da Figura 24.	41
Figura 26	Processos do gerador de código	42
Figura 27	Apresentação das partes que compõe a ferramenta.	43
Figura 28	Verilog Circuito C17.	45
Figura 29	Circuito C17 analisado pela ferramenta.	45
Figura 30	Descrição do Circuito C17.	45
Figura 31	Código Scilab para C17 gerado pela Ferramenta.	46

Figura 32	Variando a confiabilidade para diferentes implementações do C17. . .	47
Figura 33	Comparação de tempo de execução DCI e GCS.	49
Figura 34	Comparação de tempo de execução DCI, GCS e Scilab.	50
Figura 35	Tempo de execução com a variação de q	50
Figura 36	Tempo de execução com a variação de q 0,90 até 1,0.	51
Figura 37	Comparação do consumo de memória.	52
Figura 38	Tamanho de matriz em função do número de entradas do circuito. . .	52
Figura 39	Análise de Três Full Adder com variação de q	53
Figura 40	Análise de Três Full Adder com q fixo em 0,99.	54
Figura 41	Outros circuitos analisados.	55

LISTA DE TABELAS

Tabela 1	Confiabilidade para diferentes implementação do C17.	48
Tabela 2	Diferentes versões C17 com variação de 0,999999 a 0,99	48
Tabela 3	Dados das diferentes versões do C17.	48
Tabela 4	Dados com variação de 0,999999 a 0,99.	53
Tabela 5	Dados das três versões de Full Adder.	54
Tabela 6	Resultados com q variando de 0,99 a 0,999999.	55
Tabela 7	Confiabilidade para diferentes circuitos.	56
Tabela 8	Dados dos circuitos analisados.	56

LISTA DE ABREVIATURAS E SIGLAS

ADDs	<i>Algebraic Decision Diagrams</i>
ASICs	<i>Application Specific Integrated Circuits</i>
CI	Circuito Integrado
CMOS	<i>Complementary Metal-Oxide Semiconductor</i>
DCI	Descrição do Circuito Intermediário
DFS	<i>Depth-First Search</i>
ER	<i>Regular Expression</i>
FIFA	<i>Fault Injection Fault Analysis</i>
FPGA	<i>Field Programmable Gate Array</i>
GCS	Gerador de Código Scilab
HDL	<i>Hardware Description Language</i>
ITM	Matriz de Transferência Ideal (<i>Ideal Transfer Matrix</i>)
MOS	<i>Metal-Oxide-Semiconductor</i>
NMOS	<i>N-type Metal-Oxide Semiconductor</i>
OTC	<i>Over-The-Cell</i>
PBR	<i>Probabilistic Binomial Model</i>
PMOS	<i>P-type Metal-Oxide Semiconductor</i>
PTM	Matriz de Transferência Probabilística (<i>Probabilistic Transfer Matrix</i>)
q	<i>Gate reliability</i>
RTL	<i>Register Transfer Level</i>
SET	<i>Single Event Transient</i>
SEU	<i>Single Event Upset</i>
SOC	<i>System On Chip</i>
SPR	<i>Signal Probability Model</i>
TID	<i>Total Ionizing Dose</i>
1 - q, \bar{q}	<i>Gate unreliability</i>

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Objetivos do trabalho	17
1.2	Organização do texto	18
2	REFERENCIAL TEÓRICO	19
2.1	Tecnologia CMOS	19
2.2	Metodologia <i>Standard Cell</i>	20
2.2.1	Biblioteca de Células	21
2.2.2	Fluxo de Síntese	21
2.3	Falhas em Sistemas Digitais	22
2.3.1	Falhas Permanentes e/ou por Envelhecimento	23
2.3.2	Falhas Transientes	23
2.3.3	Efeitos da Radiação em Semicondutores	24
2.4	Falha Erro e Defeito	26
2.5	Mascaramento	27
2.5.1	Mascaramento Lógico	27
2.5.2	Mascaramento Elétrico	27
2.5.3	Mascaramento Temporal	28
3	ANÁLISE DA CONFIABILIDADE	29
3.1	Método PTM	29
3.2	Modelo PBR	34
3.3	Modelo SPR	35
3.3.1	SPR <i>Multi-pass</i>	37
3.4	Plataforma FIFA	38
4	MÉTODO IMPLEMENTADO	39
4.1	Detalhes da implementação	39
4.1.1	Processamento do Verilog HDL	40
4.1.2	Descrição do Circuito Intermediário (DCI)	41
4.1.3	Gerador de Código Scilab (GCS)	42
4.2	Outros detalhes da implementação	43
4.2.1	Biblioteca	44
4.2.2	Gerador ITM	44
4.3	Estudo de Caso	44
4.3.1	Verilog do Circuito C17	44
4.3.2	Diagrama do Circuito Analisado	44

4.3.3	Descrição do Circuito Intermediário	44
4.3.4	Gerador Código Scilab	45
5	RESULTADOS	47
5.1	Identificação da confiabilidade de mesmos circuitos com implementações diferentes	47
5.2	Comparação do tempo de execução	49
5.2.1	Diferenças no tempo de execução	50
5.3	Comparação de consumo de memória	51
5.4	Tamanho das matrizes	52
5.5	Circuitos analisados	53
6	CONSIDERAÇÕES FINAIS	57
	REFERÊNCIAS	59
	APÊNDICE A CIRCUITOS ANALISADOS	63

1 INTRODUÇÃO

A eletrônica começou a se destacar na metade do século XX e influenciou áreas da indústria, comunicação e tecnologia espacial. O ponto de partida ocorreu com o advento da válvula na década de 40, dispositivo que trabalhava com tensões elétricas altas, aquecia bastante e costumava queimar com pouco tempo de funcionamento. Em 1946, surge o ENIAC, um sistema completamente eletrônico da Universidade da Pensilvânia, composto de 18 mil válvulas e pesando 30 toneladas. Após, em 1947, um dispositivo denominado de transistor começou a substituir a válvula. Esse dispositivo possuía propriedades físicas especiais. Ele é um componente de estado sólido e não aquecia como a válvula, além de ser muito menor e não necessitar de tensão de filamento para iniciar o seu funcionamento. A válvula foi substituída gradativamente por transistores que consumiam menos energia elétrica e ocupavam menor espaço físico. Em 1958, o primeiro Circuito Integrado (CI) foi desenvolvido pela *Texas Instruments*. Três anos depois, o primeiro CI comercial foi fabricado pela *Fairchild Corporation*, o qual possuía vários transistores em um só componente, dentro de uma só pastilha de silício. Esses circuitos ficaram conhecidas como *Chips*. Foi com o início da era do estado sólido em que os computadores mudaram de tamanho, velocidade e capacidade (RABAEY; CHANDRAKASAN; NIKOLIC, 2002) (AYERS, 2009).

Em 1965, Gordon E. Moore previu que a cada período de 12 meses o número de transistores de um *chip* aumentaria em 100%, mantendo os custos de fabricação constante (MOORE, 1998). Em 1975, Moore reviu sua previsão para um aumento de 100% no número de transistores a cada período de 24 meses. Assim, a Lei de Moore ficou conhecida e começou a guiar a evolução tecnológica. Logo, a indústria de semicondutores passou a se basear na Lei de Moore para o desenvolvimento de novos produtos. A Figura 1 ilustra o crescente número de transistores nos microprocessadores Intel.

A evolução dos sistemas computacionais embarcados tem tornado os *chips* mais complexos e com elevado número de funcionalidades. Em contrapartida, o custo final de mercado dos *chips* tem diminuído significativamente. Esses dispositivos estão muito presentes no nosso dia a dia, em aparelhos celulares, fornos de microondas, computadores de bordo automotivos, no controle de temperatura de ar-condicionado, em MP3 players,

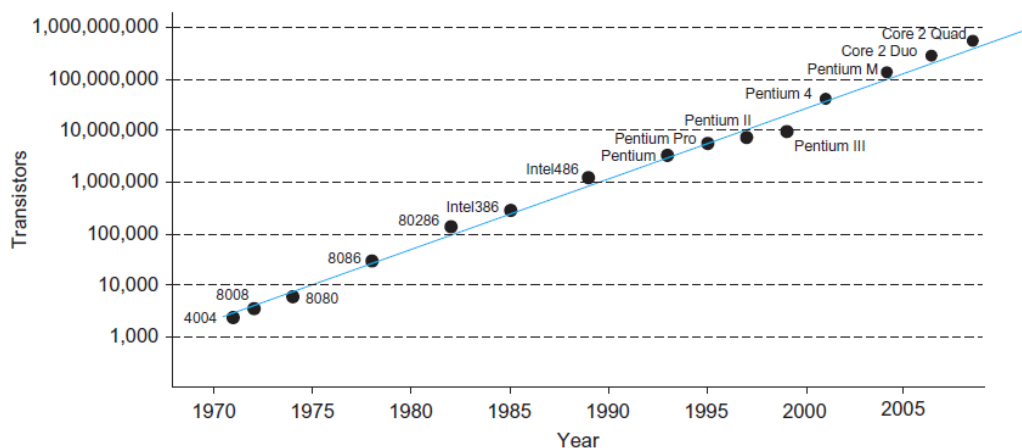


Figura 1: Transistores em microprocessadores Intel (WESTE; HARRIS, 2005).

equipamentos de rede, sistemas de monitoramento médico, equipamentos portáteis de medição e impressoras. Segundo alguns dados estimados por pesquisas em alta tecnologia, mais de 90% dos microprocessadores fabricados mundialmente são destinados a máquinas que usualmente não são chamadas de computadores (CHASE; ALMEIDA, 2007).

Os custos dos circuitos integrados caíram exponencialmente apesar do processo básico de manufatura do silício não ter mudado. Os principais fatores que influenciam no custo do CI são: custo do *die*, encapsulamento e rendimento do *die* (PATTERSON; HENNESSY, 2014). Essa redução contribui para o desenvolvimento da indústria de semicondutores. Hoje em dia, sistemas como satélites e sistemas aviônicas requerem alto grau de confiabilidade. Os avanços da tecnologia empregada em circuitos integrados visam à redução da dimensão dos transistores com o objetivo de reduzir o consumo de energia e melhorar o desempenho e a densidade dos sistemas. O número de transistores em um circuito integrado vem aumentando graças à miniaturização desses componentes, e que possibilitam assim uma integração de muitas funcionalidades em um mesmo circuito integrado.

À medida que os transistores se tornam menores, dissipam menos energia e são mais baratos de fabricar. Desde 1995, o ritmo da inovação realmente se acelerou por causa da concorrência da indústria. A redução das dimensões dos transistores é sem precedentes na história da tecnologia (WESTE; HARRIS, 2005). No entanto, esta redução também introduz problemas de confiabilidade, aumenta a complexidade e dificulta a fabricação e teste dos circuitos. Os projetistas precisam ser capazes de prever o efeito desse dimensionamento no desempenho do *chip* para planejar futuros produtos (WESTE; HARRIS, 2005).

A miniaturização que ocorreu nos últimos 30 anos tem buscado constantemente a redução dos custos, além de aumentar o desempenho dos *chips*. Infelizmente, uma consequência direta da miniaturização dos transistores é o aumento da vulnerabilidade dos

dispositivos à radiação. Esta vulnerabilidade é causada pela redução no nível de energia necessário para causar alterações nos estados dos transistores (condução e corte). Dessa forma, impactos originados por partículas energizadas, que eram desprezíveis, agora podem produzir falhas. À medida que as dimensões e as tensões de operação de dispositivos são reduzidas para satisfazer à demanda do consumidor por maior densidade, funcionalidade e menor potência, sua sensibilidade à radiação aumenta dramaticamente (BAUMANN, 2005). O *Scaling* da tecnologia faz com que tenhamos maior necessidade por confiabilidade. Com a evolução da tecnologia e da indústria de semicondutores, os circuitos têm se tornado cada vez mais complexos, e com isso, aumenta a dificuldade em calcular a confiabilidade desses circuitos. A confiabilidade pode ser definida como a probabilidade de um sistema executar sua função requerida, em condições determinadas, por um determinado período de tempo (LALA, 2001).

As partículas ionizantes, ao incidirem em uma região sensível de um semicondutor, como a região ativa dos transistores, podem causar um pulso transiente, que, por sua vez, pode alterar o estado lógico do circuito, propagar-se e ser capturado por um elemento de memória, o qual produz uma falha transiente (SHIVAKUMAR et al., 2002).

Nos últimos anos, algumas técnicas foram propostas com o objetivo de melhorar a confiabilidade dos circuitos integrados. Essas técnicas sugerem diferentes tipos de redundância como: redundância temporal, espacial e de informações. Normalmente, essas técnicas acabam adicionando custo ao sistema. É o caso da redundância de *hardware* TMR (*Triple Modular Redundancy*), que aumenta em três vezes a área do circuito. Dessa maneira, saber onde aplicar as formas de redundância é importante. Para isso, é possível utilizar métodos probabilísticos.

Um dos métodos que permite avaliar a confiabilidade de um circuito é o de Matrizes de Transferência Probabilística (PTM), em que as portas lógicas são representadas por matrizes. De acordo com (PATEL; MARKOV; HAYES, 2003), os métodos probabilísticos podem ser modelados como uma matriz. O método PTM é considerado um método preciso para determinar a confiabilidade de circuitos.

1.1 Objetivos do trabalho

O desenvolvimento de técnicas de tolerância a falhas vem se tornando uma tarefa essencial com a entrada na era nanométrica. No entanto, antes de considerar o projeto tolerante a falhas, é pertinente a existência de um método preciso de avaliação da confiabilidade. Nesse contexto, faz-se necessário o desenvolvimento de ferramentas capazes de agilizar e auxiliar no processo de determinação da confiabilidade dos circuitos. O método de Matrizes de Transferência Probabilística (PTM) calculam a confiabilidade de um circuito composto de portas lógicas realizando operações matemáticas entre as matrizes.

Este trabalho tem por objetivo desenvolver uma ferramenta que possibilite a realização

do cálculo da confiabilidade de circuitos lógicos combinacionais, ao utilizar estruturas de dados que permitam a extração automática de informações do circuito, a partir de uma HDL (*Hardware Description Language*).

1.2 Organização do texto

A dissertação está organizada da seguinte forma: O Capítulo 2, trata sobre os conceitos que envolvem a tecnologia CMOS como também a metodologia empregada na concepção de circuitos integrados. Em seguida, o Capítulo 3, traz um estudo sobre alguns dos métodos/modelos que permitem determinar a confiabilidade de um circuito. No Capítulo 4, tratamos especificamente sobre o método PTM (*Probabilistic Transfer Matrices*), que será implementado. Também traz um estudo de caso sobre o funcionamento da ferramenta desenvolvida, aplicando o método PTM. Na sequência, o Capítulo 5, apresenta os resultados obtidos das análises de circuitos utilizando a ferramenta desenvolvida. Por fim, o Capítulo 6, relata as conclusões sobre o trabalho e possibilidades para aumentar a escalabilidade do método, incluindo considerações sobre possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

Esta Seção aborda os conceitos que envolvem a tecnologia CMOS (*Complementary Metal-Oxide-Semiconductor*) e apresenta a diferença existente nos transistores MOSFET de canal N e P, e como são utilizados para formarem portas lógicas.

2.1 Tecnologia CMOS

A tecnologia CMOS (*Complementary Metal-Oxide-Semiconductor*) tornou-se a tecnologia dominante utilizada na fabricação de Circuitos Integrados (CI). O *complementary* em seu nome se dá pelo fato da utilização de dois tipos de transistores MOSFET, sendo um de canal N e outro de canal P, um tendo comportamento complementar ao o outro com objetivo de implementar funções lógicas.

O processo de fabricação CMOS dispõe de dois tipos de transistores: PMOS (*P-type Metal-Oxide Semiconductor*) e NMOS (*N-type Metal-Oxide Semiconductor*). A tecnologia CMOS possui duas redes lógicas independentes e complementares constituídas por transistores PMOS e NMOS. Os transistores PMOS são responsáveis pela conexão do sinal de saída à linha de alimentação positiva ou VDD (valor lógico 1), conhecida como rede *pull-up*. Os transistores NMOS são responsáveis pela conexão do sinal de saída à linha de referência terra ou GND (valor lógico 0), conhecida como rede *pull-down*. A representação destas redes pode ser vista na Figura 2, onde, E1 e EN representam as entradas de uma porta lógica

A estrutura básica de um transistor MOS (*Metal-Oxide-Semiconductor*) possui quatro terminais: *source* (S), *drain* (D), *gate* (G) e *bulk* (B), exemplificados na Figura 3. A principal diferença entre os transistores MOS ocorre na dopagem. O NMOS apresenta dopagem negativa (N+) no *drain* e *source* e dopagem positiva (P) no substrato/canal, enquanto que o transistor PMOS tem a dopagem oposta.

Os circuitos CMOS estáticos com redes complementares NMOS *pull-down* e PMOS *pull-up* são usados para a grande maioria das portas lógicas em circuitos integrados. Eles têm boas margens de ruído e são rápidos, de baixa potência, fáceis de projetar e facilmente disponíveis em bibliotecas de células padrão (WESTE; HARRIS, 2005).

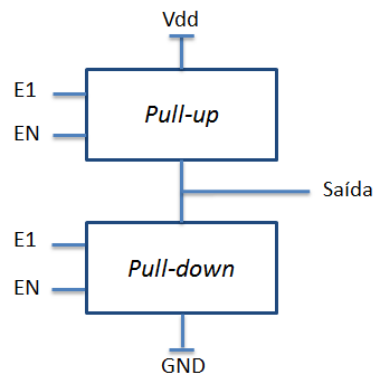


Figura 2: Representação dos planos *pull-up* e *pull-down*.

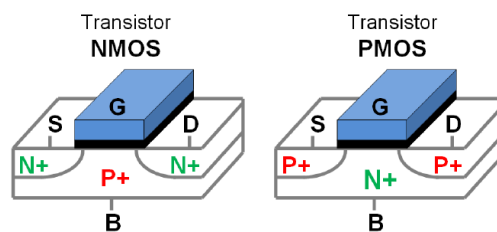


Figura 3: Estrutura física dos transistores NMOS e PMOS (BUTZEN et al., 2012).

Infelizmente os transistores não podem ser considerados chaves ideais devido a algumas características elétricas intrínsecas. A partir de uma tensão específica aplicada ao *gate* do transistor acontece a troca do seu modo de operação que será condução (chave fechada) ou corte (chave aberta). Esta tensão é denominada tensão de limiar ou tensão de *threshold* (V_{th}). A figura 4 apresenta a simbologia destes transistores e seu comportamento como chave lógica de acordo com o sinal no terminal *gate*.

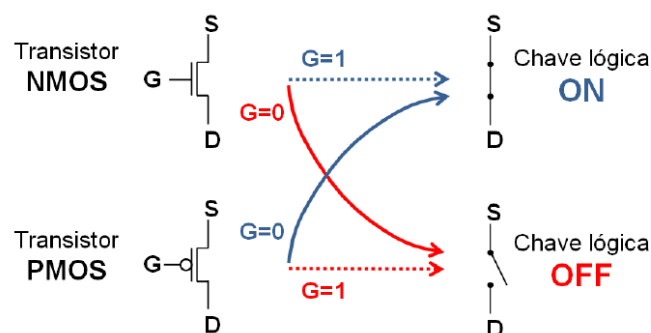


Figura 4: Simbologia dos transistores NMOS e PMOS (BUTZEN et al., 2012).

