Eduardo Liebl

Avaliação da Robustez de Diferentes Topologias de Circuitos Votadores

Brasil

Rio Grande, 2016

Eduardo Liebl

Avaliação da Robustez de Diferentes Topologias de Circuitos Votadores

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 da Computação

Universidade Federal do Rio Grande – FURG

Centro de Ciências Computacionais

Programa de Pós-Graduação em Computação

Curso de Mestrado em Engenharia da Computação

Orientador: Prof. Dr. Paulo Francisco Butzen

Coorientador: Profa. Dra. Cristina Meinhardt

Brasil

Rio Grande, 2016

Ficha catalográfica

L716c Liebl, Eduardo.
Avaliação da robustez de diferentes topologias de circuitos votadores / Eduardo Liebl. – 2016.
76 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-graduação em Modelagem Computacional, Rio Grande/RS, 2016.
Orientador: Dr. Paulo Francisco Butzen.
Coorientadora: Drª. Cristina Meinhardt.

1. Tolerância a falhas 2. Redundância modular tripla. 3. Votadores majoritários 4. Robustez I. Butzen, Paulo Francisco II. Meinhardt, Cristina III. Título.

CDU 004.621.3

Catalogação na Fonte: Bibliotecário Me. João Paulo Borges da Silveira CRB 10/2130

UNIVERSIDADE FEDERAL DO RIO GRANDE

Centro de Ciências Computacionais

Programa da Pós-Graduação em Computação

Curso de Mestrado em Engenharia de Computação

ATA DE SESSÃO DE DEFESA DE DISSERTAÇÃO DE MESTRADO

Ata No. 01 /2016

Na data de 22 de março de 2016, às 15 horas, ocorreu a Sessão de Defesa de Dissertação de Mestrado de Eduardo Liebl, que apresentou a dissertação intitulada Avaliação da Robustez de Diferentes Topologias de Circuitos Votadores, realizada sob a orientação do Prof. Paulo Francisco Butzen e co-orientação da Profa. Cristina Meinhardt. A banca examinadora foi constituída pelos Profs. Denis Teixeira Franco (UFPEL), José Rodrigo Furlanetto Azambuja (FURG) e Leomar Soares da Rosa Junior (UFPEL), sob a presidência do orientador. Após a apresentação do trabalho, a banca arguiu o candidato e, a seguir, deliberou pela

(X) aprovação da Dissertação

( ) aprovação da Dissertação, sugerindo modificações no texto

( ) reprovação da Dissertação

Rio Grande, 22 de março de 2016

Prof. Dr. Denis Teixeira Franco

Prof. Dr José Rodrigo Furlanetto Azambuja

Prof. Dr. Leomar Soares da Rosa Junior

Prof. Dr. Paulo Francisco Butzen
Orientador

Agradecimentos

Ao Dr. Paulo Francisco Butzen pela impecável orientação deste trabalho, paciência, dedicação e por ser um grande incentivador da superação dos meus limites.

À Dra. Cristina Meinhardt pela paciência, dedicação e carinho durante a orientação deste trabalho.

Ao Dr. José Rodrigo Azambuja, que acompanhou meu processo de formação, pelo apoio, incentivo e contribuições ao meu trabalho.

Ao Dr. Denis Franco, que também acompanhou meu processo de formação, pelo apoio, incentivo e contribuições ao meu trabalho.

Ao Dr. Leomar S. Rosa Junior pelas contribuições com o trabalho.

A Ariany Rabello da Silva, por sempre estar ao meu lado, sendo meu principal exemplo de superação e dedicação, pelo carinho e apoio durante meu processo de formação.

Ao meu amigo Helder Henrique Avelar, por ser meu exemplo de esforço e dedicação e pelo apoio ao meu trabalho.

Ao meu amigo Daniel Buchinger Junior pela ajuda prestada no desenvolvimento do algoritmo de simulação.

À minha família e amigos que sempre estiveram me apoiando, tornando esse trabalho possível.

Agradecimentos especiais ao Centro de Ciências Computacionais (C3), da Universidade Federal do Rio Grande (FURG), e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) pelo financiamento deste trabalho.

"E tu, Daniel, fecha estas palavras e sela
este livro, até o fim do tempo; muitos correrão
de uma parte para outra, e a ciência se multiplicará.
(Bíblia Sagrada, Daniel 12, 4)

Resumo

Com a miniaturização da tecnologia, alguns problemas, tais como a redução da confiabilidade nos processos de fabricação de circuitos integrados e o aumento da vulnerabilidade dos sistemas integrados a partículas de alta energia, tornou-se uma grande preocupação no projeto de circuitos nanométricos. Sistemas projetados para a segurança, missões críticas, aplicações aéreas e espaciais, entre outras, fazem uso de técnicas de tolerância a falhas para desenvolver sistemas confiáveis com base em componentes eletrônicos não confiáveis. Uma técnica amplamente utilizada chama-se redundância modular tripla (TMR). A arquitetura TMR é livre de falhas únicas, exceto quando a falha ocorre no votador de maioria. Quando uma falha ocorre no votador, um erro pode ser propagado pelo sistema. Por este motivo, é importante investigar a robustez e as principais características elétricas de diferentes implementações de votadores.

Neste trabalho, foi realizada uma extensa revisão de projetos de votadores majoritários. Diferentes abordagens de votadores majoritários serão avaliadas em termos de desempenho, consumo de energia e taxa de erros para falhas permanentes e transientes. Todos os votadores majoritários executam corretamente a lógica para a qual foram concebidos. No entanto, quando avaliados em condições igualitária, para fazer uma comparação justa entre eles, eles mostram diferentes resultados elétricos e de taxa de erros. Os melhores resultados apresentaram até 3% de taxa de erros para falhas permanentes, enquanto os piores resultados atingiram até 15%. O trabalho também demonstra que falhas transientes são um aspecto importante em tecnologias nanométricas, porque o melhor resultado apresentou cerca de 15% de taxa de erros para falhas transientes e no pior dos casos mostrou cerca de 28% de taxa de erros. Com os resultados apresentados neste trabalho, é possível encontrar votadores com diferentes compromissos entre as características de taxa de erros, desempenho e consumo de energia. **Palavras-chave**: tolerância a falhas, redundância modular tripla, votadores majoritários, robustez.

Abstract

As the tecnology scales, some problems such as the decreased reliability on integrated circuits fabrication processes and the increasing vulnerability of integrated systems to high energy particles has become a major concern in nanometer circuit designs. Systems designed for safety, critical missions, air and space applications, among others, make use of fault-tolerant techniques to develop reliable systems based on unreliable electronic components. A widely used technique is called triple-modular redundancy (TMR). The TMR architecture is single fault free except when the fault occurs in the majority voter. When a fault occurs in the voter, an error may be propagated throughout the system. For this reason, it is important to investigate the robustness and the main electrical characteristics of different voters implementations.

In this work, an extensive review of majority voters designs was performed. Different approaches of majority voters will be evaluated in terms of performance, power consumption and error rate for permanent and transient faults. All majority voters perform properly the logic purpose for which they were designed. However, when evaluated in equalitary conditions, to make a fair comparison between them, they showed different electrical and error rate results. The best results showed about 3% of error rate for permanent faults and the worst results showed up to 15%. Work also demonstrates that transient faults are an important aspect in nanometric technologies, because the best result presented about 15% of error rate for transient faults and the worst case showed about 28% of error rate. With the results presented in this work it is possible to find majority voters with different balance between error rate, performance and power consumption characteristics.

Keywords: fault tolerance, triple modular redundancy, majority voters, robustness.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ATMR	*Aproximate Triple Modular Redundancy*
CMOS	*Complementary Metal Oxide Semiconductor*
CST	Circuito Sob Teste
CWSP	*Code Word State Preserving*
DTMR	*Diversity Triple Modular Redundancy*
FSM	*Finite State Machine*
HP	*High Performance*
LET	*Linear Energy Transfer*
NMR	*N-Modular Redundancy*
SEE	*Single Event Effect*
SET	*Single Event Transient*
SEU	*Single Event Upset*
TMR	*Triple Modular Redundancy*
VLSI	*Very Large-Scale Integration*

Sumário

Introdução

Gordon E. Moore elaborou uma estimativa de desenvolvimento das tecnologias semicondutoras, no qual o número de transistores em um circuito integrado dobraria a cada 18 meses. A indústria microeletrônica estabeleceu como meta de evolução a lei de Moore (MOORE, 1998). Desde então o número de transistor em um circuito integrado vem aumentando devido à miniaturização desses componentes, possibilitando uma integração cada vez maior de funcionalidades em uma mesma área de silício. O nível de integração de transistores chegou a um ponto que começou a ser classificado como *very large-scale integration* (VLSI).

Dennard, Gansslen e YU (1974) afirmaram que quanto maior a miniaturização do transistor, mais rápido ele se torna, menos energia ele consome e a sua produção fica mais barata. Apesar disso, o consumo de energia continua a ser um problema no projeto VLSI, pois o consumo dinâmico e as correntes de fuga dos transistores, que antes eram ignorados, passaram a ser relevantes quando considerada a operação de milhões de transistores (LI et al., 2009). A constante redução nas dimensões dos transistores também aumentou consideravelmente a complexidade do processo de fabricação, tornando os circuitos fabricados em tecnologias nanométricas muito mais suscetíveis a falhas permanentes causadas por contaminação, contatos impróprios, corrosão, oxidação, etc (COLBOURNE; COVERLEY; BEHERA, 1974; SCHNABLE; GALLACE; PUJOL, 1978; BOLIN, 1980; MANGIR, 1985). Além disso os circuitos começaram a se tornar mais sensíveis a falhas causadas por incidência de radiação. As falhas por radiação que antes eram uma preocupação apenas para o setor espacial, também começaram a ter importância no nível terrestre (BAUMANN, 2005a). O aumento da ocorrência de falhas permanentes e transientes nos circuitos integrados tem um efeito direto sobre a confiabilidade dos sistemas. Por este motivo passa a ser importante a utilização de técnicas de tolerância a falhas em aplicações críticas e sistemas que exijam alta confiabilidade.

Os sistemas computacionais estão cada vez mais presentes no cotidiano das pessoas na forma de *smartphones*, *tablets*, computadores e outros dispositivos pessoais. Também estão presentes como peças vitais de carros, equipamentos de medicina, aeronaves, plantas industriais, geração de energia, exploração espacial e comunicação. Além disso, os servidores atuais permitem um grande volume de transações entre instituições financeiras, propiciando que o comércio eletrônico (*e-commerce*) movimente grandes quantidades de dinheiro todos os dias. De todo modo, espera-se que esses sistemas sempre funcionem adequadamente (KOREN; KRISHNA, 2010).

Em áreas críticas, como medicina, aplicações militares e espaciais, é exigido um

alto nível de confiabilidade dos sistemas computacionais. O sistema não pode simplesmente travar ou deixar de funcionar, como ocasionalmente ocorre com os computadores pessoais. Pode-se citar alguns exemplos das consequências que falhas no sistema computacional podem trazer: Em 4 de junho de 1996, o foguete espacial Ariane 501, da ESA (Agência Espacial Europeia), explodiu em menos de 1 minuto após o lançamento devido a falhas do sistema computacional (LANN, 1997). Outro evento aconteceu em 6 de agosto de 1999, quando o site do eBay ficou paralisado por aproximadamente 9 horas em decorrência de uma falha dos seus sistemas (CNNMONEY, 1999). Engenheiros e cientistas da computação tem desenvolvido ferramentas e técnicas para reduzir a quantidade de falhas dos sistemas, porém, mais estudos e pesquisas ainda são necessários. Para evitar tais situações, é preciso fazer uso de técnicas de tolerância a falhas para que o sistema continue funcionando corretamente (LI et al., 2009).

A maioria das técnicas de tolerância a falhas utilizam redundância, que é a utilização de mais recursos que o necessário para o funcionamento do sistema. Para lidar com falhas, existem três tipos de redundância: redundância temporal, de informação e de hardware. Motivados pela técnica de Neumann (1956) que apresentava componentes redundantes para tolerar falhas, surgiram outras arquiteturas otimizadas como o NMR, abreviação de *N-tuple Modular Redundancy*. Quando o número de módulos utilizados é igual a três passa a ser conhecido como TMR (*Triple Modular Redundancy*). O TMR é a técnica de tolerância a falhas mais utilizada em sistemas de alta confiabilidade (VIAL et al., 2008). É uma técnica muito eficaz por ser capaz de tolerar 100% de falhas únicas em qualquer parte do circuito, ao custo de ter um aumento de pelo menos 200% em área de chip e consequente aumento de consumo de energia. O TMR se baseia em utilizar três módulos idêntico processando a mesma informação. No entanto a saída dos três módulos converge para o votador majoritário. O votador majoritário é o ponto crítico do TMR, pois caso uma falha ocorra nele, a saída do circuito pode atingir um estado de erro. Dada a sua importância nas arquiteturas TMR, o votador majoritário tem recebido grande atenção por parte da comunidade científica e diversos circuitos foram propostos para tentar aumentar a confiabilidade dos circuitos votadores (KSHIRSAGAR; PATRIKAR, 2009; BAN; De Barros Naviner, 2010; BAZE; BUCHNER; MCMORROW, 2000; ALMUKHAIZIM; SINANOGLU, 2007; CAZEAUX; ROSSI; METRA, 2005; EL-RAZOUK; ABID, 2007).

0.1 Objetivo

Este trabalho tem como principal objetivo avaliar diferentes implementações de votadores majoritários quanto a robustez à falhas permanentes e transientes. As características de desempenho e dissipação de potência dos circuitos também serão observadas.

0.1.1 Objetivos Específicos

- Realizar um levantamento de diferentes circuitos votadores propostos na literatura;
- Avaliar as características de desempenho e dissipação de potência destes votadores majoritários, de forma igualitária, em uma tecnologia nanométrica;
- Avaliar a robustez dos votadores em relação a falhas permanentes e a falhas transientes;
- Produção de artigos científicos para disseminação dos resultados encontrados.

1 Fundamentação Teórica

1.1 Conceito de tolerância a falhas

Sistemas computacionais são provavelmente os sistemas mais complexos já criados pela humanidade. Essa complexidade vem aumentando a cada geração para atender a demanda por maior capacidade de processamento, maior capacidade de armazenamento e maior integração de funções. Com o aumento da complexidade dos sistema computacionais também é esperado que eles se tornem mais propensos a falhas. A evolução dos sistemas computacionais tem explorado extensivamente a miniaturização dos transistores, resultando em circuitos integrados com uma alta densidade de transistores. O aumento da densidade de transistores aumenta a complexidade do projeto e dos processos de fabricação, ficando cada vez mais difícil garantir que os sistemas estejam livres de falhas. Como a evolução da tecnologia tem se estendido para áreas críticas como exploração espacial, aplicações médicas, aplicações militares, setor de transporte, entre outras, é imprescindível garantir que esses sistemas sempre funcionem adequadamente. O desafio da área de tolerância a falhas é assumir que os sistemas computacionais possuem ou podem apresentar falhas durante a sua vida útil e aplicar técnicas para que esses sistemas continuem funcionando adequadamente mesmo na presença de falhas.

1.1.1 Falha, Erro e Defeito

Termos como falha, erro e defeito são usados para dizer que algo não está funcionando corretamente. Em tolerância a falhas, esses termos tem significados distintos. Uma falha pode ser entendida como um estado incorreto do hardware ou *software* e é causada por problemas físicos de componentes, erros de projeto ou causada por efeitos de radiação. A manifestação de uma falha causa um erro. Erros podem se propagar pelo sistema, pois a saída incorreta de um componente será utilizada como entrada para outros componentes fazendo com que esses valores computados incorretamente se propagem pelo sistema. A manifestação de um erro a nível de sistema é chamada de defeito (KOREN; KRISHNA, 2010).

Para ilustrar a diferença entre os conceitos, vamos supor que uma porta lógica tenha uma falha, fazendo com que sua saída sempre tenha o valor lógico 1. Se ela for requisitada em uma operação onde o valor lógico esperado seja 0, então um erro é observado na sua saída. Caso o valor lógico esperado da operação seja 1, então a falha será mascarada e o erro não é observado. Um erro pode causar um comportamento incorreto do sistema computacional, chamado de defeito. Isso ocorre quando o erro é propagado pelo sistema,

fazendo com que esses valores sejam computados incorretamente, e quando estes atingirem uma saída observável pelo usuário, um defeito é caracterizado. Este defeito pode ser uma simples falha na computação mas também é possível causar travamentos ou até colapso do sistema (RIVERS; KUDVA, 2009).

De acordo com Johnson (1996), o conceito de falha, erro e defeito pode ser apresentado em um modelo de três universos como pode ser visto na Figura 1. As falhas ocorrem no universo físico, e são anomalias nos componentes físicos como transistores, interconexões, resistores, etc. Os erros ocorrem no universo da informação e provocam valores incorretos em dados que podem ser armazenados em elementos de memória ou processados pelo sistema. Os defeitos ocorrem no universo do usuário ou universo externo, que é quando um erro propagado atinge o nível de sistema e provoca comportamentos inesperados e muitas vezes pode ser percebido pelo usuário como travamentos ou falhas críticas do sistema.

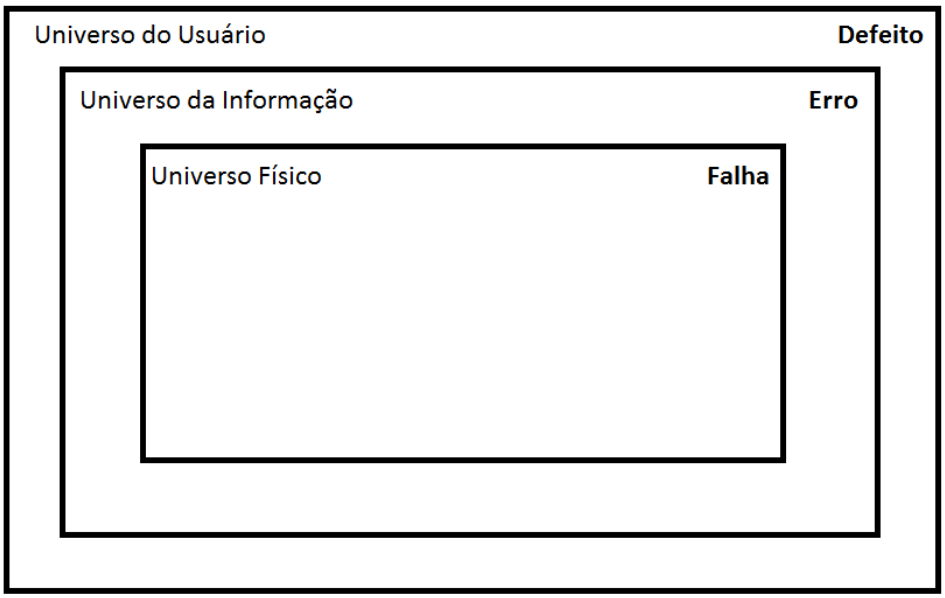


Figura 1 – Modelo de três universos: falha, erro e defeito

1.1.2 Mascaramento

Quando uma falha não se manifesta na forma de uma erro, ela foi mascarada, e consequentemente não levou o sistema a um comportamento inesperado. Existem técnicas de tolerância a falhas que exploram os conceitos de mascaramento de falhas. Existem três tipos de mascaramentos: mascaramento lógico, elétrico e por janela de amostragem. Cada um deles será apresentado em mais detalhes nas seções seguintes.

