UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO VIEIRA PEREIRA

Teste da Rede de Interconexões de Field Programmable Analog Arrays

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Marcelo Lubaszewski
Orientador

Porto Alegre, setembro de 2005.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Pereira, Gustavo Vieira

Teste da Rede de Interconexões de Field Programmable Analog Arrays / Gustavo Vieira Pereira – Porto Alegre: PPGC da UFRGS, 2005.

208 f. : il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2005. Orientador: Marcelo Lubaszewski.

1. Dispositivos Analógicos Programáveis. 2. Rede de Interconexões. 3. Simulação de Falhas. 4. Algoritmos de Coloração de Grafos. 5. BIST. I. Lubaszewski, Marcelo. II. Título.

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Ciência da Computação, PPGC, pela oportunidade de realização de trabalhos em minha área de pesquisa, e ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), pelo suporte e apoio financeiro concedidos durante o período do mestrado.

Ao meu orientador, Marcelo Lubaszewski, por me conduzir e me iniciar na área de microeletrônica, até então desconhecida para mim. Seus conhecimentos, disponibilidade e paciência, sempre me incentivaram a desenvolver os trabalhos da melhor maneira possível.

A toda minha família, especialmente aos meus pais, Mariano e Ilda, pela total confiança creditada em todos os momentos, e a quem devo minha formação e minha vida.

Às minhas irmãs, Daniela, Viviane e Rita, que sempre me incentivaram a estudar e sempre me auxiliaram, de uma forma ou outra, quando precisei.

Aos colegas do PPGC pelo seu auxílio nas tarefas desenvolvidas durante o primeiro período do curso e apoio nos trabalhos realizados em aula.

Aos colegas do PPGEE que nunca hesitaram em ajudar-me quando precisei nos testes em bancada, e pela verdadeira amizade que tivemos.

E a todas as pessoas que de alguma forma contribuíram para minha formação, e, portanto, viabilizaram a execução deste trabalho, meus sinceros agradecimentos!

Ao Brasil!

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ABILBO: Analog Built-In Block Observer
BIST: Built in Self-test
CAB: Configurable Analog Block
CAD: Computer Aided Design
CLB: Configurable Logic Block
CT: Configuração de Teste
FPAA: Field Programmable Analog Array
FPGA: Field Programmable Gate Array
OBIST: Oscillation Built in Self Test
OPAMP: Operational Amplifier
OTS: Oscillation Test Strategy
PSO: Phase-Shift Oscillator
SC: Switched Capacitor
SM: Switch Matrix
TRAC: Totally Reconfigurable Analog Circuit
VHDL: Very-High Speed Integrated Circuit Hardware Description Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Os dispositivos analógicos programáveis (FPAAs, do inglês, *Field Programmable Analog Arrays*), apesar de ainda não terem a mesma popularidade de seus pares digitais (FPGAs, do inglês, *Field Programmable Gate Arrays*), possuem uma gama de aplicações bastante ampla, que vai desde o condicionamento de sinais em sistemas de instrumentação, até o processamento de sinais de radiofreqüência (RF) em telecomunicações.

Porém, ao mesmo tempo em que os FPAAs trouxeram um impressionante ganho na agilidade de concepção de circuitos analógicos, também trouxeram um conjunto de novos problemas relativos ao teste deste tipo de dispositivo. Os FPAAs podem ser divididos em duas partes fundamentais: seus blocos programáveis básicos (CABs, do inglês, *Configurable Analog Blocks*) e sua rede de interconexões. A rede de interconexões, por sua vez, pode ser dividida em duas partes: interconexões internas (locais e globais entre CABs) e interconexões externas (envolvendo células de I/O). Todas estas partes apresentam características estruturais e funcionais distintas, de forma que devem ser testadas separadamente, pois necessitam que se considerem modelos de falhas, configurações e estímulos de teste específicos para assegurar uma boa taxa de detecção de defeitos.

Como trabalhos anteriores já estudaram o teste dos CABs, o foco desta dissertação está direcionado ao desenvolvimento de metodologias que se propõem a testar a rede de interconexões de FPAAs. Apesar das várias diferenças entre as redes de interconexões de FPGAs e FPAAs, muitas também são as semelhanças entre elas, sendo, portanto, indiscutível que o ponto de partida deste trabalho tenha que ser o estudo das muitas técnicas propostas para o teste de interconexões em FPGAs, para posterior adaptação ao caso dos FPAAs. Além disto, embora o seu foco não recaia sobre o teste de CABs, pretende-se utilizá-los como recursos internos do dispositivo passíveis de gerar sinais e analisar respostas de teste, propondo uma abordagem de auto-teste integrado de interconexões que reduza o custo relativo ao equipamento externo de teste. Eventualmente, estes mesmos recursos poderão também ser utilizados para diagnóstico das partes defeituosas. Neste trabalho, utiliza-se como veículo de experimentação um dispositivo específico (Anadigm AN10E40), mas pretende-se que as metodologias de teste propostas sejam abrangentes e possam ser facilmente adaptadas a outros FPAAs comerciais que apresentem redes de interconexão semelhantes.

Palavras-Chaves: Dispositivos Analógicos Programáveis, Redes de Interconexões, Simulação de Falhas, Algoritmos de Coloração de Grafos, Autoteste Integrado.

Testing Interconnection Networks of Field Programmable Analog Arrays

ABSTRACT

Although Field Programmable Analog Arrays (FPAAs) are not yet as popular as their digital counterparts, the Field Programmable Gate Arrays (FPGAs), these programmable analog devices have a huge scope of applications that goes from signal conditioning in instrumentation systems to radio frequency signal processing in telecommunications.

However, at the same time as the FPAAs brought an impressive flexibility in the design of analog circuits, they also brought a set of new problems related to the test of this type of device. The FPAAs can be divided into two basic parts: the basic programmable blocks (Configurable Analog Blocks, CABs) and the interconnections network. The interconnections network can be divided into two additional parts: the internal interconnects (local and global connections between CABs) and the external interconnects (involving I/O cells). All these parts have particular structural and functional features, so that they need to be tested separately, since specific fault models, test configurations and test stimuli are required to ensure a good fault coverage.

As previous works have already studied the test of CABs, this work focuses on the development of methodologies to test the interconnections network of FPAAs. Despite the differences between the interconnection networks of FPGAs and FPAAs, many are also the similarities between them. Due to that, it becomes a must that the starting point of this work is the study of techniques proposed for the test of interconnections in FPGAs that can be considered for adaption to the case of the FPAAs. In addition, although the focus of this dissertation is not the test of CABs, it is intended to use them as internal device resources to generate signals and to analyze test responses. The idea is to propose a built-in self-test approach for the interconnections that leads to the reduction of the cost of the external test equipment. Eventually, these same resources could also be used to diagnose the defective parts. In this work, a specific test vehicle is used (the Anadigm AN10E40 device), but the goal is that the test methodologies proposed herein are general enough and can be easily adapted to other commercial FPAAs that have similar interconnection networks.

Keywords: Programmable Analog Devices, Interconnect Networks, Fault Simulation, Graph Colouring Algorithms, Built-In Self-Test.

1 INTRODUÇÃO

A crescente demanda de dispositivos digitais programáveis, FPGAs, e sua aplicação em sistemas em que falhas não podem ocorrer, por colocar em perigo vidas humanas ou equipamentos caros, por exemplo, motivou a pesquisa e desenvolvimento de estratégias de teste de tais dispositivos. Diversos trabalhos foram apresentados ao longo das últimas décadas, aprimorando tempo de teste, cobertura de falhas e o desenvolvimento de esquemas BIST (do inglês, Built in Self-test) para este tipo de dispositivo.

Recentemente, dispositivos analógicos programáveis, FPAAs, vêm sendo cada vez mais usados. Hoje já é possível encontrar FPAAs com diferentes tecnologias que possibilitam que estes componentes operem em freqüências na ordem de megahertz. A existência destes componentes permite ao projetista desenvolver sistemas complexos preocupando-se apenas com as questões de comunicação entre seus módulos, sem considerar detalhes de implementação de cada núcleo que o compõe, diminuindo, conseqüentemente, o tempo de projeto.

Diferentemente dos FPGAs, que possuem uma rede de interconexões global complexa, os FPAAs priorizam ligações locais em detrimento de ligações globais e, portanto, se por um lado o tempo de teste da rede global dos FPAAs deve ser menor, por outro, este tempo é maior quando a rede local é testada. Estas e outras diferenças entre as redes de interconexões de FPGAs e FPAAs existem, mas as semelhanças permanecem grandes, pois se trata de ligação física em blocos que implementam sub-funções de um sistema mais complexo. É, portanto, indiscutível que o ponto de partida tenha que ser o estudo dos muitos trabalhos que propõem técnicas de teste de interconexões em FPGAs para que possam ser adaptadas ao caso dos FPAAs.

O foco desta dissertação é o teste que visa detecção de falhas do tipo colagem (*stuck-open*/*stuck-on*) e paramétrica nas chaves, bem como, do tipo *open*, *short*, *bridging* e de *timing* nas linhas que interconectam os elementos internos do FPAA (envolvendo a rede local e global de interconexões) e nas chaves e *buffers* que habilitam a comunicação deste dispositivo com o mundo externo (envolvendo células de I/O). Várias foram às técnicas estudadas para o teste e detecção destas falhas em FPGAs e outros dispositivos digitais. Pode-se citar, como uma alternativa viável, o auto-teste integrado, o qual consiste em integrar algumas ou todas as funções de teste no próprio *chip* ou na placa a ser testada, conforme abordado em [CHE 2003], [FER 2003], [GIB 97], [AND 2004a], [LAL 2003], [PER 2005], [STR 98] e [SUN 2000a]. Todavia, esta técnica pode levar a um custo proibitivo em termos de área, como mostrado em [AMO 2003].

Deve-se considerar, também, que devido à diversidade de blocos incorporados em um mesmo CI, como por exemplo, memórias, circuitos reconfiguráveis e circuitos responsáveis pelas interconexões entre blocos, há a necessidade do uso de diferentes

técnicas de auto-teste a fim de se verificar o funcionamento correto de todo o sistema. O que se faz normalmente é adicionar ao *chip* diversos blocos de *hardware*, sendo cada bloco dedicado exclusivamente ao teste de uma parte do circuito. Conseqüentemente, aumenta-se a área em silício necessária para acomodar todo o sistema juntamente com os blocos de teste correspondentes. Logo, ao se integrar diversos blocos de teste a um dispositivo pré-existente, deve-se considerar os custos e limite dos recursos de *hardware* disponíveis. Essa limitação de recursos traz a necessidade da reutilização de módulos. Uma estratégia para a reutilização dos recursos de *hardware* é a reconfiguração, que consiste em alterar a funcionalidade de um circuito durante o seu intervalo operacional, como delineado em [ABR 2002], [STR 97] e [STR 2003].

É interessante observar que se os testes não necessitarem ser executados em paralelo, é possível utilizar os blocos funcionais para o teste de toda a estrutura do dispositivo ao custo de realizar, muitas vezes, a multiplexação dos pontos de observação de teste [STR 98]. Nesta linha, a utilização dos CABs para a geração dos padrões de teste e para a análise das respostas obtidas, como proposto em [AND 2004b], [AND 2005] e [PER 2005], visando o teste da rede de interconexões e células de I/O de FPAAs torna-se atraente no sentido de reduzir o custo do equipamento necessário ao teste deste tipo de dispositivo. Uma estratégia de baixo custo possível para o BIST das chaves e *buffers* das células de I/Os de FPAAs e que não exige a escolha de vetores de teste externos oferecendo alta cobertura de falhas é denominada OBIST (do inglês, *Oscillation Built in Self Test*), descrita em [ARA 96a], [ARA 96b] e [LUB 96].

Além do auto-teste integrado, outras metodologias de teste de FPGAs existentes vêm sendo utilizadas há algum tempo. Dentre aquelas estudadas, pode-se citar: técnicas propostas em [LAK 2000], [MIC 96], [SHN 98] e [TAH 2003], onde se constroem barramentos globais para a aplicação externa de estímulos; técnicas visando ampliar os pontos de observabilidade, explorando, para isto, a própria lógica programável do dispositivo [PER 2005] [TAH 2003]; e, técnicas envolvendo a detecção de falhas de *delay* que funcionam como "espelhos" de falhas de *bridgings* e *opens* existentes na rede de interconexões de dispositivos reconfiguráveis, conforme demonstradas nos trabalhos [CHM 2003], [KRA 2001] e [TAH 2004].

A adaptação ao teste de FPAAs de estratégias de teste de interconexões de FPGAs baseadas na pesquisa em grafos, abordadas em [SUN 2002b] [SUN 2002d] e [SUN 2004], é outro ponto aqui considerado. A modelagem da estrutura de interconexão e células de I/O em grafos nos permite encontrar um número mínimo de configurações de teste que garanta uma boa cobertura de falhas, reutilizando algoritmos conhecidos de pesquisa. Portanto, aproveitando o fato de o FPAA ter uma densidade menor de blocos configuráveis e de roteamento, também se torna atraente um modelamento das suas interconexões e I/Os através de matrizes e grafos.

Além dos problemas de teste mencionados e resolvidos pelas estratégias de teste e modelamento de rede de interconexões de dispositivos programáveis citadas acima, deve-se considerar que, atualmente, estes circuitos apresentam redes de interconexões compostas por barramentos longos que são susceptíveis a defeitos de *crosstalk* (*glitch* e *delay*). Estes defeitos muitas vezes são críticos, pois, podem provocar falhas funcionais e de *timing* que causam a perda da integridade dos sinais quando o circuito opera em freqüências elevadas. Técnicas descritas em [BAI 2003], [CHE 2001], [MET 2001] e [ZHO 98], propõem o modelamento destas falhas e o auto-teste integrado onde somente os estímulos de teste necessários para cobrir as falhas agentes são gerados.

Pretende-se, neste trabalho, priorizar a elaboração de estratégias de teste que visam à detecção de falhas em FPAAs. Porém, após a apresentação de soluções para o teste do dispositivo como um todo, faz-se necessária a elaboração de técnicas que visam o diagnóstico das suas estruturas com defeitos. Técnicas delineadas em [ABR 99], [COT 2000], [HAR 2002], [STR 2001], [STR 2002] e [WEY 1990] que geram diversas configurações BIST para realizar o diagnóstico na rede de interconexões dos FPGAs também foram estudadas, porém, nesta dissertação, não são utilizadas.

Assim, a integração de algumas dessas importantes metodologias desponta como alvo desta dissertação. A partir destes estudos, surge a motivação para o desenvolvimento deste trabalho, que propõe estratégias de teste que visam à detecção de diferentes tipos de falhas na rede local e global de interconexões, bem como, nas células de I/O. Além disto, com estas estratégias pretende-se ter ganhos no tempo de teste e garantir uma boa cobertura de falhas típicas das interconexões do FPAA.

1.1 Metodologia e Organização do Texto

Conforme delineado na Seção anterior, para o desenvolvimento deste trabalho fez-se inicialmente uma revisão bibliográfica sobre o estado da arte do projeto de dispositivos analógicos programáveis e sobre as diversas técnicas utilizadas atualmente para modelar e testar a rede de interconexões em FPGAs. O objetivo do estudo teórico foi descobrir possíveis semelhanças entre as estruturas dos FPGAs e FPAAs, o que ajudaria na adaptação das técnicas de teste dos dispositivos digitais ao caso dos FPAAs.

Deste modo, diversos trabalhos na área de teste de interconexões foram analisados. Nesse sentido, verificou-se a vastidão do tema e, dessa forma, algumas decisões tiveram de ser feitas a priori, a fim de prover uma implementação em tempo hábil para a conclusão deste trabalho.

Este trabalho inicia abordando, no **Capítulo 2**, os diversos tipos de FPAAs produzidos por diferentes fabricantes e que são baseados em tecnologias diversas. Além disto, são apresentadas algumas técnicas que visam à redução dos efeitos parasitas sobre os transistores e interconexões e à ampliação da resposta em freqüência destes dispositivos.

O **Capítulo 3** expõe conceitos de cada uma das estruturas que compõem o dispositivo analógico reconfigurável que serve como veículo de experimentação nos testes, mostrando graficamente os circuitos que constituem a rede local e global de interconexões, e os circuitos que constituem as células de I/O e registradores de memória. Além disto, são dados detalhes sobre a sua programação, escolha dos parâmetros de projeto de alguns CABs e são descritas as filosofias para simulação de falhas nos CABs, alterando-se o seu algoritmo funcional, e para simulação de falhas *stuck-at* nos registradores de memória, alterando-se o *bitstream default* dos blocos programáveis.

O **Capítulo 4** diz respeito às técnicas de modelamento que vêm a contribuir na escolha dos algoritmos de pesquisa em grafos utilizados nos testes da rede de interconexões de FPAAs.

O **Capítulo 5** descreve que a simulação de falhas na rede de interconexões do veículo de experimentação é feita de forma manual, acessando-se diretamente as chaves que se deseja testar e que, portanto, trata-se de um processo simples e rápido. Adicionalmente, neste capítulo são apresentadas as estratégias propostas para o teste da

rede de interconexões do dispositivo sob teste e são descritos os algoritmos que visam detectar falhas nas chaves e linhas desta rede.

Por outro lado, uma técnica denominada OBIST para testar as chaves e os *buffers* das células de I/O também é apresentada nesta parte do trabalho. A fase de validação destas propostas é feita conforme a estratégia de simulações de falhas citada anteriormente.

No **Capítulo 6**, propõe-se uma alternativa de auto-teste integrado que utiliza recursos internos do FPAA para realizar tarefas de geração e análise de teste. Aspectos de projeto do circuito analisador das respostas de teste (ORA) e do gerador de estímulos, tais como repetibilidade, linearidade e sensibilidade, também são tratados nesse Capítulo.

O **Capítulo 7** trata dos defeitos de *crosstalk* e das falhas que são originadas a partir deles. Adicionalmente, os projetos do circuito gerador e analisador de teste e do controlador global responsável pelo sincronismo das tarefas também são delineados neste Capítulo. Para validar a estrutura BIST desenvolvida, as arquiteturas propostas são descritas em uma linguagem de descrição de *hardware*, e são submetidas a simulações de falhas na própria entidade do gerador.

Finalmente, no **Capítulo 8** são delineadas algumas conclusões sobre este trabalho, suas principais contribuições e indicações de trabalhos futuros.

2 DISPOSITIVOS ANALÓGICOS PROGRAMÁVEIS

Os FPGAs são uma excelente solução para a elaboração rápida de projetos, oferecendo alta confiabilidade proporcionando um baixo custo para sistemas lógicos digitais. Tendo em vista que muitos sistemas de processamento de sinais requerem a implementação tanto de circuitos digitais como de circuitos analógicos, hoje já encontramos, em projetos que envolvem circuitos de natureza digital, circuitos programáveis analógicos integrados aos FPGAs. Estes circuitos analógicos programáveis que oferecem aos projetistas flexibilidade de programação e um menor tempo de projeto são denominados de *Field Programmable Analog Arrays* (FPAAs) ou *Totally Reconfigurable Analog Circuits* (TRACs).

Os principais elementos que compõem a arquitetura dos FPAAs são mostrados na Figura 2.1, incluindo um *array* de blocos analógicos configuráveis, uma rede de interconexões entre estes blocos e um *shift register* [ANA 2004] [LEE 95a]. Os CABs possuem parâmetros programáveis bastante flexíveis, possibilitando aos projetistas facilidades no momento de desenvolver circuitos analógicos. A rede de interconexões conecta sinais entre os CABs a fim de realizar as funções requeridas pelo circuito projetado. O *shift register* armazena as configurações dos *bits* que são usadas para o controle apropriado das funções dos CABs e das interconexões entre eles.

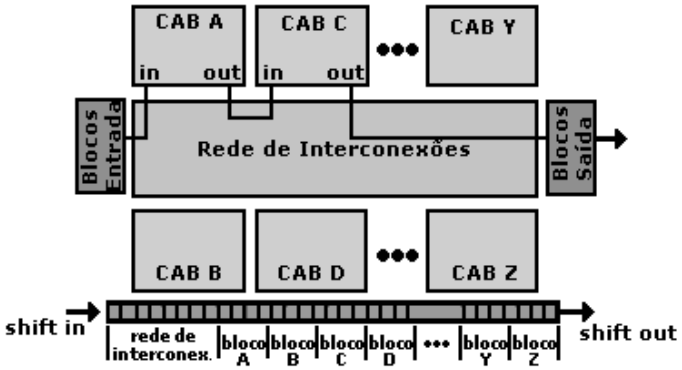


Figura 2.1: Arquitetura de um FPAA genérico.

Circuitos analógicos estão sujeitos a fatores que limita o seu desempenho. Entre eles pode-se citar a existência de ruído, a não linearidade dos seus componentes, os efeitos parasitas e, conseqüentemente, a limitação da resposta em freqüência destes circuitos. Os FPAAs trouxeram um impressionante ganho em termos de facilidade de projeto, na medida em que estes fatores podem ser abstraídos. Mantidas as devidas proporções, o projetista ao desenvolver um sistema complexo deve se preocupar apenas com as questões de comunicação entre os blocos configuráveis, sem considerar detalhes de implementação de cada núcleo que compõe o sistema, facilitando a elaboração de vários projetos em um curto espaço de tempo. As interconexões entre os CABs devem ser

projetadas sem afetar a precisão do circuito e os blocos de funções analógicas devem fornecer uma variedade muito grande de funções programáveis úteis.

Junto com estes dispositivos programáveis desenvolvidos por diferentes fabricantes, são oferecidas ferramentas de CAD de uso público que têm por finalidade traduzir o circuito analógico projetado para um conjunto de blocos, funções e interconexões, configuráveis *bit* a *bit* via interface serial.

Nesta seção são discutidos assuntos referentes ao projeto de circuitos analógicos programáveis, as soluções que podem ser utilizadas a fim de otimizar a sua confiabilidade e reconfigurabilidade e aos avanços obtidos na área de projeto de FPAAs.

2.1 Projeto de Circuitos Analógicos Programáveis

Ao mesmo tempo em que os FPAAs facilitam a elaboração de sistemas de natureza analógica diminuindo o tempo de projeto, trazem alguns problemas referentes as interconexões entre os CABs e à escolha destes blocos. Este novo desafio inclui as diversas configurações possíveis dos vários módulos de FPAAs baseados em tecnologias diversas (capacitor chaveado, tempo contínuo, etc) e a definição da melhor rede de interconexão entre os CABs. Devido a isto, os projetistas necessitam, inicialmente, definir quais são os CABs com maior versatilidade e, posteriormente, como interconectá-los. Nas próximas Seções são descritas algumas técnicas de projeto de blocos analógicos configuráveis e de rede de interconexões de dispositivos analógicos programáveis que correspondem ao estado da arte nesta linha de pesquisa.

2.1.1 Projeto dos Blocos Analógicos Configuráveis (CABs)

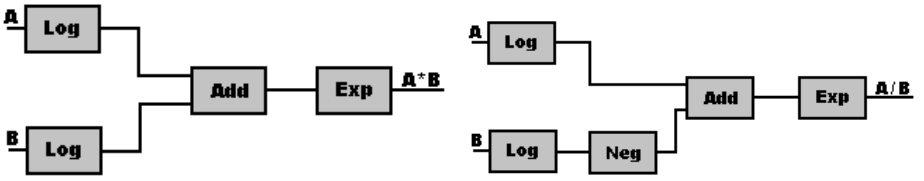
A função e a implementação dos CABs dependem do nível de complexidade em que os blocos do circuito são constituídos. Em geral, pode-se citar como exemplo de CABs de baixa complexidade os transistores, e como exemplo de CABs de alta complexidade os blocos de subsistemas. Cada bloco de subsistema pode ser dividido em macros que facilitam o projeto de sistemas complexos. Estas macros são formadas por sub-circuitos que, por sua vez, são formados por transistores. A Tabela 2.1 descreve os níveis de complexidade dos circuitos analógicos mostrando quais os componentes que constituem cada nível [LEE 95b].

Tabela 2.1: Definição dos níveis dos circuitos analógicos em geral.

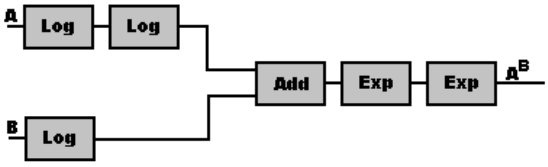
NIVEL	GRANULARIDADE	DESCRIÇÃO	EXEMPLOS
1	Transistor	Transistores	Mosfet, BJT, etc.
2	Sub-circuito	Circuitos simples de função específica composto por poucos transistores.	Espelhos de corrente, estágio diferencial, estágio de saída, etc.
3	Bloco básico	Estágio versátil básico para circuitos complexos.	Opamp, transcondutores, conversores de corrente.
4	Bloco Funcional	Estágio versátil simples com elementos passivos.	Integradores, Samples/ Hold, etc.
5	Subsistemas	Comunicação de muitos estágios versáteis	Filtro *Biquad*, VCO, ADC/DAC, etc.

Entre os níveis mostrados acima, os blocos básicos (do inglês, *building blocks*) correspondem à escolha mais simples para o desenvolvimento de circuitos analógicos programáveis devido à sua alta versatilidade. Sendo assim, muitos projetistas optam em utilizá-los nos seus projetos. Os amplificadores operacionais (opamp) e os transcondutores são os componentes mais utilizados nestes blocos. A escolha dos CABs afeta significativamente o projeto do dispositivo programável, pois, entre outras coisas, determina o tipo de interconexões a utilizar. Se existirem muitas conexões entre CABs de complexidade excessivamente baixa para um determinado tipo de aplicação, a área de interconexão pode ficar imensa e impraticável com a presença de excessivos efeitos parasitas. Recentemente, pesquisadores, juntamente com as indústrias que produzem os FPAAs, deram início a estudos referentes à otimização dos esquemas de interconexões entre os componentes dos *building blocks*.

Além disto, foi elaborada uma nova proposta de configuração dos blocos analógicos programáveis baseada em uma metodologia de projeto computacional para o processamento analógico de sinais [BUX 2000]. A idéia principal desta proposta é utilizar o grupo de operações dos *building blocks* disponíveis no FPAA em substituição a um ou mais CABs versáteis. Esta metodologia de processamento de sinal analógico utiliza cinco operações básicas facilmente implementadas eletronicamente: “ADD”, “NEGATE”, “LOG”, “ANTILOG” e “DIFFERENTIATE”, sendo que, a operação “LOG” é utilizada para realizar funções adicionais como multiplicação, divisão e elevação de potência. A origem matemática destas três operações adicionais é ilustrada na Figura 2.2. Portanto, na prática, muitas operações de processamento analógico de sinais podem ser implementadas utilizando uma combinação de quaisquer das cinco operações básicas, permitindo, por conseguinte, que o projeto do circuito analógico programável possua uma descrição de alto nível do circuito desejado.



(a) Multiplicação (b) Divisão.



(c) Elevação de potência.

Figura 2.2: Operações Aritméticas Adicionais.

2.1.2 Projeto da Rede de Interconexões

O projeto da rede de interconexões de um *chip* é uma tarefa crítica, uma vez que, através desta rede é possível à comunicação entre blocos funcionais vizinhos e entre blocos que estão distantes uns dos outros. Além disto, muitas vezes, elas são exploradas como interface do *chip* com o mundo real; portanto, erros de projeto da rede de interconexões podem causar, dentre outros problemas, defeitos de *crosstalk* e, conseqüentemente, erros no sistema. A Figura 2.3 ilustra a conexão entre o bloco A, correspondente à saída, e o bloco B, correspondente à entrada do sistema, através da rede de interconexão disponível. Pode-se, também, observar através desta figura que os "fios" que realizam as conexões e as chaves responsáveis pela ligação destes "fios" são ativados através de *bits* de configuração vindos de um *shift register*. Estas chaves são geralmente transistores CMOS que possuem capacitâncias e resistências não-lineares e parasitas que afetam a linearidade e a resposta em freqüência do circuito analógico projetado. Por exemplo, em [SIV 88] o circuito integrado reconfigurável baseado em uma arquitetura analógica *neural-network* faz uso de transistores do tipo CMOS para conectar os recursos programáveis tais como os pares diferenciais e os espelhos de corrente. O problema causado pela resistência parasita dos transistores do tipo CMOS é ilustrado através da Figura 2.4. Este problema deve-se à resistência em condução (R_{on}) do transistor CMOS que provoca uma queda de tensão entre a fonte do bloco 1 e a impedância de carga do bloco 2. As impedâncias de saída do bloco 1 e do bloco 2 são finitas e dependem da configuração dos blocos, enquanto que R_{on} depende da tensão, da temperatura e de outras variáveis do processo. Além disto, a tensão de entrada do bloco 2 terá um valor dependente da configuração adotada. A Equação 2.1 proposta em [SIV 88] descreve formalmente o efeito da resistência em condução do transistor CMOS.

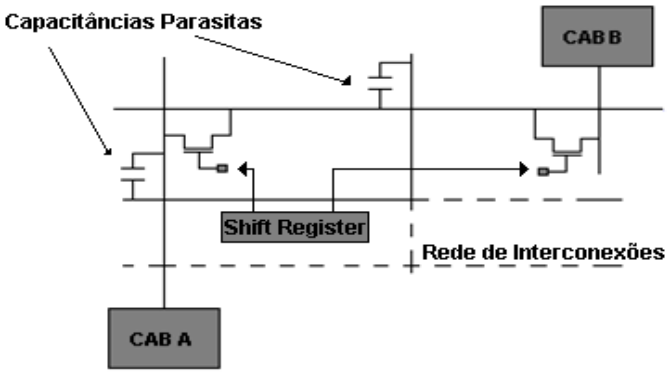


Figura 2.3: Efeitos parasitas devido às interconexões das chaves.

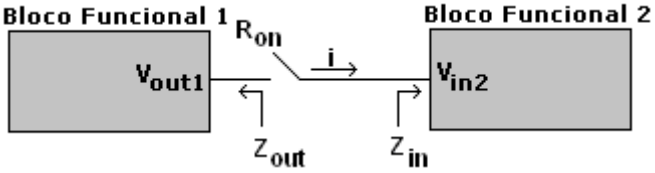


Figura 2.4: Ilustração do problema causado pela resistência parasita dos transistores do tipo CMOS.

$$V_{in2} = V_{out1}\left[Z_{in}/(Z_{out} + R_{on} + Z_{in}\right] \qquad (2.1)$$

Onde:

V_{in2} é a tensão na entrada do bloco 2;

V_{out1} é a tensão na saída do bloco 1;

Z_{in} é a impedância de entrada do bloco 2;

Z_{out} é a impedância de saída do bloco 1 e

R_{on} é a resistência em condução da chave CMOS.

Diversas tecnologias foram desenvolvidas com o intuito de solucionar problemas como o da resistência parasita dos transistores. Entre elas destaca-se a tecnologia baseada em capacitores chaveados (do inglês, *switched capacitor discrete-time*) em que não ocorrem perdas através da resistência dos transistores. Entretanto, esta tecnologia é baseada em amostragem no tempo e, portanto, possui a desvantagem de limitar a resposta em freqüência do sistema, pois a freqüência do sinal deve ser menor do que a freqüência de *clock* para que a operação ocorra corretamente (Teorema de NYQUIST). Em geral, a tecnologia baseada em capacitores chaveados pode ser utilizada em aplicações com limites de freqüência do sinal de vários KHz até cerca de 10MHz. Além disto, outros fatores devem ser considerados em aplicações que utilizem uma largura de faixa superior a 10MHz, como, por exemplo, o remodelamento dos filtros e a elaboração de técnicas contra efeitos indesejáveis, tais como, as distorções de sinais.

Entretanto, se a área de implementação do projeto for levada em consideração, surge mais um problema quando se utiliza a tecnologia baseada em capacitores chaveados, uma vez que um *array* de capacitores programáveis estará presente no circuito. A Figura 2.5 e a Figura 2.6 mostram, respectivamente, a rede de interconexões e o circuito esquemático de um bloco analógico configurável de um FPAA cuja tecnologia é baseada em capacitor chaveado.

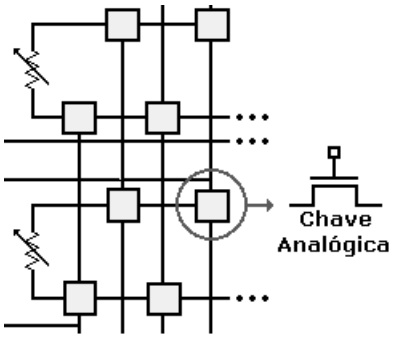


Figura 2.5: Rede de interconexões de um FPAA com tecnologia SC.

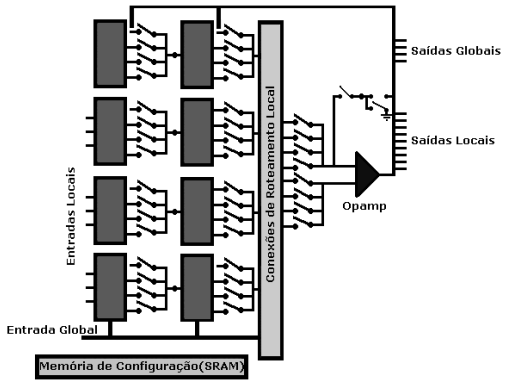


Figura 2.6: Bloco analógico configurável com tecnologia SC.

Outra possibilidade de implementação é baseada em uma tecnologia que utiliza tempo contínuo, onde estão presentes resistores lineares programáveis, um resistor controlador de sinal, um multiplicador de sinal ou uma chave inversora de polaridade. Por operar em tempo contínuo, a resposta em freqüência do sistema é ampla já que seus limites são maiores se comparados aos limites de freqüência empregados na tecnologia baseada em capacitores chaveados. A Figura 2.7 mostra um transistor MOS que é classificado como um transcondutor triodo devido aos transistores operarem na região linear das suas curvas de transferência.

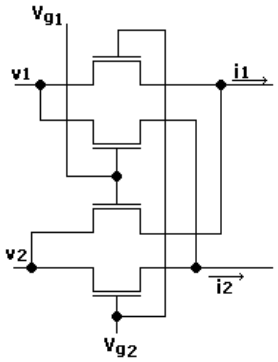


Figura 2.7: Transcondutor MOS.

A transcondutância deste transistor no estado "on" (G_{on}) é a seguinte:

$$G_{on} = (i_1 - i_2)/(v_1 - v_2) = \mu C_{ox}(W/L)(V_{g1} - V_{g2}) \qquad (2.2)$$

onde:

V_{g1} e V_{g2} são as tensões de controle das chaves;

v_1,v_2 min[V_{g1}-V_T, V_{g2}-V_T] e

V_T é a tensão de *threshold* do transistor.

De acordo com a Equação 2.2, G_{on} é diretamente proporcional à (V_{g1}-V_{g2}) e a polaridade da corrente diferencial (i_1-i_2) é determinada pelo sinal de (V_{g1}-V_{g2}). Também, o transcondutor atua como um multiplicador linear para G_{on} variar linearmente com o controle da diferença de tensão entre V_{g1} e V_{g2}. Portanto, ao observar a Equação 2.2 conclui-se que o transcondutor além de atuar como uma chave que controla o fluxo do

sinal analógico na rede de interconexões, atua como uma chave de inversão de polaridade, como um resistor linear variável e como um multiplicador. A Figura 2.8 ilustra como o transcondutor MOS atua como uma chave de interconexão.

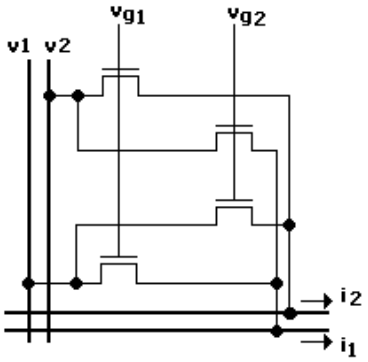


Figura 2.8: Transcondutor MOS atuando como uma chave de interconexão.

Uma vez que os transcondutores podem atuar como chaves nos circuitos FPAAs, o nível de granularidade dos CABs deve ser definido para que isto ocorra. Com esta técnica, os CABs em nível de sub-circuito podem ser utilizados como resistências lineares necessárias para a implementação de circuitos analógicos. Os sub-circuitos CABs incluem circuitos simples como transistores que desempenham funções como: espelho de corrente, estágio de saída e estágio diferencial. A desvantagem da tecnologia baseada em transcondutores é que a tensão sobre seus sub-circuitos é limitada pela linearidade do transcondutor que, por sua vez, é sensível a variações de temperatura.

2.2 Otimização dos Circuitos Analógicos Programáveis

Nesta seção são discutidas algumas técnicas que visam à otimização das interconexões entre os blocos analógicos configuráveis e o aumento da confiabilidade e do desempenho dos circuitos analógicos programáveis.

Recentemente, diferentes técnicas de interconexões entre os CABs e de desenvolvimento de *building blocks* foram elaboradas com o objetivo de ampliar a resposta em freqüência dos circuitos analógicos programáveis, diminuir a área de projeto e, conseqüentemente, diminuir o custo e a probabilidade de surgirem falhas no sistema. Abaixo são descritas algumas técnicas que visam à redução dos efeitos parasitas sobre os transistores e interconexões e à ampliação da resposta em freqüência dos circuitos analógicos programáveis, além de serem descritas novas técnicas de processamento analógico de sinais.

2.2.1 Redução dos Efeitos Parasitas sobre Transistores e Interconexões

No início deste Capítulo conduziu-se uma breve discussão sobre a tecnologia baseada em capacitores chaveados (tempo discreto) e sobre a tecnologia baseada em transcondutores (tempo contínuo), onde são citadas algumas vantagens e desvantagens de cada uma das tecnologias. Baseados nos benefícios de cada uma das tecnologias apresentadas, uma nova tecnologia de fácil compreensão foi elaborada. Dentre os benefícios desta moderna tecnologia de interconexão e configuração, pode-se citar:

1. A estabilidade da tensão obtida através de um circuito apropriado;

2. O sinal de informação é conduzido por tensões em tempo contínuo.

Uma proposta desenvolvida por Looby e Lyden [LOO 2000] chamada de *buffered switching* para eliminar os efeitos parasitas sobre as chaves é delineada a seguir. Na Figura 2.9 são mostrados os dois tipos de *buffered switches* projetadas para que os sinais de entrada e saída tornem-se tolerantes a variações da carga.

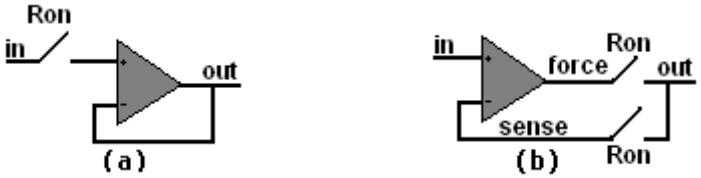


Figura 2.9: (a) *Buffered switch* de entrada (b) *Force-sense buffered* de saída.

Nesta proposta são utilizados transistores, geralmente projetados nas tecnologias CMOS, que apresentam uma alta impedância de entrada devido a um opamp de ganho unitário. Se por um lado este amplificador maximiza os limites de tensão linear e a impedância de entrada da *buffered switch*, por outro, ele minimiza a sua impedância de saída diminuindo a sua sensibilidade à carga. A *buffered switching* ilustrada na Figura 2.9a utiliza opamp e, portanto, o fluxo de corrente nas chaves será eliminado e, com ele, também a principal causa da instabilidade nos nodos internos.

Considerando agora a proposta desenvolvida para os sinais de saída, tem-se que somente um *buffer* é necessário para cada saída ativa. Além disto, se for inserida uma chave no *loop* de retorno do opamp de ganho unitário, o circuito passará a ter uma saída chaveada. A esta chave dá-se o nome de *force-sense buffered switch* e a sua configuração pode ser observada através da Figura 2.9b. A queda de tensão resistiva da *force switch* ocorre no *loop* de retorno e não afeta a função de transferência do sistema. A lógica desta chave permite que a sua saída seja multiplexada por vários outros *buffers* o que contribui para a ampliação da resposta em freqüência do circuito.

2.2.2 Ampliação da Resposta em Freqüência dos FPAAs

Assim como é desejada uma ampla resposta em freqüência para os circuitos de processamento de sinal digitais, para os circuitos analógicos programáveis não é diferente. Atualmente, muitos projetos de FPAAs limitam-se a trabalhar com uma largura de faixa de freqüência abaixo de 1MHz. No entanto, esta largura de faixa de freqüência é insuficiente para aplicações como a de processamento de imagens que requer, normalmente, uma largura de faixa de 10MHz. Desta maneira, a fim de ampliar a resposta em freqüência dos FPAAs, novas arquiteturas devem ser desenvolvidas para evitar a limitação da largura de faixa de freqüência proporcionada pelo desempenho dos opamps que as constituem. Em recentes pesquisas [GAU 97], os conversores de corrente são propostos em substituição aos opamps convencionais dos *building blocks*. Um conversor de corrente é um *building block* do mesmo nível dos opamps convencionais, logo, ele pode ser usado para implementar macros analógicas tais como: integradores, *sample-and-hold*, amplificadores, etc. De acordo com a Figura 2.10, o nodo Y possui impedância "infinita" e, conseqüentemente, o fluxo de corrente sobre ele é pequeno. Se uma tensão é aplicada sobre o nodo Y, a mesma tensão será medida no nodo X uma vez que este se comporta como um curto-circuito virtual. Se uma corrente for injetada sobre X, a mesma corrente será obtida no nodo Z.

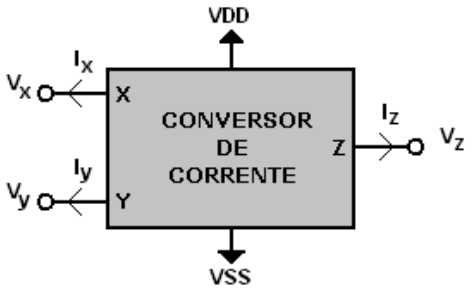


Figura 2.10: Simbologia de um conversor de corrente.

A resposta em freqüência dos circuitos analógicos programáveis pode ser otimizada se a reconfiguração do sistema for implementada utilizando conversores de corrente, pois eles operam basicamente em tempo contínuo. Os conversores de corrente são projetados para atuarem como elementos que ampliam o ganho e a resposta em freqüência do sistema sem utilizar um *loop* de retorno, diferentemente dos opamps convencionais. Além disto, os conversores de corrente podem ser utilizados para construir amplificadores que maximizam a largura de faixa de freqüência dos circuitos analógicos programáveis independentemente do seu ganho. A Figura 2.11 ilustra como um bloco analógico configurável pode ser construído utilizando um conversor de corrente e o circuito esquemático de um conversor de corrente.

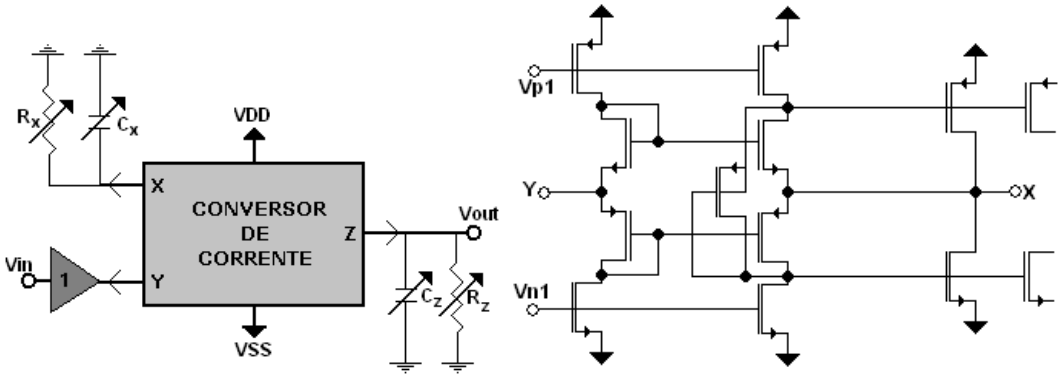


Figura 2.11: (a) CAB constituído por um conversor de corrente (b) Circuito esquemático de um conversor de corrente.

2.3 Pesquisas e Trabalhos Futuros

As pesquisas e os trabalhos futuros na área de circuitos analógicos programáveis visam o aumento do desempenho e da confiabilidade destes circuitos, bem como, das ferramentas de suporte que facilitam o desenvolvimento dos projetos. Além disto, estes estudos trazem grandes avanços referentes à ampliação dos níveis de integração a fim de solucionar problemas que existem nos projetos destes circuitos.

Pesquisas referentes à otimização das técnicas de interconexão e das técnicas que visam à ampliação da resposta em freqüência dos circuitos analógicos programáveis continuarão com a finalidade de eliminar efetivamente os erros observados.

Ferramentas de CAD de fácil uso deverão ser elaboradas visando solucionar problemas de excesso de tempo em projetos de maior complexidade.

Visto que em circuitos integrados analógicos há um número muito grande de dispositivos digitais de controle, avanços devem ser obtidos referentes à diminuição da área de silício utilizada no projeto. Adicionalmente, considerando que os circuitos analógicos atuam como interface dos circuitos digitais com o mundo real, o desenvolvimento destes dispositivos com circuitos programáveis de sinal misto (do inglês, *Mixed-Signal Programmable Circuit*), já integrados, conforme realizado em [GUL 95], será necessário.

3 FPAA ANADIGM AN10E40

Este Capítulo apresenta a descrição detalhada do Anadigm's FPAA AN10E40 que corresponde ao veículo de experimentação deste trabalho. Dentre os assuntos dissertados aqui, pode-se destacar as características da sua arquitetura, o modo de programá-lo e como definir parâmetros para alguns dos seus blocos.

3.1 Aspectos Gerais do Anadigm AN10E40

O dispositivo utilizado no trabalho é o AN10E40 da *Anadigm, the Programmable Analog Company*. Este FPAA utiliza tecnologia baseada em capacitor chaveado com freqüência de amostragem programável. Segundo os manuais do fabricante [ANA 2004] e [PAL 98], a freqüência máxima de relógio suportada pelo dispositivo é 1MHz, o que, de acordo como o Teorema de Nyquist, limita a sua largura da banda de freqüência em 500KHz. Além disto, a escolha da melhor relação de capacitores utilizados nos blocos configuráveis é feita através de algoritmos que simulam suas funcionalidades. Este dispositivo contém 20 CABs distribuídos em um *array* de 4x5 que podem ser conectados uns aos outros, ou a uma das 13 células de I/O disponíveis. As células de I/O contêm *buffers* que podem ser programados para atuar como interface de entrada e saída de sinais. A representação por blocos do AN10E40 é mostrada na Figura 3.1, onde se pode observar os barramentos globais envolvendo os 20 CABs distribuídos no *array*. Esta rede contém barramentos verticais e horizontais organizados em 5 linhas e 6 colunas, cada uma composta de dois "fios". Deve-se referenciar a rede global de interconexões, pois através dela é possível conectar um CAB a qualquer outro no *array* ou a qualquer célula de I/O. Entre cada linha horizontal e vertical existe uma chave denominada *crossover* que realiza a interconexão entre elas.

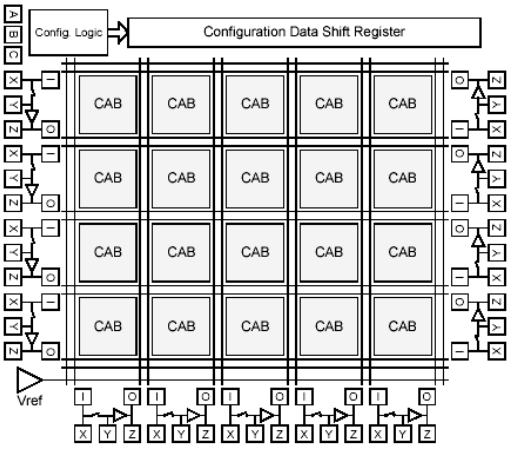


Figura 3.1: Diagrama em blocos da arquitetura do AN10E40.

A ferramenta de CAD disponibilizada pelo fabricante do AN10E40 chama-se *AnadigmDesigner*. Com ela é possível construir circuitos analógicos complexos rapidamente selecionando, posicionando, e ligando blocos funcionais referenciados como *IPmodules*. Logo após o projeto do circuito desejado estar concluído, basta realizar o *download* da sua configuração para o *chip* via porta serial e verificar a funcionalidade do circuito através de qualquer uma das 13 células de I/O disponíveis. Por outro lado, é possível observar os resultados do circuito projetado imediatamente, sem que ele seja programado no *chip*, pois um simulador funcional é incluído na ferramenta para facilitar o projeto e a experimentação sem a necessidade de qualquer equipamento de análise de circuitos externo. Como características deste simulador pode-se destacar a sua interface intuitiva com o usuário e o *display* que mostra os resultados do circuito projetado no domínio do tempo.

Utilizando o *AnadigmDesigner*, também é possível construir múltiplos circuitos independentes, cada um com as suas próprias entradas e saídas. Por exemplo, pode-se ter 3 estágios de ganho, 2 retificadores e um comparador operando totalmente independentes e simultaneamente. A Figura 3.2 e a Figura 3.3 mostram a interface de projeto do AN10E40 disponível na ferramenta de CAD e o *display* mostrando o resultado da simulação de um retificador onda completa, respectivamente.

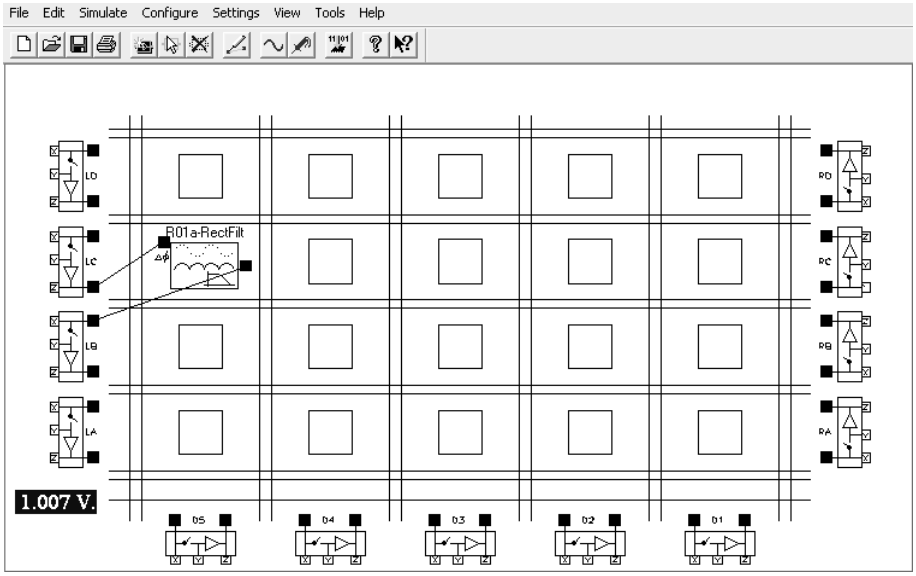


Figura 3.2: Interface de projeto do AN10E40 disponível na ferramenta de CAD.

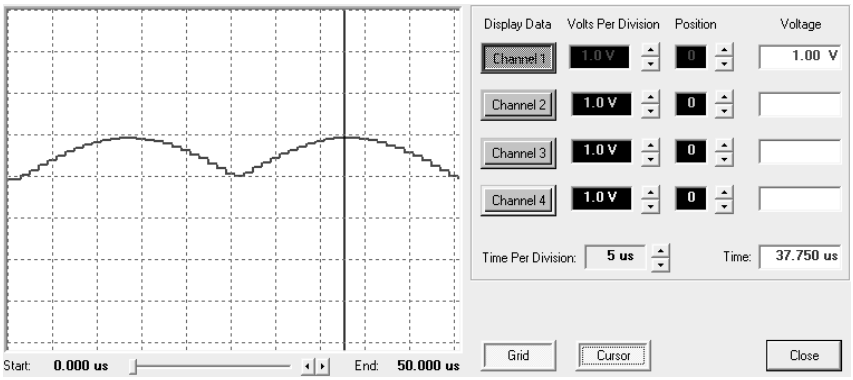


Figura 3.3: *Display* mostrando o resultado da simulação de um retificador onda completa.

3.1.1 Tecnologia de Fabricação do AN10E40

O FPAA AN10E40 possui tecnologia baseada em capacitor chaveado que é utilizada há várias décadas por projetistas de circuitos analógicos. Muitos filtros de alto desempenho atuais utilizam tecnologia baseada em capacitor chaveado. A *Anadigm* combinou topologias que trabalham em tempo discreto com a programabilidade existente em FPGAs e criou o *Field Programmable Analog Array*.

Existem 20 zonas analógicas utilizáveis no *chip*, cada uma contendo um opamp e 5 capacitores de realimentação programáveis. Cada um destes capacitores é constituído por um banco de 255 capacitores que podem ser programados para estarem ligados ou desligados. Os valores dos capacitores são definidos através da habilitação de chaves estáticas, que além de realizarem esta tarefa, fazem com que as interconexões entre o opamp e os capacitores sejam realizadas de diversas maneiras em cada zona analógica. Entretanto, no *chip* também há chaves dinâmicas que realizam a tarefa de tornar o comportamento de capacitores semelhante ao dos resistores e que definem a fase de um sinal na entrada ou saída de um CAB.

A biblioteca de um bloco funcional é composta, dentre outras partes, por comandos que habilitam a conectividade interna dos opamps e capacitores em uma ou mais zonas. O *AnadigmDesigner* manuseia todos os parâmetros sem que o usuário da ferramenta se preocupe com a disposição e com o modo com que os componentes são interligados. Desta forma, o projetista pode desenvolver sistemas analógicos complexos preocupando-se fundamentalmente com a funcionalidade de seus blocos básicos programáveis e com as questões de comunicação entre CABs e destes com o mundo externo. Passa a não mais ser necessário considerar detalhes de implementação de cada bloco que compõe o sistema diminuindo-se, assim, o tempo de projeto.

Além disto, a programação das chaves que interligam os barramentos entre si e estes às células de I/O ou aos CABs é feita de forma manual, utilizando como ferramenta a interface gráfica de programação do AN10E40, sendo, portanto, bastante prática e dinâmica.

3.1.2 Programação do AN10E40

A programação dos dados de configuração para o AN10E40 é realizada via comunicação serial. Durante o *download* do projeto para o dispositivo, várias informações sobre parâmetros escolhidos, interconexões entre CABs que estão em zonas diferentes, escolha da melhor relação de capacitores utilizados feita por um algoritmo que descreve a funcionalidade de cada CAB, dentre outras informações, são programadas nos registradores de memória do dispositivo. Em média, conforme [ANA 2004], a etapa de escrita de uma configuração dura 3 segundos

Abaixo, mostra-se o formato específico de *bitstream* (MOTOROLA S-RECORD FORMAT) que pode ser adquirido durante a etapa de programação do projeto. Além disto, para melhor compreensão, é feita a descrição detalhada de cada uma das suas partes.

Ex:

S224000000138502B7000000000000000000000000000000000000000000000000000000000000000026

S224000020000000000000000000000000000000000000000000000000000000000000000000000000BB

. . .

S20E000360080000080000080000 1462

S9030000FC

Onde:

S2 – Informa que este campo de endereços tem o comprimento de 24 *bits*;

24 – Informa que existem 36(HEX 24) *bytes* a seguir;

000000 – Informa que o endereço inicial deste dado é 0x000000;

138502B7 – Representa o primeiro dado específico do dispositivo, identifica o tipo de dispositivo;

26 – Representa o fim do S-Record check *byte*;

S9 – Informa fim de arquivo;

03 – Informa quantos *bytes* restam no registro;

0000 – Dados não utilizados;

FC – *checkbyte*.

3.1.3 Definição dos Parâmetros de Projeto nos CABs

Na biblioteca de blocos funcionais do AN10E40 existem 50 CABs que realizam funções específicas de processamento analógico. Dentre eles pode-se citar: comparador inversor (C01), comparador não-inversor (C01a), filtro passa-baixas com apenas um pólo (F09), filtro passa-altas *biquad* com alto Q (F04), integrador (I01A), etc. Cada um destes blocos analógicos configuráveis possui parâmetros que podem ser definidos facilmente conforme as necessidades do projetista. Em particular, pode-se citar como exemplo, o estágio inversor de ganho simples (G01), onde apenas um parâmetro de projeto é definido. A interface oferecida pela ferramenta ao usuário que lhe possibilita a definição do ganho e a freqüência de amostragem do sinal é mostrada abaixo:

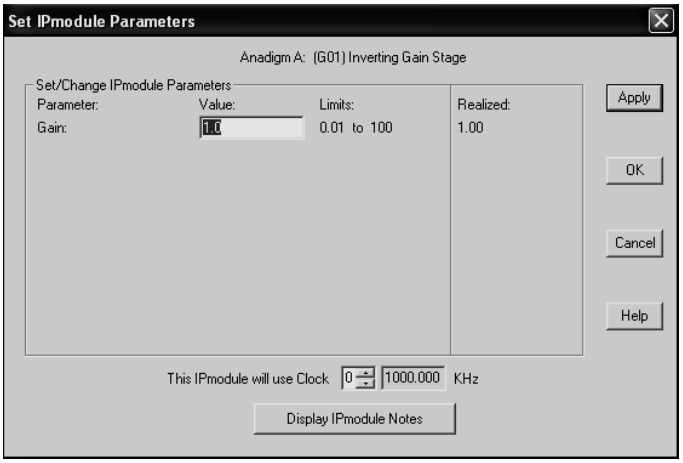


Figura 3.4: Interface que possibilita a escolha do ganho do bloco G01.

Como pode ser visto na Figura 3.4, nesta interface ainda é possível adquirir informações sobre o CAB escolhido, tais como, função de transferência, fase em que os sinais de entrada e saídas são amostrados, etc.

Como informações adicionais, tem-se que cada CAB possui 200 chaves programáveis de um total de 6864 presentes no AN10E40. A programação de uma chave é representada por um ou mais *bits* no *bitstream default* de um bloco analógico programável específico. Aqui, dá-se dois exemplos de representações em hexadecimal (cada caractere representa 4 *bits*) necessárias para expressar um módulo programável.

IPmodule	Bytes												
Simple gain stage	003f	c040	0022	ff24	1000	0ff3	fc00	0018	2270	01c0	0805	2090	8000
Rectifier	003f	c040	0082	ff20	1000	0ff0	0000	0080	2a70	01c0	0000	2090	9500

Figura 3.5: *Bitstream default* de um estágio de ganho simples e de um retificador.

Portanto, qualquer parâmetro de projeto definido, que for diferente dos parâmetros *default*, provocará alteração no *bitstream default* de um bloco funcional específico. Por exemplo, tem-se como parâmetro *default* do estágio inversor de ganho simples "1"; ao ser alterado para "2", pôde-se observar através do arquivo de configuração do projeto que o *bitstream* foi alterado para:

Ex:

CORECELLBITSTREAM: 0 0 003F 8040 0082 7F21 1000 07F3 F800 0040 2A70 01C0 0304 2090 8000

Não é possível definir com precisão quais foram às alterações em *hardware* feitas em conseqüência desta alteração de parâmetro, porém, pode-se fazer uma estimativa em alto nível de quais são as palavras que são alteradas no *bitstream default* possibilitando que simulações de falhas sejam feitas nestes blocos.

3.1.4 Simulação de Falhas

Métodos de teste baseados em simulação de falhas através da alteração do *Bitstream* de dispositivos digitais reprogramáveis já foram apresentados em [ABR 85] e [ALD 2003], porém, nenhum destes métodos visa o teste de dispositivos reprogramáveis analógicos. Nesta Seção, serão propostos métodos de teste baseados em simulação de falhas através da alteração do *Bitstream* ou dos algoritmos que descrevem as funcionalidades dos CABs do FPAA utilizado como veículo de teste.

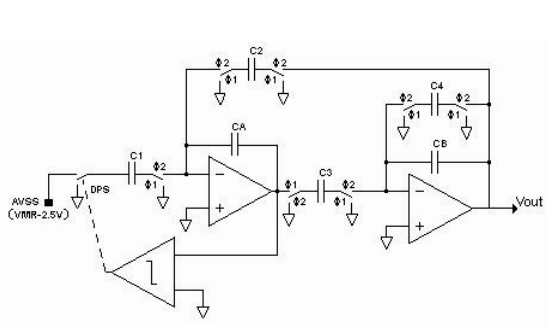
Como mencionado na Seção 3.1.1, este dispositivo possui tecnologia baseada em capacitor chaveado onde a escolha da melhor relação de capacitores utilizados é feita através de algoritmos. As equações que mais se aproximam do comportamento real de um CAB estão contidas em uma biblioteca de arquivos .amm. Além disto, seu simulador é do tipo "intérprete", logo, sua velocidade é menor do que a de um simulador compilado.

Considerando que as funções transferências do oscilador são dadas por:

$$f_{osc} = \frac{f_c}{2\pi}\sqrt{\frac{C_2 \cdot C_3}{C_A \cdot C_B}} \qquad v_p = \frac{300.c_1}{\pi.c_2} \qquad (3.1)$$

onde f_{osc} é a freqüência de oscilação e v_p é a amplitude do sinal gerado pelo oscilador, pode-se simular falhas através da alteração do algoritmo que descreve a funcionalidade deste circuito e observar o quão sensível são a f_{osc} e a v_p a estas falhas. A Figura 3.6 mostra o circuito esquemático do oscilador fornecido pela ferramenta e uma parte do

algoritmo que foi alterado para simular falhas paramétricas de 20% do valor “C2” de um oscilador. Logo após, a Figura 3.7 mostra os resultados práticos de bancada onde foram injetadas falhas de ±10% ao valor de “CB” para diferentes freqüências e de ±20% ao valor de “C2” para diferentes amplitudes escolhidas.




```
while(dCA>0)
{
      dC2 = adjustCap(nC2*dCA);
      dC1 = adjustCap(Glo * dC2);
      aFoa1 = FCdiv2PI*dC2/dCA;
      aGlo = 1.0* dC1/dC2;
      err = aGlo/Glo + Glo/aGlo + aFoa1/Foa + Foa/aFoa1;
      if (err < bestErr)
      {
            bestErr = err;
            C2 = dC2 + 0.2 * dC2;
            C1 = dC1;
            CA = dCA;
      }
      dCA--;
}
```

Figura 3.6: Circuito Esquemático do oscilador fornecido e seu algoritmo alterado (C2 + 20%).

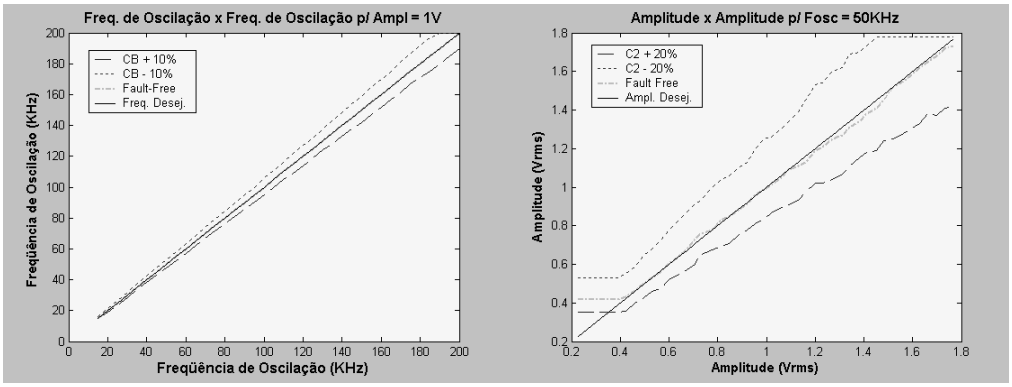


Figura 3.7: Resultados práticos obtidos.