2.2 Metodologia *Standard Cell*

A utilização de *software* para auxiliar o projeto de circuitos integrados surgiu da necessidade de resolver problemas mais complexos em menos tempo. Assim, a utilização

de um estilo de projeto fornece uma estrutura. Dentre os populares estilos de projeto de circuitos integrados temos a metodologia *Standard Cell*.

A metodologia *Standard Cell* faz uso de células previamente projetadas com funções lógicas bem definidas considerando os diferentes cenários de funcionamento. Esta metodologia é utilizada para projetar circuitos integrados para aplicações específicas, *Application Specific Integrated Circuits (ASICs)*. Este fluxo de projeto é muito utilizado na concepção de circuitos *Very-Large-Scale Integration (VLSI)*, devido à relação entre custo, tempo de projeto e desempenho.

Em um ASIC, o circuito pode ser desenvolvido no nível HDL (*Hardware Description Language*) e, em seguida, enviado para uma empresa que desenvolve o projeto do chip propriamente dito. Desta forma, a empresa que descreveu o circuito não precisa investir pessoal ou ferramentas necessárias para traduzir uma especificação HDL em um chip físico.

2.2.1 Biblioteca de Células

O leiaute dos transistores formam células pré-desenhadas e pré-caracterizadas que possuem algumas funções lógicas básicas como: portas lógicas, flip-flops e outros circuitos simples. Algumas versões de leiaute podem ser desenhadas implementando a mesma função lógica, formando assim, um conjunto de leiautes que dá origem a *Standard Cells Library*.

O projeto de circuitos utilizando a abordagem *Standard Cell* segue uma organização, onde as células são posicionadas em bandas de alturas iguais e interconectadas por canais de roteamento ou por roteamento sobre as células *Over-The-Cell (OTC)* (RABAEY, 1996), (SARRAFZADEH, 1996). Em um fluxo de projeto baseado em células é necessário conhecer os valores de área, atraso e consumo da célula. Essa caracterização geralmente é realizada através de ferramentas de simulação elétrica.

2.2.2 Fluxo de Síntese

O estilo mais popular de ferramentas que realizam a síntese comportamental é o que transforma uma descrição RTL (*Register Transfer Level*) comportamental para um *netlist* estrutural de portas lógicas (WESTE; HARRIS, 2005). A Figura 5, apresenta um fluxo comportamental típico para um ASIC. Dentre os principais fornecedores de ferramentas estão: *Synopsys*, *Cadence Design Systems* e *Mentor Graphics*.

O início do processo de *design* ocorre com uma especificação. Essa especificação normalmente é feita em formato texto ou em alguma linguagem de especificação de sistema. Após essa especificação converte-se para uma descrição de comportamento RTL, como Verilog ou VHDL. Em seguida, um conjunto de padrões de teste é então construído e o HDL é simulado para verificar o comportamento correto conforme definido pela especificação e os requisitos do produto. O próximo passo é sintetizar a descrição com-

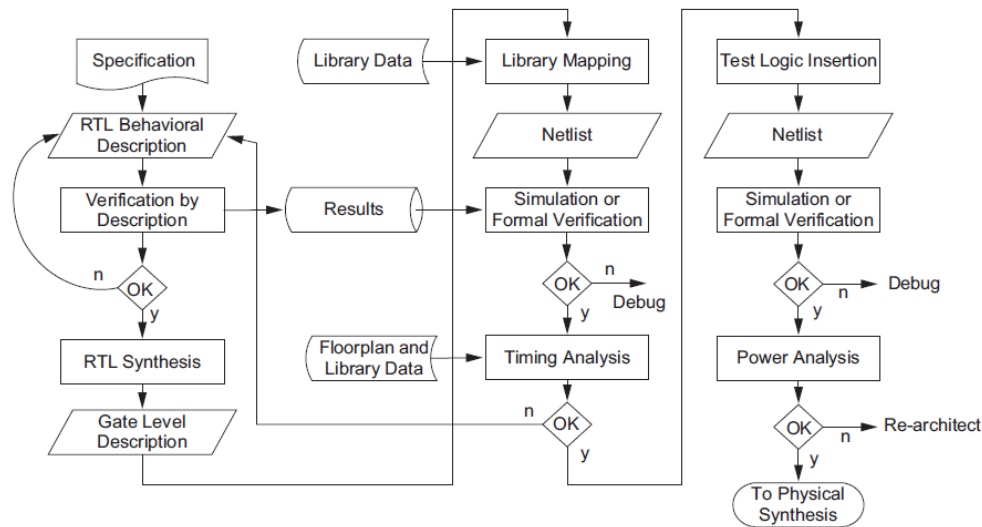


Figura 5: Fluxo de Síntese RTL (WESTE; HARRIS, 2005).

portamental. Isso envolve a conversão do RTL em portas genéricas e registros, otimizando a lógica para melhorar a velocidade e a área e mapeando as portas lógicas genéricas para uma biblioteca de células padrão. Outras etapas envolvidas nesta fase são a decomposição de máquina de estado, otimização de *datapath* e otimização de energia. Esse processo é realizado por produtos como *Design Compiler da Synopsys* e *RTL Compiler da Cadence* (WESTE; HARRIS, 2005). Para finalizar, testes são realizados para provar que o *netlist* estrutural desempenha a mesma função que o HDL comportamental original.

2.3 Falhas em Sistemas Digitais

As técnicas de detecção de erros fornecem a informação essencial que permite um maior diagnóstico e controle, levando a uma melhor robustez do sistema. Esta informação pode ser integrada com técnicas de *design* adaptativo, como a tensão dinâmica e a variação de frequência, para ajustar o sistema e reduzir a taxa de erro durante o tempo de execução. (CAO; TSCHANZ; BOSE, 2009).

As falhas podem ser classificadas em dois grupos: *software* e *hardware*. As falhas de *software* devem ser previstas e prevenidas através da aplicação da engenharia de *software* no desenvolvimento de aplicativos. Já as falhas de *hardware* se encontram relacionadas com os componentes do sistema computacional e podem acontecer em diferentes níveis. Estas podem ser divididas em falhas de projeto, falhas de portas lógicas e falhas em transistores (ZIMPECK; MEINHARDT; BUTZEN, 2014). As falhas relacionadas ao *hardware* serão abordadas com mais detalhes nesse trabalho.

As falhas podem ser consideradas como uma operação incorreta do sistema, que pode ter como origem interferência externa. Os dispositivos apresentam sensibilização à radiação e a interferência externa. Essa sensibilidade ocorre à medida que as dimensões

dos transistores diminuem. Outro fator que também colabora para a sensibilização é a diminuição da tensão de operação dos transistores. Assim, surge a necessidade de estudar os efeitos e analisar como essa sensibilização pode vir a afetar os circuitos. As falhas podem ser classificadas em falhas permanentes, intermitentes e transientes.

2.3.1 Falhas Permanentes e/ou por Envelhecimento

O efeito do envelhecimento nos circuitos também tem uma grande influência no desempenho dos transistores. A tendência é que esta degradação aumente à medida que novas tecnologias de transistores forem criadas (CAO; TSCHANZ; BOSE, 2009). Existem várias pesquisas que tem mostrado que a corrente de saturação do transistor degrada ao longo dos anos devido ao desgaste do óxido (MATTOS; ROSA JR; PILLA, 2009).

As falhas permanentes ocorrem quando um componente falha e não volta a funcionar de forma correta. Esta falha seguirá ocorrendo até que alguma ação corretiva seja tomada. Caso nada seja feito continuará no mesmo estado. As falhas permanentes se originam do processo de manufatura ou por efeitos de envelhecimento. Esta falha faz com que o dispositivo fique permanentemente apresentando defeito, de maneira que, a falha modifica o comportamento do transistor. Assim, a saída do circuito poderá apresentar um resultado incorreto. A correção desse tipo de falha ocorre com a substituição do circuito.

2.3.2 Falhas Transientes

As falhas transientes normalmente tem origem em partículas de radiação externas. Essas partículas podem afetar tanto circuitos combinacionais como sequenciais.

Em circuitos integrados, as partículas de radiação pode gerar desde ruídos com baixa intensidade até pulsos tão grandes que podem danificar equipamentos eletrônicos de forma permanente (LUZ REIS, 2011). Quando essas partículas atingem o circuito, é gerado um pulso transiente. Este pulso transiente pode se propagar no circuito combinacional, e assim, causar um *Soft Error*. Esse tipo de falha é chamada de SET (*Single Event Transient*). O *Soft Error* também é frequentemente referido como um SEU (*Single Event Upset*). Um *Soft Error* ocorre quando um evento de radiação causa bastante perturbação de carga para inverter o estado de dados de uma célula de memória (BAUMANN, 2005). Um circuito sequencial também pode ser atingido pelas partículas de radiação, nesse caso, a falha é chamada de SEU (*Single Event Upset*) e, como resultado, poderá inverter um valor armazenado no circuito.

Os sistemas aviônicos contendo grande quantidade de memória, quando em altitudes de aeronaves, estão experimentando na ordem de um SEU por voo. Os dados existentes sugerem que estes SEUs são causados direta ou indiretamente por raios cósmicos, em particular pelos nêutrons atmosféricos gerados pelos raios cósmicos (TABER; NORMAND, 1993).

As falhas transitórias, as quais se manifestam por períodos de tempo muito curtos,

podem provir de diversas fontes como: nêutrons e partículas alfa da atmosfera com alta energia, ruído de acoplamento ou crosstalk, ruído térmico, variações de tensão devido a correntes de comutação, redução da tensão devido à resistência dos condutores, e ruído no substrato (FRANCO, 2008).

As falhas transientes possuem duração limitada e são causadas por um mau funcionamento temporário, sendo que estas também podem ser intermitentes oscilando entre um estado normal e de falha. Uma falha intermitente tem como característica ficar aparecendo e desaparecendo no sistema durante um período relativamente curto de tempo, mas esta falha não causa danos permanentes no circuito (BAUMANN, 2005).

2.3.3 Efeitos da Radiação em Semicondutores

Sabe-se que o sol apresenta um ciclo de atividade de 11 anos, durante o qual ocorre a multiplicação das tempestades solares, e a intensidade da atividade solar pode ser acompanhada através das manchas solares, as quais são causadas pelo campo magnético variável do Sol. Os registros dos ciclos iniciaram em 1755 e atualmente estamos no ciclo de número 24 que teve seu pico solar em 2010 ou 2011, não demonstrando ser um ciclo solar muito intenso. A Figura 6 demonstra a comparação dos últimos ciclos solares, com número de manchas solares mensal e número de meses após o início.

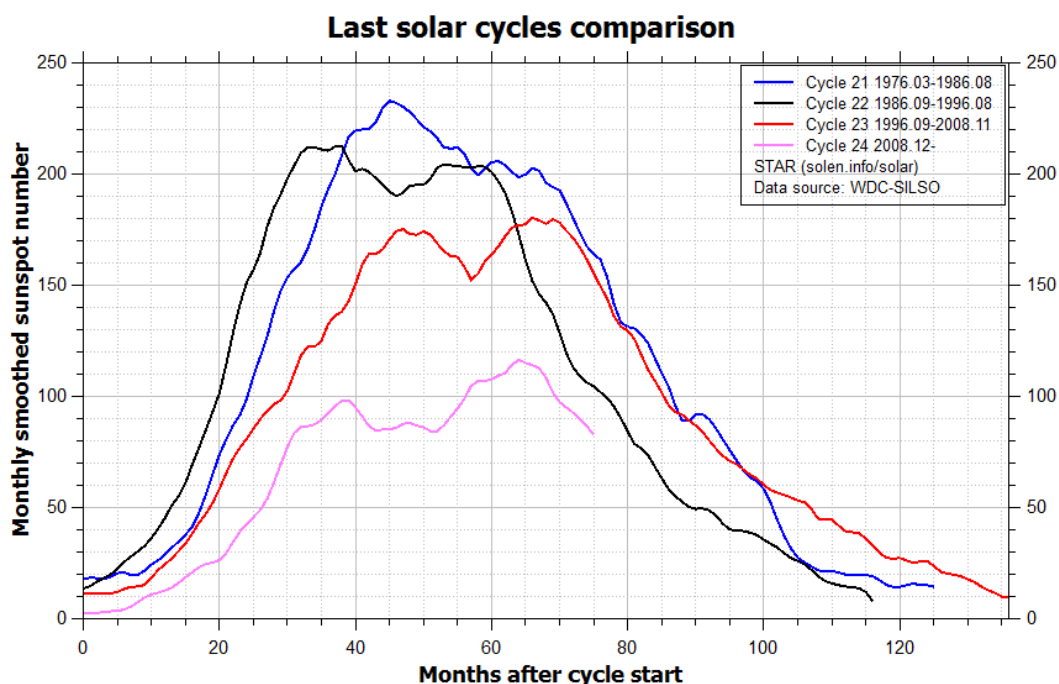


Figura 6: Comparação dos últimos ciclos solares (SOLEN, 2017).

Atualmente, a nossa compreensão limitada do ciclo solar não permite previsões de futuro da atividade solar, mas previsões são importantes para o planejamento e gestão de missões espaciais, comunicações e sistemas de energia (SVALGAARD; CLIVER;

KAMIDE, 2005). As manchas solares podem causar interferências nos sistemas de telecomunicações, e provocar danos em sistemas de distribuição de energia. A intensidade das manchas solares dos últimos anos pode ser vista na Figura 7, onde a projeção do número de manchas solares encontra-se em declínio.

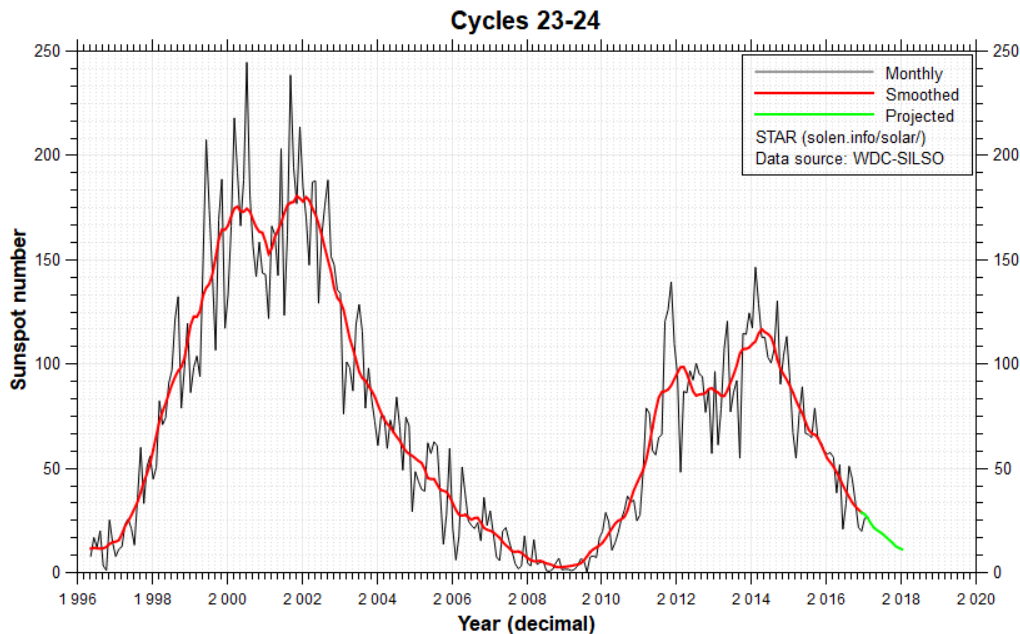


Figura 7: Número de manchas solares ao longo dos anos (SOLEN, 2017).

O satélite de telecomunicações Telstar lançado em 10 de julho de 1962, foi o primeiro a ser equipado com *transponders* de sinal com propósito de transmissão de televisão ao vivo. Na véspera do lançamento do Telstar, os Estados Unidos realizaram um teste nuclear em grande altitude que causou altos níveis de radiação induzida pelos elétrons injetados no cinto de radiação de Van Allen, o que causou degradação em certos componentes do Telstar. Após 7 meses e 11 dias de seu lançamento o satélite foi perdido. Desde então sabe-se que a radiação atua causando danos em dispositivos e circuitos eletrônicos (ECOFFET, 2007). Os danos em circuitos eletrônicos podem dar origem a falhas, as quais podem ter um comportamento transiente (ou transitório), intermitente ou permanente.

A exposição à radiação nos circuitos eletrônicos faz com que ocorra uma degradação no seu sinal elétrico, e isso ocorre devido a dois efeitos: efeitos cumulativos e efeitos singulares. Os efeitos cumulativos são decorrentes do acúmulo de defeitos nos dispositivos semicondutores ao longo do tempo, e sua origem se dá pela incidência de partículas ou componentes eletromagnéticos de radiação, que podem ionizar o material ou modificar o arranjo estrutural. Este tipo de efeito cumulativo pode ser dividido em efeitos de dose ionizante *Total Ionizing Dose* (TID) e danos por deslocamento *displacement damage* (SCHRIMPF, 2007). O efeito dose total ionizante faz com que os dispositivos semicondutores sofram absorção de energia e cargas nas camadas dielétricas do dispositivo. Já

os efeitos de danos por deslocamento provocam a interação da partícula incidente com a rede cristalina do material semiconductor, que danifica sua estrutura e modifica suas propriedades elétricas. Os efeitos singulares ocorrem quando partículas de alta energia incidem no material semiconductor, provocando a geração de pares elétron-lacuna, resultando em perturbações elétricas transitórias em regiões sensíveis do circuito, podendo ou não resultar em uma falha.

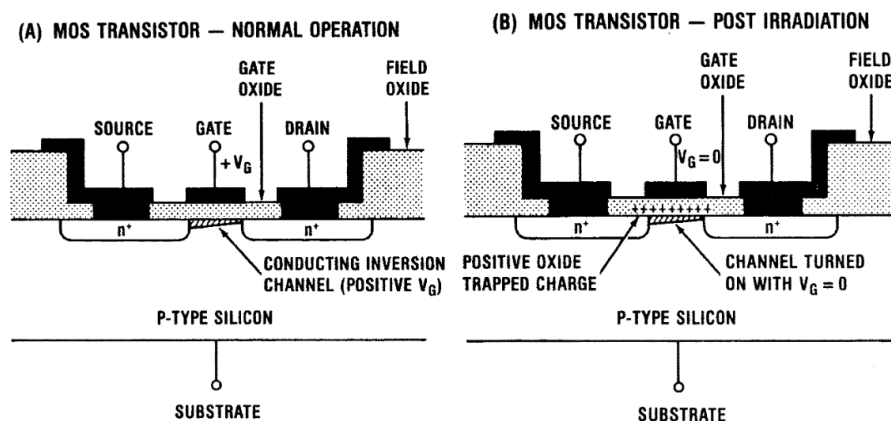


Figura 8: Radiação induzida no óxido do gate, em (a) operação normal, em (b) após a irradiação (OLDHAM; MCLEAN, 2003).