1.1.2.1 Mascaramento Elétrico

Este tipo de mascaramento está normalmente associado a falhas transientes causadas por partículas radioativas. Estas partículas, ao incidirem no circuito geram um pulso de corrente elétrica. O mascaramento elétrico ocorre quando o pulso elétrico gerado pela falha é atenuado conforme ele se propaga através das portas lógicas.

A atenuação ocorre tanto na amplitude do transiente quanto em seus tempos de subida e descida, causando eventualmente, o desaparecimento do pulso. Um exemplo pode ser visto na Figura 2.

Nem sempre o pulso gerado é mascarado eletricamente. Dentre os diversos fatores que influenciam o mascaramento estão a amplitude e duração do pulso e o atraso da porta lógica em questão.

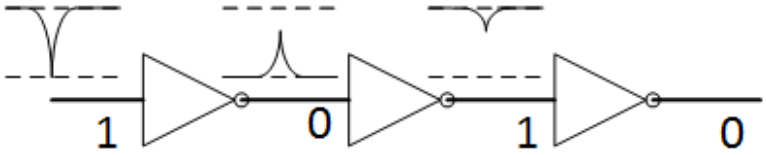


Figura 2 – Mascaramento Elétrico

1.1.2.2 Mascaramento Lógico

O mascaramento lógico ocorre quando uma falha afeta uma entrada que não é capaz modificar a saída da porta lógica. No caso da Figura 3 o pulso elétrico aconteceu em um estado lógico das portas NOR e NAND, que não provoca transição nessas condições, em função do valor aplicado à outra entrada da porta lógica.

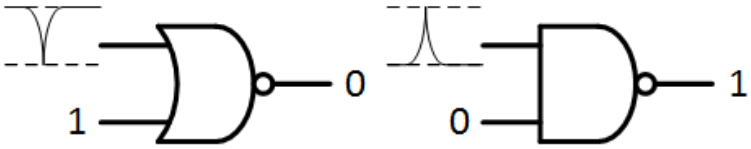


Figura 3 – Mascaramento Lógico

1.1.2.3 Mascaramento Temporal

A janela de amostragem é o período de tempo em torno da borda de transição do sinal de *clock*, responsável por controlar o armazenamento dos sinais das linhas de dados nos elementos de memória. Esse mascaramento está relacionado a falhas transientes, quando estas ocorrem fora da janela de amostragem. A Figura 4 mostra um exemplo, quando pulsos gerados por falhas transientes ocorrem fora da janela de amostragem e não se propagam pelo sistema.

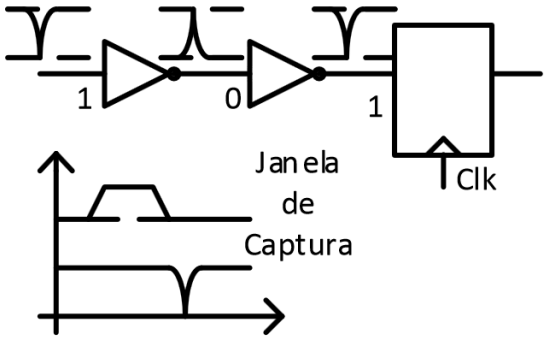


Figura 4 – Mascaramento Temporal por Janela de Amostragem

1.1.3 Classificação de Falhas

Falhas podem ser classificadas de acordo com sua duração temporal em transientes, intermitentes ou permanentes (KOREN; KRISHNA, 2010). Falhas transientes são falhas que ocorrem por um curto período de tempo e desaparecem. Apesar de durarem um curto período de tempo, são um grande problema pois podem ser capturadas por elementos de memória que armazenam e propagam o erro. Falhas permanentes se referem a componentes que saíram de sua especificação durante o processo de fabricação ou componentes que pararam de funcionar adequadamente devido a efeitos de envelhecimento. Falhas permanentes são irreversíveis e sempre estarão presentes no sistema. Falhas intermitentes aparecem esporadicamente na mesma parte do sistema e são muitas vezes tratadas como falhas transientes ou permanentes.

Existem diversos tipos de falhas que podem ocorrer nos circuitos VLSI. É necessário entender a natureza dessas falhas para gerar testes adequados aos circuitos. Geralmente, um bom modelo de falhas deve representar de forma satisfatória a natureza da falha e deve ser computacionalmente eficiente para a simulação das mesmas (WANG; WU; WEN, 2006). Falhas podem ser descritas em vários níveis de abstração do sistema, de acordo com o escopo em que deseja-se avaliar as mesmas. Quando uma falha é descrita em um nível de abstração muito baixo, em nível de transistor por exemplo, pode descrever com precisão o fenômeno físico da falha, porém torna-se custosa a simulação em sistemas muito complexos onde o número de transistores é muito grande. Já falhas descritas em um nível de abstração alto, conseguem lidar muito bem com sistemas complexos ao custo de simplificações no modelo.

1.2 Falhas Permanentes

Um modelo de falhas muito utilizado por décadas é o *stuck-at* (MCCLUSKEY; CLEGG, 1971). Esse modelo de falhas é utilizado no nível de portas lógicas. Ele se refere a falhas que afetam o estado lógico das linhas dos sinais em entradas e saídas das portas

lógicas do circuito. A falha do tipo *stuck-at* faz o sinal de um nodo do circuito ficar constantemente com o valor lógico '0' ou '1', respectivamente denominados de *stuck-at-0* e *stuck-at-1*.

Considere como exemplo o votador da Figura 5, onde os nodos A, B, C e S podem ser afetados por falhas *stuck-at*. No circuito do exemplo foram inseridas 4 falhas do tipo *stuck-at-0* e 4 falhas do tipo *stuck-at-1*, totalizando 8 falhas. A Tabela 1 mostra as saídas para o circuito sem falhas e para o circuito com falhas para todos os vetores de entrada, onde SA0 representa que foi inserida uma falha *stuck-at-0* e SA1 representa que foi inserida uma falha *stuck-at-1*.

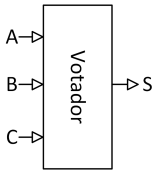


Figura 5 – Votador Majoritário

Esse tipo de teste é considerado exaustivo, pois foram consideradas 1 falha em cada nó para todos os vetores de entrada. É possível calcular a cobertura de falhas do circuito pela Equação 1.1. De acordo com a Equação 1 (WANG; WU; WEN, 2006), a cobertura de falhas do tipo stuck-at para o votador, é de 100% quando excitado por todos os possíveis vetores de entrada.

Tabela 1 – Saídas esperados para o votador votador com falhas e sem falhas

Vetores de Entrada	**Saída Esperada**	**Saída esperada com presença de falha em cada nó**							
		A SA0	**A SA1**	**B SA0**	**B SA1**	**C SA0**	**C SA1**	**C SA0**	**C SA1**
000	0	0	0	0	0	0	0	0	1
001	0	0	1	0	1	0	0	0	1
010	0	0	1	0	0	0	1	0	1
011	1	1	1	0	1	0	1	0	1
100	0	0	0	0	1	0	1	0	1
101	1	0	1	1	1	0	1	0	1
110	1	0	1	0	1	1	1	0	1
111	1	1	1	1	1	1	1	0	1

$$\textbf{Cobertura de Falhas} = \frac{\textbf{Número de Falhas Detectadas}}{\textbf{Número Total de Falhas}} \tag{1.1}$$

Os vetores de teste gerados pelo modelo *stuck-at* detectam muitas das falhas que acontecem em circuitos combinacionais, porém não necessariamente garantem a detecção de todas as falhas, por isso outros modelos de falhas são necessários.

As falhas *Stuck-On* (Kuang-Wei Chiang; VRANESIC, 1983) e *Stuck-Open* (WADSACK, 1978) são falhas permanentes que representam muitas das falhas que ocorrem nos circuitos integrados (ABRAHAM; FUCHS, 1986). São falhas que ocorrem no nível de transistores fazendo com que fiquem permanentemente conduzindo ou permanentemente

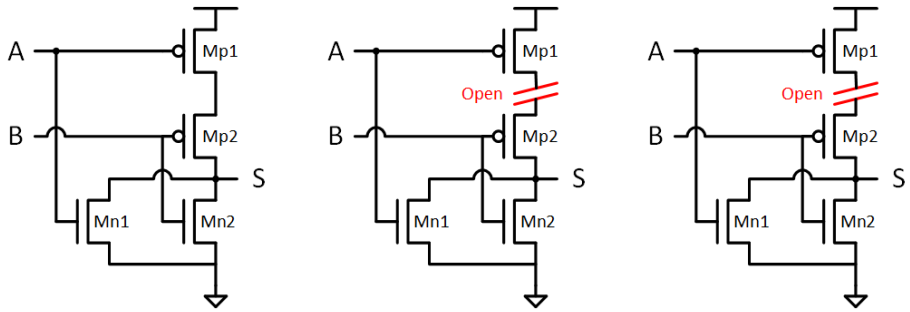


Figura 6 – Representação de falhas *stuck-open* e *stuck-on* em uma porta NOR

bloqueados. O modelo de falhas *Stuck-at* não detecta todos os efeitos que as falhas *Stuck-On* e *Stuck-Open* causam nos circuitos combinacionais. Quando o circuito analisado não tiver uma complexidade elevada, aplicar o modelo de falhas *Stuck-On* e *Stuck-Open* é mais indicado para se obter uma melhor compreensão da robustez do sistema.

1.2.1 Falhas Stuck-On

Falhas *Stuck-On* (Kuang-Wei Chiang; VRANESIC, 1983) emulam um transistor com falhas que fazem com que ele fique permanentemente no estado de condução. Tomando como exemplo a porta lógica NOR da Figura 6 (a), é um clássico exemplo de uma porta lógica que utiliza redes *pull-up* e *pull-down* para alternar o valor lógico da saída entre 0 e 1. A saída da porta NOR somente será 1 quando as entradas AB = 00. Se houver uma falha *stuck-on* no transistor MN1, como exemplificado na Figura 6 (b), a rede *pull-down* estará permanentemente conduzindo. É possível concluir que quando AB = 00, a rede *pull-up* e *pull-down* encontram-se conduzindo ao mesmo tempo. A saída será um divisor resistivo que pode ou não ser interpretado como uma falha pelas portas lógicas subsequentes. Outra característica dessa falha é a alta corrente de curto-circuito que será drenada da fonte de alimentação.

A quantidade de vetores de entrada necessários para testar falhas stuck-on é calculada pela Equação 1.2, onde E é o número de entradas. No caso da porta NOR que possui 2 entradas, são necessários 4 vetores para testar todas as possíveis falhas, ou seja, os mesmos vetores de entradas utilizados para testar a cobertura de falhas para falhas do tipo *stuck-at* também detectarão falhas do tipo *stuck-on*. A Tabela 2 apresenta a tabela verdade para o circuito sem falha e para o circuito com uma falha em cada transistor onde Z representa as condições em que a saída será um divisor resistivo causado pela falha. De acordo com a Equação 1.2, a cobertura de falhas do tipo stuck-on para portas NOR, é de 100% quando excitado por todos os possíveis vetores de entrada.

$$V_{TESTE} = 2^{(E)} \tag{1.2}$$

Tabela 2 – Saídas esperadas para a porta NOR com falhas e sem falhas

Vetores de Entrada	Saída Esperada	Saída sob falhas			
		MP1	MP2	MN1	MN2
00	1	1	1	Z	Z
01	0	Z	0	0	0
10	0	0	Z	0	0
11	0	0	0	0	0

1.2.2 Falhas Stuck-Open

Falhas *stuck-open* (WADSACK, 1978) são falhas que fazem com que um transistor permaneça no estado de alta impedância independente do sinal inserido no *gate*. Essa falha em muitos casos faz com que o circuito armazene o estado anterior, mas isso nem sempre é verdade devido à ação das correntes de fuga presentes nas tecnologias nanométricas. Um valor armazenado devido à ação das capacitâncias internas pode ser descarregado mais rapidamente devido a ação dessas correntes de fuga, e isso faz com que o valor de tensão de saída seja um valor intermediário que pode ou não ser interpretado como uma falha pelos circuitos subsequentes. Quando uma falha desse tipo ocorre, o circuito se comportará como um circuito sequencial mantendo o estado lógico anterior. Por exemplo: na Figura 6 (c), quando a condição inicial do circuito for A = B = 0, a saída S terá o valor lógico 1; quando o circuito fizer uma transição para A = 1 e B = 0, a saída S não fará a transição para 0, pois uma falha no transistor Mn1 está impedindo a rede *pull-down* de conduzir. Assim, o valor lógico 0 será mantido na saída por um certo período de tempo devido às capacitâncias internas do circuito.

Como as falhas *stuck-open* possuem essa propriedade de manter o estado anterior, é necessário uma sequência de dois vetores de teste para detectar esse tipo de falha. A quantidade de pares de vetores de entrada necessários para testar as falhas *stuck-open* é calculada pela Equação 1.3, onde $n = 2^E$, E é o número de entradas e $p = 2$ para pares de vetores. No caso da porta NOR que possui 2 entradas, são necessários 12 pares de vetores para testar todas as possíveis falhas. A Tabela 3 apresenta os resultados para o circuito sem falhas e para o circuito com falhas. De acordo com a Equação 1.2, a cobertura de falhas do tipo stuck-open para portas NOR, é de 100% quando excitado por todos os possíveis vetores de entrada.

$$V_{Entrada}(n,p) = \frac{n!}{(n-p)!} \tag{1.3}$$

1.3 Falhas Transientes

A redução nas dimensões dos transistores associado à redução das tensões de operação, vem tornando os circuitos digitais cada vez mais sensíveis a falhas induzidas

Tabela 3 – Saídas esperadas para a porta NOR com falhas e sem falhas

Vetores de Entrada	Saída Esperada	Saída sob Falhas			
		MP1	MP2	MN1	MN2
00→01	0	0	0	0	1
00→10	0	0	0	1	0
00→11	0	0	0	0	0
01→00	1	0	0	1	1
01→10	0	0	0	0	0
01→11	0	0	0	0	0
10→00	1	0	0	1	1
10→01	0	0	0	0	0
10→11	0	0	0	0	0
11→00	1	0	0	1	1
11→01	0	0	0	0	0
11→10	0	0	0	0	0

por radiação (BAUMANN, 2005b). Quando uma partícula energética atinge o material semicondutor de um transistor, causa um efeito chamado de *single-event effect*(SEE).

Quando um SEE acontece em uma célula de memória, registrador, latch, ou flip-flop, o valor lógico armazenado nesses elementos pode ser alterado, ocasionando um *bit-flip*. O elemento não é danificado pelo SEE, pois a informação pode ser reescrita corrigindo o valor armazenado. Quando um SEE ocorre nesses elementos é chamado de *Single-Event Upset* (SEU). Quando um SEE acontece na lógica combinacional é chamado de *Single-Event Transient* (SET). Essa interação entre uma partícula de alta energia e um elemento do circuito combinacional gera um pulso de tensão/corrente que, se tiver força suficiente para se propagar pela lógica combinacional, pode ser capturado por um elemento sequencial e armazenado em um elemento de memória. Como esse trabalho irá avaliar o efeito de falhas transientes em votadores majoritários, que são lógica combinacional, as falhas do tipo SET serão abordadas em mais detalhes.

1.3.1 Mecanismo de Deposição de Cargas

O impacto de um íon em uma região sensível do circuito, causa um distúrbio que depende da transferência linear de energia (LET) e geralmente é expresso em unidades de Mega elétron-volts centímetro ao quadrado por miligrama ($MeV - cm^2/mg$). Partículas com maior massa e maior energia em um material mais denso causarão um LET com maior intensidade ao atingir a junção p-n de um material semicondutor. A interação entre a partícula energética pode acontecer através de dois mecanismos:

- Ionização direta: quando uma partícula energética cria pares de eletron-lacuna à medida que viaja pelo material semicondutor e perde energia. O impacto da partícula cria um caminho de pares de eletron-lacuna em formato de funil (*funnel*) como pode ser visto na Figura 7 (a). A carga é rapidamente coletada pelo material semicondutor pelo processo de deriva (*drift*) como observado na Figura 7 (b), gerando um pulso de corrente/tensão. Em seguida a carga adicional é coletada à medida que os elétrons se

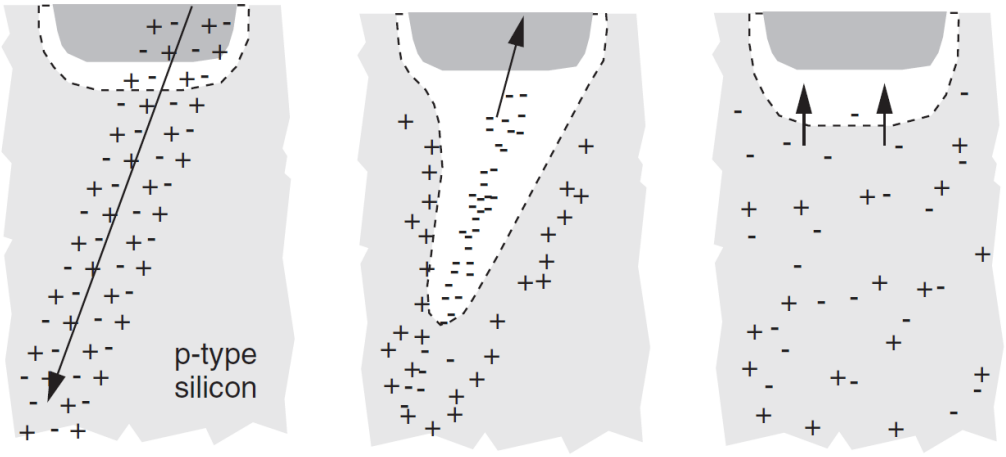


Figura 7 – Fases do efeito de geração e coleta de cargas em uma junção p-n (BAUMANN, 2005b)

difundem para a região de depleção em um processo chamado de difusão (*diffusion*), como pode ser visto na Figura 7 (c).