Com isto, pode-se concluir que a amplitude de oscilação é sensível a variações de “C2” e que a freqüência de oscilação é sensível a variações de “CB”, conforme ilustra a Equação 3.1.

Uma maneira de simular falhas do tipo *stuck-at* nos registradores de memória dos FPAAs é através da alteração do *Bitstream Default* de um CAB. Um sub-circuito ao ser programado em um FPAA gera um arquivo de configuração no formato ASCII HEX que será alterado quando alguma falha deste tipo for simulada (conforme descrito em 3.1.2). A metodologia adotada para simular falhas através do *Bitstream* de um CAB foi a seguinte:

1. Alterou-se um dos parâmetros *default* do CAB e verificou-se o efeito desta alteração no arquivo de configuração;
2. Introduziram-se falhas simples e/ou duplas nas palavras que são sensíveis a esta alteração e verificou-se se estas falhas produziam ou não erros na saída.

A Figura 3.8 mostra os resultados práticos obtidos aplicando-se um sinal de entrada de $0{,}5V_p$, quando o *Bitstream* do integrador é alterado, injetando-se uma falha simples sobre o valor hexadecimal “F”, e a sua constante de integração k é igual a 1.

Ex:

```
################### Default Bitstream ##############################
CORECELLBITSTREAM:00003FC0000082FF200000000000000000010A7001C0000000008000
```

B

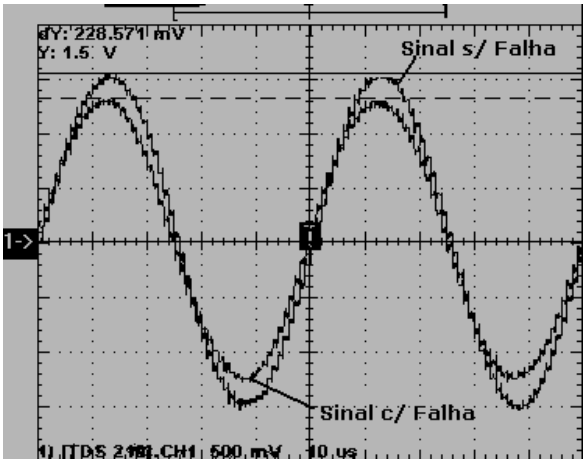


Figura 3.8: Resultados práticos obtidos.

4 MODELOS DE MATRIZ DE INTERCONEXÕES E CHAVES PROGRAMÁVEIS

Como mencionado no Capítulo 1, a escolha de modelos de interconexões adequados facilitam a elaboração de estratégias de teste que garantam alta cobertura de defeitos e baixo tempo de teste. Portanto, para a implementação das funções de modelamento da rede global e local de interconexões, bem como das células de I/O, utilizaram-se estratégias baseadas em grafos de adjacências.

Sendo assim, a seguir são feitas analogias entre as respectivas redes de interconexões de FPGAs e FPAAs, são descritas as regras de interconexões locais e globais do AN10E40, e, finalmente, são mostrados os modelos desenvolvidos que obedecem a estas regras e que as representam.

4.1 Rede Global de Interconexões de Circuitos Programáveis

A rede global de interconexões de FPGAs consiste de um barramento de linhas verticais e horizontais que se intersectam. A conexão entre elas é definida por chaves programáveis, cujo estado é definido pelo *bitstream* gerado pela ferramenta de síntese. Devido à grande densidade de blocos reconfiguráveis em FPGAs, bem como à natureza digital do sinal e à maneira como ele é roteado dentro do circuito, as matrizes de chaveamento possibilitam que várias interconexões globais entre linhas sejam feitas, conforme mostrado em [REN 97], [REN 98a], [REN 98c], [REN 99], [REN 2000a] e [REN 2002] e na Figura 4.1. Estas matrizes permitem que uma linha horizontal conecte-se a uma linha que está na sua mesma direção, ou a um segmento “Ni” ou “Si” de linhas verticais adjacentes, além de permitir que uma linha vertical conecte-se a uma linha que está na sua mesma direção ou a um segmento “Wi” ou “Ei” de linhas horizontais adjacentes.

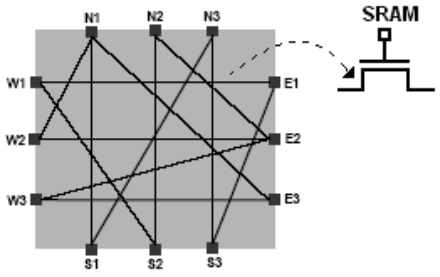


Figura 4.1: Matriz de chaveamento.

Por outro lado, devido à natureza analógica do sinal e à menor densidade de blocos reconfiguráveis, nos FPAAs o roteamento do sinal não se apresenta tão crítico quanto

nos FPGAs. Deste modo, o modelo de interconexões globais de um FPAA segue regras não tão complexas quanto às regras existentes em FPGAs. Isto se deve, em primeiro lugar, ao fato de que uma mesma linha, vertical ou horizontal, não pode ser interrompida, já que a filosofia neste caso é rotear o sinal de uma célula de I/O para um CAB e vice-versa. Assim, não são necessárias chaves para ligá-las ou interrompê-las ao longo do dispositivo, o que significa que existe apenas uma chave a cada interseção entre linhas horizontais e verticais. Abaixo é mostrada, em detalhes, uma parte da rede global de interconexões do circuito analógico programável utilizado como veículo de experimentação.

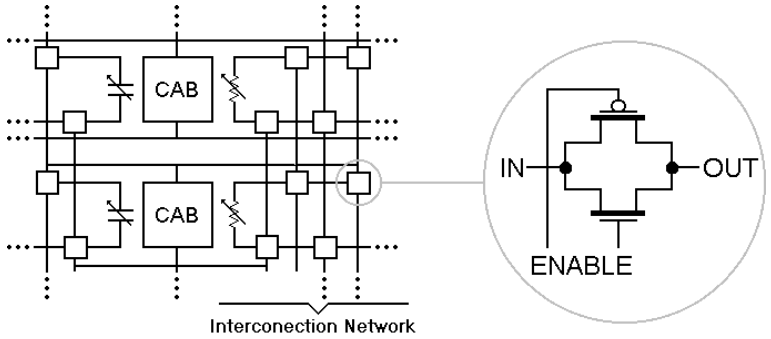


Figura 4.2: Parte da rede global de interconexões do FPAA testado.

Assim sendo, considerando que a rede global de interconexões dos circuitos analógicos programáveis é menos complexa do que a rede do seu dual digital, não há dúvidas que o modelo gerado para este caso é mais simples.

4.1.1 Modelamento da Rede Global de Interconexões de FPAAs

Nesta seção, propõe-se um método baseado na teoria de grafos para modelamento da rede global de interconexões do AN10E40. Entretanto, primeiramente é desenvolvido um modelo matemático através de uma matriz de adjacências que descreve as regras de interconexões globais entre as linhas dos barramentos verticais, horizontais e destas com as células de I/O.

Considere-se Hi, Vi, CHi e CVi as linhas horizontais, verticais, as células de I/O verticais e horizontais do dispositivo, respectivamente, e i $\in$ N. Baseado nas regras citadas na Seção anterior construiu-se um modelo matemático destas interconexões cuja matriz simplificada é mostrada na Tabela 4.1. Como pode ser visto através deste modelo, linhas verticais não podem ser conectadas entre si, o mesmo acontecendo com as horizontais.

Tabela 4.1: Matriz simplificada que representa as interconexões globais.

		H2	**H4**	**H6**	**H8**	**H10**	**H12**	**H14**	**CH1**	**CH2**	**CH3**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**11**	**12**	**13**
V1	**1**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0
V3	**2**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0
V5	**3**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0
V7	**4**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0

Onde:

1. O símbolo "**PS**" (do inglês, *programmable switch*) corresponde às chaves programáveis que conectam as linhas dos barramentos entre si ou estas às células de I/O.
2. "0" significa a inexistência de chaves entre pares específicos.

Ainda neste contexto, com o modelo matemático concluído, um grafo pode ser esboçado, uma vez que a representação em grafo é equivalente a uma representação matricial, onde as adjacências entre os vértices são representadas pelas posições da matriz que possuem chaves analógicas programáveis.

Considerando que cada linha e célula de I/O do dispositivo é mapeada em um vértice e cada chave programável, que liga uma linha horizontal a uma linha vertical ou estas às células de I/O, é mapeada em uma aresta deste grafo, tem-se um modelo correto desta rede e que, conseqüentemente, possibilita a adaptação de algoritmos já conhecidos de pesquisa em grafos ao caso de FPAAs [BAA 93] [COR 2002]. Primeiramente, a idéia era representar a rede global como um grafo direcionado com caminhos duplicados, evitando a geração de caminhos cíclicos. No entanto, devido à complexidade do grafo gerado, esta idéia foi descartada e decidiu-se representá-la através de um grafo não direcionado. Deste modo, neste grafo um vértice *u* que é predecessor de um vértice *v*, também é adjacente deste vértice, conforme abaixo:

$$\pi[v] \leftarrow u \qquad (4.1)$$

$$\pi[u] \leftarrow v \qquad (4.2)$$

onde o símbolo π representa a lista de adjacência dos respectivos vértices.

A Figura 4.3 mostra o grafo resumido que representa as interconexões que podem ser realizadas na rede global de interconexões.

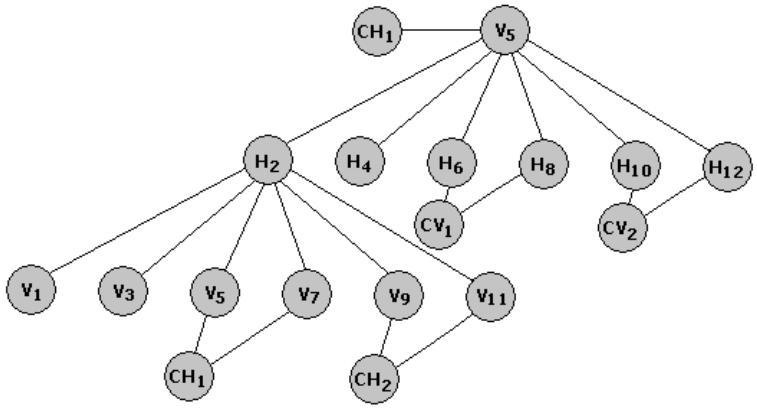


Figura 4.3: Grafo resumido representando parte da rede global de interconexões do AN10E40.

4.2 Rede Local de Interconexões de Circuitos Programáveis

A rede local de interconexões dos dispositivos digitais programáveis comerciais é constituída, basicamente, por multiplexadores, matrizes de chaves e chaves programáveis que possibilitam a comunicação dos CLBs (do inglês, *Configurable Logic*

Blocks) entre si e destes com a rede global de interconexões. A estrutura básica de um *array* com dimensões mxm é descrita em [ALT 2004] e [XIL 2000] e é ilustrada na Figura 4.4. Neste caso, o CLB possui quatro entradas (I1-I4), duas saídas (O1-O2) e pode ser programado para realizar algumas funções lógicas. Existem três tipos de chaves programáveis:

1. Chaves programáveis básicas: são denotadas por caixas quadradas e podem ser programadas para conectar dois segmentos de "fios";

2. Multiplexadores programáveis: são denotados por triângulos sobre linhas sólidas no interior de caixas tracejadas e podem ser programadas para realizar funções que exigem multiplexação de entradas.

3. Chaves programáveis *cross-point*: são denotadas por caixas losangulares no interior das matrizes de chaveamento (SM), compostas por seis chaves programáveis básicas e podem conectar segmentos de "fios" horizontais (Hi) e verticais (Vi), em seis direções diferentes. Estas direções são: Norte-Sul (N, S), Norte-Oeste (N, W), Sul-Oeste (S, W), Norte-Leste (N, E), Sul-Leste (S, E) e Oeste-Leste (W, E). Todas as chaves programáveis são habilitadas através de *bits* de configuração durante a fase de programação do dispositivo.

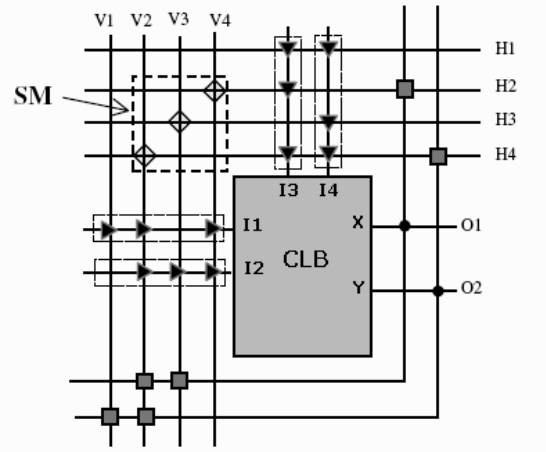


Figura 4.4: Estrutura básica de um *array* com dimensões mxm de um FPGA.

Assim como os FPGAs, os FPAAs possuem uma rede local de interconexões que possibilita a ligação de blocos programáveis entre si e com algumas células de I/O. Se por uma lado a rede global de interconexões não é crítica para os circuitos analógicos programáveis, por outro, a rede local é de suma importância para o desenvolvimento de projetos analógicos. Esta importância deve-se à natureza dos circuitos e sinais analógicos que faz com que os projetistas realizem funções mais complexas a partir do "cascateamento" de funções básicas, explorando com mais freqüência às pequenas distâncias de comunicação entre os módulos, evitando, desta forma, os efeitos parasitas que muitas vezes degradam os sinais provocando erros no sistema.

No caso dos FPAAs, o número de interconexões locais possíveis depende da posição do CAB ou da célula de I/O no *array*. O circuito esquemático abaixo ilustra, dentre outros elementos, as conexões físicas possíveis com a rede local e global de interconexões que um módulo analógico configurável pode realizar.

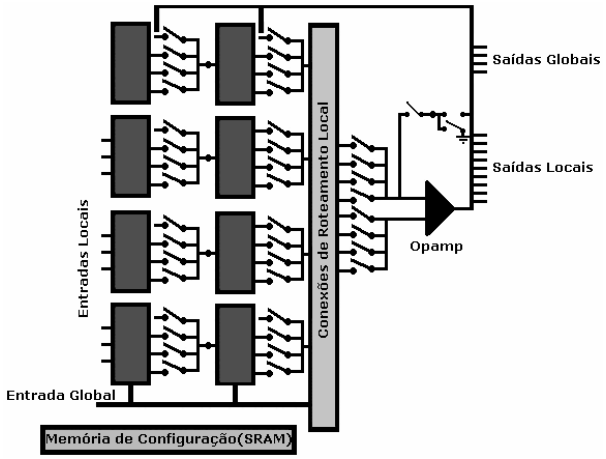
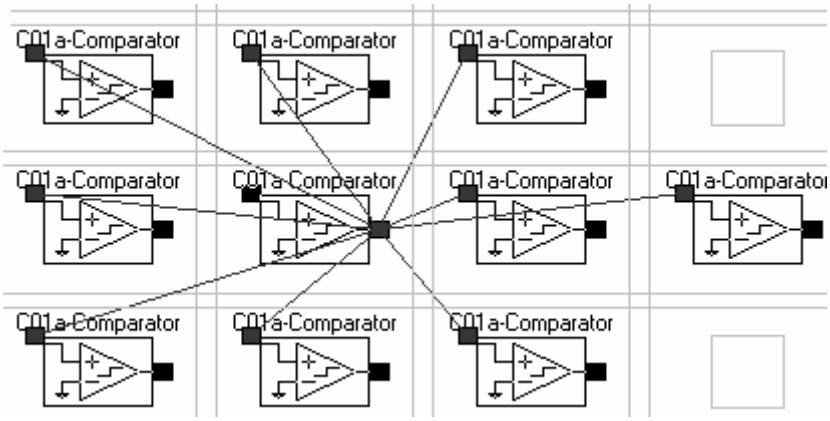
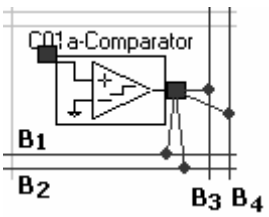


Figura 4.5: Circuito Esquemático do AN10E40's CAB com tecnologia SC.

Como é possível observar a partir desta figura, o número máximo de interconexões locais que um CAB pode fazer com os CABs adjacentes é nove, sendo que ainda são possíveis quatro conexões deste com os barramentos verticais e horizontais. A saída de um CAB pode conectar-se às entradas dos CABs posicionados à sua direita, esquerda, dos CABs posicionados logo acima, dos CABs posicionados logo abaixo e do CAB posicionado duas posições à sua direita. Além disto, os barramentos situados à direita, abaixo e que pertencem à mesma zona de programação deste CAB também são acessíveis. Na Figura 4.6 são ilustradas as interconexões que as saídas de um CAB podem fazer com os CABs vizinhos e com os barramentos, respectivamente.



(a)



(b)

Figura 4.6: (a) e (b) Interconexões que as saídas de um CAB podem fazer com os CABs vizinhos e com os barramentos, respectivamente.

As saídas das células de I/O seguem as mesma regras de interconexões que as saídas dos CABs, porém, é possível realizar, no máximo, quatro ligações locais destas com os CABs e duas com os barramentos conectáveis. Além disto, assim como as saídas dos CABs, as saídas das células correspondem a *jumpers* que possibilitam, dentro do possível, que mais de uma ligação seja realizada simultaneamente. Na Figura 4.7 são apresentadas, respectivamente, as interconexões que as saídas das células de I/O laterais e centrais podem fazer com os CABs vizinhos e com os barramentos conectáveis.

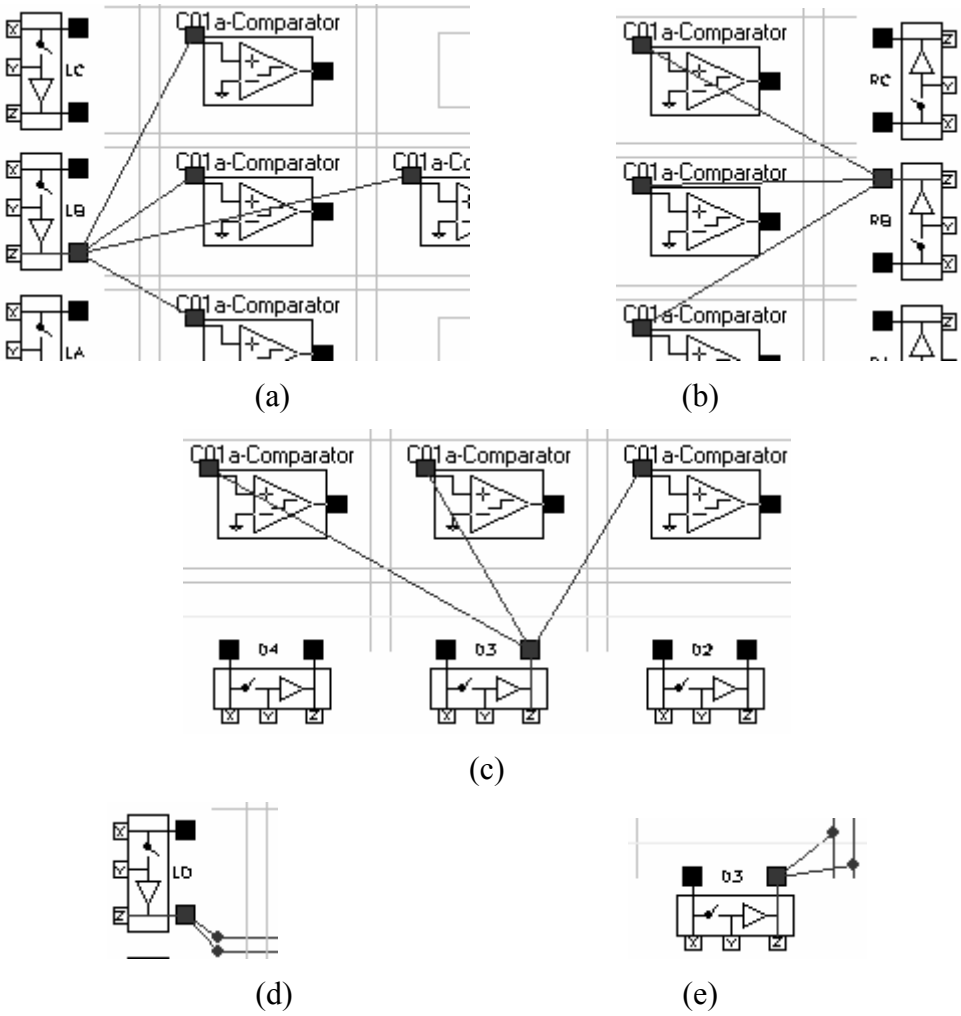


Figura 4.7: (a) (b) e (c) Interconexões que as saídas das células de I/O laterais e centrais podem realizar com os CABs vizinhos, respectivamente. (d) e (e) Interconexões que as saídas das células de I/O laterais e centrais podem realizar com os barramentos, respectivamente.

Através destas figuras é possível observar que o número de interconexões que uma célula de I/O pode realizar também depende da sua posição no *array*. Além disto, é importante salientar que tanto as saídas destas células como as saídas dos CABs não podem ser ligadas ao mesmo instante a um mesmo barramento ou entrada de CAB por razões que serão explicadas a seguir.

Diferentemente das saídas dos CABs que permitem que sejam realizadas mais de um interconexão concomitantemente, suas entradas permitem apenas uma ligação por vez, pois, tratam-se de entradas multiplexadas. Por outro lado, assim como as saídas dos CABs, dependendo da posição do CAB no *array*, também são possíveis, no máximo,

nove interconexões com CABs vizinhos. Abaixo são mostradas, na Figura 4.8, as interconexões que as entradas de um CAB podem realizar com os CABs adjacentes e com os barramentos.

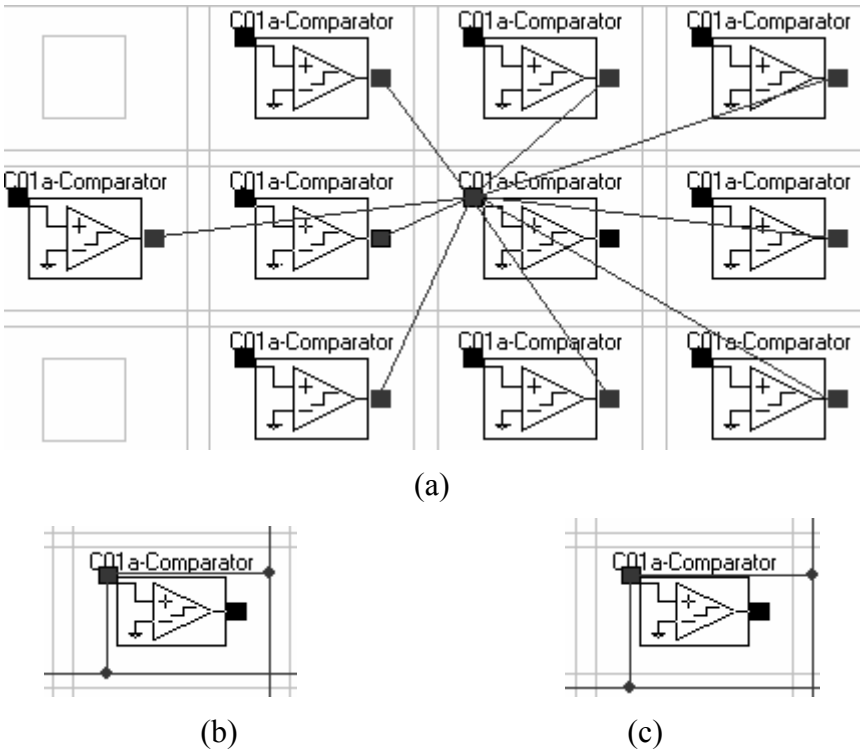


Figura 4.8: (a) Interconexões que as entradas de um CAB podem fazer com os CABs vizinhos. (b) e (c) Interconexões que as entradas de um CAB podem fazer com os barramentos.

Através das figuras 4.8b e 4.8c é possível observar que as interconexões das entradas de um CAB com os barramentos internos pertencentes à mesma zona também dependem da posição deste CAB no *array*.

Da mesma forma que as saídas das células de I/O obedecem às mesmas regras de interconexões das saídas dos CABs, suas entradas obedecem às regras impostas pelas entradas destes blocos. Além disto, para as células de I/O, são possíveis, no máximo, quatro interconexões das suas entradas com os CABs vizinhos e duas ligações com os barramentos conectáveis. Porém, assim como nos CABs, apenas uma conexão por vez pode ser realizada. Na Figura 4.9 são mostradas as interconexões que as entradas das células de I/O laterais e centrais podem realizar com os CABs vizinhos e com os barramentos conectáveis, respectivamente.

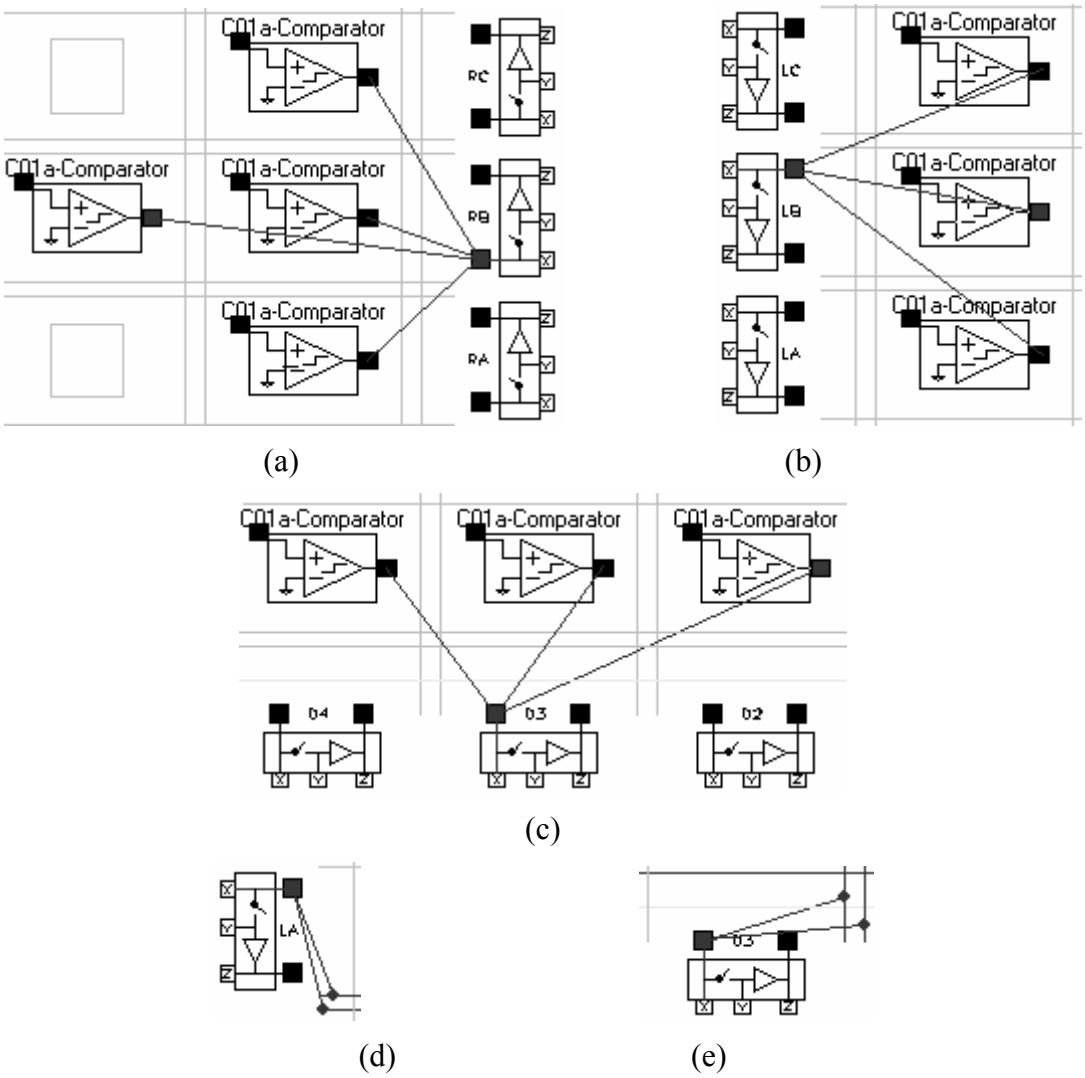


Figura 4.9: (a) (b) e (c) Interconexões que as entradas de uma célula de I/O podem fazer com os CABs vizinhos. (d) e (e) Interconexões que as entradas de uma célula de I/O podem fazer com os barramentos.

Por meio das figuras 4.9d e 4.9e é possível observar que as entradas de uma célula de I/O posicionada na lateral do *array* podem ser conectadas aos barramentos situados logo abaixo e que as entradas de uma célula inferior podem ser conectadas aos barramentos posicionados logo à sua direita, uma vez que estes pertencem à mesma zona programável desta célula.

4.2.1 Modelamento da Rede Local de Interconexões e I/Os de FPAAs

Assim como no caso da rede global, inicialmente foram elaborados modelos matemáticos baseados na teoria de matrizes e, posteriormente, modelos utilizando grafos de adjacências resultantes das matrizes que descrevem as regras de interconexões locais do AN10E40. Uma vez que, as entradas e saídas das células de I/O possuem as mesmas regras de interconexões que os respectivos pinos dos CABs, elaboraram-se modelos idênticos para delinear o comportamento destas ligações.

Deste modo, com o intuito de elaborar um modelo adequado para que a partir dele possa ser elaborado um algoritmo de pesquisa em grafos que satisfaça as exigências de teste da rede local, construiu-se um modelo matemático **X1** das interconexões locais de

um CAB alvo que servirá como ponto de partida para a elaboração do modelo que representa toda a estrutura desta rede. Este modelo e a explicação detalhada sobre cada elemento que o compõe são mostrados abaixo.

Tabela 4.2: Matriz **X1** que representa as interconexões locais de um CAB com seus vizinhos.

		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**B_1**	**B_2**	**B_3**	**B_4**	
A_o	**A_i**	**PS**	0	0	0	0	0	0	0	0	0	0	0	0	
B_o	**B_i**	0	**PS**	0	0	0	0	0	0	0	0	0	0	0	
C_o	**C_i**	0	0	**PS**	0	0	0	0	0	0	0	0	0	0	
D_o	**D_i**	0	0	0	**PS**	0	0	0	0	0	0	0	0	0	
E_o	**E_i**	0	0	0	0	**PS**	0	0	0	0	0	0	0	0	
F_o	**F_i**	0	0	0	0	0	**PS**	0	0	0	0	0	0	0	**=X1**
G_o	**G_i**	0	0	0	0	0	0	**PS**	0	0	0	0	0	0	
H_o	**H_i**	0	0	0	0	0	0	0	**PS**	0	0	0	0	0	
I_o	**I_i**	0	0	0	0	0	0	0	0	**PS**	0	0	0	0	
Gl_o		0	0	0	0	0	0	0	0	0	**PS**	0	0	0	
Gl_o		0	0	0	0	0	0	0	0	0	0	**PS**	0	0	
Gl_o		0	0	0	0	0	0	0	0	0	0	0	**PS**	0	
Gl_o		0	0	0	0	0	0	0	0	0	0	0	0	**PS**	
Gl_i		0	0	0	0	0	0	0	0	0	**PS**	0	**PS**	0	

Onde:

1. O símbolo "**PS**" corresponde às chaves programáveis que conectam o CAB alvo aos CABs vizinhos ou às células de I/O, ou ainda aos barramentos globais;
2. "0" significa a inexistência de chaves entre pares específicos;
3. **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** e **I** correspondem aos CABs posicionados no *array* segundo a Figura 4.10(a), quando as interconexões com as saídas do CAB alvo são analisadas, e, segundo a Figura 4.10(b), para o caso das suas entradas;
4. **A_o**, **B_o**, **C_o**, **D_o**, **E_o**, **F_o**, **G_o**, **H_o**, **I_o** são as saídas locais do CAB alvo (ou as saídas de uma célula de I/O);
5. **A_i**, **B_i**, **C_i**, **D_i**, **E_i**, **F_i**, **G_i**, **H_i**, **I_i** são as entradas locais do CAB alvo (ou as entradas de uma célula de I/O);
6. Os **B's** são os barramentos conectáveis mostrados na Figura 4.6(b), **Gl_o's** são as quatro saídas globais do CAB alvo e **Gl_i** é a entrada global do CAB alvo;

O esquema abaixo mostra graficamente as ligações locais entre o CAB alvo e os CABs que estão nas posições A e B que foram descritas no modelo **X1**.

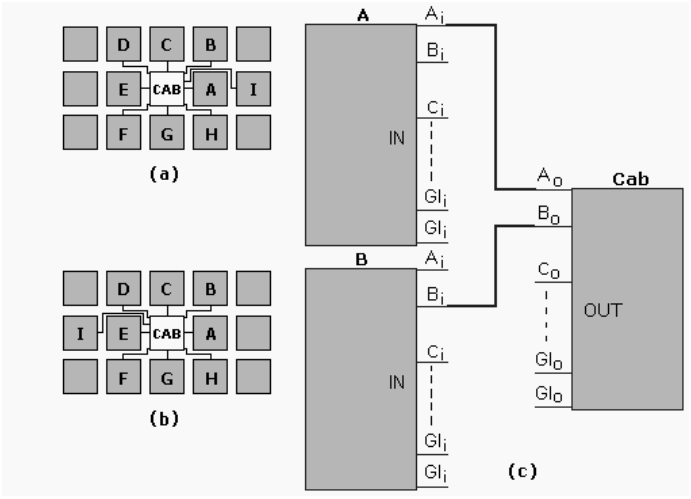


Figura 4.10: (a) e (b) Esquema de interconexões entre as saídas/entradas de um CAB e as entradas/saídas de alguns dos CABs vizinhos, respectivamente (c) Esquema genérico de ligação.

As conexões físicas de saída dos elementos que constituem o AN10E40 apresentam limitações, dentre elas, as cargas nominais dos *drivers* que determinam o número de ligações que os elementos podem realizar. Por outro lado, as conexões de entrada são multiplexadas e, conseqüentemente, permitem uma conexão por vez.

Ainda nesta linha, com a definição do modelo matemático das interconexões de um CAB alvo, conclui-se que uma matriz de adjacências **X=X1**+**X2**+**X3**... pode ser gerada a partir da combinação de matrizes semelhantes à ilustrada na Tabela 4.2 que representam as interconexões de cada um dos CABs e células de I/O do *array*. Neste caso, **X1**+**X2**+**X3**... representa a combinação dos modelos matemáticos do CAB1, CAB2, CAB3 e assim por diante. Logo, seguindo este raciocínio, elaborou-se o modelo matemático **X** da rede local de interconexões do AN10E40 cuja versão simplificada é mostrada na Tabela 4.3.

Tabela 4.3: Versão simplificada do modelo matemático **X** da rede local de interconexões do AN10E40.

			A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**B1**	**B2**	**B3**	**B4**	
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
OUT1	**IN1**	**1**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT2	**IN2**	**2**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT3	**IN3**	**3**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT4	**IN4**	**4**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT5	**IN5**	**5**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT6	**IN6**	**6**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT7	**IN7**	**7**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
OUT8	**IN8**	**8**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	**=X**
OUT9	**IN9**	**9**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	**PS**	0	0	0	0	
Glo1		**10**	0	0	0	0	0	0	0	0	0	**PS**	**PS**	**PS**	**PS**	
Glo2		**11**	0	0	0	0	0	0	0	0	0	**PS**	**PS**	**PS**	**PS**	
Glo3		**12**	0	0	0	0	0	0	0	0	0	**PS**	**PS**	**PS**	**PS**	
Glo4		**13**	0	0	0	0	0	0	0	0	0	**PS**	**PS**	**PS**	**PS**	
Glo5		**14**	0	0	0	0	0	0	0	0	0	**PS**	**PS**	**PS**	**PS**	
Gli1		**15**	0	0	0	0	0	0	0	0	0	**PS**	0	**PS**	0	
Gli2		**16**	0	0	0	0	0	0	0	0	0	0	**PS**	0	**PS**	
Gli3		**17**	0	0	0	0	0	0	0	0	0	**PS**	0	**PS**	0	

Onde:

1. **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H** e **I** correspondem aos CABs posicionados no *array* segundo a Figura 4.10(a), quando as interconexões com as saídas dos CABs são analisadas, e segundo a Figura 4.10(b), para o caso das suas entradas;
2. $\mathbf{OUT_n}$ são matrizes que representam as saídas locais dos "n" CABs do *array*;
3. $\mathbf{IN_n}$ são matrizes que representam as entradas locais dos "n" CABs do *array*;
4. **B's** representam os barramentos dispostos em relação a cada CAB do *array* conforme a Figura 4.6(b), $\mathbf{Gl_{on}}$ e $\mathbf{Gl_{in}}$ são as saídas e as entradas globais dos "n" CABs do *array*, respectivamente.

A partir desta matriz de adjacências **X**, é possível gerar grafos de adjacências disjuntos **G(X)** que representam a rede local de interconexões do AN10E40 e que são utilizados nas pesquisas por caminhos elétricos nos algoritmos elaborados nesta dissertação. A versão simplificada deste grafo **G(X)** é mostrada abaixo:

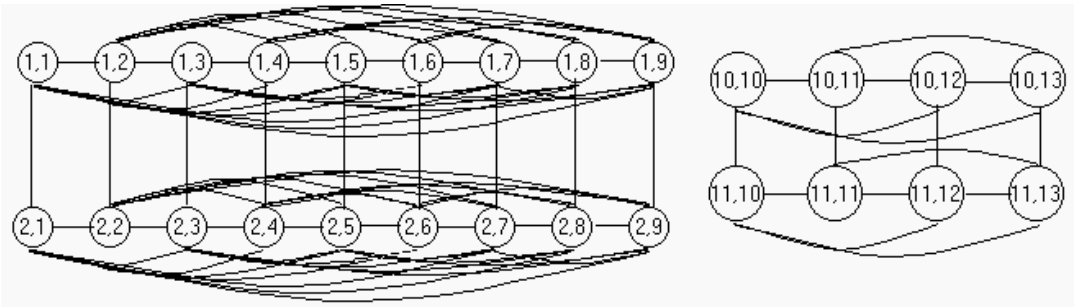


Figura 4.11: Versão simplificada dos Grafos de adjacências disjuntos **G(X)** representando as interconexões locais dos CABs entre si e destes com os barramentos.

Nestes grafos, os pares contidos em cada vértice correspondem aos elementos de conexão "**PS**" citados no seu modelo matemático de interconexões locais.

Vale lembrar que, embora estes grafos sejam disjuntos, eles devem ser interpretados conjuntamente devido às dependências e às limitações das ligações de entrada e saída.

5 TESTE DA REDE DE INTERCONEXÕES E I/OS DE FPAAS

Neste Capítulo são propostas estratégias de teste que visam detectar falhas nas chaves e linhas da matriz de interconexões e I/Os de FPAAs. Como todas as partes destes dispositivos apresentam características estruturais e funcionais distintas, adotou-se a estratégia "Dividir para Conquistar" a fim de testá-las separadamente. Muitos trabalhos anteriores que descrevem o teste de FPGAs, tais como [REN 96], [REN 98b], [REN 2000b], [REN 2000c], [HAR 2000a], [SOU 2002] e [SUN 2004], utilizam esta estratégia que considera modelos de falhas, configurações e estímulos de teste específicos para cada uma das partes a serem testadas, assegurando, portanto, uma boa taxa de detecção de defeitos. O teste de CABs já foi abordado em [BAL 2004a], [BAL 2004b] e [SOU 2002], assim sendo, para que o teste se torne completo, é necessário discutir como testar as demais partes do FPAA tais como, células de I/O e a rede global e local de interconexões.

Como foi explicado na Seção 4.1.1, devido a menor complexidade da rede global de interconexões do dispositivo alvo, que apresenta poucos barramentos não segmentados, pode-se afirmar, a priori, que o procedimento de teste desta rede é mais rápido do que o teste da rede global de FPGAs. Além disto, como conseqüência da pouca área ocupada para roteamento de sinais em FPAAs, a probabilidade de ocorrer alguma falha na sua rede global é menor. De fato, pode-se dizer, sem exagero, que a rede global de interconexões de um FPGA tem uma importância maior para este dispositivo do que para um FPAA.

Por outro lado, embora o número de blocos programáveis seja bem menor em um FPAA se comparado ao do seu equivalente digital, em conseqüência das dimensões do *array* deste dispositivo, pode-se afirmar que o número de interconexões locais que um CAB pode realizar é maior e que elas têm grande importância para os circuitos analógicos programáveis. Logo, é de suma importância que a estratégia adotada para testá-la seja confiável e que, conseqüentemente, garantam-se boas coberturas de falhas.

Por fim, para testar os elementos complementares deste dispositivo, ou seja, as células de I/O, utilizam-se aqui três abordagens distintas:

- Como será visto nas próximas seções deste Capítulo, as interconexões locais que as células de I/O podem realizar seguem as mesmas regras das interconexões entre os CABs e, portanto, a estratégia adotada para testá-las é a mesma.

- No entanto, o teste das interconexões que as células de I/O podem realizar com os barramentos é feito junto com o teste das chaves da rede global.

- No teste dos *buffers* e das chaves destas células, adotou-se uma terceira estratégia baseada em osciladores onde as respostas de teste destes elementos são avaliadas externamente.

Assim sendo, este Capítulo inicia apresentando a metodologia que foi adotada para a elaboração das estratégias de teste visando-se à independência de sinais do teste que é suportada pelos modelos de grafos que representam suas interconexões. Logo após, são apresentados os modelos de falhas que são considerados neste trabalho. Em seguida, fazem-se descrições detalhadas de cada um dos algoritmos de teste propostos para cada uma das partes testadas: redes global e local de interconexões, bem com, células de I/O. O resultados obtidos na etapa de validação das estratégias também são apresentados nestas descrições.

5.1 Metodologia de Teste

Para o desenvolvimento deste trabalho, fez-se inicialmente o modelamento da rede global e local (equivalente ao modelo de I/Os) através de grafos de adjacências, como mostrado no Capítulo 4. O objetivo deste procedimento foi verificar quais os requisitos, em termos de estruturas de interconexões necessárias para a implementação de tais estratégias.

Além disto, é importante salientar que nesta Seção é tratado apenas o teste onde a análise de respostas de teste e a aplicação de estímulos de teste são feitos externamente e que, assim como os grafos, garantem a independência destes estímulos.

5.2 Modelo de Falhas e Estímulos de Teste

A rigor, deve-se considerar que as falhas estruturais na rede de interconexões ocorrem nas linhas ou nas chaves analógicas de programação. Para cada um destes elementos, têm-se modelos de falhas semelhantes aos aplicados em [HAR 2000b], [MIC 96], [STR 97], [SUN 2000a] e [YIN 98] relativos ao teste de interconexões em FPGAs. A seguir, são delineados os modelos de falhas definidos para cada um dos componentes da rede de interconexões de FPAAs:

Linhas: dificilmente ocorrem falhas de curto permanente (do inglês, *short*), já que, originalmente, todas as linhas verticais e horizontais são contínuas. Contudo, elas podem estar abertas, caracterizando falhas de circuito aberto permanente (do inglês, *open*). Além disso, duas linhas próximas podem apresentar falhas de *bridging* (curto entre as linhas). Um *bridging* entre uma linha vertical e uma horizontal é equivalente a uma falha do tipo *stuck-on* na chave que conecta estas duas linhas. Entretanto, pode haver um curto entre duas linhas verticais (ou horizontais) adjacentes que é correspondente a uma falha dupla do tipo *stuck-on* em duas chaves e que, portanto, não é tratado explicitamente nesta dissertação, pois, trabalha-se somente com a hipótese de falhas simples nos barramentos e chaves da rede. Além destas, como se tratam de barramentos analógicos, outros tipos de falhas podem ocorrer, como por exemplo: *stuck-at-Vdd* e *stuck-at-Gnd*, que são equivalentes a um *bridging* entre uma linha de dados e as linhas de alimentação e/ou terra, respectivamente. Dependendo da disposição das linhas de alimentação e terra no *array*, estas últimas falhas podem ser consideradas improváveis.

Chaves: em geral, as falhas que ocorrem nas chaves programáveis são do tipo *stuck-on* ou *stuck-open*. Uma falha *stuck-on* indica que a chave está permanentemente ligada, conectando linhas de sinal mesmo que elas não estejam programadas para realizar esta tarefa. Analogamente, uma falha *stuck-open* indica que a conexão entre linhas através de uma chave não ocorre, uma vez que, esta chave está sempre aberta independente do valor do *bit* de configuração.

É importante observar que, devido às equivalências de falhas citadas anteriormente, uma estratégia que cobre uma falha, cobre, também, a outra que é equivalente. Portanto, é possível definir estratégias de teste que garantam coberturas amplas de defeitos utilizando poucas configurações. Neste caso, as estratégias desenvolvidas adotam os seguintes critérios:

Uma falha do tipo *stuck-open* em uma chave é equivalente a uma falha do tipo *open* nas linhas adjacentes à chave. Isso pode ser facilmente verificado aplicando-se um sinal em um segmento de caminho de teste e programando-se uma das chaves que o liga a qualquer outro segmento para o estado "on".

Uma falha de *stuck-on* em uma chave é equivalente a um curto entre duas linhas adjacentes, sendo que na rede global uma delas é vertical e a outra é horizontal. Para o caso desta rede, entre duas linhas adjacentes verticais (ou horizontais), pode haver também uma falha de curto, ou de *bridging*. Neste caso, ela é equivalente a uma falha do tipo *stuck-on* em duas chaves, que conectam estas linhas a uma mesma linha ortogonal, e por isso essa situação não é considerada explicitamente. Entretanto, caso uma das chaves entre as linhas em questão estiver programada para estar ligada, então há equivalência entre a falha de *bridging* e a falha *stuck-on* na chave adjacente.

A fim de estimular estas falhas e, considerando que as linhas e chaves conduzirão sinais análogos durante a aplicação, optou-se por estímulos dinâmicos no teste da rede de interconexões de FPAAs. Com estímulos estáticos seria impossível analisar, por exemplo, os efeitos parasitas que levam a um comportamento fora da tolerância definida para estas interconexões. Assim, pode-se escolher estímulos onde a freqüência fundamental predomina ou sinais que apresentam grandes conteúdos harmônicos, como por exemplo, um trem de pulsos. A aplicação deste último é mais apropriada para a avaliação do comportamento dinâmico, pois este tipo de sinal permite que os tempos de transição sejam aferidos na saída quando são realizadas variações no número de chaves no caminho do sinal ou variações na carga da linha devido às outras linhas ou a CABs a ela conectados. Portanto, o efeito de cargas capacitivas muito grandes em linhas ou chaves da rede pode ser detectado. Por exemplo, se uma linha que é estimulada por um trem de pulsos apresenta um tempo de transição maior do que o tempo de assinatura livre de falhas obtida previamente, conclui-se que esta linha possui capacitância acima da especificada. Além disto, falhas paramétricas nas chaves, nas linhas da rede de interconexões e nos *buffers* e chaves das células de I/O são implicitamente cobertas aplicando-se estímulos dinâmicos nos caminhos de teste gerados.

Finalmente, após a definição do modelo de falhas e dos estímulos de teste adotados, durante o procedimento de teste, explora-se a possibilidade de programação manual das chaves da rede global e local, bem como dos I/Os, para simulação de falhas através da configuração das chaves no estado oposto ao da falha que se deseja detectar. Como já foi mencionada no Capítulo 3, a programação manual é feita via interface gráfica de programação do dispositivo, garantindo, portanto, o dinamismo e a praticidade do teste.

5.3 Teste da Rede Global de Interconexões

Como foi delineada na Seção 4.1.1, a rede global de interconexões é composta por barramentos longos e chaves analógicas programáveis que possibilitam a comunicação dos módulos programáveis entre si e destes com células de I/O que se encontram em posições onde ligações não podem ser realizadas diretamente. Portanto, para que o projeto analógico apresente níveis de confiabilidade que vão ao encontro dos limites de tolerância impostos, é de grande importância que esta rede seja testada adequadamente a fim de detectar problemas em decorrência de falhas estruturais nas chaves e barramentos que a constitui.

5.3.1 Proposta de Algoritmos para Teste da Rede Global de Interconexões

Analogamente ao que foi feito em [SUN 2002b] e [SUN 2004] visando testar a rede global de FPGAs, uma estratégia de teste baseada na teoria de grafos e na seleção de caminhos foi desenvolvida para a rede global de interconexões do FPAA. Basicamente, esta estratégia consiste em realizar pesquisa gulosa por caminhos elétricos no grafo que representa a rede global de interconexões. O procedimento de teste da rede global é abordado de dois modos distintos: um focado em falhas do tipo *stuck-open* e *open* nas chaves e linhas, respectivamente, e outro focado nas falhas de *stuck-on* e *bridging* destes componentes.

5.3.1.1 Teste e Detecção de Falhas do Tipo Stuck-open e Open

No presente problema, falhas do tipo *stuck-open* e *open* são consideradas como equivalentes, por isso, a estratégia de teste aqui apresentada se limita a realizar somente a detecção destas falhas e não o seu diagnóstico. Para a detecção de *stuck-open* nas chaves que interconectam os barramentos globais, são propostos algoritmos recursivos que têm por finalidade encontrar os caminhos que englobam o maior número de chaves a serem testadas (ou arestas no grafo), diminuindo, conseqüentemente, o número de configurações de teste (CTs) necessárias para garantir uma alta cobertura deste tipo de falha.

Por fim, o modelo que representa a rede global de interconexões citado na Seção 4.1.1.1 serve como base para a elaboração destes algoritmos, pois a partir dele podem ser definidas regras que vão ao encontro das exigências de teste.

5.3.1.1.1 Geração de Caminhos Não Bufferizados

Neste caso, os caminhos críticos gerados possuem apenas um ponto de observação da resposta de teste, correspondente a uma das saídas primárias, e um ponto de controle do estímulo de teste, correspondente a uma das entradas primárias, conforme ilustra a Figura 5.1.

Figura 5.1: Ilustração de um caminho não bufferizado.

Conhecendo-se o tipo de caminho que se deseja selecionar, já é possível definir algumas regras impostas por algumas particularidades da rede que devem ser seguidas a fim de que o algoritmo proposto execute corretamente as pesquisas por caminhos não

bufferizados. Uma destas regras está relacionada aos vértices que representam os pinos de saída das células de I/O e que determina que seja criado um vértice virtual (VERTEX_VIRTUAL) ligado a cada uma destas células para indicar que, durante a pesquisa, em um determinado momento está sendo visitado um vértice correspondente a uma saída. Outra exigência é a criação de uma saída virtual (OUTPUT_VIRTUAL) conectada aos vértices que representam as linhas do barramento que não têm acesso direto às células de I/O. Com estes artifícios, garante-se a continuidade da pesquisa por outros vértices disponíveis ou, caso contrário, garante-se o desfecho da pesquisa.

Além destas imposições, derivam-se duas regras gerais resultantes da atribuição "visitado" a cada vértice inserido no caminho e da atribuição "explorada" a cada aresta envolvida na pesquisa.

Os vértices visitados em uma determinada configuração de teste, ou seja, em uma determinada pesquisa por um caminho que envolva o maior número de arestas possível, não são visitados novamente nesta configuração. Com isto, evita-se a geração de *loops* no algoritmo.

As arestas que já foram exploradas, independentemente da configuração de teste ou do número de caminhos críticos já encontrados, não são exploradas novamente. Com isto, evita-se que chaves que já foram testadas sejam testadas novamente.

A rigor, após serem definidas as principais condições de teste que devem ser seguidas durante a execução do algoritmo, pode-se citar ainda como outros artifícios de projeto o fato de que os vértices virtuais e os de saídas virtuais nunca são considerados visitados, e as arestas que correspondem às conexões entre os vértices e o vértice virtual nunca são consideradas exploradas.

Com isto, pode-se afirmar que a heurística adotada visa encontrar a solução ótima selecionando todos os caminhos que englobam o maior número de chaves e assegura, conseqüentemente, o menor número de configurações de teste. Além disto, é importante mencionar que o algoritmo utilizado é um algoritmo de coloração de grafos citado em algumas bibliografias, tais como [BAA 93] e [COR 2002].

Portanto, o pseudocódigo que gera caminhos não bufferizados e que estima o número de configurações de testes necessárias para cobrir todas as falhas *stuck-open* e *open* no *array* de interconexões globais do AN10E40 é mostrado abaixo:

a) Etapa de Inicialização.

for cada vértice $u \in V[G]$
color[u] ←"white";
$\pi[u]$ ←NIL; **- Lista de adjacência do vértice *u***;
Expl[ ] ←NIL; **- Lista que contém todas as arestas exploradas**;
Vertice[ ] ←NIL; **- Lista que contém todos os vértices visitados**.

b) Etapa de Mapeamento.

for cada vértice $u \in V[G]$
    for cada $v \in \pi[u]$
    π [u] ← v;

c) Etapa de Pesquisa, Seleção de Caminhos Críticos e Finalização.

```
repeat
        Vertice[ ] ← NIL;
        Search_New_Vertex(source);   --Onde "source" é a origem do caminho
until all (u,v) ∈ Expl[ ];

Search_New_Vertex(u)
        for cada vértice u ∈ V[G]
                for cada v ∈ π [u]
                        if (v != VERTEX_VIRTUAL) then
                                if (v != OUTPUT_VIRTUAL) then
                                        if (((u,v) ∉ Expl[ ]) && (v ∉ Vertice[ ])) then
                                                Expl[ ] ← (u,v);
                                                Select_Path&&Coloring( );
                                        Search_New_Vertex(u);
                                else
                                        if (Search_Other_Vertex(u) = FALSE) then
                                                Backtracking( );
                                        Search_New_Vertex(u);
                        else
                                if (Search_Other_Vertex(u) = FALSE) then
                                        Search_New_Vertex(u);
                                else
                                        return;

Select_Path&&Coloring( )
        Path_critical[ ] ← v;
        Vertice[ ] ← v;
        color[v] ← "gray";
```

Como pode se observar, primeiramente, a inicialização das listas é realizada. Logo após, é realizado o mapeamento dos elementos do *array* no grafo através da definição das respectivas listas de adjacências dos vértices, que seguem as regras de interconexões globais do dispositivo, conforme delineadas na Seção 4.1.1.1. Por último, é desenvolvido o processo de pesquisa e seleção dos caminhos críticos. A idéia principal deste algoritmo é encontrar vértices que não foram visitados ($v \notin$ Vertice[ ]) e arestas que ainda não foram exploradas ($(u,v) \notin$ Expl[ ]). Estas são, portanto, as principais condições de teste. Cada vez que estas condições são satisfeitas, um vértice destino v é introduzido no caminho (**Select_Path&&Coloring( )**) e a aresta correspondente é introduzida na lista de controle. Ainda neste contexto, cada vez que um vértice v correspondente a uma célula de I/O for visitado, uma pesquisa por um outro vértice diferente é desenvolvida (**Search_Other_Vertex(*u*)**). Porém, se não houver outro vértice que não tenha sido visitado, uma CT é concluída. Outra condição de teste é definida quando o vértice v visitado foi mapeado em um barramento que não tem acesso direto às células de I/O. Neste caso, também é realizada uma nova pesquisa para encontrar outro vértice destino. Porém, caso ele não exista, a pesquisa continua, sendo que agora, o vértice origem é o vértice u anterior (**Backtracking( )**) e o vértice u atual é definido como visitado. O processo continua até que todas as arestas sejam exploradas.

Este algoritmo foi descrito na linguagem de programação C em quatro versões diferentes a fim de validar o trabalho. Nelas executaram-se pesquisas aleatórias e seqüências com predição de destino e, conseqüentemente, pôde-se avaliar o número de

CTs geradas em cada caso e o tempo de execução destes algoritmos que são mostrados na Tabela 5.1.

Tabela 5.1: Resultados obtidos após a execução do primeiro algoritmo implementado em quatro versões distintas.

VERSÃO	NÚMERO DE CTs GERADAS	TEMPO DE EXECUÇÃO (s)
A origem é fornecida e os destinos são aleatórios.	7	4
A origem e o 1o destino são fornecidos e os demais são randômicos.	10	4
A origem é fornecida e os destinos são escolhidos seqüencialmente.	10	6
A origem e o 1o destino são fornecidos e os demais são escolhidos seqüencialmente.	11	6

A Tabela 5.1 mostra que a versão do algoritmo projetado cuja origem do caminho é fornecida e os demais destinos são selecionados aleatoriamente obteve o melhor resultado, pois, com ela, gerou-se 7 CTs. Os números diferentes de CTs encontrados pelo algoritmo desenvolvido nestas versões devem-se aos tamanhos dos caminhos encontrados que são diferentes em cada caso. Portanto, a versão que encontrou o número maior de CTs é também a versão que gerou mais caminhos com tamanhos distintos. Ainda nesta linha, considerando que o tempo de configuração de uma CT, no pior caso, é de 3s, o tempo de teste pode ser previsto:

$$t_{stuck-open} = 3s * N^{o}CT = 3s * 7 = 21s \qquad (5.1)$$

Quanto aos tempos de execução de cada versão do algoritmo, pode-se dizer que, devido ao pequeno número de elementos da rede global, eles são muito semelhantes, baixos e, conseqüentemente, não proibitivos.

A Figura 5.2 mostra, em detalhes, a primeira CT gerada com este algoritmo. As demais CTs podem ser vistas no APÊNDICE A desta dissertação.

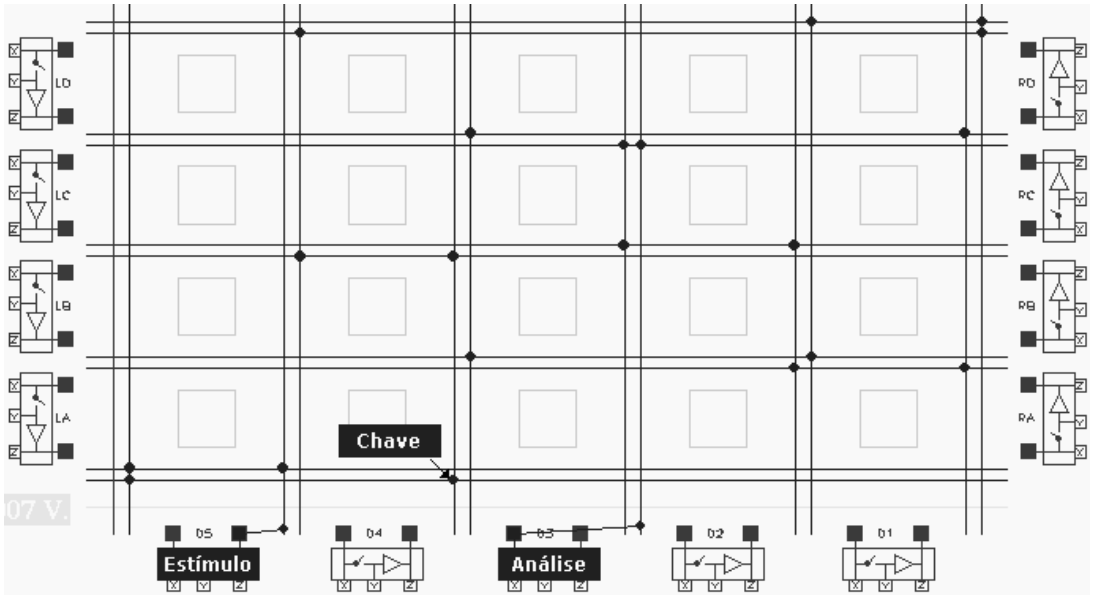


Figura 5.2: Geração de caminhos não bufferizados.

Pode-se também observar, através desta figura, que as chaves da rede global que pertencem ao caminho selecionado são programadas para o estado “on”, ou seja, o estado oposto ao da falha *stuck-open* que se deseja testar, conforme descrito na Seção 5.2.

Para fins de controle e formalização dos resultados obtidos, pode-se representar os caminhos críticos selecionados no *array* de interconexões através do modelo matemático que representa a rede. Mas, para que estes procedimentos sejam feitos corretamente, é preciso que inicialmente seja feita uma consideração:

- Durante o procedimento de teste, substitui-se o valor “**PS**” pelo valor lógico “**1**” no modelo matemático quando o elemento de ligação correspondente será ou já foi testado. Desta maneira, quando todo a matriz contiver “**1s**” em substituição aos “**PSs**”, tem-se que o teste foi concluído. A Figura 5.3 ilustra uma parte da interface de projeto do AN10E40 onde alguns barramentos e I/Os são nomeados conforme sua disposição no *array*. A seguir, a Tabela 5.2 ilustra um segmento de caminho selecionado durante o teste.

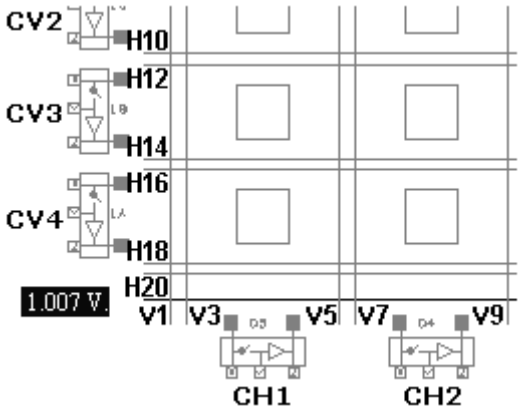


Figura 5.3: Interface de projeto com barramentos e I/Os nomeados.

Tabela 5.2: Matriz resumida mostrando um segmento de caminho selecionado.

		H2	H4	H6	H8	H10	H12	H14	H16	H18	H20	CH1	CH2	CH3	CH4
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
V1	1	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	0	0	0	0
V3	2	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	0	0	0	0
V5	3	1	PS	PS	PS	PS	PS	PS	PS	PS	PS	1	0	0	0
V7	4	1	1	PS	PS	PS	PS	PS	PS	PS	PS	PS	0	0	0
V9	5	PS	1	1	PS	PS	PS	PS	PS	PS	PS	0	PS	0	0
V11	6	PS	PS	1	1	PS	PS	PS	PS	PS	PS	0	PS	0	0
V13	7	PS	PS	PS	1	1	PS	PS	PS	PS	PS	0	0	PS	0
CV1	13	0	0	PS	PS	0	0	0	0	0	0	0	0	0	0

Por fim, partindo da formalização matemática desenvolvida, o comportamento do algoritmo pode ser explicado graficamente através da Figura 5.4.