O Transistor MOS na Figura 8 demonstra o problema da radiação ionizante, onde em (a) demonstra o funcionamento normal e em (b) a ilustração do efeito da radiação ionizante, onde cargas residuais aparecem no óxido do *gate* que acaba deslocando a tensão de limiar, ou seja, provoca alteração na tensão que deve ser aplicada para ligar o dispositivo. Se este deslocamento é grande o suficiente, o dispositivo não pode ser desligado (OLDHAM; MCLEAN, 2003).

O objetivo na abordagem deste tema é apresentar alguns conceitos sobre os efeitos da radiação em semicondutores, mas não nos aprofundaremos mais nesse tema por fugir do escopo desse trabalho. Para mais informações pode-se consultar o trabalho de (OLDHAM; MCLEAN, 2003) e (ECOFFET, 2007).

2.4 Falha Erro e Defeito

As falhas são inevitáveis, pois os componentes físicos sofrem a ação do tempo e envelhecem e sofrem interferência externa. Devido a diferentes traduções dos termos da área de tolerância a falhas adotou-se as definições de falha, erro e defeito de acordo com (FRANCO, 2008).

As falhas representam uma condição inesperada que pode levar o sistema a um estado anormal, podem ser originadas de falhas de projeto, defeitos físicos ou interferências externas. A presença de uma falha podem levar a um erro que causa uma mudança indesejada no estado do sistema. A presença de um erro pode levar a uma resposta incorreta

do sistema. A definição de que o sistema estará no estado errôneo é quando o processamento seguinte ao momento em que ocorreu a falha produzir um defeito. Um erro de um transistor de uma porta lógica pode representar uma falha na saída da porta, que pode representar uma falha no circuito onde a mesma está inserida.

Uma ilustração para esses conceitos pode ser vista na Figura 9, classificando-os no modelo de três universos: o universo físico onde as falhas acontecem, o universo da informação onde ocorrem os erros, e o universo do usuário onde aparecem os defeitos.

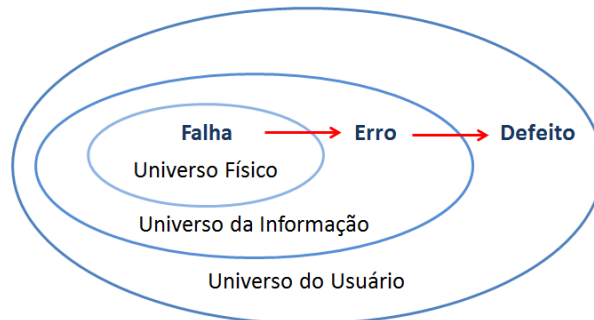


Figura 9: Modelo de três universos: falha, erro e defeito

2.5 Mascaramento

O mascaramento de uma falha garante que mesmo na presença de falha se tenha uma resposta correta. Quando uma falha não se manifesta em forma de erro ela foi mascarada, e deste modo a falha não irá levar a um erro. Também não haverá defeitos visíveis ao usuário. Existem três tipos de mascaramentos: mascaramento lógico, elétrico e temporal.

2.5.1 Mascaramento Lógico

O mascaramento lógico, exemplificado na Figura 10, acontece quando uma falha na entrada de uma determinada porta lógica não é capaz de modificar sua saída. A saída pode ser determinada pelas partes do circuito que não foram afetadas pela falha. O mascaramento lógico ocorre quando o evento transiente acontece em um caminho lógico que não está influenciando a saída naquele momento. Para que aconteça a propagação de um transiente ao longo do circuito, é preciso um caminho sensibilizado entre o nodo afetado e a saída (KARNIK; HAZUCHA, 2004).

2.5.2 Mascaramento Elétrico

Este tipo de falha está associado a falhas transientes que são causadas por partículas radioativas, e estas partículas podem gerar um pulso de corrente elétrica. O mascaramento ocorre quando o pulso elétrico gerado pela falha é atenuado à medida que se propaga através das portas lógicas. Esta atenuação ocorre tanto em amplitude do transiente como nos tempos de subida e decida.

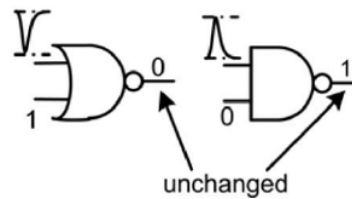


Figura 10: Mascaramento lógico (KARNIK; HAZUCHA, 2004)

O mascaramento elétrico, exemplificado na Figura 11, ocorre quando a perturbação transiente possui largura de banda superior à frequência de corte do circuito (KARNIK; HAZUCHA, 2004). Quando os eventos transientes têm largura de pulso menor que o atraso de porta, não são propagados (ENTRENA et al., 2009).

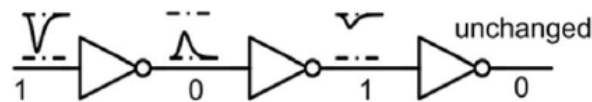


Figura 11: Mascaramento elétrico (KARNIK; HAZUCHA, 2004)

2.5.3 Mascaramento Temporal

O mascaramento temporal, exemplificado na Figura 12, ocorre quando o pulso transiente se propaga através do circuito até um elemento de memória, porém durante a sua transição não ocorre transição de clock (RIBEIRO, 2010). Este mascaramento acontece quando o evento transiente ocorre em um instante de tempo que esteja fora da janela de captura do circuito sequencial (KARNIK; HAZUCHA, 2004).

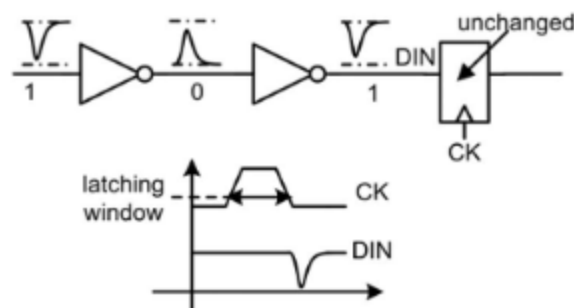


Figura 12: Mascaramento temporal (KARNIK; HAZUCHA, 2004)

3 ANÁLISE DA CONFIABILIDADE

Nesse Capítulo é apresentado o referencial teórico relativo a quatro abordagens que determinam a confiabilidade de circuitos lógicos. Inicialmente, será abordado o método de Matrizes de Transferência Probabilística (PTM) e, em seguida, o Modelo Binomial Probabilístico (PBR). Após, o Modelo de Probabilidade de Sinal (SPR), e por fim, a plataforma de Análise de Falhas por Injeção de Falhas (FIFA).

3.1 Método PTM

Nesta Seção são apresentadas as matrizes de transferência probabilística, do inglês, *Probabilistic Transfer Matrix* (PTM). O método de matrizes de transferência probabilísticas foi introduzido por (PATEL; MARKOV; HAYES, 2003) que teve como base o trabalho de (LEVIN, 1964), o qual foi mais explorado pelas obras de (KRISHNASWAMY et al., 2005).

A PTM é uma metodologia interessante para análise exata de circuitos lógicos em um contexto de múltiplas falhas simultâneas. A fim de atingir o funcionamento livre de erros, a função de um circuito lógico combinatório ou porta pode ser representada por uma tabela verdade que forma um mapeamento determinista de valores de entrada para valores de saída. No entanto, na presença de *soft errors*, este valor de entrada pode, ocasionalmente, levar a um valor de saída errado. Este comportamento pode ser modelado usando PTM se sabemos quantas vezes isso é provável que aconteça (PATEL; MARKOV; HAYES, 2003). A PTM do circuito é uma representação da matriz de probabilidade de ocorrência de todas as combinações de entradas e saídas. Quando temos um circuito com n entradas e m saídas, sua PTM é uma matriz $2^n \times 2^m$. Ou seja, em um circuito com 4 entradas e 1 saída a matriz PTM que representará o circuito terá 16 linhas e 2 colunas.

A PTM de um *gate* individual é derivada a partir da sua tabela verdade, bem como a sua matriz de transferência ideal (ITM) (FRANCO, 2008). Isto é, cada porta lógica possuirá uma matriz PTM e uma matriz ITM, que serão formadas a partir de sua tabela verdade. A matriz ITM reproduz o funcionamento da porta lógica seguindo seu comportamento normal sem a ocorrência de falhas, sendo a matriz ideal do circuito. A Figura

		saídas				saídas					
		0	1			0	1				
entradas	00	[1	0]	entradas	00	[q	$1 - q$]
	01		0	1			01		$1 - q$	q	
	10		0	1			10		$1 - q$	q	
	11		0	1			11		$1 - q$	q	
ITM_{OR}				PTM_{OR}							

Figura 13: Matriz ITM e PTM da porta lógica OR

13, apresenta a matriz ITM e a matriz PTM da porta lógica OR, onde a probabilidade de cada valor de saída é explícita para cada combinação de entrada e a confiabilidade é representada por q e a probabilidade de falha é referida como $1 - q$.

As linhas de interconexão das portas são consideradas ideais, sem probabilidade de geração de erros, sendo modeladas por uma matriz identidade 2×2 (PATEL; MARKOV; HAYES, 2003). A matriz ITM também pode realizar a modelagem da divisão de sinal ou *fanout* das permutações que ocorrem no circuito. A Figura 14, apresenta as matrizes ITMs correspondentes à divisão de sinal demonstrada em (a), o cruzamento de fios, em (b) e interconexões em (c), onde as linhas representam as entradas e as colunas as saídas. Estas matrizes representam como se dá a propagação do sinal quando ocorre uma divisão de sinal, cruzamento de fios e interconexões.

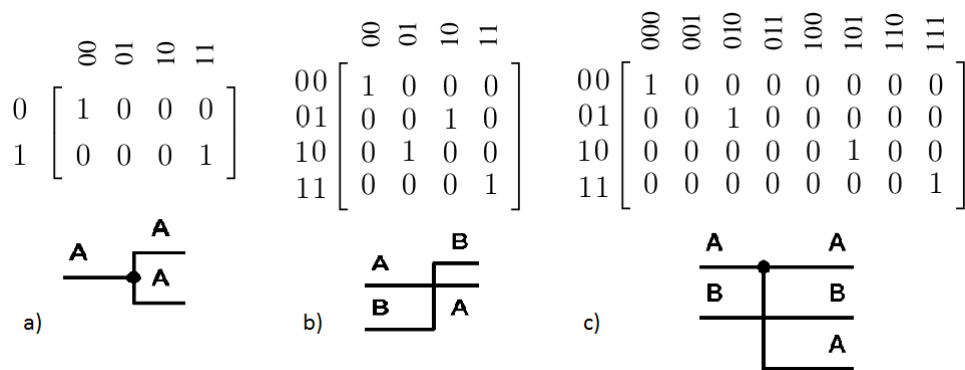


Figura 14: Matriz ITM da divisão de sinal, cruzamento de fios e interconexões

O primeiro passo para obter a confiabilidade do circuito é conhecer a topologia de interconexão de portas lógicas do mesmo. A partir desta, pode-se realizar a divisão do circuito em níveis. Estes níveis podem ser formados por: portas, portas e fios ou de interconexões. Quando o nível for formado somente por portas, ou portas e fios, o mesmo terá uma matriz ITM e uma matriz PTM. Caso a composição do nível seja apenas de interconexão (fios) o mesmo será composto apenas da matriz ITM.

Os elementos que se encontram no mesmo nível lógico utilizarão um produto tensor chamado (produto de *kroncker*, \otimes). Este tensor envolverá elementos que estiverem no mesmo nível lógico, gerando as matrizes ITM e PTM do nível. Após a conclusão da aplicação do produto tensor em todos os níveis compostos de portas lógicas e também gerar a matriz ITM dos níveis de interconexão, faz-se necessário realizar a multiplicação entre os níveis. Esta multiplicação pode ser dividida em dois grupos. O primeiro realizará a multiplicação da matriz ITM de cada nível, enquanto o segundo multiplicará as matrizes PTM. Quando o nível não possui matriz PTM (nível de interconexões), a matriz ITM do nível correspondente será utilizada na multiplicação.

Para preservar a matriz de correspondência entrada e saída de cada nível, precisa-se manter a ordem de aplicação do produto tensor nos demais níveis do circuito. A matriz PTM que representa o circuito como um todo será composta da combinação das PTMs e ITMs. A Figura 15, apresenta a PTM das portas NOT, AND e OR. Para simplificar a apresentação das matrizes, q representa a confiabilidade e $\bar{q} = 1 - q$ a probabilidade de falha.

$$PTM_{NOT} = \begin{bmatrix} \bar{q} & q \\ q & \bar{q} \end{bmatrix} \quad PTM_{AND} = \begin{bmatrix} q & \bar{q} \\ q & \bar{q} \\ q & \bar{q} \\ \bar{q} & q \end{bmatrix} \quad PTM_{OR} = \begin{bmatrix} q & \bar{q} \\ \bar{q} & q \\ \bar{q} & q \\ \bar{q} & q \end{bmatrix}$$

Figura 15: Matrizes PTM das portas NOT, AND e OR

A seguir aplicaremos os conceitos abordados ao circuito da Figura 16, com objetivo de elucidar todo processo que envolve o método da PTM. O primeiro passo consiste na divisão do circuito em níveis como demonstrado na Figura 16, sendo L1, L2 e L4 níveis formados por: portas ou portas e fios. Já no nível L3, um nível de interconexão, ocorre um cruzamento de fios.

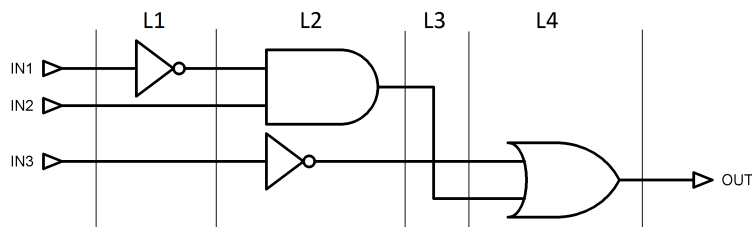


Figura 16: Divisão do circuito em níveis

A primeira etapa consiste em atribuir uma matriz de probabilidade do sinal para cada entrada, como demonstra a Equação 1. Para calcular a probabilidade conjunta das entradas utiliza-se o produto tensor, como demonstrado na Equação 2, iniciando sempre pela

entrada mais significativa do circuito. A seguir, obtemos a matriz ITM e PTM do nível L1. Esta operação consiste em aplicar o produto tensor às PTMs/ITMs de todos elementos que se encontram no mesmo nível. A Equação 3, apresenta o cálculo do nível PTM_{L1} , demonstrando como se dá a multiplicação da matriz ITM_{WIRE} , devido à passagem dos fios $IN2$ e $IN3$ para o próximo nível. A Equação 4 corresponde à multiplicação do nível L2, o qual é composto somente por portas. Já no nível L3, representado na Equação 5, temos apenas a atribuição da matriz de permutação ao nível ITM_{L3} . Em L4 ocorre o mesmo processo, mas a diferença é que neste nível existem as matrizes ITM_{OR} e PTM_{OR} , como pode ser visto na Equação 6. A partir do ponto em que possuímos as matrizes correspondentes de cada nível, torna-se possível a obtenção da matriz ITM e PTM do circuito. Para obter a ITM do circuito multiplica-se as ITM_{L1} até ITM_{L4} . A obtenção da PTM do circuito envolve a matriz da probabilidade conjunta das entradas (IN_{CIR}), a qual deverá ser inserida no cálculo, como demonstra a Equação 7. E por fim a Equação 8, responsável por realizar a multiplicação elemento a elemento entre a $ITM_{CIRCUITO}$ e $PTM_{CIRCUITO}$.

$$\begin{aligned} IN1 &= PROB_{INPUT}; \\ IN2 &= PROB_{INPUT}; \\ IN3 &= PROB_{INPUT}; \end{aligned} \tag{1}$$

$$IN_{CIR} = IN1 \otimes IN2 \otimes IN3; \tag{2}$$

$$\begin{aligned} ITM_{L1} &= ITM_{NOT} \otimes ITM_{WIRE} \otimes ITM_{WIRE}; \\ PTM_{L1} &= PTM_{NOT} \otimes ITM_{WIRE} \otimes ITM_{WIRE}; \end{aligned} \tag{3}$$

$$\begin{aligned} ITM_{L2} &= ITM_{AND} \otimes ITM_{NOT}; \\ PTM_{L2} &= PTM_{AND} \otimes PTM_{NOT}; \end{aligned} \tag{4}$$

$$ITM_{L3} = ITM_{PER}; \tag{5}$$

$$\begin{aligned} ITM_{L4} &= ITM_{OR}; \\ PTM_{L4} &= PTM_{OR}; \end{aligned} \tag{6}$$

$$ITM_{CIRCUITO} = ITM_{L1} * ITM_{L2} * ITM_{L3} * ITM_{L4};$$

$$PTM_{CIRCUITO} = IN_{CIR} * PTM_{L1} * PTM_{L2} * ITM_{L3} * PTM_{L4}; \quad (7)$$

$$CONF_{CIRCUITO} = sum(ITM_{CIRCUITO} .* PTM_{CIRCUITO}); \quad (8)$$

Os cálculos são realizados de forma simultânea sobre todas as combinações de entrada possíveis, calculando as probabilidades exatas de erros. A abordagem PTM é uma metodologia interessante, mas uma simplificação do método não é tão simples e com a limitação do tamanho das matrizes necessárias, seu uso é restrito. Algumas abordagens são propostas para diminuir o gargalo de memória como o uso de diagramas de decisão algébricos (ADDs), para comprimir PTMs (BAHAR et al., 1993).

Na codificação ADD de uma matriz, os nós representam decisões em variáveis de linha e coluna, e os terminais representam valores em entradas de matriz. As próprias entradas são representadas por caminhos a partir do nó raiz e terminando em um terminal. O mesmo caminho pode codificar várias entradas se as variáveis forem ignoradas em um caminho. Um grande desafio técnico no uso de ADDs para a representação PTM é a falta de algoritmos ADD para matrizes não-quadradas (KRISHNASWAMY et al., 2005). Nesse processo, o ADD é reduzido na medida em que os nós idênticos não são representados duas vezes. A Figura 17 mostra a representação de uma matriz com ADDs.

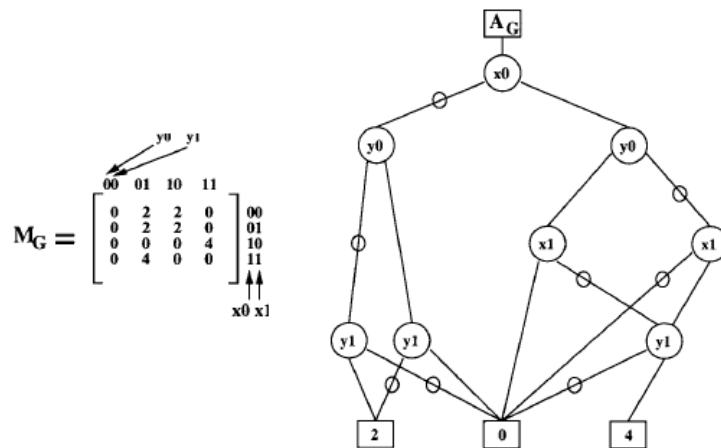


Figura 17: Representação de matriz com ADDs (BAHAR et al., 1993).