- Ionização indireta: geralmente causada por prótons ou nêutrons, essa interação não gera ionização direta no material semicondutor. Quando um nêutron atinge o núcleo do silício, por exemplo, pode quebrá-lo em íons mais leves como nêutrons, prótons, e/ou partículas alfa. Quanto maior for a energia do nêutron ou próton, maior será a energia cinética com que esses íons serão lançados para direções opostas. Essas reações podem ter energia suficiente para causar uma falha transiente e aumentar a taxa de falhas (BAUMANN, 2005b).

1.3.2 *Single Event Transient* (SET)

SET é um pulso de tensão/corrente produzido por uma partícula energética ao atingir o material semicondutor em um circuito combinacional. Com as dimensões dos transistores e as tensões de operação cada vez menores, a probabilidade do pulso de tensão/corrente ter amplitude suficiente para se propagar é maior. Além disso, em decorrência das elevadas frequências de operação, a chance de um SET ser propagado até circuitos sequenciais é grande. Sendo uma grande preocupação para a tecnologia atual, são necessários métodos de simulação para melhor compreender o mecanismo desse tipo de falha.

A forma característica de um pulso de corrente gerado por ionização direta pode ser visto na Figura 8 (MESSENGER, 1982). No nível de circuito, o mecanismo de deposição de cargas pode ser modelado por um pulso de corrente de exponencial dupla de acordo com a Equação 1.4 (MESSENGER, 1982), onde Q_{coll} é a carga coletada pela junção p-n durante o impacto de uma partícula energética, τ_α é a constante de tempo de coleta da junção e τ_β é a constante de tempo para estabelecer inicialmente a faixa de íons.

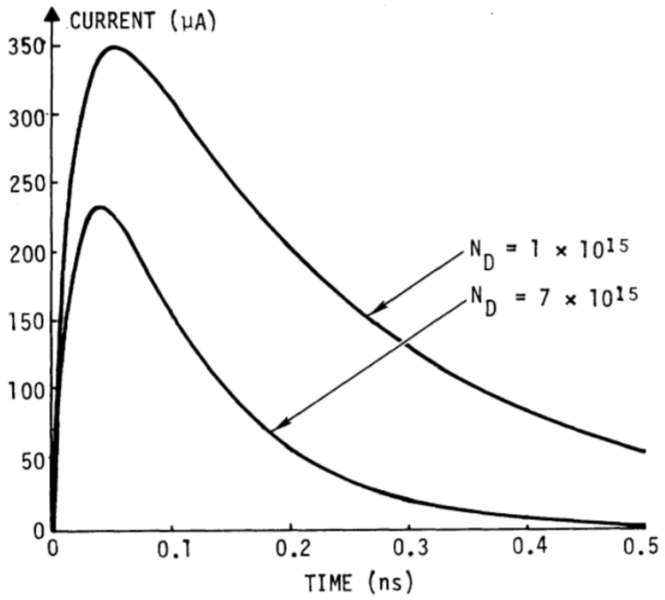


Figura 8 – Resposta transiente de um íon de alta energia (MESSENGER, 1982)

Simulações em nível elétrico são feitas através de simuladores elétricos. O pulso de corrente é inserido em cada nó do circuito, simulando o impacto de uma partícula em cada dreno dos transistores. A vantagem desse método é a quantidade reduzida de recursos necessários para obter os resultados. Porém, a precisão das simulações está diretamente associada às contantes utilizadas na Equação 1.4.

$$I(t) = \frac{Q_{coll}}{\tau\alpha - \tau\beta}\left(e^{-\left(\frac{t}{\tau\alpha}\right)} - e^{-\left(\frac{t}{\tau\beta}\right)}\right) \tag{1.4}$$

Quando uma partícula atinge a junção p-n do dreno de um transistor, a carga coletada Q_{coll} deve exceder a carga crítica Q_{crit} para o transistor começar a conduzir. A Equação 3.4 é utilizada para calcular Q_{coll}, onde L possui um valor típico de $2\mu m$ para cada $1MeV - cm^2/mg$. Um pulso de corrente aparece na saída do transistor quando Q_{coll} exceder Q_{crit}. A propagação desse pulso de corrente para a saída de um circuito combinacional depende de vários fatores como a existência de um caminho até a saída, atenuação do sinal ao passar por vários estágios de portas lógicas e a chance do pulso ser capturado por um elemento sequencial.

$$Q_{coll} = 10,8 \times L \times LET \tag{1.5}$$

1.4 Técnicas de Tolerância a Falhas Baseadas em Redundância

Muitas técnicas de tolerância a falhas utilizam algum tipo de redundância. Isso significa ter mais recursos, além do mínimo necessário, para executar uma função com o intuito de mascarar falhas e manter um nível de funcionalidade adequado do sistema. Segundo Koren e Krishna (2010), existem três tipos de redundância: redundância de hardware, redundância de informação e redundância de tempo.

A redundância de informação consiste em adicionar bits extras (chamados de *check bits*) ao dado original, permitindo a detecção e/ou correção do erro. Os melhores exemplos desse tipo de redundância são os códigos de detecção e correção de erros, que são largamente utilizados para proteger dados em unidades de memória, caminhos de dados e dispositivos de armazenamento. Como outras técnicas, a redundância de informação também utiliza hardware adicional para armazenar e processar a informação redundante (KOREN; KRISHNA, 2010).

A redundância de informação também é utilizada em comunicação de dados. Por exemplo, em redes locais ou internet, os canais de comunicação estão sujeitos a falhas transientes. Então a redundância de informação é utilizada para verificar a veracidade das informações. Os dados podem ser reenviados na ocorrência de um erro, empregando redundância temporal.

A técnicas baseadas em redundância de tempo fazem um reprocessamento dos dados, então elas tem capacidade de tolerar apenas falhas transientes (KOREN; KRISHNA, 2010). A Figura 9 mostra o princípio de funcionamento de uma técnica baseada em redundância de tempo. Nesse tipo de lógica, a saída do circuito combinacional é duplicada no domínio do tempo, ou seja, temos a mesma saída em dois instantes de tempo diferentes.

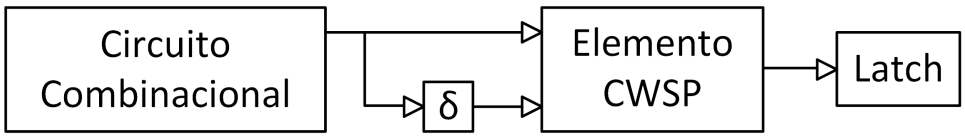


Figura 9 – Princípios da redundância de tempo

Uma das entradas do elemento *Code Word State Preserving* (CWSP) recebe o sinal diretamente da saída da lógica combinacional, enquanto a outra entrada recebe o mesmo sinal com um atraso δ (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000). O elemento CWSP recebe esse nome pois ele preserva o estado anterior, se as duas entradas não forem iguais.

Basicamente essa lógica compara a mesma entrada em dois instantes de tempo diferentes. Se as duas entradas forem iguais, a entrada será copiada para a saída e será o próximo estado do circuito combinacional. Se as duas entradas forem diferentes, o estado do circuito combinacional não muda. Comparado a outras formas de redundância,

a redundância temporal utiliza muito menos hardware, porém essa técnica sofre com altas perdas de desempenho.

A redundância de hardware consiste em ter hardware adicional disponível para detectar ou sobrepor o componente que falhar. Há três tipos de redundância de hardware: a redundância de hardware estática, redundância de hardware dinâmica e a combinação das duas formas é chamada de redundância de hardware híbrida.

A redundância de hardware estática consiste em ter mais de um módulo igual executando a mesma função. No caso de uma arquitetura com três módulos, se um dos módulos falhar, a saída será decidida por um módulo votador que irá escolher a saída pela maioria. Dessa forma, se dois módulos tiverem a saída correta ela irá sobrepor a saída incorreta do terceiro módulo. Uma observação importante é que esse tipo de redundância de hardware não detecta falhas, apenas as mascara. A redundância de hardware dinâmica utiliza o conceito de detecção de falhas. Caso uma falha seja detectada, outros componentes entram em ação para substituir o componente com falha. Essa forma de redundância de hardware visa reduzir o consumo de energia. A redundância híbrida, que não passa da combinação das duas formas, utiliza módulos estáticos em partes do circuito e módulos dinâmicos em outras.

A redundância de hardware pode ir de uma simples duplicação a outras estruturas complexas com unidades ativas. Todas as formas de redundância de hardware tem um alto custo em área de *chip* e consumo de energia, sendo geralmente justificável sua aplicação apenas em sistemas críticos.

1.4.1 *Triple Modular Redundancy*

A técnica mais clássica de redundância de hardware é chamada de NMR (*N-Modular Redundancy*). Uma estrutura NMR tem N módulos executando a mesma função e a sua saída é escolhida por um votador de maioria. Nessa técnica, o número de falhas que podem ser toleradas é igual a $(N-1)/2$. Se $N = 3$, então é chamado de TMR (*Triple Modular Redundancy*). A Figura 10 mostra o princípio de funcionamento dessa arquitetura. São três módulos, M1, M2 e M3 executando a mesma função e a sua saída é escolhida por um votador majoritário. Essa técnica é largamente utilizada em aplicações práticas (KOREN; KRISHNA, 2010). O TMR tem a capacidade de tolerar qualquer tipo de falhas em um módulo, graças a presença dos outros dois módulos livres de falhas. O votador é o ponto fraco dessa arquitetura e pode ser implementado em *software* ou com técnicas mais robustas de projeto (CAZEAUX; ROSSI; METRA, 2004).

Segundo Vial et al. (2009), um arquitetura TMR tolera uma falha única, mas pode tolerar falhas múltiplas em determinados casos. Se ocorrerem duas falhas no mesmo módulo por exemplo, elas serão mascaradas. Mas se essas falhas ocorrerem em módulos diferentes,

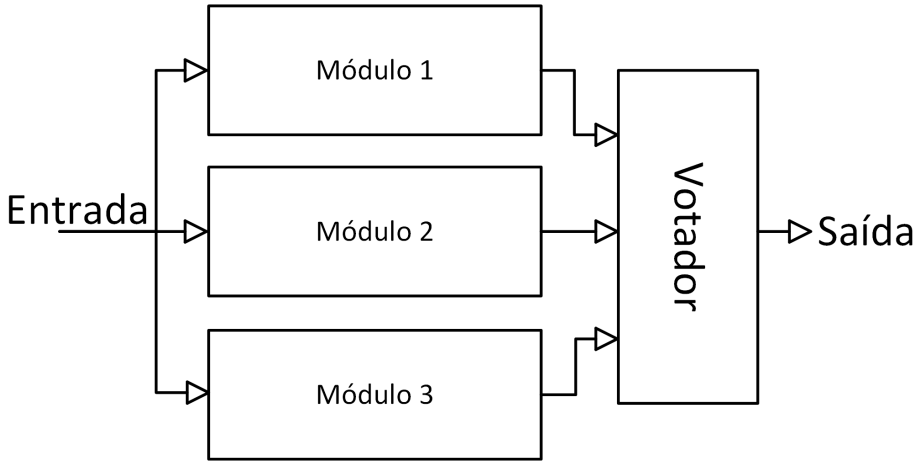


Figura 10 – Princípios do TMR (VIAL et al., 2009)

consequentemente a saída será errada. Seguindo esse pensamento, uma estrutura TMR básica pode tolerar 33,33% de falhas duplas considerando que o votador é livre de falhas (VIAL et al., 2009).

Para aumentar a robustez da arquitetura TMR o circuito pode ser particionado em dois ou três blocos equivalentes. No caso da Figura 11, a arquitetura foi particionada em três blocos independentes, sendo que há votadores em cada nível de blocos. Sendo assim, uma falha no primeiro bloco não interfere no funcionamento do segundo bloco e uma falha no secundo bloco não interfere no funcionamento do terceiro bloco. A quantidade de falhas duplas que podem ser toleradas pelo TMR particionado é de 77,77%, considerando que o votador é livre de falhas (VIAL et al., 2009).

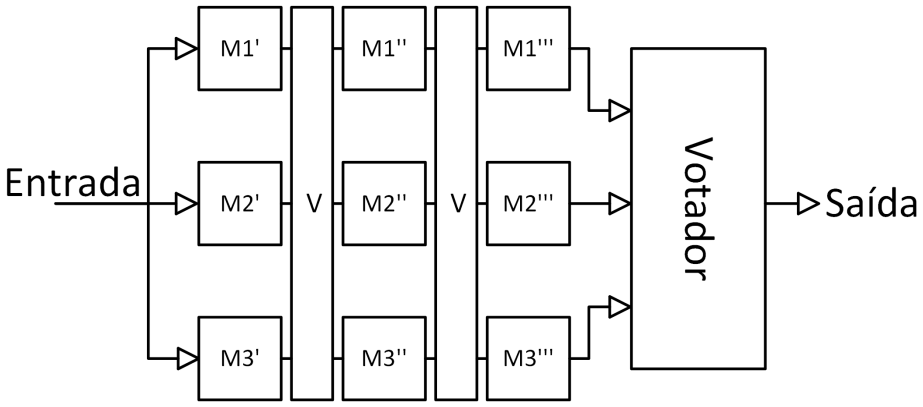


Figura 11 – TMR particionado em três blocos (VIAL et al., 2009)

Com essa técnica, a tolerância a falhas aumenta de acordo com o número de particionamentos. A porcentagem de falhas toleradas pode ser calculada de acordo com a Equação 1.6, onde k é o número de blocos (VIAL et al., 2009).

$$100 \times \frac{3k-2}{3k} \tag{1.6}$$

Ao particionar a arquitetura TMR em módulos, diminui-se a taxa de falhas, porém existem algumas consequências. Uma delas é número de votadores adicionais que aumentam a área de *chip*, o que pode ser um grande aumento de área em circuitos pequenos. Circuitos grandes tem um aumento de área relativamente pequeno. Outro ponto é a complexidade do projeto que aumenta com o número de blocos utilizados. Existem ferramentas responsáveis por fazer os cortes no circuito nos pontos onde serão inseridos os votadores e os seus algoritmos podem levar bastante tempo para particionar circuitos grandes ou com muitas partições, aumentando o tempo de projeto.

1.4.2 DTMR

O TMR diversificado (DTMR), foi inspirado em uma técnica de redundância de *software* chamada de *N-Version Programming* (ELMENDORF, 1972). Mais tarde a técnica foi estendida para ser utilizada em hardware por Avizienis e Kelly (1984). O princípio básico dessa técnica é que tenha três módulos executando a mesma função e a saída é escolhida por um votador de maioria. A diferença para o TMR é que o DTMR utiliza três módulos com arquiteturas diferentes.

Tambara et al. (2013) fez uma implementação do DTMR para avaliar sua eficiência em tolerar falhas transientes, o esquemático da implementação pode ser visto na Figura 12. A ideia do experimento é fazer a multiplicação de duas matrizes 8x8 em um sistema TMR, sendo que cada módulo possui uma arquitetura diferente. O primeiro módulo é uma *pipeline* de máquina de estados finita (FSM), o segundo módulo é construído por lógica combinacional e o terceiro módulo é implementado em *software* e executado por um processador miniMIPS. Os votadores de maioria são assíncronos e implementados em hardware.

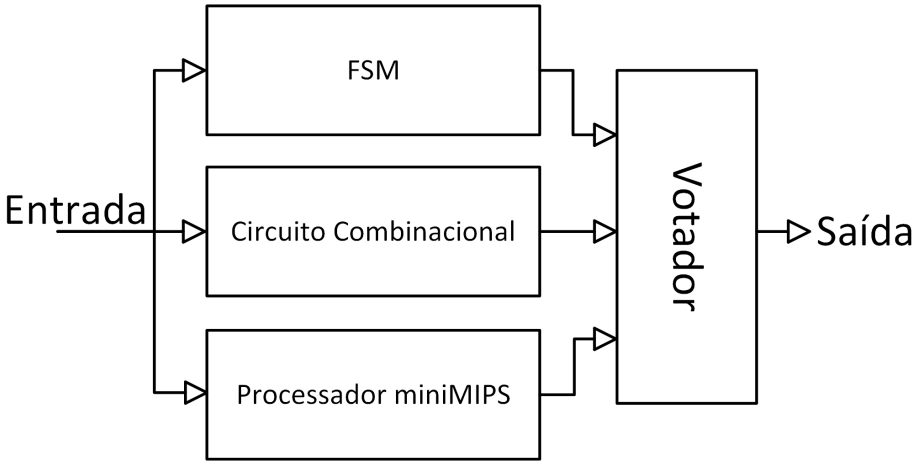


Figura 12 – Esquemático do DTMR implementado (TAMBARA et al., 2013)

Pela metodologia adotada no trabalho, o DTMR se mostrou robusto. Em 2000 injeções de falhas, apenas 0,25% provocaram erros, sendo 60% causado por falhas que

atingiram mais de um módulo ao mesmo tempo e 40% por erros nos votadores majoritários. O tamanho do circuito depende das arquiteturas utilizadas, mas pode ser comparado ao tamanho de um sistema TMR. Assim como no TMR, o DTMR tem capacidade de tolerar falhas únicas em um dos módulos e uma variedade de falhas múltiplas, caso elas ocorram no mesmo módulo.

1.4.3 ATMR

O TMR tem um aumento de área de aproximadamente 200% ou mais do circuito original, consequentemente tendo um grande impacto no consumo de energia. Com o TMR Aproximado (ATMR, do inglês *Aproximate Triple Modular Redundancy*), os módulos se utilizam de circuitos aproximados para reduzir a área de *chip* e consequentemente o consumo de energia. No entanto, para se beneficiar do ATMR é necessário explorar o *trade-off* entre área de *chip* e taxa de falhas (GOMES; KASTENSMIDT, 2013).

A técnica de TMR Aproximado foi proposta inicialmente por Sierawski, Bhuva e Massengill (2006). O uso da lógica simplificada reduz o consumo de energia, ao custo de reduzir a robustez do circuito. A Figura 13 mostra um esquemático do ATMR, onde G é o módulo original e F e H são módulos aproximados.