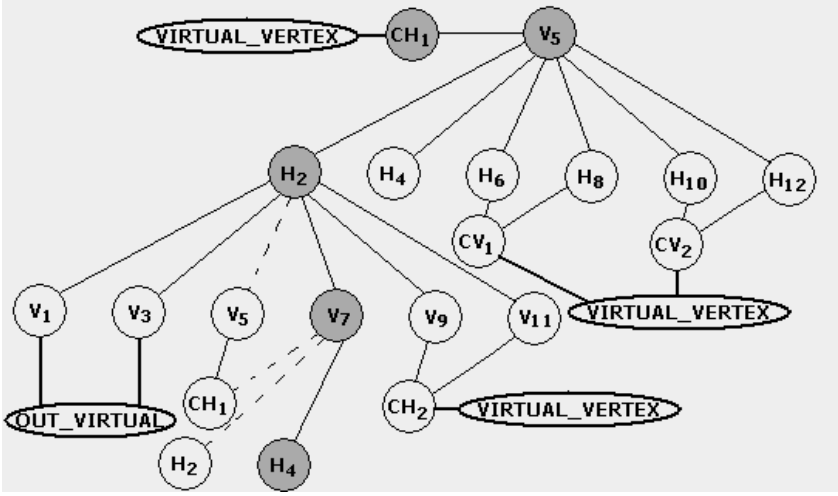


Figura 5.4: Representação gráfica resumida da execução do primeiro algoritmo.

Na figura, as linhas tracejadas representam as arestas já exploradas e os vértices coloridos representam os vértices já visitados em uma configuração de teste.

5.1.1.1.2 Geração de Caminhos Bufferizados

Caminhos bufferizados englobam não só as linhas de barramento e chaves da rede global, mas também as células de I/O que servem como *buffers* gerando, conseqüentemente, vários pontos de observação durante o teste e facilidades para o diagnóstico de falhas. Além disto, em conseqüência da alta impedância de entrada dos *buffers* selecionados nestes caminhos, não ocorre perda da integridade do sinal de teste. A Figura 5.5 ilustra um exemplo de caminho bufferizado.

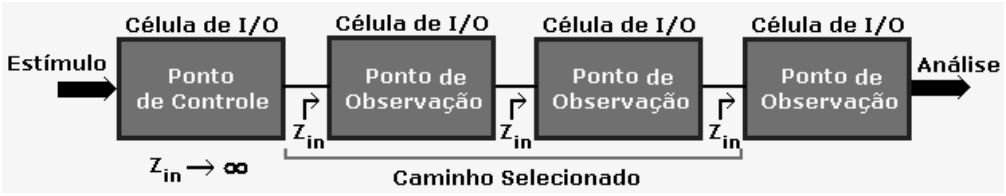


Figura 5.5: Ilustração de um caminho bufferizado.

Neste caso, além de serem selecionados caminhos diferentes dos propostos na Seção anterior, optou-se por uma nova estratégia de teste e, portanto, novas regras que devem ser seguidas a fim de que satisfazê-la, como por exemplo:

- As arestas que ligam as células de I/O aos barramentos que já foram exploradas em CTs anteriores podem ser exploradas novamente. Com isto, sempre estão disponíveis células para que a observação do sinal seja feita.

Por outro lado, da mesma forma que a estratégia anterior determina, os vértices visitados em uma determinada configuração de teste não são visitados novamente em uma mesma configuração, evitando-se, assim, a geração de *loops* no algoritmo.

Portanto, baseado no modelamento matemático da rede global e na estratégia de teste desenvolvida, é possível a adaptação de algoritmos, já utilizados nos testes de FPGAs [SUN 2002b] [SUN 2004], que, neste caso, geram caminhos bufferizados e que,

conseqüentemente, estimam o número de configurações de teste necessárias para cobrir todas as falhas *stuck-open* no *array* de interconexões globais do AN10E40. Deste modo, segue abaixo a descrição do pseudocódigo desenvolvido para selecionar estes caminhos:

c) Etapa de Pesquisa, Seleção de Caminhos Críticos e Finalização.

```
repeat
        Vertice[ ] ← NIL;
        for cada vértice u ∈ V[G]
                v = Search_IO_Vertex(u);
                if (v = = NULL) then
                        v = Search_New_Vertex(u);
                        if (v = = NULL) then
                                Backtracking( );
                                break;          -- Término de uma CT.
                        else
                                Select_Path&&Coloring( );
                                u = v;
                else
                        Select_Path&&Coloring( );
                        u = v;
until all (u,v) ∈ Expl[ ]

Search_IO_Vertex(u) -- Função que procura por células de I/O
        for cada v ∈ π [u]
                if ( (v ←Io_cell = TRUE) && (v ∉ Vertice[ ])) then
                        return(v);
        return(NULL);

Search_New_Vertex(u) -- Função que procura por um novo vértice.
        for cada v ∈ π [u]
                if (v ∉ Vertice[ ]) && (u,v) ∉ Expl[ ]) then
                        Expl[ ] ← (u,v);--Lista com as arestas exploradas.
                        return(v);
        return(NULL);
```

As etapas de inicialização e mapeamento não foram descritas aqui, pois são iguais às apresentadas no algoritmo anterior, com a ressalva de que, nesta ocasião, as células de I/O também são mapeadas no grafo.

Neste caso, como os caminhos englobam células de I/O, é dada a prioridade de destino a estas células. Portanto, inicialmente, a procura por um vértice *v* correspondente a I/O é desenvolvida (*v* = **Search_IO_Vertex(*u*)**). Caso exista um vértice destino *v* correspondente a uma célula de I/O que ainda não foi visitado, ele é inserido no caminho (**Select_Path&&Coloring( )**). No entanto, se não existir nenhuma célula de I/O disponível que satisfaça estas condições, a função **Search_New_Vertex(*u*)** determina que um vértice *v*, correspondente a uma linha da rede global que ainda não foi visitado e cuja aresta que liga *u* a *v* ainda não foi explorada em CTs anteriores, seja selecionado. No último caso, quando nenhuma das condições é satisfeita, a pesquisa retorna para a última célula de I/O introduzida no

caminho (**Backtracking( )**) e uma CT é concluída. A pesquisa continua até que todas as arestas do grafo sejam exploradas.

A validação da proposta também foi desenvolvida através da descrição do pseudocódigo na linguagem C em quatro versões diferentes visando, com isto, obter o menor número de CTs, ter ganhos no tempo de execução do algoritmo e no tempo de teste.

Tabela 5.3: Resultados obtidos após a execução do segundo algoritmo implementado em quatro versões distintas.

VERSÃO	NÚMERO DE CTs GERADAS	TEMPO DE EXECUÇÃO (s)
A origem é fornecida e os destinos são aleatórios.	17	13
A origem e o 1o destino são fornecidos e os demais são aleatórios.	14	11
A origem é fornecida e os destinos são escolhidos seqüencialmente.	15	9
A origem e o 1o destino são fornecidos e os demais são escolhidos seqüencialmente.	15	10

Aqui, a versão onde a origem do caminho e o primeiro destino são definidos por uma função e os demais destinos são escolhidos aleatoriamente apresentou melhores resultados por gerar o menor número de CTs. Também aqui se pode justificar o número diferente de CTs encontrados pelo algoritmo desenvolvido nestas versões devido ao fato do tamanho dos caminhos encontrados serem diferentes em cada caso. Portanto, conhecendo-se o número de CTs necessárias para cobrir toda a rede global, pode-se obter o tempo de teste destas falhas:

$$t_{stuck-open} = 3s * N^{o}CT = 3s * 14 = 42s \qquad (5.2)$$

Na Tabela 5.3, observa-se ainda que os tempos de execução e o número de CTs geradas são maiores utilizando-se o segundo algoritmo uma vez que o número de arestas a serem exploradas e a heurística adota é outra. Abaixo é mostrada, em detalhes, a primeira CT obtida com este algoritmo. Entretanto, todas as 14 CTs são apresentadas no APÊNDICE B desta dissertação.

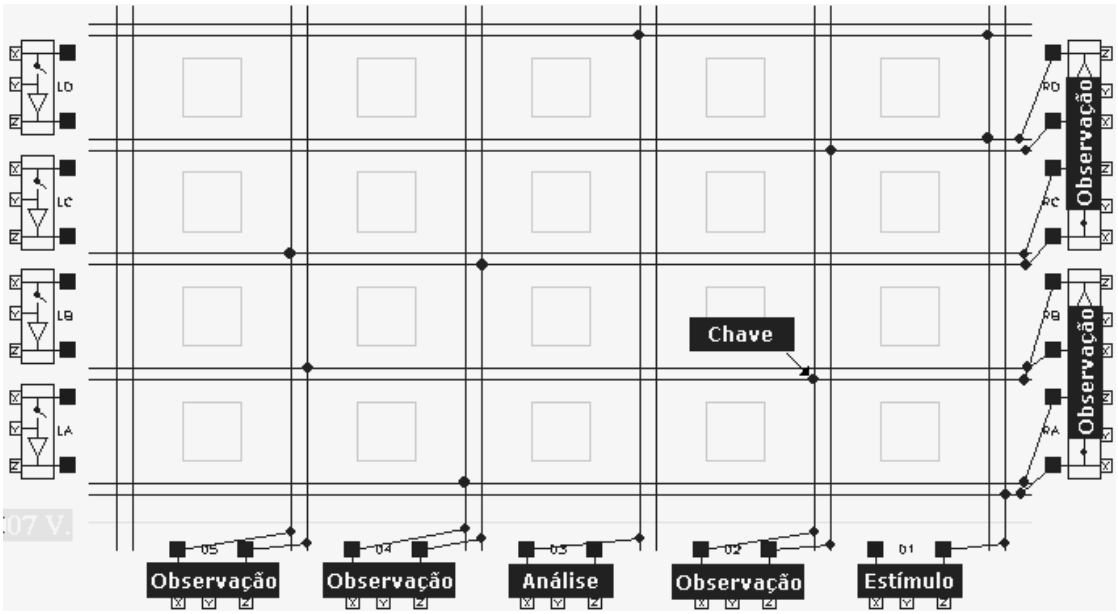


Figura 5.6: Geração de caminhos bufferizados.

Por fim, os modelos matemático e gráfico também podem ser utilizados a fim de formalizar e controlar os resultados obtidos, conforme foi proposto no caso anterior. Aqui, é ilustrado o comportamento do algoritmo apenas através do modelo gráfico resultante da representação matemática da rede global.

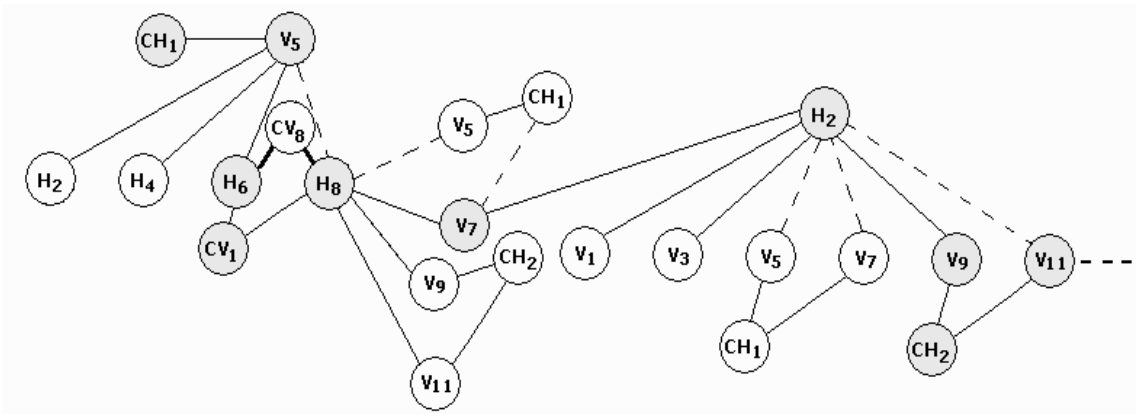


Figura 5.7: Representação gráfica resumida do segundo algoritmo.

Na Figura 5.7, as linhas tracejadas representam as arestas já exploradas e os vértices coloridos representam os vértices já visitados em uma configuração de teste. As linhas mais espessas representam as ligações que foram processadas, mas que foram desprezadas devido ao fato do vértice alvo já ter sido visitado.

5.3.1.2 Teste e Detecção de Falhas do Tipo Stuck-on e Bridging

Nesta seção são descritas duas estratégias distintas para teste e detecção de falhas do tipo *stuck-on* ou *bridging* na rede global de interconexões de FPAAs. Embora, muitas vezes, estas estratégias possibilitem também o diagnóstico de falhas, é importante salientar que o foco desta dissertação se limita à detecção destas falhas.

5.3.1.2.1 Estratégia de Teste

O modelo que representa a rede global de interconexões citado na Seção 4.1.1.1, também é utilizado neste caso para definir regras para o teste de *stuck-ons* e *bridgings*. Questões sobre o mapeamento dos elementos de conexão e dos barramentos da rede global de interconexões no grafo, já foram solucionadas naquela Seção.

Esta estratégia tem como idéia principal a seleção do maior número de caminhos elétricos de teste em uma mesma configuração e, conseqüentemente, a diminuição do número total de configurações. Assim como no caso anterior, os estímulos de teste são aplicados externamente. O número de células de I/O disponíveis para a aplicação de estímulos e observação de respostas de teste é suficiente para que se obtenha reduções significativas no número de CTs.

Como as linhas horizontais e verticais são contínuas e, de uma forma geral, têm acesso às células de I/O, várias chaves programáveis podem ser testadas inserindo-se, por exemplo, um estímulo de teste em uma linha do barramento vertical. Como esta pode ser ligada a todas as linhas horizontais, que, por sua vez, podem ser ligadas diretamente a algumas células de I/O do dispositivo, suas chaves são programadas para o estado "off". Com esta simples configuração já seria possível testar algumas chaves, uma vez que, existindo a falha, o sinal será transmitido para estas células. Alguns caminhos disjuntos selecionados aleatoriamente são ilustrados na Figura 5.8.

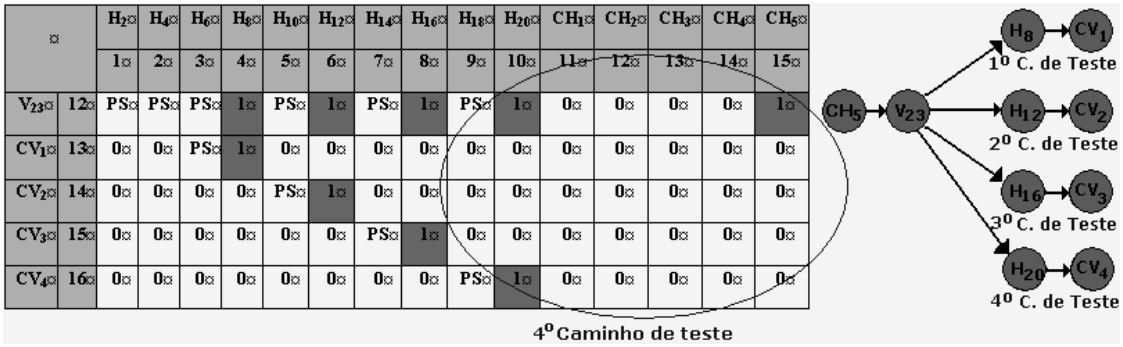


Figura 5.8: Caminhos de teste selecionados.

Uma breve descrição do teste foi dada acima, porém, o procedimento de teste completo é delineado a seguir:

- Em uma primeira configuração, o estímulo é inserido em uma linha (vertical, por exemplo), e programa-se as chaves que a conecta a algumas linhas do barramento ortogonal (linhas horizontais), conforme ilustra a Figura 5.9a. Com isso, são testadas as chaves que conectam a linha vertical às outras linhas horizontais, bem como as chaves que conectam as linhas horizontais selecionadas às outras linhas verticais. Além disso, como o sinal de estímulo flui por algumas linhas horizontais, no caso do sinal ser observado na saída das outras paralelas, é uma indicação de uma falha de *bridging* entre essas linhas.

- Para cobertura das outras as chaves, bastaria inverter as chaves programáveis, conectando, agora, outras linhas horizontais à linha vertical em que é inserido o sinal, a fim de testar as demais chaves, conforme ilustra a Figura 5.9b.

- Uma terceira CT é necessária para detectar falhas de *bridging* entre as linhas verticais e para testar as chaves complementares. Com isso, obtêm-se as três configurações necessárias para a cobertura de todas as falhas *stuck-on* nas chaves e de

bridging entre linhas. A Figura 5.9c ilustra a terceira TC para uma rede com seis linhas verticais e horizontais. Além disto, nesta figura, para cada CT é ilustrada a sua representação matricial.

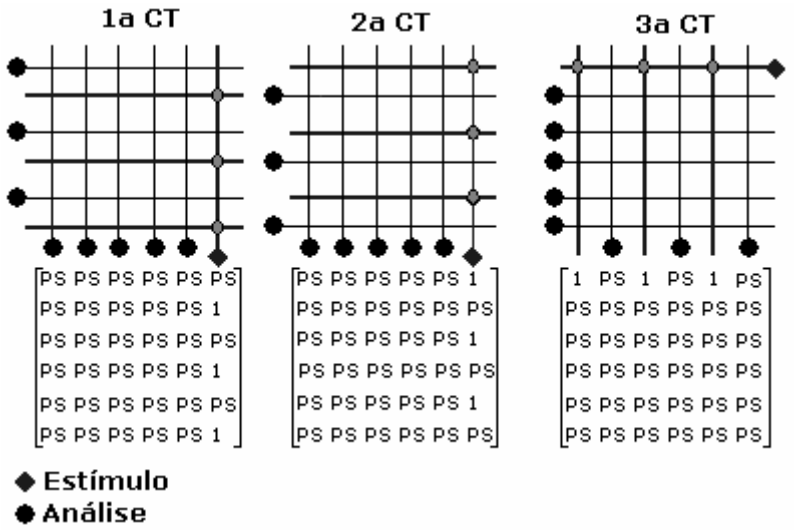


Figura 5.9: Representação em matriz das configurações de teste para falha *stuck-on*.

Portanto, sabendo-se que são necessárias 3 CTs para cobrir falhas *stuck-on* na rede global de interconexões utilizando esta estratégia, pode-se obter o tempo de teste destas falhas:

$$t_{stuck-on} = 3s * N^{o}CT = 3s * 3 = 9s \qquad (5.3)$$

Embora esta estratégia permita, muitas vezes, inclusive o diagnóstico das falhas, se o tempo de teste for um critério importante, outra estratégia que gera menos CTs pode ainda ser obtida. Propõe-se, como segunda alternativa, a aplicação de sinais com freqüências fundamentais distintas em cada linha. Assim, em determinadas situações, a presença de falhas *stuck-on* será observada pela soma ponderada destas freqüências nas linhas curto-circuitadas. Entretanto, dependendo do dispositivo, ainda seria necessário um número razoável de sinais distintos, bem como um número razoável de pinos de entrada para conduzir esses sinais de estímulo à rede de interconexões. Assim, esta proposta usa no máximo três sinais de estímulo. As configurações obtidas serviriam também para o teste de *bridging* entre duas linhas adjacentes e o número de CTs geradas diminui para 2. A Figura 5.10 ilustra a primeira CT gerada utilizando a estratégia de múltiplas freqüências.

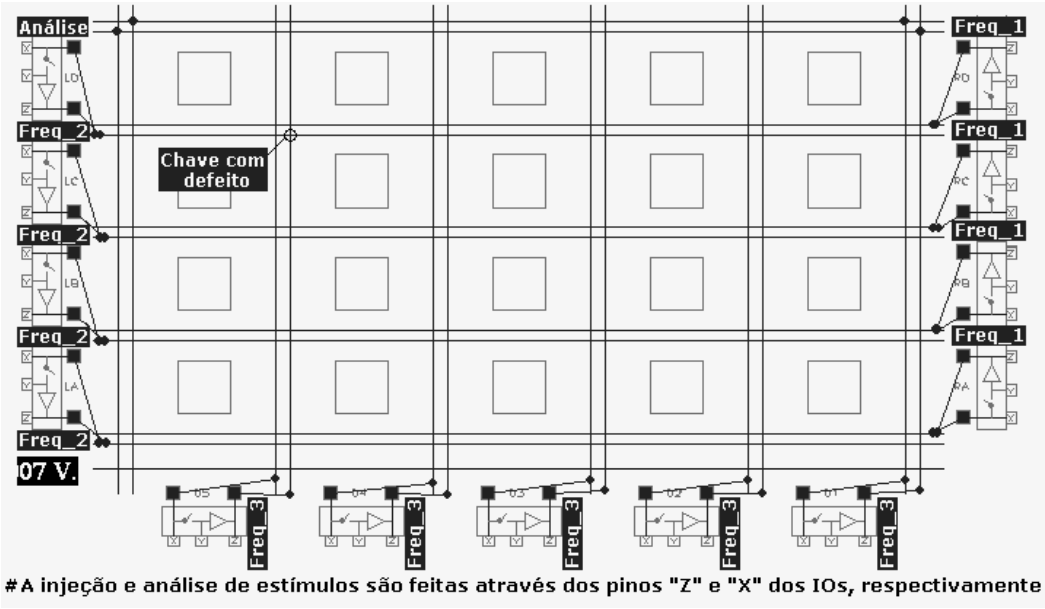


Figura 5.10: Primeira CT gerada utilizando a estratégia de múltiplas freqüências.

Ainda neste contexto, a fim de validar esta proposta, uma etapa de caracterização dos elementos de conexão e da resistência de saída de um CAB do AN10E40 foi realizada, aplicando-se estímulos DC (5V) externos e utilizando-se uma carga externa de 1KΩ. Os resultados obtidos nesta etapa e após emular uma falha do tipo *stuck-on* na chave destacada na Figura 5.10, utilizando-se estímulos de 50KHz(Freq_2) e 22KHz(Freq_3), são mostrados abaixo:

Tabela 5.4: Caracterização dos elementos do AN10E40.

ELEMENTO CARACTERIZADO	RESISTÊNCIA (Ω)
Chaves da Rede Global	145
Resistência de Saída de um CAB	70

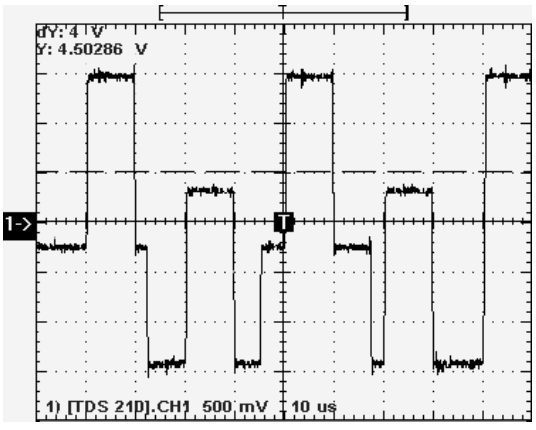


Figura 5.11: Soma ponderada de freqüências quando uma falha do tipo *stuck-on* é emulada.

Experimentalmente, pôde-se observar que a resistência das linhas tem magnitude muito inferior a dos elementos de conexão, portanto, ela foi desprezada. Além disto, a resistência da fonte de estímulos DC também não foi considerada neste procedimento.

A partir dos resultados de emulação da rede, geraram-se simulações em SPICE a fim de avaliar melhor o seu comportamento quando o valor da resistência dos CABs é diferente do seu valor nominal (70Ω), conforme ilustra a Figura 5.12. Concluiu-se que, devido à resistência pequena, porém não-nula, das linhas, o sinal gerado na chave que conecta um barramento a outro com sinais distintos é equivalente à soma ponderada destes sinais, como mostra a Figura 5.11. Para verificar esta possibilidade, na simulação, definiram-se valores de resistência das linhas da ordem de $10^{-3}\Omega$ e aplicaram-se duas ondas quadradas com freqüências fundamentais de 22KHz e 50KHz sendo que os resultados obtidos quando a resistência R1 tende a diminuir são mostrados na Figura 5.13.

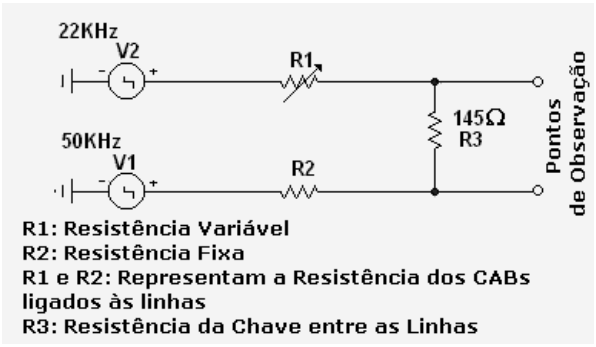


Figura 5.12: Circuito simulado.

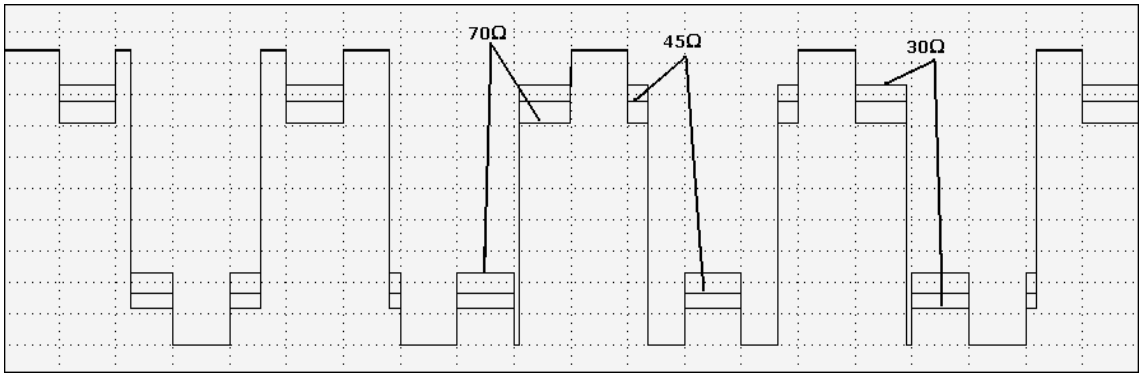


Figura 5.13: Soma ponderada das freqüências dos estímulos aplicados.

Nesta última figura, pode-se observar que quando a resistência da carga conectada na linha de 22KHz é 30Ω, esta freqüência tende a se tornar dominante sobre a linha de 50KHz. Estes resultados são mais evidentes quando os sinais utilizados como estímulos possuem freqüências bastante distintas.

Portanto, uma vez que o número de CTs geradas utilizando a segunda estratégia é menor do que o número obtido quando a primeira estratégia é escolhida, verificando-se a Equação 5.3, é possível ver que o tempo de teste também assim será:

$$t_{stuck-on} = 3s * N^{o}CT = 3s * 2 = 6s \qquad (5.4)$$

As configurações de teste elaboradas conforme a primeira e segunda propostas descritas para o teste de falhas *stuck-on* na rede global do AN10E40 são apresentadas, respectivamente, no APÊNDICE C e APÊNDICE D desta dissertação.

5.4 Teste da Rede Local de Interconexões e I/O

Como foi visto no Capítulo anterior, a rede local de interconexões possibilita a ligação de blocos programáveis entre si e com algumas células de I/O sendo, portanto, de suma importância para o desenvolvimento de projetos analógicos. Também foi visto que, o número de interconexões locais que um CAB ou uma célula de I/O pode realizar depende da sua posição no *array* e que os seus respectivos pinos de entrada e saída possuem as mesmas regras de interconexão e, conseqüentemente, os mesmos modelos.

Sendo assim, nesta Seção são descritos o algoritmo e uma estratégia que foram elaborados com base no modelo de interconexões da rede local a fim de satisfazer as exigências de seu teste. Por fim, são mostradas as CTs geradas a partir do algoritmo desenvolvido.

5.4.1 Estratégia de Teste das Interconexões Locais dos CABs e Células de I/O

Assim como no teste da rede global, o teste da rede local de interconexões segue uma metodologia similar às adotadas em [SUN 2002c], [SUN 2002d] e [SUN 2004], tendo por objetivo detectar prováveis falhas *stuck-on* e *stuck-open* em todas as suas chaves. Contudo, diferentemente daquele teste, ambas as falhas são aqui abordadas em uma única estratégia, a seleção de caminhos. Além disto, a simulação de falhas nesta rede obedece ao mesmo procedimento descrito na Seção 5.2, isto é, quando se deseja detectar falhas *stuck-on* em uma chave da rede local, programa-se a chave para o estado "off" e aplica-se o estímulo de teste externamente. A presença do estímulo em um dos pontos de observação significa que esta chave tem defeito. Entretanto, quando o teste visa detectar falhas *stuck-open*, todas as chaves do caminho são mantidas no estado "on" e a ausência do estímulo aplicado em qualquer dos pontos de observação indica que há falha em uma das chaves. A Figura 5.14 mostra os procedimentos descritos para o teste de falhas *stuck-on* e *stuck-open*.

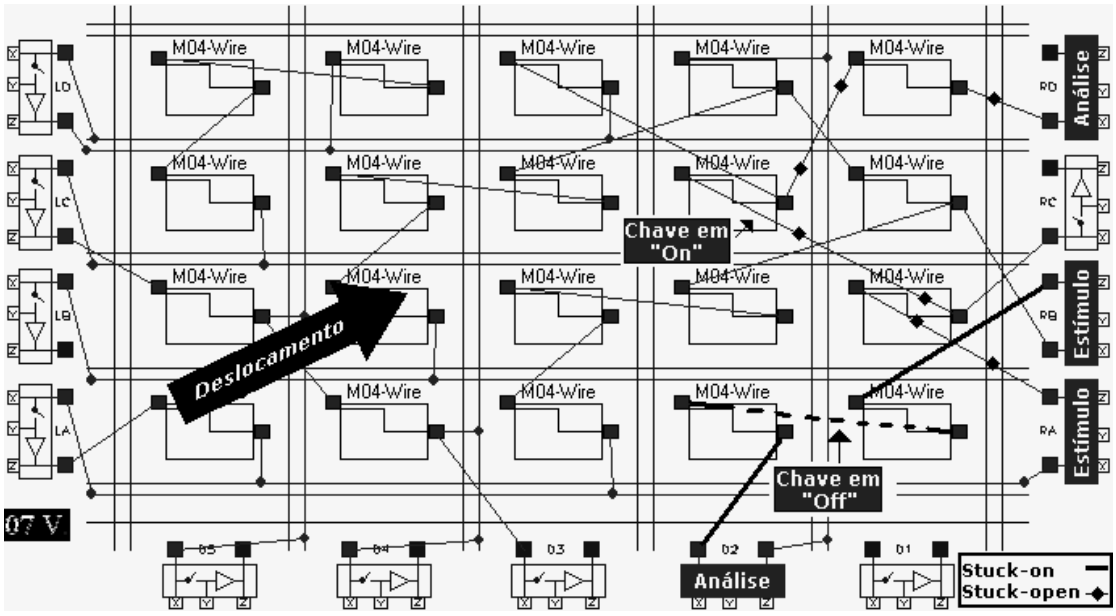


Figura 5.14: Procedimentos de teste de falhas *stuck-on* e *stuck-open* da rede local.

Para a elaboração desta estratégia de teste das interconexões locais dos CABs e células de I/O, fixaram-se normas derivadas do modelo que as representam, tais quais:

- Quando as entradas de um dos CABs são analisadas no modelo descrito na Tabela 4.3, na Seção 4.2.1, tem-se que, uma aresta horizontal que conecta dois vértices adjacentes em **G(X)** indica que as duas chaves correspondentes a estes vértices não podem estar no estado "on" simultaneamente na mesma configuração de teste, ou seja, um CAB não pode ser visitado mais do que uma vez em uma CT. Isto se deve ao fato das suas entradas serem multiplexadas.

- Quando as saídas de um dos CABs são analisadas no modelo descrito na Tabela 4.3, na Seção 4.2.1, tem-se que, uma aresta horizontal que conecta dois vértices em **G(X)** indica que no máximo duas chaves "**PS**" localizadas na mesma linha de saída podem estar programadas para o estado "on" simultaneamente em uma CT. Esta regra deve-se à limitação do *driver* de saída de cada CAB citada também naquela Seção que, segundo o fabricante deste dispositivo programável, suporta uma carga de 100pf, ou seja, equivalente a duas interconexões locais ou a uma interconexão local e duas interconexões com os barramentos.

- Ainda neste contexto, uma aresta vertical que conecta dois vértices no grafo **G(X)** indica que uma determinada entrada/saída de um CAB pode ser conectada a uma saída/entrada de um dos CABs adjacentes em uma CT, desde que, entre estes, o CAB que for escolhido como destino não tenha sido fonte de estímulo para outro que já foi selecionado na pesquisa. Com isto, evita-se a geração de laços durante a pesquisa.

5.4.2 Proposta de Algoritmo para Teste das Interconexões Locais dos CABs e I/Os

Assim como no teste da rede global, a seleção de caminhos utilizando grafos de adjacências também é utilizada neste caso. Deste modo, conhecendo-se a estratégia de teste e as regras que devem ser seguidas a fim de atender as necessidades de teste da rede local de interconexões do dispositivo, já é possível descrever o pseudocódigo do algoritmo que vai ao encontro destas exigências:

a) Etapa de Inicialização.

```
for cada  u ∈ V[G]
        color[u] ←"white";
π[u] ←NIL;              - Lista de adjacência do vértice u;
Expl[ ] ←NIL;           - Lista que contém todas as arestas exploradas;
Vertice[ ] ←NIL;        - Lista que contém todos os vértices visitados.
N0_IO_CELLS = 13        Número de células de I/O disponíveis.
MAXIMO_OUTPUT = 2       Nº máximo de saídas de um CAB que podem
                        atuar como fontes de estímulos.
```

b) Etapa de Mapeamento.

```
for cada u ∈ V[G]
        for cada v ∈ π [u]
                π [u] ← v;
```

c) Etapa de Pesquisa, Seleção de Caminhos e Finalização.

```
main( )
        source = Search_New_Origin( ); --Onde "source" é a origem do caminho.
        Search_One_Path(source);
        for cada u ∈ V[G]
```

```
            if ((u ∈ Vertice[ ]) && (u→Output = = "0") then
                    counter_outputs++;
            if (counter_outputs> N0_IO_CELLS) then -Realiza o teste sobre a 1ª CT.
                    source = Search_New_Origin( );
                    Search_One_Path(source);
            else
                    repeat
                            Shifting_Configurations();--Desloca CTs resultantes.
                    until all ((u,v) ∈ Expl[ ]);
                    break;
    return;

Search_One_Path(u)
    for cada v ∈ π [u]
        if (v ∉ Vertice[ ]) then
            if (Analyse_All_Cabs_Same_Diagonal(u, v→posicao) = = TRUE) then
                    if (counter_output >= MAXIMO_OUTPUT) then
                            counter_output++;
                            Setting_All_Cabs_Same_Diagonal(u, v→posicao);
                            Search_One_Path(source);
                    else
                            counter_output = 0;
                            source = Search_New_Origin( );
                    return;
            else
                    Vertice[ ]← v;
                    Search_One_Path(source);
    return;

Search_New_Origin( )
    for cada u ∈ V[G]
        if (u ∉ Vertice[ ]) then
            for cada u ∈ V[G]
                    if ((u ∈ Vertice[ ]) && (u→Output = = "0") then
                            return(u);

Analyse_All_Cabs_Same_Diagonal(v, destination)
    for cada u ∈ π [v]
        if ((u→Diagonal = = "1") then
            for cada y ∈ π [u]
                    if (y→posicao = =destination) && (y ∉ Vertice[ ])) then
                            Analyse_All_Cabs_Same_Diagonal(u, y→posicao);
                    else
                            return(FALSE);
     return(TRUE);

Setting_All_Cabs_Same_Diagonal(u, destination)
    for cada v ∈ π [u]
        if ((v→Diagonal = = "1") then
            for cada y ∈ π [v]
                    if (y→posicao = =destination) then
                            Vertice[ ]← y;
```

```
                    v→Output="1";
                    Expl[ ]= (v,y);
                    Setting_All_Cabs_Same_Diagonal(v, y→posicao);
return;
```

A idéia principal deste algoritmo é gerar a primeira CT e a partir dela gerar as CTs complementares que cobrem o restante das chaves da rede local através de deslocamentos dos caminhos gerados, conforme mostra a Figura 5.14. Sendo assim, o algoritmo começa selecionando um vértice fonte *u* chamando a função **Search_New_Origin( )** e, posteriormente, inicia o processo de seleção de caminhos propriamente dito através da função **Search_One_Path(***source***)**. A condição principal de teste que é realizada nesta função é verificar se o vértice *v* pertencente à lista de *u* ainda não foi visitado ($v \notin$ Vertice[ ]). Caso ele não tenha sido visitado, a função **Analyse_All_Cabs_Same_Diagonal(***v*, destination**)** é chamada e nela, a cada execução do laço, é selecionado um outro vértice *v* pertencente à diagonal de *u*, além de ser verificado se o vértice *y* adjacente a este vértice *v* ainda não foi selecionado. Para controle de posição de *y* e do vértice *v*, esta função recebe dois parâmetros denominados "destination" e "*v*", respectivamente. Ainda nesta linha, caso esta função retorne o valor "TRUE" todos os vértices *y* são assinalados como visitados, as arestas (*v*,*y*), constituídas pelo vértice *y* e pelo seu predecessor *v*, são exploradas e os campos "saídas" dos vértices *v's* são setadas para "**1**" (**Setting_All_Cabs_Same_Diagonal(***u*, destination**)**). Por outro lado, caso ela retorne "FALSE", um novo vértice *v* ligado ao vértice fonte *u* é selecionado.

Voltando à função **Search_One_Path(***u***)**, uma outra condição de teste é realizada cada vez que a resposta de **Analyse_All_Cabs_Same_Diagonal(***v*, destination**)** for "TRUE". Nela é verificado se o número de saídas ativas dos vértices que servem como fontes de estímulos é menor ou igual a 2 (counter_output >= MAXIMO_OUTPUT). Caso o número seja maior, um novo vértice fonte *u* é selecionado. A pesquisa continua até que a primeira CT esteja concluída, ou seja, até que não existam mais vértices a serem visitados.

A partir daí, ocorre o retorno ao programa principal onde será definido se uma pesquisa deve ser realizada a fim de obter uma nova primeira CT que satisfaça a condição de originar um número de pontos de observação menor ou igual ao número de células de I/O disponíveis no dispositivo testado. Caso esta condição seja satisfeita, a função que realiza os deslocamentos das CTs geradas (**Shifting_Configurations( )**) é executada até que todas as arestas do (*u*,*v*) do grafo sejam exploradas.

Com este algoritmo conseguiu-se obter 13 CTs para cobrir as falhas *stuck-open* das chaves da rede local e 71 CTs para cobrir 100% das falhas *stuck-on* desta rede. Vale lembrar, que o número de CTs para cobrir estas últimas falhas é determinado pelo número de vértices que são selecionados no caminho crítico de cada configuração. Sendo assim, depois de concluída esta etapa, já é possível prever o tempo total de teste da rede local de interconexões aplicando-se a estratégia delineada acima:

$$t_{stuck-open} = 3s * N^{o}CT = 3s * 13 = 39s \qquad (5.5)$$

$$t_{stuck-on} = 3s * N^{o}CT = 3s * 71 = 213s \qquad (5.6)$$

portanto, o tempo total de teste é:

$$t_{teste} = t_{stuck-on} + t_{stuck-open} = 39s + 213s = 252s \qquad (5.7)$$

Por fim, os tempos de execução do algoritmo foram distribuídos segundo a Tabela 5.5. Como pode ser visto, os tempos totais para executar o algoritmo a fim de obter estas CTs são 9,4s no pior caso e 5,1s no melhor caso. Uma das CTs obtidas utilizando este algoritmo é mostrada, em detalhes, na Figura 5.14. As demais CTs são ilustradas no APÊNDICE E desta dissertação.

Tabela 5.5: Tempos de execução das funções do algoritmo.

FUNÇÃO	TEMPO DE EXECUÇÃO(s)	
	PIOR CASO	MELHOR CASO
Responsáveis pela geração da 1ª CT	5.3	1.1
Shifting_Configurations( )	4.0	4.0

5.5 Teste das Chaves e *Buffers* de Células de I/O

Como descritas na Seção 3.1, as células de I/O são constituídas por entradas multiplexadas, por *jumpers* nas saídas e por chaves e *buffers*, conforme ilustra a Figura 5.15. Como a estratégia para teste das ligações locais das entradas e saídas destas células já foi delineada na Seção anterior, para que o teste de toda a sua estrutura seja concluído, ainda é preciso testar os seus *buffers* e chaves internas. Aqui, é feita uma adaptação da estratégia de teste descrita em [ARA 96a], [ARA 96b] e [BAL 2002], que se baseia em oscilação (OTS), para testar estas chaves e *buffers* internos ás células de I/O. Basicamente, o circuito analógico é particionado em blocos funcionais simples que, durante o modo teste, são convertidos em osciladores sendo que, para isto, muitas vezes, alguns circuitos adicionais são necessários. A freqüência de oscilação é expressa como função dos componentes do CUT (do inglês, *Circuit Under Test*) ou como função dos seus parâmetros importantes e a amplitude do sinal gerado pode ser utilizada como medida complementar de teste. Dentre outras vantagens desta técnica, pode-se citar o baixo custo e a não exigência de vetores de teste externos adequados para que altas coberturas de defeitos sejam atingidas.

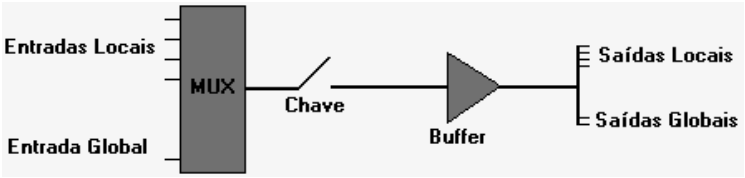


Figura 5.15: Circuito esquemático de uma célula de I/O.

Especificamente para o caso do AN10E40, tendo-se como objetivo minimizar o número de CTs, todos os *buffers* e chaves são conectados em cadeias, similarmente à técnica *scan-chain* utilizada no teste digital para avaliar o funcionamento de *flip-flops* [IEE 90]. Além disto, ainda com o mesmo objetivo de minimizar a número de CTs, algumas topologias de oscilador foram avaliadas observando-se a estabilidade, linearidade e repetibilidade de estímulos gerados. Inicialmente, implementou-se um Oscilador em Quadratura [MAN 2000] [SED 2004] que por utilizar apenas dois componentes com inversão de fase, que podem ser programados em qualquer das 9 zonas acessíveis a um CAB, tornar-se-ia interessante para satisfazer a exigência de minimizar o número de CTs. No entanto, ao ser implementado utilizando os CABs do AN10E40, ele apresentou baixa estabilidade e repetibilidade. Por outro lado, apesar do

Oscilador com Deslocamento de Fase (PSO - *Phase-Shift Oscillators*) [MAN 2000] [SED 2004] exigir, no mínimo, quatro componentes, ele tornou-se atraente, uma vez que, apresentou alta repetibilidade, linearidade e estabilidade. Este oscilador é composto por um amplificador inversor com ganho *A*, e por uma rede com 3 filtros passa-baixas que forma um sistema de terceira ordem e compõe o seu laço de realimentação. Adicionalmente, assim como o Oscilador em Quadratura, todos os seus componentes podem ser construídos utilizando os recursos internos deste FPAA. A razão de utilizar três filtros na rede de realimentação é que este é o número mínimo para produzir uma mudança de fase de 180^o para uma freqüência finita. Logo, supondo que as seções do deslocamento de fase são independentes umas das outras e que deslocam a fase em -60^o, conseqüentemente, é possível a elaboração das equações abaixo.

$$\frac{V_o(s)}{V_i(s)} = \frac{2\pi f_o G}{s + 2\pi f_o} \tag{5.8}$$

Onde $V_o(s)$ e $V_i(s)$ são as tensões de saída e entrada de cada seção, respectivamente, f_o é a freqüência de corte do filtro e G é o ganho do filtro. Agora, obtendo-se a expressão de fase deste filtro, tem-se:

$$\phi = \tan^{-1}\left(\frac{0}{2\pi f_o G}\right) - \tan^{-1}\left(\frac{\varpi}{2\pi f_o}\right) = 0 - \tan^{-1}\left(\frac{f}{f_o}\right) \tag{5.9}$$

Considerando que $\phi = 60^o$, tem-se que $f = 1.732 * f_o$, onde $\tan 60^0 \approx 1.732$.

Por outro lado, baseado na Figura 5.16, que ilustra o circuito esquemático do PSO englobando algumas células de I/O e o diagrama em blocos de um sistema oscilatório, elaborou-se a expressão que representa o ganho de um PSO:

$$A\beta = A\left(\frac{1}{RCs + 1}\right)^3 \tag{5.10}$$

onde β é o fator de realimentação equivalente a $(1/2)^3$ e A é o ganho do amplificador e deve ser 8 para um sistema de ganho unitário.

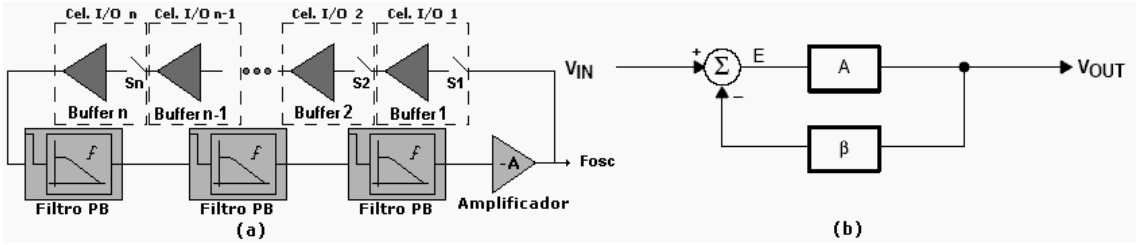


Figura 5.16: (a) Circuito esquemático do PSO englobando múltiplos *buffers* e chaves de células de I/O, (b) Diagrama em blocos de um sistema oscilatório.

5.5.1 Análise de Repetibilidade de PSOs

No teste de circuitos analógicos é de grande importância que os geradores de estímulos apresentem uma alta precisão. Portanto, nesta Seção é realizada a avaliação da repetibilidade de PSOs que englobam um número diferente de células de I/O no seu laço de realimentação. A análise é desenvolvida programando-se 10 vezes o AN10E40 com PSOs atuando com freqüências e amplitudes diferentes e o critério de avaliação é feito observando-se se as variações destas grandezas estão fora da margem de tolerância

definida previamente que aceita erros na ordem de ±3% e ±5% dos seus respectivos valores nominais. Os resultados obtidos durante este procedimento estão contidos na Figura 5.17.

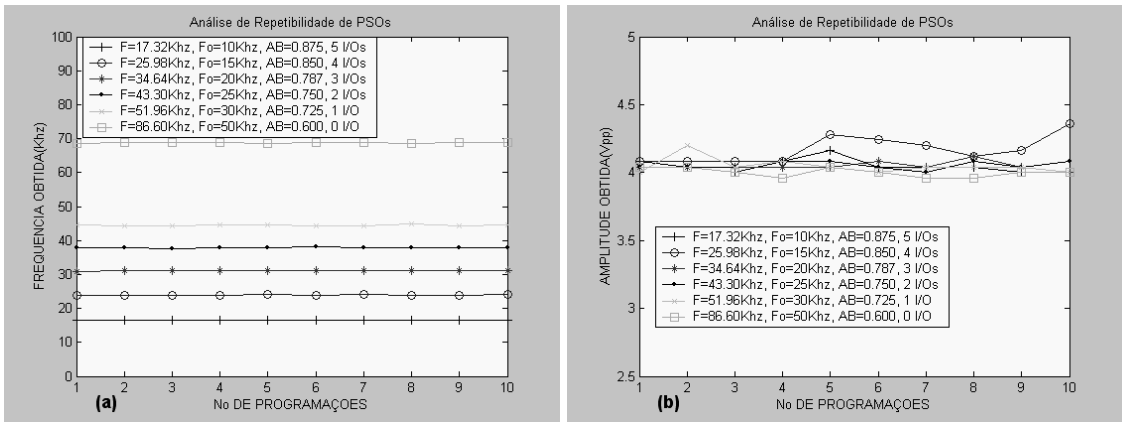


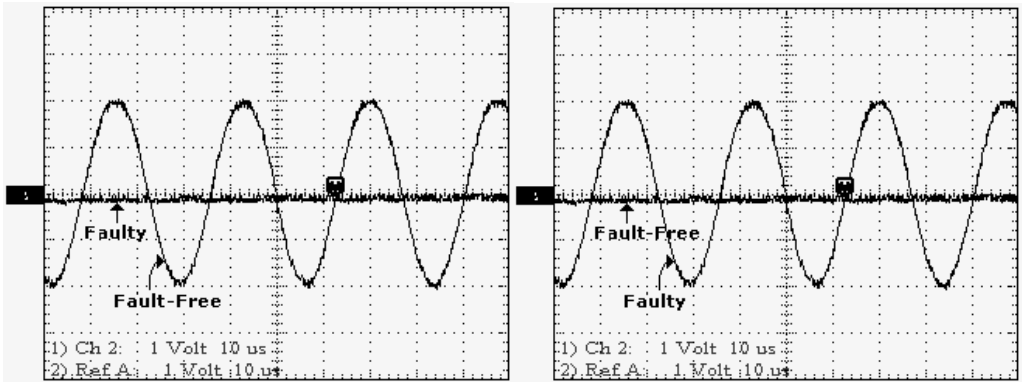
Figura 5.17: Análise da repetibilidade de PSOs.

A Figura 5.17a mostra a análise realizada quando diferentes freqüências de oscilação são programadas, enquanto que, a Figura 5.17b ilustra esta análise quando ganhos distintos são programados nos PSOs. Pode-se observar que, em nenhum dos casos avaliados, os valores de freqüências e amplitudes estiveram fora dos limites de tolerância definidos, porém, à medida que a freqüência de corte do filtro aumenta, a freqüência de oscilação obtida tende a ficar menor que a freqüência calculada. Assim sendo, pode-se concluir que, a repetibilidade de um PSO é alta, independe do número de I/Os inseridos no seu laço de realimentação, contudo, a exatidão do PSO tende a diminuir com o aumento da Fo dos filtros.

5.5.2 Estratégia de Teste

A metodologia adotada neste teste é a mesma adotada no teste da rede local de interconexões, ou seja, para detectar falhas *stuck-open* em qualquer das chaves das células é preciso que todas as chaves que formam o laço de realimentação, junto com os filtros, sejam programadas para o estado "on". Neste caso, se qualquer das chaves apresentar falha *stuck-open*, o circuito não oscila. No entanto, se a freqüência e/ou amplitude do sinal gerado varia em relação à resposta do circuito sem falhas, então isto significa que os parâmetros dos *buffers* estão fora dos limites de tolerância e/ou seus componentes diferem dos seus respectivos valores nominais.

Entretanto, quando se almeja detectar falhas *stuck-on* em uma chave de uma determinada célula, é preciso programá-la para o estado "off" e manter as demais chaves programadas para o estado "on". Neste caso, se a chave estiver com defeito, o circuito oscilará na freqüência e amplitude nominais, porém, caso exista algum desvio paramétrico obtido durante o seu projeto ou processo de fabricação, o circuito oscilará com amplitude e/ou freqüência diferentes. A Figura 5.18 mostra os sinais gerados por um PSO quando 5 células de I/O são introduzidas no seu laço, $A\beta$ é equivalente a 0,875, Fo é igual a 22KHz e, conseqüentemente, a freqüência de oscilação é de 38,1KHz. Nela é possível observar a resposta do circuito sem falhas, quando uma das suas chaves apresenta falhas *stuck-open* e *stuck-on*, respectivamente, e a diferença entre as freqüências de oscilação calculada e obtida (aproximadamente, 33KHz) que é delineada na Seção anterior.



(a) (b)

Figura 5.18: (a) e (b) Sinais gerados quando o PSO não apresenta falhas e quando falhas *stuck-open* e *stuck-on* são detectadas, respectivamente.

A fim de facilitar o diagnóstico de falhas, procurou-se distribuir as células de I/O do AN10E40 em vários osciladores. Logo, construíram-se 5 PSOs, tal que, 3 contêm 2 células de I/O, 1 contém 3 células e o último contém 4 células de I/O. Desta maneira, foi necessária 1 CT para testar falhas *stuck-open* e dela derivaram-se 4 CTs para testar falhas *stuck-on* nas chaves destas células. Conseqüentemente, os tempos de teste obtidos são:

$$t_{stuck-open} = 3s * N^{o}CT = 3s * 1 = 3s \qquad (5.11)$$

$$t_{stuck-on} = 3s * N^{o}CT = 3s * 4 = 12s \qquad (5.12)$$

portanto, o tempo total de teste é:

$$t_{teste} = t_{stuck-on} + t_{stuck-open} = 3s + 12s = 15s \qquad (5.13)$$

A CT obtida utilizando este algoritmo é mostrada, em detalhes, no APÊNDICE F desta dissertação.

5.6 Conclusão

Nas seções anteriores deste Capítulo foram apresentadas as estratégias de teste da rede global e local de interconexões, bem como dos I/Os de dispositivos analógicos reconfiguráveis. Aqui são mostrados, através da Tabela 5.6, todos os resultados referentes aos números de CTs e aos tempos de teste quando falhas *stuck-ons* e *stuck-opens* são simuladas nestas estruturas, considerando que os estímulos de teste são aplicados externamente.

Tabela 5.6: Resultados finais dos testes.

CUT	**Falha**	**Nº de CTs**		**Tempo de Teste (s)**	
Interconexões Globais	*Stuck-open*	Não Bufferizado	7	21	
		Bufferizado	14	42	
	Stuck-on	1 Estímulo	3	9	
		3 Estímulos	2	6	
Interconexões Locais	*Stuck-open*	13		39	
	Stuck-on	71		213	
I/Os	*Stuck-open*	1		3	
	Stuck-on	4		12	
Total		Melhor Caso	98	Melhor Caso	294
		Pior Caso	106	Pior Caso	318

Nos cálculos dos tempos de teste foi utilizado o tempo de configuração de 3s, que corresponde ao pior caso definido para o AN10E40.

6 AUTO-TESTE DE INTERCONEXÕES

Neste capítulo é apresentada uma proposta que integra os elementos de geração e análise de teste no FPAA estudado. A idéia principal é implementar uma estrutura de teste analógico multifuncional para geração de estímulos e análise de teste utilizando os recursos internos oferecidos por este dispositivo.

Pesquisas na área de BIST digital desenvolvidas no passado propuseram o projeto de blocos multifuncionais capazes de realizar a varredura de dados (teste *scan*), de gerar padrões e compactar respostas de teste. Estas estruturas, chamadas de *Built-In Logic Observers* (BILBO), deram origem a uma versão analógica, denominada *Analog Built-In Block Observer* (ABILBO) [LUB 96], que têm características similares às do BILBO, tais como, capacidade de geração de vetores e avaliação de resposta de teste.

Em um contexto BIST, este capítulo inicia descrevendo os blocos que compõem a estrutura ABILBO proposta. Em seguida, faz-se uma análise de repetibilidade, linearidade e sensibilidade dos circuitos geradores e analisadores de teste e, por fim, aplica-se esta proposta ao teste da rede global e local de interconexões, bem como, das células de I/O do AN10E40 apresentando os resultados obtidos.

6.1 Proposta de Auto-Teste Integrado

Basicamente, a estrutura ABILBO é composta por dois integradores e alguns circuitos adicionais e pode se programada como um oscilador em anel, para a geração de estímulos, ou como um duplo integrador, para a análise de resposta de saída (ORA – *Output Response Analyser*), conforme mostra a Figura 6.1. A versatilidade de programação destes circuitos é alta: por um lado a ferramenta de desenvolvimento da *Anadigm* permite que as freqüências e amplitudes dos sinais gerados sejam definidas por *software* e, por outro, permite que as constantes de tempo τ dos integradores sejam programadas para que a resposta de teste atinja um valor de tensão de referência (V_{ref}) em um determinado tempo. Assim, uma assinatura pode ser obtida computando o tempo em que o sinal atinge este valor de tensão. Como pode ser visto na Figura 6.1a, o circuito gerador de estímulos possui um circuito adicional denominado *Dynamic Phase Swapping* (DPS) responsável pelo controle de fase do oscilador e que está disponível na biblioteca do AN10E40. Ainda nesta linha, o estudo deste circuito possibilitou que as suas funções de transferência fossem elaboradas:

$$f_{osc} = \frac{f_c}{2\pi}\sqrt{\frac{C2.C3}{CA.CB}} \qquad V_p = \frac{300.C1}{\pi C2} \qquad (6.1)$$

onde f_{osc}, V_p e f_c são a freqüência e a amplitude do sinal gerado e a freqüência de *clock* do dispositivo, respectivamente.

Já a Figura 6.1b mostra um circuito adicional e externo à estrutura de análise de respostas, um comparador, que tem por função monitorar a resposta de teste e sinalizar quando esta atingir o valor V_{ref}. Outro circuito adicional proposto, e que não é mostrado nesta figura, é um contador responsável por fornecer uma assinatura digital proporcional ao tempo desde o momento de *start up* da integração até a resposta atingir V_{ref}.

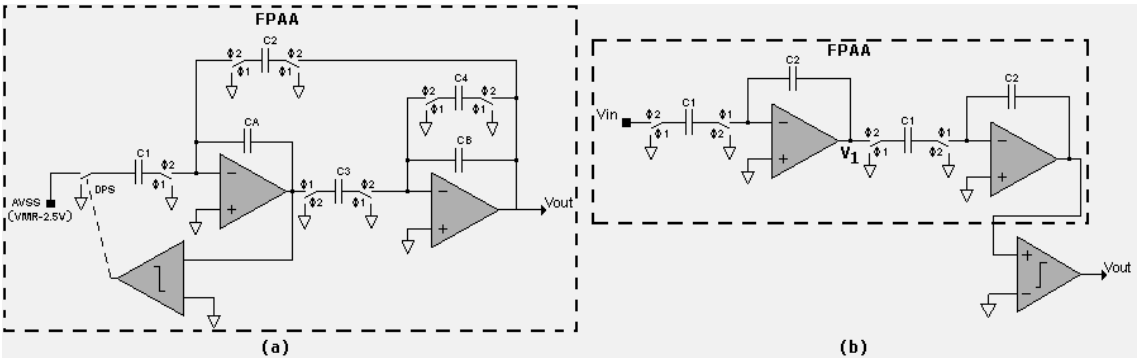


Figura 6.1: (a) Implementação do oscilador utilizando dois CABs, (b) ORA utilizando dois CABs e um comparador externo.

Ainda neste contexto, assumindo que o sinal de entrada é $V_{in} = -V_p sin(\varpi t + \phi)$ e $V_c(t=0) = 0$ para os capacitores do duplo integrador, as equações 6.2 e 6.3 fornecem o valor de tensão do sinal V_1 na saída do primeiro integrador e V_{out} na saída do segundo integrador do ORA. Considere também que $\tau = RC$ é a constante de tempo dos integradores.

$$V_1 = \frac{Vp}{(\omega\tau)}\left[\cos\phi - \cos(\omega t + \phi)\right] \qquad (6.2)$$

$$V_{out} = \frac{Vp}{(\omega\tau)^2}\left[-sin(\omega t + \phi) + \omega t.\cos\phi + sin\phi\right] \qquad (6.3)$$

A Figura 6.2 ilustra o efeito da dupla integração sobre em um sinal com V_p = 1V e freqüência igual a 50KHz. Para este exemplo foram utilizadas V_{ref} = 2,5V e τ = 10μs, então, a assinatura para o V_{in} considerado é de t = 77μs.

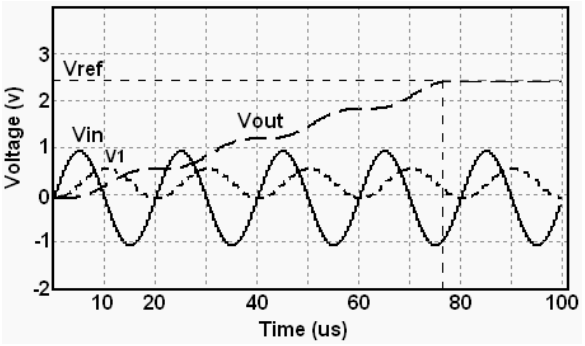


Figura 6.2: Efeito da dupla integração sobre Vin.

6.1.1 Análise de Sensibilidade do ORA

Considerando que os tempos de resposta do ORA estão intimamente relacionados com a amplitude e freqüência do estímulo utilizado e, logicamente, com os valores programados de constante de tempo, uma análise de sensibilidade deste circuito foi

desenvolvida com o intuito de limitar o espaço de projeto e, conseqüentemente, de facilitar o teste. A partir desta análise é possível definir as características dos estímulos de teste quando determinadas constantes de tempo do ORA são definidas. Considere, por exemplo, que V_{ref} = 5V e que τ é 10μs. A Figura 6.3b mostra a relação entre o tempo para atingir V_{ref} e a freqüência do estímulo aplicado. Em geral, sinais com diferentes amplitudes, porém, com freqüências iguais, exigem tempos diferentes para atingir V_{ref}. Esta característica do ORA também pode ser observada analisando-se as equações 6.2 e 6.3.

Por outro lado, a Figura 6.3c mostra que, quanto maior o valor do τ escolhido no projeto, mais sensível ao estímulo é o ORA. Esta afirmação pode ser explicada por meio do seguinte exemplo: considere que um sinal com amplitude fixa, porém, com freqüências que variam dentro de um limite determinado, seja utilizado como estímulo de teste. Pode-se observar que as assinaturas do ORA programado para atuar com τ igual a 100μs apresentam resoluções melhores se comparadas às assinaturas obtidas quando o ORA é programado para atuar, por exemplo, com τ igual a 10μs. Contudo, ao aumentar as constantes de tempo dos integradores, os tempos de teste tornam-se maiores.

Na Figura 6.3d é possível observar que, quando o ORA é programado para atuar com uma determinada constante de tempo, utilizando-se sinais com amplitudes baixas, geram-se assinaturas de tempo com resoluções melhores.

Quanto à limitação do espaço de projeto, um tempo máximo T_{max} deve ser definido, uma vez que alguns sinais exigem tempos muito longos para atingir V_{ref} e, conseqüentemente, causam *overflow* do contador. Além de T_{max}, parâmetros como V_{ref} e τ também limitam o espaço de projeto, definindo as características do sinal que deve ser utilizado como estímulo. A Figura 6.3a ilustra o espaço de entradas válidas relacionando as freqüências e as amplitudes de operação para V_{ref} = 5V, T_{max} = 500μs e valores diferentes de τ. Com isto tem-se que, todos os sinais acima de uma dada curva podem ser utilizados como estímulos quando o ORA é programado para atuar com o τ correspondente.

Almejando-se aumentar ainda mais o espaço de projeto da estrutura BIST, atribuíram-se valores diferentes às constantes de tempo dos dois integradores do ORA. Conseqüentemente, como pode ser observado na Figura 6.3e, a gama de freqüências dos estímulos utilizados aumentou consideravelmente se comparada à ilustrada na Figura 6.3a. Portanto, projetando-se, por exemplo, o ORA para operar com o primeiro integrador com τ_1 = 70μs e o segundo com τ_2 = 20μs e aplicando-se um sinal com freqüência de 20KHz e com amplitude igual a 1,5V_p, garante-se o compromisso entre a validade da assinatura, sensibilidade do ORA e tempo de teste.