Outra abordagem proposta para diminuir o gargalo de memória é a utilização de *macro-gate*, o que é definido como uma porta elementar ou grupo de portas até um *fanout* para portas convergentes ou saídas primárias (XIAO et al., 2011). Uma comparação entre os métodos diagrama de decisão algébrico (ADD) e de *macro-gate* (MG), analisados por (XIAO et al., 2011), demonstra que, para circuitos pequenos, a PTM é mais rápida que MG, e, às vezes, o método utilizando ADD é mais rápido que os outros dois métodos. O diagrama de decisão algébrico comprime os blocos de matriz, e possui a mesma precisão que o PTM, e tem sobrecarga de tempo menor para circuitos grandes. As melhorias

não ajudam a resolver o problema de complexidade mesmo quando os circuitos possuem poucas dezenas de portas lógicas (BHADURI et al., 2007). A Figura 18 apresenta um exemplo de MG em um circuito.

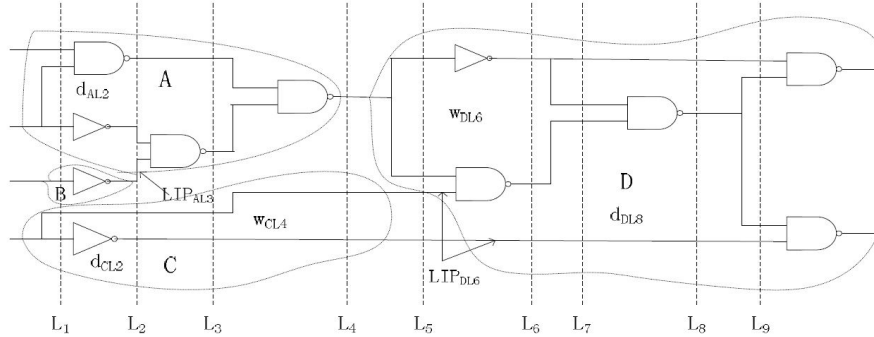


Figura 18: Um exemplo de circuito constituído por quatro *macro-gates* (XIAO et al., 2011).

Outra abordagem foi proposta se concentrando nos requisitos de memória do algoritmo PTM. Os resultados do algoritmo apresentam melhoras nos requisitos de tempo de execução e de ocupação de memória em relação à implementação direta da PTM (NAVINER et al., 2008). A melhora na abordagem PTM com base na computação de submatrizes úteis de produto tensores leva à mesma confiabilidade exata, exigindo sempre menos memória do que a solução original PTM, e, dessa forma, aumentando a escalabilidade do método.

3.2 Modelo PBR

Esta Seção apresenta o Modelo Binomial Probabilístico (PBR), do inglês, *Probabilistic Binomial Model*, para análise de confiabilidade, desenvolvido como uma abordagem analítica para modelar a confiabilidade de circuitos lógicos com falhas. A implementação ocorre por meio de simulação de falhas ou injeção de falhas em um ambiente de emulação de circuito. Dessa forma, é possível produzir resultados exatos ou altamente precisos para grandes circuitos. Consideremos um circuito genérico de lógica combinacional com vetor de entrada n -bit \vec{x} e um vetor m -bit \vec{y} , como mostrado na Figura 19.

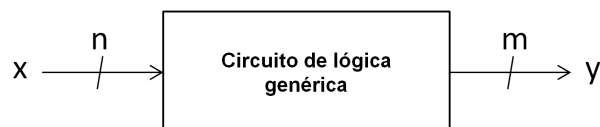


Figura 19: Circuito genérico de lógica combinacional.

Caso os *gates* deste circuito representado pela caixa falharem com uma certa probabilidade, a confiabilidade pode ser determinada como demonstrado pela equação 9. Na equação, $p(\vec{y} = correct|\vec{x}_i)$, representa a probabilidade de ocorrência de um vetor de

saída correto. O vetor de entrada \vec{x}_i e $p(\vec{x}_i)$ representam a probabilidade de ocorrência desse vetor de entrada.

$$R_{cir} = \sum_{i=0}^{2^n-1} p(\vec{y} = correct|\vec{x}_i)p(\vec{x}_i) \quad (9)$$

Para modelar a contribuição de cada *gate* do circuito genérico para a confiabilidade geral, deve ser determinada a probabilidade de mascaramento de falhas com todas as combinações de entrada e falhas.

A abordagem PBR visava calcular o valor exato da confiabilidade do sinal dos circuitos em teste, com base em ferramentas bem conhecidas e modelagem simples. O resultado é uma metodologia que automatiza o processo de computação das capacidades de mascaramento lógico de um circuito, usando procedimentos que já estão presentes no processo de *design*. Os modelos calculam o valor exato de confiabilidade dos circuitos lógicos combinacionais, mas a determinação depende de uma simulação de falhas exaustiva do circuito em teste.

A principal desvantagem da metodologia PBR é a necessidade de simulação de falhas ou de emulação de falhas, pois estas tarefas consomem muito tempo. No entanto, a solução analítica proposta pela abordagem PBR é economizar tempo do ponto de vista da reutilização, permitindo análises sucessivas e repetitivas do circuito simulado. Isto é realizado por meio de uma resolução de expressão de confiabilidade simples. Além disso, as otimizações apresentadas permitem uma redução do tempo de simulação de acordo com o tipo de análise a ser feita. Desta forma, a abordagem PBR é indicada como uma metodologia de análise de confiabilidade precisa, que pode ser usada para uma análise detalhada das versões finais dos circuitos e como ferramenta de validação para outras metodologias de análise de confiabilidade.

A dependência de simulação/emulação de falhas limita o escopo de interesse na abordagem PBR, reduzindo seu potencial de uso em *design* automatizado confiável. No entanto, continua a ser a melhor abordagem em relação à precisão e é adequada para uma avaliação precisa da confiabilidade das versões finais dos circuitos. A precisão do PBR depende do percentual de falhas injetadas (FRANCO, 2008).

3.3 Modelo SPR

A Confiabilidade pela Probabilidade do Sinal, do inglês, *Signal Probability Reliability (SPR)*, é um modelo que baseia-se nas probabilidades de um sinal lógico assumir valores corretos e incorretos. O algoritmo simples é a principal vantagem do método para determinação e propagação de probabilidades de sinal. Para isso, utiliza uma representação de probabilidade de sinal de quatro estados, que é crucial para o algoritmo de propagação. Desse modo, engloba explicitamente a informação de confiabilidade de

signal para qualquer ponto no circuito. Como esperado, as correlações de sinal *fanouts* são a restrição principal ao lidar com probabilidades de sinal e compensações entre precisão e tempo de processamento são sempre necessárias (FRANCO et al., 2008).

A representação de quatro estados de um sinal binário representa um 0 correto, um 1 correto, um 0 incorreto e um 1 incorreto. Considere as probabilidades de ocorrência desses estados: $P(\text{signal} = \text{correto } 0)$, $P(\text{signal} = \text{incorreto } 0)$, $P(\text{signal} = \text{correto } 1)$ e $P(\text{signal} = \text{incorreto } 1)$. A representação da matriz 2 x 2 dessas probabilidades é demonstrada na Figura 20.

$$P_4(\text{signal}) = \begin{bmatrix} P(\text{signal} = \text{correto } 0) & P(\text{signal} = \text{incorreto } 1) \\ P(\text{signal} = \text{incorreto } 0) & P(\text{signal} = \text{correto } 1) \end{bmatrix} = \begin{bmatrix} \text{signal}_0 & \text{signal}_1 \\ \text{signal}_2 & \text{signal}_3 \end{bmatrix}$$

Figura 20: Representação matricial das probabilidades de sinal de quatro estados.

A representação de quatro estados de probabilidades de sinal incorpora a informação de confiabilidade do sinal (R), expressada pelas probabilidades de um valor correto ocorrer (1 ou 0). Dessa forma, $R(\text{signal}) = \text{signal}_0 + \text{signal}_3$. A confiabilidade do sinal de um circuito pode ser determinada pela propagação das probabilidades do sinal de entrada através de todos os *gates* do circuito e a computação da probabilidade do sinal de saída (FRANCO et al., 2008).

A propagação das probabilidades do sinal de entrada através de uma porta lógica é calculada multiplicando as probabilidades de entrada do gate (I_{gate}) pela função de transferência do gate (PTM_{gate}), como demonstrado na Equação 10.

$$P(S) = I_{gate} \times PTM_{gate} \quad (10)$$

$$A_4 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad B_4 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

$$\begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix} \times \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \\ 0.02 & 0.98 \\ 0.98 & 0.02 \end{bmatrix} = \begin{bmatrix} 0.245 & 0.005 \\ 0.005 & 0.245 \\ 0.005 & 0.245 \\ 0.245 & 0.005 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.49 & 0.01 \\ 0.01 & 0.49 \end{bmatrix}$$

$I = A_4 \otimes B_4 \quad PTM_{XOR} \quad P(S) \quad S_4$

Figura 21: Exemplo de propagação da probabilidade do sinal em uma porta XOR.

As probabilidades de entrada conjunta correspondem a um produto tensor (produto Kronecker, \otimes) das probabilidades dos sinais de entrada da porta lógica, como apresentado na Figura 21.

A probabilidade do sinal de saída no formato de matriz 2×2 (S_4) é calculada reagrupando as probabilidades de saída em $P(S)$ de acordo com a função da porta lógica, representada pela sua ITM. Para saídas múltiplas, a confiabilidade do circuito pode ser obtida pela multiplicação da confiabilidade individual dos sinais de saída. O procedimento de propagação deve ser aplicado em uma ordem topológica, desde as entradas do circuito até às saídas (FRANCO et al., 2008).

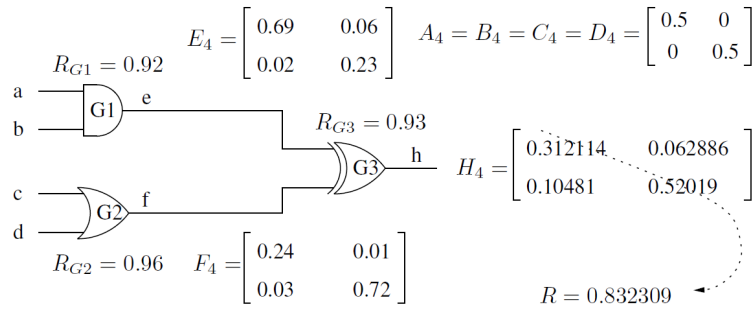


Figura 22: Exemplo de propagação da probabilidade do sinal em um circuito (FRANCO et al., 2008).

3.3.1 SPR *Multi-pass*

O problema da correlação do sinal foi estudado há muito tempo por (ERCOLANI et al., 1989), (CHOUDHURY; MOHANRAM, 2007), e foram propostas muitas maneiras para lidar com a complexidade exponencial das correlações de probabilidade.

Os efeitos dos *fanouts* reconvergentes são levados em consideração pelo cálculo da confiabilidade do circuito em múltiplas etapas de propagação. Neste processo de propagação de passagem múltipla, apenas um estado de probabilidade de sinal (de um *fanout* reconvergente) é propagado de cada vez e a confiabilidade de saída resultante é acumulada ao longo das várias iterações. Ao final do algoritmo de passagem múltipla, o valor de confiabilidade exato foi determinado, como demonstra a Equação 11, onde F é o número de nós de *fanout* reconvergente e f representa a f -ésima interação no algoritmo.

$$R_{circuito} = \sum_{f=1}^{4^F} R_{circuito}(f) \quad (11)$$

A análise de confiabilidade pela probabilidade do sinal ou *Sinal Probability Reliability Analysis (SPRA)*, é uma metodologia proposta por (FRANCO et al., 2008) e visa a utilização de forma autônoma ou incorporada no fluxo de síntese de circuitos lógicos. A precisão do SPRA depende do número de sinais reconvergentes que são levados em consideração pelo algoritmo de passagem múltipla. O tempo de execução depende do número de portas lógicas no circuito, do número de sinais de *fanout* e da topologia do circuito, ou seja, número de sinais que estão correlacionados com os *fanouts* reconvergentes.

A metodologia SPRA é uma tentativa de desenvolver uma metodologia escalável para avaliação de confiabilidade de circuitos lógicos. Os resultados apresentados utilizando esta metodologia são promissores. O algoritmo proposto demonstra ser flexível permitindo muitos tipos diferentes de análise, como por exemplo, a análise que determina a contribuição de células específicas para a confiabilidade do circuito, o que serve como uma indicação dos componentes cuja melhoria de confiabilidade seria mais útil.

3.4 Plataforma FIFA

A injeção de falhas é considerada muito útil para avaliar o comportamento dos sistemas de computação na presença de falhas (JEITLER; DELVAI; REICHOR, 2009). A idéia básica dessa abordagem é produzir ou simular falhas durante a operação do sistema. As saídas do circuito, com e sem falhas internas, são comparadas para avaliar a sua robustez.

A plataforma FIFA, do inglês, *Fault Injection Fault Analysis*, é apresentada como uma plataforma eficiente para a estimativa de robustez de falhas de circuitos digitais. Os resultados de síntese mostraram que a plataforma pode exceder as existentes na literatura em termos de eficiência e desempenho da área. Além disso, a plataforma FIFA permite que o projetista controle complexidade e completude do processo de análise (NAVINER et al., 2011). A maioria das ferramentas lidam com falhas únicas, mas não falhas múltiplas simultâneas. Além disso, eles consideram falhas transitórias ou permanentes, mas não as duas. Se forem consideradas várias falhas simultâneas, isto é, se várias portas lógicas podem falhar ao mesmo tempo, o número de testes para análise exaustiva em grandes circuitos pode tornar-se proibitivo (NAVINER et al., 2011).

O trabalho de (NAVINER et al., 2011), apresenta uma nova plataforma para avaliar a robustez dos circuitos digitais contra falhas, é projetado como um IP de hardware para acelerar a abordagem de Injeção de falhas. A plataforma suporta vários modelos de falhas, bem como falhas únicas e múltiplas. A plataforma proposta foi implementada em FPGA (*Field Programmable Gate Array*) e a análise é realizada no nível de transferência de registradores (RTL). O IP desenvolvido é totalmente parametrizável, e os resultados da síntese mostraram que excede os relatados na literatura em termos de eficiência e desempenho em área.

4 MÉTODO IMPLEMENTADO

Neste capítulo apresentamos as características gerais que envolvem a ferramenta que implementa o método de Matrizes de Transferência Probabilística (PTM), método que abordamos no Capítulo 3, bem como são apresentados os detalhes da implementação e parâmetros de entrada, bibliotecas e funções utilizadas.

Após estudar vários métodos que permitem determinar a confiabilidade optou-se pelo método de Matrizes de Transferência Probabilística (PTM). O método PTM é estudado nos trabalhos de (PATEL; MARKOV; HAYES, 2003), (LEVIN, 1964) e (KRISHNASWAMY et al., 2005). A escolha do método se dá pela possibilidade da obtenção da confiabilidade exata do circuito analisado e da necessidade de se ter uma ferramenta que realize a análise a partir da descrição de *hardware* de um circuito.

A linguagem escolhida para o desenvolvimento da ferramenta foi Java, visto que já havia trabalhado com a linguagem e estava familiarizado com a mesma.

A ferramenta desenvolvida está dividida em duas partes. A primeira parte faz a leitura do arquivo Verilog (HDL) contendo a descrição de *hardware* do circuito, gerando uma na Descrição do Circuito Intermediário (DCI). A segunda parte é responsável por ler a DCI e gerar um modelo PTM do circuito no formato compatível com o *software* matemático Scilab, sendo que essa segunda parte é chamada de Gerador de Código Scilab (GCS). Desse modo, o presente método permite determinar a confiabilidade de um circuito a partir do arquivo Verilog (HDL) contendo a descrição de *hardware* do circuito.

4.1 Detalhes da implementação

O ambiente de desenvolvimento escolhido para o desenvolvimento foi NetBeans IDE, utilizando a linguagem de programação Java. A ferramenta consiste em um código para a leitura de um arquivo Verilog (HDL) contendo a descrição de *hardware* do circuito para ser analisado posteriormente. Esses dados contêm informações de identificação de cada porta lógica, identificando quais são os sinais de entrada e de saída, e sua posição. Outra informação importante para o processo é a identificação dos sinais de entrada do circuito.

A extração dos dados do arquivo HDL foi realizado por Expressão Regular (ER),

também conhecida como *regex*, a qual é capaz de encontrar um determinado padrão em um texto. Utilizou-se quatro expressões regulares para ler as linhas que contêm uma porta lógica. Para executar a operação utilizou-se o pacote *java.util.regex*, que possui as classes *Pattern* e *Matcher*. A classe *Pattern* é responsável por representar o que se está procurando, já a classe *Matcher* realiza buscas e retorna os resultados encontrados.

A seguir, apresentamos na Figura 23, uma visão geral da ferramenta, a qual se encontra dividida em duas partes. A primeira parte, realiza o tratamento dos dados extraídos a partir do Verilog (HDL), e tem como saída a Descrição do Circuito Intermediário (DCI). A segunda parte, executa no cálculo no *software* matemático Scilab, apresentando como resultado final a determinação da confiabilidade do circuito através do método PTM.

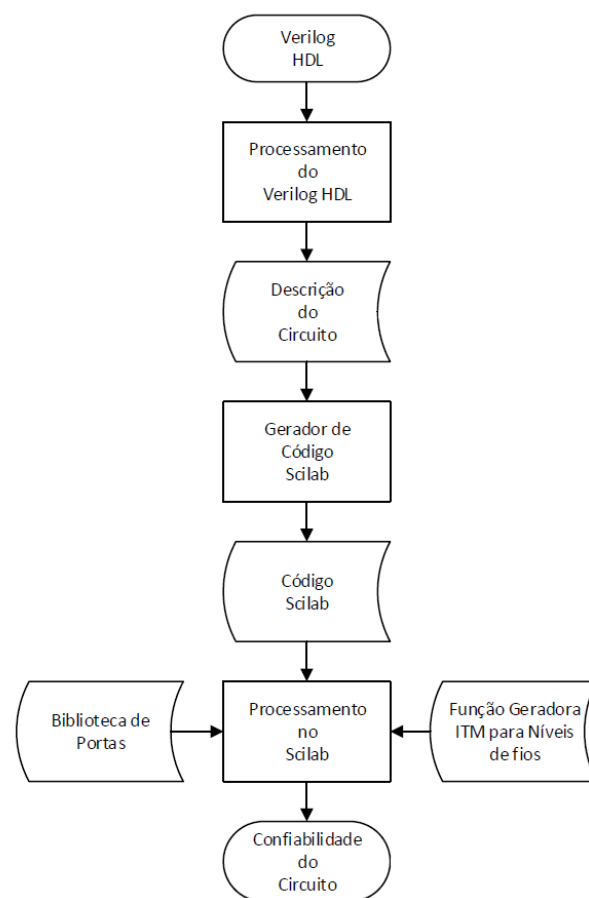


Figura 23: Visão geral da ferramenta

4.1.1 Processamento do Verilog HDL

Após a extração dos dados do HDL para uma estrutura de Grafo que representa as informações das portas lógicas do circuito, as portas são agrupadas de modo que formem níveis de portas com a mesma profundidade lógica. Para isso, é utilizado um algoritmo de busca em profundidade DFS (*Depth-First Search*).