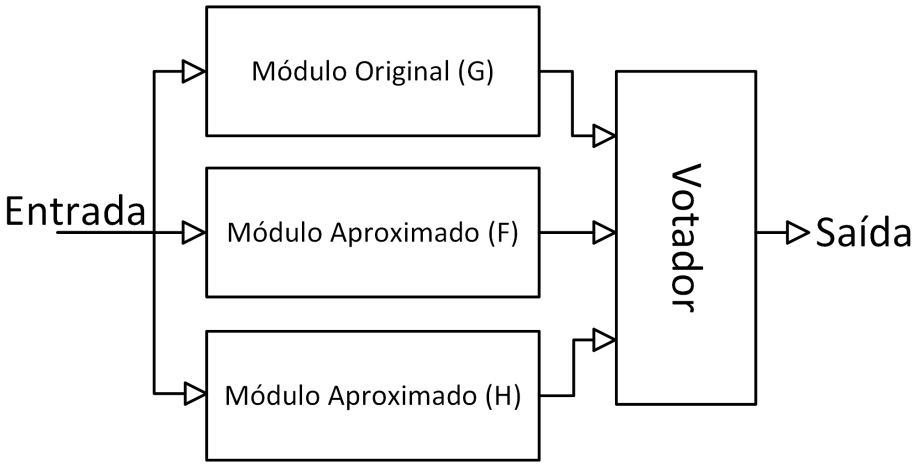


Figura 13 – Esquemático do ATMR (GOMES; KASTENSMIDT, 2013)

Uma característica importante do ATMR é que mesmo na ausência de falhas um dos módulos pode divergir da resposta, então o votador irá garantir a saída correta. Mas, para isso, é necessário seguir uma regra: $F \subseteq G \subseteq H$ (GOMES, 2014), ou seja, F deve ser um mintermo de G, e todo mintermo de G deve ser um mintermo de H. Isso irá garantir que, pelo menos, dois módulos sempre terão a mesma resposta.

No ATMR, a estrutura lógica dos circuitos afeta diretamente a taxa de falhas. Com boas estruturas, o circuito pode ficar acima de 95% de proteção contra falhas únicas, com uma variação de aumento de área entre 83% até 183% (GOMES, 2014). Como dito

anteriormente, é necessário explorar o *trade-off* entre aumento de área e taxa de falhas que melhor se adapta à aplicação.

2 Votadores Majoritários

Sistemas que adotam estratégias de tolerância a falhas baseadas no TMR possuem três módulos idênticos processando a mesma informação, sendo que a saída é escolhida por um votador majoritário. O votador majoritário executa a função lógica representada na Equação 2.1.

$$S = AB + AC + BC \tag{2.1}$$

O votador é o ponto vulnerável do TMR, por isso a confiabilidade depende fortemente da robustez do votador. Se o votador majoritário produzir uma saída incorreta, terá sido um desperdício de recursos utilizar o TMR. Um votador majoritário tem a capacidade de tolerar uma falha do tipo *stuck-at* em qualquer uma de suas entradas. Parte dos estudos com o TMR considera que o votador majoritário é livre de falhas, mas na realidade podem ocorrer falhas nos transistores ou nos nodos internos dessa porta lógica. O objetivo do presente trabalho é avaliar diferentes implementações de votadores quanto à sua robustez. Os seguintes votadores majoritários serão abordados nesse trabalho: Clássico (CAZEAUX; ROSSI; METRA, 2004), NAND, NOR, Kshirsagar (KSHIRSAGAR; PATRIKAR, 2009), BAN (BAN; De Barros Naviner, 2010), MUX (DANILOV; GORBUNOV; ANTONOV, 2013) e TR (EL-RAZOUK; ABID, 2007).

2.1 Implementação Clássica de um Votador Majoritário

O votador majoritário da Figura 14 é uma implementação clássica da função lógica descrita na equação 2.1 na família lógica Static CMOS. Essa implementação é uma porta lógica de um único estágio. A vantagem dessa implementação é utilizar poucos transistores, apenas 14 transistores, mas segundo trabalhos encontrados na literatura essa estrutura apresenta pouca robustez a falhas. Esse tipo de implementação geralmente é utilizado como comparação com outras implementações e abordagens (CAZEAUX; ROSSI; METRA, 2004; DANILOV; GORBUNOV; ANTONOV, 2013; METRA; FAVALLI; RICCO, 1997).

2.2 Votador Majoritário NAND-NAND

O votador majoritário da Figura 15 é uma implementação do votador majoritário utilizando portas NAND em dois níveis. Normalmente por ser uma porta lógica maior que o votador clássico, não tem sua robustez investigada por outros trabalhos. Esse votador possui 18 transistores.

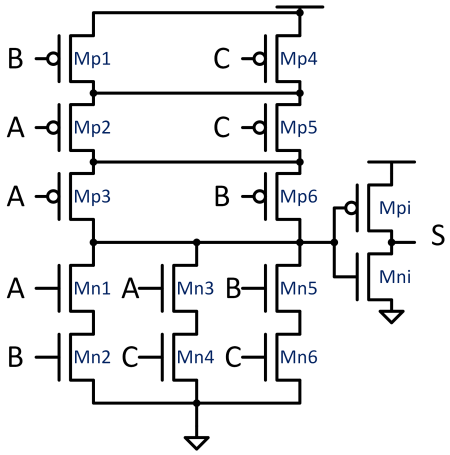


Figura 14 – Implementação clássica de um votador

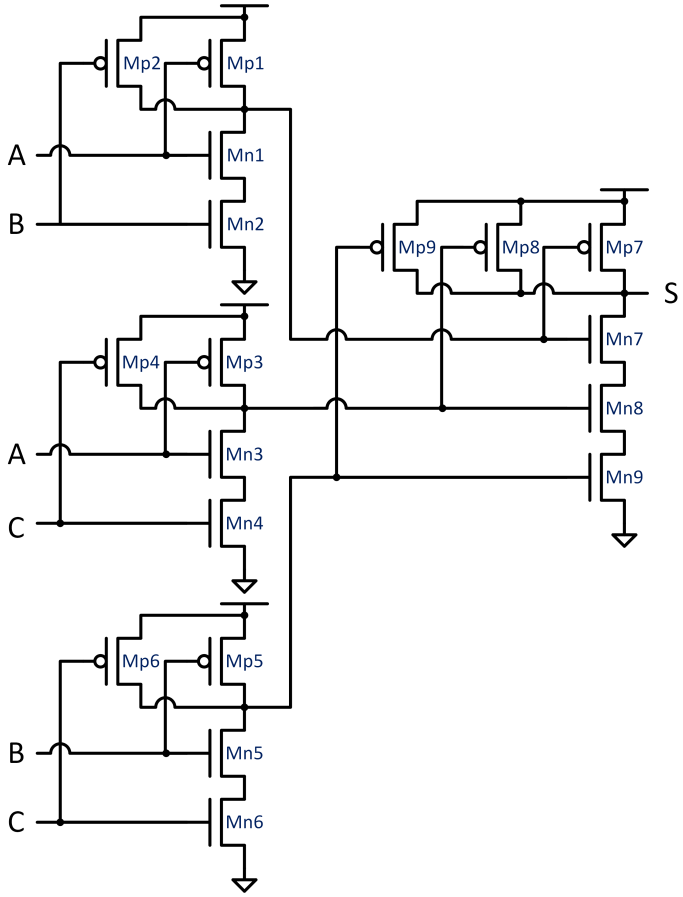


Figura 15 – Implementação clássica de um votador composto por portas NAND

2.3 Votador Majoritário NOR-NOR

O votador majoritário da Figura 16 é uma implementação utilizando portas NOR em dois níveis. Sendo o mesmo caso do votador majoritário composto por postas NAND, geralmente não tem sua robustez investigada por outros trabalhos. Essa implementação possui 18 transistores.

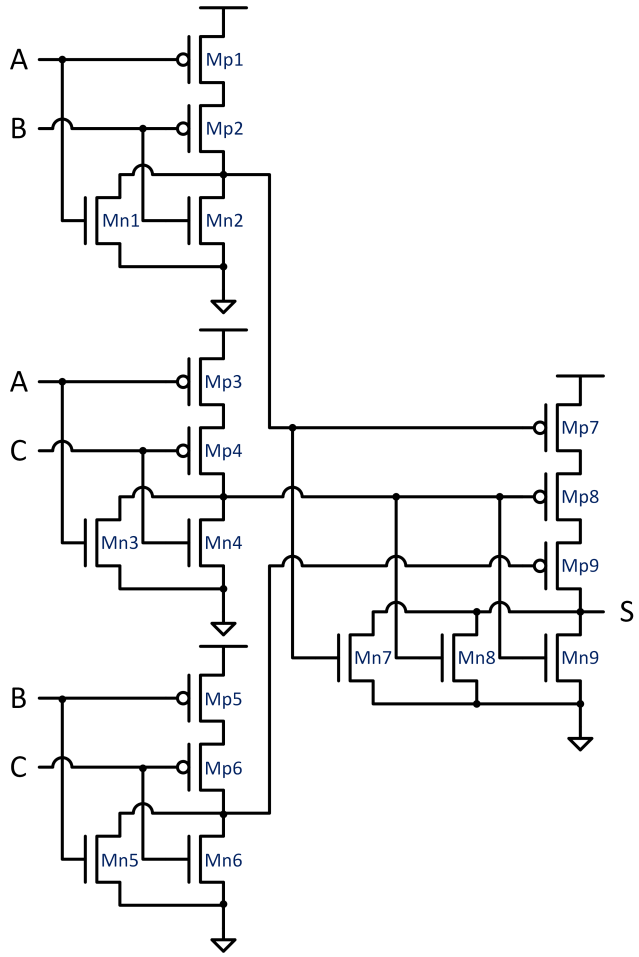


Figura 16 – Implementação clássica de um votador composto por portas NOR

2.4 Votador Majoritário Kshirsagar e Patrikar (2009)

A Figura 17 mostra a implementação de um votador majoritário proposto por Kshirsagar e Patrikar (2009). Essa implementação utiliza duas portas XOR, um codificador de prioridade e um multiplexador. Segundo o autor, essa implementação tem confiabilidade seis vezes maior que a implementação clássica de um votador majoritário em relação às falhas *stuck-at*, quando aplicadas em seus nós internos e que essa implementação somente irá falhar se ocorrer uma falha no multiplexador. Esse circuito foi implementado com 30 transistores.

2.5 Votador Majoritário Ban e De Barros Naviner (2010)

O votador proposto por Ban e De Barros Naviner (2010), mostrado na Figura 18, foi inspirado no votador de Kshirsagar e Patrikar (2009). Ban e De Barros Naviner (2010) afirma que circuitos menores possuem uma menor probabilidade de serem atingidos por uma falha. Uma vantagem do circuito é o consumo de potência reduzido e menor área de chip, pois foi implementado com 14 transistores, a mesma quantidade de transistores do

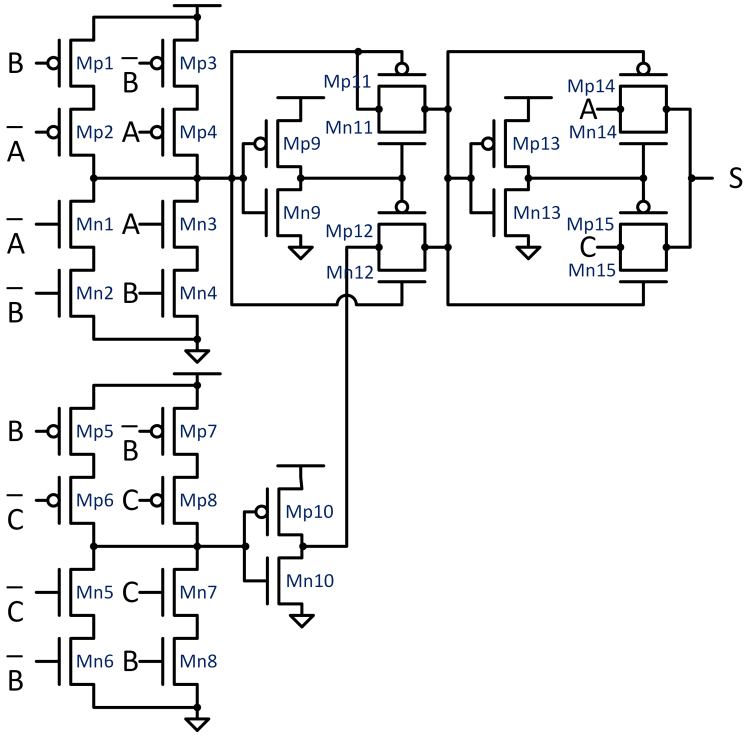


Figura 17 – Implementação de um votador proposto por Kshirsagar e Patrikar (2009)

votador clássico.

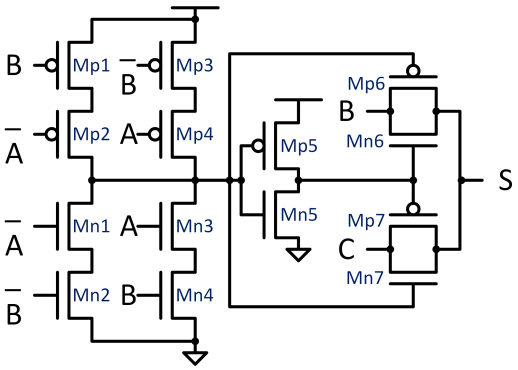


Figura 18 – Implementação de um votador proposto por Ban e De Barros Naviner (2010)

2.6 Votador Majoritário com MUX

A Figura 19 mostra uma implementação de um vontador majoritário que também utiliza um multiplexador para fazer a seleção do sinal de saída. Essa implementação foi descrita por Danilov, Gorbunov e Antonov (2013). É uma implementação muito parecida com a de Ban e De Barros Naviner (2010), com a diferença de que o multiplexador é construído por portas NAND e NOT. Esse circuito possui mais transistores que a implementação de Ban e De Barros Naviner (2010), sendo implementado com 22 transistores.

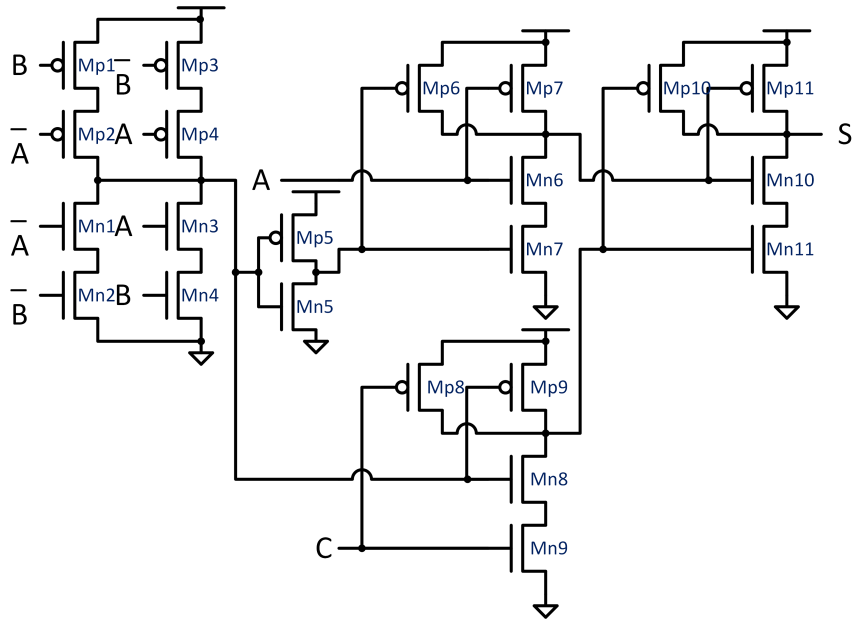


Figura 19 – Implementação de um votador com um MUX composto por portas NAND

2.7 Votador Majoritário com Redundância de Transistor

O votador majoritário da Figura 20 foi proposto por El-Razouk e Abid (2007). Essa técnica utiliza redundância a nível de transistor para aumentar a robustez do circuito a falhas permanentes do tipo *stuck-on* e *stuck-open*. A técnica utiliza redundância de transistores, ou seja, 2 transistores no lugar de 1, aplicada da seguinte forma: 2 transistores em série na rede *pull-up* e dois transistores em paralelo na rede *pull-down*. A técnica foi implementada em um votador NAND com 36 transistores.

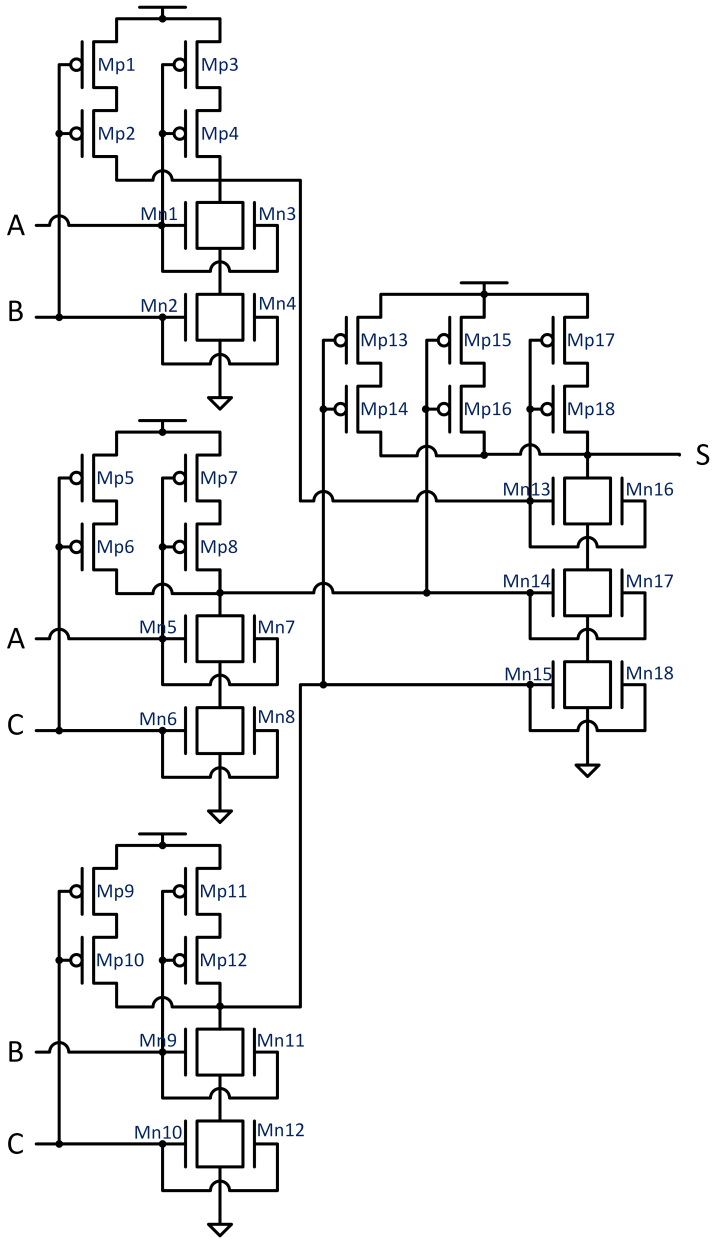


Figura 20 – Implementação de um votador proposto por El-Razouk e Abid (2007)

3 Materiais e Métodos

Neste Capítulo serão apresentadas as etapas da metodologia adotada para a execução deste trabalho. Cada uma das seções pode ser considerada uma das etapas do mesmo. O projeto será dividido em quatro etapas: Levantamento bibliográfico, descrição e validação lógica dos votadores; avaliação de desempenho e dissipação de potência; avaliação de falhas permanentes; avaliação de falhas transientes.