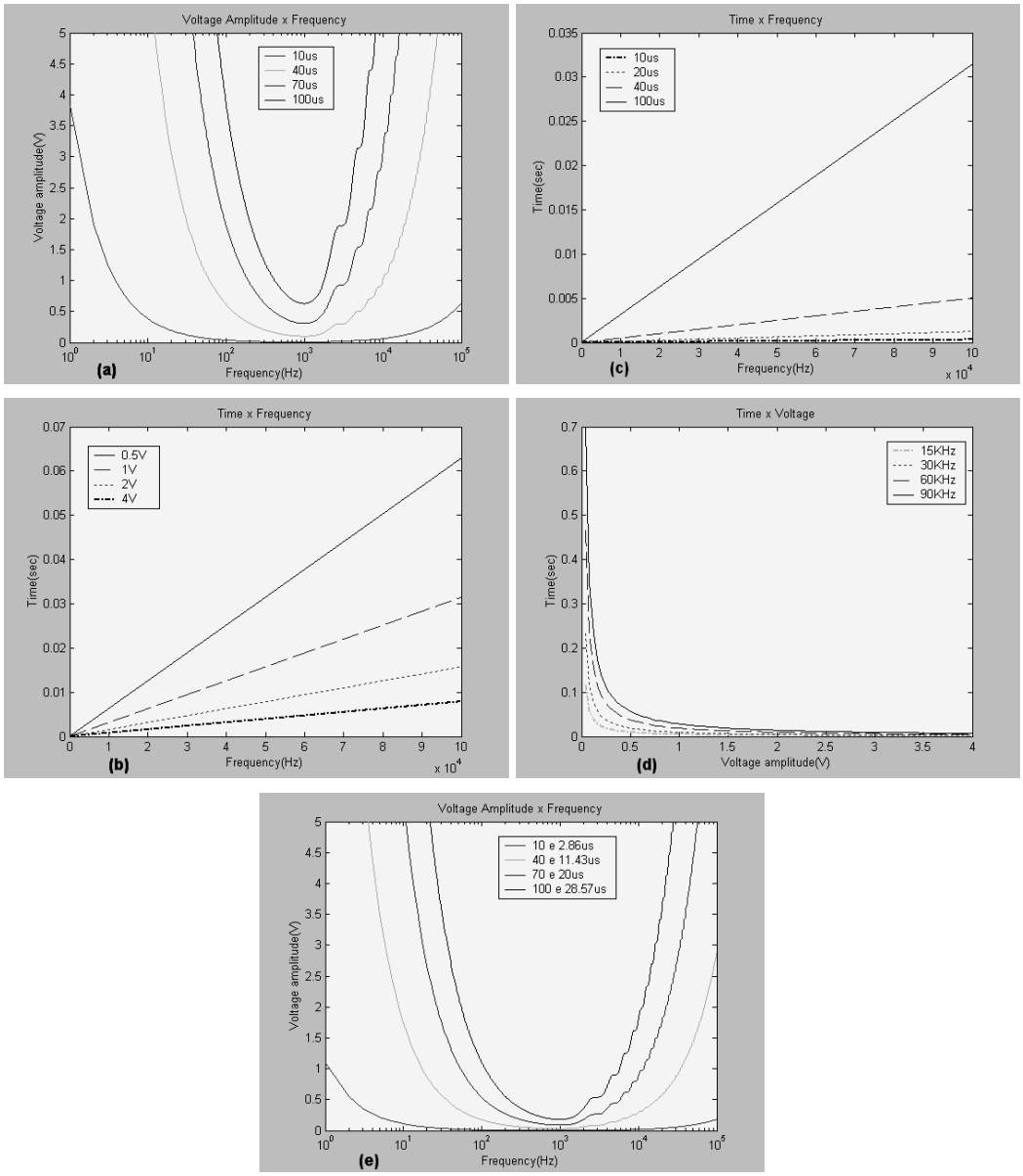


Figura 6.3: Análise de Sensibilidade do ORA.

6.1.2 Análise de Linearidade do Gerador de Estímulos

Uma vez que a utilização de estímulos com amplitudes e freqüências de oscilação diferentes garante maiores coberturas de falhas, outro procedimento importante que deve ser feito antes de iniciar o teste de circuitos analógicos é a análise de linearidade do gerador de estímulos utilizado. Com esta análise é possível observar se este circuito é confiável, ou seja, gera sinais com erros que não venham a influenciar no teste. Assim sendo, a metodologia adotada durante este experimento consiste em, primeiramente, programar o oscilador com freqüências desejadas e medir quais os valores de freqüência obtidos e, posteriormente, o mesmo procedimento é feito, porém, programando-se amplitudes diferentes para os sinais. Estas avaliações são mostradas na Figura 6.4a, onde se observa que à medida que a freqüência de oscilação do gerador aumenta, diminui a sua linearidade independente da amplitude do sinal, e na Figura 6.4b, que mostra que para freqüências elevadas a tendência é que surjam erros maiores nas amplitudes dos sinais gerados.

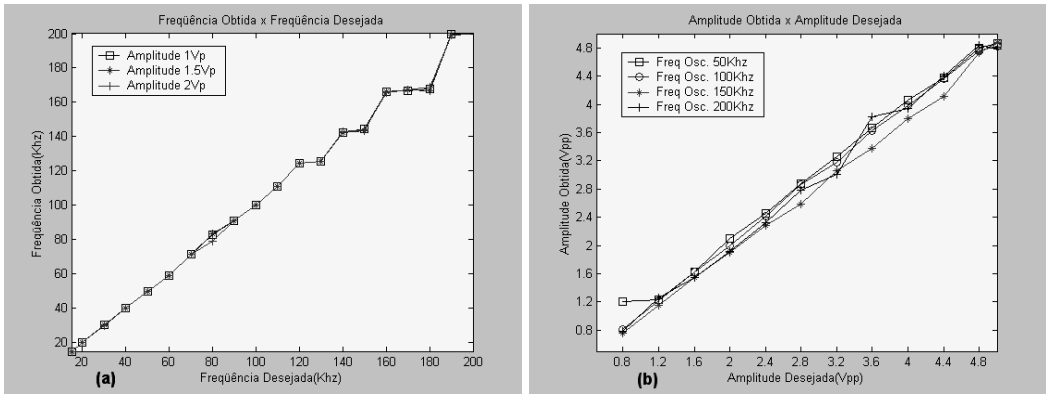


Figura 6.4: Avaliação de Linearidade do Gerador de Estímulos.

Para estas avaliações deve-se considerar as limitações do equipamento de aquisição de sinais (Tektronik TDS 210) que apresenta resolução limitada e, portanto, pode influenciar no resultado de algumas medidas.

6.1.3 Análise de Repetibilidade da Estrutura BIST

Os circuitos analógicos utilizados para análise e geração de teste devem apresentar alta precisão de medidas tal que a detecção de falhas estruturais, paramétricas e de *delay* sejam asseguradas dentro de um limite de tolerância. Assim sendo, alguns experimentos foram realizados a fim de avaliar a repetibilidade do ABILBO, variando-se as freqüências e amplitudes dos estímulos, para o caso do gerador, e as constantes de tempo, para o caso do ORA. Similarmente à avaliação de repetibilidade do PSO descrita na Seção 5.5.1, esta análise é desenvolvida programando-se 10 vezes o AN10E40 com a topologia do oscilador projetado e adquirindo-se as freqüências e as amplitudes dos sinais gerados. Já para o caso do ORA, o mesmo número de programações do dispositivo é feito, porém, adquirem-se as assinaturas de tempo resultantes. O critério de avaliação adotado é verificar se para ambos os casos ocorrem variações que estão fora de uma margem de tolerância.

Primeiramente, o procedimento foi desenvolvido sobre o oscilador utilizado para gerar padrões de teste. Foram considerados os seguintes limites de tolerância: erros de ±4% e ±10% para a amplitude e freqüência programadas, respectivamente.

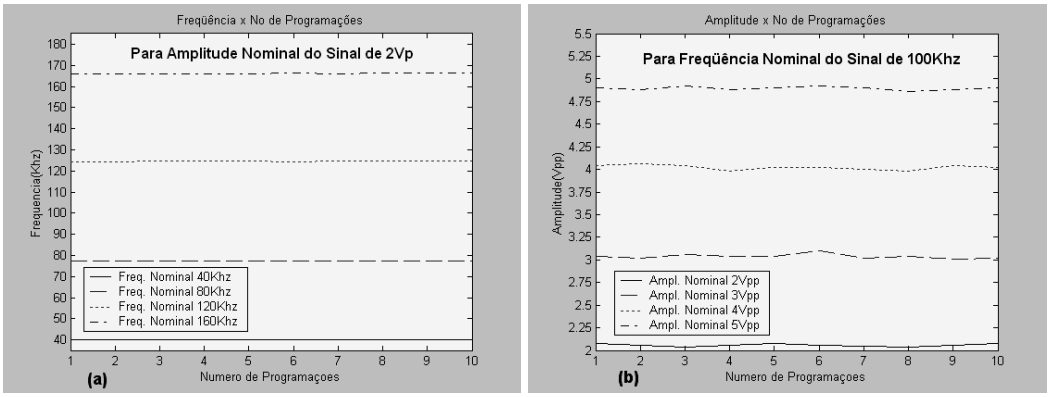


Figura 6.5: (a) e (b) Respostas do oscilador para diferentes freqüências e amplitudes, respectivamente.

A Figura 6.5a mostra que à medida que a freqüência de operação do oscilador tende a aumentar, a exatidão das medidas tende a cair. Porém, a precisão vai ao encontro das

exigências estabelecidas, uma vez que, em nenhum dos casos foram obtidos erros maiores do que 10%. Provavelmente, a perda de exatidão das medidas deve-se ao Critério de Nyquist que estabelece que a freqüência de amostragem (para este caso, 1MHz) deve ser no mínimo igual ao dobro da freqüência do sinal que se deseja amostrar, mas o que se observa na prática é que ela deve ser no mínimo 10 vezes o valor desta freqüência.

Por outro lado, a Figura 6.5b ilustra que à medida que a amplitude do oscilador tende a aumentar, a exatidão das medidas também tende a cair. No entanto, o que se observou é que a precisão das medidas está dentro dos limites previamente definidos.

A análise da repetibilidade do ORA abordou dois testes distintos: um focalizado em verificar a precisão das assinaturas de tempo quando diferentes τ dos integradores são programados, aplicando-se estímulos que vão ao encontro das análises de sensibilidade e linearidade descritas nas seções anteriores (Figura 6.6a), e outro focalizado em verificar a precisão destas assinaturas quando as constantes de tempo do integradores são $\tau_1 = 10\mu s$ e $\tau_2 = 2{,}857\mu s$, a amplitude do estímulo é $0{,}5V_p$ e diferentes freqüências de operação são escolhidas. Além disto, em uma das etapas desta última avaliação, programa-se a entrada do ORA para estar "flutuando" e verifica-se o efeito do *offset* dos amplificadores operacionais sobre a repetibilidade das respostas, conforme mostra a Figura 6.6b.

Assim como na avaliação de repetibilidade do gerador, foram definidos limites de tolerância para prováveis variações das grandezas analisadas, tal qual, admitem-se erros de ±7% dos valores nominais de assinatura de tempo.

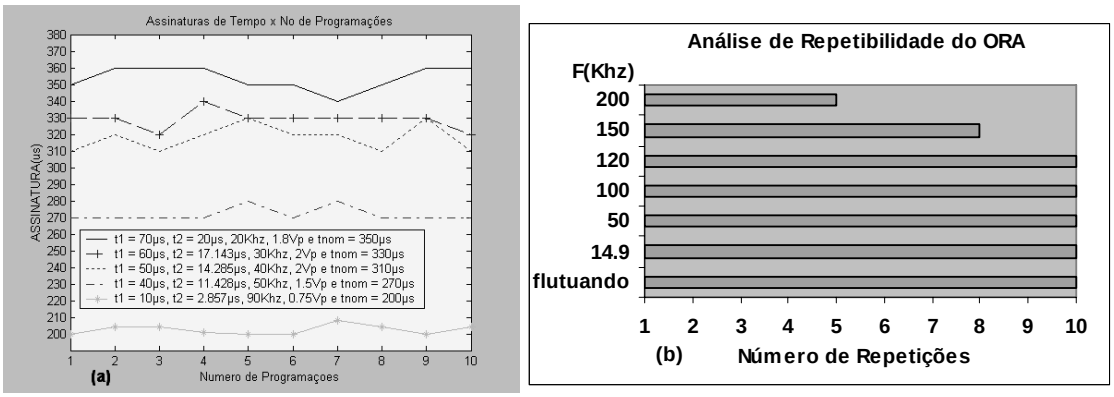


Figura 6.6: Análise de Repetibilidade do ORA.

Como pode ser visto pela Figura 6.6a, em nenhum dos casos testados o valor das assinaturas esteve fora dos limites impostos. Porém, o que pôde ser visto durante a segunda análise é que o *offset* dos amplificadores não influencia na precisão das respostas, pois o ORA apresentou alta precisão quando nenhum estímulo é aplicado na entrada, e que quando a freqüência dos estímulos de teste tende a aumentar, a repetibilidade tende a cair, conforme mostra a Figura 6.6b. Logo, ao término das avaliações, vale lembrar que, no teste que utiliza a estrutura BIST proposta, é importante que haja o compromisso entre a repetibilidade, linearidade e a sensibilidade dos elementos envolvidos e que, os resultados destas avaliações também podem ser influenciados pela baixa resolução do equipamento de aquisição de sinais (Tektronik TDS 210).

6.2 Auto-Teste da Rede de Interconexões e I/Os de FPAAs

Diferentemente do Capítulo 5 que descreve o teste externo da arquitetura do AN10E40, aqui é tratado o auto-teste integrado de cada um dos seus elementos utilizando o gerador e o analisador de teste propostos. Porém, assim como naquele capítulo, nesta Seção são descritas as estratégias de teste adotadas a fim de minimizar o número de configurações e o tempo de teste.

Assim sendo, esta Seção inicia descrevendo o auto-teste da rede global de interconexões e, por fim, apresenta o auto-teste da rede local de interconexões e das chaves e *buffers* das células de I/O.

6.2.1 Auto-Teste da Rede Global de Interconexões

Similarmente ao teste externo, duas abordagens distintas foram desenvolvidas no auto-teste integrado da rede global de interconexões: uma focada em falhas do tipo *stuck-open* e *open* nas chaves e linhas, respectivamente, e outra focada nas falhas de *stuck-on* e *bridging* destes componentes. Além disto, assim como lá, os modelos de interconexões globais desenvolvidos no Capítulo 4 foram utilizados.

6.2.1.1 Auto-Teste e Detecção de Falhas do Tipo Stuck-open e Open

A adaptação ao caso BIST do algoritmo proposto na Seção 5.3.1.1.1 é inviável, pois ele gera caminhos que englobam muitas chaves em um único segmento que podem provocar a perda de integridade do sinal e, conseqüentemente, influenciar na resposta do ORA. Entretanto, nada impede que o algoritmo proposto na Seção 5.3.1.1.2, que gera caminhos bufferizados, seja adaptado ao caso do auto-teste da rede global, uma vez que, agora a impedância "vista" pelo ORA é equivalente à impedância de apenas um dos segmentos deste caminho (Z_{seg}). Portanto, primeiramente, o número e as CTs obtidas para o teste e detecção de falhas *stuck-open* e *open* utilizando a seleção de caminhos críticos bufferizados são apresentados e, a seguir, é mostrada a resposta do ORA obtida para um determinado estímulo de teste.

6.2.1.1.1 Geração de Caminhos Bufferizados

Após a comparação entre diferentes versões do algoritmo descrito na Seção 5.3.1.1.2, conclui-se que a versão que gera menos CTs é aquela cuja origem do caminho e o primeiro destino são definidos por uma função e os demais destinos são escolhidos aleatoriamente. Naquele momento, também foram explicados os motivos pelos quais esta versão é a melhor e foram apresentadas as 14 CTs obtidas. Deste modo, estas CTs foram adaptadas ao caso BIST e as configurações resultantes desta adaptação são ilustradas no APÊNDICE G desta dissertação. A Figura 6.7 ilustra, em detalhes, uma das CTs encontradas utilizando o algoritmo, o gerador e o analisador de teste propostos e descreve a Z_{seg} para este caso.

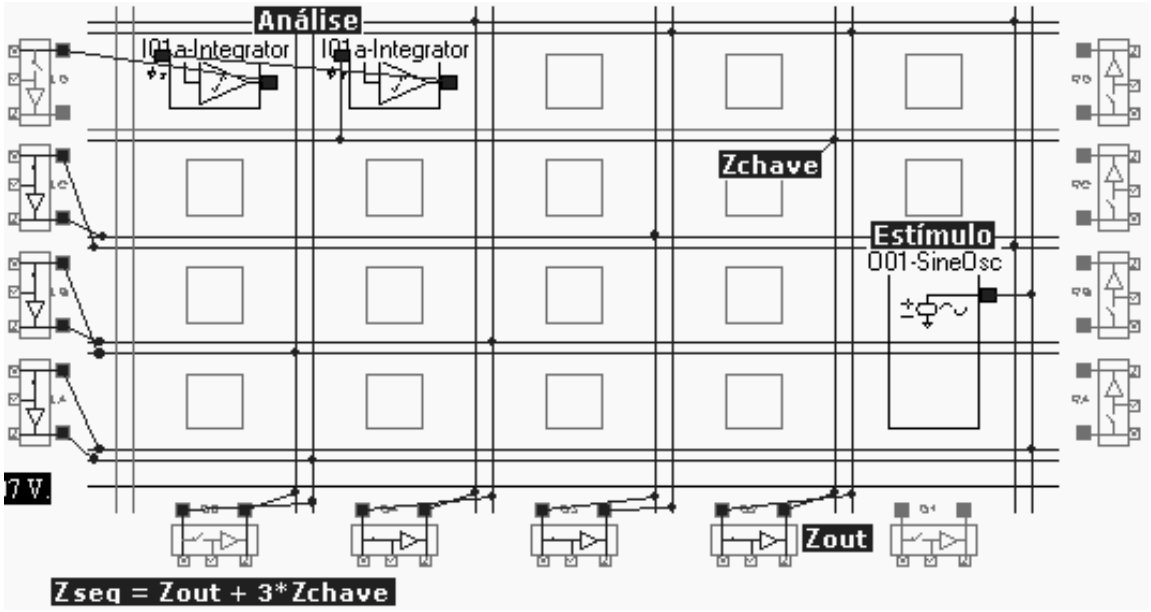


Figura 6.7: CT gerada pelo algoritmo proposto.

Depois de concluída a etapa onde foi definido o número de CTs, pode-se definir o tempo de teste destas falhas:

$$t_{stuck-open} = 3s * N^{o}CT = 3s * 14 = 42s \qquad (6.4)$$

Por fim, a resposta do ORA programado com constantes de tempo τ_1 = 40 μs e τ_2 = 11,428μs para um estímulo com amplitude igual a 1,25V_p e freqüência de 40KHz é mostrada na Figura 6.8.

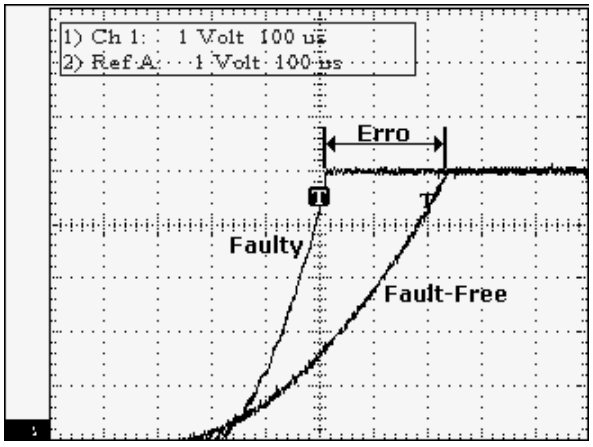


Figura 6.8: Respostas *Fault-Free* e *Faulty* ao estímulo aplicado.

Considerando que o erro obtido é de, aproximadamente, 210μs, a discriminação entre as assinaturas de tempo *faulty* e *fault-free* pode ser executada com um simples contador operando, por exemplo, à 50KHz.

6.2.1.2 Auto-Teste e Detecção de Falhas do Tipo Stuck-on e Bridging

As estratégias de teste que visam à detecção de falhas *stuck-on* e *bridging* delineadas na Seção 5.3.1.2.1 também foram adaptadas ao caso BIST. No entanto, o número de CTs utilizando os recursos internos do AN10E40 para a geração e análise de teste é maior do que o encontrado quando o teste externo é escolhido. Este acréscimo no

número de CTs deve-se às dimensões limitadas do *array* de programação para comportar a demanda de blocos configuráveis e pinos de I/O utilizados para o BIST de todo dispositivo. A Figura 6.9 mostra, em detalhes, a primeira das 3 CTs que utilizam múltiplas ondas de freqüências fundamentais diferentes para cobrir 100% das chaves da rede global do dispositivo sob teste. Todas as outras CTs multi-onda e as 5 CTs que utilizam apenas um estímulo de teste são mostradas, respectivamente, no APÊNDICE H e APÊNDICE I desta dissertação.

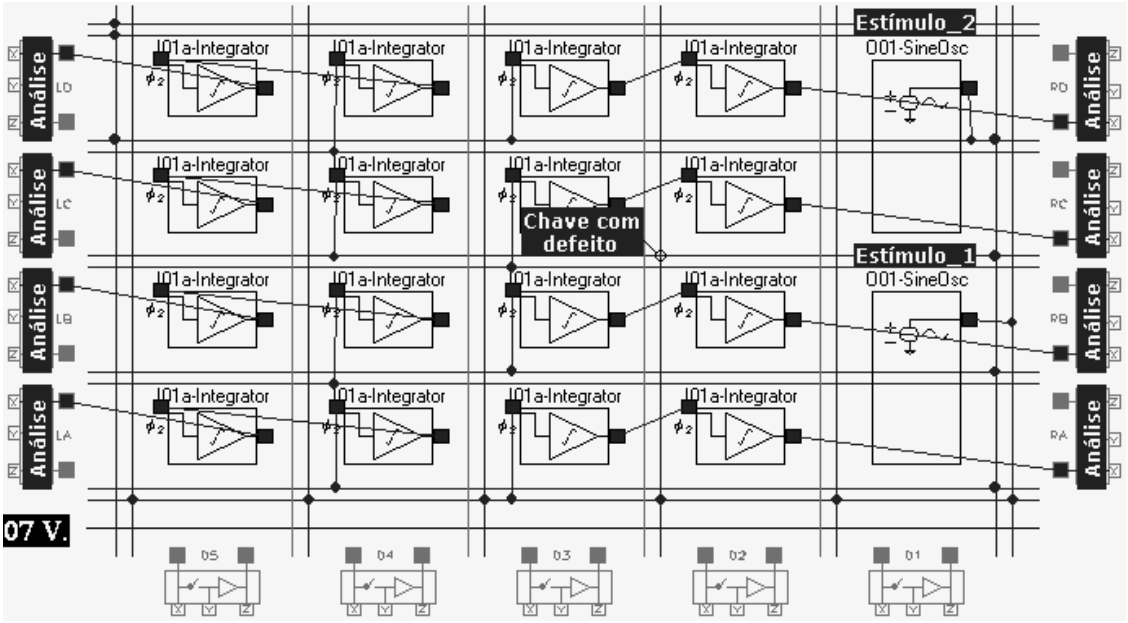


Figura 6.9: Primeira CT de falhas *stuck-on* e *bridging* da rede global.

Suponha que a chave em destaque na Figura 6.9 esteja sob o efeito de uma falha *stuck-on*. Suponha, também, que os estímulos aplicados têm freqüências de 22KHz e 50KHz, respectivamente, e amplitude igual a 1,75V_p. Em decorrência deste defeito, o sinal resultante apresenta atenuação de amplitude e o seu conteúdo harmônico aumenta devido à soma ponderada das freqüências envolvidas, conforme mostra a Figura 5.9. Logo, segundo a Figura 6.3b, a assinatura de tempo do circuito com falhas é maior do que as assinaturas dos dois estímulos de teste, como ilustra a Figura 6.10.

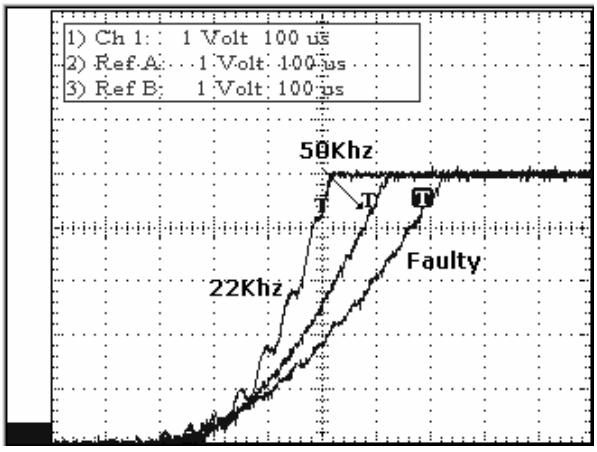


Figura 6.10: Respostas aos estímulos de 50KHz e 22KHz e com falha.

Por fim, conhecendo-se o número de CTs para cobrir 100% das falhas *stuck-on* na rede global de interconexões, pode-se estimar os tempos de teste exigidos por estas estratégias:

Tabela 6.1: Tempos de teste utilizando ambas as estratégias.

ESTRATÉGIA	TEMPO
Utilizando-se uma onda quadrada como estímulo	$t_{stuck-on} = 3s * N^oCT = 3s * 5 = 15s$
Utilizando-se múltiplas ondas quadradas como estímulo	$t_{stuck-on} = 3s * N^oCT = 3s * 3 = 9s$

6.2.2 Auto-Teste da Rede Local de Interconexões e I/Os

No auto-teste da rede local de interconexões e das chaves e *buffers* das células de I/O, utilizou-se uma estratégia similar à citada na Seção 5.5.2 que faz uso de PSOs para a geração de estímulos. Contudo, diferentemente daquela estratégia, aqui a análise de teste é realizada internamente através do ORA. Além disto, valendo-se do fato de alguns CABs serem utilizados neste teste e de existirem *buffers* disponíveis na biblioteca do AN10E40, sempre que possível, construíram-se PSOs bufferizados [MAN 2000] ao invés dos PSOs utilizados no teste externo de I/Os. O PSO utilizado é mostrado na Figura 6.11. Como vantagens à versão não bufferizada pode-se citar a maior linearidade do sinal gerado que apresenta características muito próximas a da freqüência e ganho calculados. Adicionalmente, o PSO bufferizado gera ondas com níveis de distorção mais baixos e a sua alta impedância de entrada evita que ocorram mudanças na freqüência do sinal decorrentes de variações na carga. Como nas seções 5.5.1 e 6.1.3 já foi desenvolvida a análise de repetibilidade dos blocos geradores e analisadores de teste, nesta Seção são mostrados o algoritmo desenvolvido para o teste da rede local, chaves e *buffers* das células de I/O, e, por fim, são apresentadas as CTs e o número de CTs obtido utilizando este algoritmo.

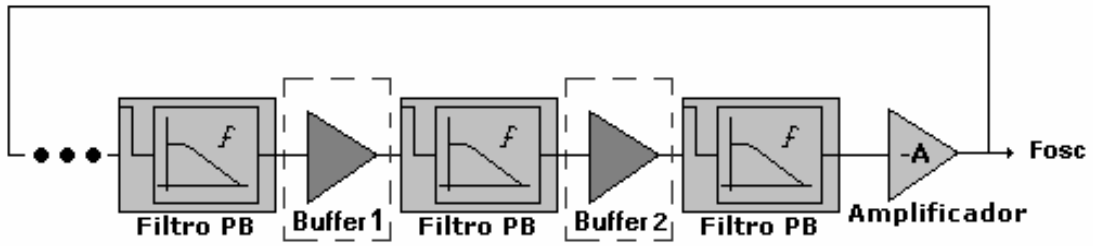


Figura 6.11: Circuito esquemático do PSO bufferizado.

6.2.2.1 Proposta de Algoritmo para o Teste da Rede Local, Chaves e Buffers I/O

Fundamentalmente, este algoritmo tem por idéia a geração de PSOs bufferizados utilizando os blocos configuráveis e/ou os *buffers* das células de I/O disponíveis na arquitetura do AN10E40. Similarmente à proposta de teste da rede global que utiliza a pesquisa e o mapeamento dos elementos envolvidos em grafos para que caminhos elétricos sejam selecionados, estes PSOs também são construídos desta forma. Entretanto, diferentemente daquele algoritmo, o algoritmo de teste da rede local gera, muitas vezes, mais do que 1 PSO por CT. Assim sendo, segue abaixo a descrição, em detalhes, do algoritmo desenvolvido para o auto-teste destes componentes:

1. A fim de limitar o espaço de pesquisa, somente os vértices que representam os CABs que podem realizar determinadas conexões são selecionados, por exemplo: primeiramente os CABs que realizam ligações com os vizinhos das posições "E" e "D" são selecionados, e, posteriormente, os CABs que realizam ligações com os vizinhos "A" e "H", conforme mostrado na Figura 4.10a. Com isto, uma CT com 2 PSOs é gerada.

2. Considerando-se que cada CT envolve 2 PSOs para a geração de estímulos e 2 ORAs são necessários para a avaliação das suas respostas, tem-se:

$$\frac{N^o\_de\_Blocos}{PSO} = \frac{N^o\_de\_Blocos\_Selecionados - N^o\_de\_CABs\_ORAs}{N^o\_de\_PSOs} \quad (6.5)$$

onde: $N^o\_de\_CABs\_ORAs = N^o$ *de CABs/ORA* * N^o *de ORAs* = 2 * 2 = 4;

Por exemplo: Se 24 dos 33 blocos (CABs + Células de I/O) do *array* realizam as ligações descritas em (1) e o algoritmo gera 2 PSOs por CT, tem-se que cada PSO engloba 10 blocos.

3. A pesquisa continua até que não exista um número suficiente de blocos adjacentes para construir um PSO. A partir daí, são construídos PSOs complementares com os blocos cujas ligações (representadas por arestas) ainda não foram testadas.

Após a implementação do algoritmo e a definição das CTs, passou-se para o procedimento de teste, onde, assim como delineado na Seção 5.4.1, a simulação de falhas é feita programando-se as chaves a serem testadas para o estado oposto ao da falha que se deseja testar. Conseqüentemente, gerando-se 2 PSOs por CT, foram necessárias 15 CTs para o teste de falhas *stuck-open* e 176 CTs para cobrir 100% das falhas *stuck-on* nas chaves da rede local. Por outro lado, quando o teste é desenvolvido para detectar estas falhas nas chaves e *buffers* das células de I/O, utilizando-se 3 PSOs por CT, são exigidas 2 CTs para cobrir falhas *stuck-open* e 5 CTs para detectar falhas *stuck-on* nestes componentes. A Tabela 6.2 e a Figura 6.12 mostram, em detalhes, os tempos de teste de cada um dos componentes citados acima e a primeira CT da rede local e das chaves e *buffers* I/O, respectivamente. As demais CTs das interconexões locais, das chaves e *buffers* I/O estão anexadas nesta dissertação (APÊNDICE J e APÊNDICE L, respectivamente).

Tabela 6.2: Tempos de teste da rede local e I/Os.

PARTE TESTADA	**TIPO DE FALHA**		**TEMPO TOTAL**
	STUCK-OPEN	***STUCK-ON***	
Interconexões Locais	45s	528s	573s
Chaves e *Buffers* I/O	6s	15s	21s

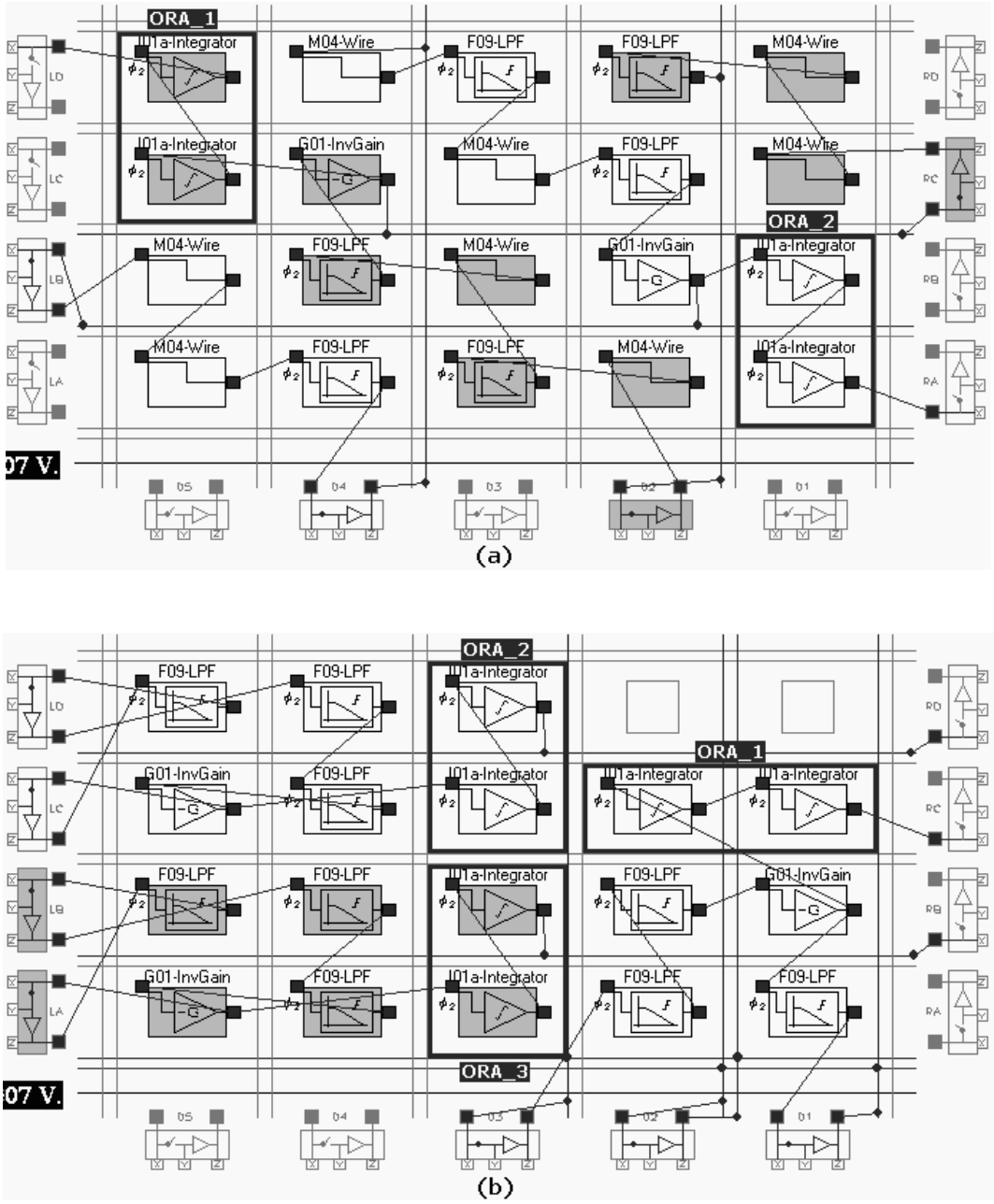


Figura 6.12: (a) e (b) Primeira CT da rede local e das chaves e *buffers* das células de I/O, respectivamente.

Nas figuras acima, os blocos em destaque correspondem a uma estrutura BIST completa: analisador e gerador de teste. As respostas sem falhas e com falha *stuck-open* do ORA_1 ilustrado na Figura 6.12a, que é programado com constantes de tempo τ_1 = 40 μs e τ_2 = 11,428μs, a um estímulo que possui amplitude de 1,25V_p e freqüência igual a 33KHz, são mostradas na Figura 6.13. O $A\beta$ do PSO é equivalente a 0,4375, a Fo é igual a 18,5KHz e, conseqüentemente, a freqüência de oscilação calculada é de 32,042KHz.

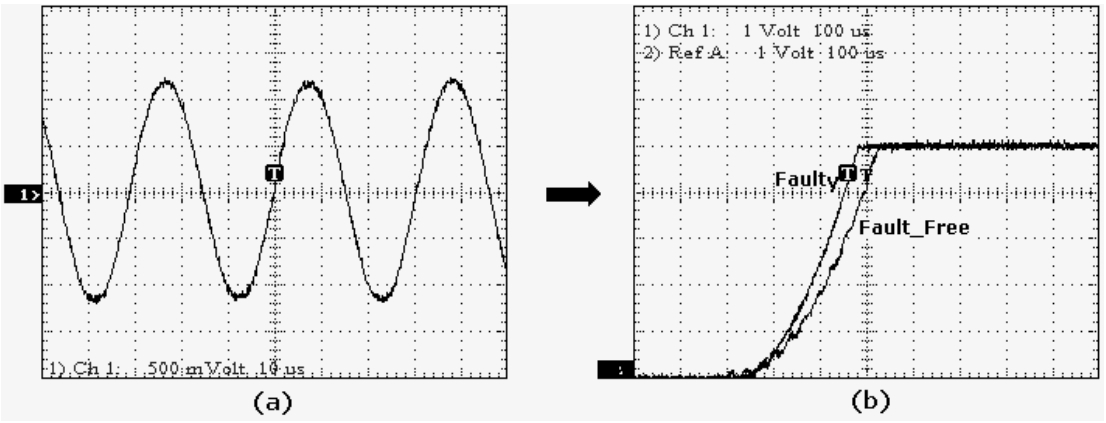


Figura 6.13: (a) Estímulo gerado pelo PSO com $A\beta$ = 0,4375 e Fo = 18,5KHz. (b) Respostas *Faulty* e *Fault-Free* ao estímulo aplicado.

Neste caso, como o gerador de estímulos é um PSO bufferizado, a freqüência obtida (aproximadamente, 33,3KHz) tem valor muito próximo ao valor calculado.

6.3 Conclusão

Nas seções anteriores deste Capítulo foram apresentadas as estratégias de teste da rede global e local de interconexões, bem como dos I/Os de dispositivos analógicos reconfiguráveis. Aqui são mostrados, através da Tabela 6.3, todos os resultados referentes aos números de CTs e aos tempos de teste quando falhas *stuck-ons* e *stuck-opens* são simuladas nestas estruturas, considerando que a geração e análise de teste são realizadas internamente ao CUT.

Tabela 6.3: Resultados finais dos testes.

CUT	FALHA	Nº de CTs		TEMPO DE TESTE (s)	
Interconexões Globais	*Stuck-open*	14		42	
	Stuck-on	1 Estímulo	5	15	
		3 Estímulos	3	9	
Interconexões Locais	*Stuck-open*	15		45	
	Stuck-on	176		528	
I/Os	*Stuck-open*	2		6	
	Stuck-on	5		15	
Total		Melhor Caso	215	Melhor Caso	645
		Pior Caso	217	Pior Caso	651

Nos cálculos dos tempos de teste foi utilizado o tempo de configuração de 3s, que corresponde ao pior caso definido para o AN10E40.

7 AUTO-TESTE E DETECÇÃO DE *CROSSTALK*

Pretende-se, nesta seção, abordar alguns conceitos, segundo a visão de alguns autores, de modelos de capacitâncias parasitas, que representam os elementos críticos no projeto, uma vez que, introduzem, dentre outros defeitos, defeitos de *crosstalk*. Em seguida, são discutidos os efeitos provocados por estes defeitos e os modelos de falhas abordados. Por fim são apresentadas as arquiteturas dos circuitos propostos para controle, geração e análise de teste, bem como, os resultados práticos obtidos nos testes.

7.1 Introdução

Já não é possível desprezar a contribuição da rede de interconexões que introduz uma gama elevada de elementos parasitas nos *layouts* de diversos circuitos, de diferentes tecnologias e configurações. Enquanto "fios" ideais não afetam a performance do circuito como um todo, um "fio" real em um circuito ou sistema integrado introduz capacitâncias, resistências e indutâncias parasitas que podem ter influência dominante na operação do circuito. Os efeitos parasitas aumentam quando as dimensões dos dispositivos diminuem, sendo agravados em conseqüência do melhoramento da tecnologia. Isto provoca o aumento do comprimento médio dos "fios" de interconexões que, por sua vez, são os principais portadores de efeitos parasitas.

As interconexões introduzem três tipos de elementos parasitas: capacitâncias, resistências e indutâncias. Estes são responsáveis pelas seguintes falhas nos circuitos:

- Introdução de ruído que afeta a confiabilidade do sistema ou circuito;
- Atrasos de propagação de sinais.

Pode-se citar como exemplos de sistemas ou circuitos vulneráveis a estes efeitos, os SOCs e *chips* que utilizam tecnologia DSM (do inglês, *deep submicron*), que possuem redes de interconexões críticas e determinantes para seu desempenho e confiabilidade. Suas interconexões e barramentos longos são susceptíveis a defeitos de *crosstalk* que podem provocar falhas de *timing* no sistema. Portanto, o teste visando à detecção de possíveis erros provocados por estas falhas, torna-se, também, crítico para estes circuitos.

7.2 Modelos de Capacitâncias Parasitas

Diversos trabalhos na área de modelagem de elementos parasitas têm sido produzidos. No entanto, ainda não há um consenso sobre alguns pontos fundamentais,

como por exemplo, a formalização matemática que melhor represente os efeitos parasitas produzidos por estes componentes. Isto se deve à complexidade das equações e à dificuldade de elaborar modelos que considerem fatores intrínsecos que muitas vezes são imperceptíveis. Dentre os elementos parasitas, as capacitâncias da rede de interconexões são, sem dúvida, os mais importantes quando os circuitos CMOS com alta velocidade de chaveamento são tratados. Geralmente, os projetistas se preocupam somente com elas ao projetar circuitos CMOS não muito complexos. O aumento da carga capacitiva de *gate* aumenta o *delay* de propagação de sinais e introduz ruídos que podem influenciar no comportamento do sistema ou circuito. Além disto, as capacitâncias entre fios introduzem efeitos de *crosstalk* que, por sua vez, afetam a robustez do projeto.

Uma maneira de modelar as capacitâncias de interconexões, descrita em [RAB 96], é através de capacitores de placas paralelas (Figura 7.1). Se a largura do "fio" é suficientemente maior do que a espessura do material isolante, pode-se assumir que as linhas do campo elétrico são ortogonais às placas do capacitor. Sob estas circunstâncias, e assumindo que o SiO_2 é o material isolante, a capacitância total do "fio" pode ser modelada como:

$$C_{\text{int}} = \frac{\varepsilon_{ox}}{t_{ox}} WL \qquad (7.1)$$

onde W e L são a largura e o comprimento do "fio" respectivamente, e $\varepsilon_{ox} = 3.97\ \varepsilon_o = 3.5 \times 10^{-11}$ F/m representa a permissividade do óxido. Os "fios" são geralmente roteados sobre o campo do óxido que é substancialmente menor do que o óxido do *gate*, resultando em capacitâncias pequenas por unidade de área. Um grupo de capacitâncias típicas é dado na Tabela 7.1 que contém valores de capacitância por unidade de área para camadas de roteamento projetadas em uma tecnologia CMOS 1µm típica.

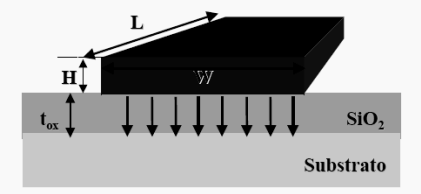


Figura 7.1: Modelo de placas paralelas.

Tabela 7.1: Capacitância por unidade de área para circuitos projetados na tecnologia 1µm.

INTERCONEXÕES	fF/µm2
Poli silício ao substrato	0.058 ± 0.004
Metal 1 ao substrato	0.031 ± 0.001
Metal 2 ao substrato	0.015 ± 0.001
Metal 3 ao substrato	0.010 ± 0.001
Difusão n+ ao substrato (@ 0V)	0.36 ± 0.02
Difusão p+ ao substrato (@ 0V)	0.46 ± 0.06

Fonte: DIGITAL INTEGRATED CIRCUITS, 1996.

A redução da tecnologia por um fator S reduz muitos parâmetros tais como a largura W e a espessura t_{ox} ao longo de uma mesma linha. Entretanto, o mesmo não ocorre com a dimensão L do "fio" que depende da localidade dos módulos a serem ligados. A rede de interconexões de um sistema pode ser dividida em duas partes principais: local,

intramódulos, e global que possibilita ligações entre módulos distantes. Na Figura 7.2 é ilustrada a distribuição típica dos comprimentos dos "fio" em um *chip*. A distribuição tem dois picos, um em torno de $0.1*L_D$ que representa a rede local, e outro em torno de $0.5*L_D$ representando a rede global. L_D é proporcional à área utilizada para projeto em um *chip* ($0.5*A_D$, onde A_D é a área do *chip*). Fórmulas empíricas foram elaboradas para expressar o comprimento médio dos "fios" da rede global, dentre elas pode-se citar a Equação 7.2 proposta por Sorkin.

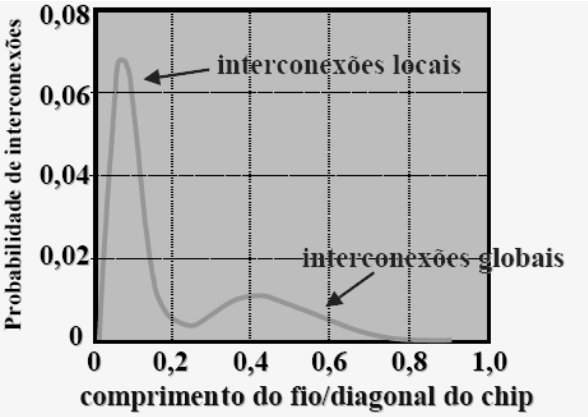


Figura 7.2: Distribuição de Interconexões (DIGITAL INTEGRATED CIRCUITS, 1996).

$$L_{av} = \frac{\sqrt{A_D}}{3} \tag{7.2}$$

Onde L_{av} corresponde ao comprimento médio do fio.

Portanto, o comportamento da escala da capacitância do "fio" pode ser expressa através do parâmetro $S_{c,fio}$ que é função do fator de escala S e do fator de escala de comprimento do "fio" S_L:

$$S_{C,fio} = \frac{S * S_L}{S} = S_L \tag{7.3}$$

ou, simplesmente através do fator de escala de comprimento S_L.

O comprimento dos "fios" da rede local é proporcional ao fator de escala S da tecnologia utilizada, sendo que quando a densidade de dispositivos aumenta, o comprimento dos fios tendem a diminuir. Por outro lado, o comprimento médio dos "fios" da rede global é proporcional à área utilizada do *chip*, como descrito na Equação 7.2. Com a melhoria da qualidade dos materiais, bem como das técnicas de fabricação, é possível tornar a área utilizada para projeto ainda maior.

Um outro modelo mais atual utilizado para representar as capacitâncias parasitas é o modelo de capacitância de bordas que considera não só a capacitância das placas paralelas, mas, também, a capacitância das bordas, sendo, portanto, mais completo (Figura 7.3).

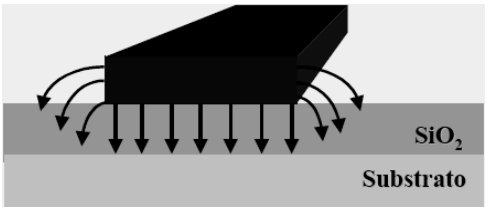


Figura 7.3: Componentes do campo elétrico (DIGITAL INTEGRATED CIRCUITS, 1996).

A capacitância total é aproximada pela soma das duas componentes, uma correspondendo à capacitância das placas paralelas determinada pelo campo ortogonal entre um "fio" de largura W-H/2 e *ground*, e a outra correspondente à capacitância das bordas definida por um "fio" cilíndrico com espessura H igual a da interconexão (Figura 7.4). O modelo analítico resultante para a C_{borda} é complexo. Uma fórmula empírica para determinar C_{int} com relativa precisão e eficiência é dada por:

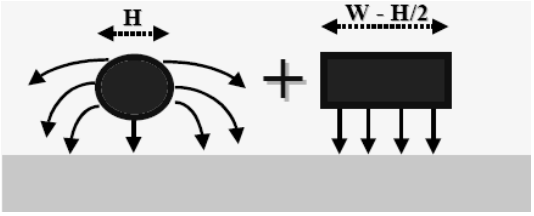


Figura 7.4: Modelo de capacitância de bordas.

$$C_{\text{int}} = \varepsilon_{ox} L \left[\left(\frac{W}{t_{ox}} \right) + 0.77 + 1.06 \left(\frac{W}{t_{ox}} \right)^{0.25} + 1.06 \left(\frac{H}{t_{ox}} \right)^{0.5} \right] \qquad (7.4)$$

Além disto, tabelas são utilizadas para determinar a contribuição da capacitância das bordas no projeto. Valores típicos destas capacitâncias por unidade de comprimento para uma tecnologia 1μm são delineados na Tabela 7.2. Através da Figura 7.5 que mostra os valores das capacitâncias em função de W/H, é possível observar que para valores elevados de W/H a capacitância total corresponde ao modelo de placas paralelas, porém, quando W/H é menor do que 1.5, as capacitâncias das bordas passam a serem dominantes. A capacitância das bordas pode aumentar a capacitância total por um fator entre 1,5 e 3 para linhas de largura menor.

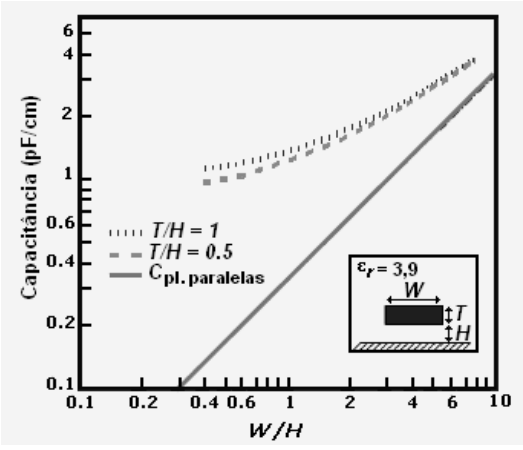


Figura 7.5: Capacitância de interconexões em função de W/H, incluindo os efeitos do campo das bordas (DIGITAL INTEGRATED CIRCUITS, 1996).

Tabela 7.2: Capacitância das bordas para processo CMOS 1μm.

INTERCONEXÕES	CAPACITÂNCIA DAS BORDAS
Poli silício ao substrato	0.043 ± 0.004
Metal 1 ao substrato	0.044 ± 0.001
Metal 2 ao substrato	0.035 ± 0.001
Metal 3 ao substrato	0.033 ± 0.001

Fonte: DIGITAL INTEGRATED CIRCUITS, 1996.

7.3 Defeitos Crosstalk

Acoplamentos incorretos entre "fios" e nós da rede de interconexões introduzem interferências que são chamadas de *crosstalk*. O distúrbio provocado por este defeito atua como fonte de ruídos que, por sua vez, pode induzir o sistema a erros intermitentes, uma vez que a injeção de ruído depende dos valores de transição dos outros sinais roteados na vizinhança. Em circuitos integrados, este acoplamento intersinais pode ser capacitivo ou indutivo. Entretanto, como foi mencionado nas seções anteriores, o efeito de capacitâncias parasitas é dominante quando se trata de defeitos *crosstalk*.

Obviamente, não é uma boa idéia projetar capacitâncias entre dois "fios" com valores super dimensionados, caso os defeitos de *crosstalk* sejam indesejados em um sistema ou circuito. No entanto, em projetos que exijam área reduzida, distâncias longas entre "fios" vizinhos tornam-se impraticáveis. O que se observa na prática para evitar estes problemas é o projeto de sistemas com dois ou mais *clocks* distribuídos em sistemas com mais de uma fase ou, a definição de distâncias máximas possíveis entre dois "fios" ou, ainda, a inserção de blindagens (do inglês, *shieldings*), correspondentes a *grounds* ou a *Vdds*, entre dois fios (Figura 7.6).

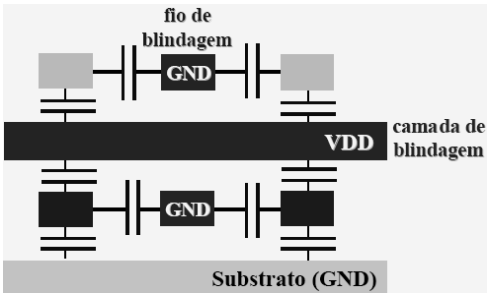


Figura 7.6: Blindagens para evitar o efeito das capacitâncias de *crosstalk*.

A não proporcionalidade de escala entre as dimensões verticais e horizontais do "fio" resulta no aumento da capacitância entre "fios". A espessura H do "fio" se mantém constante, enquanto que à distância D entre os "fios" na mesma camada ou em camadas diferentes de roteamento é reduzida (Figura 7.7).

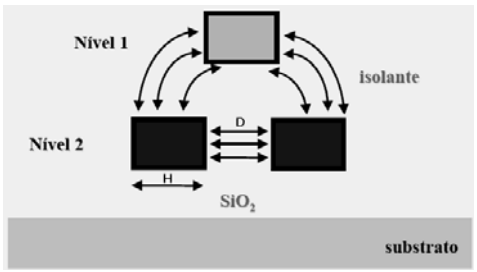


Figura 7.7: Capacitância entre fios.

Enfim, conforme foi descrito anteriormente, para sanar o problema de super dimensionamento da capacitância entre "fios", deve-se adicionar camadas de roteamento extra entre "fios" vizinhos a fim de diminuir sua magnitude.

7.3.1 Propriedades e Efeitos *Crosstalk*

Três são os efeitos adversos do aumento da capacitância e indutância entre fios na integridade do sinal:

1. Quando a capacitância de acoplamento corresponde a um parâmetro de primeira ordem entre duas interconexões, dois efeitos podem ser observados quando um sinal *step* é inserido na entrada da rede de interconexões:

a) No primeiro caso, quando um sinal é chaveado (por exemplo, X_1 é chaveado para nível alto na Figura 7.8) e o outro é mantido em regime permanente (X_2 é mantido em "0") a energia é transferida através da capacitância de acoplamento resultando em um *glitch* no sinal que está em regime permanente (X_2). Este efeito é mostrado na Figura 7.8a.

b) A segunda anomalia ocorre quando as duas linhas são chaveadas para valores opostos (por exemplo, X_1 é chaveado para nível alto e X_2 é chaveado para nível baixo). Neste caso, ocorre o aumento do tempo de transição dos sinais, conforme mostra a Figura 7.8b.

2. Quando a indutância é combinada com outros elementos do modelo do circuito, as funções de transferência da tensão resultantes geralmente equivalem a equações diferenciais de ordem elevada. Conseqüentemente, além de *glitches* e *delays*, são observadas oscilações de tensão superpostas a estes efeitos (Figura 7.8c). Todavia, neste trabalho são considerados somente os efeitos causados pelas capacitâncias de acoplamento.

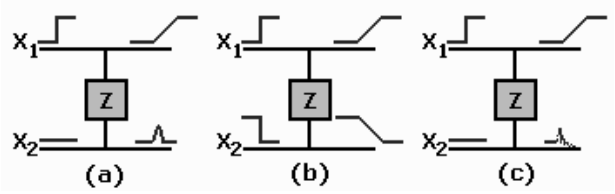


Figura 7.8: Efeitos *crosstalk*: (a) *glitch*, (b) *delay* e (c) oscilações.

7.4 Modelo de Falhas

A partir da descrição acima, pode-se classificar os efeitos de *glitch* e *delay* de acordo com o estado atual dos sinais que percorrem linhas vizinhas:

1. Quando o sinal está em regime permanente e em nível alto em uma das linhas da rede (X_2) e os sinais nas linhas vizinhas (X_1, X_3, X_n) transicionam fortemente do nível alto para o nível baixo, ele induzem *glitches* negativos (g_n) em X_2 (Figura 7.9a). Uma vez que X_2 é afetada, podendo causar erros de operação no sistema, ela é chamada de vítima e as outras linhas são chamadas de agressoras, conforme a estratégia adotada em [BAI 2000], [BAI 2001] e [CHE 2001].

2. Analogamente, o *glitch* com polaridade inversa a do gerado em (1), é chamado *glitch* positivo (g_p). Este efeito pode ser visto na Figura 7.9b;

3. Se a transição de subida do sinal é atrasada devido à capacitância de acoplamento, o efeito é denominado de *delay* de subida (d_s). Neste caso, o efeito descrito pode ser observado na Figura 7.9c;

4. Por outro lado, quando a transição de descida do sinal é atrasada devido à capacitância de acoplamento, o efeito é denominado de *delay* de descida (d_d). Na Figura 7.9d é possível notar este efeito.

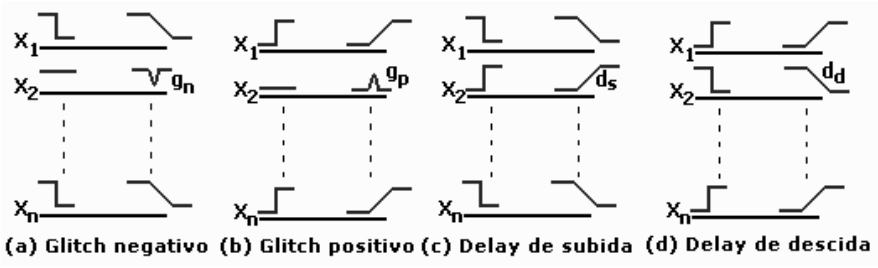


Figura 7.9: Quatro efeitos induzidos pelas capacitâncias de acoplamentos.

Para simular defeitos de *crosstalk*, que podem resultar em erros intermitentes no sistema, é necessário que sejam definidos estes erros a partir dos parâmetros elétricos intrínsecos do receptor e da tecnologia utilizada no seu projeto. Quando *glitches* positivos (g_p) são considerados, amostras deste efeito podem ser adquiridas sob a seguinte condição:

- A amplitude do *glitch* deve ser maior do que o valor de *threshold* V_{pth}, durante Δt_g, conforme é mostrado na Figura 7.10. Isto significa que o *glitch* deve ter magnitude e largura suficiente para ser reconhecido pela janela de amostragem do receptor. Neste caso, os parâmetros V_{pth} e Δt_g são determinados pelo valor de *threshold* e pelo tempo de *start up* do receptor.

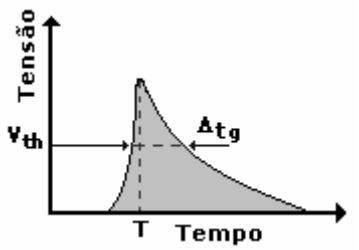


Figura 7.10: Definição de *glitch* positivo.

Segundo as bibliografias [CHE 2001] e [RAB 96], os efeitos descritos anteriormente aumentam monotonicamente com o aumento também monotônico da capacitância acoplada à vítima e às linhas agressoras.

7.5 Proposta de uma Arquitetura para Detecção de Defeitos *Crosstalk*

Esta seção inicia apresentando na Figura 7.11 o diagrama em blocos da arquitetura que foi desenvolvida neste trabalho. Logo após, faz-se uma descrição detalhada dos componentes dos blocos lógicos propostos, responsáveis pelo controle e geração de vetores de teste, do circuito receptor embarcado no AN10E40, bem como, da função de cada um dentro deste contexto.

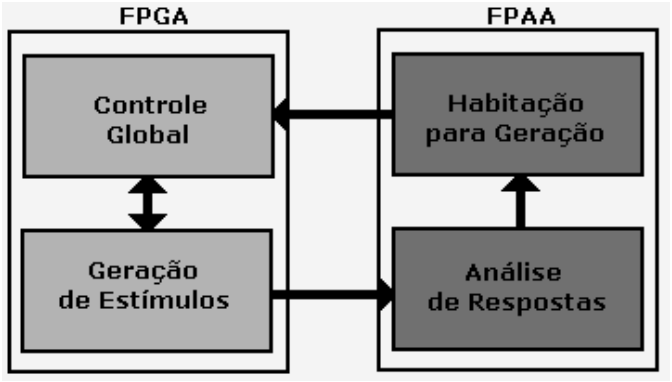


Figura 7.11: Arquitetura Proposta.

7.5.1 Arquitetura dos Blocos Lógicos

Para a implementação das funções lógicas das células bases responsáveis pela geração de estímulos e controle global das tarefas, procuraram-se soluções que viessem a atender as tarefas referentes à geração da seqüência de vetores de teste necessária para cobrir as falhas descritas na Seção anterior e às exigências de sincronismo e controle necessárias neste processo.

Além disto, tinha-se como meta evitar desperdícios de área que podem representar custos elevados de projeto. Para isto, optou-se pela utilização de alguns blocos lógicos cujas áreas são proporcionais ao logaritmo do número N de linhas a serem testadas, tais como contadores e decodificadores (ambos com áreas proporcionais a $\log_2 N$). Durante a implementação destes blocos foi necessária à utilização de portas lógicas responsáveis pelas tarefas de decodificação e multiplexação que realizam a seleção de linhas a serem testadas. A configuração e o sincronismo dos blocos lógicos são feitos através de um controlador global que consiste, basicamente, de uma máquina de estados.

7.5.2 Arquitetura do Circuito Receptor

O projeto do circuito receptor visa à verificação correta das transições dos vetores gerados, garantindo, portanto, a confiabilidade do teste. O circuito utilizado para realizar esta tarefa encontra-se totalmente embarcado no dispositivo que se deseja testar e, assim como o analisador de auto-teste da rede global, local e I/Os proposto no Capítulo 6, é composto de dois circuitos integradores que são programados para que os estímulos de teste atinjam um nível de tensão de referência V_{ref} após a dupla integração. Além disto, utilizando-se alguns circuitos adicionais, tais como um comparador e um circuito projetado para habilitar a geração de seqüências de vetores de teste, garante-se o sincronismo e a eficácia da arquitetura proposta.

7.5.3 Gerador de Estímulos de Teste

Para cada falha g_p, g_n, d_s e d_d, em uma linha vítima de um barramento de largura n-*bits* é necessário que seja gerada uma seqüência de teste de dois vetores. Contudo, os vetores podem ser sobrepostos tal que uma seqüência de seis vetores de teste é suficiente para testar todas as quatro falhas possíveis em cada uma das vítimas da rede de interconexões [BAI 2000]. A Figura 7.12 mostra a redução do número de vetores decorrente desta sobreposição. Esta seqüência de teste possui muitas propriedades interessantes que possibilitam que sejam projetados geradores de teste e analisadores de erros eficientes e de baixo custo. Por exemplo, para cada vetor de teste de n-*bits* existem

apenas dois valores distintos a serem analisados: o valor da vítima e o valor que é idêntico em cada uma das linhas agressoras.

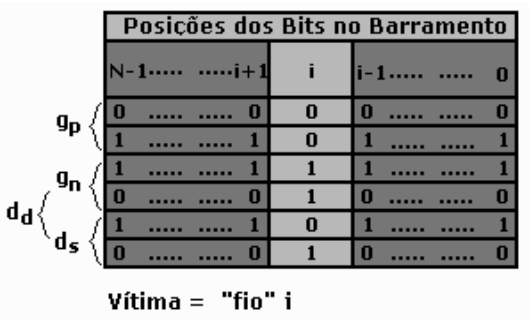


Figura 7.12: Seqüência de Vetores de Teste.