Assim, esses níveis podem ser formados somente por portas lógicas, ou por portas

e sinais/fios. Concluída a formação dos níveis que compõem o circuito, temos que propagar os sinais que estão em um determinado nível. Desse modo, os sinais aparecerão nos níveis intermediários também. Nesse ponto, temos o nível de portas e sinais completamente formado. Os níveis de interconexão que ocorrem entre os níveis de portas são obtidos realizando uma leitura nos sinais de saída de um nível com os de entrada do nível seguinte. Com uma leitura nos níveis de portas e de interconexões obtemos as informações necessárias para criar um arquivo com a Descrição do Circuito Intermediário (DCI).

4.1.2 Descrição do Circuito Intermediário (DCI)

A Descrição do Circuito Intermediário (DCI) pode ser vista como uma composição de fragmentos que estão separados por um delimitador, o ponto e vírgula. A representação do circuito é dividida em três blocos distintos: o primeiro bloco representa as entradas do circuito; o segundo e terceiro blocos são intercalados dependendo do circuito analisado, até a representação completa do circuito. O segundo e terceiro blocos podem representar níveis compostos de interconexões ou de portas. A ordem da sequência destes blocos dependerá do formato da construção do circuito que se está analisando. Ao término do conjunto dos três blocos temos uma representação completa da estrutura do circuito, formando assim a descrição do mesmo. A seguir, apresentamos o circuito da Figura 24, o qual servirá de exemplo para demonstrar a estrutura da DCI. A representação da DCI correspondente ao circuito exemplo pode ser visualizada na Figura 25.

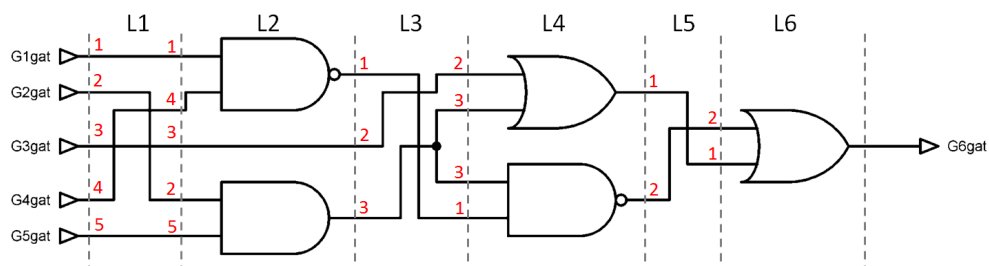


Figura 24: Circuito exemplo.

```

1  G1gat, G2gat, G3gat, G4gat, G5gat ;
2  ([1, 2, 3, 4, 5], [1, 4, 3, 2, 5]) ;
3  {NAND2, WIRE, AND2} ;
4  ([1, 2, 3], [2, 3, 3, 1]) ;
5  {OR2, NAND2} ;
6  ([1, 2], [2, 1]) ;
7  {OR2} ;

```

Figura 25: Descrição do circuito da Figura 24.

4.1.3 Gerador de Código Scilab (GCS)

O Gerador de Código Scilab é a parte da ferramenta responsável por ler a descrição do circuito intermediário. A geração do código consiste no recebimento de uma estrutura ordenada que contém a representação dos níveis, os quais são formados por interconexões ou de portas lógicas e fios. Durante a construção da estrutura de cada nível, uma ordem na leitura deve ser seguida, a qual será sempre a entrada mais significativa do circuito, ou seja, de cima para baixo. O resultado gerado ao final do processo será um arquivo de texto no formato compatível para ser executado no Scilab. Esse código possui em seu cabeçalho a chamada a dois arquivos que serão necessários quando o código estiver em execução no Scilab. A primeira chamada será para um arquivo responsável por gerar a matriz ITM do nível. A segunda chamada será para a biblioteca de portas com a matriz ITM e PTM de cada porta lógica.

A seguir, descreveremos com maior nível de detalhamento o gerador de código Scilab e seus sub-processos. Uma visão geral desses processos é vista no fluxograma apresentado na Figura 26.

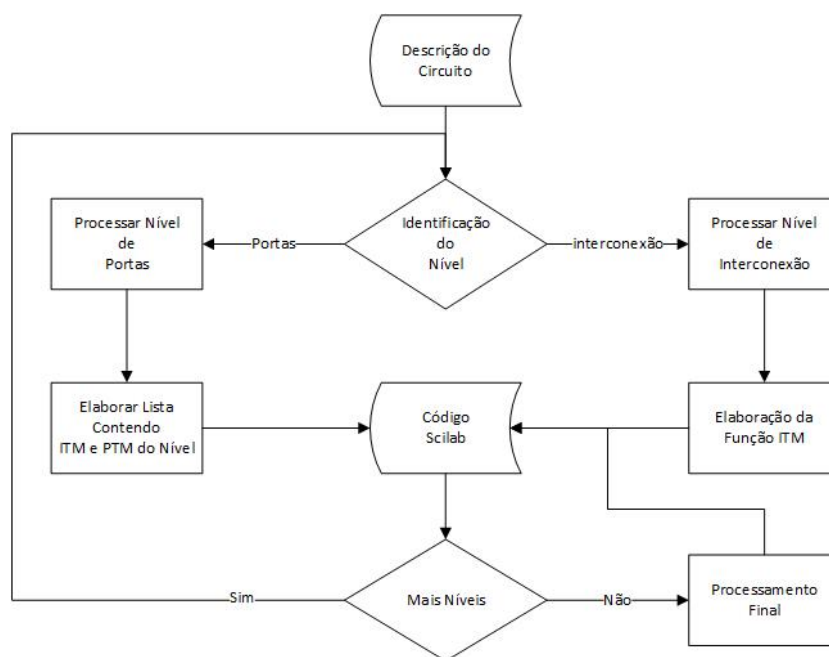


Figura 26: Processos do gerador de código

A primeira parte do fragmento corresponde à identificação das entradas do circuito. Todos os demais fragmentos passarão por um analisador com o objetivo de identificar a composição do mesmo. Estes fragmentos podem estar em dois formatos, um para representar portas e passagem de fios e outro para níveis que representam interconexões. Dessa forma, o fragmento possui modos com dois processos cada. A sequência seguida dependerá da composição de cada fragmento.

O primeiro modo, em seu primeiro processo, tratará fragmentos compostos por por-

tas e passagem de fios. A sua leitura é realizada para cada componente do fragmento na ordem em que se encontra. São realizadas algumas alterações na sua estrutura adequando ao formato necessário. No caso das portas lógicas precisa-se gerar a partir de sua identificação a chamada às matrizes ITM e PTM. No entanto, quando o componente for um fio, apenas a chamada à matriz ITM é gerada. O segundo processo consiste na elaboração da lista contendo todas ITMs e PTMs do nível. Na falta da existência de PTM como é o caso dos fios, será considerada a matriz ITM seguindo a mesma ordem. O resultado com a chamada para as matrizes ITM e PTM que representam o nível será gravado no arquivo Código Scilab.

O segundo modo, em seu primeiro processo, tem como função processar o nível de interconexão, o qual receberá o fragmento com a sequência de entrada e saída do nível. Este processo consiste na transformação nos dados recebidos. Esses dados são compostos de letras e números e serão transformados levando em conta a relação existente entre a sua entrada e saída, a qual deverá permanecer a mesma. A transformação se dará para um formato numérico. Esta relação dos números de entrada e de saída servirão de entrada para o próximo processo. O segundo processo consiste em receber a sequência numérica que representa as entradas e saídas do nível de interconexão. Esta sequência será incluída na função que será executada no Scilab.

Esta sequência descrita continuará se repetindo para todos os níveis da descrição do circuito. Ao término desta, será executado o processamento final, responsável por criar a uma matriz ITM e uma matriz PTM do circuito, a qual será gerada a partir das matrizes ITM e PTM individuais de cada nível.

4.2 Outros detalhes da implementação

A seguir serão apresentados detalhes da implementação do desenvolvimento da ferramenta para o método PTM. A Figura 27 contém uma visão geral de todas as partes da ferramenta.

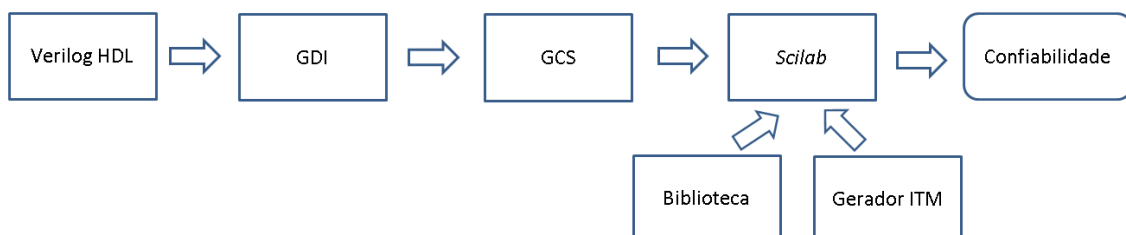


Figura 27: Apresentação das partes que compõe a ferramenta.

4.2.1 Biblioteca

A biblioteca de portas é um arquivo que contém uma matriz ITM e uma matriz PTM para cada porta lógica. A execução dessas matrizes ocorrerá nos níveis compostos por portas lógicas. Nessa biblioteca também está a matriz ITM de um fio (*itm_wire*) e a matriz de probabilidade de uma entrada (*prob_input*).

4.2.2 Gerador ITM

O código responsável por gerar os níveis de interconexões foi desenvolvido por (FRANCO, 2008). Uma transformação foi necessária no código *Scilab*, transformando o mesmo em uma função que recebe dois parâmetros. O primeiro parâmetro, representa a sequência numérica correspondente a ordem de entrada do nível ITM, o segundo parâmetro corresponde a posição de saída dos elementos. Esta função (chamamos de gerador()), é executada no início do código para que seja reconhecida nas linhas de código que implementam o método PTM. Sempre que for necessário criar um nível de interconexões, existirá uma chamada para a função gerador.

4.3 Estudo de Caso

Nessa Seção apresentamos um estudo de caso onde aplicamos o método PTM. Para demonstração da ferramenta, tomamos como base o Verilog do circuito C17. A seguir, demonstraremos os passos realizados para verificar o funcionamento da ferramenta desenvolvida.

4.3.1 Verilog do Circuito C17

A Figura 28 apresenta o Verilog HDL referente ao circuito C17 e contém toda a descrição de *hardware*. O início das declarações do arquivo Verilog é identificada por *module* e contém a identificação do circuito e suas entradas e saídas. A próxima declaração identificada como *input*, contém as entradas do circuito. A identificação *wire*, contém os demais sinais utilizados nas conexões do circuito. As linhas seguintes contém as portas lógicas do circuito. A declaração *endmodule* indica o fim do arquivo Verilog.

4.3.2 Diagrama do Circuito Analisado

O circuito da Figura 29, demonstra o diagrama do circuito C17. O circuito está desenhado da mesma forma que a ferramenta desenvolvida organizou o circuito em 6 níveis.

4.3.3 Descrição do Circuito Intermediário

A Figura 30 apresenta a Descrição do Circuito Intermediário (DCI) gerada pela ferramenta após a leitura do arquivo Verilog apresentado na Figura 28. A primeira linha identifica as entradas do circuito. Essa representação intermediária possui os dados do

```

1 module C17 (G1gat, G2gat, G3gat, G4gat, G5gat, G6gat, G7gat);
2
3   input  G1gat, G2gat, G3gat, G4gat, G5gat;
4   output G6gat, G7gat;
5   wire  n1, n2, n3, n4;
6
7   NAND2X4 g000 (.A(G1gat), .B(G2gat), .Y(n1));
8   NAND2X4 g001 (.A(G2gat), .B(G4gat), .Y(n2));
9   NAND2X4 g002 (.A(G3gat), .B(n2), .Y(n3));
10  NAND2X4 g003 (.A(n2), .B(G5gat), .Y(n4));
11  NAND2X4 g004 (.A(n1), .B(n3), .Y(G6gat));
12  NAND2X4 g005 (.A(n3), .B(n4), .Y(G7gat));
13 endmodule

```

Figura 28: Verilog Circuito C17.

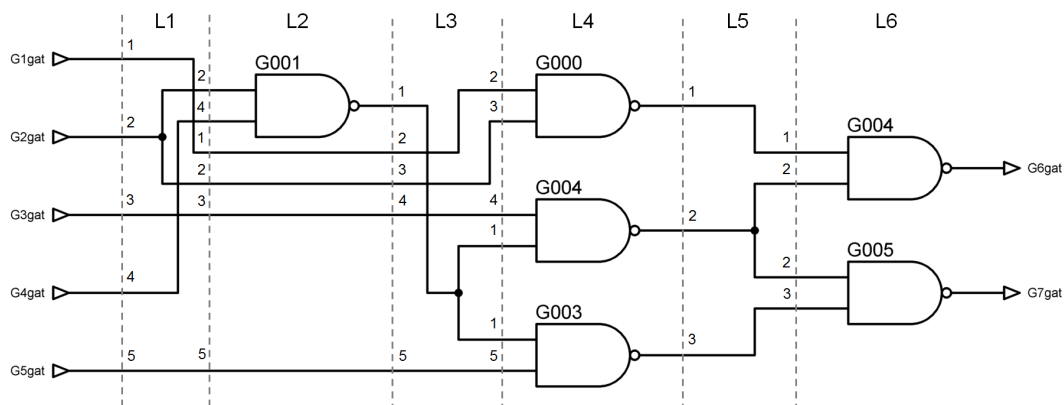


Figura 29: Circuito C17 analisado pela ferramenta.

Verilog HDL de maneira que as portas estão organizadas de acordo com sua profundidade lógica. Em seguida, as portas de mesma profundidade lógica são agrupadas em níveis, formando os níveis compostos por portas ou portas e fios.

```

1 G1gat, G2gat, G3gat, G4gat, G5gat;
2 ([1, 2, 3, 4, 5], [2, 4, 1, 2, 3, 5]);
3 {NAND2X4, wire, wire, wire, wire};
4 ([1, 2, 3, 4, 5], [2, 3, 4, 1, 1, 5]);
5 {NAND2X4, NAND2X4, NAND2X4};
6 ([1, 2, 3], [1, 2, 2, 3]);
7 {NAND2X4, NAND2X4};

```

Figura 30: Descrição do Circuito C17.

Os níveis de interconexões são formados por uma sequência de números de entrada e sequência de saída. Essa estrutura servirá de entrada para a função no *software* matemático *Scilab*. Assim, essa função gera a matriz ITM do nível.

4.3.4 Gerador Código Scilab

A seguir, apresentamos o código gerado pelo *software* para o circuito C17. A ferramenta faz a leitura da descrição do circuito intermediário, e como saída a ferramenta

gera código no formato Scilab que implementa o método PTM. A Figura 31 demonstra o código gerado pela ferramenta. O código contém dois arquivos que são incluídos no código, o primeiro contém a biblioteca de portas, o segundo, a função geradora para os níveis ITM. No Scilab ocorre uma série de multiplicações das matrizes do código gerado que resulta na confiabilidade do circuito analisado.

```

1 //Biblioteca
2 exec('biblioteca.sce', -1)
3 //Função auxiliar do Scilab
4 exec('function.sce', -1)
5
6 //Probabilidade individual das entradas
7 in_1 = prob_input;
8 in_2 = prob_input;
9 in_3 = prob_input;
10 in_4 = prob_input;
11 in_5 = prob_input;
12
13 //Probabilidade conjunta das entradas
14 in_cir = in_1 .* in_2 .* in_3 .* in_4 .* in_5;
15
16 itm_L1 = gerador([1,2,3,4,5],[2,4,1,2,3,5]);
17
18 itm_L2 = itm_NAND2X4 .* itm_wire .* itm_wire .* itm_wire .* itm_wire;
19 ptm_L2 = ptm_NAND2X4 .* itm_wire .* itm_wire .* itm_wire .* itm_wire;
20
21 itm_L3 = gerador([1,2,3,4,5],[2,3,4,1,1,5]);
22
23 itm_L4 = itm_NAND2X4 .* itm_NAND2X4 .* itm_NAND2X4;
24 ptm_L4 = ptm_NAND2X4 .* ptm_NAND2X4 .* ptm_NAND2X4;
25
26 itm_L5 = gerador([1,2,3],[1,2,2,3]);
27
28 itm_L6 = itm_NAND2X4 .* itm_NAND2X4;
29 ptm_L6 = ptm_NAND2X4 .* ptm_NAND2X4;
30
31 itm_circuito = itm_L1 * itm_L2 * itm_L3 * itm_L4 * itm_L5 * itm_L6;
32 ptm_circuito = in_cir * itm_L1 * ptm_L2 * itm_L3 * ptm_L4 * itm_L5 * ptm_L6;
33
34 conf_circuito = sum(itm_circuito .* ptm_circuito)

```

Figura 31: Código Scilab para C17 gerado pela Ferramenta.

5 RESULTADOS

Neste Capítulo serão apresentados os resultados obtidos para a confiabilidade de circuitos de mesma função mas com implementações diferentes, e de outros circuitos. Também são comparados os tempos de execução e de consumo de memória de diferentes partes do circuito, como a Descrição do Circuito Intermediária (DCI), Gerador de Código Scilab (GCS) e *Scilab*. Também será apresentado o comportamento do crescimento da matriz em função do número de entradas do circuito e os resultados das análises dos circuitos realizados com a ferramenta desenvolvida. Todos os diagramas dos circuitos analisados nesse Capítulo estão no (Apêndice A).

5.1 Identificação da confiabilidade de mesmos circuitos com implementações diferentes

Foram analisados quatro implementações diferentes do circuito C17. A Figura 32 apresenta os resultados de confiabilidade. Dentre as quatro versões do C17 analisadas, duas delas (C17-v2 e C17-v3) apresentaram valores de confiabilidade com pequenas variações para $q = 0.99$ (confiabilidade de cada porta individual), permanecendo entre 0,9519 e 0,9525. O circuito (C17-v2) apresentou a melhor confiabilidade compa-

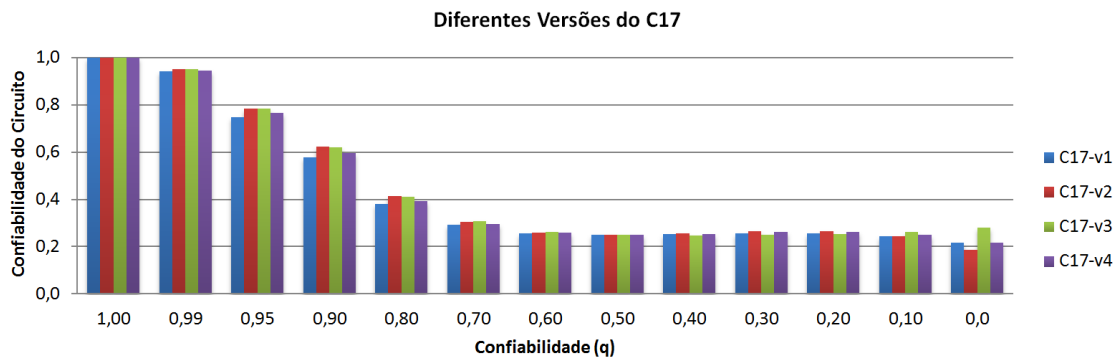


Figura 32: Variando a confiabilidade para diferentes implementações do C17.

rado com os demais. A Tabela 1, demonstra os dados da confiabilidade das diferentes implementações do C17, para diferentes valores de q . A Tabela 2 demonstra a variação

Tabela 1: Confiabilidade para diferentes implementação do C17.