3.1 Levantamento, Descrição, Validação dos Votadores Majoritários

Através de revisão bibliográfica, foi feito o levantamento de diversas implementações diferentes de votadores majoritários. Os votadores majoritários que serão avaliados nesse trabalho serão chamados com o nome dos seus autores ou de acordo com suas implementações:

- Classico (CAZEAUX; ROSSI; METRA, 2005);
- NAND (CAZEAUX; ROSSI; METRA, 2005);
- NOR (CAZEAUX; ROSSI; METRA, 2005);
- BAN (BAN; De Barros Naviner, 2010);
- KSHIRSAGAR (KSHIRSAGAR; PATRIKAR, 2009);
- MUX (DANILOV; GORBUNOV; ANTONOV, 2013);
- TR (EL-RAZOUK; ABID, 2007).

Como discutido anteriormente, falhas *stuck-at* que são simuladas em nível lógico do circuito não refletem corretamente o comportamento de falhas *stuck-on* e *stuck-open*. As falhas *stuck-on* produzem um divisor resistivo que pode ou não ser interpretado como uma falha pelas portas lógicas subsequentes. As falhas *stuck-open* fazem com que o circuito mantenha o estado lógico anterior em certas condições, porém, isso é diretamente influenciado pelas correntes de fuga que influenciam diferentes tenologias. Para garantir maior precisão dos resultados, a descrição dos circuitos votadores e as simulações de falhas permanentes será feita em nível elétrico. Para manter o padrão no trabalho, as simulações de falhas transientes também serão feitas em nível elétrico. Os votadores foram descritos em nível elétrico em linguagem SPICE e serão feitas simulações através do

software ngspice (NENZI; VOGT, 2014) utilizando a tecnologia preditiva de 32 nm com modelos de transistores de *High Performance* (HP) (PTM, 2014).

Os transistores de cada votador foram dimensionados pelo método de esforço lógico (*Logical Effort*) (SUTHERLAND; SPROULL; HARRIS, 1999). A menor largura de canal dos transistores (λ) foi definido como sendo $64nm$, dessa forma o dimensionamento foi considerado $W = 1 \times \lambda$ para transistores do tipo NMOS e $W = 2 \times \lambda$ para transistores PMOS. Quando os transistores faziam parte de uma associação série, o seu valor de W era multiplicado pela quantidade de transistores na associação.

Após os circuitos serem descritos, o simulador elétrico NGSPICE (NENZI; VOGT, 2014) é utilizado para comprovar (validar) que o circuito se comporta como um votador. Os sinais de entrada dos votadores são configurados para representar toda a tabela verdade e a saída obtida é comparada com a saída da tabela verdade.

3.2 Avaliação do Desempenho e Consumo de Potência

O simulador elétrico utilizado para obter os dados de atraso e potência é o NGSPICE (NENZI; VOGT, 2014). A Figura 21 representa o circuito que será utilizado para fazer a avaliação dos votadores quanto ao desempenho e consumo de potência. De acordo com Hernandez e Aranda (2011), os inversores da entrada tem a função de simular alguma degradação nos sinais de entrada e os inversores na saída simulam a capacitância de carga de quatro inversores. A vantagem dessa abordagem é que esses dispositivos influenciam na análise dinâmica do circuito sob teste (CST) simulando uma situação real. Outra vantagem dessa abordagem, é que como alguns dos circuitos votadores possuem transistores de passagem, fazendo com que a corrente de carga seja drenada dos inversores ligados às entradas e saídas do circuito. Nestes casos a energia necessária para carregar ou descarregar os nós internos do circuito não vem exclusivamente da alimentação do CST. Dessa forma, para medir o consumo dinâmico dos circuitos votadores, os inversores de entrada serão considerados no consumo de potência. Como o consumo intrínseco dos inversores tende a ser sempre o mesmo, este foi o modo encontrado para fazer uma comparação mais justa entre os circuitos avaliados.

O consumo dinâmico e a avaliação de desempenho dos votadores será obtido pela análise dos arcos de atraso. A sequência de testes para analisar os arcos de atraso está apresentada na Tabela 4. Cada arco de atraso depende de combinações diferentes dos valores de entrada do circuito e espera-se que apresentem atrasos diferentes. Para a comparação dos atrasos entre todos os votadores majoritários será calculado o atraso médio e o desvio padrão dos atrasos para decidir qual apresenta o menor atraso com pouca variação entre os arcos de atrasos.

Através da ferramenta NGSPICE é possível obter o consumo dinâmico dos circuitos

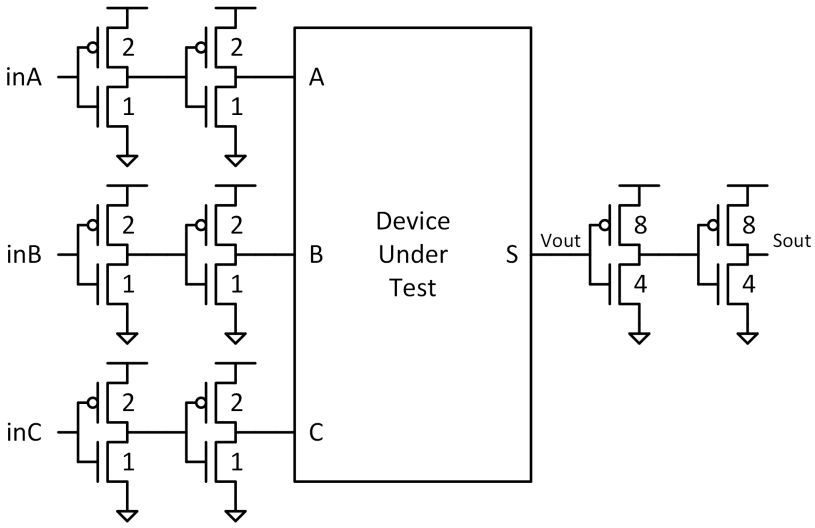


Figura 21 – Circuito utilizado para avaliação de desempenho e consumo de potência dos votadores majoritários

Tabela 4 – Sequência de testes para os arcos de atraso

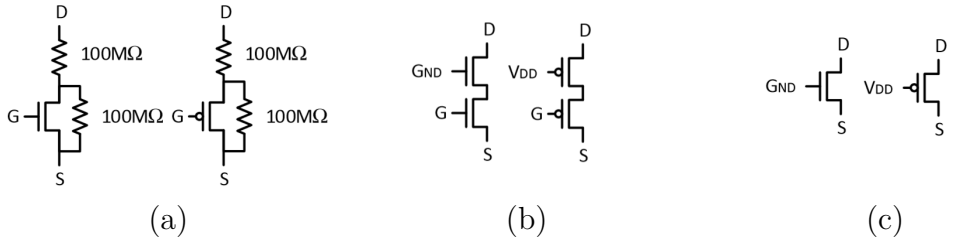
A	0	1	0	0	0	0	1	0	0	0	1	1	1	1	1
B	0	0	0	1	0	1	1	1	1	1	0	1	0	0	0
C	1	1	1	1	1	0	0	0	1	0	0	0	0	1	0
S	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0

que é fornecido através de um valor de energia. A potência dinâmica do circuito pode ser calculada através da Equação 3.1, onde E é a energia consumida em Joules (J) e T é o tempo de simulação em segundos (s). A Potência dinâmica é expressada em Watts (W), sendo que $1W = 1J/s$. Como a frequência de operação influencia na potência dinâmica, todas as simulações serão feitas em uma frequência de transição dos sinais de entrada de $1GHz$. Para comparação entre os votadores majoritários será calculado o PDP de cada circuito, que é o produto da potência pelo atraso (do inglês, *power-delay product*). Esse valor possui uma informação adicional que auxiliará a obter o circuito com melhor comprometimento entre o desempenho e a potência. Serão calculados dois valores de PDP, sendo o PDP médio calculado pelo produto entre a média dos atrasos e a potência total, e o PDP máximo, calculado pelo produto do maior valor dos atrasos pela potência total.

$$P_{dynamic} = \frac{E}{T} \tag{3.1}$$

3.3 Avaliação de Falhas Permanentes

Como abordado anteriormente, foi escolhido fazer a injeção de falhas em nível elétrico para se conseguir uma maior precisão nos resultados. Existem diversas formas de simular falhas *stuck-on* e *stuck-open* em nível elétrico (METRA; FAVALLI; RICCO, 1997;



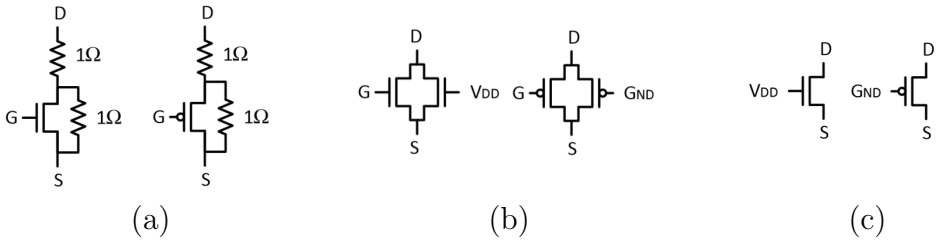
(a) (b) (c)

Figura 22 – Representação de falhas do tipo *stuck-open*

GOMEZ et al., 2009; SCHUSTER; BRYANT, 1983; JIANG et al., 2009).

Uma das metodologias para simular falhas do tipo *stuck-open* baseia-se em remover o transistor do circuito, deixando um circuito aberto. Outra forma de simular esse tipo de falhas é introduzir resistências em série e em paralelo com o transistor, devendo ter valor suficiente para impedir que uma tensão significativa seja transferida para a saída do circuito. Essa abordagem foi utilizada por Jiang et al. (2009) utilizando resistências com o valor de 100 MΩ e sua representação pode ser vista na Figura 22 (a). Na abordagem de Schuster e Bryant (1983) são utilizados transistores adicionais em série com o transistor onde se quer inserir a falha, com as mesmas especificações. O transistor adicional será forçado a permanecer aberto através de um sinal fixo no terminal de controle do *gate*. Esse sinal terá o valor lógico zero para os transistores do tipo NMOS e terá valor lógico um para os transistores do tipo PMOS. A representação dessa abordagem pode ser vista na Figura 22(b) onde o transistor original se torna transparente, ou seja, não tem ação sobre o estado lógico do circuito. Outra abordagem se baseia em forçar os transistores NMOS ou PMOS, a permanecerem sempre abertos através do terminal de controle do *gate*. O terminal *gate* receberá o zero lógico quando a simulação da falha for na rede *pull-down*, ou seja, nos transistores NMOS. Já quando a simulação for na rede *pull-up*, os transistores PMOS irão receber o nível alto lógico em seu terminal de controle. A Figura 22 (c) mostra como funciona essa metodologia. GND é o zero lógico, VDD o nível alto lógico do circuito, D é o dreno (*drain*), S é a fonte (*source*) e G é a porta (*gate*) de controle dos transistores.

Para as falhas do tipo *stuck-on*, existe a metodologia que baseia-se em remover o transistor do circuito e substituir uma linha. Também seguindo a metodologia de Jiang et al. (2009), pode-se utilizar resistência em série e em paralelo com o transistor, com valor suficientemente baixo para simular um curto-circuito entre o terminal de dreno e fonte. Jiang et al. (2009) utilizou resistências com o valor de 1 Ω na sua metodologia e sua representação pode ser vista na Figura 23 (a). Na abordagem de Schuster e Bryant (1983) são utilizados transistores adicionais em paralelo com o transistor onde se quer inserir a falha, com as mesmas especificações. O transistor adicional, será forçado a permanecer conduzindo através de um sinal fixo no terminal de controle do *gate*. Esse sinal terá o valor lógico **um** para os transistores do tipo NMOS e valor lógico **zero** para os transistores do tipo PMOS. A representação dessa abordagem pode ser vista na Figura 23(b) onde



(a) (b) (c)

Figura 23 – Representação de falhas do tipo *stuck-on*

o transistor original se torna transparente, ou seja, não tem ação sobre o estado lógico do circuito. Outra abordagem se baseia em forçar os transistores NMOS ou PMOS, a permanecerem sempre conduzindo através do terminal de controle do *gate* sem a utilização dos transistores adicionais. O terminal *gate* receberá o nível alto lógico quando a simulação da falha for na rede *pull-down*, ou seja, nos transistores NMOS. Já quando a simulação for na rede *pull-up*, os transistores PMOS irão receber o zero lógico em seu terminal de controle. A Figura 23 (c) mostra como funciona essa metodologia. GND é o zero lógico, VDD o nível alto lógico do circuito, D é o dreno (*drain*), S é a fonte (*source*) e G é a porta (*gate*) de controle dos transistores.

Foram feitas simulações com todos os quatro métodos e verificou-se algumas diferenças entre eles com relação às falhas *stuck-on*. Durante uma falha *stuck-on* pode ocorrer uma condição onde as duas redes *pull-up* e *pull-down* conduzem ao mesmo tempo, provocando um curto-circuito. Nessa condição a saída apresentará um valor de tensão que será um divisor resistivo entre os transistores que estão conduzindo. Todas a metodologias apresentam diferença de valor nessa condição em específico, justamente por causa do divisor resistivo.

A metodologia adotada por Schuster e Bryant (1983) foi escolhida para descrever as falhas *stuck-on* e *stuck-open* nesse trabalho. Um dos motivos para essa escolha é o fato de não alterar a resistência equivalente dos transistores. Outro motivo interessante é o de não ser necessário alterar o sinal do *gate* do transistor onde a falha está sendo inserida. Por exemplo, em uma falha *stuck-on*, será adicionado um transistor em série que permanecerá sempre ligado deixando o transistor do circuito transparente. Independente do valor que aquele transistor tiver no seu *gate*, o transistor adicional fará com que aquele caminho esteja sempre conduzindo.

Ao se fazer a injeção de falhas é necessário uma metodologia para detectar se a falha se propagou até a saída do circuito. Falhas stuck-on e stuck-open possuem metodologias diferentes para serem detectadas e serão abordadas nas próximas subseções.

3.3.1 Falhas Stuck-On

As falhas to tipo *stuck-on*, fazem com que um transistor esteja sempre no estado de condução. Esse tipo de falha faz com que as redes *pull-up* e *pull-down* de um circuito conduzam ao mesmo tempo em certas condições e isso causa um divisor resistivo entre os transistores que estão conduzindo. Esse tipo de falha é caracterizado por uma alta corrente de curto-circuito e o valor da saída pode ou não ser interpretado como uma falha pelas portas lógicas subsequentes. Por esse motivo, medições serão feitas diretamente na saída do votador para detectar qualquer influência da falha inserida. Medições serão feitas na saída de dois inversores em série, conectados na saída do votador, para detectar quais falhas irão se propagar pelos estágios subsequentes.

A quantidade de vetores necessários para testar todas as falhas *stuck-on* é calculada pela Equação 1.2, sendo E o número de entradas do circuito. Votadores majoritários possuem 3 entradas, logo $V_{teste} = 8$. A tabela verdade de um votador majoritário é o suficiente para testar todas as falhas *stuck-on*.

A injeção de falhas foi feita de modo exaustivo, ou seja, testando todas as possibilidades para averiguar em quais situações a falha irá se propagar. É importante salientar que está sendo considerado apenas a presença de uma falha por vez. Para o circuito da Figura 14, por exemplo, para cada falha *stuck-on* é necessário testar 8 vetores de entrada. Esse circuito possui 14 transistores, logo são necessárias 112 análises para testar todas as possibilidades de uma falha *stuck-on* se propagar. Como o número de análises se torna muito grande para testar todos os votadores majoritários, um algoritmo foi desenvolvido para automatizar as simulações.

O Algoritmo 1 faz a leitura de um arquivo de entrada que contenha a descrição dos circuitos em linguagem SPICE. Uma simulação é feita para gerar os resultados do circuito sem falhas que posteriormente será comparado com os resultados na presença de falhas. Para cada transistor é feita a injeção de falha no modelo descrito por Schuster e Bryant (1983) e é gerado um arquivo que é simulado através do *software* NGSPICE (NENZI; VOGT, 2014). Os resultados gerados por essa simulação são comparados com os resultados do circuito sem falha. O processo é repetido até que uma falha seja inserida em cada transistor.

3.3.2 Falhas Stuck-Open

Uma falha *stuck-open* faz com que um transistor permaneça em um estado de alta impedância, ou seja, nenhuma corrente flui através dele. A presença de uma falha *stuck-open* em um circuito, faz com que dois terminais jamais se conectem. Esse tipo de falha pode não se manifestar imediatamente, mas nas próximas transições. Quando acontecer uma transição que dependa do transistor com falha, essa transição não irá

Algoritmo 1: Simulação de Falhas Stuck-On

```
1  início
2      Lê Netlist
3      para cada combinação do vetor de entrada faça
4          Simula o circuito livre de falhas
5      fim
6      para cada transistor do circuito faça
7          Adiciona 1 falha stuck-on
8          para cada combinação de entrada faça
9              Simula o circuito com falha
10         fim
11     fim
12     Compara os resultados obtidos com os resultados livres de falha
13     Identifica todas as falhas na saída do votador
14     Verifica se a falha é propagada
15 fim
```

ocorrer e o estado anterior será mantido por um certo tempo devido às capacitâncias internas e, nesse caso, o circuito se comporta como uma lógica sequencial. Por esse motivo são necessários dois vetores de entrada para testar se a falha irá se manifestar.

No circuito votador clássico da Figura 14, por exemplo, se houver uma falha *stuck-open* no transistor MP1, quando as entradas A, B e C estiverem no estado 001, deveria haver um caminho condutivo entre MP1, MP2 e MP3. Com a presença da falha, a entrada do inversor de saída nunca será conectada ao Vdd da fonte. Nesse caso, não é possível prever qual será o valor da saída, salvo quando utilizada uma sequência de dois vetores de entrada. Por exemplo, se as entradas A, B e C estiverem no estado lógico 011, haverá um caminho condutivo entre os transistores MN5 e MN6 e a entrada do inversor de saída será conectada ao terra da fonte. Nessa condição a saída do votador possui o valor lógico 1. Mas se, no próximo ciclo, houver uma transição para um estado em que A, B e C sejam 001, a entrada do inversor nunca será conectado ao Vdd da fonte devido à falha no transistor MP1. Nessa condição, as capacitâncias internas do circuito fazem com que o valor lógico 1 seja mantido por um certo período de tempo na saída do circuito quando o valor esperado é o valor lógico 0, caracterizando um erro causado pela falha. Simulações em nível lógico não são suficientes para testar esse tipo de falha, pois o tempo que o estado é mantido na saída durante a ocorrência da falha depende das características elétricas do circuito.