A Figura 7.13 mostra o projeto do gerador de vetores de teste desenvolvido baseado nas propriedades citadas acima. Uma máquina de estados finita gera os dois valores distintos, para a vítima e para as linhas agressoras, em cada um dos estados st1 a st8. Uma transição de st8 para st1 corresponde à geração dos seis vetores de teste que são aplicados a uma vítima e às linhas agressoras. A configuração das linhas é feita por um contador, que conta de 0 a N, atribuindo, desta maneira, a função de vítima a cada uma das N linhas do barramento, e o decodificador, que é responsável pela seleção (através do sinal q_i) da linha que deve ser a vítima e das linhas que devem atuar como agressoras. O contador de vítimas é inicializado através de um sinal (**rst_ctr** = "0") enviado pela máquina de estados finita quando recebe um sinal (**start** = "1"), que habilita a geração dos estímulos de teste do controlador global. Durante cada transição de st8 para st1, a máquina de estados finitos permite que o contador selecione a próxima vítima enviando-lhe um sinal de **enable** em nível alto. Cada vez que o gerador recebe um sinal do controlador (**ptr_inc** = "1"), informando que a resposta aos estímulos de teste atuais já foi armazenada, a máquina de estados migra para a seqüência correspondente aos dois próximos vetores. Depois de completar N seqüências de seis vetores, a máquina de estados recebe um sinal (q_{N-1}) do decodificador indicando que a última vítima foi testada e, por conseguinte, retorna ao st0 para aguardar um novo sinal de **start** do controlador global.

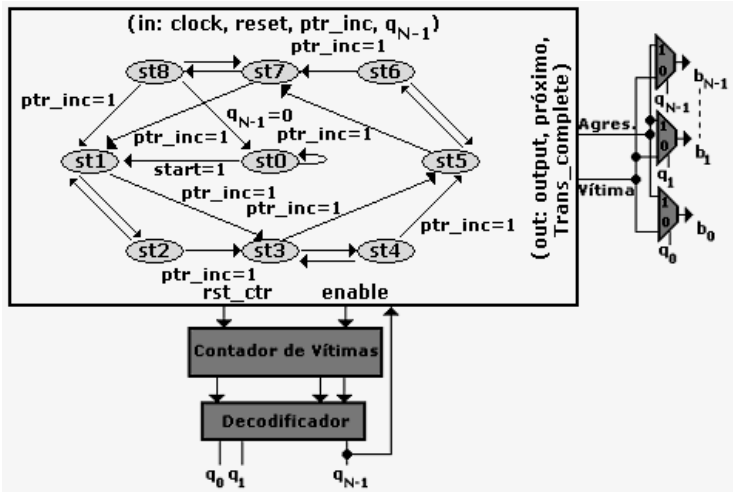


Figura 7.13: Circuito gerador de estímulos de teste.

Para o caso dos circuitos analógicos programáveis, propõe-se que a célula base responsável pela geração de vetores de teste seja projetada para atuar externamente à

sua arquitetura. Conseqüentemente, as possíveis falhas de *crosstalk* que venham a ocorrer na sua rede de interconexões são estimuladas, possibilitando, assim, que elas sejam percebidas pelo analisador de respostas.

7.5.4 Controlador Global

O controlador global tem por função garantir o sincronismo entre as operações de geração e análise de respostas de teste. Basicamente, este circuito é formado por uma máquina de estados finita que, dentre outras funções, define quando o sistema deve entrar no modo teste e quando o processo de geração deve ser iniciado, enviando-lhe um sinal (**ptr_inc**) em nível alto que habilita o incremento do contador de vítimas logo após receber do circuito que habilita a geração de vetores um sinal **Next_trans** = "1". A Figura 7.14 ilustra o projeto da máquina de estados finita desenvolvida neste trabalho.

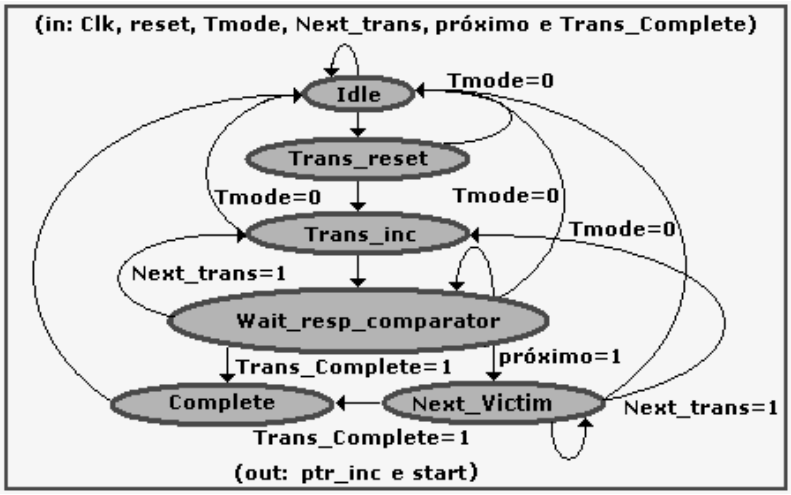


Figura 7.14: Projeto do controlador global.

Quando o circuito está em modo teste (**Tmode** = "1"), a máquina de estados finita que está em um estado ocioso (**Idle**) migra para um estado (**Trans_reset**) que faz com que o gerador inicie suas tarefas (**start** = "1"). Logo após, a máquina migra para um estado (**Trans_Inc**) que permite que o módulo gerador incremente um vez o contador de vítimas (**ptr_inc** = "1"). A partir daí, cada vez que o controlador receber um sinal (**Next_trans** = "1") do circuito que habilita a geração de vetores indicando que uma comparação já foi realizada, ou um sinal (**próximo** = "1") do gerador informando que uma nova vítima será testada, a máquina de estados migra do estado de espera (**Wait_resp_comparator**) para o estado que admite a seleção de duas novas seqüências de vetores de teste (**Trans_Inc**) ou para o estado que informa que uma nova vítima será testada (**Next_Victim**), respectivamente. Todavia, caso o controlador receba um sinal (**Trans_Complete** = "1") do bloco gerador indicando que todas os teste já foram realizados, o processo é finalizado (**Complete**).

Assim como o gerador de vetores de teste, a célula base responsável pelo sincronismo das operações de geração e análise de respostas de teste foi projetada para atuar externamente à arquitetura do circuito analógico programável estudado.

7.5.5 Analisador de Respostas de Teste

Para esta proposta, o analisador de respostas de teste tem como função verificar se as transições dos pares de vetores estão corretas. Este circuito é constituído, basicamente, por dois integradores que são programados com constantes de tempo que possibilitam

que os estímulos de teste atinjam um valor de tensão de referência V_{ref} em um tempo menor que T_{max}. Caso exista algum erro resultante de uma falha na rede de interconexões de FPAAs percebida por este circuito, é possível localizar onde existe o defeito, uma vez que, os resultados podem ser armazenados para posterior apreciação.

Além do analisador de teste, alguns circuitos adicionais são necessários, como por exemplo, um registrador que armazena os tempos de resposta do analisador quando determinados estímulos são utilizados (Figura 7.15a) ou um gerador de vetores de teste local, idêntico ao gerador fonte mostrado na Seção anterior, que é encarregado de gerar padrões de teste que são comparados aos vetores que provêm da rede de interconexões do dispositivo analógico programável. O circuito que determina quando a seqüência de dois vetores deve ser gerada, conduzindo o sinal **Next_trans** para nível alto cada vez que todas as respostas dos ORAs envolvidos no teste atingirem o valor V_{ref}, é mostrado na Figura 7.15b.

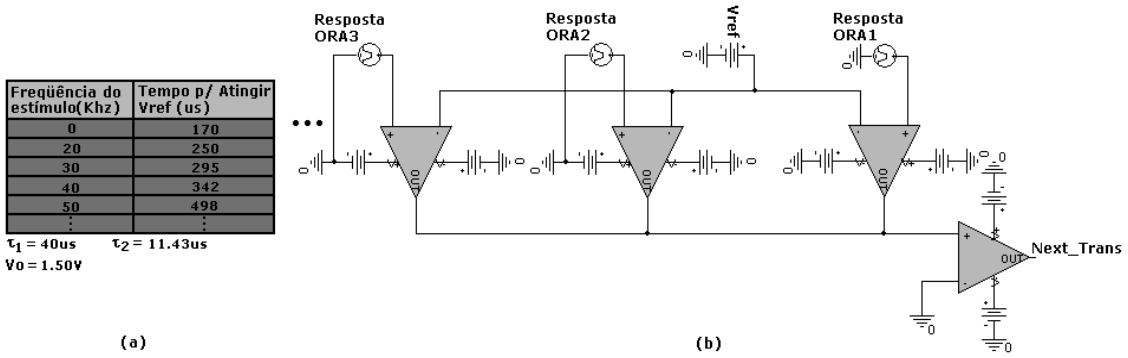


Figura 7.15: (a) Tabela com os tempos para atingir Vref quando estímulos com freqüências diferentes são aplicados (b) Circuito que habilita a geração de seqüências de dois vetores de teste.

Diferentemente do circuito gerador de vetores de teste e do controlador de tarefas, que são projetados para atuar externamente, os circuitos analisadores de resposta de teste e o circuito que habilita a geração de seqüências de dois vetores que estimulam cada uma das falhas citadas estão embarcados no dispositivo sob teste.

7.6 Validação da Arquitetura BIST Proposta

7.6.1 Descrição VHDL

Para a modelagem da arquitetura do circuito responsável pela geração de vetores e do circuito que controla as tarefas de geração e análise de teste utilizou-se a linguagem VHDL (*Very-High Speed Integrated Circuit Hardware Description Language*). O projeto foi organizado de forma estrutural, com módulos parametrizáveis, sempre que possível.

Abaixo são delineadas as funções dos arquivos utilizados na implementação de cada um dos módulos e, a seguir, a Figura 7.16 mostra a hierarquia destes arquivos dentro dos seus respectivos módulos:

- gerador_sys: Bloco lógico ou célula básica; é o arquivo topo do projeto.

- gerador_fsm: representa a interface com o controlador.

- gerador_contador: cuja área é proporcional a $\log_2 N$, onde N representa o número de linhas a serem testadas.

- gerador_decodificador: cuja área é proporcional a $\log_2 N$.

- gerador_mux: permite a seleção das linhas a serem testadas.

- global_control: bloco responsável pelo sincronismo de tarefas; é composto por uma máquina de estados finita.

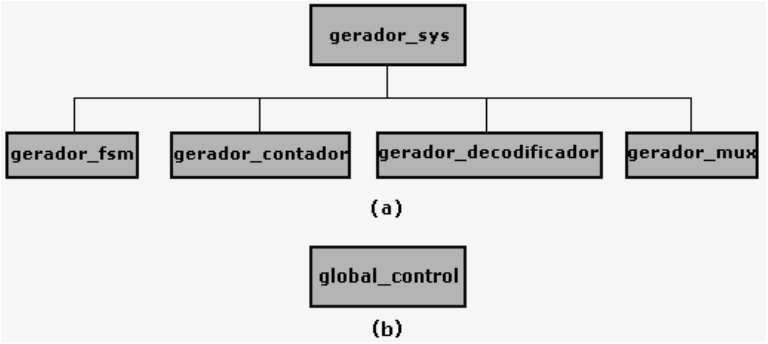


Figura 7.16: Hierarquias de Arquivos.

Como resultados de síntese destes componentes em FPGA, para testar um barramento com cinco linhas, foram obtidas as áreas citadas nas tabelas abaixo. A ferramenta de síntese utilizada foi a LeonardoSpectrum e o dispositivo comercial utilizado foi o FLEX10KE EPF10K10QC208 da ALTERA [ALT 2004].

Tabela 7.3: Resultados de síntese do gerador de estímulos de teste.

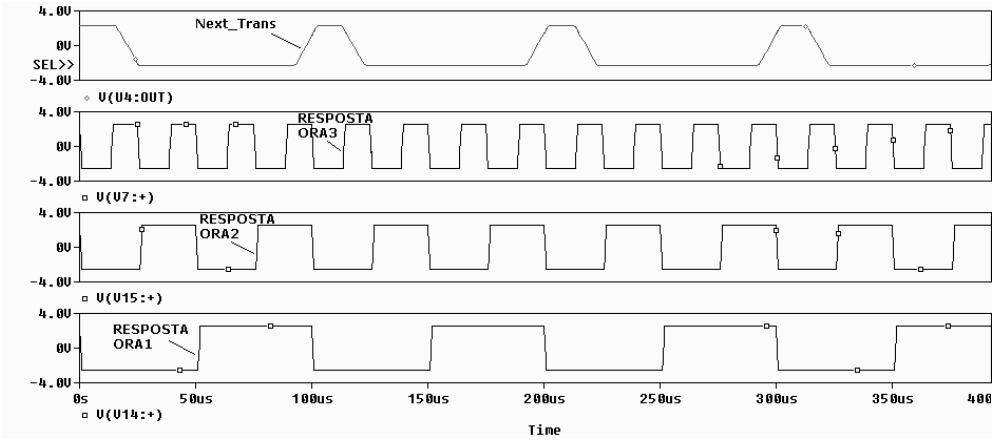
COMPONENTE	ÁREA (nº de células lógicas)	FREQÜÊNCIA MÁXIMA (MHz)
gerador_fsm	110	58,5
gerador_contador	4	206,8
gerador_decodificador	5	470.2
gerador_mux	1	436,7
gerador_sys	122	63.2

Tabela 7.4: Resultados de síntese do controlador global.

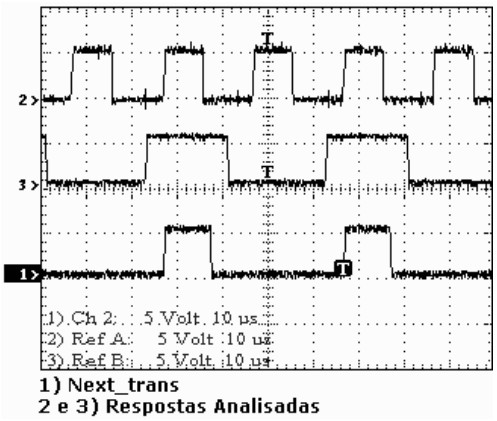
COMPONENTE	ÁREA (nº de células lógicas)	FREQÜÊNCIA MÁXIMA (MHz)
global_control	106	88,2

7.6.2 Resultados Experimentais

A princípio, o comportamento do circuito mostrado na Figura 7.15b que habilita a geração de seqüências de dois vetores de teste que estimulam cada uma das falhas é explicado através de simulações e resultados práticos. Este circuito foi projetado para que um sinal seja gerado cada vez que todas as respostas dos ORAs aos estímulos das linhas agressoras e da vítima atinjam o valor de tensão de referência V_{ref}. Com isto, garante-se o sincronismo entre as operações de geração e análise de teste, conforme mostra a Figura 7.17.



(a)



(b)

Figura 7.17: Simulação e implementação prática do circuito que habilita a geração de vetores de teste.

Devido ao tempo disponível para término deste trabalho, não foi possível validar a implementação do modelo desenvolvido que representa a rede de interconexões do AN10E40 a fim de que os módulos propostos pudessem ser aferidos sobre ele. Pode-se citar ainda como empecilho para a sua validação, as regras de projeto deste dispositivo que, por serem propriedades intelectuais da *Anadigm Company*, são desconhecidas do grande público. No entanto, com a simulação de falhas desenvolvida, pode-se garantir que os circuitos projetados possuem funcionalidade correta.

Estas simulações de falhas consistem em alterar a própria estrutura de geração de vetores a fim de verificar as modificações na funcionalidade de todo sistema. Conforme foi mostrado na Seção 7.5.3, uma seqüência de seis vetores de teste é necessária para que todas as falhas decorrentes de defeitos de *crosstalk* sejam estimuladas e detectadas. Portanto, alguns destes vetores foram modificados e desenvolveram-se simulações da arquitetura com o intuito de detectar as falhas sensíveis a eles.

A Figura 7.18 mostra a metodologia adotada para a simulação de falhas no circuito gerador de estímulos. Neste caso, falhas do tipo g_p e g_n foram simuladas alterando-se o segundo e o quarto vetor de teste do gerador fonte. Logo, cada vez que o analisador de respostas comparar os vetores gerados por este gerador com os gerados pelo gerador local ou com os resultados armazenados na tabela que contém os tempos em que cada

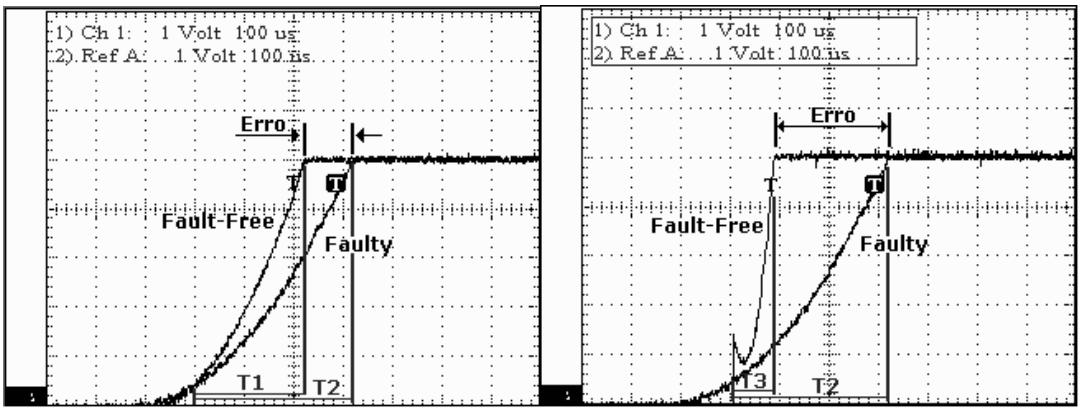
estímulo atinge o valor V_{ref} após a dupla integração (Figura 7.15a), erros devem ser detectados na sua saída decorrentes destas falhas.

		Posições dos Bits no Barramento		
		N-1..... .....i+1	i	i-1..... 0
g_p	0	0 0	0	0 0
	1	1 1	1→(0)	1 1
g_n	1	1 1	1	1 1
	0	0 0	0→(1)	0 0
d_d d_s	1	1 1	0	1 1
	0	0 0	1	0 0

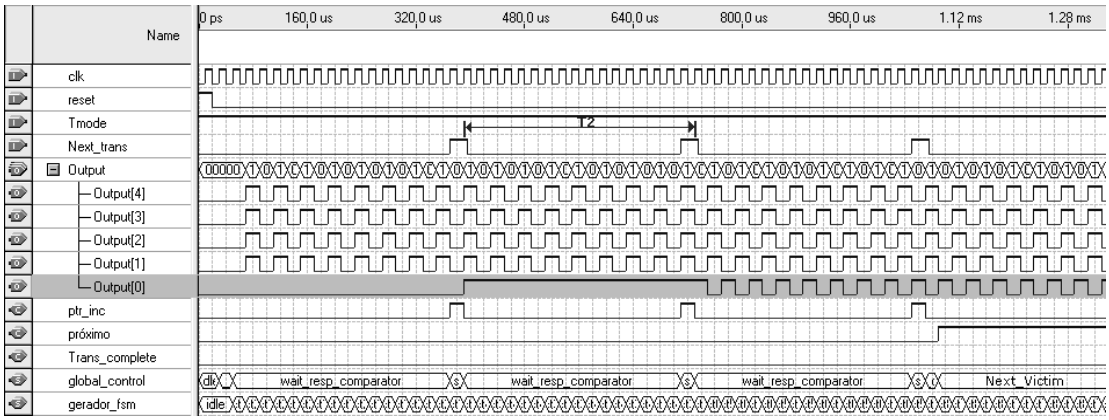
Vítima = "fio" i

Figura 7.18: Simulação de falhas do tipo g_p e g_n através da alteração da seqüência de vetores do gerador fonte.

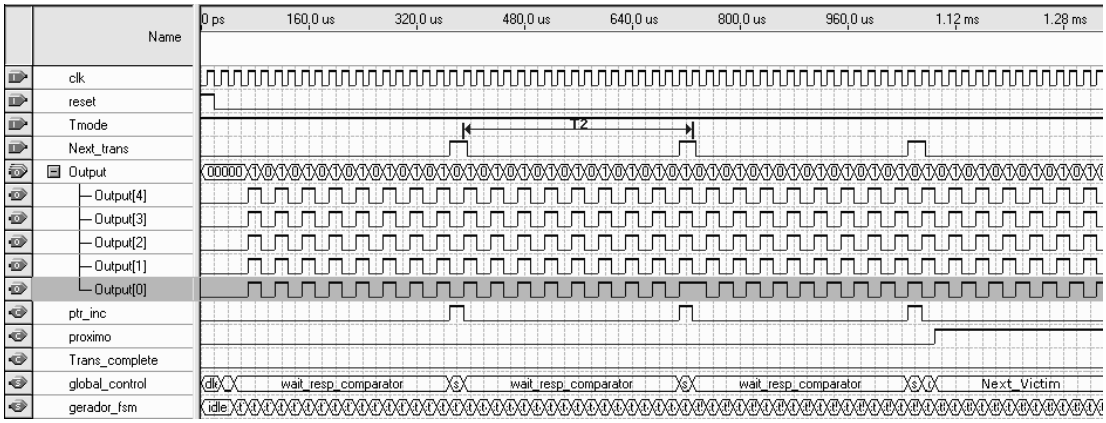
Com isto, pode-se simular o comportamento de uma rede de interconexões com defeitos *crosstalk* que resultam nestes tipos de falhas e, muitas vezes, em erros intermitentes. Nos experimentos, o ORA foi programado para que o primeiro integrador tivesse constante de tempo igual 40µs (τ_1) e, o segundo, 11,43µs (τ_2). Além disto, para este caso, foram utilizados como estímulos de teste ondas quadradas com amplitude de $1{,}5V_p$ e freqüência de 50KHz que, conforme a Figura 6.3e, estão dentro dos limites de espaço de projeto do ORA. Portanto, as respostas esperadas e diante destas falhas da saída do analisador e as simulações do comportamento de toda a arquitetura projetada são mostradas na Figura 7.19.



(a) (b)



(c)



(d)

Figura 7.19: (a) e (b) Respostas sem e com falhas g_p e g_n, respectivamente (c) e (d) Simulação da arquitetura projetada sem e com falhas, respectivamente.

Como podem ser observados nas figuras acima, erros são detectados na linha vítima quando falhas g_p e g_n são injetadas na rede. A Figura 7.19a mostra o atraso da resposta do ORA resultante de *glitches* positivos, ou seja, de transições de "0" para "1", comparada à resposta ideal, quando o regime permanente em "0" prevalece. Por outro lado, a Figura 7.19b mostra que a resposta com falha, referente a *glitches* negativos (transições de "1" para "0"), atinge o valor V_{ref} com atraso em relação à resposta esperada, isto é, quando o regime permanente em "1" é sustentado na linha da vítima. Considerando que o tempo de *reset* dos analisadores, isto é, o tempo para que as saídas destes circuitos sejam 0V, é equivalente a dois ciclos de *clock*, as simulações ratificam as alterações no comportamento do sistema.

Por fim, falhas de *delay* foram simuladas utilizando as chaves programáveis da rede global de interconexões do dispositivo sob teste. Observou-se que, ao inserir um determinado número de chaves no caminho de teste, a resposta do ORA variava devido às capacitâncias (capacitância de acoplamento, capacitância de *gate*, etc) introduzidas por estes elementos. A Tabela 7.5 mostra os atrasos durante as transições que são resultantes das capacitâncias de algumas destas chaves, utilizando-se como estímulo um sinal correspondente a uma onda quadrada, com amplitude igual a $1{,}25V_p$ e com freqüência igual a 40KHz. A seguir, são mostradas, na Figura 7.20, as respostas do ORA obtidas durante a introdução destas chaves no caminho de teste.

Tabela 7.5: Tempo de transição para diferente número de chaves no caminho de teste.

NÚMERO DE CHAVES	1	5	11	15	19	21
TEMPO DE SUBIDA (ns)	77	100	292	484	756	788
TEMPO DE DESCIDA (ns)	75	97	308	462	790	880

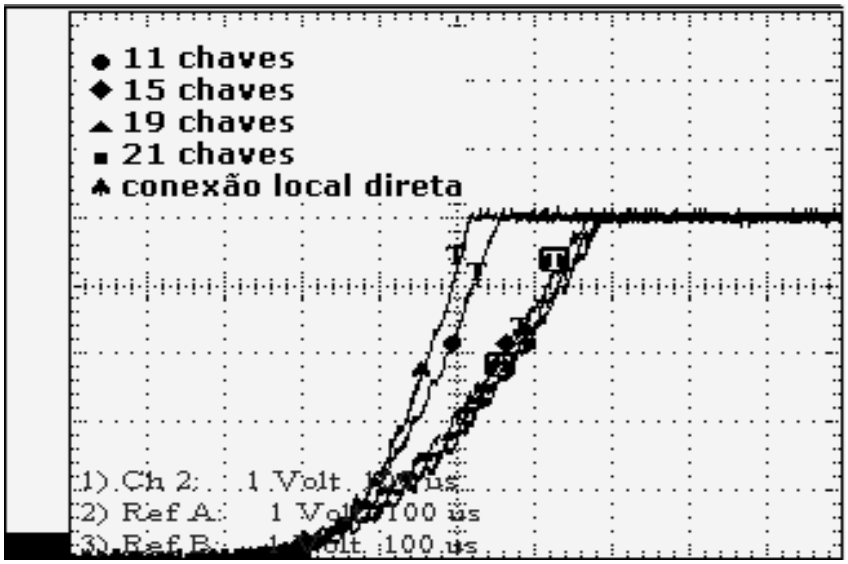


Figura 7.20: Respostas do ORA quando um número diferente de chaves é introduzido no caminho de teste.

Aplicando-se os vetores citados na Figura 7.12 que estimulam falhas d_s e d_d, utilizando-se quatro linhas da rede global no teste, sendo que destas, três atuam como agressoras, e utilizando-se o ORA proposto, é possível observar o efeito das capacitâncias localizadas entre linhas que resultam em atrasos nas transições dos estímulos. As figuras 7.21 e 7.22 mostram, respectivamente, o esquema utilizado para gerar os vetores de teste com amplitudes de $2{,}5V_p$, freqüências iguais à 80KHz e as respostas obtidas para estes estímulos.

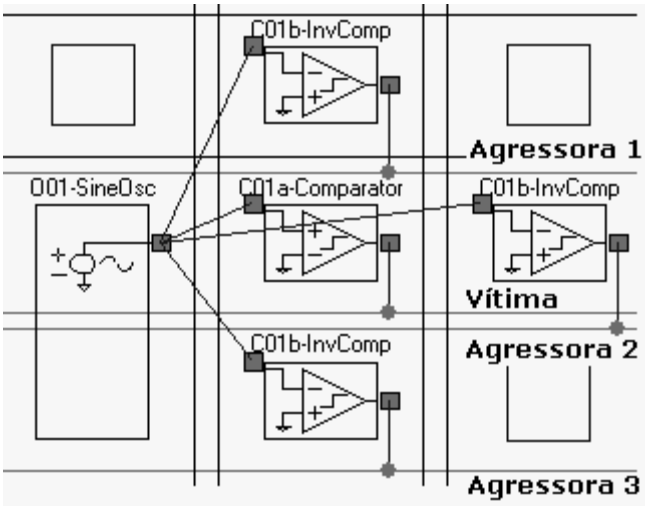


Figura 7.21: Esquema utilizado para geração de vetores de teste.

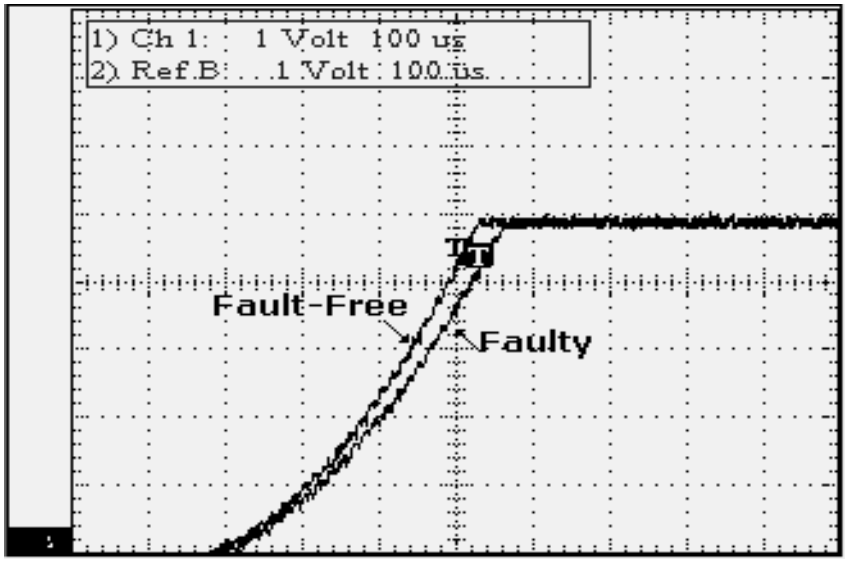


Figura 7.22: Respostas do ORA aos vetores que estimulam falhas d_s e d_d.

7.7 Conclusão

Considerando que os circuitos da estrutura BIST propostos apresentam bons resultados na tarefa de detectar falhas de *glitches* e *delay* decorrentes de defeitos *crosstalk*, e que a repetibilidade dos circuitos dedicados à geração de vetores (circuito externo) e à análise de teste (circuito embarcado) é alta**,** pode-se concluir que esta proposta de arquitetura vai ao encontro do objetivo delineado na Seção 6.1.3. Além disto, os custos em área e o desempenho em freqüência de operação dos circuitos de controle e geração de teste, sintetizados em um circuito específico com lógica digital programável, não são proibitivos.

8 CONCLUSÃO

O uso de circuitos analógicos que possam ser facilmente implementados e programados se tornou imprescindível para a sustentação e evolução da eletrônica de consumo. A contínua busca pela modernização tem levado ao desenvolvimento de circuitos que se tornam cada dia menores e mais complexos. A competitividade entre os fabricantes de semicondutores, a necessidade tecnológica de alta velocidade, a diversidade de aplicações e o anseio da sociedade moderna por novos produtos têm obrigado os fabricantes a buscarem formas alternativas de diminuir o tempo entre o projeto e a disponibilidade dos seus produtos para os consumidores.

Grandes progressos foram feitos nas áreas referentes ao *hardware* e à programação de FPGAs. Os dispositivos analógicos programáveis, apesar de ainda não terem a mesma popularidade de seus pares digitais FPGAs, possuem uma gama de aplicações bastante ampla, que vai desde o condicionamento de sinais em sistemas de instrumentação, até o processamento de sinais de RF em telecomunicações. A fabricação e utilização dos FPAAs, em muito difere da dos FPGAs em pontos tais como quantidade de circuitos integráveis, programabilidade, granularidade, freqüência de operação e quantidade de recursos de interconexão.

A existência de FPAAs permite ao projetista desenvolver hoje sistemas analógicos complexos preocupando-se fundamentalmente com a funcionalidade de seus blocos básicos programáveis e com as questões de comunicação entre CABs e destes com o mundo externo. Passa a não mais ser necessário considerar detalhes de implementação de cada bloco que compõe o sistema, diminuindo-se, assim, o tempo de projeto.

Porém, junto com os impressionantes ganhos obtidos na agilidade de concepção de circuitos analógicos, os FPAAs trouxeram um conjunto de novos problemas relativos ao teste deste tipo de dispositivo. Procurando-se sanar estes problemas, nesta dissertação foram apresentadas propostas de estratégias para teste e detecção de falhas nas redes local e global de interconexões, bem como, nos pinos de I/O destes dispositivos. Uma vez que as características estruturais e funcionais de cada elemento a ser testado são distintas, eles foram analisados separadamente, considerando-se modelos de falhas, configurações e estímulos de teste específicos para assegurar uma boa taxa de detecção de defeitos. Além disto, vale lembrar que as estratégias de teste desenvolvidas visam a detecção de defeitos pós-fabricação na rede de interconexões e I/Os, sem configuração prévia do dispositivo para uma determinada aplicação.

Inicialmente apresentaram-se modelos que representam as redes globais e locais de interconexões, sendo que este último envolve também os I/Os do dispositivo sob teste. Optou-se por esta estratégia porque, ao verificar os avanços relativos ao teste de FPGAs, foram observados os excelentes ganhos referentes ao tempo e às coberturas de teste de toda a arquitetura destes dispositivos programáveis utilizando-se grafos de

adjacência para o modelamento. Além disto, estes modelos possibilitam que algoritmos de pesquisa em grafos já conhecidos sejam adaptados ao caso de FPAAs.

Posteriormente, com os modelos de interconexões e de falhas já definidos, deu-se início à elaboração dos algoritmos de pesquisa que geram as configurações de teste de cada uma das redes a serem testadas. Elaboraram-se heurísticas que visam soluções ótimas tanto no teste da rede global, quanto no teste da rede local de interconexões e que garantem a independência dos estímulos que são aplicados externamente. Entretanto, duas estratégias distintas foram adotadas no teste da rede global: uma visando detectar falhas *stuck-on* e outra falhas *stuck-open*. Quando se almeja detectar a primeira delas, utiliza-se um algoritmo que seleciona caminhos críticos nos grafos pesquisados e que correspondem aos caminhos elétricos bufferizados ou não bufferizados com o maior número de chaves desta rede. Já quando se deseja detectar falhas *stuck-on*, caminhos elétricos disjuntos são selecionados, sendo que, em uma estratégia aplica-se apenas um estímulo, enquanto que na outra, aplicam-se três estímulos com freqüências fundamentais distintas.

O teste da rede local englobou em uma única estratégia tanto o teste de falhas *stuck-on* quanto o teste de falhas *stuck-open*, utilizando, para isto, apenas um algoritmo guloso. Este algoritmo também seleciona caminhos críticos em grafos que correspondem aos caminhos que abrangem o maior número de chaves a serem testadas no dispositivo. Nesta estratégia são incluídas não só as ligações que os CABs podem fazer entre si e com os barramentos, mas também as ligações que as células de I/O realizam com os seus vizinhos. No entanto, faltava ainda o teste das chaves e *buffers* destas células para que o teste fosse concluído. Foi então que se optou pela adaptação de uma estratégia que determina que algumas células sejam "cascateadas" a fim de originarem osciladores com deslocamento de fase, os PSOs. Assim, qualquer alteração na freqüência e/ou na amplitude do estímulo gerado por estes PSOs corresponde a uma falha em uma chave ou *buffer* das células envolvidas.

Ainda nesta dissertação abordou-se o auto-teste da rede de interconexões e I/Os onde foram propostos circuitos responsáveis pela geração e análise de teste que são implementados utilizando os recursos internos do AN10E40 e que, conseqüentemente, não elevam os custos referentes à área de projeto. O circuito responsável pela análise é formado por dois integradores e por alguns circuitos adicionais; enquanto que, neste caso, para geração de estímulos foram escolhidos dois osciladores distintos: um que utiliza um amplificador inversor, três filtros passa-baixas e *buffers* (PSO bufferizado) e um outro disponível na biblioteca de *hardware* do AN10E40 que utiliza a própria estrutura do ORA na sua topologia, além de circuitos adicionais. Avaliações importantes no teste analógico, como por exemplo, repetibilidade, linearidade e sensibilidade dos circuitos que constituem o esquema BIST proposto também foram realizadas.

Por fim, o auto-teste visando à detecção de falhas provindas de defeitos *crosstalk* foi tratado. Neste contexto, também foram elaborados circuitos para controle e geração de teste, que atuam externamente ao dispositivo, e um circuito responsável pela a análise de teste, correspondente ao duplo integrador proposto no auto-teste da rede de interconexões. A arquitetura dos dois primeiros foi descrita em linguagem de descrição de *hardware* e a validação da proposta foi feita através de simulações e resultados práticos.

É possível, verificar, portanto, que as propostas de teste apresentadas nesta dissertação vão ao encontro das metas declaradas previamente, uma vez que o menor número de CTs foi gerado e, conseqüentemente, garantiu-se menor tempo de teste. Com a elaboração de diferentes versões dos algoritmos para teste externo, obteve-se o número mínimo de 7 CTs e 14 CTs para detectar falhas *stuck-open* e 2 CTs e 3 CTs para detectar falhas *stuck-on* nas chaves da rede global de interconexões. Quando a rede local foi testada, obteve-se o número mínimo de 13 CTs para cobrir as falhas *stuck-open* e 71 CTs para detectar falhas *stuck-on* em 100% das suas chaves. Além disto, utilizando-se a proposta OBIST, obteve-se apenas 1 CT para cobrir falhas *stuck-open* nas chaves e *buffers* das células de I/O e 4 CTs para cobrir falhas *stuck-on*.

Já quando os elementos de teste são integrados ao dispositivo, obteve-se o número mínimo de 14 CTs para detectar todas as falhas *stuck-open* e apenas 3 CTs e 5 CTs para cobrir as falhas *stuck-on* nas chaves da rede global deste circuito. Para cobrir falhas *stuck-open* na rede local obteve-se o número mínimo de 15 CTs e para cobrir 100% das falhas *stuck-on* foram necessárias 176 CTs utilizando-se 2 PSOs por CT. No teste integrado de I/Os foram exigidas 2 CTs para detectar falhas *stuck-open* e 5 CTs para cobrir falhas *stuck-on* nas suas chaves.

Além disto, após o estudo detalhado dos circuitos propostos para geração e análise de teste integrado, concluiu-se que eles satisfazem as exigências previamente delineadas de apresentarem alta repetibilidade, linearidade e sensibilidade.

Como contribuições adicionais, este trabalho poderá servir de referência para pesquisa e desenvolvimento de novas estratégias de teste. Além disto, novos circuitos geradores e analisadores de teste poderão ser implementados utilizando como base os circuitos propostos neste texto.

REFÊRENCIAS

[ABR 99] ABRAMOVICI, M. et al. Using Roving STAR's for On-Line Testing and Diagnosis of FPGA's in Fault-Tolerant Applications. In: INTERNATIONAL TEST CONFERENCE, ITC, 1999. **Proceedings…**[S.l.: s.n.], 1999.

[ABR 2002] ABRAMOVICI, M.; STROUD, C; EMMERT, M. Using Embedded FPGAs for SOCs Yield Improvement. In: DESIGN AUTOMATION CONFERENCE, DAC, 39., 2002, New Orleans, EUA. **Proceedings…** New Orleans: [s.n.], 2002.

[ABR 85] ABRAMOVICI, M.; MENON, P. R. A Practical Approach to Fault Simulation and Test Generation for Bridging Faults. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,** New York, v. C-34, p. 658–663, July 1985.

[ALD 2003] ALDERIGHI, M. et al. A Fault Injection Tool for SRAM-Based FPGAs. In: IEEE INTERNATIONAL ON-LINE TESTING SYMPOSIUM, IOLTS, 9., 2003. **Proceedings…**[S.l.: s.n.], 2003.

[ALT 2004] ALTERA Web Site [Online]. Disponível em: <http://www.altera.com/>. Acesso em: jul. 2004.

[AMO 2003] AMORY, A. **Integração de Avaliação de Técnicas de Teste Baseado em Software no Fluxo de Projeto de SOCs**. 2003. Dissertação (Mestrado em Ciência da Computação) - PUC-RS, Porto Alegre.

[ANA 2004] ANADIGM. **AN10E40 User Manual**. [S.l.], 2004.

[AND 2004a] ANDRADE JUNIOR, A. et al. Test Planning for Mixed-Signal SoC and Analog BIST: a case study. In: DESIGN OF CIRCUITS AND INTEGRATED SYSTEMS, DCIS, 19., 2004, Bordeaux, França. **Proceedings…** Bordeaux: [s.n.], 2004. p. 567-572.

[AND 2004b] ANDRADE JUNIOR, A.; PEREIRA, G. V.; BALEN, T. R.; LUBASZEWSKI, M.; FLORENCE, A.; RENOVELL, M. Testing Global Interconnects of Field Programmable Analog Arrays. In: IEEE INTERNATIONAL MIXED-SIGNAL TEST WORKSHOP, 10., 2004, Portland, USA. **[Papers]**. [S.l: s.n.], 2004.

[AND 2005] ANDRADE JUNIOR, A.; PEREIRA, G. V.; BALEN, T. R.; LUBASZEWSKI, M.; FLORENCE, A.; RENOVELL, M. Built-in Self Test of Global Interconnects of Field Programmable Analog

Arrays. In: MICROELECTRONICS JOURNAL: SPECIAL ISSUE ON IMSTW 2004, 2005. **Proceedings…**[S.l.: s.n.], 2004.

[ARA 96a] ARABI, K.; KAMINSKA, B. Oscillation-Test Strategy for Analog and Mixed-Signal Integrated Circuits. In: VLSI TEST SYMPOSIUM, 14., 1996. **Proceedings…**[S.l.: s.n.], 1996. p. 476-482.

[ARA 96b] ARABI, K.; KAMINSKA, B.; SUNTER, S. Design for Testability of Integrated Operational Amplifiers Using Oscillation-Test Strategy. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1996, California, USA. **Proceedings…** California: [s.n.], 1996. p. 40-45.

[BAA 93] BAASE, S. **Computer Algorithms**: Introduction to Design & Analysis. 2nd ed. Reading: Addison Wesley, 1993.

[BAI 2000] BAI, X.; DEY, S.; RAJSKI, J. Self-Test Methodology for At-Speed Test of Crosstalk in Chip Interconnects. In: DESIGN AUTOMATION CONFERENCE, DAC, 2000. **Proceedings…**[S.l.: s.n.], 2000. p. 619-624.

[BAI 2001] BAI, X.; DEY, S. High-Level Crosstalk Detect Simulation for System-on-Chip Interconnects. In: IEEE VLSI TEST SYMPOSIUM, VTS, 19., 2001. **Proceedings…**[S.l.: s.n.], 2001. p. 169-175.

[BAI 2003] BAI, X.; CHEN, L.; DEY, S. Software-Based Self-Test for Crosstalk in Processors. In: INTERNATIONAL HIGH LEVEL DESIGN VALIDATION AND TEST, HLDVT, 2003. **Proceedings…**[S.l.: s.n.], 2003. p.11-16.

[BAL 2002] BALEN, T. R.; SCHEREIBER, M.; LUBASZEWSKI, M. OBIST Applied to FPAAs: A Case Study. In: LATIN AMERICAN TEST WORKSHOP, LATW, 4., 2002, Natal. **Proceedings…** Natal: [s.n.], 2002.

[BAL 2004a] BALEN, T. R. et al. An Approach to the BIST of FPAA. In: IEEE VLSI TEST SYMPOSIUM, 22., 2004**,** Napa Valley, California. **Proceedings…** California: [s.n.], 2004. p. 383.

[BAL 2004b] BALEN, T. R. et al. Testing the Configurable Analog Blocks of Field Programmable Analog Array. In: INTERNATIONAL TEST CONFERENCE, ITC, 2004. **Proceedings…**[S.l.: s.n.], 2004.

[BUX 2000] BUXTON, A. Totally Reconfigurable Analog Circuit, Concept and Practical Implementation. In: CIRCUITS AND SYSTEMS AND MIDWEST SYMPOSIUM, 42., 2000. **Proceedings…**[S.l.: s.n.], 2000. p. 292 –295.

[CHE 2001] CHEN, L.; BAI, X.; DEY, S. Testing for Interconnect Crosstalk Defects Using On-Chip Embedded Processor Cores. In: DESIGN AUTOMATION CONFERENCE, DAC, 38., 2001, Las Vegas. **Proceedings…** Las Vegas: [s.n.], 2001.

[CHE 2003] CHENG, J.; LINDA, M. A BIST Solution for the Test of I/O Speed. In: INTERNATIONAL TEST CONFERENCE, ITC, 2003. **Proceedings…**[S.l.: s.n.], 2003. p. 1023-1030.

[CHM 2003] CHMELAF, E. FPGA Interconnect Delay Fault Testing. In: INTERNATIONAL TEST CONFERENCE, ITC, 2003. **Proceedings…**[S.l.: s.n.], 2003. p. 1239-1247.

[COR 2002] CORMEN, T. H.; LEISERSON, C. E.; STEIN, C. **Algoritmos:** Teoria e Prática. Rio de Janeiro: Campus, 2002.

[COT 2000] COTA, E.; NEGREIROS, M.; CARRO, L.; LUBASZEWSKI, M. A New Adaptive Analog Test and Diagnosis System. **IEEE Transactions on Instrumentation and Measurement**, New York, v. 49, n. 2, p. 223-227, 2000.

[FER 2003] FERNANDES, D. A.; HARRIS, I. G. Application of Built in Self-Test For Interconnect Testing of FPGAs**.** In: INTERNATIONAL TEST CONFERENCE, ITC, 2003. **Proceedings…**[S.l.: s.n.], 2003. p. 1248-1257.

[GAU 97] GAUDET, V. C.; GULAK, G. CMOS Implementation of a Current Conveyor-Based Field-Programmable Analog Array. In: SIGNALS, SYSTEMS & COMPUTER AND CONFERENCE RECORD OF THE THIRTY-FIRST ASILOMAR CONFERENCE, 1997. **Proceedings…**[S.l.: s.n.], 1997. v. 2, p. 1156-1159.

[GIB 97] GIBSON, G.; GRAY, L.; STROUD, C. E. Boundary Scan Access of Built-In Self-Test for Field Programmable Gate Arrays. In: IEEE INTERNATIONAL ASIC, 1997. **Proceedings…**[S.l.: s.n.], 1997. p. 57–61.

[GUL 95] GULAK, G. Field-Programmable Analog Arrays: Past, Present and Future Perspectives. In: MICROELECTRONICS AND VLSI AND IEEE REGION 10 INTERNATIONAL CONFERENCE ON MICROELECTRONICS AND VLSI, 1995. **Proceedings…**[S.l.: s.n.], 1995.

[HAR 2000a] HARRIS, I. G.; TESSIER, R. Interconnect Testing in Cluster-Based FPGA Architectures. DESIGN AUTOMATION CONFERENCE, 37**.**, 2000. **Proceedings…**[S.l.: s.n.], 2000. p. 49-54.

[HAR 2000b] HARRIS, I. G.; TESSIER, R. Diagnosis of Interconnect Faults in Cluster-Based FPGA Architectures. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2000. **Proceedings…**[S.l.: s.n.], Nov. 2000.

[HAR 2002] HARRIS, I. G.; TESSIER, R. Testing and Diagnosis of Interconnect Faults in Cluster-Based FPGA Architectures**. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 21, n. 11, p.1337-1343, Nov. 2002.

[IEE 90] IEEE. **IEEE STANDARD 1149.1**: IEEE Standard Test Access Port and Boundary Scan Architecture. New York, 1990.

[KRA 2001] KRASNIEWSKI, A. Testing FPGA Delay Faults in the System Environment is Very Different from "Ordinary" Delay Fault Testing. In: IEEE INTERNATIONAL ON-LINE TEST WORKSHOP, 2001. **Proceedings…**[S.l.: s.n.], 2001. p. 37–40.

[LAL 2003] LALA, K.; BURRESS, L. Self-Checking Logic Design for FPGA Implementation. **IEEE Transactions on Instrumentation and Measurement**, New York, v. 52, n. 5, Oct. 2003.

[LAK 2000] LAKAMRAJU, V.; TESSIER, R. Tolerating Operational Faults in Cluster Based FPGAs. In: INTERNATIONAL ACM/SIGDA SYMPOSIUM FIELD PROGRAMMABLE GATE ARRAYS, 8., 2000, Monterey, CA, USA. **Proceedings…** Monterey: ACM, 2000.

[LEE 95a] LEE, E.; GULAK, G. Transconductor-Based Field-Programmable Analog Array. In: ISSCC DIGEST OF TECHNICAL PAPERS, 1995. **Proceedings…**[S.l.: s.n.], 1995. p.198-199.

[LEE 95b] LEE, E. **Field-Programmable Analog Arrays Based on MOS Transconductors**. 1995. Tese (Doutorado em Engenharia Elétrica) - Department of Electrical and Computer Engineering, University of Toronto, Canadian.

[LOO 2000] LOOBY, C. A.; LYDEN, C. Op-Amp Based CMOS Field-Programmable Analogue Array. **IEEE Pro-Circuits Systems**, Livingston , v. 147, n.2, 2000.

[LUB 96] LUBASZEWSKI, M.; SALVADOR, M. L. P. ABILBO: Analog Built-in Block Observer. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1996, San Jose, California. **Proceedings…** San Jose: ACM, 1996. p.600-603.

[MAN 2000] MANCINI, Ron. Design of Op Amp Sine Wave Oscillators. Texas Instruments Incorporated. **Analog Applications Journal**, [S.l.], Aug 2000.

[MET 2001] METRA, C.; PAGANO, A.; RICCO, B. On-Line Testing of Transient and Crosstalk Faults Affecting Interconnections of FPGA-Implemented Systems. In: INTERNATIONAL TEST CONFERENCE, ITC, 2001. **Proceedings…**[S.l.: s.n.], 2001. p. 939–947.

[MIC 96] MICHINISSHI, H. et al. A Test Methodology for Interconnect Structures of LUT-Based FPGAs. In: ASIAN TEST SYMPOSIUM, 5., ATS, Taiwan, 1996. **Proceedings…** Taiwan: [s.n.], 1996. p. 68-74.

[PAL 98] PALUSINSKI, O. A. et al. Motorola Field Programmable Analog Array. In: **Simulation, Control, and Circuit Design Laboratories**. USA: University of Arizona, 1998.

[PER 2005] PEREIRA, G. V. et al. Testing the Interconnect Networks and I/O Resources of Field Programmable Analog Arrays. In: IEEE VLSI TEST SYMPOSIUM, 23., 2005, Palm Springs, CA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2005. p. 389-394.

[RAB 96] RABAEY, J. **Digital Integrated Circuits:** A Design Perspective. Upper Saddle River: Prentice-Hall, 1996.

[REN 96] RENOVELL, M.; FIGUEIRAS, J.; ZORIAN, Y. Testing the Interconnect Structures of Unconfigurated FPGAs. In: IEEE EUROPEAN TEST WORKSHOP, ETW, 7., 1996, Sète, Montpellier, França. **Proceedings…** Montpellier: [s.n.], 1996. p. 125-129.

[REN 97] RENOVELL, M.; FIGUEIRAS, J.; ZORIAN, Y. Test of RAM-Based FPGA: Methodology and Application to the Interconnection Structure. In: IEEE VLSI TEST SYMPOSIUM, VTS, 15., 1997. **Proceedings…**[S.l.: s.n.], 1997. p. 230-237.

[REN 98a] RENOVELL, M.; FIGUEIRAS, J.; ZORIAN, Y. Testing the Interconnect of RAM-Based FPGAs. **IEEE Design & Test of Computers**, Los Alamitos, v.15, n. 1, p. 45-50, 1998.

[REN 98b] RENOVELL, M. SRAM-Based FPGAs: A Structural Test Approach. In: BRAZILIAN SYMPOSIUM ON INTEGRATED CIRCUIT DESIGN, SBCCI, 11., 1998, Rio de Janeiro. **Proceedings…** Los Alamitos: IEEE Computer Society, 1998. p. 67-72.

[REN 98c] RENOVELL, M. et al. SRAM-Based FPGAs: Testing the LUT/RAM Modules. INTERNATIONAL TEST CONFERENCE, ITC, 1998. **Proceedings…**[S.l.: s.n.], 1998. p. 1102–1111.

[REN 99] RENOVELL, M.; PORTAL, J. .M.; FIGUEIRAS, J.; ZORIAN, Y. Testing the Configurable Interconnect/Logic Interface of SRAM-Based FPGA's. In: DESIGN, AUTOMATION AND TESTING EUROPE, DATE, 1999, Munich. **Proceedings…** Munich: [s.n.], 1999. p. 619-622.

[REN 2000a] RENOVELL, M.; ZORIAN, Y. Different Experiments in Test Generation for XILINX FPGAs. In: INTERNATIONAL TEST CONFERENCE, ITC, 2000. **Proceedings…**[S.l.: s.n.], 2000. p. 854 – 862.

[REN 2000b] RENOVELL, M.; PORTAL, J. M. Testing the Local Interconnect of SRAM-Based FPGA's. **Journal of Electronic Testing: Theory and Applications**, Hingham, p.513-520, 2000.

[REN 2000c] RENOVELL, M.; PORTAL, J. M.; FAURE, P.; FIGUEIRAS, J.; ZORIAN, Y. TOF: A Tool for Test Pattern Generation Optimization of FPGA Application-Oriented Test. In: ASIAN TEST SYMPOSIUM, ATS, 9., 2000. **Proceedings…**[S.l.: s.n.], 2000. p. 323-328.

[REN 2002] RENOVELL, M.; FAURE, P.; PRINETTO, P.; ZORIAN, Y. Testing the Unidimensional Interconnect Architecture of Symmetrical SRAM-Based FPGA. In: IEEE INTERNATIONAL WORKSHOP ON ELECTRONIC DESIGN, TEST AND APPLICATIONS, DELTA, 1., 2002. **Proceedings…**[S.l.: s.n.], 2002. p. 297-301.

[SED 2004] SEDRA, A. S.; SMITH, K. C. **Microelectronic Circuits**. 3rd ed. [S.l.]: Harcourt Brace College Publishers, 2004.

[SHN 98] SHNIDMAN, N. R.; MANGIONE-SMITH, W. H.; POTKONJAK, M. On-line Fault Detection for Bus-Based Field Programmable Gate Arrays. **Transactions on VLSI Systems**, Piscataway, v. 6, p. 656–666, Dec. 1998.

[SIV 88] SIVILOTTI, M. A Dynamically Configurable Architecture for Prototyping Analog Circuits. In: MIT VLSI CONFERENCE, 1988, Massachusetts. **Proceedings…** [S.l.]: Laboratory for Computer Science, MIT, 1988. p. 237-258.

[SOU 2002] SOUHAIMI, Y. **Test des FPAA:** Field Programmable Analog Array. 2002. Tese (Doutorado em Engenharia Elétrica) - Systèmes Automatiques et Microélectroniques, Académie de Montpellier, Université Montpeller II: Sciences et Techniques du Languedoc, Montpeller.

[STR 97] STROUD, C. E.; LEE, E.; ABRAMOVICI, M. BIST-Based Diagnostics of FPGA Logic Blocks. In: INTERNATIONAL TEST CONFERENCE, ITC, 1997. **Proceedings…**[S.l.: s.n.], 1997. p. 539–547.

[STR 98] STROUD, C. et al. Built-in Self-Test of FPGA Interconnect. In: INTERNATIONAL TEST CONFERENCE, ITC, 1998. **Proceedings…**[S.l.: s.n.], 1998. p. 404–441.

[STR 2001] STROUD, C. et al. On line BIST and Diagnosis of FPGA Interconnect Using Roving Stars. In: INTERNATIONAL TEST CONFERENCE, ITC, 2001. **Proceedings…**[S.l.: s.n.], 2001.

[STR 2002] STROUD, C.; NALL, J.; LASHINSKY, M. BIST-Based Diagnosis of FPGA Interconnect. In: INTERNATIONAL TEST CONFERENCE, ITC, 2002. **Proceedings…**[S.l.: s.n.], 2002. p. 618-627.

[STR 2003] STROUD, C. E.; LEACH, K. N.; SLAUGHTER, T. A. Bist for Xilinx 4000 and Spartan Series FPGAs: a Case Study. In: INTERNATIONAL TEST CONFERENCE, ITC, 2003, Charlotte. **Proceedings…** Charlotte: [s.n.], 2003. p. 1258-1266.

[SUN 2000a] SUN, X.; XU, J.; CHAN, B.; TROUBORST, P. Novel Technique for Built-In Self-Test of FPGA Interconnects. In: INTERNATIONAL TEST CONFERENCE, ITC, 2002. **Proceedings…**[S.l.: s.n.], 2002. p. 795-803.

[SUN 2002b] SUN, X.; ALIMOHAMMAD, A.; TROUBORST, P. Modeling of FPGA Local/Global Interconnect Resources and Derivation of Minimal Test Configurations. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, DFT, 17., 2002. **Proceedings…**[S.l.: s.n.], 2002.

[SUN 2002c] SUN, X.; XU, J.; ALIMOHAMMAD, A. **An Algorithms Method for Deriving Minimum TCs for FPGAs Local Interconnects**. [S.l.]: Department of Electrical and Computer Engineering, Unversity of Aberta, 2002.

[SUN 2002d] SUN, X. et al. Minimal Test Configuration for FPGA Local Interconnects. In: IEEE CANADIAN CONFERENCE ON ELECTRICAL & COMPUTER ENGINEERING, 2002. **Proceedings...** [S.l.:s.n.], 2002. p. 427-432.

[SUN 2004] SUN, X.; TROUBORST, P. A Unified Global and Local Interconnect Test Scheme for Xilinx XC4000 FPGAs. **IEEE Transactions on Instrumentation and Measurement**, New York, v. 53, n. 2, April 2004.

[TAH 2003] TAHOORI, M. B.; MITRA, S. Automatic Configuration Generation for FPGA Interconnect Testing. In: VLSI TEST SYMPOSIUM, VTS, 21., 2003, Napa Valley. **Proceedings…** Napa Valley: [s.n.], 2003. p.134 – 139.

[TAH 2004] TAHOORI, M. B.; ABRAMOVICI, M. Application-Specific Bridging Fault Testing of FPGAs. **Journal of Electronic Testing: Theory and Applications**, Dordrecht, p. 279-289, 2004.

[WEY 1990] WEY, C. L. Built-In Self-Test Structure for Analog Circuit Fault Diagnosis. **IEEE Transactions on Instrumentation and Measurement**, New York, v. 39, n. 3, p. 517-521, 1990.

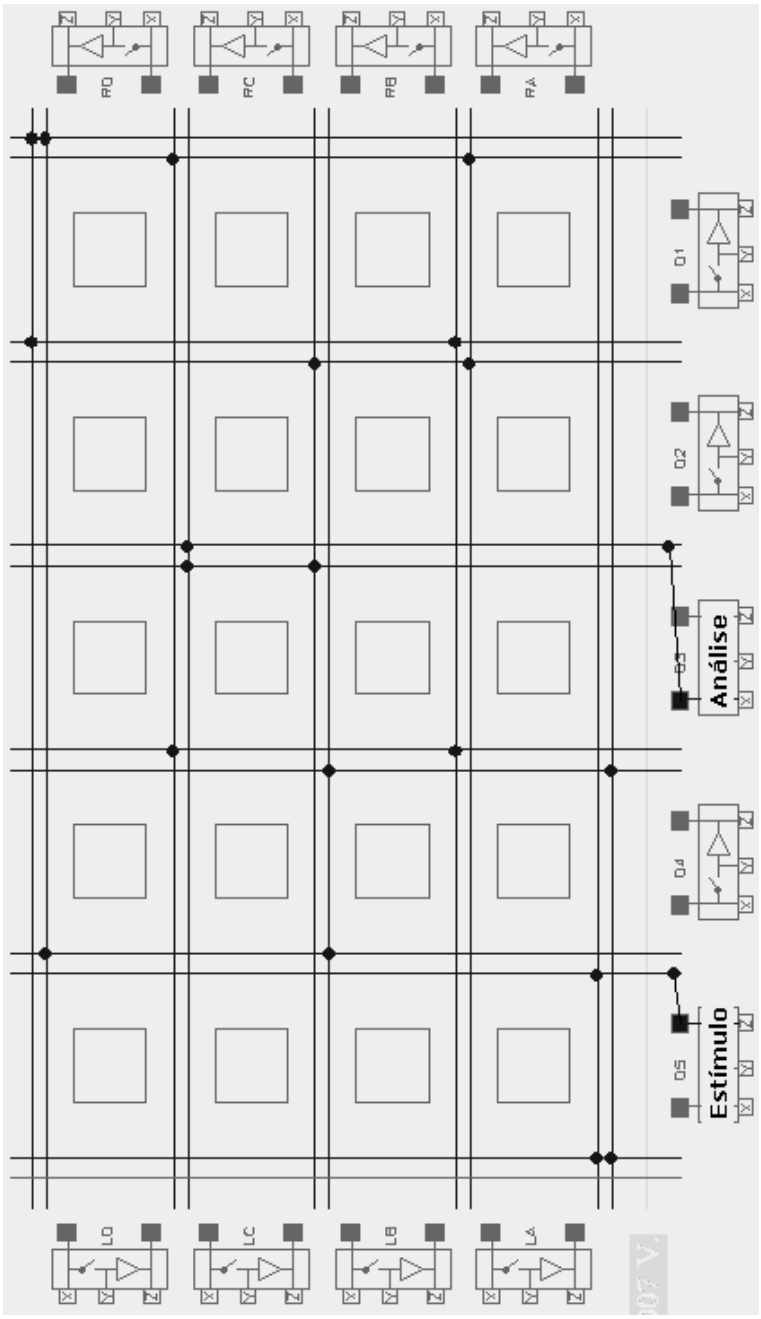
[XIL 2000] XILINX CORP. **Virtex Data Sheet**, 2000. Disponível em: <http://www.xilinx.com/>. Acesso em: may. 2004.

[YIN 98] YINLEI, Yu. et al. A Diagnosis Method for Interconnects in SRAM Based FPGAs. In: ASIAN TEST SYMPOSIUM, ATS, 7., 1998. **Proceedings…**[S.l.: s.n.], 1998.

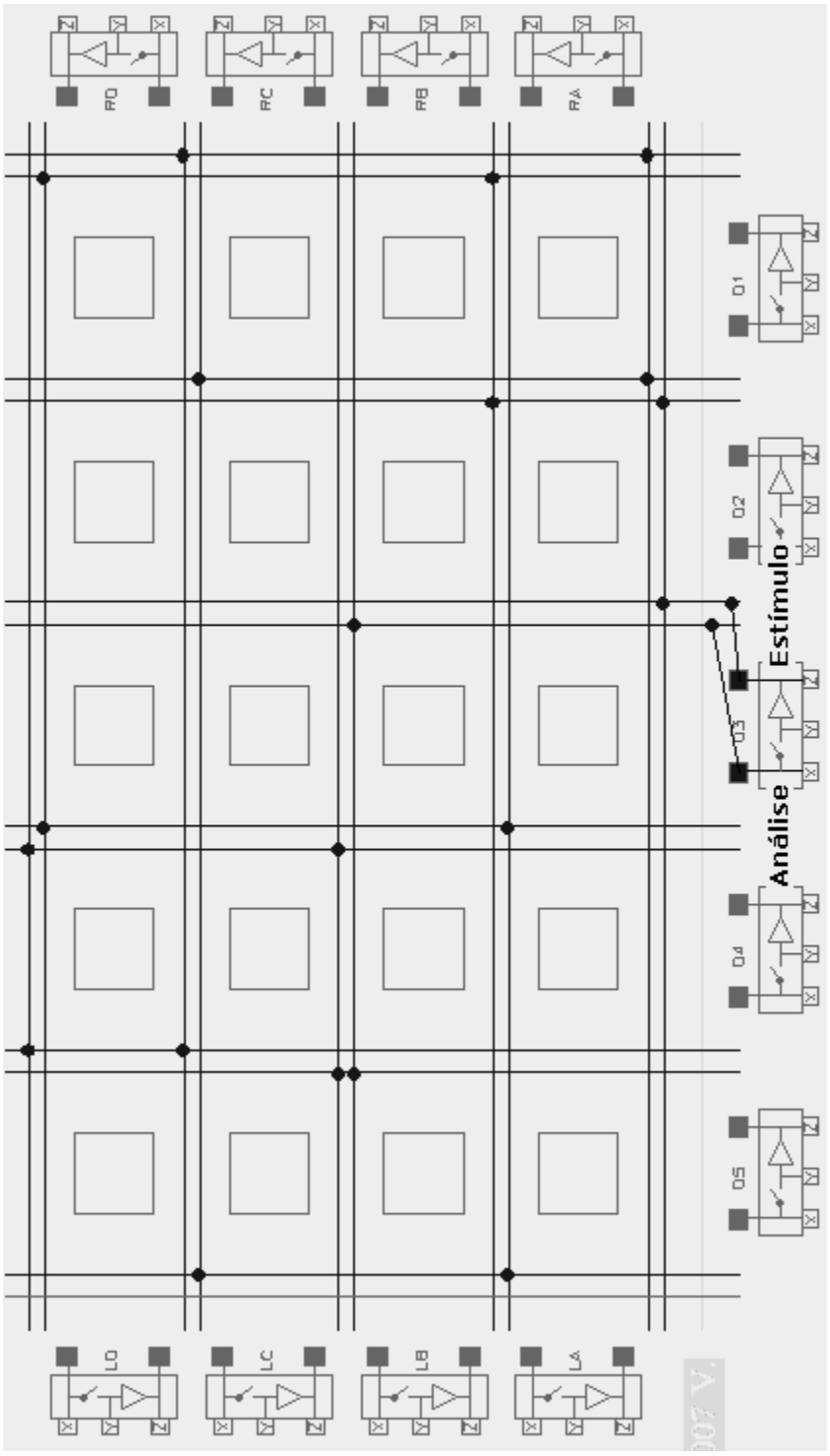
[ZHO 98] ZHOU, H.; WONG, D. F. Global Routing with Crosstalk Constraints. In: DESIGN AUTOMATION CONFERENCE, DAC, 1998, San Francisco. **Proceedings…** San Francisco: [s.n.], 1998. p. 374-377.