(q)	C17-v1	C17-v2	C17-v3	C17-v4
1,0	1,0	1,0	1,0	1,0
0,99	0,9421	0,9525	0,9519	0,9467
0,95	0,7497	0,7861	0,7839	0,7654
0,90	0,5779	0,6240	0,6210	0,5974
0,80	0,3810	0,4142	0,4134	0,3945
0,70	0,2923	0,3070	0,3096	0,2984
0,60	0,2577	0,2610	0,2647	0,2591
0,50	0,2500	0,2500	0,2500	0,2500
0,40	0,2538	0,2560	0,2482	0,2548
0,30	0,2589	0,2659	0,2510	0,2624
0,20	0,2577	0,2667	0,2558	0,2635
0,10	0,2447	0,2455	0,2639	0,2508
0,00	0,2187	0,1875	0,2812	0,2187

Tabela 2: Diferentes versões C17 com variação de 0.999999 a 0.99

(q)	C17-v1	C17-v2	C17-v3	C17-v4
0,999999	0,999994	0,999995	0,999995	0,999995
0,999990	0,999940	0,999951	0,999951	0,999945
0,999900	0,999400	0,999513	0,999506	0,999450
0,999000	0,994022	0,995138	0,995076	0,994518
0,990000	0,942187	0,952519	0,951928	0,946749

da confiabilidade para valores próximos a 1,0 das versões do C17.

As análises dos circuitos C17-v2, C17v3 e C17v4 apresentaram a mesma confiabilidade quando o valor de q é próximo a 1,0. Com valores de $q = 0,999990$ os circuitos C17-v2 e C17-v3 tem a mesma confiabilidade. Nos casos (0,999900, 0,999000 e 0,990000) o circuito C17-v2 apresenta o melhor resultados quando comparado com os demais. Dessa forma, das versões do C17 analisadas a mais confiável é a versão C17-v2.

Ao comparar os resultados obtidos das análises dos circuitos C17 para valores próximos a 1.0, verificou-se que C17-v1 apresenta diferença em relação aos demais, e possui maior número de portas lógicas e de *fanouts*, talvez esse fator tenha influenciado

Tabela 3: Dados das diferentes versões do C17.

Circuitos	Nº Entradas	Nº Saídas	Nº Portas	Nº Níveis	Nº <i>Fanouts</i>	Nº Figura
C17-v1	5	2	9	6	4	50
C17-v2	5	2	6	6	3	51
C17-v3	5	2	6	6	3	52
C17-v4	5	2	7	6	3	53

na confiabilidade do circuito. A Tabela 3, apresenta os resultados obtidos.

5.2 Comparação do tempo de execução

A análise do tempo de execução está dividida em três partes: A primeira parte verifica o tempo de execução da leitura do Verilog até a Descrição do Circuito Intermediário (DCI). A segunda parte verifica o tempo de execução da leitura da DCI até o Gerador do Código Scilab (GCS). A terceira avalia o tempo gasto para execução do código que contém a implementação com o método PTM pelo *software* matemático Scilab. Como o tempo gasto no *Scilab* em alguns casos é maior que na etapa de geração DCI e GCS, os dados foram separados em duas Figuras. A Figura 33 apresenta os dados temporais do DCI e GCS, enquanto que a Figura 34 apresenta os mesmos dados incluindo o tempo gasto pelo *Scilab*.

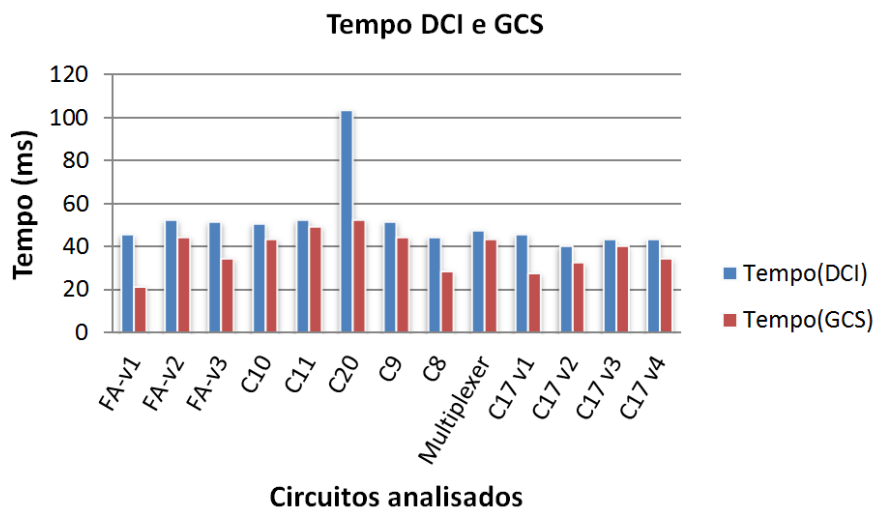


Figura 33: Comparação de tempo de execução DCI e GCS.

Analisamos o tempo gasto no processamento da Descrição do Circuito Intermediário (DCI), Gerador de Código Scilab (GCS) e Scilab. Quando comparando o tempo gasto pela DCI e GCS, os resultados mostram que o processo de construção da DCI foi maior que o tempo GCS em 100% dos circuitos analisados, como demonstrado a Figura 33. Quando comparados os tempos da DCI, GCS e Scilab, nota-se que, para circuitos pequenos, os tempos tendem a ter valores que não apresentam muita diferença um do outro. Mas, as vezes, mesmo circuitos pequenos, como é o caso do *FA-v3*, apresentam uma grande diferença no tempo de processamento em relação a DCI e GCS. Nesse caso, para o *FA-v3* (Figura 46 do Apêndice A), o que possibilitou esse aumento significativo em relação aos outros circuitos foi o grande número de interconexões existentes, resultando em uma matriz ITM $2^6 \times 2^{18}$. Outro fator que colaborou é a existência de 7 *Gates* em um mesmo nível, resultando em duas matrizes $2^{18} \times 2^7$, sendo uma ITM e outra PTM. Assim, tanto níveis ITM de interconexões como de *Gates* têm grande influência no tempo de execução.

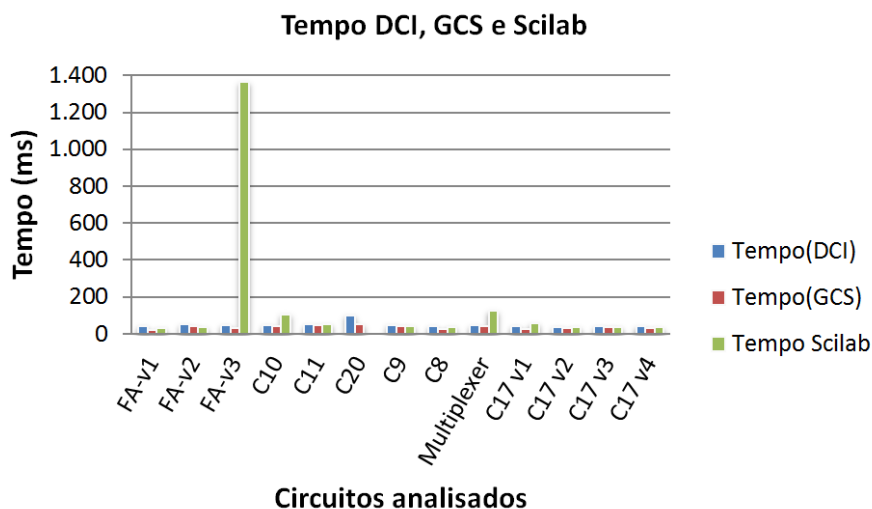


Figura 34: Comparação de tempo de execução DCI, GCS e Scilab.

5.2.1 Diferenças no tempo de execução

Ao executar os cálculos, constatou-se que existem diferenças nos tempos de cada cálculo quando o valor de q é alterado. Para investigar esse efeito foi realizado um total de 10 amostras para cada valor de q . Após, foi realizada uma média das amostras coletadas para cada valor de q . O circuito utilizado está no (Apêndice A, Figura 46). A Figura 35 apresenta o gráfico com o comportamento do tempo de execução em função da variação de q . Na Figura 35 observamos que quando q está entre 0,90 e 1,00 ocorre maior

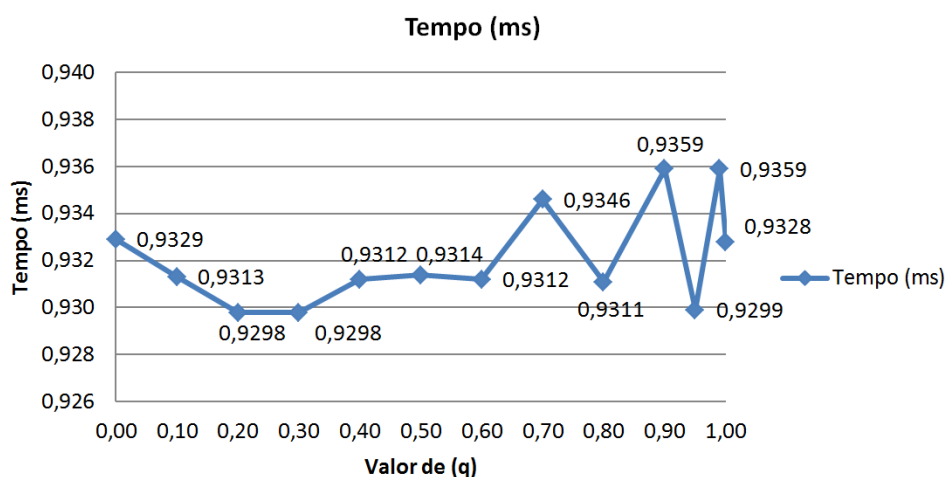


Figura 35: Tempo de execução com a variação de q .

oscilação no tempo do cálculo e também maior tempo para execução do cálculo. A fim de investigar a variação demonstrada realizou-se outro experimento variando o q de 0,90 até 1,0, como apresentado na Figura 36.

A causa que leva a essa variação no tempo de execução em função da variação de q poderá estar relacionada com sistema operacional. Um dos fatores que podem influenciar

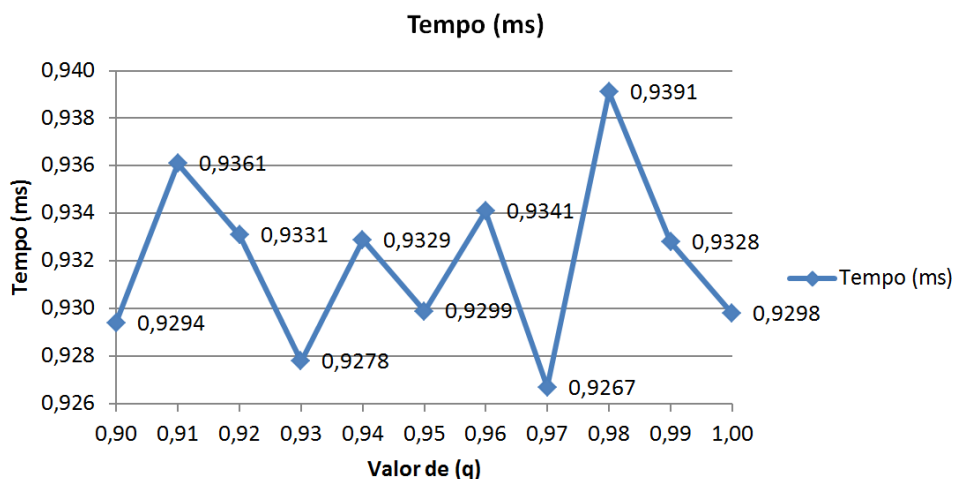


Figura 36: Tempo de execução com a variação de q 0,90 até 1,0.

é a quantidade de memória disponível em cada instante do cálculo. Outro fator poderá ser otimizações existentes no Scilab. A Figura 36, demonstra o ponto que consome mais tempo é quando $q = 0,98$, e é o maior tempo dos pontos analisados (0,00 até 1,0). Também observou-se que quando $q = 0,00$ o tempo é de 0,9329(ms) e $q = 1,0$ é de 0,9328(ms) a diferença de tempo não existe. Nesse experimento foi demonstrado que existem variações nos tempos para cada operação dependendo do valor de q , apesar de não serem tão significativas.

5.3 Comparação de consumo de memória

O consumo de memória como mostra a Figura 37, está relacionado com número de *Gates* e número de níveis que um circuito é composto. Analisamos o consumo de memória do GCI e GCS. Observa-se que a geração de código do método PTM consome sempre mais memória que a geração da descrição do circuito intermediário.

Dentre os circuitos analisados, um não foi possível executar no Scilab devido a um erro retornado pelo software matemático. Esse erro está relacionado com o consumo de memória necessária durante a execução. O erro retornado informa que o tamanho da pilha foi excedido. Na tentativa de solucionar tal erro encontramos uma solução que recomenda usar a função *stacksize* para aumentar o tamanho da pilha ao máximo. Foi realizado tal procedimento mas o erro permanece para o circuito C20 (Figura 49 do Apêndice A). Esse erro ocorre no *Scilab-5.4.1*. A versão do *Scilab-6.0.0* apresenta a quantidade de memória necessária em *Megabyte* para executar o código. O valor exigido pelo *Scilab* é de 34359.74 MB ou 33 GB para o circuito C20.

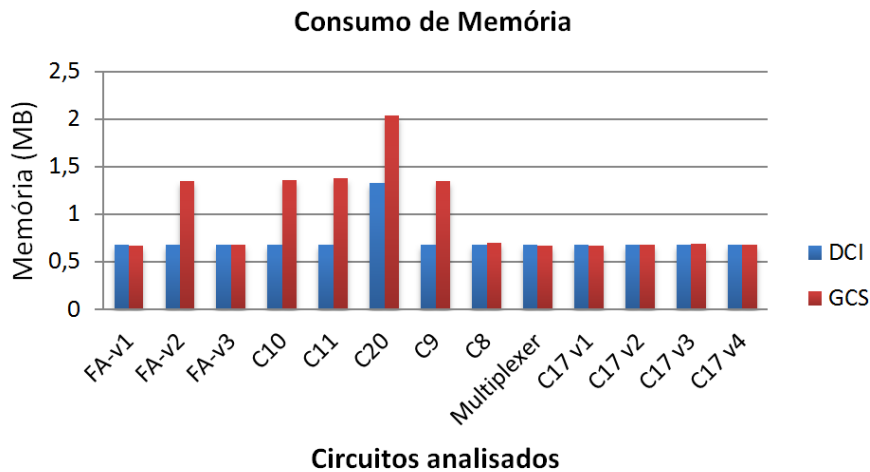


Figura 37: Comparação do consumo de memória.

5.4 Tamanho das matrizes

A Figura 38, demonstra o comportamento do crescimento da matriz que representa as probabilidades conjuntas das entradas de um circuito, onde essa matriz quadruplica a cada nova entrada. O limite para o método PTM está relacionado à quantidade de memória alocada para realizar o cálculo. Isso depende do *hardware* utilizado. Nesse trabalho foi utilizado um computador *Intel(R) Core i5-4210U CPU @ 1.70GHz 2.40GHz*, 8GB de memória e sistema operacional *Windows 7 (64 bits)*.

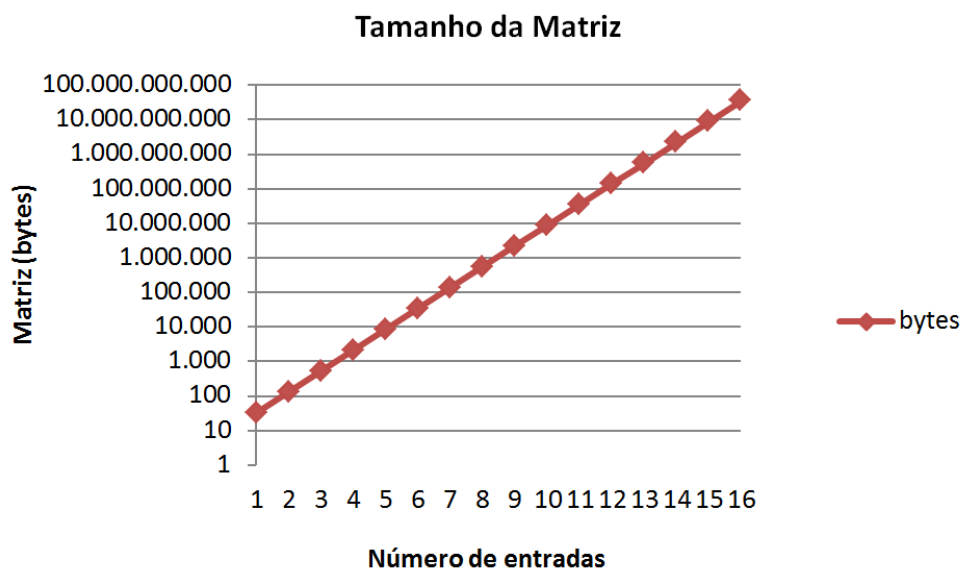


Figura 38: Tamanho de matriz em função do número de entradas do circuito.

Em experimento realizado, conseguimos criar a matriz da probabilidades conjunta das entradas com até 13 entradas. Foi possível criar duas dessas matrizes sem erro, mas a terceira não foi possível.

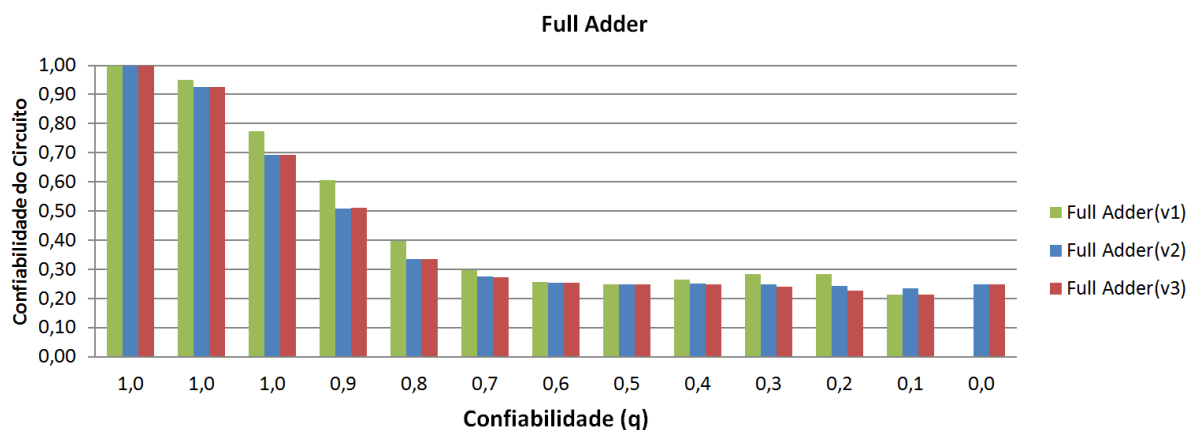
Tabela 4: Dados com variação de 0,999999 a 0,99.