Os votadores majoritários possuem três entradas, logo o número total de combinações de vetores de entrada é 56. Para o circuito da Figura 14, por exemplo, que possui 14 transistores, são necessárias 784 análises para testar todas as possibilidades de uma falha *stuck-open* se propagar. Como o número de análises para as falhas *stuck-open* também é grande, outro algoritmo foi desenvolvido para automatizar as simulações.

O Algoritmo 2 faz a leitura de um arquivo de entrada que contenha a descrição dos circuitos em linguagem SPICE. Uma simulação é feita para cada par de vetores de entrada para gerar os resultados do circuito sem falhas. Para cada transistor, é feita a injeção de falha no modelo descrito por Schuster e Bryant (1983) e gerado um arquivo que é simulado através do *software* NGSPICE (NENZI; VOGT, 2014). Para cada falha é feita uma simulação para cada par de vetores de entrada e, então, os resultados são comparados com os resultados do circuito sem falha. O processo é repetido até que uma falha seja inserida em cada transistor.

Algoritmo 2: Simulação de Falhas Stuck-Open

```
1  início
2      Lê Netlist
3      para cada combinação dos pares de vetores de entrada faça
4          Simula o circuito livre de falhas
5      fim
6      para cada transistor do circuito faça
7          Adiciona 1 falha stuck-open
8          para cada par de vetor de entrada faça
9              Simula o circuito com falha
10         fim
11     fim
12     Compara os resultados obtidos com os resultados livres de falha
13     Identifica todas as falhas na saída do votador
14     Verifica se a falha é propagada
15 fim
```

3.4 Avaliação de Falhas Transientes

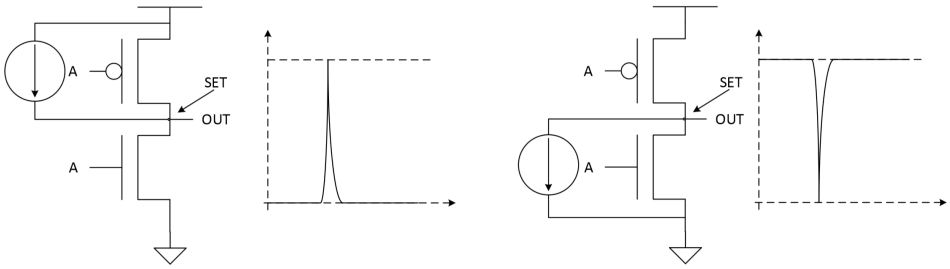
A simulação SET é feita através da inserção de uma fonte de corrente (dupla exponencial) no nó em que se deseja emular a falha. A corrente transiente gerada por um SEE normalmente é um pulso curto com um tempo de subida rápido e o tempo de descida dependente do tipo de partícula energética e das características da junção p-n atingida. A declaração de uma fonte de corrente dupla exponencial para o simulador NGSPICE é feita na seguinte forma:

$$I\{nome\} \quad no1 \quad no2 \quad EXP\{I1 \quad I2 \quad T_R \quad \tau_R \quad T_F \quad \tau_F\} \tag{3.2}$$

A Tabela 5 apresenta o significado de cada um dos termos da fonte de corrente mencionada na Equação 3.2. Os parâmetros τ_R e τ_F foram obtidos do relatório técnico de Carreno, Choi e Iyer (1990) que são valores utilizados por vários trabalhos e correspondem respectivamente a $163.93ps$ e $50ps$. A amplitude do pulso é definida de acordo com a

Tabela 5 – Constantes de tempo em uma exponencial dupla

Constantes	**Parâmetros**
I1	Corrente inicial
I2	Amplitude do pulso
T_R	Atraso do tempo de subida
τ_R	Constante de subida
T_F	Atraso do tempo de descida
τ_F	Constante de descida



(a) SET_{010} SET_{101}

Figura 24 – Simulação de um SET através de uma fonte de corrente

carga coletada pelo nó atingido de acordo com a Equação 3.3. Q_{coll} é calculado pela fórmula 3.4, onde L possui um valor típico de $2\mu m$ para cada let de $1MeV - cm^2/mg$. Para o experimento, os votadores serão avaliados sob a ação de LETs de intensidades de $1MeV - cm^2/mg$, $2MeV - cm^2/mg$ e $3MeV - cm^2/mg$ que correspondem aos valores de 189,64μA, 379,28μA e 568,92μA respectivamente. A simulação de SET será feita de modo exaustivo, inserindo um pulso de corrente em cada nó do circuito para todos os vetores de entrada, verificando se a falha se propaga para a saída do circuito.

$$I2 = \frac{Q_{coll}}{\tau_R - \tau_F} \tag{3.3}$$

$$Q_{coll} = 10,8 \times L \times LET \tag{3.4}$$

É possível modelar dois tipos de SET. Quando uma partícula atinge uma junção p-n de um transistor PMOS um pulso de corrente flui do vdd da fonte para o nó atingido. Esse pulso tem a característica 0 1 0, sendo denominado SET_{010} e pode ser visto na Figura 24 (a). Quando uma partícula atinge uma junção p-n de um transistor NMOS, um pulso de corrente flui do nó para o gnd da fonte. Esse pulso possui a característica 1 0 1, sendo denominado SET_{101}, e pode ser visto na Figura 24 (b).

A avaliação de falhas será feita de modo exaustivo, sendo que o número de falhas inseridas é calculado pelo número de nós de cada circuito multiplicado por oito, que é o número total de vetores de entrada para os votadores majoritários. A taxa de falhas é

calculada pelo número de falhas que se propagaram até a saída divido pelo número de falhas inseridas de acordo com a Equação 3.5.

$$\textbf{Taxa de Falhas} = \frac{\textbf{Número de Falhas Detectadas}}{\textbf{Número Falhas Inseridas}} \tag{3.5}$$

Foi desenvolvido um algoritmo para automatizar o processo de injeção e simulação de falhas transientes. O Algoritmo 3 faz a leitura de um arquivo de entrada que contenha a descrição dos circuitos em linguagem SPICE. Uma simulação é feita para cada par de vetores de entrada para gerar os resultados do circuito sem falhas. O algoritmo identifica todos os nós e insere um pulso de corrente com característica 010 para nós que conectam transistores PMOS ou insere um pulso de corrente com característica 101 para nós que conectam transistores do tipo NMOS. Em nós que conectam tanto transistores PMOS como NMOS, são necessárias a inserção das duas fontes de corrente 010 e 101 de cada vez. Os resultados das simulações de SET são comparados com os resultados do circuito sem falhas para avaliação das falhas que se propagaram pelo circuito.

Algoritmo 3: Simulação de Falhas SET

```
1  início
2      Lê Netlist
3      Simula circuito livre de falhas cobrindo todos os vetores de entrada
4      para cada nó do circuito faça
5          Insere pulsos de corrente que contemplem todas combinações de vetores de
             entrada
6          Simula circuito com falha
7      fim
8      Compara os resultados obtidos
9      Identifica cada falha na saída do votador
10     Verifica se a falha é propagada
11 fim
```

4 Resultados

A escolha de um votador majoritário não deve se basear somente no mascaramento de falhas. Uma boa relação entre desempenho, consumo de potência e mascaramento de falhas deve ser levada em consideração. Desse modo a avaliação dos circuitos votadores foi feita em três etapas:

- Avaliação de desempenho e potência;
- Avaliação de falhas permanentes;
- Avaliação de falhas transientes.

Todas as três etapas foram realizadas em um cenário onde os transistores foram dimensionados pelo esforço lógico, visando o equilíbrio dos atrasos. Esse dimensionamento foi estipulado para se ter uma comparação justa entre todos os votadores. As simulações foram realizadas através do simulador elétrico NGSPICE (NENZI; VOGT, 2014) em uma frequência de transição dos sinais de entrada de $1Ghz$.

4.1 Desempenho e Potência

Como abordado na metodologia, foi utilizado o modelo da Figura 21, sendo os votadores majoritários o CST. Para o desempenho, os inversores de entrada simulam uma degradação nos sinais de entrada. Os inversores na saída simulam a capacitância de carga como se o circuito estivesse inserido em uma condição real de funcionamento. Os tempos de propagação e transição foram medidos em cada arco de atraso. A Tabela 6 mostra os resultados de desempenho dos votadores avaliados.

Tabela 6 – Resultados de Desempenho

Votadores	**Tempo de Propagação (ps)**			**Tempo de Transição (ps)**		
	Máximo	**Médio**	**Desvio Padrão**	**Máximo**	**Médio**	**Desvio Padrão**
BAN	31,75	18,50	9,04	43,36	34,91	3,99
Clássico	36,96	27,74	5,81	20,81	19,72	0,87
Kshirsagar	47,16	27,06	16,84	50,64	37,45	5,03
MUX	40,95	29,07	7,13	17,57	15,42	1,52
NAND	18,74	16,36	1,52	20,42	18,53	1,27
NOR	29,76	23,27	3,86	29,76	23,27	3,86
TR	44,16	32,90	6,23	55,43	30,90	16,96

Para avaliar o desempenho foi medido o tempo de propagação, pois quanto menor esse tempo mais rápido é possível operar com este circuito. Dois circuitos se destacaram no desempenho: o votador BAN e o NAND obtiveram os menores tempos de propagação. A

média dos tempos de propagação é de 24,98ps com desvio padrão de 5,92ps. Como pode ser visto na Tabela 6, esses dois votadores possuem valores de atraso médio abaixo da média de atrasos decrementada de um desvio padrão. O votador NAND possui um desvio padrão pequeno, ou seja, todas os vetores de entrada possuem um tempo de resposta similar. O votador BAN possui um alto desvio padrão, ou seja, enquanto alguns vetores de entrada possuem um bom tempo de resposta, outros possuem um tempo de propagação longo. O pior tempo de propagação do votador BAN, de 31,75ps, ficou acima da média de atrasos incrementada de um desvio padrão. O gráfico da Figura 25 mostra uma comparação entre os tempos de propagação dos votadores NAND e BAN. Como pode ser visto, o NAND teve pouca variação nos tempos de propagação, e associado ao fato de ter a menor média de atrasos, faz com que seja considerado como o votador de melhor desempenho.

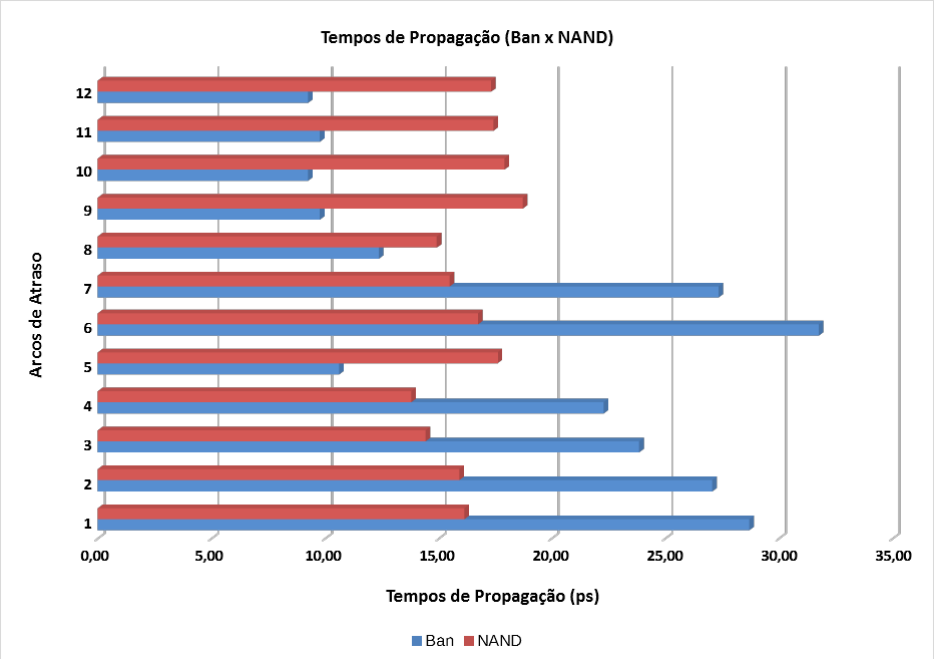


Figura 25 – Gráfico dos melhores resultados em desempenho (Ban × NAND)

Os tempos de transição da Tabela 6 representam o tempo que cada votador demora para transicionar a saída de 0 para 1 ou vice-versa. Como em um circuito real, a saída dos votadores será a entrada de outras portas lógicas, e o tempo que o sinal demora para transicionar de um estado lógico para outro, afeta o tempo de propagação global (WESTE; HARRIS, 2011). Tempos de transição mais longos causarão maior impacto no tempo de propagação global. O votador MUX apresentou o melhor tempo de transição, porém, quando comparado aos tempos de propagação, o votador MUX não é a melhor escolha. O votador NAND possui o segundo melhor tempo de transição e o melhor tempo de propagação, assim ele pode ser considerado o votador com melhor desempenho.

A potência dos circuitos avaliados é apresentado na Tabela 7. A média das potências dos circuitos avaliados é de 1,66μW, com um desvio padrão de 0,31μW. Os votadores BAN e NAND obtiveram o melhor resultado em consumo de energia, pois possuem um

valor abaixo da média das potências decrementado do desvio padrão. O PDP médio possui um valor médio de $42{,}81\mu W \times ps$ com um desvio padrão de $16{,}30\mu W \times ps$. Os votadores BAN e NAND tem um PDP médio abaixo da média decrementada do desvio padrão e podem ser considerados os melhores resultados considerando essa métrica. O PDP máximo possui um valor médio de $61{,}35\mu W \times ps$ com um desvio padrão de $25{,}79\mu W \times ps$. Por essa métrica o votador NAND é o único que possui um valor abaixo da média decrementada do desvio padrão e pode ser considerado o melhor resultado em desempenho e potência.

Tabela 7 – Resultados de Dissipação de Potência

Votadores	**Potência Total (uW) 1Ghz**	**PDP**		**Potência Estática (nW)**	
		Médio	**Máximo**	**Média**	**Desvio Padrão**
BAN	1,25	23,13	39,69	28,23	0,67
Clássico	1,91	52,98	70,59	17,33	11,73
Kshirsagar	2,05	55,47	96,68	66,35	5,25
MUX	1,69	49,13	69,21	64,36	6,51
NAND	1,28	20,94	23,99	29,56	6,56
NOR	1,54	35,84	45,83	26,45	7,58
TR	1,89	62,18	83,46	24,50	17,05

A potência estática dos circuitos é a potência quando o circuito não está transicionando, quando os sinais do circuitos estão estáveis. Para tecnologias nanométricas, a potência estática pode ser uma parte considerável da potência ativa e não deve ser ignorada. Pela Tabela 7, os votadores clássicos e o TR apresentaram os melhores resultados. O desvio padrão da potência estática desses votadores é alto, e isso quer dizer que existem algumas condições dos vetores de entrada onde esses votadores consomem mais energia. Os votadores NOR e BAN possuem o terceiro e o quarto menor consumo estático, mas claramente o BAN é o votador com o menor desvio padrão, e isso significa que o votador BAN possui um consumo estático semelhante, independente da condição dos vetores de entrada. Como o votador BAN também possui o melhor consumo ativo, pode-se dizer que é o votador com o menor consumo de energia.

4.2 Avaliação de Falhas Permanentes

Para a simulação de falhas permanentes também foi utilizado o circuito a Figura 21. A Tabela 8 mostra os resultados para a injeção de falhas do tipo *stuck-open* realizada de modo exaustivo considerando uma falha em cada transistor por vez. O votador TR obteve o melhor resultado em taxa de falhas desse tipo. Esse circuito foi projetado por El-Razouk e Abid (2007) utilizando um votador composto por portas NAND e adicionando transistores adicionais em paralelo na rede *pull-up* para aumentar a tolerância a falhas permanentes do tipo *stuck-on*. Como é observado, essa estratégia realmente tornou o circuito mais robusto contra esse tipo de falhas. Além disso, pode-se notar que esse circuito preserva a resposta de 46,43% dos seus pares de vetores de entrada enquanto todos os outros votadores não tiveram nenhum, ou quase nenhum, dos seus pares de vetores de

entrada protegidos contra falhas *stuck-open*. Essa implementação prova que redundância de transistor bem adicionada aumenta consideravelmente a robustez do circuito. O votador TR tem pelo menos duas vezes mais área de chip que o votador NAND, em compensação melhorou-se a taxa de falhas em mais de três vezes.

Tabela 8 – Resultados da taxa de falhas do tipo *stuck-open*

Stuck-Open								
Votadores	**Número de Transistores**	**Número de Simulações**	**Saída dos Votadores**			**Saída dos Inversores**		
			# Erros	**% Erros**	**Pares de Vetores Livres de Falhas**	**# Erros**	**% Erros**	**Pares de Vetores Livres de Falhas**
BAN	14	784	143	18,24	0	52	6,63	35
Classico	14	784	112	14,29	0	93	11,86	12
Kshirsagar	30	1680	182	10,83	0	66	3,93	37
MUX	22	1232	220	17,86	0	185	15,02	4
NAND	18	1008	155	15,38	7	106	10,52	22
NOR	18	1008	171	16,96	7	152	15,08	7
TR	36	2016	80	3,97	26	62	3,08	34

A Tabela 8 mostra que grande parte das falhas é mascarada eletricamente quando passa por dois níveis de inversores. Circuitos como o BAN e Kshirsagar apresentaram mais de 60% de falhas mascaradas eletricamente pelos dois inversores de saída. Esses dois circuitos apresentam uma peculiaridade, pois possuem transistores de passagem no multiplexador da saída do circuito. Quando existe alguma falha *stuck-open* em um transistor do tipo PMOS dos inversores de passagem, por exemplo, e a saída esperada for o valor lógico zero, em vez disso a saída será um valor de tensão correspondente à tensão de *threshold* do transistor do tipo NMOS. Essa tensão de *threshold* sempre será mascarada pelos inversores em série com o CST. O mesmo acontece se houver uma falha *stuck-open* em um transistor do tipo NMOS dos transistores de passagem e a saída esperada for o valor lógico 1. Em vez disso a saída será uma tensão correspondente a tensão de alimentação menos a tensão de *threshold* (V_{TH}) do transistor do tipo PMOS. Isso refletiu diretamente no mascaramento elétrico após dois níveis de inversores dos votadores BAN e Kshirsagar.