APÊNDICE A CTS DA REDE GLOBAL – FALHAS STUCK-OPEN – CAMINHOS NÃO BUFFERIZADOS

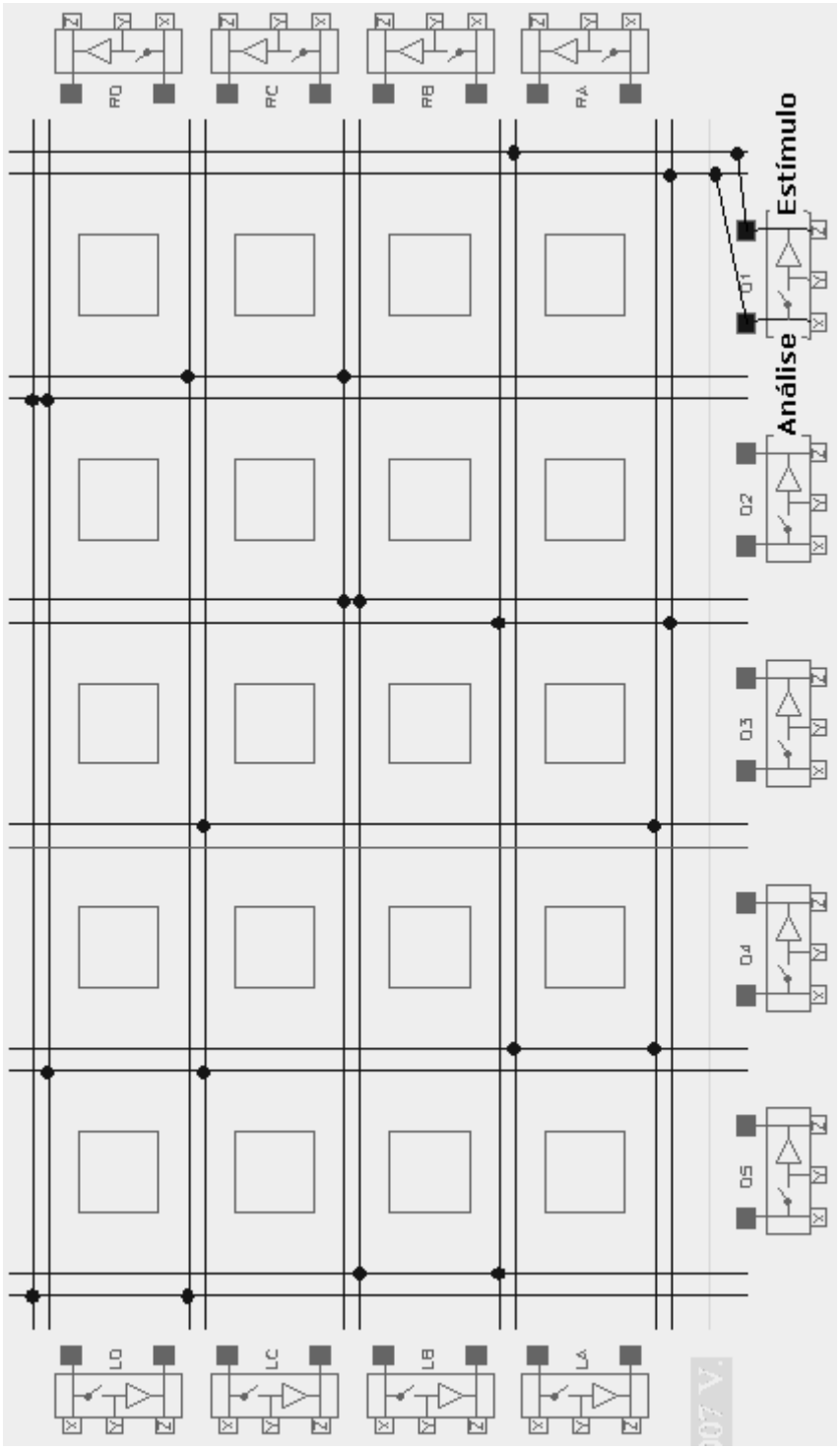
1ª CONFIGURAÇÃO DE TESTE



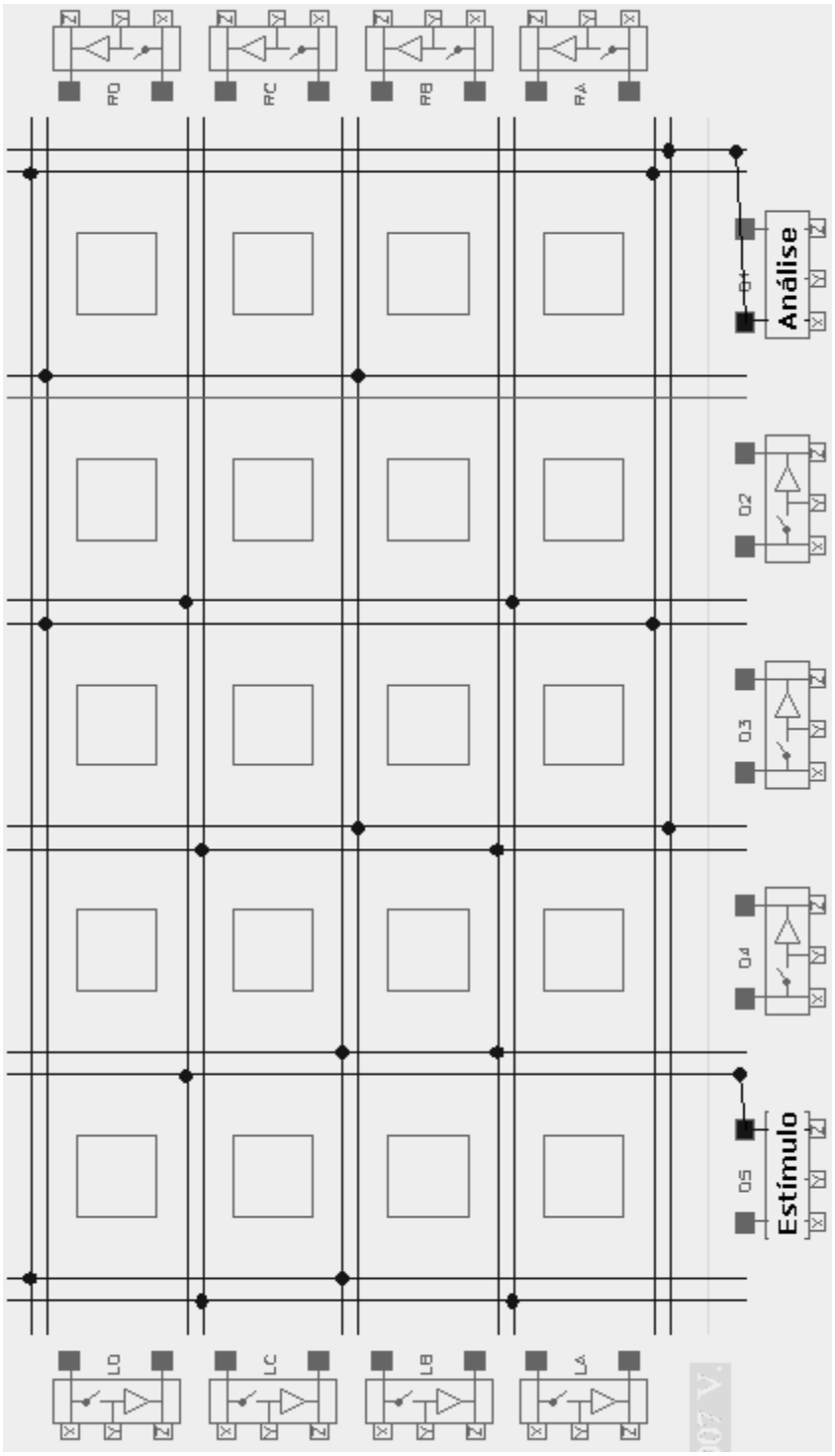
2ª CONFIGURAÇÃO DE TESTE



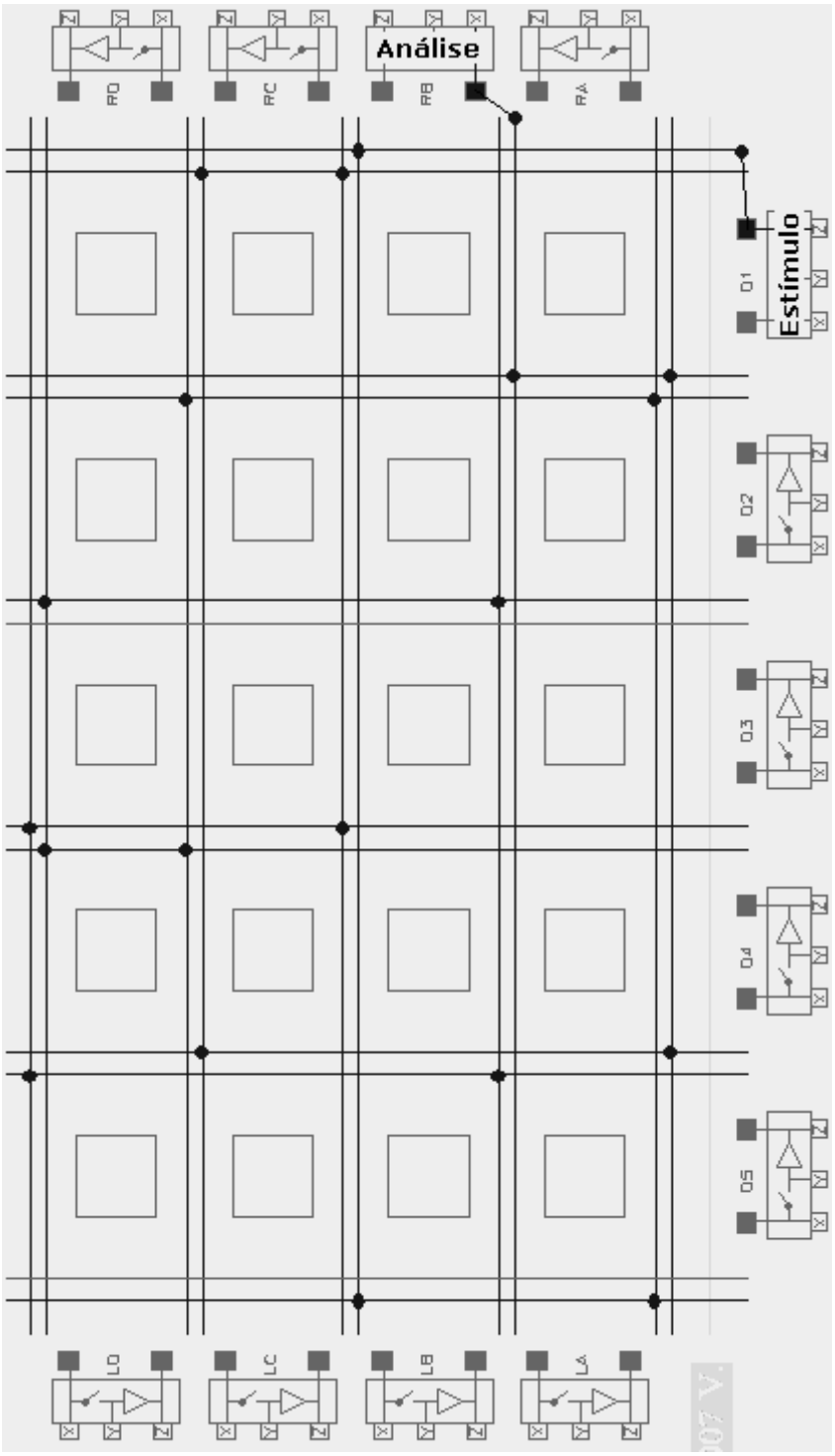
3ª CONFIGURAÇÃO DE TESTE



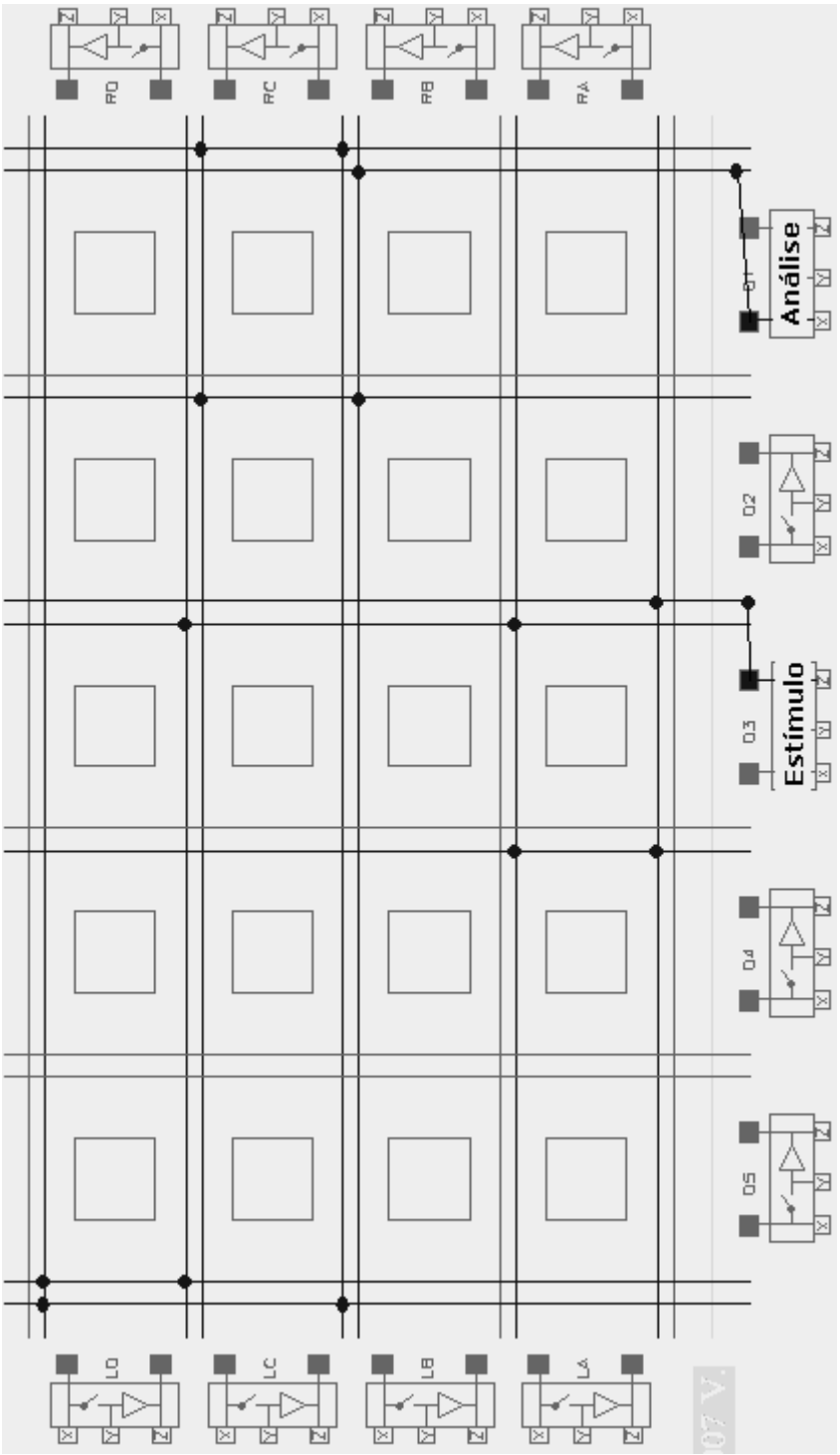
4ª CONFIGURAÇÃO DE TESTE



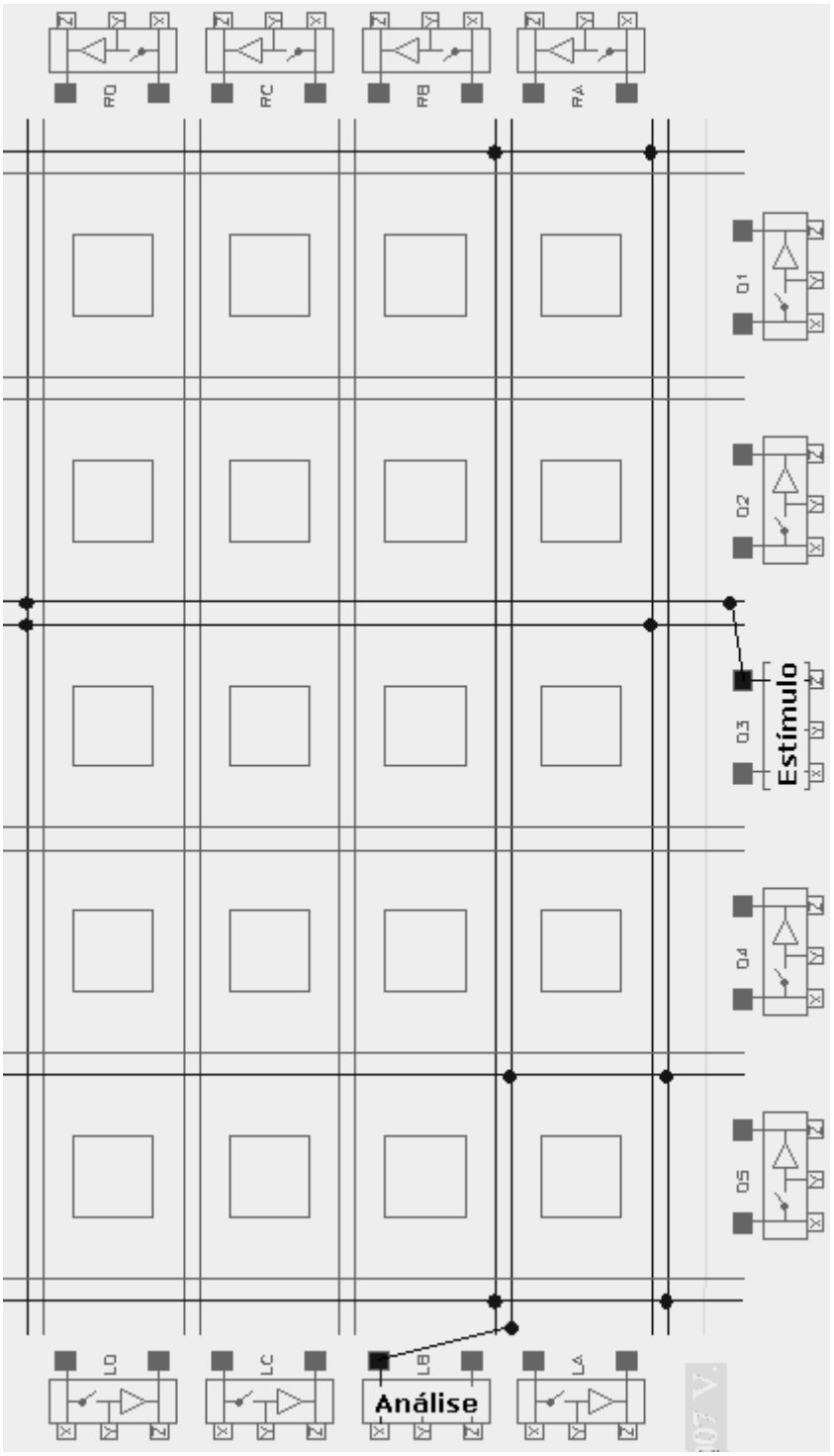
5ª CONFIGURAÇÃO DE TESTE



6ª CONFIGURAÇÃO DE TESTE

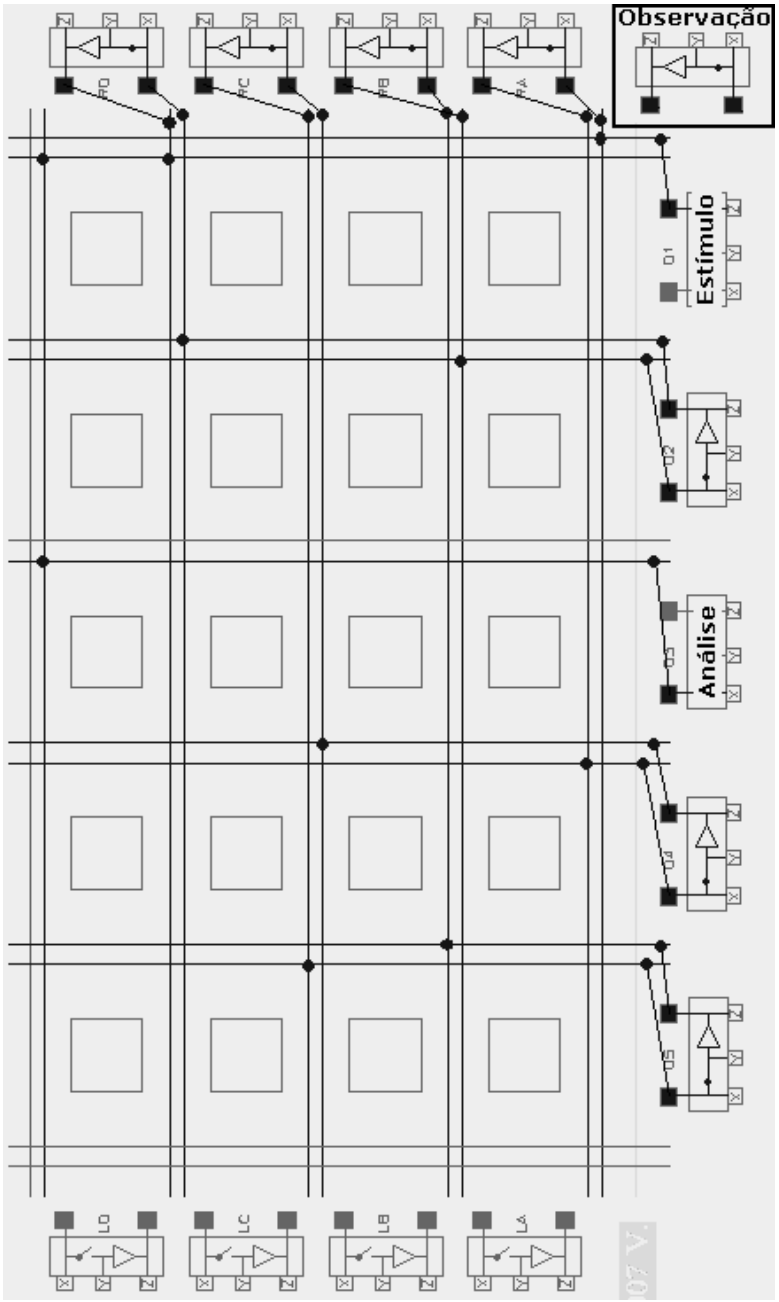


7ª CONFIGURAÇÃO DE TESTE

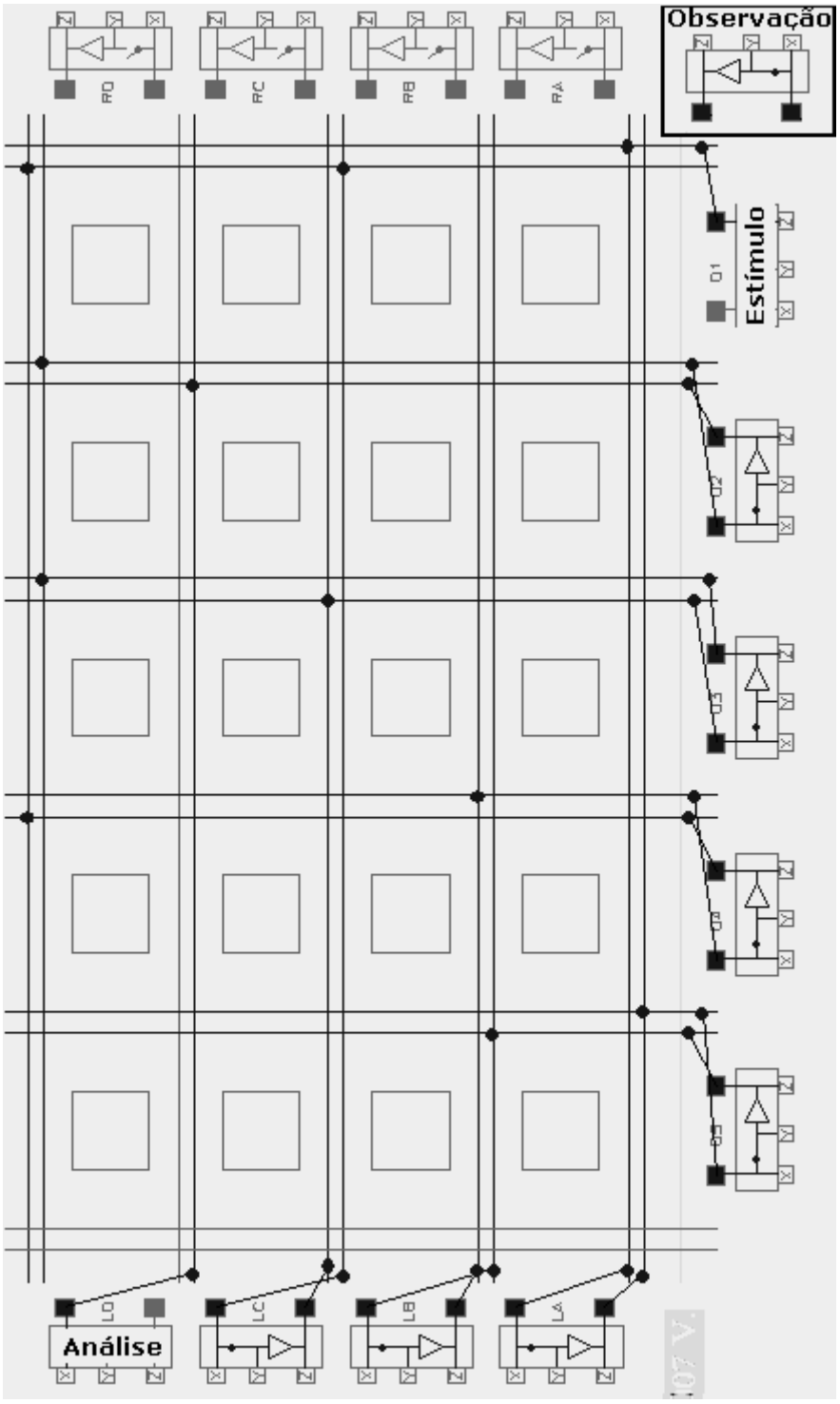


APÊNDICE B CTS DA REDE GLOBAL – FALHAS STUCK-OPEN – CAMINHOS BUFFERIZADOS

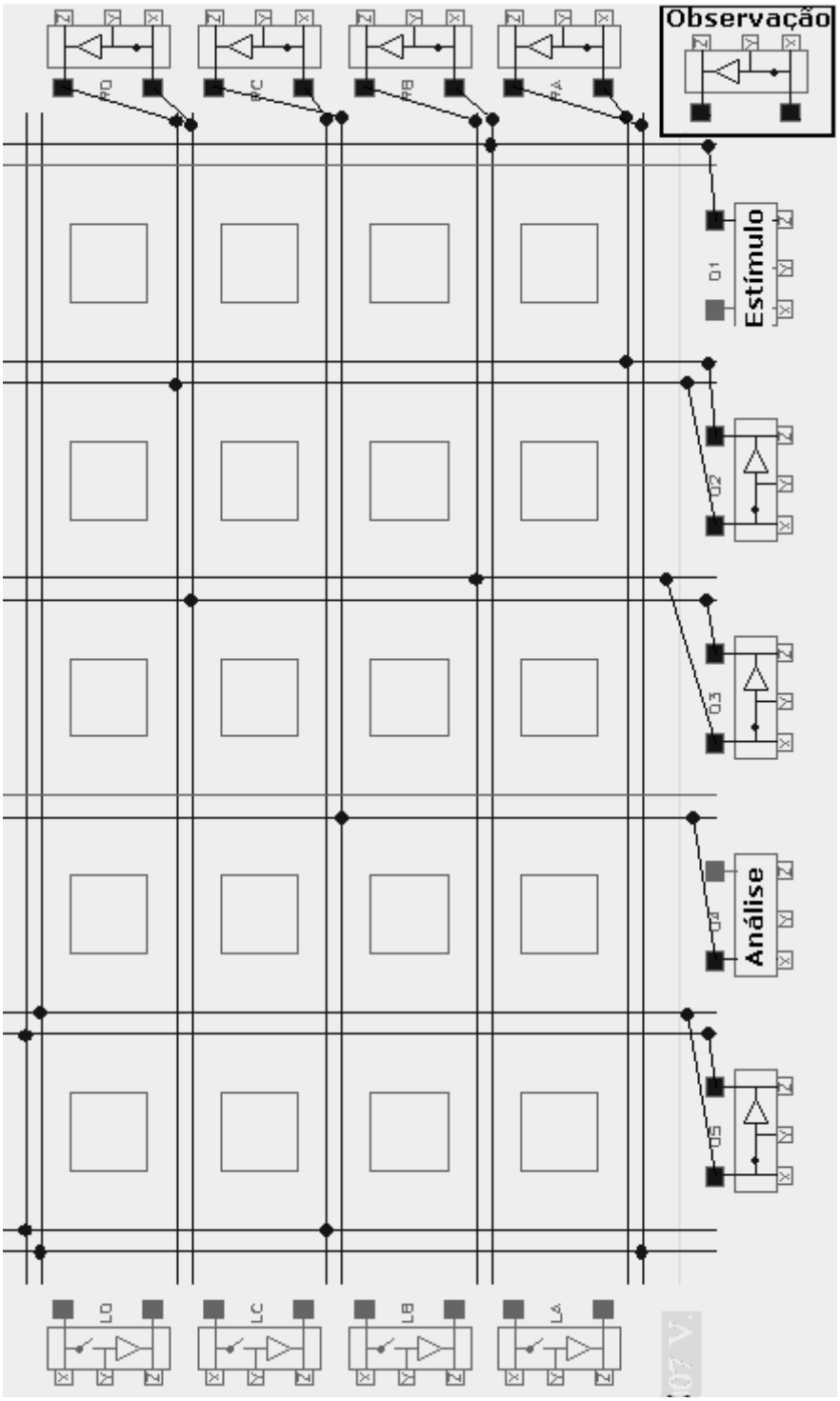
1ª CONFIGURAÇÃO DE TESTE



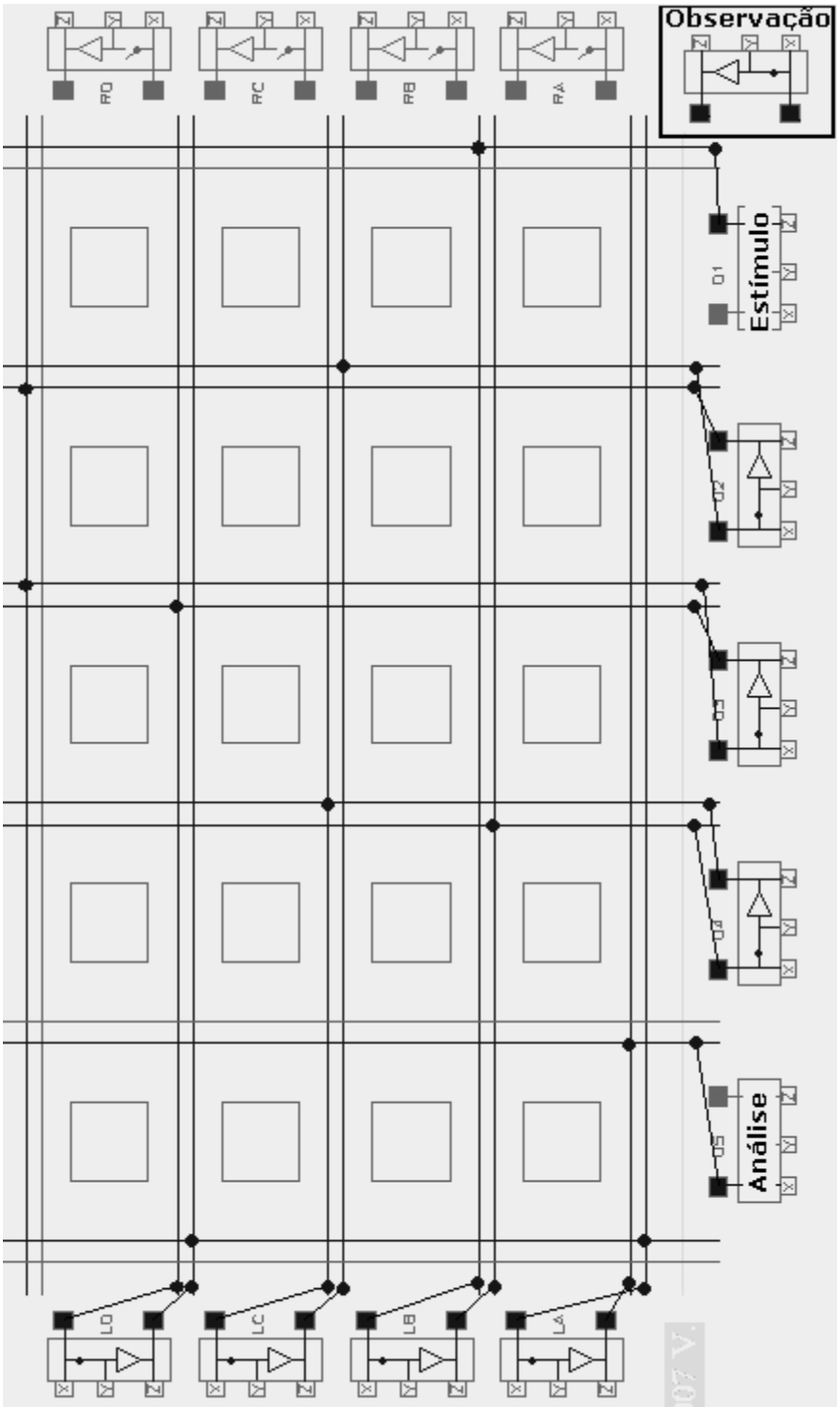
2ª CONFIGURAÇÃO DE TESTE



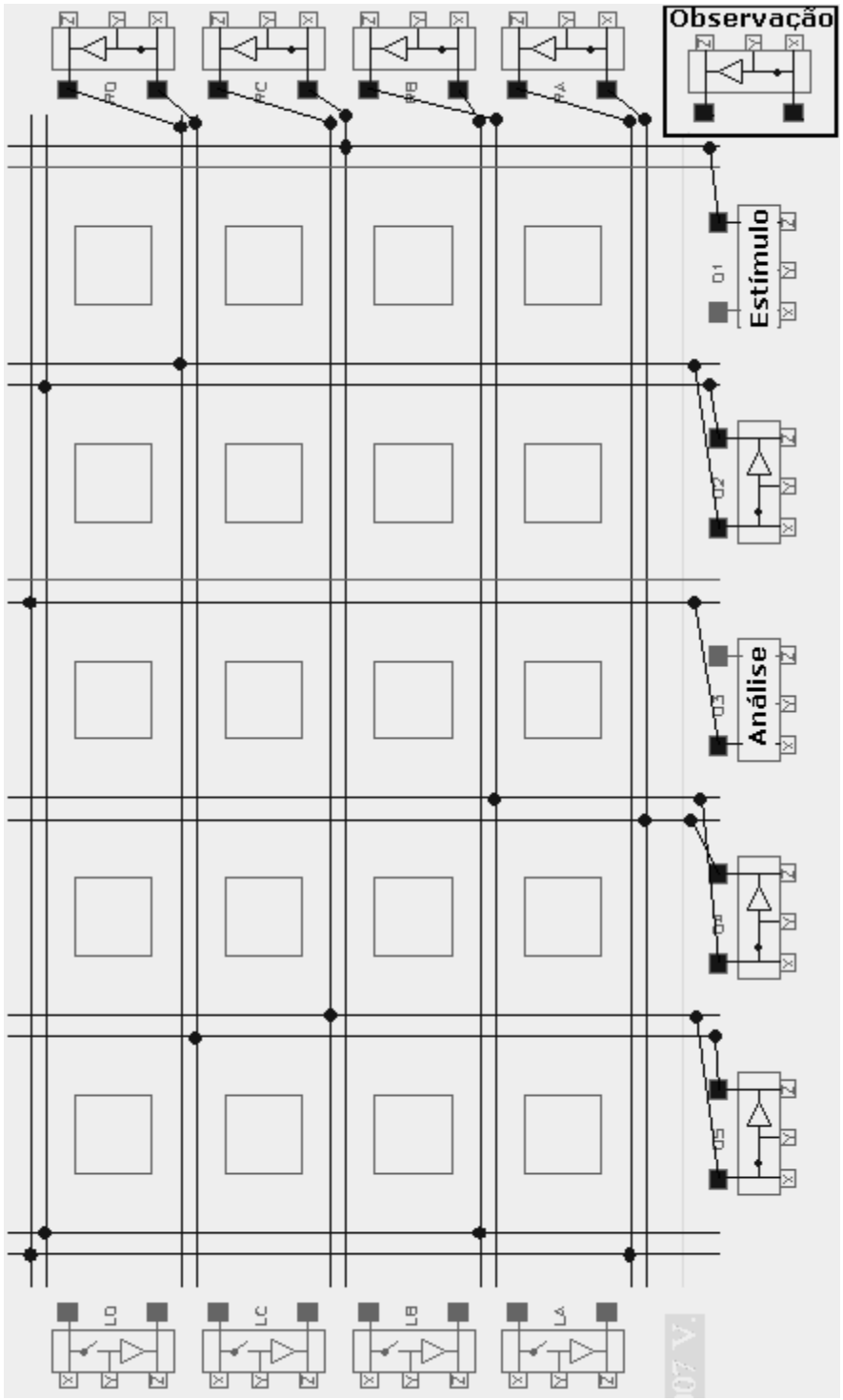
3ª CONFIGURAÇÃO DE TESTE



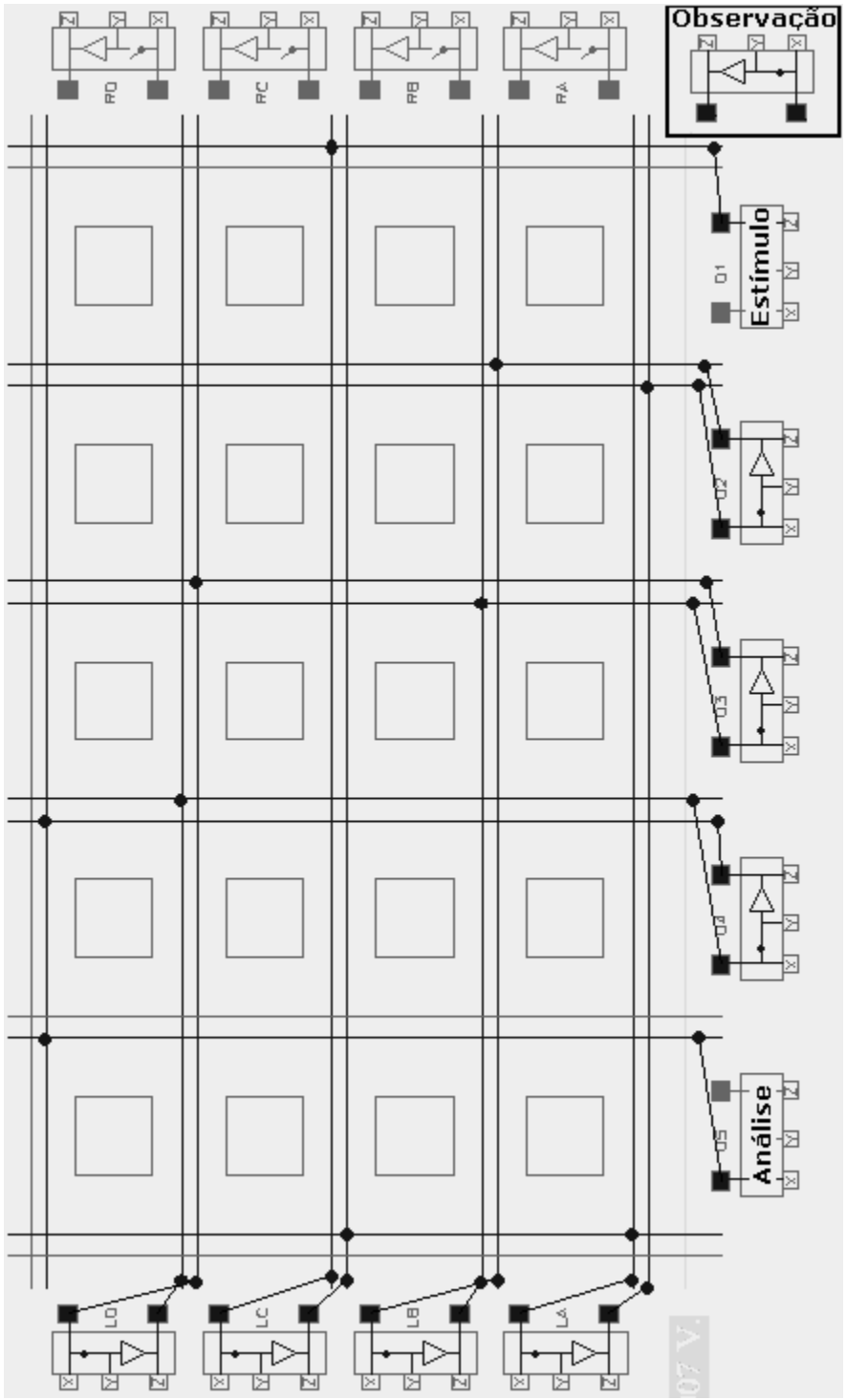
4ª CONFIGURAÇÃO DE TESTE



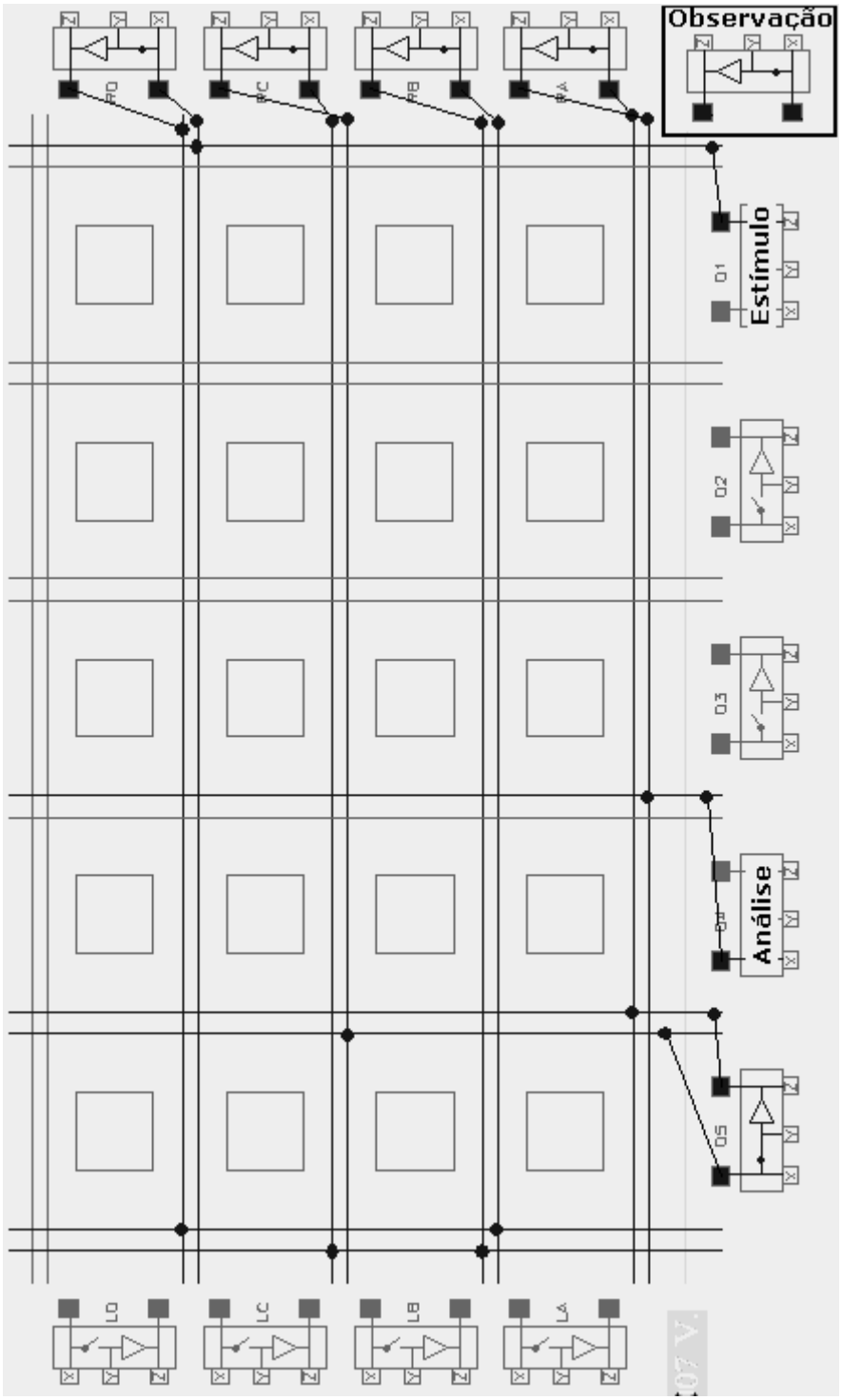
5ª CONFIGURAÇÃO DE TESTE



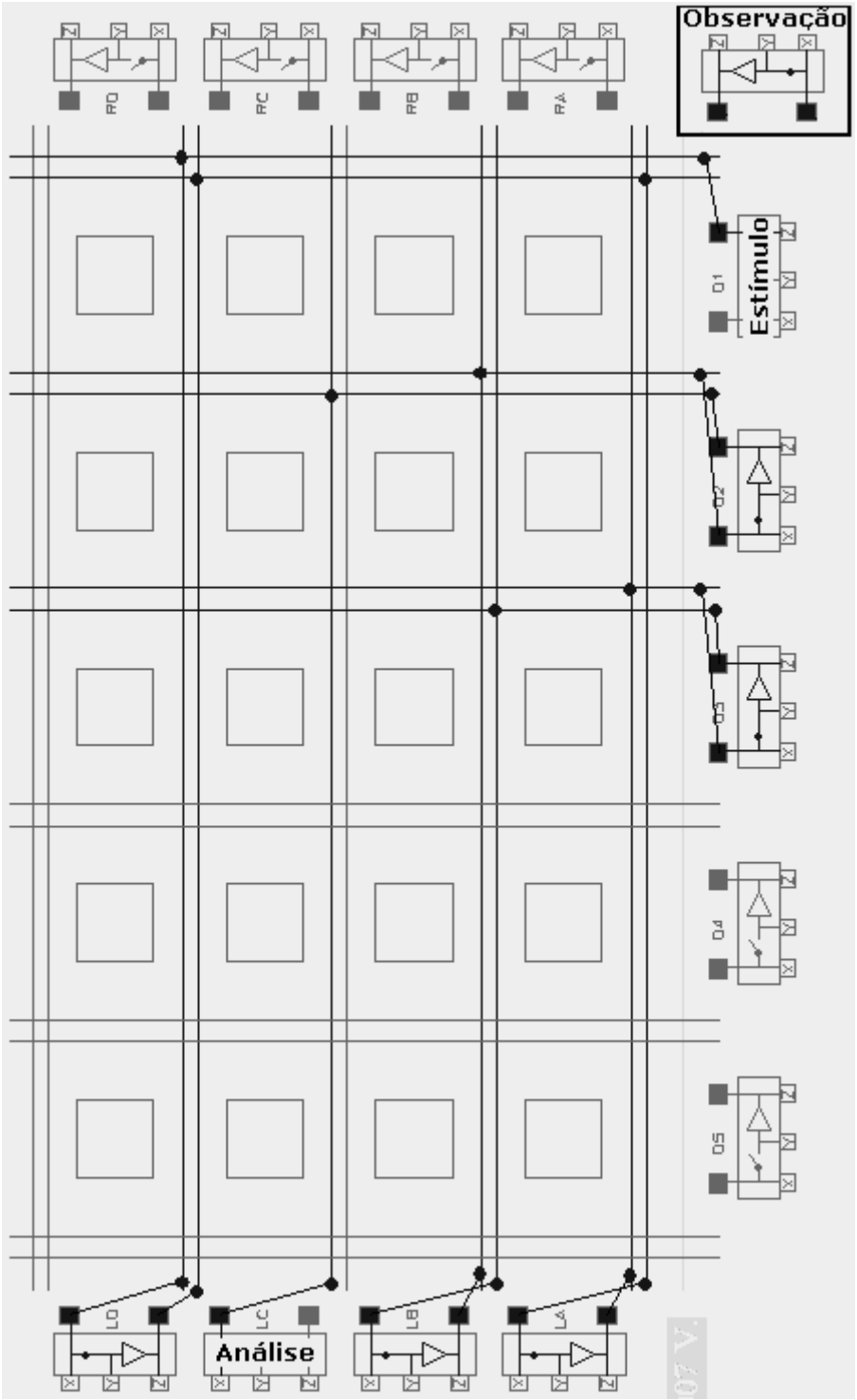
6ª CONFIGURAÇÃO DE TESTE



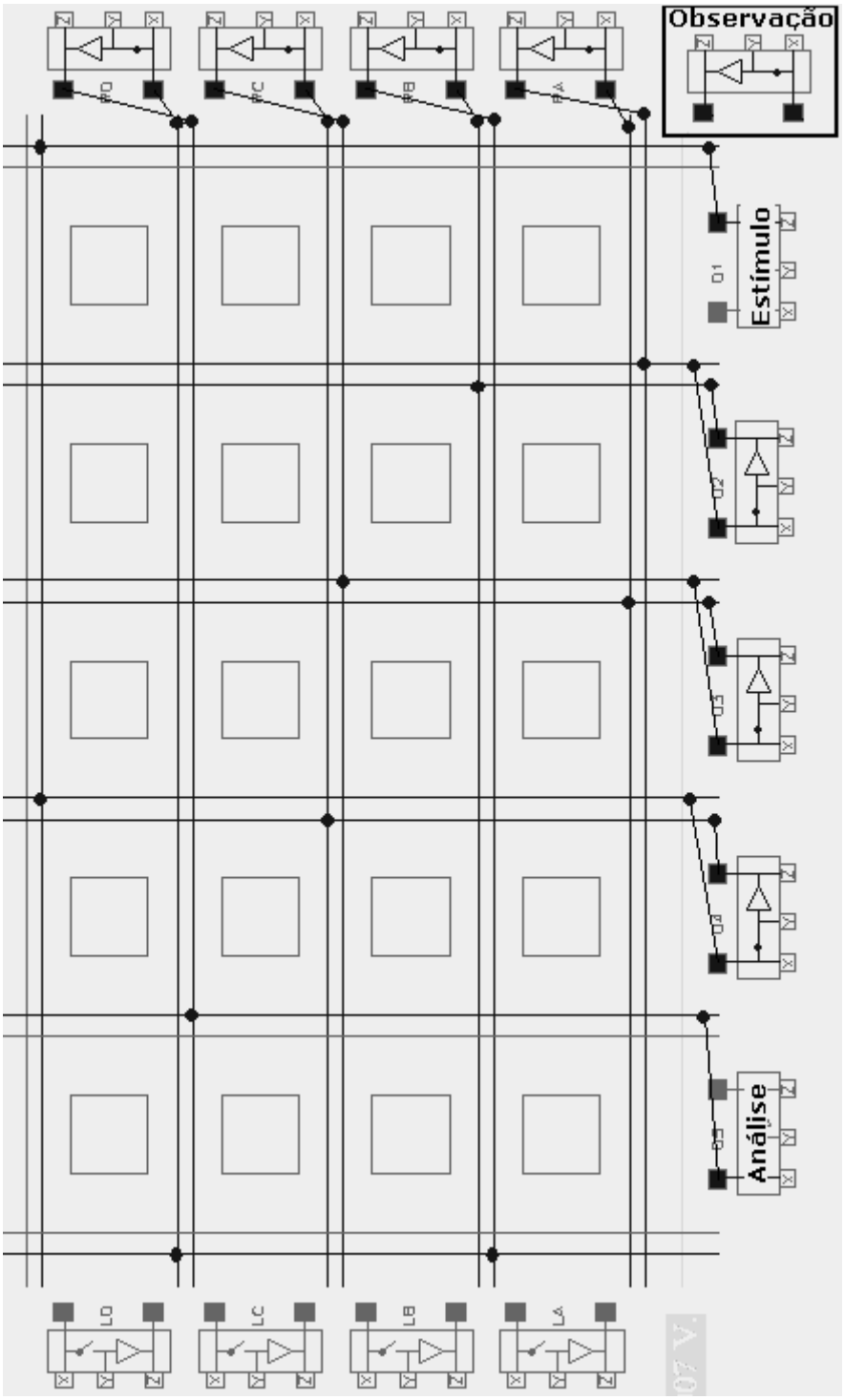
7ª CONFIGURAÇÃO DE TESTE



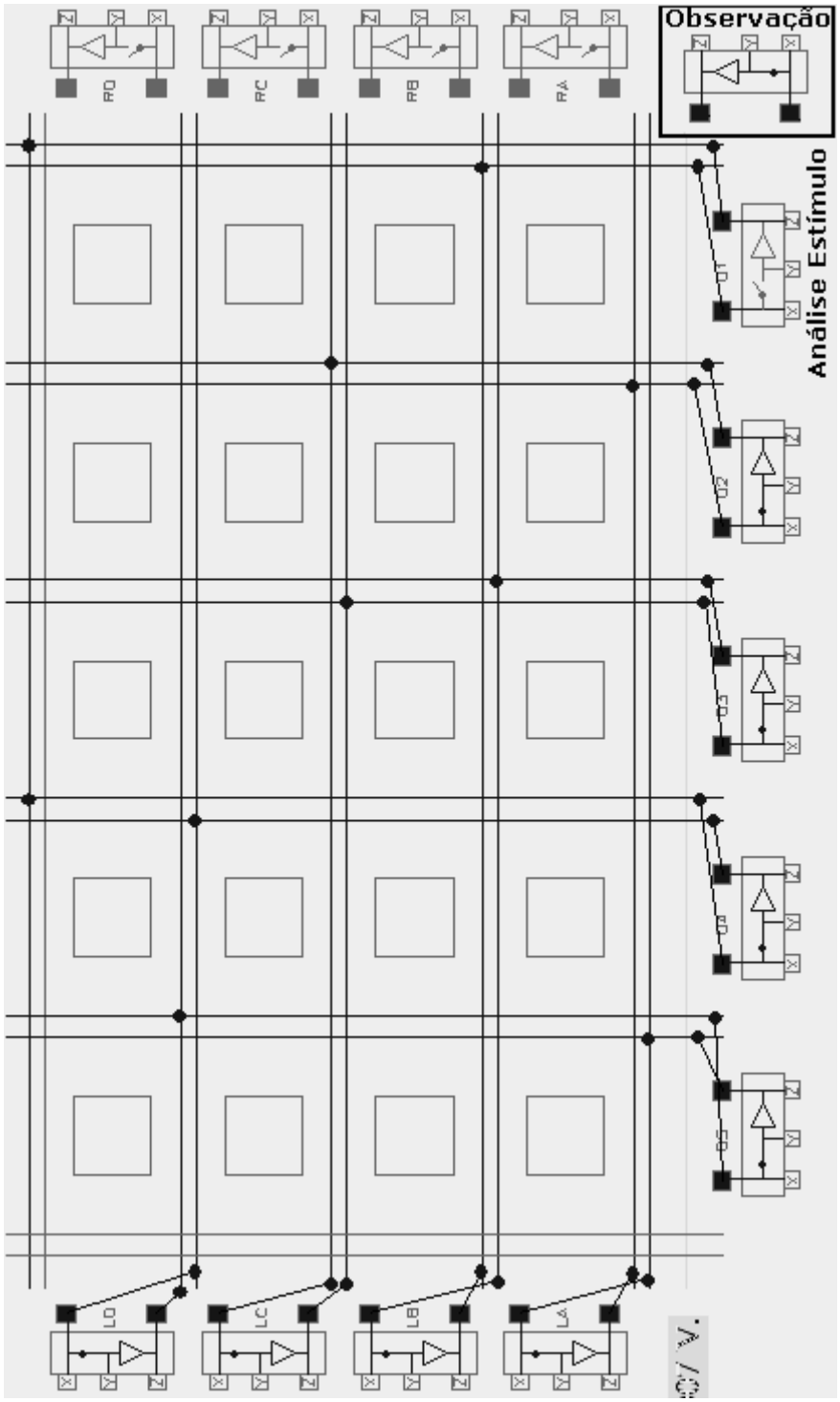
8ª CONFIGURAÇÃO DE TESTE



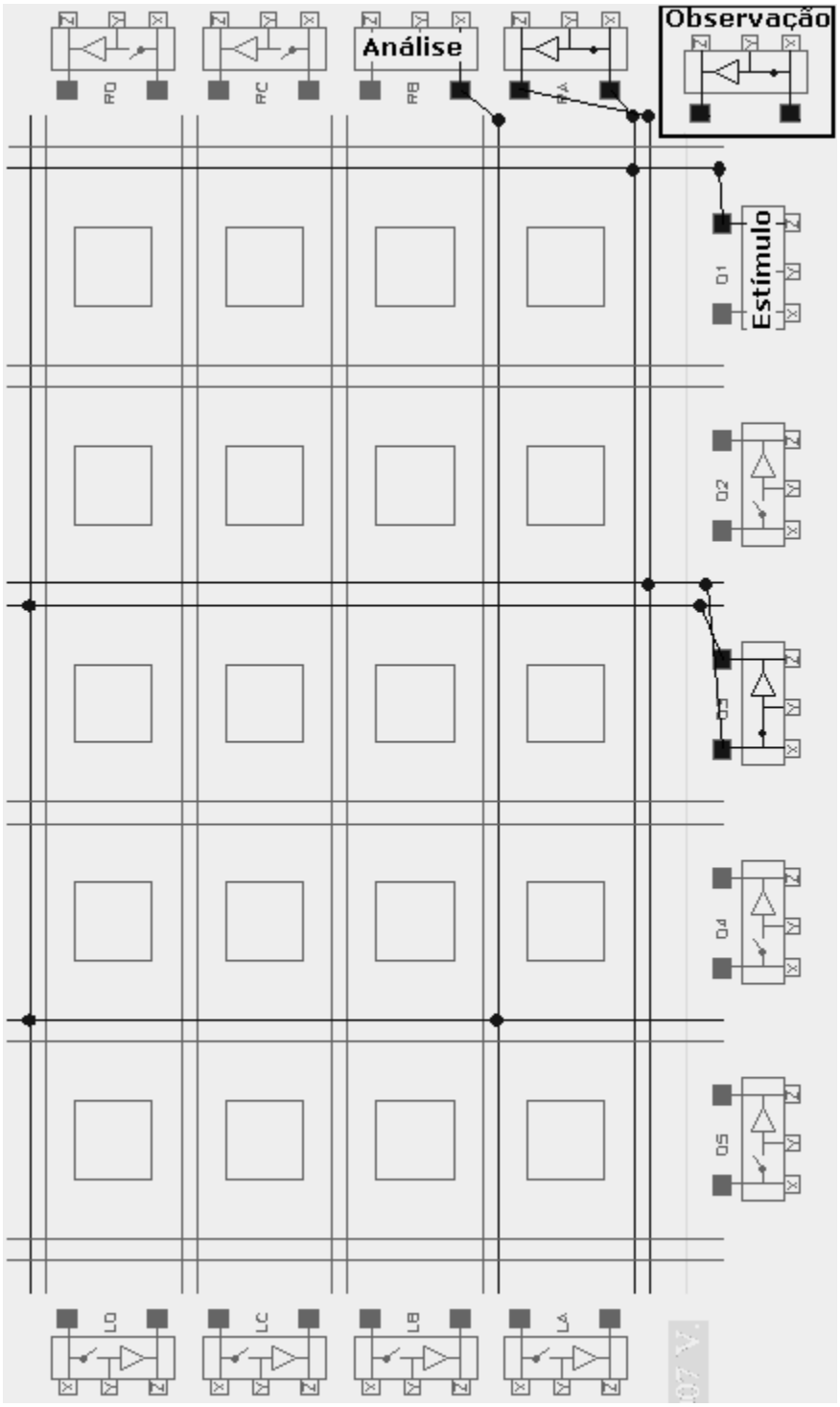
9ª CONFIGURAÇÃO DE TESTE



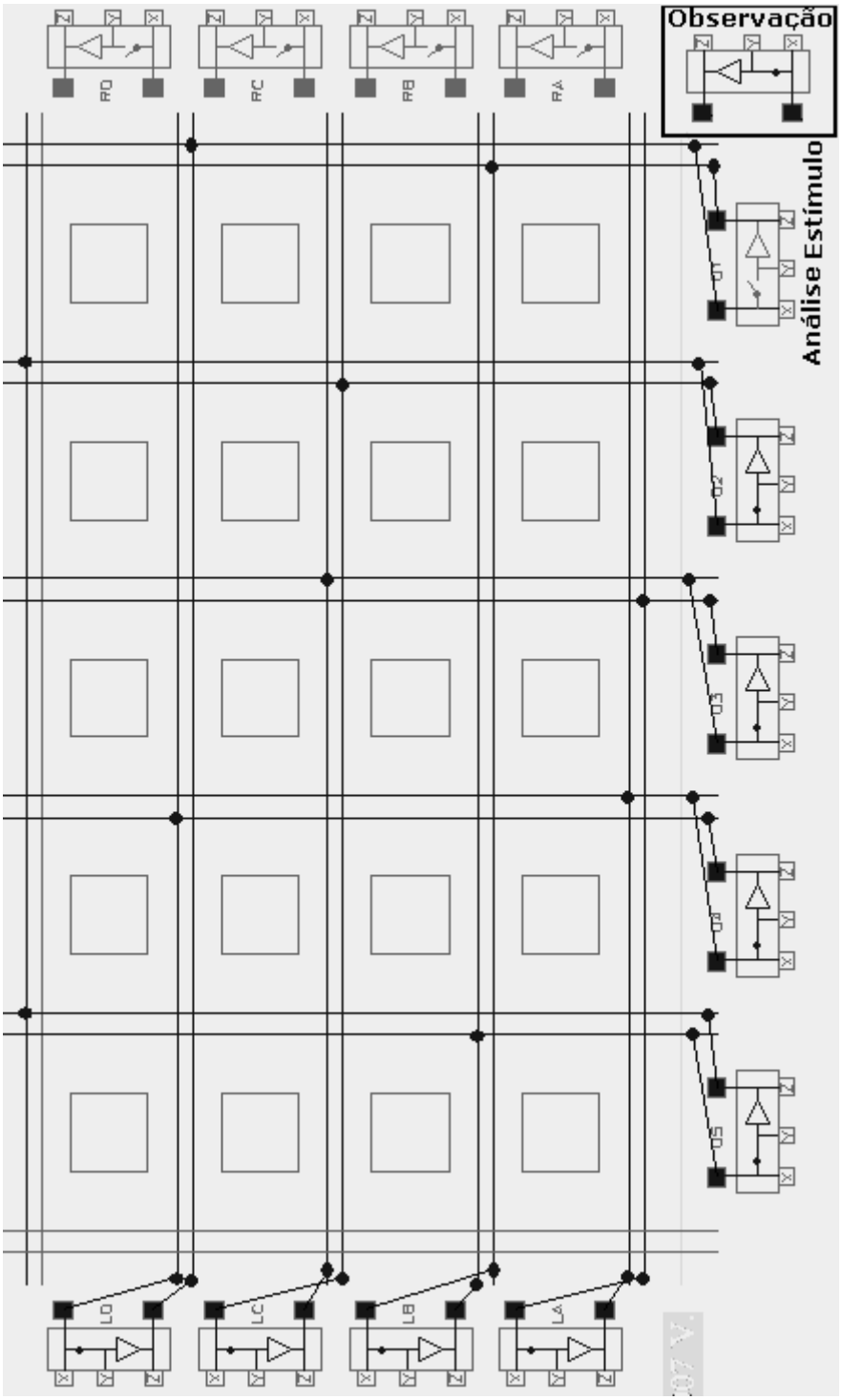
10ª CONFIGURAÇÃO DE TESTE



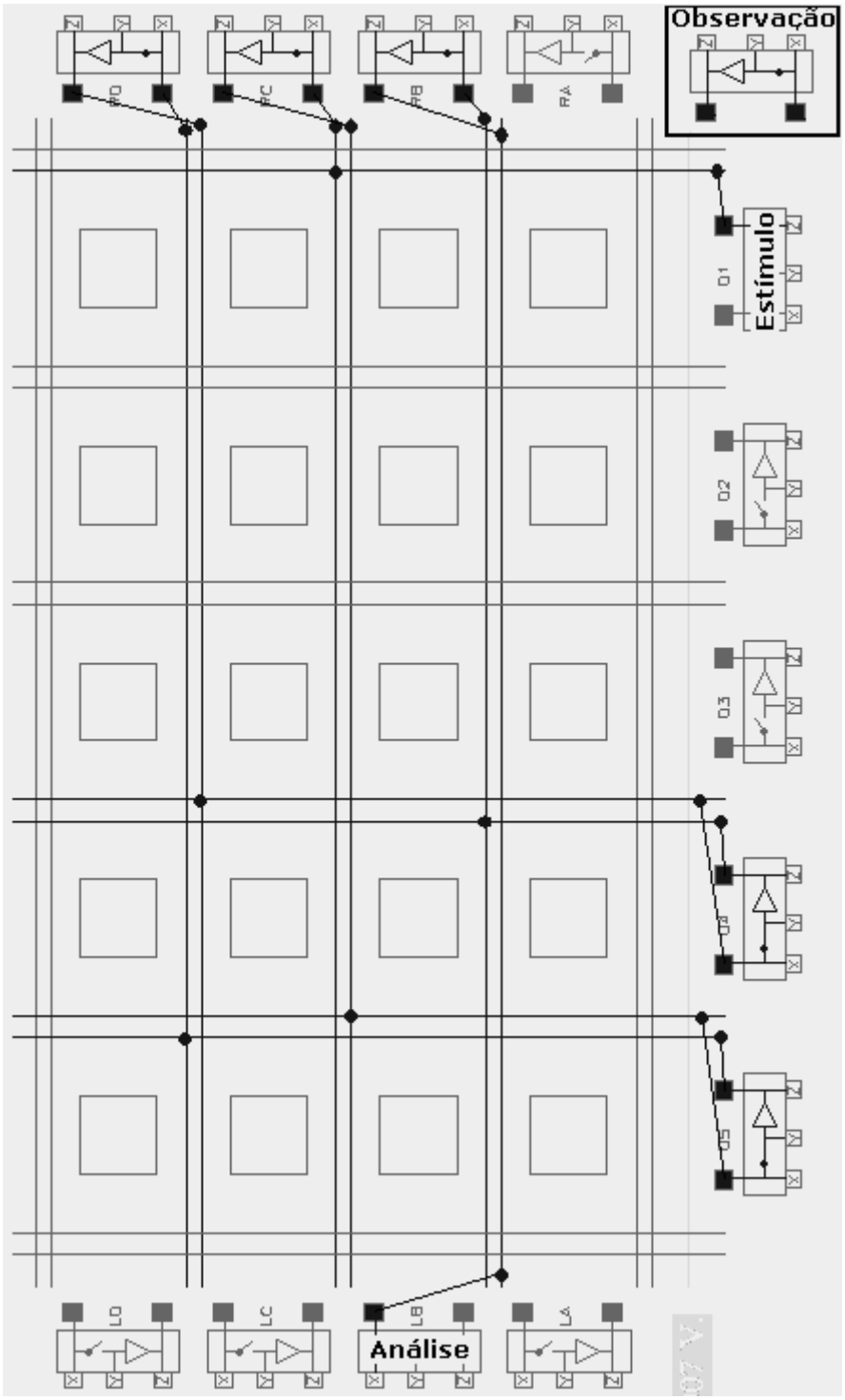
11ª CONFIGURAÇÃO DE TESTE



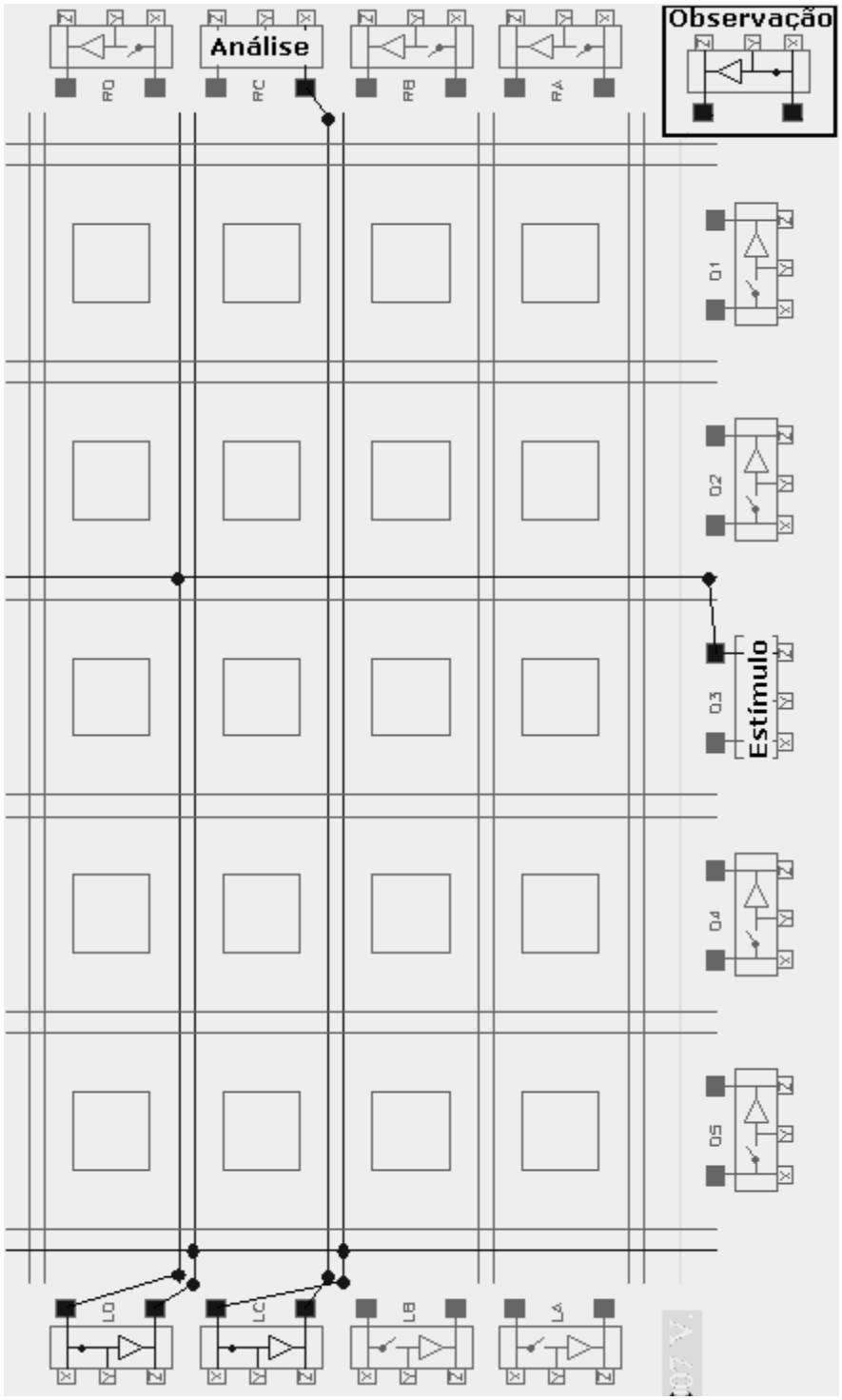
12ª CONFIGURAÇÃO DE TESTE



13ª CONFIGURAÇÃO DE TESTE

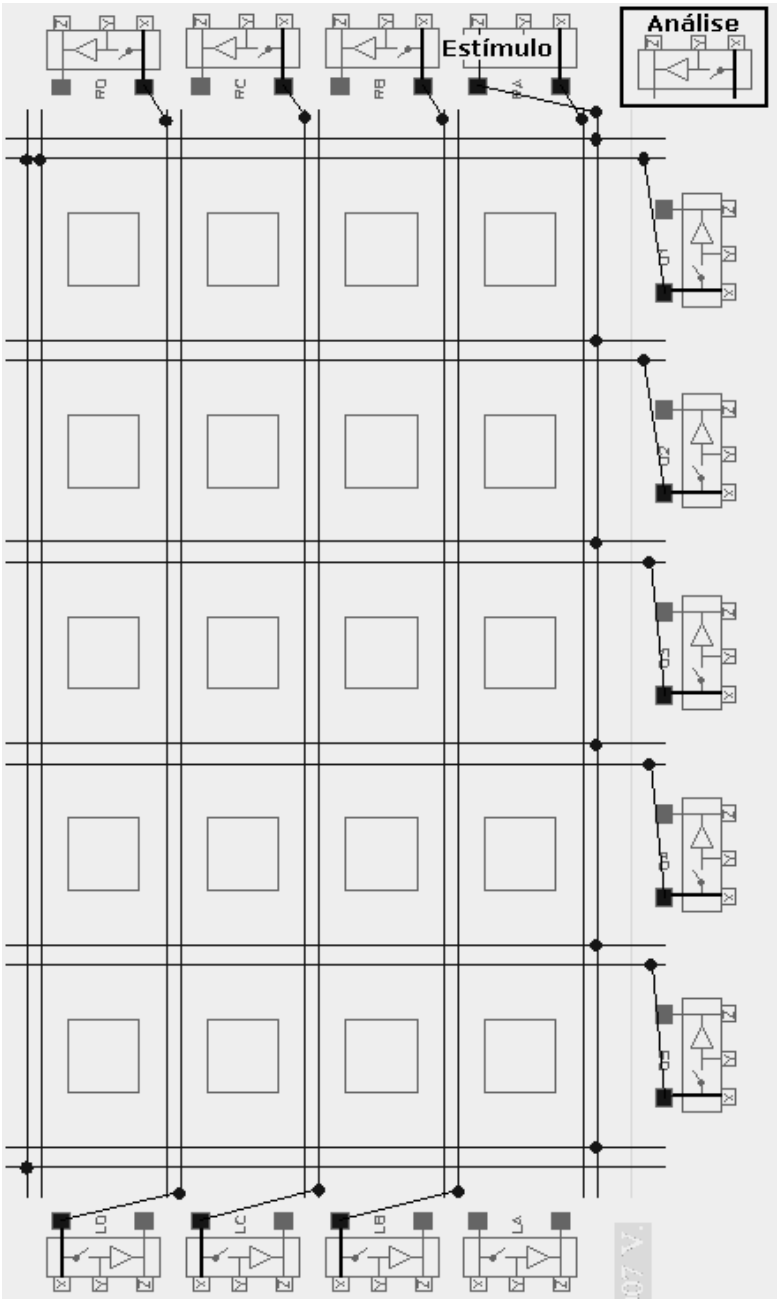


14ª CONFIGURAÇÃO DE TESTE