(q)	Full Adder(v1)	Full Adder(v2)	Full Adder(v3)
0,999999	0,999995	0,999992	0,999992
0,999990	0,999948	0,999921	0,999921
0,999900	0,999475	0,999213	0,999213
0,999000	0,994765	0,992165	0,992166
0,990000	0,949001	0,925103	0,925255

O problema de memória não aparece só para a matriz de entrada. Para alguns circuitos pode aparecer nos níveis intermediários. Esses níveis podem ser formados de interconexões ou de portas lógicas, ambos podem fazer com que a execução do cálculo não seja possível. O consumo de memória ocorre nos níveis de interconexão e de portas lógicas. Nos níveis de interconexão, o grande número de *fanouts* ou cruzamento de fios geram matrizes grandes que consomem muita memória. Esse consumo de memória também se aplica ao nível de portas, quando há várias portas em um mesmo nível.

5.5 Circuitos analisados

A seguir, é demonstrado a confiabilidade para vários circuitos analisados. A Figura 39, apresenta a confiabilidade para três Full Adder. O diagrama desses circuitos está no Apêndice A, Figuras 44, 45 e 46. O gráfico da Figura 39 demonstra que, quando $q = 0,99$, o Full Adder(v1) tem confiabilidade superior ao Full Adder(v2) e Full Adder(v3). Já o Full Adder(v2) e (v3) tendem a terem valores praticamente iguais. A Figura 40 mostra em mais detalhe a diferença existente dos circuitos analisados quando $q = 0,99$.

Figura 39: Análise de Três Full Adder com variação de q

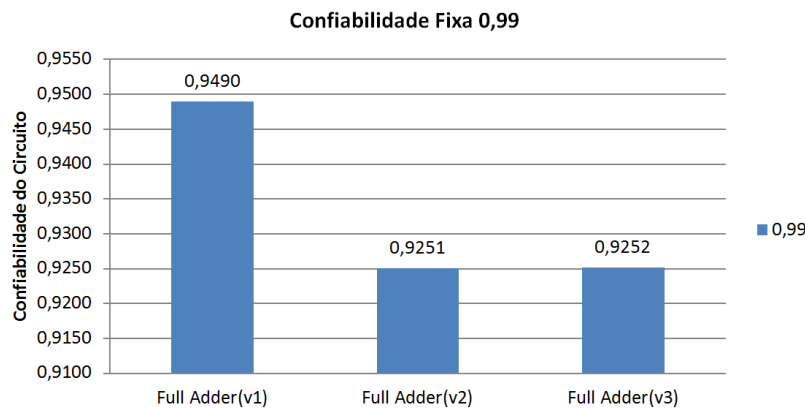


Figura 40: Análise de Três Full Adder com q fixo em 0,99.

Tabela 5: Dados das três versões de Full Adder.

Circuitos	Nº Entradas	Nº Saídas	Nº Portas	Nº Níveis	Nº <i>Fanouts</i>	Nº Figura
FA-v1	3	2	6	4	6	44
FA-v2	3	2	9	12	8	45
FA-v3	3	2	12	6	15	46

Outros circuitos foram analisados, como mostra o gráfico da Figura 41. Os circuitos C10 e C11 foram criados para investigar os limites da aplicação do método PTM com uso do *Scilab*. A versão do C17 somente com portas *NAND* é utilizada neste gráfico.

A versão do *Scilab-5.4.1 (64-bit)*, permitiu criar matriz da probabilidade conjunta das entradas para um circuito com até 13 entradas, que ocupa um espaço de 512 MB. Já na versão do *Scilab-6.0.0 (64-bit)* esse número passou para 15 entradas, e isso gera uma matriz de 8 GB. E esse processo demorou 3 minutos e 36 segundos para ser executado. Contudo, isso não significa que todos circuitos com 15 entradas irão executar, pois depende do número de portas lógicas e *fanouts*, o que faz com que o circuito aumente os níveis intermediários aumentando assim o consumo de memória.

A Tabela 7 apresenta resultados das análises para alguns circuitos com diferentes valores de q e a Tabela 8, demonstra outras informações desses circuitos: número de entradas, número de saídas, número de portas lógicas, número de níveis e número de *fanouts* de cada circuito analisado. A Tabela 6, demonstra os resultados quando q varia de 0,99 a 0,999999.

Os resultados obtidos demonstraram que a ferramenta desenvolvida é capaz de determinar a confiabilidade de um circuito a partir da descrição de *hardware* de um circuito. Desse modo, esse processo que era realizado de forma manual e que compreendia desenhar o circuito a partir do Verilog, dividir em níveis e escrever o código Scilab, tornou-se mais ágil com o auxílio da ferramenta desenvolvida. Além disso, esse processo poderá ser incluído em um processo de síntese que poderá escolher o circuito mais confiável para

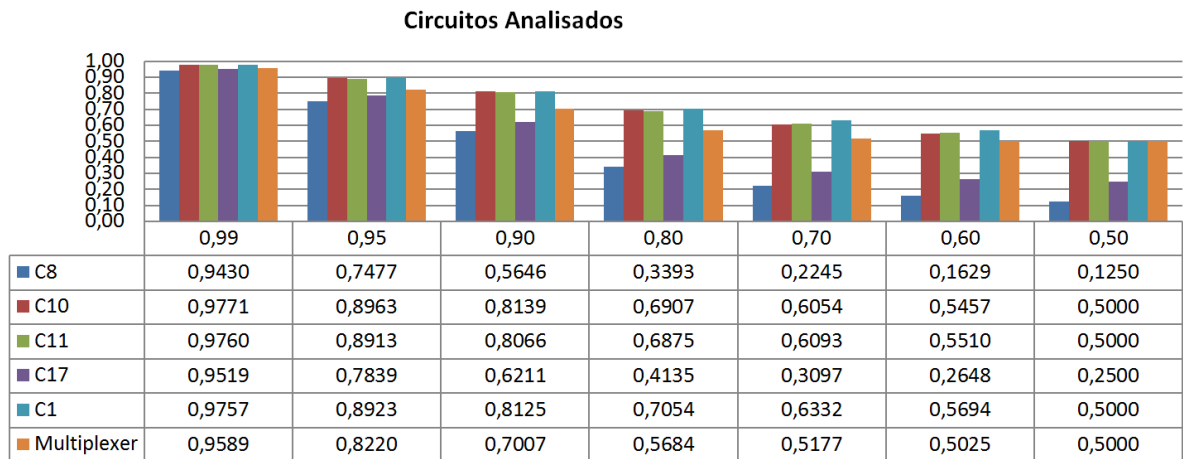


Figura 41: Outros circuitos analisados.

Tabela 6: Resultados com q variando de 0,99 a 0,999999.

(q)	C8	C9	C10	C11	C17	MULT.
0,999999	0,999994	0,999998	0,999998	0,999998	0,999995	0,999996
0,999990	0,999941	0,999975	0,999977	0,999975	0,999951	0,999958
0,999900	0,999413	0,999750	0,999766	0,999755	0,999506	0,999575
0,999000	0,994143	0,997508	0,997662	0,997551	0,995076	0,995765
0,990000	0,943031	0,975753	0,977164	0,976053	0,951928	0,958997

ser utilizado em um determinado projeto.

A Tabela 7 apresenta os dados da confiabilidade de outros circuitos analisados, desses circuitos os que apresentaram melhor resultado são: C10, C11, MULT. e C17. A Tabela 8 demonstra outras informações como número de entradas, saídas, número de portas lógicas, número de níveis e número de *fanouts* dos circuitos analisados.

Tabela 7: Confiabilidade para diferentes circuitos.

(q)	FA-v1	FA-v2	FA-v3	C8	C10	C11	C17	MULT.
1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
0,99	0,9489	0,9251	0,9252	0,9430	0,9771	0,9760	0,9519	0,9589
0,95	0,7727	0,6915	0,6936	0,7477	0,8963	0,8913	0,7839	0,8220
0,90	0,6051	0,5079	0,5116	0,5646	0,8139	0,8066	0,6211	0,7006
0,80	0,3971	0,3352	0,3361	0,3393	0,6907	0,6875	0,4135	0,5684
0,70	0,2964	0,2746	0,2723	0,2245	0,6054	0,6093	0,3097	0,5177
0,60	0,2558	0,2540	0,2528	0,1629	0,5457	0,5510	0,2648	0,5025
0,50	0,2500	0,2500	0,2500	0,1250	0,5000	0,5000	0,2500	0,5000
0,40	0,2641	0,2510	0,2490	0,0977	0,4541	0,4495	0,2482	0,5008
0,30	0,2836	0,2497	0,2418	0,0756	0,3937	0,3967	0,2510	0,5021
0,20	0,2828	0,2428	0,2259	0,0551	0,3080	0,3406	0,2558	0,5042
0,10	0,2148	0,2359	0,2124	0,0317	0,1954	0,2806	0,2639	0,5082
0,00	0,0000	0,2500	0,2500	0,0000	0,0703	0,2148	0,2812	0,5156

Tabela 8: Dados dos circuitos analisados.

Circuitos	Nº Entradas	Nº Saídas	Nº Portas	Nº Níveis	Nº Fanouts	Nº Figura
FA-v1	3	2	6	4	6	44
FA-v2	3	2	9	12	8	45
FA-v3	3	2	12	6	15	46
C10	10	1	10	8	1	42
C11	10	1	12	10	3	43
C20	16	1	20	14	4	49
C9	4	1	9	7	6	47
C8	4	3	8	6	7	48
MULT	6	1	7	5	6	54

6 CONSIDERAÇÕES FINAIS

Nesse trabalho foi desenvolvida uma ferramenta para determinar a confiabilidade de circuitos lógicos combinacionais que implementa o método PTM. A ferramenta desenvolvida está dividida em duas partes.

A primeira parte, é responsável por ler a descrição de *hardware* de um circuito e definir a profundidade lógica das portas lógicas, onde o resultado desse processo resulta na Descrição do Circuito Intermediária (DCI). Essa é a parte que demandou mais tempo de programação e também a mais complexa.

A segunda parte, o Gerador de Código *Scilab* (GCS) tem como entrada a DCI, e gera código compatível com *Scilab* que implementa o método PTM. Os resultados gerados demonstram que, com o auxílio da ferramenta desenvolvida, o processo de análise de circuitos tornou-se mais ágil, pois não necessita realizar as etapas que eram executadas manualmente. Esse processo poderá ser incluído em um processo de síntese que poderá escolher o circuito mais confiável para ser utilizado no projeto.

Diante das limitações encontradas apresentados algumas considerações a respeito do método PTM e, o que poderá ser feito em relação ao consumo de memória para que seja possível a análise de circuitos maiores. O consumo de memória é um limitador para o método PTM. Portanto, algumas abordagens podem ser adotadas para diminuir esse consumo. Uma das técnicas é limitar o número de elementos em um mesmo nível e propagar portas lógicas para o nível seguinte. Desse modo, o consumo de memória por nível poderá ser controlado ou reduzido. No código *Scilab* melhorias na estrutura do código podem ser investigadas para reduzir o consumo de memória com a eliminação das matrizes já computadas.

O processo de execução no *Scilab* consome muita memória e pode fazer com que o usuário acabe travando a máquina. Assim, um código pode ser criado para calcular qual a capacidade de memória que a análise do circuito vai exigir. Esse código poderá determinar a quantidade de memória necessária para cada nível do circuito. Outras tentativas para reduzir o consumo de memória podem ser adotadas como o particionamento do circuito de modo a formar blocos, como abordado em (KRISHNASWAMY et al., 2005). A utilização de matrizes esparsas também poderá expandir o método da PTM a circuitos maiores. As

matrizes esparsas têm como característica a maioria de suas posições preenchidas por elementos nulos. No entanto, podemos economizar um espaço significativo armazenando somente as posições com elementos diferentes de nulo.

A ferramenta desenvolvida e arquivos utilizados ficarão disponível em um CD na biblioteca da Universidade Federal do Rio Grande.

REFERÊNCIAS

AYERS, J. E. **Digital integrated circuits: analysis and design**. [S.l.]: CRC Press, Inc., 2009.

BAHAR, R. I.; FROHM, E. A.; GAONA, C. M.; HACHTEL, G. D.; MACII, E.; PARDO, A.; SOMENZI, F. Algebraic decision diagrams and their applications. In: COMPUTER-AIDED DESIGN, 1993. ICCAD-93. DIGEST OF TECHNICAL PAPERS., 1993 IEEE/ACM INTERNATIONAL CONFERENCE ON, 1993. **Anais...** [S.l.: s.n.], 1993. p.188–191.

BAUMANN, R. C. Radiation-induced soft errors in advanced semiconductor technologies. **Device and Materials Reliability, IEEE Transactions on**, [S.l.], v.5, n.3, p.305–316, 2005.

BHADURI, D.; SHUKLA, S. K.; GRAHAM, P. S.; GOKHALE, M. B. Reliability analysis of large circuits using scalable techniques and tools. **IEEE Transactions on Circuits and Systems I: Regular Papers**, [S.l.], v.54, n.11, p.2447–2460, 2007.

BUTZEN, P. F.; DAL BEM, V.; ROSA JR, L. da; REIS, A. I.; RIBAS, R. P. Efeitos Físicos Nanométricos em Circuitos Integrados Digitais. , [S.l.], p.1–20, 2012.

CAO, Y.; TSCHANZ, J.; BOSE, P. Guest editors' introduction: Reliability challenges in nano-CMOS design. **IEEE Design & Test of Computers**, [S.l.], v.26, n.6, p.6–7, 2009.

CHASE, O.; ALMEIDA, F. Sistemas embarcados. **Mídia Eletrônica. Página na internet:** <<http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf>>, capturado em **10/06/17**, [S.l.], v.10, n.11, 2007.

CHOUDHURY, M. R.; MOHANRAM, K. Accurate and scalable reliability analysis of logic circuits. In: DESIGN, AUTOMATION AND TEST IN EUROPE, 2007. **Proceedings...** [S.l.: s.n.], 2007. p.1454–1459.

ECOFFET, R. In-flight anomalies on electronic devices. In: **Radiation Effects on Embedded Systems**. [S.l.]: Springer, 2007. p.31–68.

ENTRENA, L.; VALDERAS, M. G.; CARDENAL, R. F.; GARCÍA, M. P.; ONGIL, C. L. SET emulation considering electrical masking effects. **IEEE Transactions on Nuclear Science**, [S.l.], v.4, n.56, p.2021–2025, 2009.

ERCOLANI, S.; FAVALLI, M.; DAMIANI, M.; OLIVO, P.; RICCO, B. Estimate of signal probability in combinational logic networks. In: EUROPEAN TEST CONFERENCE, 1989., PROCEEDINGS OF THE 1ST, 1989. **Anais...** [S.l.: s.n.], 1989. p.132–138.

FRANCO, D. T. **Fiabilidade do sinal dos circuitos lógicos combinatórios sob fautas simultâneas múltiplas**. 2008. Tese (Doutorado) — Télécom ParisTech.

FRANCO, D. T.; VASCONCELOS, M. C.; NAVINER, L.; NAVINER, J.-F. Reliability analysis of logic circuits based on signal probability. In: ELECTRONICS, CIRCUITS AND SYSTEMS, 2008. ICECS 2008. 15TH IEEE INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. p.670–673.

JEITLER, M.; DELVAI, M.; REICHOR, S. FuSE-a hardware accelerated HDL fault injection tool. In: PROGRAMMABLE LOGIC, 2009. SPL. 5TH SOUTHERN CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p.89–94.

KARNIK, T.; HAZUCHA, P. Characterization of soft errors caused by single event upsets in CMOS processes. **Dependable and Secure Computing, IEEE Transactions on**, [S.l.], v.1, n.2, p.128–143, 2004.

KRISHNASWAMY, S.; VIAMONTES, G. F.; MARKOV, I. L.; HAYES, J. P. Accurate reliability evaluation and enhancement via probabilistic transfer matrices. In: DESIGN, AUTOMATION AND TEST IN EUROPE, 2005. PROCEEDINGS, 2005. **Anais...** [S.l.: s.n.], 2005. p.282–287.

LALA, P. K. **Self-checking and fault-tolerant digital design**. [S.l.]: Morgan Kaufmann, 2001.

LEVIN, V. Probability analysis of combination systems and their reliability. **Engin. Cybernetics**, [S.l.], n.6, p.78–84, 1964.

LUZ REIS, R. A. da. **Ferramentas para a Simulação de Falhas Transientes**. 2011. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul.

MATTOS, J.; ROSA JR, L.; PILLA, M. **Desafios e avanços em computação: o estado da arte**. [S.l.]: Pelotas, 2009.

MOORE, G. Cramming more components onto integrated circuits. **Proceedings of the IEEE**, [S.l.], v.86, n.1, p.82–85, 1998.

NAVINER, L. A.; NAVINER, J.-F.; SANTOS, G. dos; MARQUES, E. C.; PAIVA, N. FIFA: A fault-injection–fault-analysis-based tool for reliability assessment at RTL level. **Microelectronics Reliability**, [S.l.], v.51, n.9, p.1459–1463, 2011.

NAVINER, L. A.; VASCONCELOS, M. C. de; FRANCO, D. T.; NAVINER, J.-F. Efficient computation of logic circuits reliability based on probabilistic transfer matrix. In: DESIGN AND TECHNOLOGY OF INTEGRATED SYSTEMS IN NANOSCALE ERA, 2008. DTIS 2008. 3RD INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. p.1–4.

OLDHAM, T. R.; MCLEAN. Total ionizing dose effects in MOS oxides and devices. **IEEE Transactions on Nuclear Science**, [S.l.], v.50, n.3, p.483–499, 2003.

PATEL, K. N.; MARKOV, I. L.; HAYES, J. P. Evaluating circuit reliability under probabilistic gate-level fault models. In: INTERNATIONAL WORKSHOP ON LOGIC AND SYNTHESIS, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.59–64.

PATTERSON, D.; HENNESSY, J. L. **Arquitetura de Computadores**: uma abordagem quantitativa. [S.l.]: Elsevier Brasil, 2014. v.5.

RABAEY, J. M. **Digital Integrated Circuits—A Design Perspective**. [S.l.]: 2nd ed. Upper Sanddle River: Prentice Hall, 1996.

RABAEY, J. M.; CHANDRAKASAN, A. P.; NIKOLIC, B. **Digital integrated circuits**. [S.l.]: Prentice hall Englewood Cliffs, 2002. v.2.

RIBEIRO, I. d. S. **Modelagem e Caracterização da Propagação de Pulsos Transientes Causados por Radiação Ionizante**. 2010. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul.