Considerando a taxa de falhas observada na saída dos inversores em série, o votador Kshirsagar obteve um resultado semelhante ao votador TR. Inclusive o votador Kshirsagar teve 66,07% dos pares de vetores de entrada protegidos contra falhas *stuck-open* enquanto o TR teve 60,71% dos seus pares de vetores protegidos.

Observa-se também que os votadores projetados para apresentarem uma maior tolerância a falhas realmente obtiveram melhores resultados na campanha de testes contra falhas *stuck-open*. O votador BAN por exemplo, apesar de não obter um resultado semelhante ao Kshirsagar e o TR, também obteve um bom resultado. O votador MUX, que é um circuito semelhante ao BAN que tem seu multiplexador implementado através de portas NAND, prova que essa implementação não é muito eficiente contra falhas *stuck-open*.

A Tabela 9 apresenta os resultados da análise exaustiva dos votadores majoritários

contra falhas permanentes do tipo *stuck-on*. Para o resultado da taxa de falhas na saída do votador majoritário, novamente o votador TR se destaca. Isso acontece pois esse circuito adiciona transistores adicionais em série na rede *pull-down* para aumentar a tolerância contra falhas desse tipo. A falha *stuck-on* se caracteriza por uma falha que faz com que o transistor permaneça no estado de condução e, dessa forma, os transistores em série garantem que o votador transicione corretamente para o estado esperado. O votador Kshirsagar também obteve um bom resultado, pois ficou abaixo do desvio padrão inferior da média dos votadores. O Kshirsagar possui um codificador de prioridade projetado para conduzir o sinal selecionado para o multiplexador, e isso também se provou ser uma boa estratégia para tolerar falhas do tipo *stuck-on*.

Tabela 9 – Resultados da taxa de falhas do tipo *stuck-on*

Stuck-On								
Votadores	**Número de Transistores**	**Número de Simulações**	**Saída dos Votadores**			**Saída dos Inversores**		
			# Erros	**% Erros**	**Vetores Livres de Falhas**	**# Erros**	**% Erros**	**Vetores Livres de Falhas**
BAN	14	112	21	18,75	3	7	6,25	4
Classico	14	112	20	12,50	0	10	8,93	1
Kshirsagar	30	240	18	7,50	4	7	2,92	5
MUX	22	176	31	17,61	0	21	11,93	0
NAND	18	144	27	18,75	1	21	14,58	1
NOR	18	144	27	18,75	1	9	6,25	2
TR	36	288	18	6,25	2	18	6,25	2

Observando-se os erros que se propagaram da saída do votador para os próximos dois níveis de inversores, nota-se um grande mascaramento elétrico de falhas para algumas arquiteturas. Os votadores BAN, Kshirsagar e NOR obtiveram mais de 60% das falhas *stuck-on* mascaradas. O votadores BAN e Kshirsagar tiveram esse comportamento devido às falhas que ocorrem nos transistores de passagem. Por exemplo, considerando uma falha *stuck-on* no transistor MP6 do votador BAN da Figura 18. Quando suas entradas ABC estiverem no estado 101, numa condição normal de operação os transistores MP7 e MN7 conduziriam o valor da entrada C para saída, que corresponde ao valor lógico 1. A falha *stuck-on* no transistor MP6 faz com que o valor de B também seja conduzido para a saída, mas como se trata de um transistor do tipo PMOS, será a tensão de *threshold* que aparecerá na saída. Nessas condições, com os transistores MP6, MP7 e MN7 conduzindo ao mesmo tempo, será criado um divisor resistivo na saída. Como o valor da saída terá um valor muito próximo de 1V, provavelmente será mascarado pelos próximos estágios de inversores. Essa condição ilustra o que ocorre para haver tantos mascaramentos elétricos nos votadores BAN e Kshirsagar. O votador NOR também obteve muitos mascaramentos elétricos devido à sua arquitetura e ao dimensionamento dos transistores. Se houver uma falha *stuck-on* em algum transistor da rede *pull-down* na porta NOR da saída do votador NOR, por exemplo, um divisor resistivo irá ocorrer em algumas condições quando essa porta lógica tentar fazer uma transição para 1, conectando o V_{DD} da fonte à saída. Como essa porta NOR possui apenas um transistor do tipo NMOS conduzindo ao mesmo tempo

que os três transistores PMOS da rede *pull-up*, muito provavelmente o valor de tensão do divisor resistivo terá um valor próximo de 1V que será mascarado pelos estágios de inversores subsequentes.

Para a taxa de falhas após o mascaramento dos inversores o votador Kshirsagar obteve o melhor resultado, pois é o único votador que ficou abaixo do desvio padrão inferior da média de todos os votadores. Além disso o Kshirsagar é o votador que tem a maior parte dos seus vetores de entrada protegidos contra falhas *stuck-on*, mais de 50%.

4.3 Avaliação de Falhas Transientes

A avaliação de falhas transientes foi realizada no mesmo cenários em que as falhas permanentes foram avaliadas. O circuito da Figura 21 foi utilizado para simular uma situação real de operação e, além disso, para observar a propagação de um erro causado por uma falha do tipo SET para os próximos dois níveis de inversores.

A Tabela 10 apresenta os resultados da taxa de falhas dos votadores contra falhas do tipo SET, onde S representa a saída do votador enquanto SM representa a saída após dois estágios de inversores. Para LET1, o votador TR obteve o melhor resultado. Como esse circuito utiliza redundância de transistores para mascarar falhas permanentes, é esperado que a quantidade adicional de transistores atenue SETs de baixa intensidade mascarando-os eletricamente.

Tabela 10 – Resultados da taxa de falhas do tipo *SET*

Votadores	**Falhas Inseridas**	**LET1**		**LET2**		**LET3**	
		% Erros (S)	**% Erros (SM)**	**% Erros (S)**	**% Erros (SM)**	**% Erros (S)**	**% Erros (SM)**
BAN	112	23,21%	22,32%	29,46%	27,68%	33,04%	32,14%
Classico	72	25,00%	25,00%	33,33%	27,78%	33,33%	33,33%
Kshirsagar	208	17,31%	15,87%	21,15%	19,71%	23,08%	21,15%
MUX	136	23,53%	23,53%	24,26%	24,26%	27,21%	26,47%
NAND	104	18,27%	18,27%	25,00%	23,08%	25,00%	25,00%
NOR	104	20,19%	20,19%	30,77%	26,92%	37,50%	37,50%
TR	176	10,80%	10,23%	17,61%	17,61%	25,57%	18,18%

O experimento mostrou que o mascaramento elétrico nos estágios subsequentes de inversores é pequeno, em média 1,35% de mascaramentos. Talvez o mascaramento elétrico possa ser maior se for utilizada uma cadeia maior de inversores na saída dos votadores.

Os votadores MUX e NAND são os que tem menor variação na taxa de falhas considerando todas as intensidades do LET. Isso significa que ocorre pouco mascaramento elétrico e que o resultado dessa taxa de falhas provém de mascaramento lógico. Considerando a média dos resultados, o votador TR obteve melhor resultado com 16,67% de taxa de falhas. O desvio padrão dos resultados do TR é de 5,64%, pois o votador TR tem grande mascaramento elétrico com LETs de baixa intensidade, mas se aproxima da taxa de falhas dos outros votadores majoritários quando a intensidade do LET aumenta. A média entre

todos os votadores é de 24,31% com desvio padrão de 4,93% e o votador TR foi o único que ficou abaixo do desvio padrão inferior. O votador Kshirsagar obteve o segundo melhor resultado com uma média de 19,71% na taxa de falhas e um desvio padrão de 2,69%. O desvio padrão menor que o votador TR indica que houve menor mascaramento elétrico nas LETs de baixa intensidade.

5 Conclusão

A miniaturização da tecnologia faz com que seja cada vez mais difícil garantir que um circuito não tenha falhas. Além disso, a redução das dimensões dos transistores os torna cada vez mais sensíveis a falhas causadas por radiação. O TMR é uma técnica bastante difundida utilizada para tolerar falhas. Contudo, o votador majoritário pode comprometer o funcionamento ideal do sistema se este também não for robusto. A utilização de um votador robusto pode incrementar a robustez do sistema sem comprometer demasiadamente o desempenho e o consumo de potência.

O presente trabalho avaliou sete diferentes implementações de votadores majoritários avaliando o desempenho, consumo de potência, taxa de falhas permanentes e taxa de falhas transientes. O levantamento dos votadores foi feito através de uma extensiva revisão bibliográfica. Cada votador majoritário foi projetado com um objetivo diferente e cumprem bem o objetivo para o qual foram projetados, mas quando todos são inseridos em um cenário de equilíbrio para avaliar falhas permanentes e transientes em nível elétrico, todos apresentam resultados diferentes. A avaliação da taxa de falhas foi feita de modo exaustivo, ou seja, testando cada falha para todas as combinações dos vetores de entrada. Nessa situação, supõe-se que em alguns momentos havia uma falha em algum dos módulos do sistema TMR além da falha inserida no votador majoritário. Há um cenário que não foi abordado neste trabalho, que é a situação em que todos os módulos estão funcionando corretamente. Neste ultimo cenário, todas as entradas dos votadores deveriam estar com o valor lógico 0 ou 1. Apesar do trabalho não ter apresentado essa análise, é possível extrair essa análise dos resultados já obtidos. Nas tabelas dos Apêndices A e B, que apresentam os resultados completos para os votadores BAN e CLASSICO, é possível extrair outra tabela para os resultados onde os módulos estão funcionando corretamente.

Na avaliação de falhas *stuck-open* o votador TR obteve os melhores resultados quando levado em consideração tanto o mascaramento de falhas na saída do inversor como na saída dos dois inversores e o votador Kshirsagar teve um resultado semelhante quando levado em consideração o mascaramento de falhas após os dois inversores. Para as falhas *stuck-on*, novamente o votador TR se destaca juntamente com o votador Kshirsagar quando levado em consideração o mascaramento de falhas na saída do votador, mas o Kshirsagar se sobressai quando levado em consideração o mascaramento após dois níveis de inversores. Como tanto o Kshirsagar como o TR possuem um PDP semelhante, pode-se optar por um circuito com menor consumo e melhor desempenho como o Ban, que também apresenta resultados de taxa de falhas melhores que o votador Clássico.

Os experimentos com falhas do tipo SET comprovaram que as tecnologias nanomé-

tricas realmente são sensíveis a falhas transientes. A média de taxa de falhas transientes para os votadores avaliados é de 24,31%, com um desvio padrão de 6,31%. O votador que se destacou no mascaramento elétrico de SET de baixa intensidade é o votador TR, porém a taxa de falhas se degrada à medida que a intensidade do SET é aumentada. O Kshirsagar é um votador que obteve um dos melhores resultados no mascaramento de SET e não teve muita variação nos resultados quando a intensidade do SET é aumentada.

A Tabela 11 apresenta um resumo dos resultados obtidos, sendo apresentada a média dos valores. É possível ver claramente que o TR e Kshirsagar são os mais robustos em taxa de falhas permanentes e transientes. Apesar destes não apresentarem os melhores resultados em termos de desempenho e consumo de energia, é importante lembrar que melhorias nestas características podem ser obtidas a partir de um dimensionamento específico, mantendo desta forma a robustez a falhas e se aproximando dos melhores resultados em desempenho.

Tabela 11 – Resumo dos resultados dos tempos de atraso, potência e a das coberturas de falhas *stuck-on*, *stuck-open* e SET

Votadores	**Atraso (ps)**	**Consumo (uW)**	**Cobertura de Falhas Stuck-On**	**Cobertura de Falhas Stuck-Open**	**Cobertura de Falhas SET**
BAN	18,50	1,95	18,75%	18,24%	29,46%
Classico	27,74	2,28	12,50%	14,29%	33,33%
Kshirsagar	27,06	3,35	7,50%	10,83%	21,15%
MUX	29,07	2,76	17,61%	17,86%	24,26%
NAND	16,36	1,88	18,75%	15,38%	25,00%
NOR	23,27	2,25	18,75%	16,96%	30,77%
TR	32,90	2,77	6,25%	3,97%	17,61%

Referências

ABRAHAM, J.; FUCHS, W. Fault and error models for VLSI. *Proceedings of the IEEE*, v. 74, n. 5, p. 639–654, 1986. Citado na página 29.

ALMUKHAIZIM, S.; SINANOGLU, O. A Hazard-Free Majority Voter for TMR-Based Fault Tolerance in Asynchronous Circuits. In: *2007 2nd International Design and Test Workshop*. [S.l.]: IEEE, 2007. p. 93–98. Citado na página 22.

ANGHEL, L.; ALEXANDRESCU, D.; NICOLAIDIS, M. Evaluation of a soft error tolerance technique based on time and/or space redundancy. *Proceedings 13th Symposium on Integrated Circuits and Systems Design (Cat. No.PR00843)*, 2000. Citado na página 35.

AVIZIENIS, A.; KELLY, J. P. J. *FAULT TOLERANCE BY DESIGN DIVERSITY: CONCEPTS AND EXPERIMENTS*. 1984. 67–80 p. Citado na página 38.

BAN, T.; De Barros Naviner, L. A. A simple fault-tolerant digital voter circuit in TMR nanoarchitectures. *Proceedings of the 8th IEEE International NEWCAS Conference, NEWCAS2010*, p. 269–272, 2010. Citado 7 vezes nas páginas 13, 19, 22, 41, 43, 44 e 47.

BAUMANN, R. Soft Errors in Advanced Computer Systems. *IEEE Design and Test of Computers*, v. 22, n. 3, p. 258–266, maio 2005. Citado na página 21.

BAUMANN, R. C. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, v. 5, n. 3, p. 305–315, 2005. Citado 3 vezes nas páginas 13, 32 e 33.

BAZE, M. P.; BUCHNER, S. P.; MCMORROW, D. A digital CMOS design technique for SEU hardening. *IEEE Transactions on Nuclear Science*, v. 47, n. 6 III, p. 2603–2608, 2000. ISSN 00189499. Citado na página 22.

BOLIN, H. R. Process Defects and Effects on Mosfet Gate Reliability. In: *18th International Reliability Physics Symposium*. [S.l.]: IEEE, 1980. p. 252–254. Citado na página 21.

CARRENO, V. A.; CHOI, G.; IYER, R. *Analog-digital simulation of transient-induced logic errors and upset susceptibility of an advanced control system*. [S.l.], 1990. Citado na página 54.

CAZEAUX, J.; ROSSI, D.; METRA, C. New high speed CMOS self-checking voter. *Proceedings. 10th IEEE International On-Line Testing Symposium*, IEEE Comput. Soc, p. 58–63, 2004. Citado 2 vezes nas páginas 36 e 41.

CAZEAUX, J. M.; ROSSI, D.; METRA, C. Self-checking voter for high speed TMR systems. In: *Journal of Electronic Testing: Theory and Applications (JETTA)*. [S.l.: s.n.], 2005. v. 21, n. 4, p. 377–389. ISSN 09238174. Citado 2 vezes nas páginas 22 e 47.

CNNMONEY. *eBay site crashes again*. 1999. Http://money.cnn.com/1999/08/06/technology/ebay/. Online; Acessado em 9 de Março de 2015. Citado na página 22.

COLBOURNE, E.; COVERLEY, G.; BEHERA, S. Reliability of MOS LSl circuits. *Proceedings of the IEEE*, v. 62, n. 2, p. 244–259, 1974. ISSN 0018-9219. Citado na página 21.

DANILOV, I. A.; GORBUNOV, M. S.; ANTONOV, A. A. SET tolerance of 65 nm CMOS majority voters: A comparative study. In: *2013 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. [S.l.: s.n.], 2013. p. 1–6. Citado 3 vezes nas páginas 41, 44 e 47.

DENNARD, R. H.; GANSSLEN, F.; YU, H.-N. Design of Ion Implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, SC-9, p. 256, 1974. Citado na página 21.

EL-RAZOUK, H.; ABID, Z. A new transistor-redundant voter for defect-tolerant digital circuits. *Canadian Conference on Electrical and Computer Engineering*, n. May, p. 1078–1081, 2007. Citado 7 vezes nas páginas 13, 22, 41, 45, 46, 47 e 59.

ELMENDORF, W. R. Fault-tolerant programming. In: *Proceedings of the 2nd International Symposium on Fault-tolerant Computing (FTCS-2)*. [S.l.: s.n.], 1972. v. 31, p. 79–83. Citado na página 38.

GOMES, I. A. C. *Uso de Redundância Modular Tripla Aproximada para Tolerância a Falhas em Circuitos Digitais*. 92 p. Tese (Dissertação de Mestrado) — Universidade Federal do Rio Grande do Sul, 2014. Citado na página 39.

GOMES, I. A. C.; KASTENSMIDT, F. G. L. Reducing TMR overhead by combining approximate circuit, transistor topology and input permutation approaches. In: *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*. [S.l.]: IEEE, 2013. p. 1–6. Citado 2 vezes nas páginas 13 e 39.

GOMEZ, R. et al. A modern look at the CMOS stuck-open fault. *2009 10th Latin American Test Workshop, LATW 2009*, 2009. Citado 2 vezes nas páginas 49 e 50.

HERNANDEZ, M. A.; ARANDA, M. L. CMOS full-adders for energy-efficient arithmetic applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 19, n. 4, p. 718–721, 2011. Citado na página 48.

JIANG, S. et al. Experiment and simulation of transistor level fault model of IDDT test. *2009 International Conference on Applied Superconductivity and Electromagnetic Devices, ASEMD 2009*, n. 60871056, p. 133–137, 2009. Citado 2 vezes nas páginas 49 e 50.

JOHNSON, B. W. An introduction to the design and analysis of fault-tolerant systems. *Fault-tolerant computer system design*, p. 1–108, 1996. Citado na página 26.

KOREN, I.; KRISHNA, C. M. *Fault Tolerant Systems*. [S.l.]: Morgan Kaufmann, 2010. 378 p. Citado 5 vezes nas páginas 21, 25, 28, 35 e 36.

KSHIRSAGAR, R. V.; PATRIKAR, R. M. Design of a novel fault-tolerant voter circuit for TMR implementation to improve reliability in digital circuits. *Microelectronics Reliability*, Elsevier Ltd, v. 49, n. 12, p. 1573–1577, 2009. Citado 7 vezes nas páginas 13, 19, 22, 41, 43, 44 e 47.

Kuang-Wei Chiang; VRANESIC, Z. On Fault Detection in CMOS Logic Networks. In: *20th Design Automation Conference Proceedings*. [S.l.]: IEEE, 1983. v. 5, n. 2, p. 50–56. Citado 2 vezes nas páginas 29 e 30.

LANN, G. L. An analysis of the ariane 5 flight 501 failure-a system engineering perspective. In: IEEE. *Engineering of Computer-Based Systems, 1997. Proceedings., International Conference and Workshop on*. [S.l.], 1997. p. 339–346. Citado na página 22.