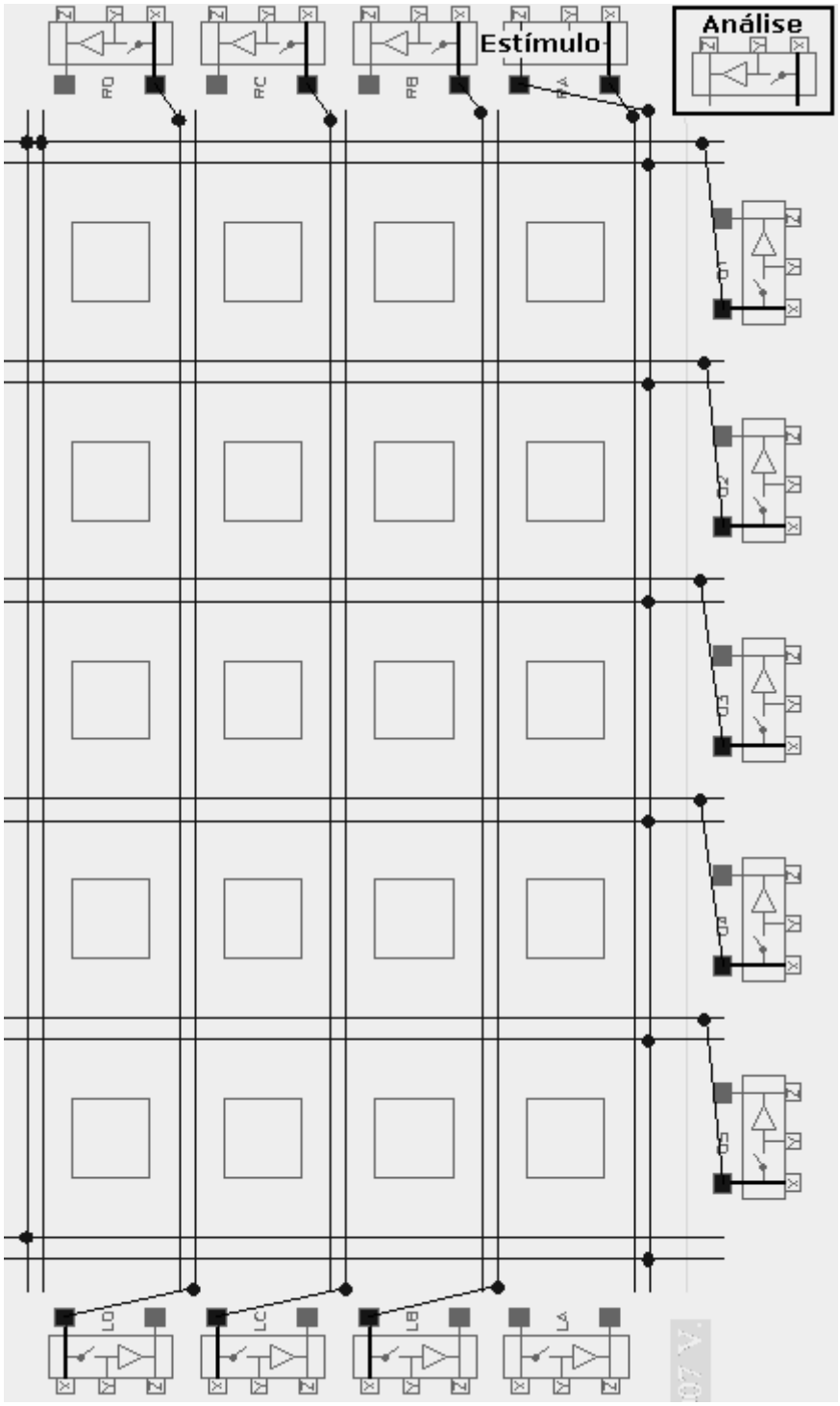


APÊNDICE C CTS DA REDE GLOBAL – FALHAS STUCK-ON – PRIMEIRA ESTRATÉGIA

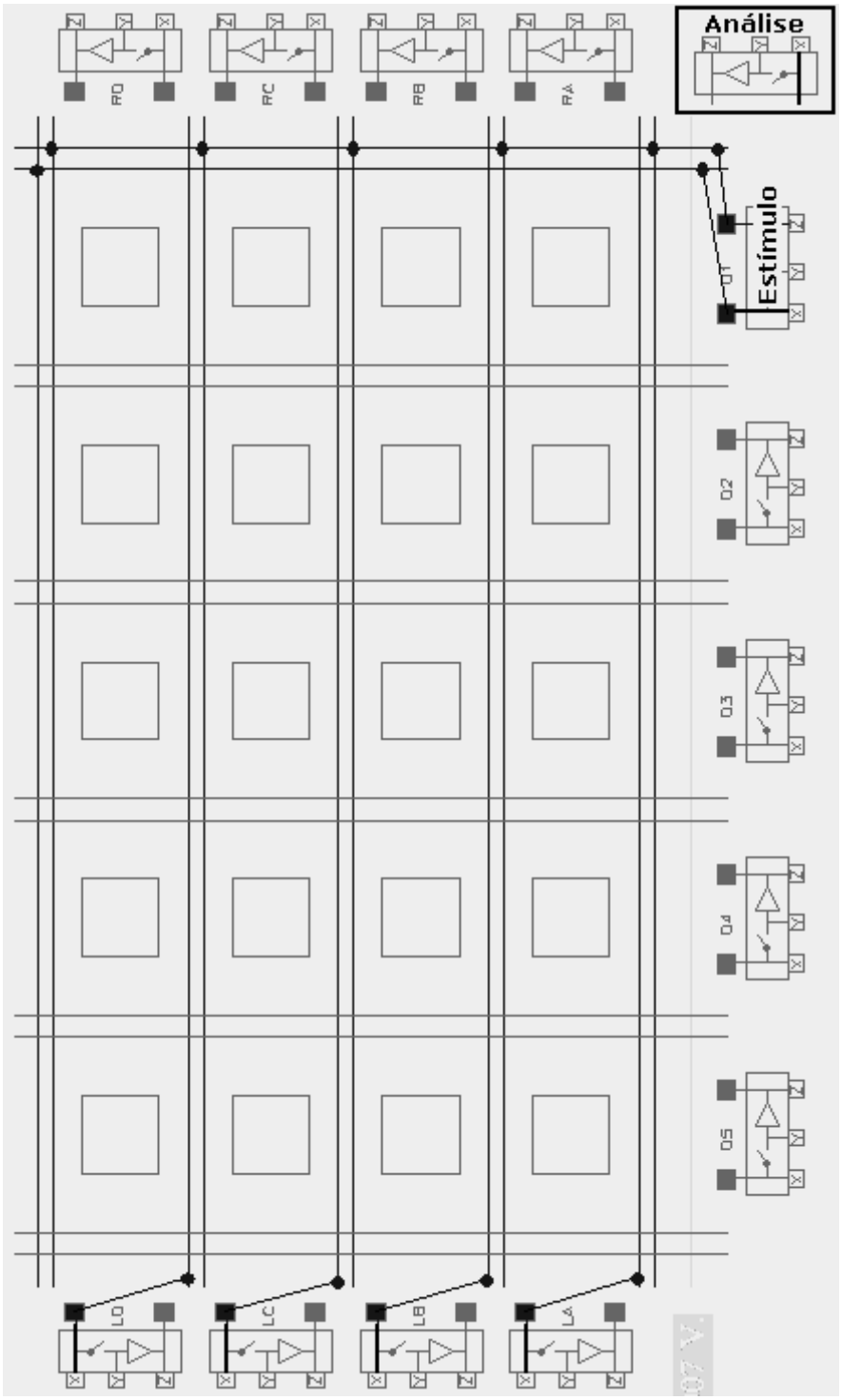
1ª CONFIGURAÇÃO DE TESTE



2ª CONFIGURAÇÃO DE TESTE

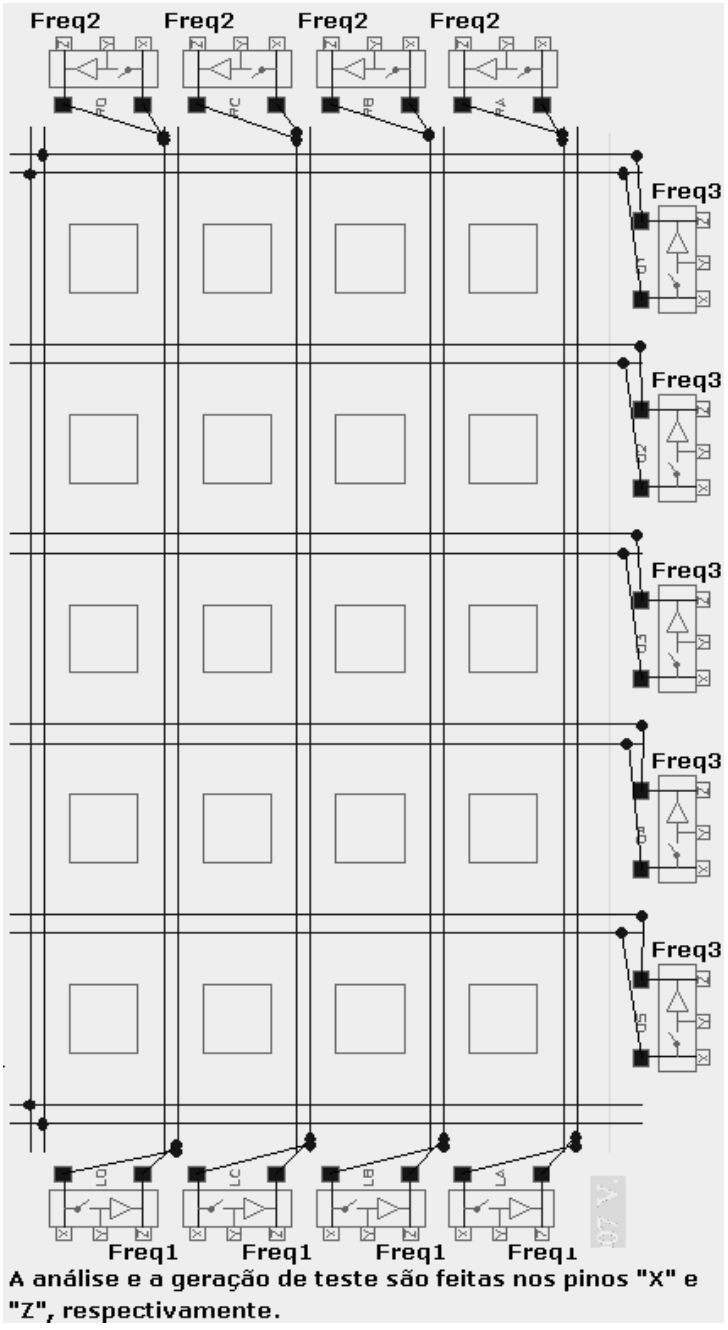


3ª CONFIGURAÇÃO DE TESTE

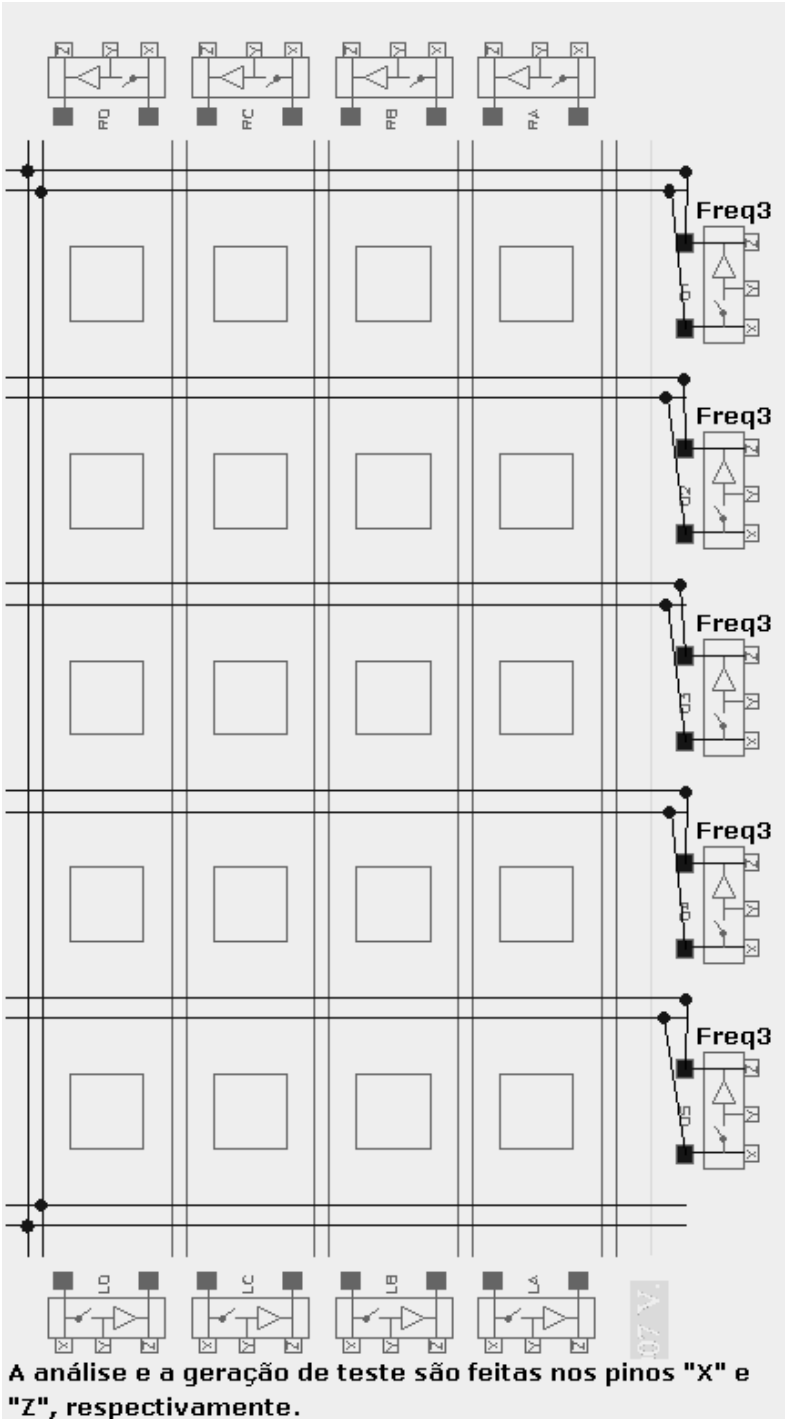


APÊNDICE D CTS DA REDE GLOBAL – FALHAS STUCK-ON – SEGUNDA ESTRATÉGIA

1ª CONFIGURAÇÃO DE TESTE

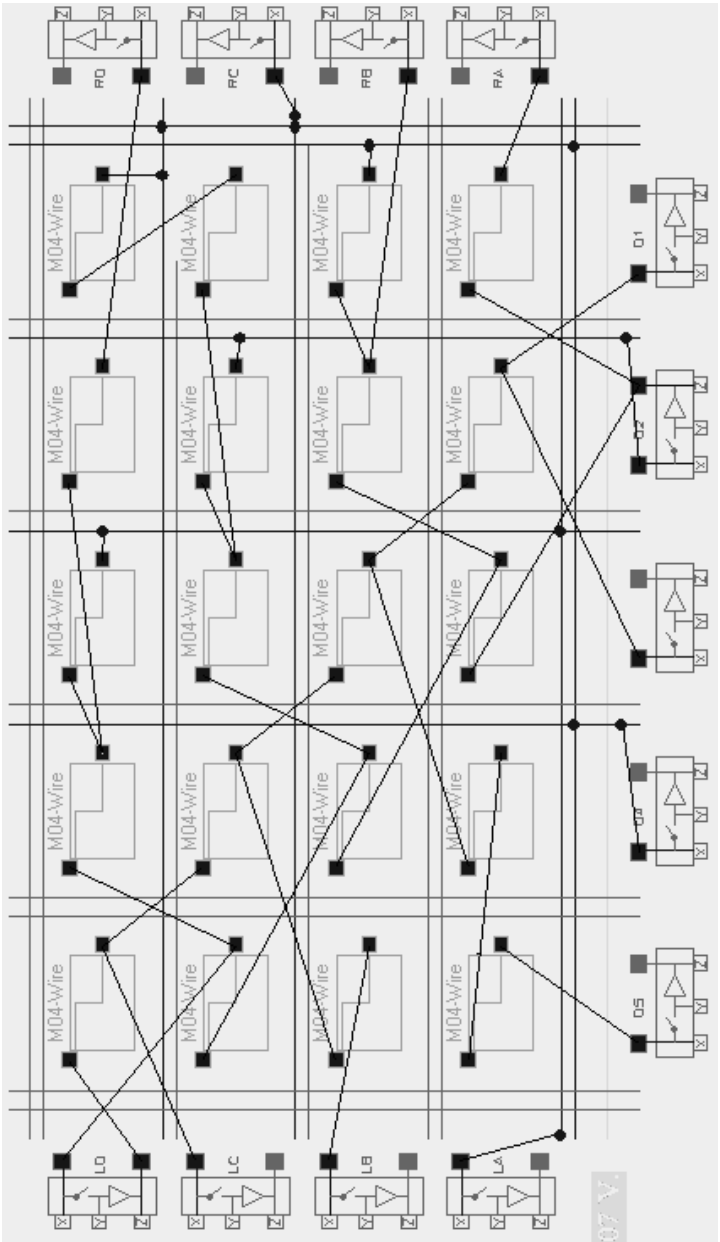


2ª CONFIGURAÇÃO DE TESTE



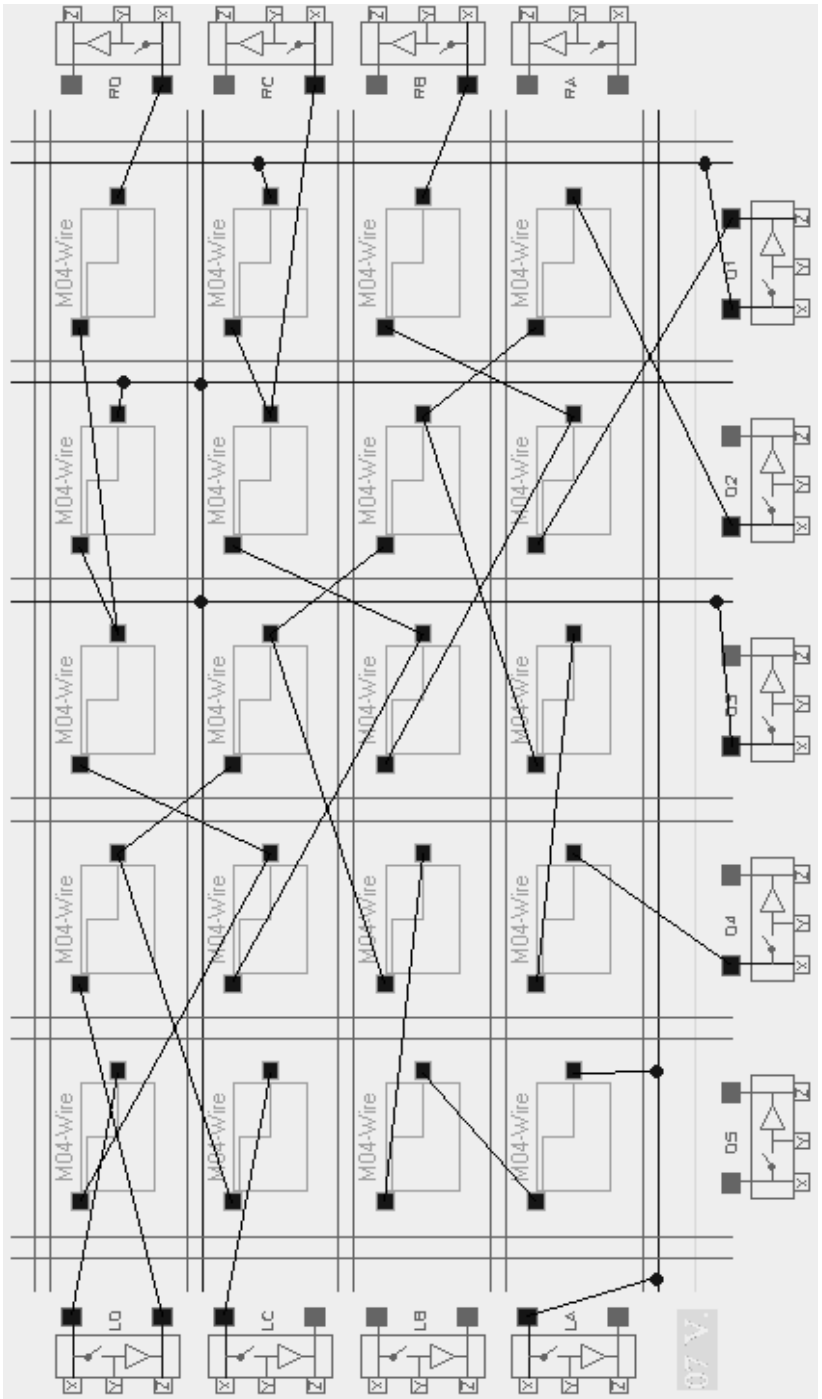
APÊNDICE E CTS DA REDE LOCAL – FALHAS STUCK-OPEN E STUCK-ON

1ª CONFIGURAÇÃO DE TESTE



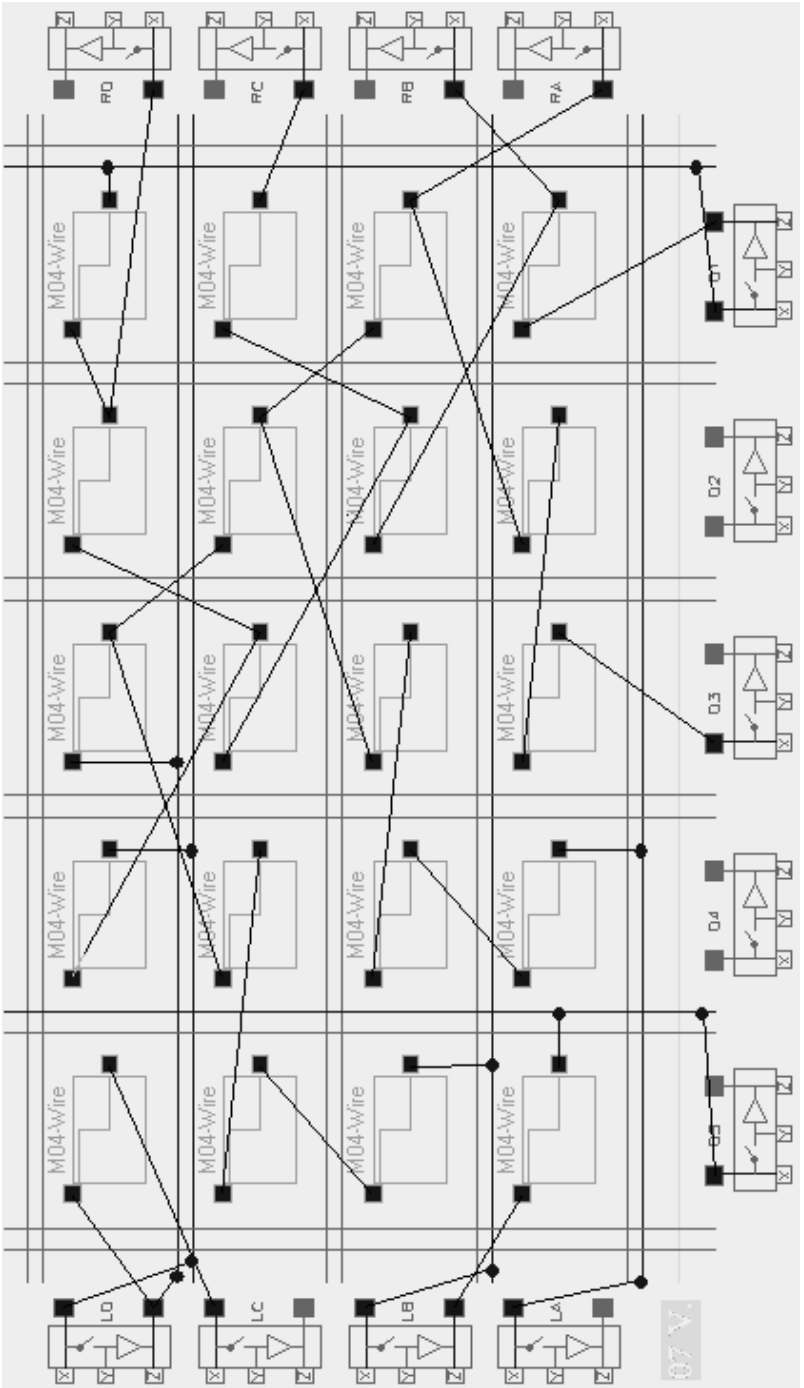
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

2ª CONFIGURAÇÃO DE TESTE



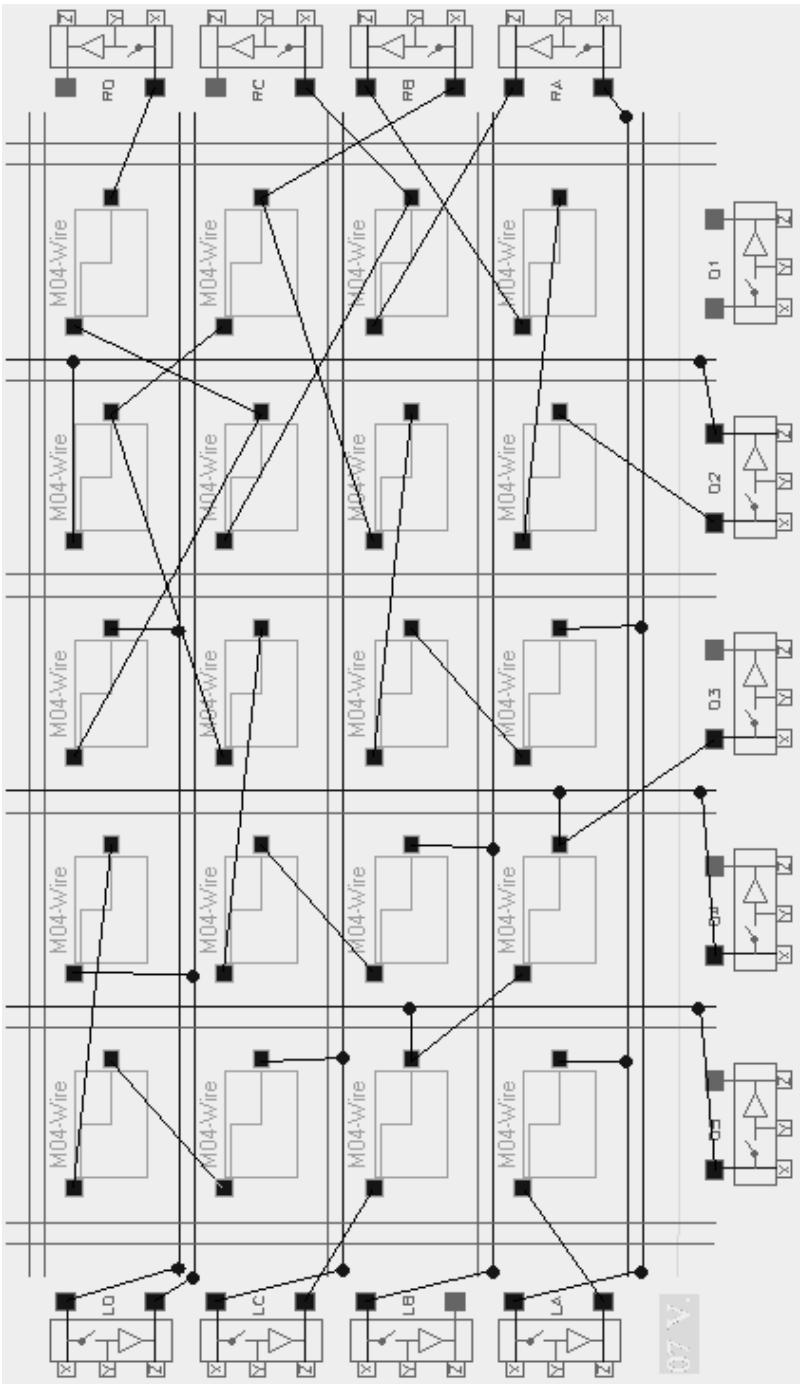
A análise e a geração de teste são feitas nos pinos 'X" e "Z", respectivamente.

3ª CONFIGURAÇÃO DE TESTE



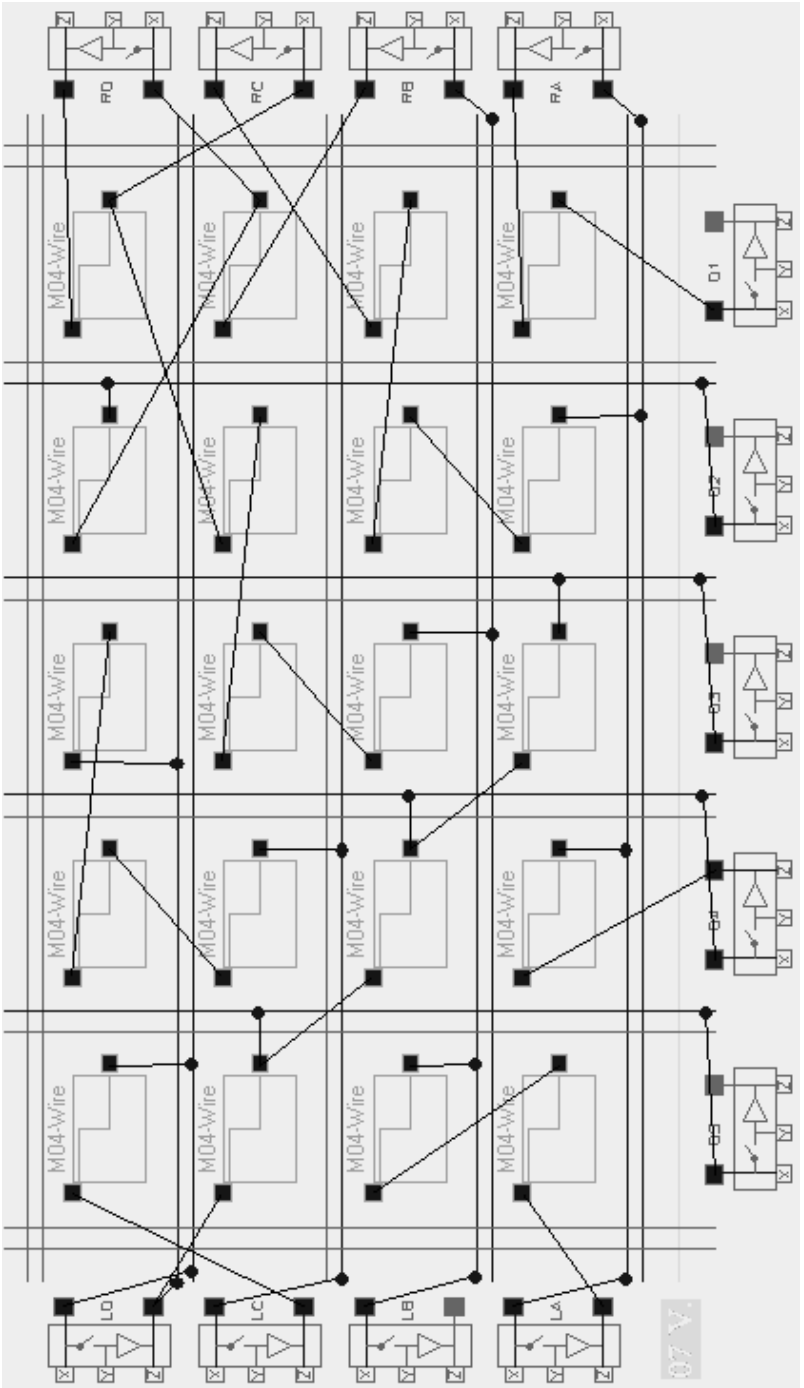
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

4ª CONFIGURAÇÃO DE TESTE



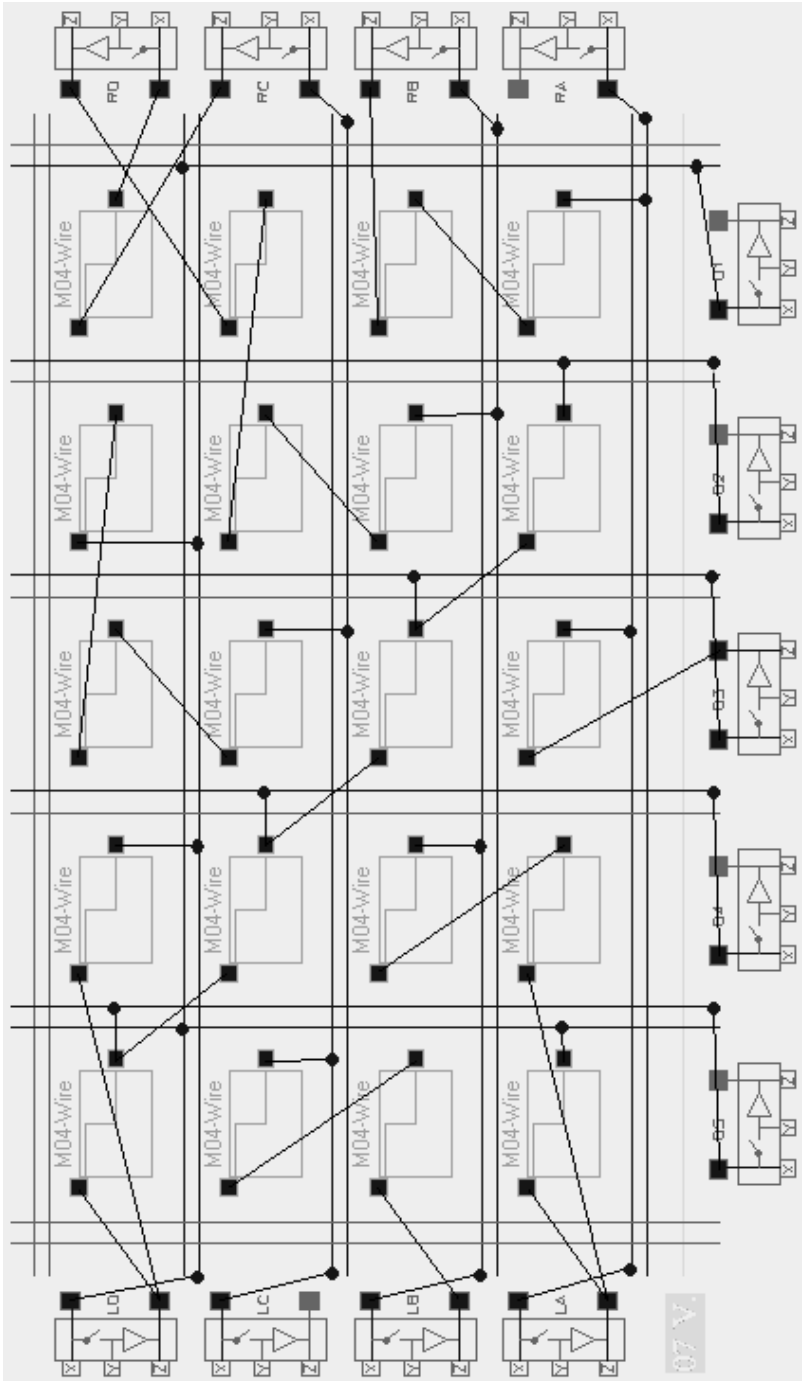
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

5ª CONFIGURAÇÃO DE TESTE



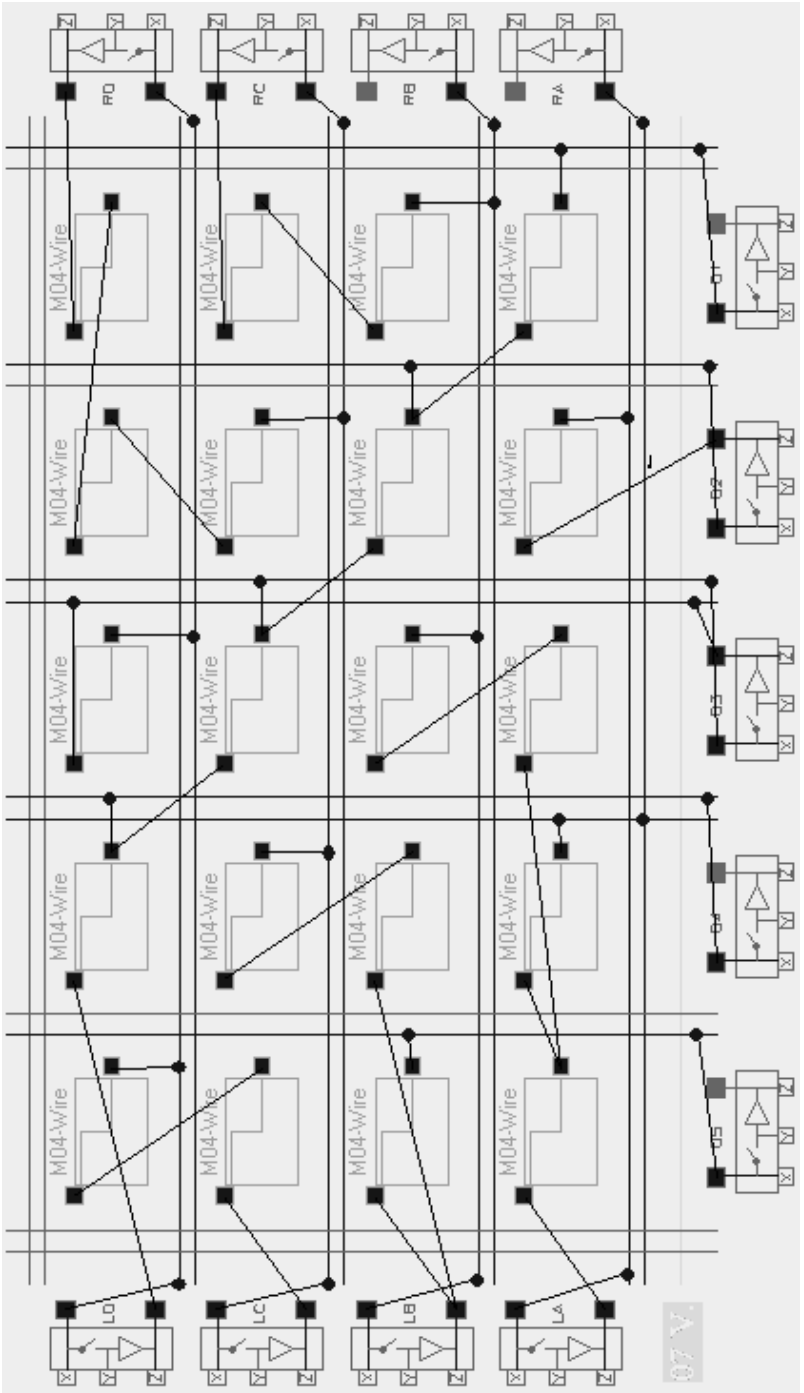
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

6ª CONFIGURAÇÃO DE TESTE



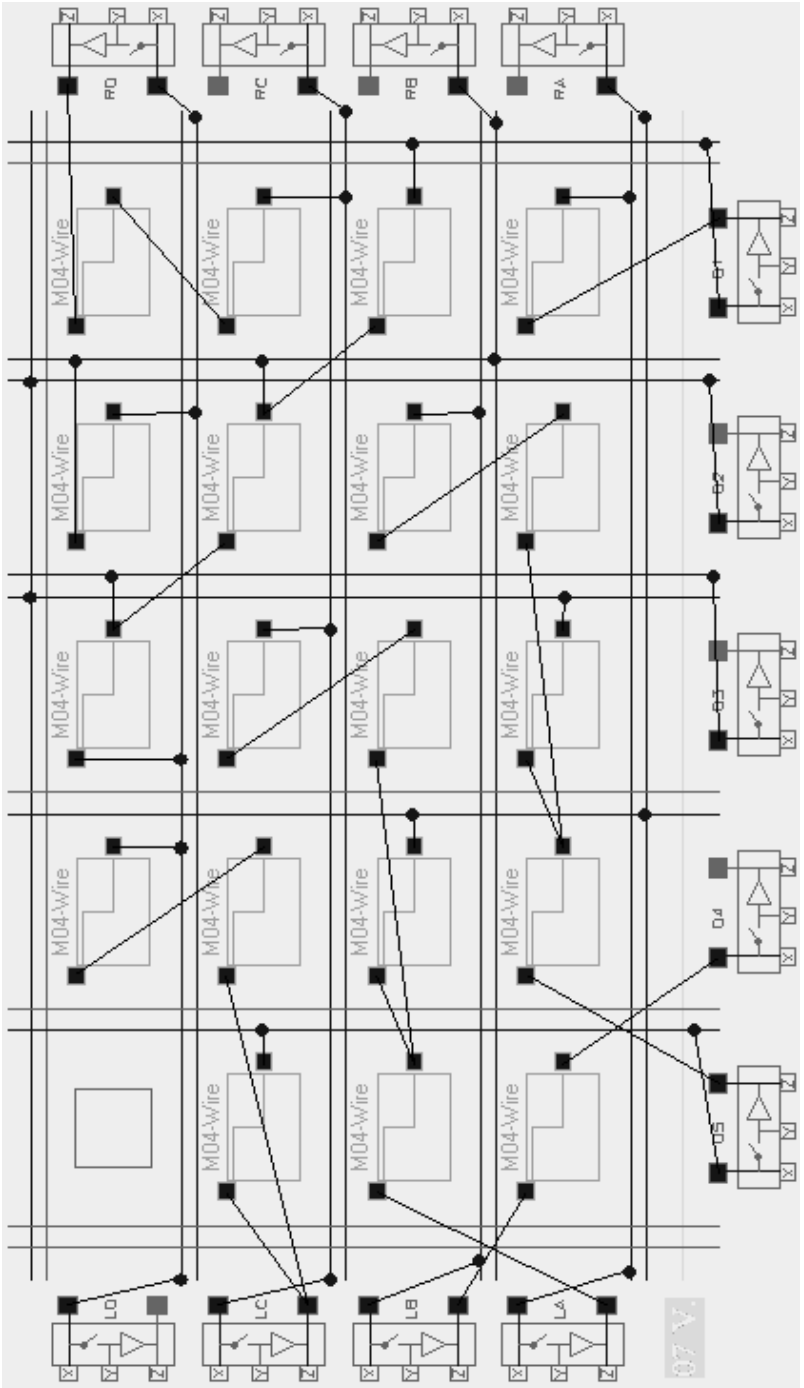
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

7ª CONFIGURAÇÃO DE TESTE



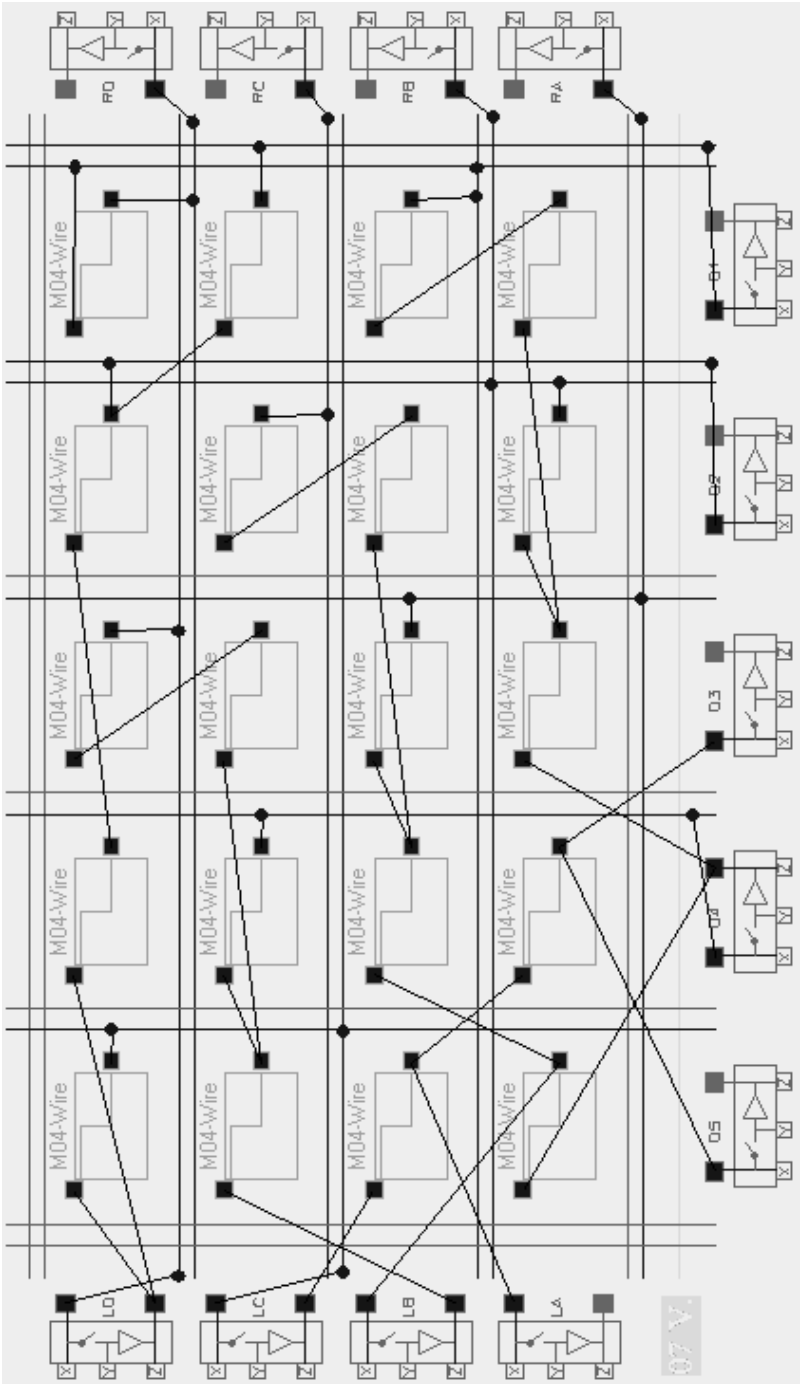
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

8ª CONFIGURAÇÃO DE TESTE



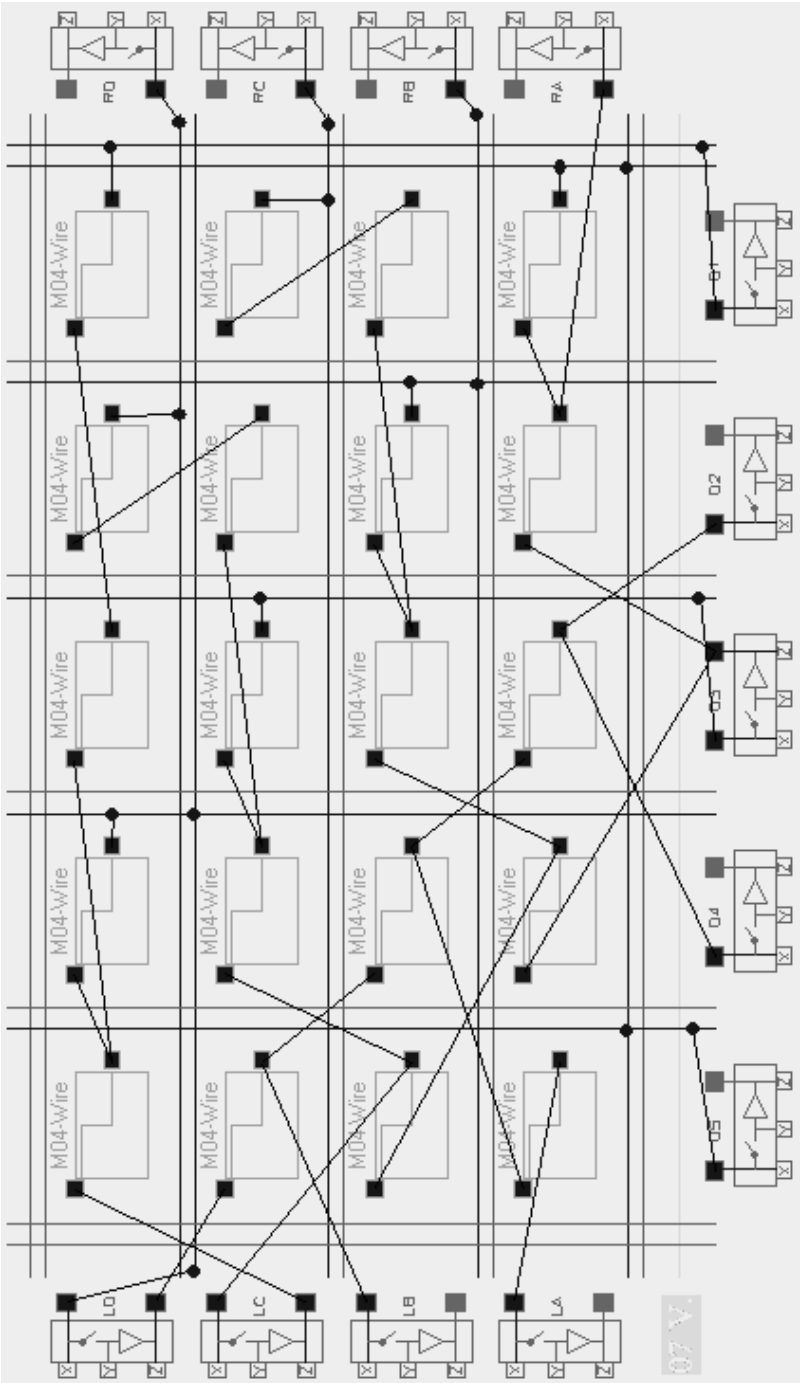
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