SARRAFZADEH. **An Introduction to VLSI Physical Design**. [S.l.]: 1st ed. San Francisco: Mc Graw - Hill, 1996.

SHIVAKUMAR, P.; KISTLER, M.; KECKLER, S. W.; BURGER, D.; ALVISI, L. Modeling the effect of technology trends on the soft error rate of combinational logic. In: DEPENDABLE SYSTEMS AND NETWORKS, 2002. DSN 2002. PROCEEDINGS. INTERNATIONAL CONFERENCE ON, 2002. **Anais...** [S.l.: s.n.], 2002. p.389–398.

SOLEN. **Solar Terrestrial Activity Report**.

SVALGAARD, L.; CLIVER, E. W.; KAMIDE, Y. Sunspot cycle 24: Smallest cycle in 100 years? **Geophysical Research Letters**, [S.l.], v.32, n.1, 2005.

TABER, A.; NORMAND, E. Single event upset in avionics. **IEEE Transactions on Nuclear Science**, [S.l.], v.40, p.120–126, 1993.

WESTE, N. H.; HARRIS, D. M. **CMOS VLSI design**: a circuits and systems perspective. [S.l.]: Pearson Education India, 2005.

XIAO, J.; JIANG, J.; ZHU, X.; OUYANG, C. A method of gate-level circuit reliability estimation based on iterative PTM model. In: DEPENDABLE COMPUTING (PRDC), 2011 IEEE 17TH PACIFIC RIM INTERNATIONAL SYMPOSIUM ON, 2011. **Anais...** [S.l.: s.n.], 2011. p.276–277.

ZIMPECK, A. L.; MEINHARDT, C.; BUTZEN, P. F. Análise do comportamento de portas lógicas CMOS com falhas Stuck-On em nanotecnologias. , [S.l.], 2014.

APÊNDICE A CIRCUITOS ANALISADOS

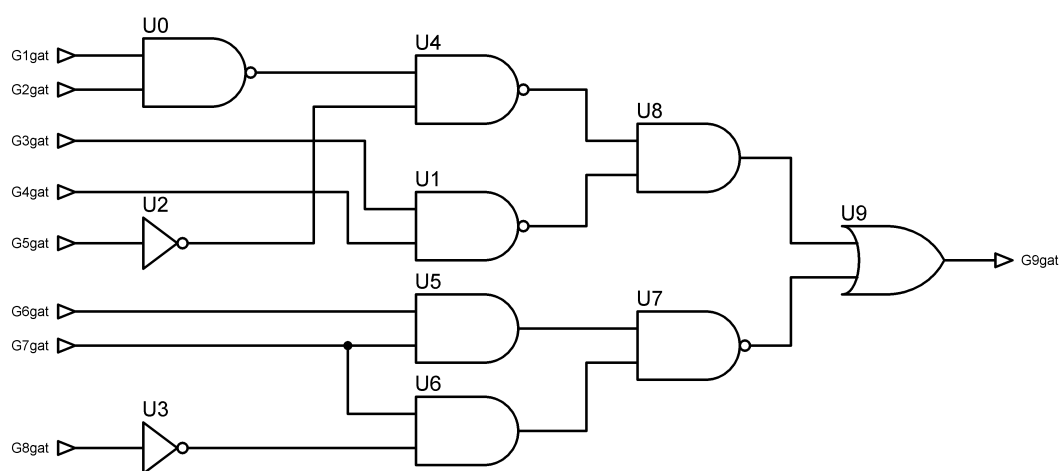


Figura 42: Circuito C10.

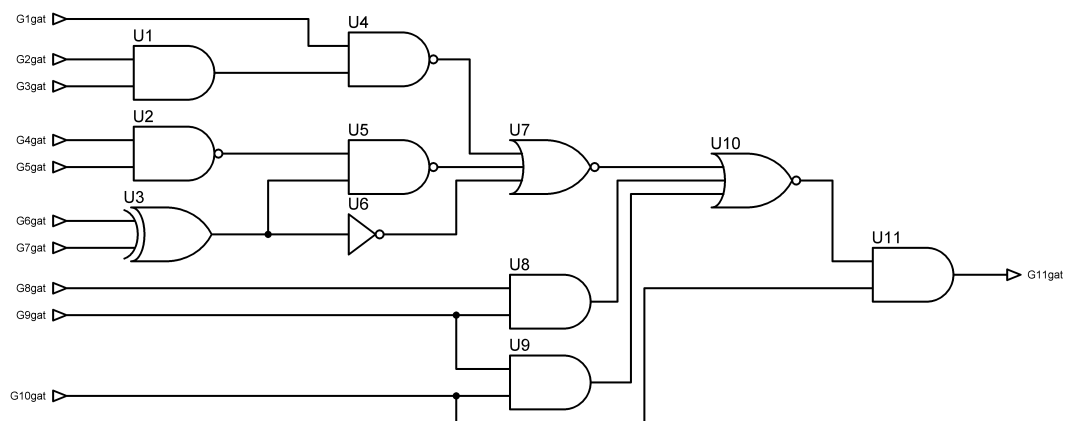


Figura 43: Circuito C11.

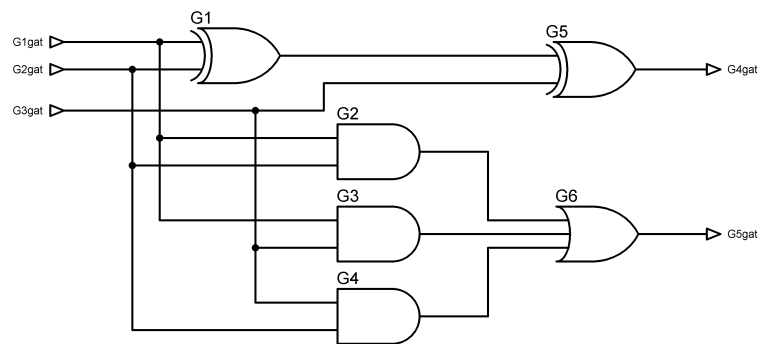


Figura 44: Full Adder-v1.

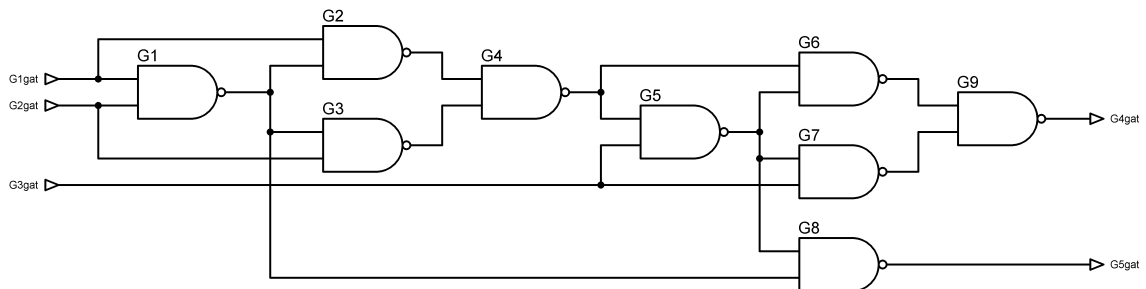


Figura 45: Full Adder-v2.

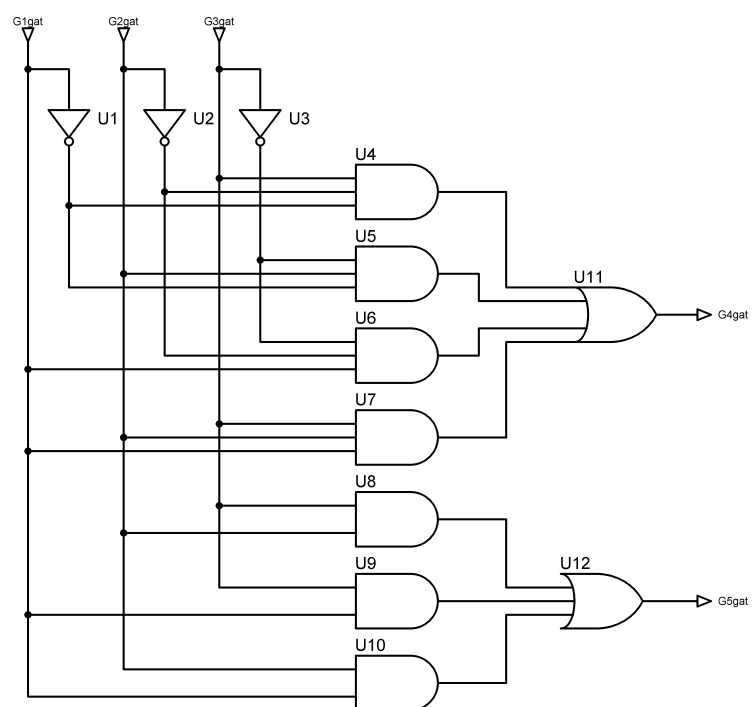


Figura 46: Full Adder-v3.

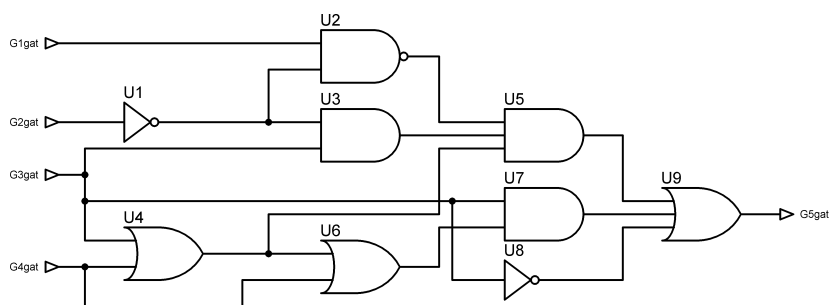


Figura 47: C9.

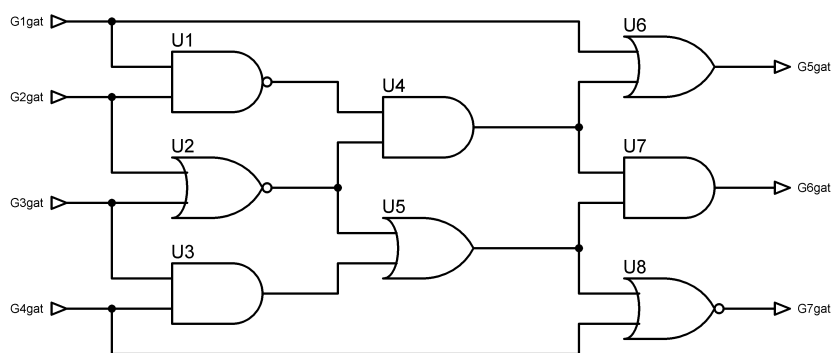


Figura 48: C8.

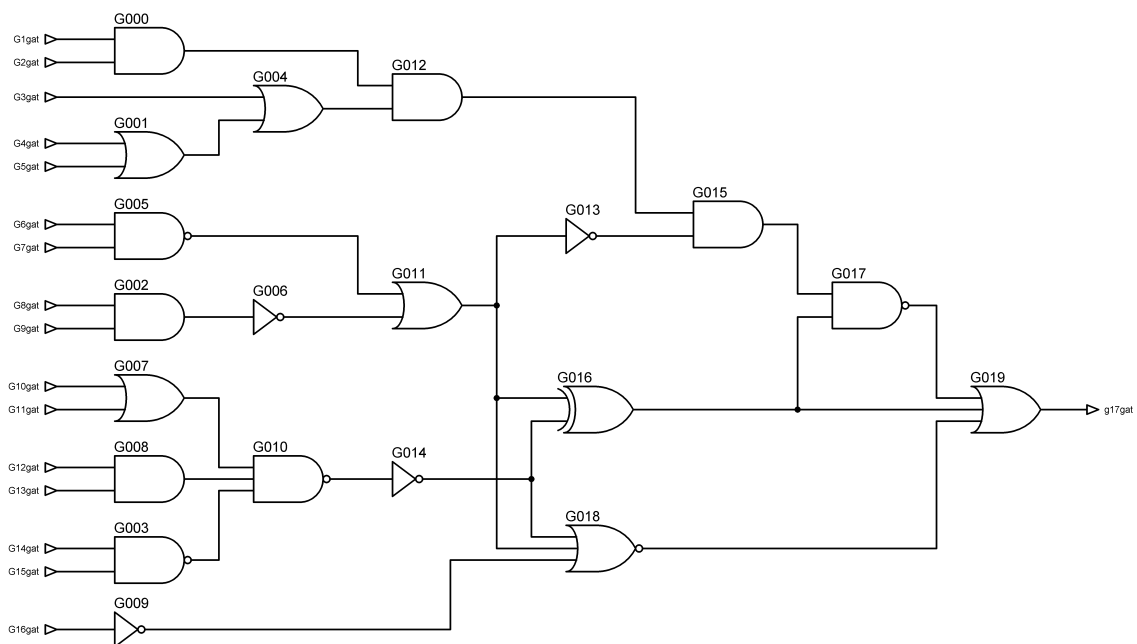


Figura 49: C20.

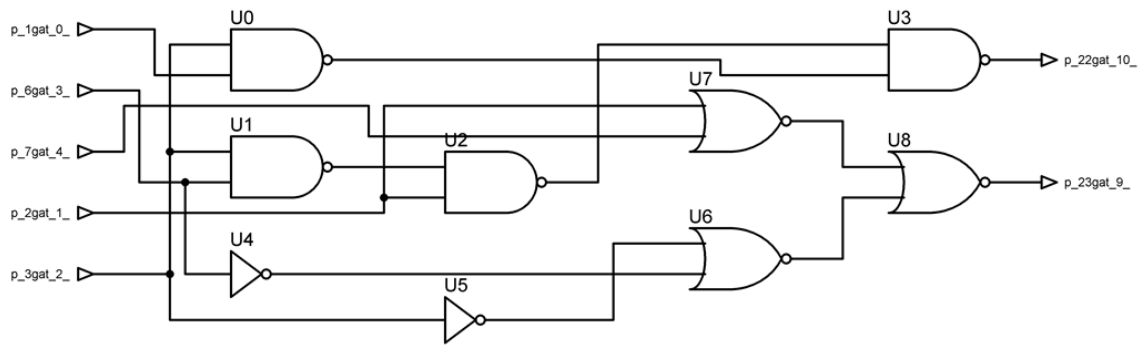


Figura 50: C17-v1.

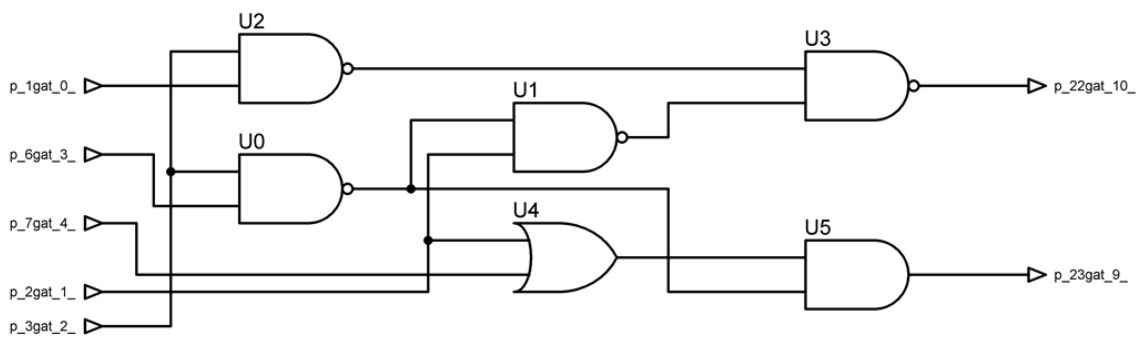


Figura 51: C17-v2.

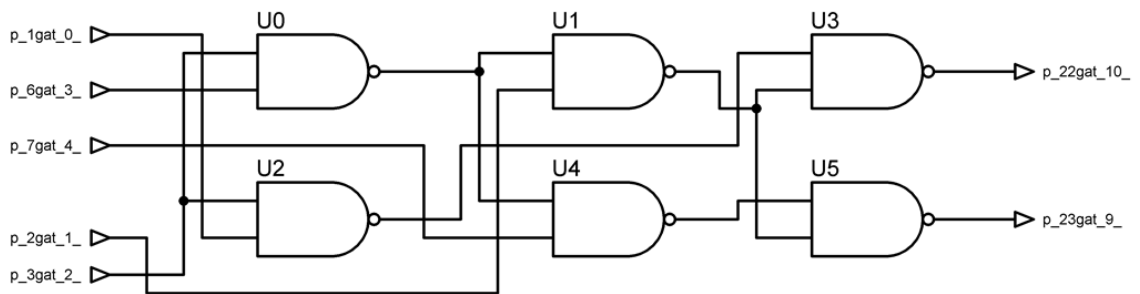


Figura 52: C17-v3.

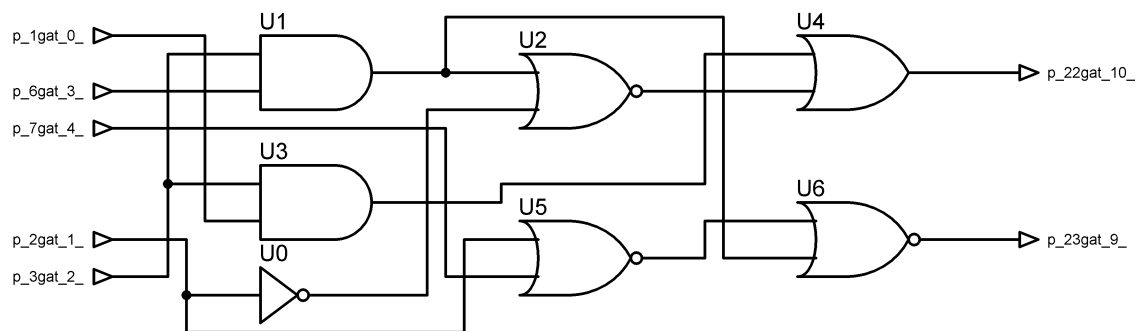


Figura 53: C17-v4.

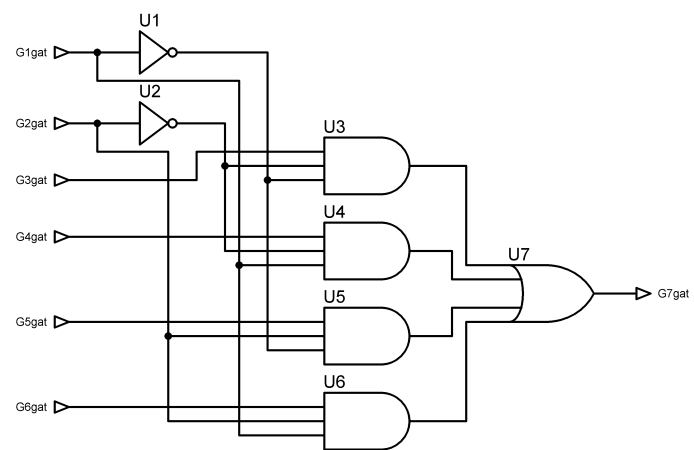


Figura 54: Multiplexer.