LI, Y. et al. Overcoming Early-Life Failure and Aging for Robust Systems. *IEEE Design & Test of Computers*, v. 26, n. 6, p. 28–39, nov. 2009. Citado 2 vezes nas páginas 21 e 22.

MANGIR, T. Sources of failures and yield improvement for VLSI and reconstructable interconnects for RVLSI and WSI: Part I-sources of failures and yield improvements for VLSI. *Microelectronics Journal*, v. 16, n. 6, p. 57, 1985. Citado na página 21.

MCCLUSKEY, E.; CLEGG, F. Fault Equivalence in Combinational Logic Networks. *IEEE Transactions on Computers*, C-20, n. 11, p. 1286–1293, 1971. Citado na página 28.

MESSENGER, G. C. Collection of Charge on Junction Nodes from Ion Tracks. *IEEE Transactions on Nuclear Science*, v. 29, n. 6, p. 2024–2031, 1982. Citado 3 vezes nas páginas 13, 33 e 34.

METRA, C.; FAVALLI, M.; RICCO, B. Compact and low power on-line self-testing voting scheme. *1997 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, p. 137–145, 1997. Citado 3 vezes nas páginas 41, 49 e 50.

MOORE, G. E. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, v. 86, n. 1, p. 82–85, 1998. Citado na página 21.

NENZI, P.; VOGT, H. *Ngspice Users Manual*. 2014. 592 p. Http://ngspice.sourceforge.net/. Online; Acessado em 10 de Agosto de 2015. Citado 4 vezes nas páginas 48, 52, 54 e 57.

NEUMANN, J. V. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, v. 34, p. 43–98, 1956. Citado na página 22.

PTM. *Predictive Technology Model*. 2014. Http://ptm.asu.edu/. Online; Acessado em 18 de Agosto de 2014. Citado na página 48.

RIVERS, J. a.; KUDVA, P. Reliability Challenges and System Performance at the Architecture Level. *IEEE Design & Test of Computers*, v. 26, n. 6, p. 62–73, nov. 2009. Citado na página 26.

SCHNABLE, G.; GALLACE, L.; PUJOL, H. Reliability of CMOS Integrated Circuits. *Computer*, v. 11, n. 10, p. 6–17, out. 1978. Citado na página 21.

SCHUSTER, M. D.; BRYANT, R. E. Concurrent fault simulation of MOS digital circuits. In: *Conference on Advanced Research in VLSI*. Pasadena: Artech House, 1983. p. 10. Citado 5 vezes nas páginas 49, 50, 51, 52 e 54.

SIERAWSKI, B. D.; BHUVA, B. L.; MASSENGILL, L. W. Reducing soft error rate in logic circuits through approximate logic functions. *IEEE Transactions on Nuclear Science*, v. 53, n. 6, p. 3417–3421, 2006. Citado na página 39.

SUTHERLAND, I. E.; SPROULL, R. F.; HARRIS, D. F. *Logical effort: designing fast CMOS circuits*. [S.l.]: Morgan Kaufmann, 1999. Citado na página 48.

TAMBARA, L. A. et al. Evaluating the effectiveness of a diversity TMR scheme under neutrons. In: *2013 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*. [S.l.]: IEEE, 2013. p. 1–5. Citado 2 vezes nas páginas 13 e 38.

VIAL, J. et al. Using TMR Architectures for Yield Improvement. *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Ieee, p. 7–15, out. 2008. Citado na página 22.

VIAL, J. et al. Is triple modular redundancy suitable for yield improvement? *IET Computers & Digital Techniques*, v. 3, n. 6, p. 581, 2009. Citado 3 vezes nas páginas 13, 36 e 37.

WADSACK, R. L. Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits. *Bell System Technical Journal*, v. 57, n. 5, p. 1449–1474, maio 1978. Citado 2 vezes nas páginas 29 e 31.

WANG, L.-T.; WU, C.-W.; WEN, X. *VLSI Test Principles and Architectures*. 1. ed. [S.l.]: Morgan Kaufmann, 2006. 808 p. Citado 2 vezes nas páginas 28 e 29.

WESTE, N. H. E.; HARRIS, D. M. *CMOS VLSI Design: A Circuits and Systems Perspective*. 4. ed. [S.l.]: Addison Wesley, 2011. 838 p. Citado na página 58.

Apêndices

APÊNDICE A – Tabelas de Taxa de Falhas Permanentes no Votador BAN

Tabela 12 – Tabela de Taxa de Falhas *Stuck-On* na Saída **S** do Votador BAN

	Esperado	MP1	MP2	MP3	MP4	MP5	MP6	MP7	MN1	MN2	MN3	MN4	MN5	MN6	MN7	Total
v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001	0	0	0,93	0,94	0	0	0	0,55	0	0	0	0	0,06	0	0,29	5
v010	0	0	0	0	0	0,13	0,55	0	0	0,19	0,17	0	0	0,29	0	5
v011	1	1	1	1	1	0,85	1	1	1	1	1	1	1	1	1	1
v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v101	1	1	1	1	1	0,4	0,69	1	0,93	1	1	0,93	1	0,58	1	5
v110	1	0,17	1	1	0,17	1	1	0,69	1	1	1	1	0,96	1	0,58	5
v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Total	0	1	1	1	1	3	2	2	1	1	1	1	2	2	2	21
%	0	12,5	12,5	12,5	12,5	37,5	25	25	12,5	12,5	12,5	12,5	25	25	25	18,8

Tabela 13 – Tabela de Taxa de Falhas *Stuck-Open* na Saída **S** do Votador BAN

Vetores	Esperado	MP1	MP2	MP3	MP4	MP5	MP6	MP7	MN1	MN2	MN3	MN4	MN5	MN6	MN7	Total
v000v001	0	0	0	0	0	0,36	0	0	0,13	0,12	0	0	0	0,12	0	4
v000v010	0	0	0	1	1	0	0	0	0	0	0	0	0,31	0	0,37	4
v000v011	1	1	1	1	1	1	1	0,65	1	1	1	1	0,74	1	1	2
v000v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,01	1
v000v101	1	0,01	0	1	1	1	1	0,64	1	1	1	1	0,28	1	1	4
v000v110	1	1	1	1	1	0,94	0,64	1	1	1	0,99	0,99	1	1	1	4
v000v111	1	1	1	1	1	1	0,64	1	1	1	1	1	1	1	1	1
v001v000	0	0	0	0	0	0,32	0	0	0	0	0	0	0	0,25	0	2
v001v010	0	0	0	1	1	0	0	0	0	0	0	0	0,32	0	0,37	4
v001v011	1	1	1	1	1	1	1	0,65	1	1	1	1	0,74	1	1	2
v001v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,02	1
v001v101	1	0,01	0	1	1	1	1	0,64	1	1	1	1	0,28	1	1	4
v001v110	1	1	1	1	1	0,88	0,64	1	1	1	0,99	0,99	1	1	1	4
v001v111	1	1	1	1	1	1	0,64	1	1	1	1	1	1	1	1	1
v010v000	0	0	0	0	0	-0,03	0	0	0	0	0	0	0	0,04	0	1
v010v001	0	0	0	0	0	0,67	0	0	1	1	0	0	0	0,37	0	4
v010v011	1	1	1	1	1	1	1	0,64	1	1	1	1	0,72	1	1	2
v010v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,25	1
v010v101	1	0,99	1	1	1	1	1	0,64	1	1	1	1	0,44	1	1	3
v010v110	1	1	1	1	1	0,85	0,64	1	1	1	0	0	1	1	1	4
v010v111	1	1	1	1	1	1	0,64	1	1	1	1	1	1	1	1	1
v011v000	0	0	0	0	0	0,35	0	0	0	0	0	0	0	0,36	0	2
v011v001	0	0	0	0	0	0,69	0	0	1	1	0	0	0	0,37	0	4
v011v010	0	0	0	0,96	0,96	0	0	0	0	0	0	0	0	0	0,37	3
v011v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,37	1
v011v101	1	0,98	0,99	1	1	1	1	0,82	1	1	1	1	0,95	1	1	4
v011v110	1	1	1	1	1	0,79	0,66	1	1	1	0	0	1	1	1	4
v011v111	1	1	1	1	1	1	0,97	1	1	1	1	1	1	1	1	1
v100v000	0	0	0	0	0	-0,04	0	0	0	0	0	0	0	0,02	0	1
v100v001	0	0	0	0	0	0,67	0	0	1	1	0	0	0	0,37	0	4
v100v010	0	0	0	0	0	0	0	0	0	0	0	0	0,08	0	0,12	2
v100v011	1	1	1	1	1	1	1	0,64	1	1	1	1	0,66	1	1	2
v100v101	1	0,03	0,03	1	1	1	1	0,64	1	1	1	1	0,4	1	1	4
v100v110	1	1	1	1	1	0,84	0,64	1	1	1	0	0	1	1	1	4
v100v111	1	1	1	1	1	1	0,65	1	1	1	1	1	1	1	1	1
v101v000	0	0	0	0	0	0,36	0	0	0	0	0	0	0	0,37	0	2
v101v001	0	0	0	0	0	0,69	0	0	1	1	0	0	0	0,37	0	4
v101v010	0	0	0	0,01	0	0	0	0	0	0	0	0	0	0	0,37	2
v101v011	1	1	1	1	1	1	1	0,75	1	1	1	1	1	1	1	1
v101v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,37	1
v101v110	1	1	1	1	1	0,79	0,64	1	1	1	0	0	1	1	1	4
v101v111	1	1	1	1	1	1	0,94	1	1	1	1	1	1	1	1	1
v110v000	0	0	0	0	0	0,34	0	0	0	0	0	0	0	0,37	0	2
v110v001	0	0	0	0	0	0,6	0	0	0,99	0,98	0	0	0	0,37	0	4
v110v010	0	0	0	0,96	1	0	0	0	0	0	0	0	0,26	0	0,37	4
v110v011	1	1	1	1	1	1	1	0,93	1	1	1	1	1,07	1	1	1
v110v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,36	1
v110v101	1	0	0	1	1	1	1	0,64	1	1	1	1	0,29	1	1	4
v110v111	1	1	1	1	1	1	0,75	1	1	1	1	1	1	1	1	1
v111v000	0	0	0	0	0	0,36	0	0	0	0	0	0	0	0,37	0	2
v111v001	0	0	0	0	0	0,68	0	0	0,99	0,98	0	0	0	0,37	0	4
v111v010	0	0	0	0,96	1	0	0	0	0	0	0	0	0,25	0	0,37	4
v111v011	1	1	1	1	1	1	1	0,95	1	1	1	1	1,07	1	1	1
v111v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,36	1
v111v101	1	0	0	1	1	1	1	0,64	1	1	1	1	0,29	1	1	4
v111v110	1	1	1	1	1	0,83	0,82	1	1	1	0,76	0,75	1	1	1	4
Total	0	7	6	6	5	19	14	14	7	7	7	7	16	14	14	143
%	0	12,5	10,71	10,71	8,93	33,93	25	25	12,5	12,5	12,5	12,5	28,57	25	25	18,24

Tabela 14 – Tabela de Taxa de Falhas *Stuck-On* na Saída **SM** do Votador BAN

	Esperado	MP1	MP2	MP3	MP4	MP5	MP6	MP7	MN1	MN2	MN3	MN4	MN5	MN6	MN7	Total
v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001	0	0	1	1	0	0	0	0,98	0	0	0	0	0	0	0	3
v010	0	0	0	0	0	0	0,98	0	0	0	0	0	0	0	0	1
v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v101	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
v110	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	2
v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Total	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	7
%	0	12,5	12,5	12,5	12,5	12,5	12,5	12,5	0	0	0	0	0	0	0	6,3

Tabela 15 – Tabela de Taxa de Falhas *Stuck-Open* na Saída **SM** do Votador BAN

Vetores	Esperado	MP1	MP2	MP3	MP4	MP5	MP6	MP7	MN1	MN2	MN3	MN4	MN5	MN6	MN7	Total
v000v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v000v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	2
v000v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v000v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v000v101	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	3
v000v110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v000v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v001v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	2
v001v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v001v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001v101	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	3
v001v110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v001v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v010v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v010v001	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	3
v010v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v010v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v010v101	1	1	1	1	1	1	1	1	1	1	1	1	0,01	1	1	1
v010v110	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	2
v010v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v011v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v011v001	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	3
v011v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	2
v011v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v011v101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v011v110	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	2
v011v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v100v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v100v001	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	3
v100v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v100v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v100v101	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	3
v100v110	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	2
v100v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v101v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v101v001	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	3
v101v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v101v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v101v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v101v110	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	2
v101v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v110v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v110v001	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	3
v110v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	2
v110v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v110v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v110v101	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	3
v110v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v111v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v111v001	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	3
v111v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	2
v111v011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
v111v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v111v101	1	0	0	1	1	1	1	1	1	1	1	1	0	1	1	3
v111v110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Total	0	5	5	5	5	6	0	0	6	6	4	4	6	0	0	52
%	0	8,93	8,93	8,93	8,93	10,71	0	0	10,71	10,71	7,14	7,14	10,71	0	0	6,63

APÊNDICE B – Tabelas de Taxa de Falhas Permanentes no Votador Clássico

Tabela 16 – Tabela de Taxa de Falhas *Stuck-On* na Saída **S** do Votador Clássico

	S	**MP1**	**MP2**	**MP3**	**MP4**	**MP5**	**MP6**	**MN1**	**MN2**	**MN3**	**MN4**	**MN5**	**MN6**	**MPi**	**MNi**	**Total**
v000	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,64	0,00	1
v001	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,04	0,00	0,04	0,00	0,64	0,00	3
v010	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,04	0,00	0,00	0,00	0,00	0,04	0,64	0,00	3
v011	1,00	0,25	1,00	1,00	0,25	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,64	3
v100	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,04	0,00	0,04	0,00	0,00	0,64	0,00	3
v101	1,00	1,00	0,25	1,00	1,00	0,25	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,64	3
v110	1,00	1,00	1,00	0,25	1,00	1,00	0,25	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,64	3
v111	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,64	1
Total	0	1	1	1	1	1	1	1	1	1	1	1	1	4	4	20
%	0,0%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	12,5%	50,0%	50,0%	17,9%

Tabela 17 – Tabelas de Taxa de Falhas *Stuck-Open* na Saída **S** do Votador Clássico

	S	MP1	MP2	MP3	MP4	MP5	MP6	MN1	MN2	MN3	MN4	MN5	MN6	Mpi	Mni	Total
v000v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v000v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v000v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0,04	1	3
v000v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v000v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0,04	1	3
v000v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0,02	1	3
v000v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,03	1	1
v001v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v001v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v001v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0,04	1	3
v001v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v001v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0,04	1	3
v001v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0,03	1	3
v001v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,04	1	1
v010v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v010v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v010v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0,04	1	3
v010v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v010v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0,04	1	3
v010v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0,03	1	3
v010v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,04	1	1
v011v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,04	1
v011v001	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1,04	3
v011v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1,05	3
v011v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1,04	3
v011v101	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v011v110	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v011v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v100v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v100v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v100v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,42	1
v100v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0,04	1	3
v100v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0,04	1	3
v100v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0,03	1	3
v100v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,04	1	1
v101v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,06	1
v101v001	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1,04	3
v101v010	0	0	0	1	0,99	0	0	0	0	0	0	0	0	0	1,04	3
v101v011	1	1	1	1	1	1	1	1	1	1	1	0,94	0,96	0,42	1	3
v101v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1,04	3
v101v110	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v101v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v110v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,04	1
v110v001	0	0,18	0,79	0	0	0	0	0	0	0	0	0	0	0	1,04	3
v110v010	0	0	0	1	0,55	0	0	0	0	0	0	0	0	0	1,04	3
v110v011	1	1	1	1	1	1	1	1	1	1	1	0,06	0,12	0,42	1	3
v110v100	0	0	0	0	0	0,95	1	0	0	0	0	0	0	0	1,04	3
v110v101	1	1	1	1	1	1	1	1	1	0,8	0,92	1	1	0,42	1	3
v110v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v111v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,06	1
v111v001	0	0,8	1	0	0	0	0	0	0	0	0	0	0	0	1,04	3
v111v010	0	0	0	1	0,94	0	0	0	0	0	0	0	0	0	1,04	3
v111v011	1	1	1	1	1	1	1	1	1	1	1	0,63	0,78	0,42	1	3
v111v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1,04	3
v111v101	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
v111v110	1	1	1	1	1	1	1	1	1	1	1	1	1	0,42	1	1
Total	0	4	4	4	4	4	4	4	4	5	5	7	7	28	28	112
%	0,00%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	8,90%	8,90%	12,50%	12,50%	50,00%	50,00%	14,30%

Tabela 18 – Tabela de Taxa de Falhas *Stuck-On* na Saída **SM** do Votador Clássico

	SM	MP1	MP2	MP3	MP4	MP5	MP6	MN1	MN2	MN3	MN4	MN5	MN6	MPi	MNi	Total
v000	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
v001	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
v010	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
v011	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	2
v100	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
v101	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	2
v110	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	2
v111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Total	0	1	1	1	1	1	1	0	0	0	0	0	0	4	0	10
%	0,00%	12,50%	12,50%	12,50%	12,50%	12,50%	12,50%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	50,00%	0,00%	8,90%

Tabela 19 – Tabelas de Taxa de Falhas *Stuck-Open* na Saída **SM** do Votador Clássico

	SM	MP1	MP2	MP3	MP4	MP5	MP6	MN1	MN2	MN3	MN4	MN5	MN6	Mpi	Mni	Total
v000v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v000v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v000v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	3
v000v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v000v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	3
v000v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	3
v000v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v001v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	3
v001v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v001v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	3
v001v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	3
v001v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v010v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v010v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v010v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	3
v010v100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v010v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	3
v010v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	3
v010v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v011v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
v011v001	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	3
v011v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	3
v011v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	3
v011v101	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v011v110	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v011v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v100v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v100v001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v100v010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v100v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	3
v100v101	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	3
v100v110	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	3
v100v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v101v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
v101v001	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	3
v101v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	3
v101v011	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v101v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	3
v101v110	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v101v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v110v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
v110v001	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	2
v110v010	0	0	0	1	0,99	0	0	0	0	0	0	0	0	0	1	3
v110v011	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	3
v110v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	3
v110v101	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v110v111	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v111v000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
v111v001	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	3
v111v010	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	3
v111v011	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v111v100	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	3
v111v101	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
v111v110	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
Total	0	3	4	4	4	4	4	4	4	4	4	5	5	28	16	93
%	0,00%	5,40%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	7,10%	8,90%	8,90%	50,00%	28,60%	11,90%