9ª CONFIGURAÇÃO DE TESTE



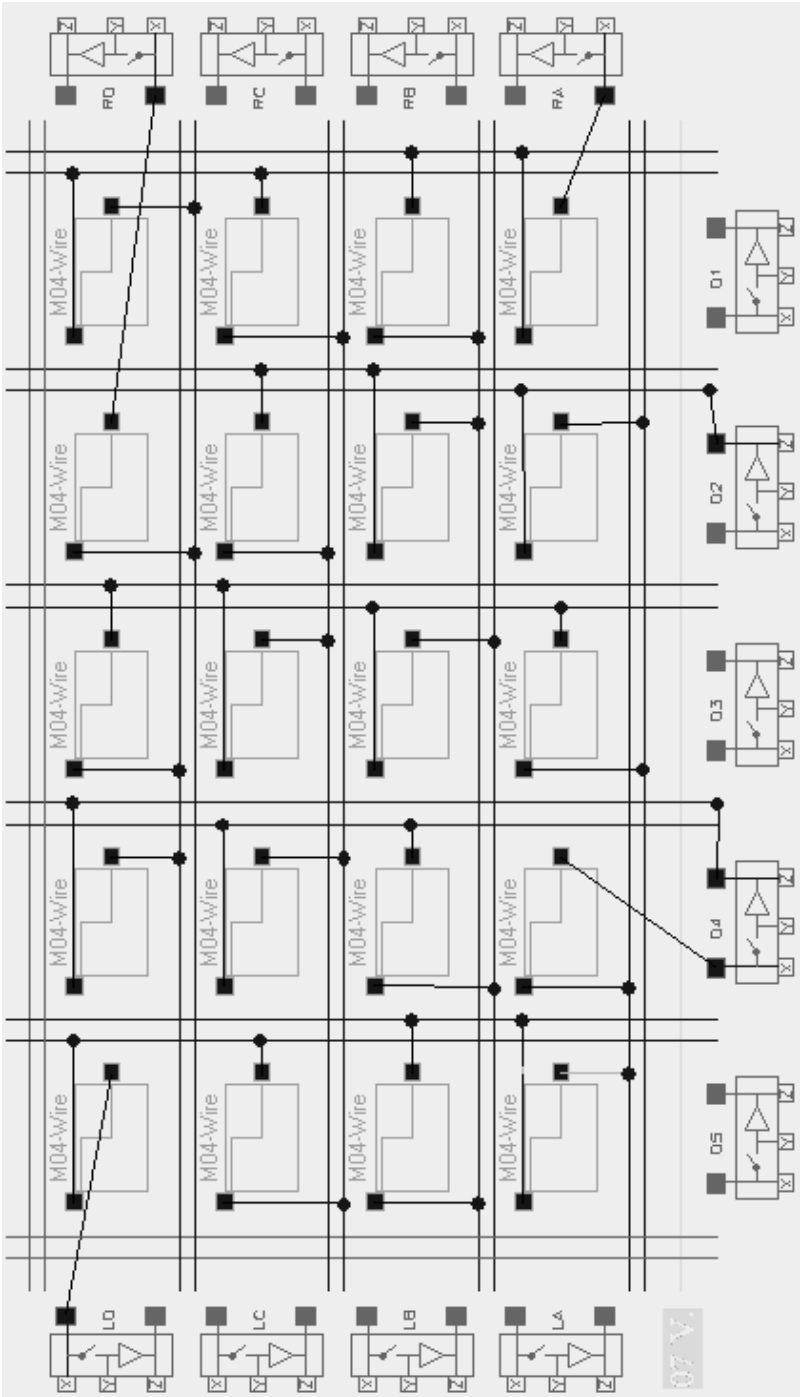
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

10ª CONFIGURAÇÃO DE TESTE



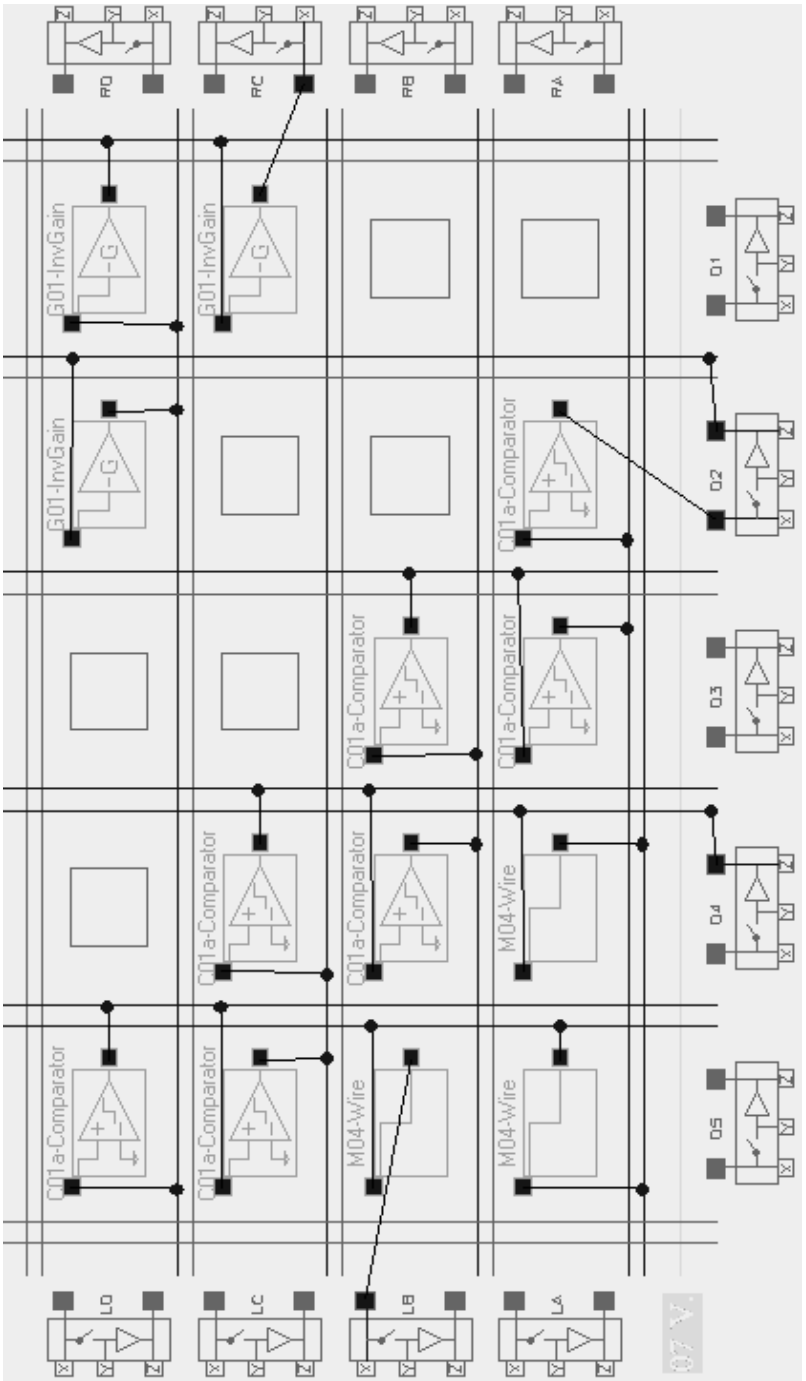
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

11ª CONFIGURAÇÃO DE TESTE



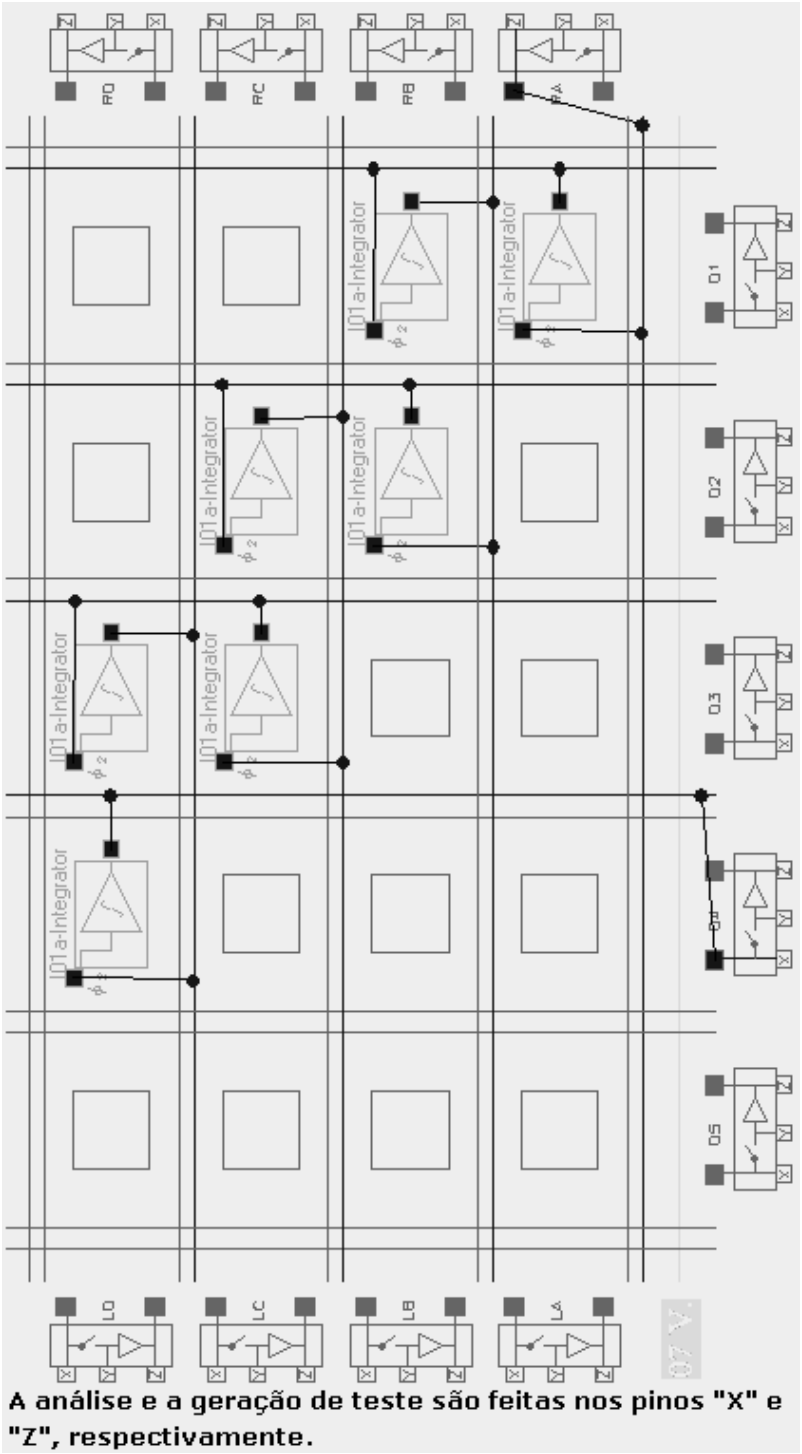
A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

12ª CONFIGURAÇÃO DE TESTE



A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

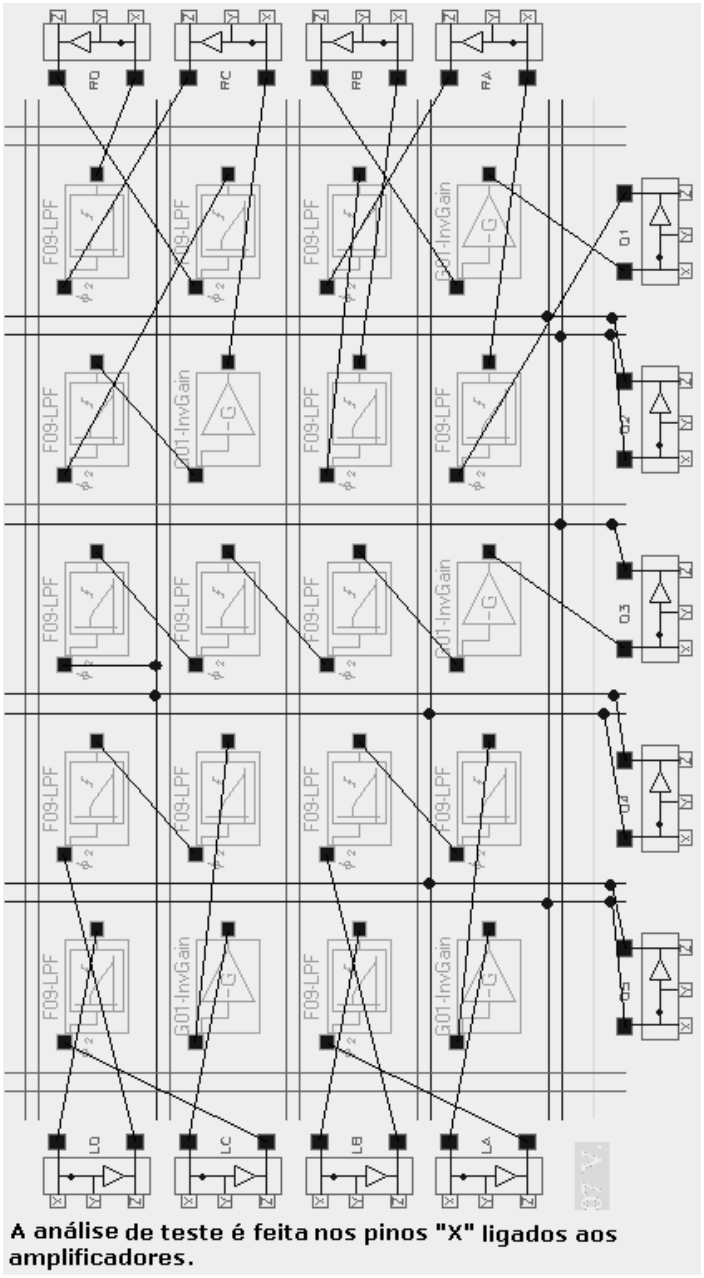
13ª CONFIGURAÇÃO DE TESTE



A análise e a geração de teste são feitas nos pinos "X" e "Z", respectivamente.

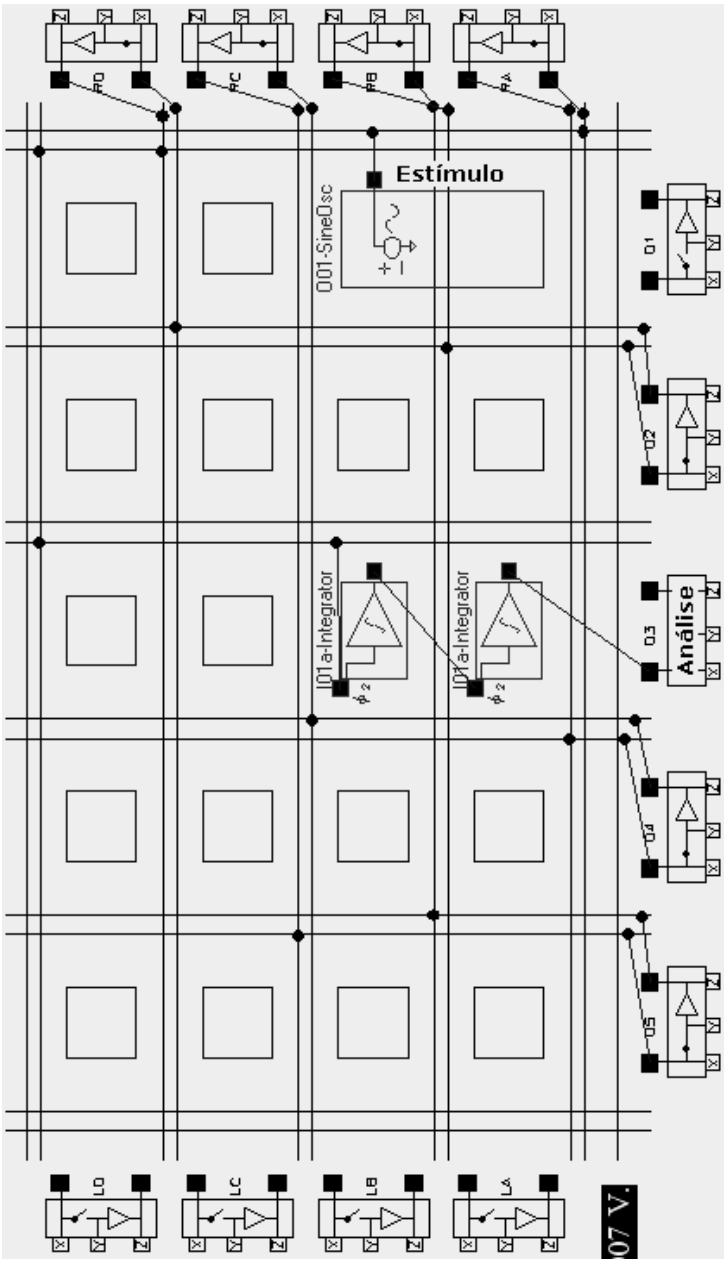
APÊNDICE F CT DOS I/OS – FALHAS STUCK-OPEN E STUCK-ON E PARAMÉTRICAS

1ª CONFIGURAÇÃO DE TESTE

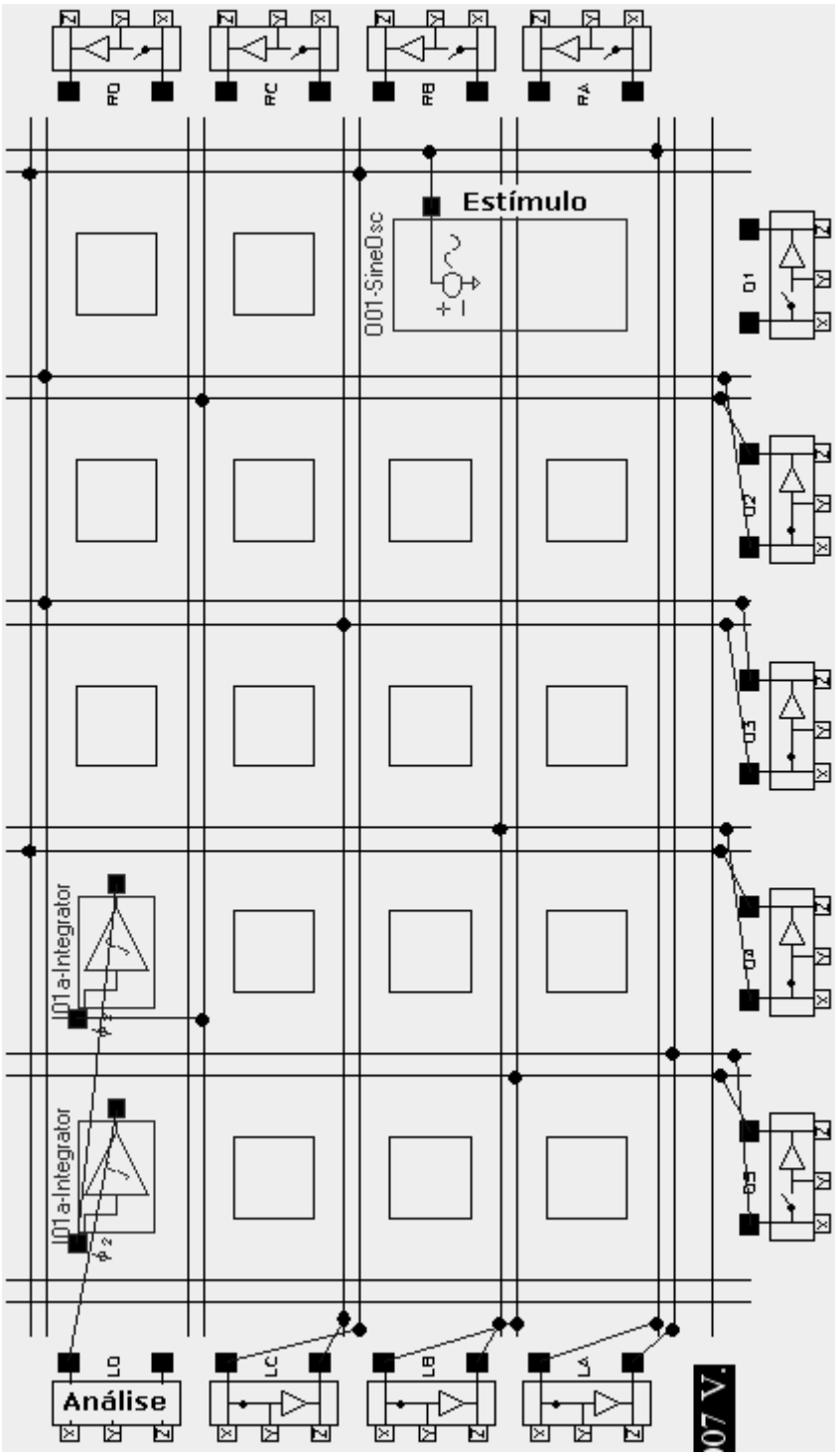


APÊNDICE G CTS DA REDE GLOBAL – FALHAS STUCK-OPEN – CAMINHOS BUFFERIZADOS – BIST

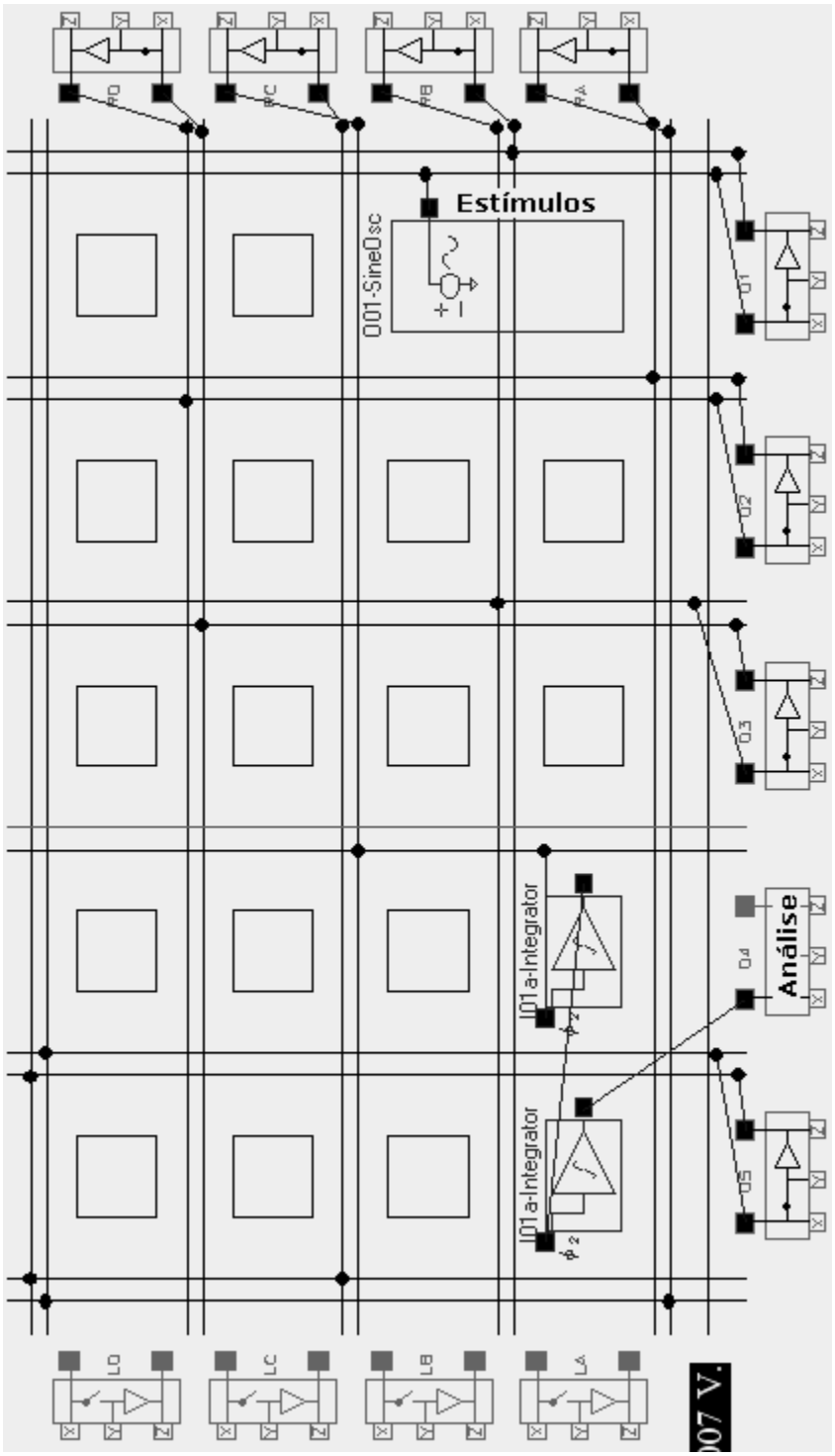
1ª CONFIGURAÇÃO DE TESTE



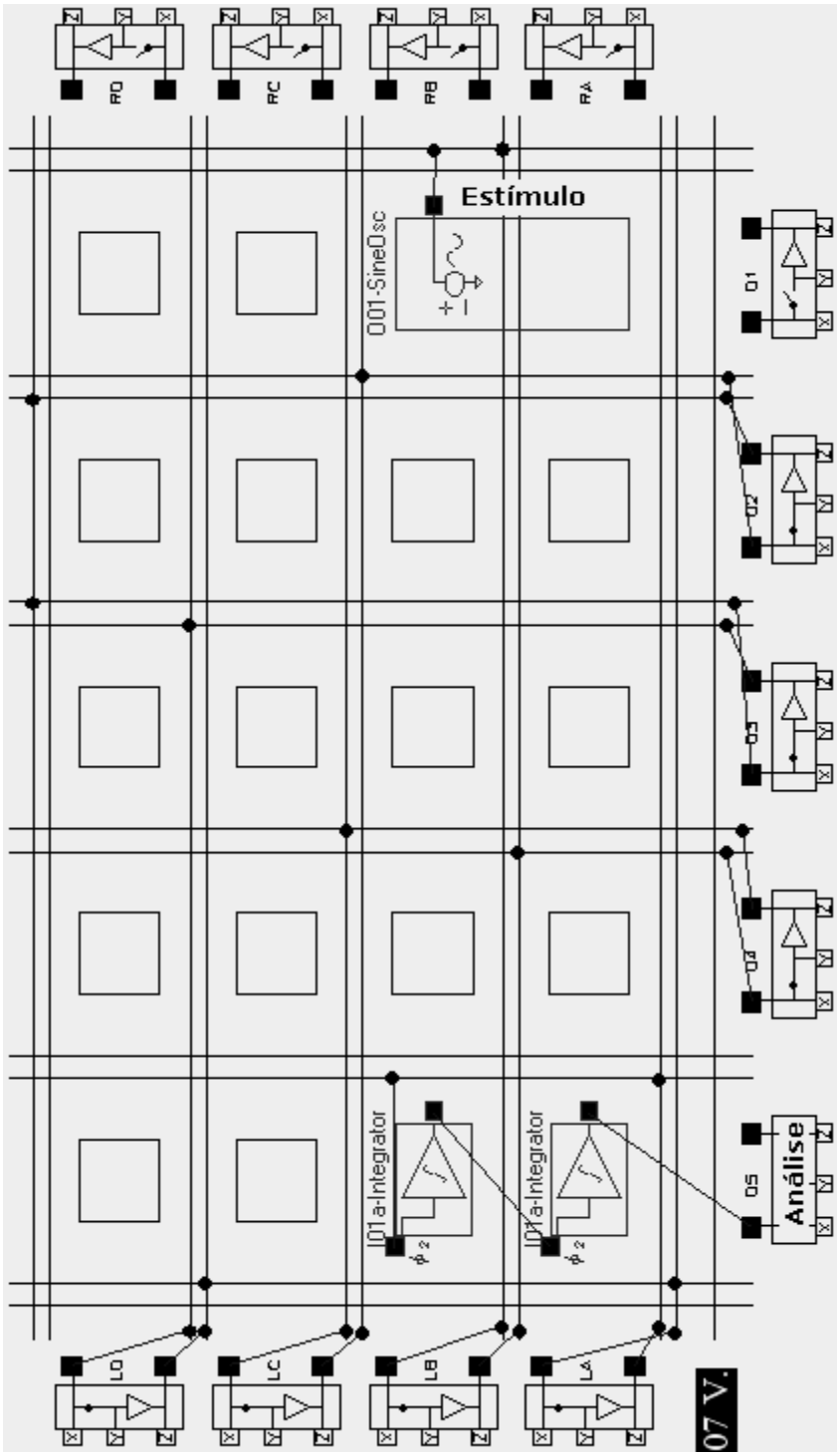
2ª CONFIGURAÇÃO DE TESTE



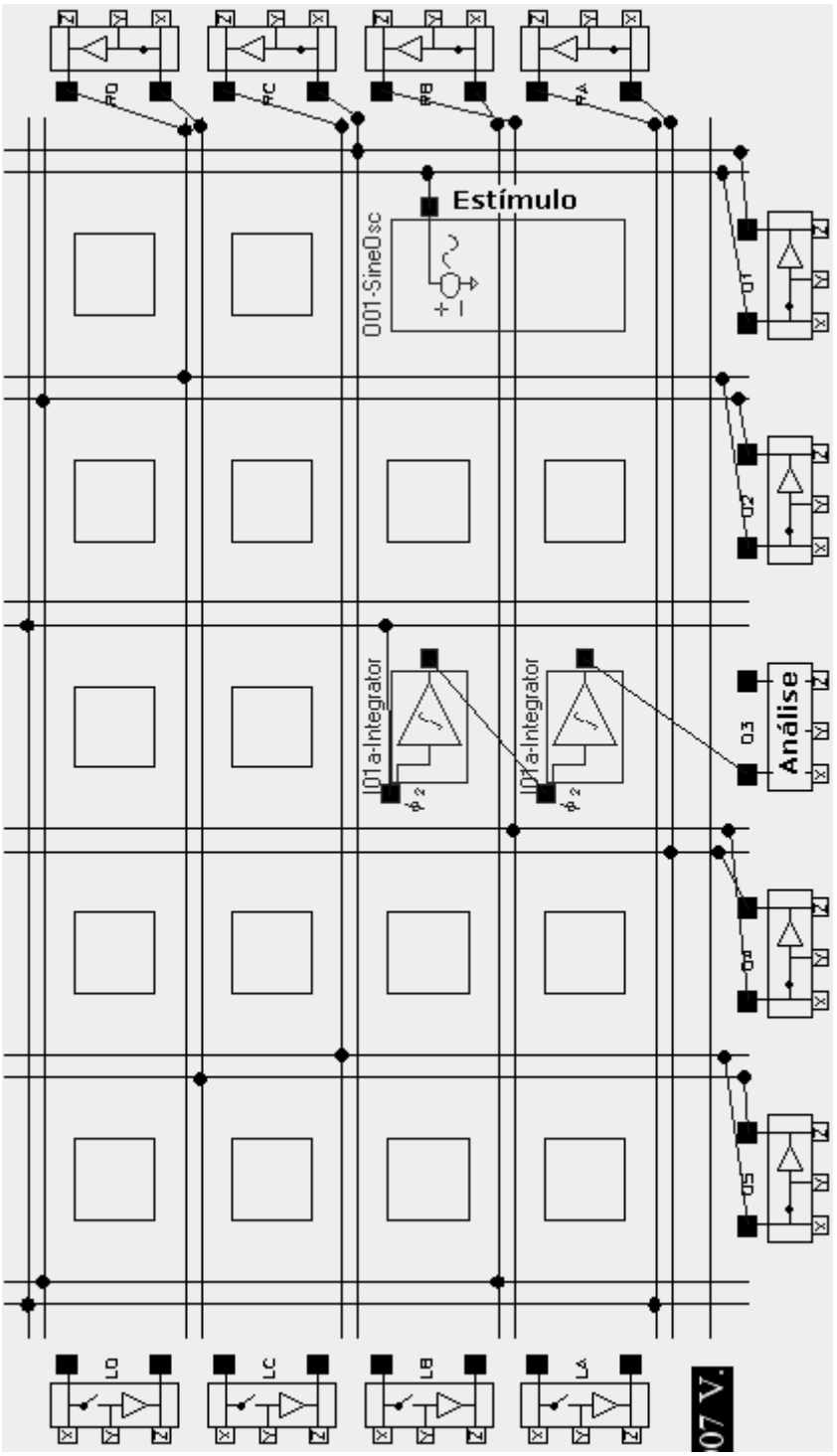
3ª CONFIGURAÇÃO DE TESTE



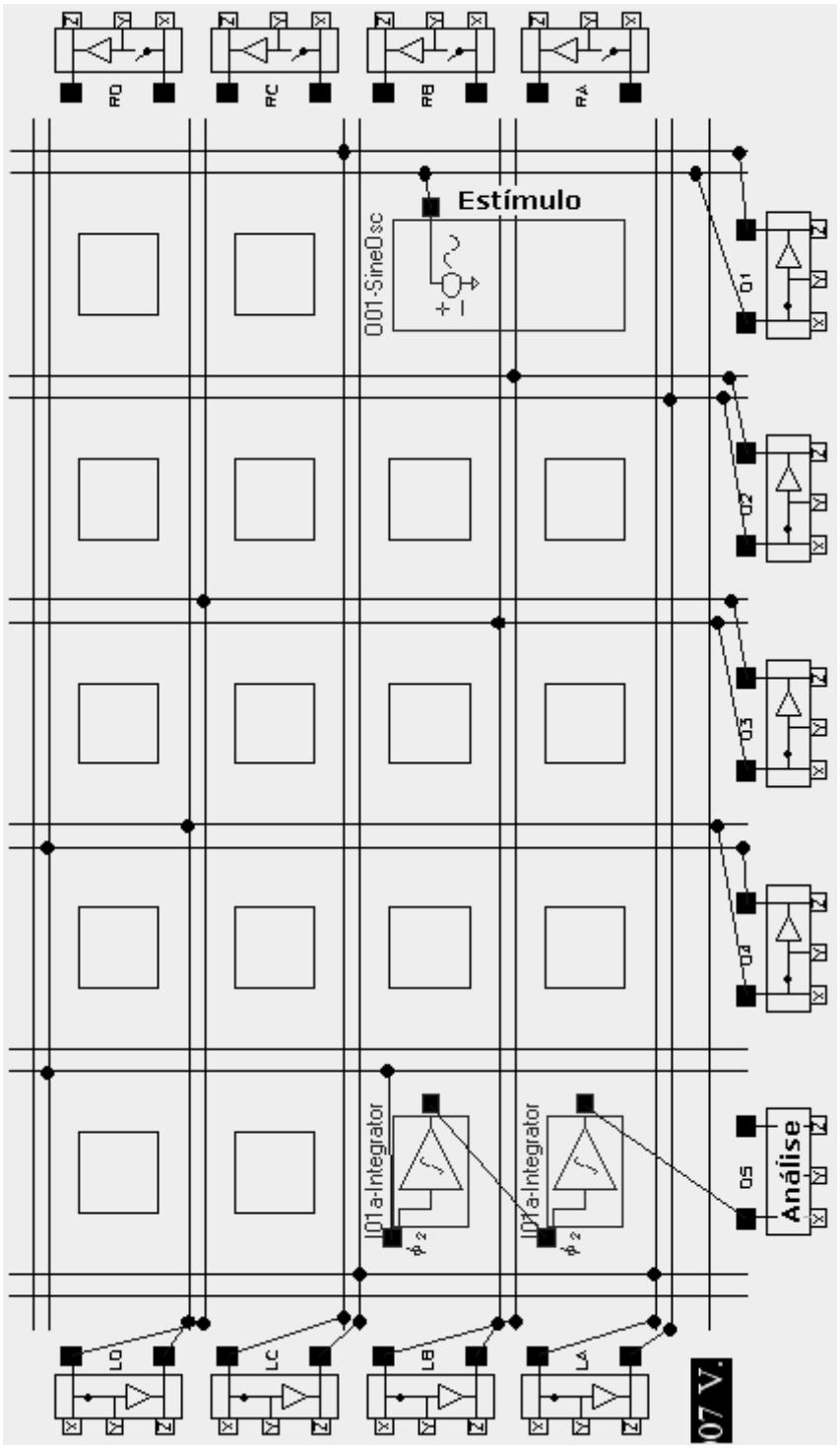
4ª CONFIGURAÇÃO DE TESTE



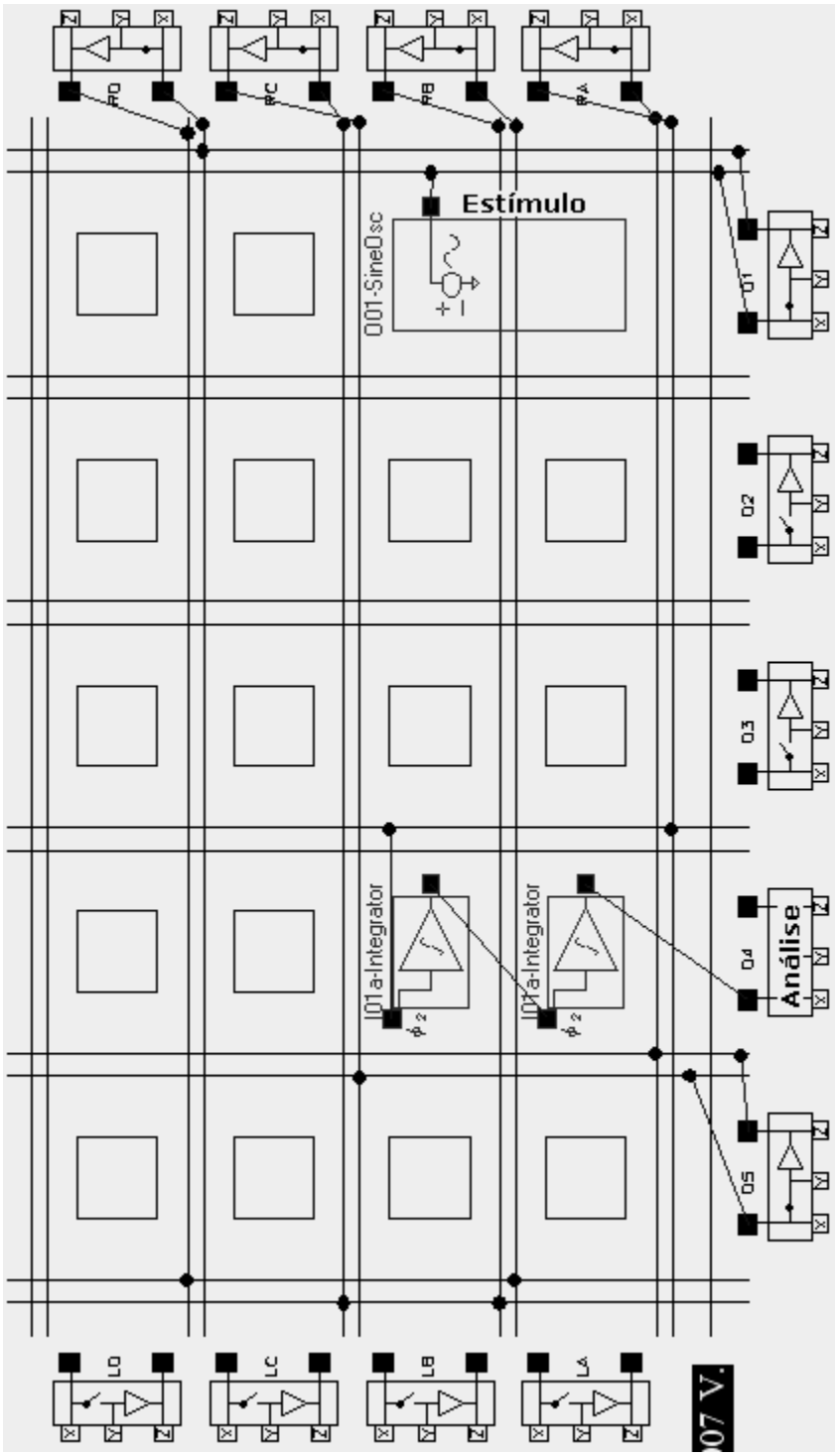
5ª CONFIGURAÇÃO DE TESTE



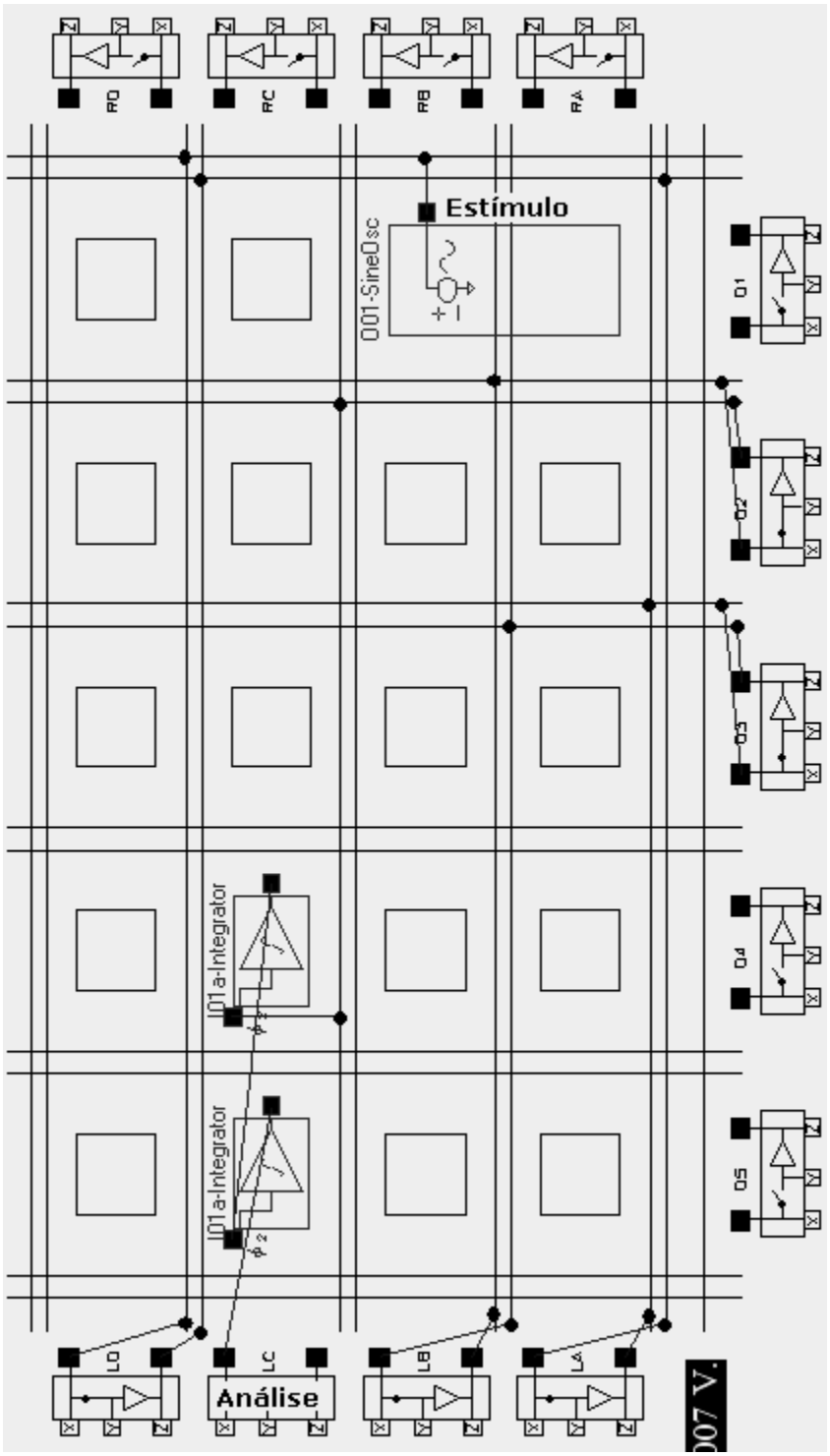
6ª CONFIGURAÇÃO DE TESTE



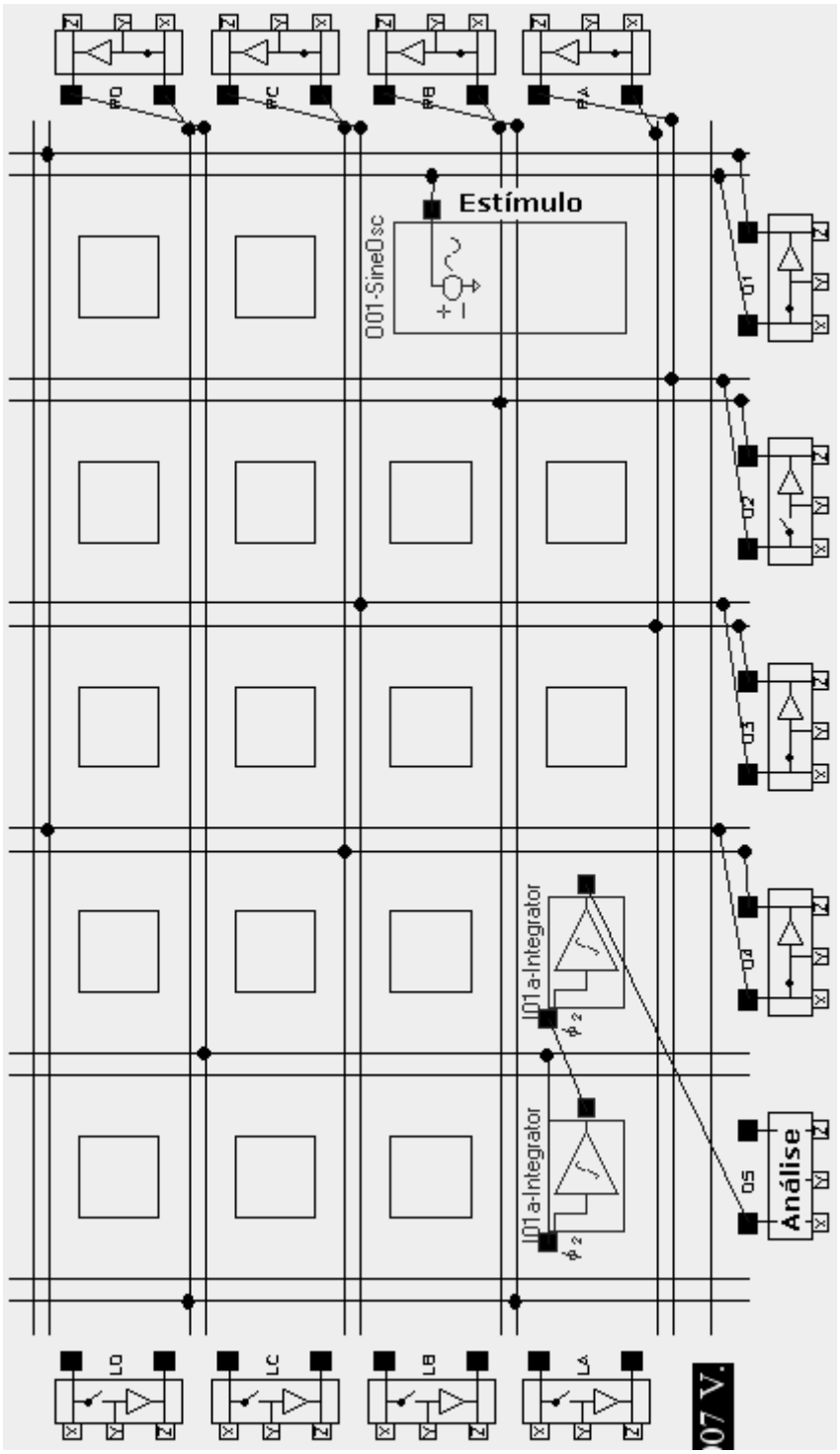
7ª CONFIGURAÇÃO DE TESTE



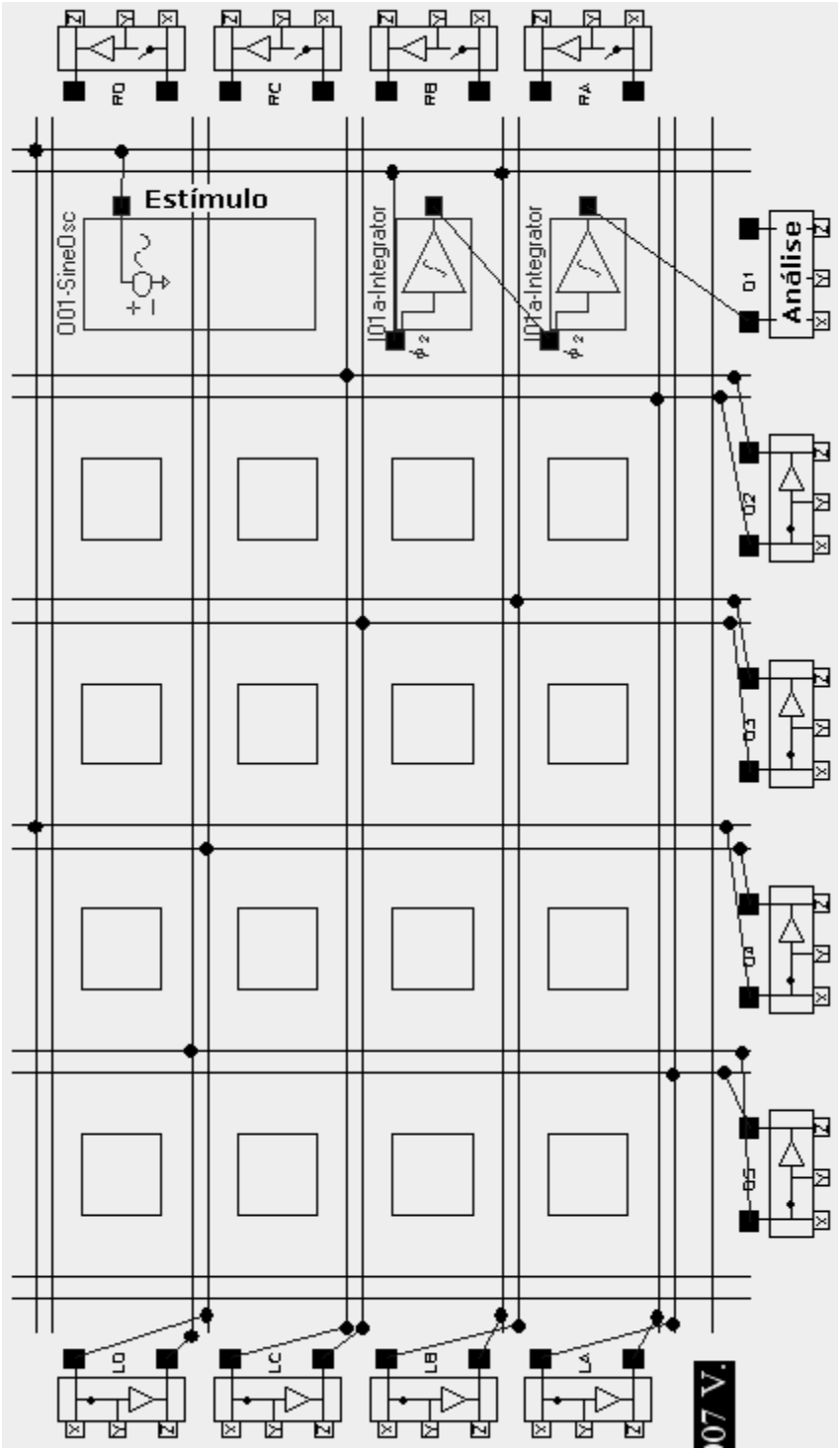
8ª CONFIGURAÇÃO DE TESTE



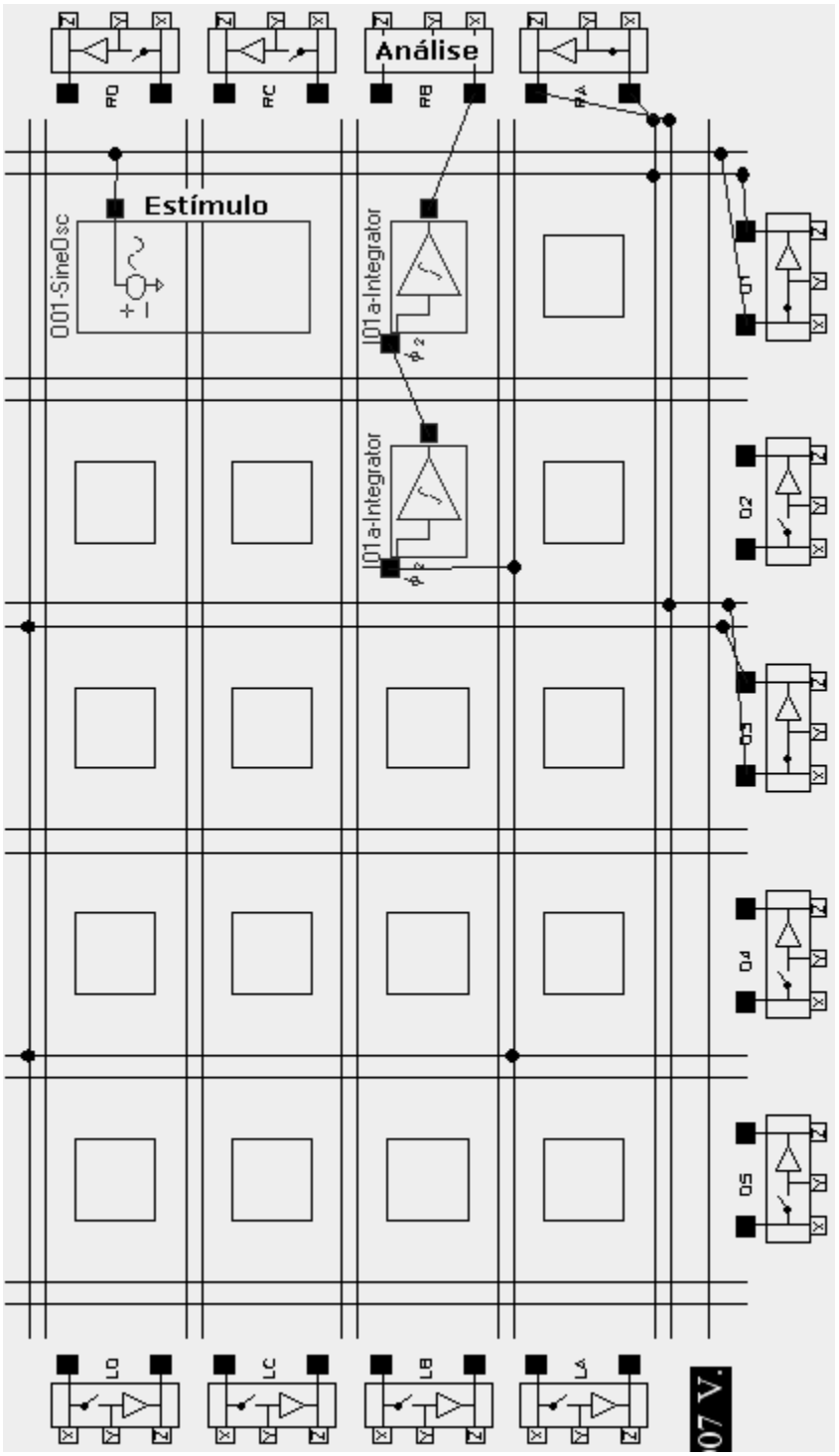
9ª CONFIGURAÇÃO DE TESTE



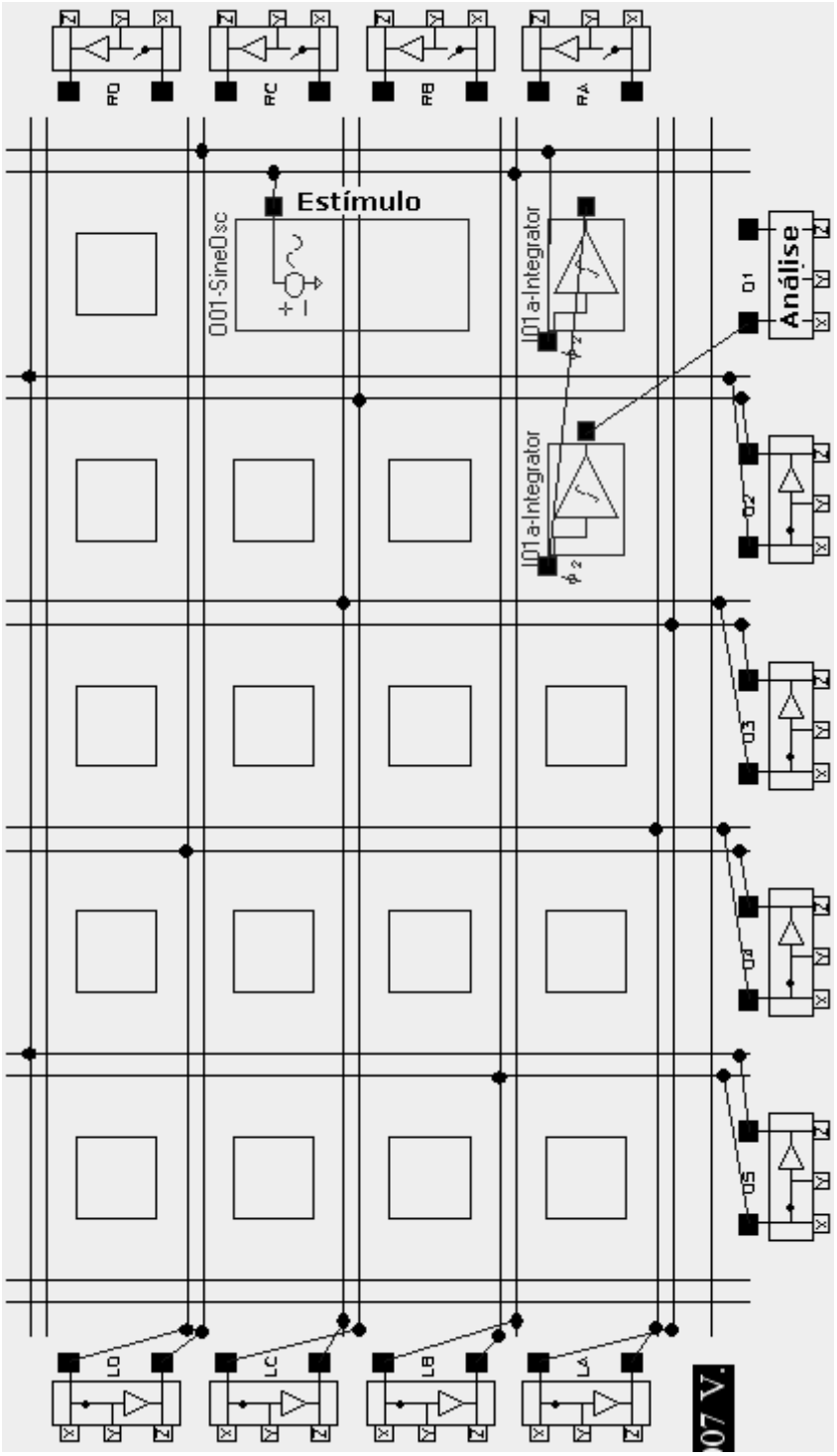
10ª CONFIGURAÇÃO DE TESTE



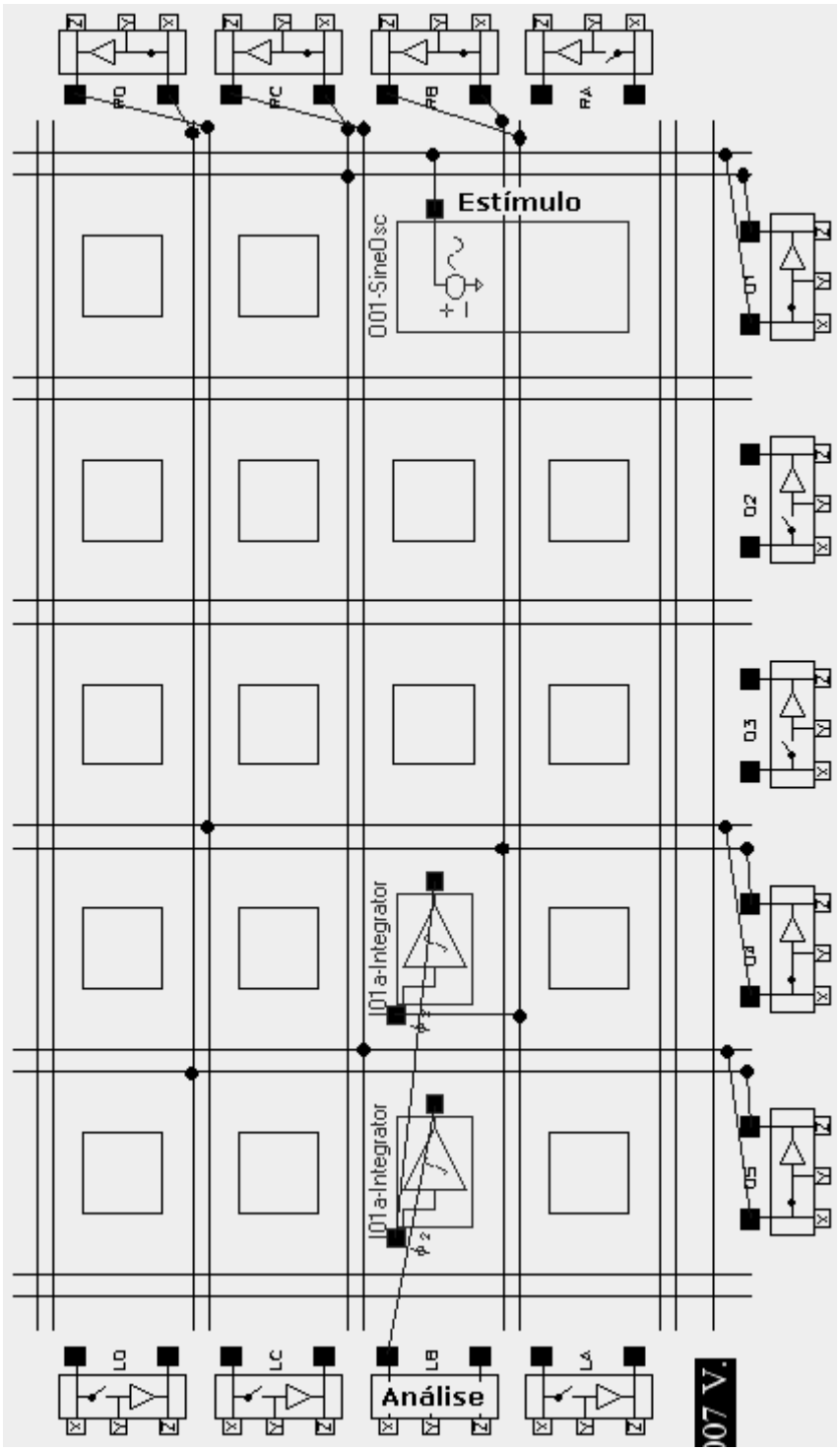
11ª CONFIGURAÇÃO DE TESTE



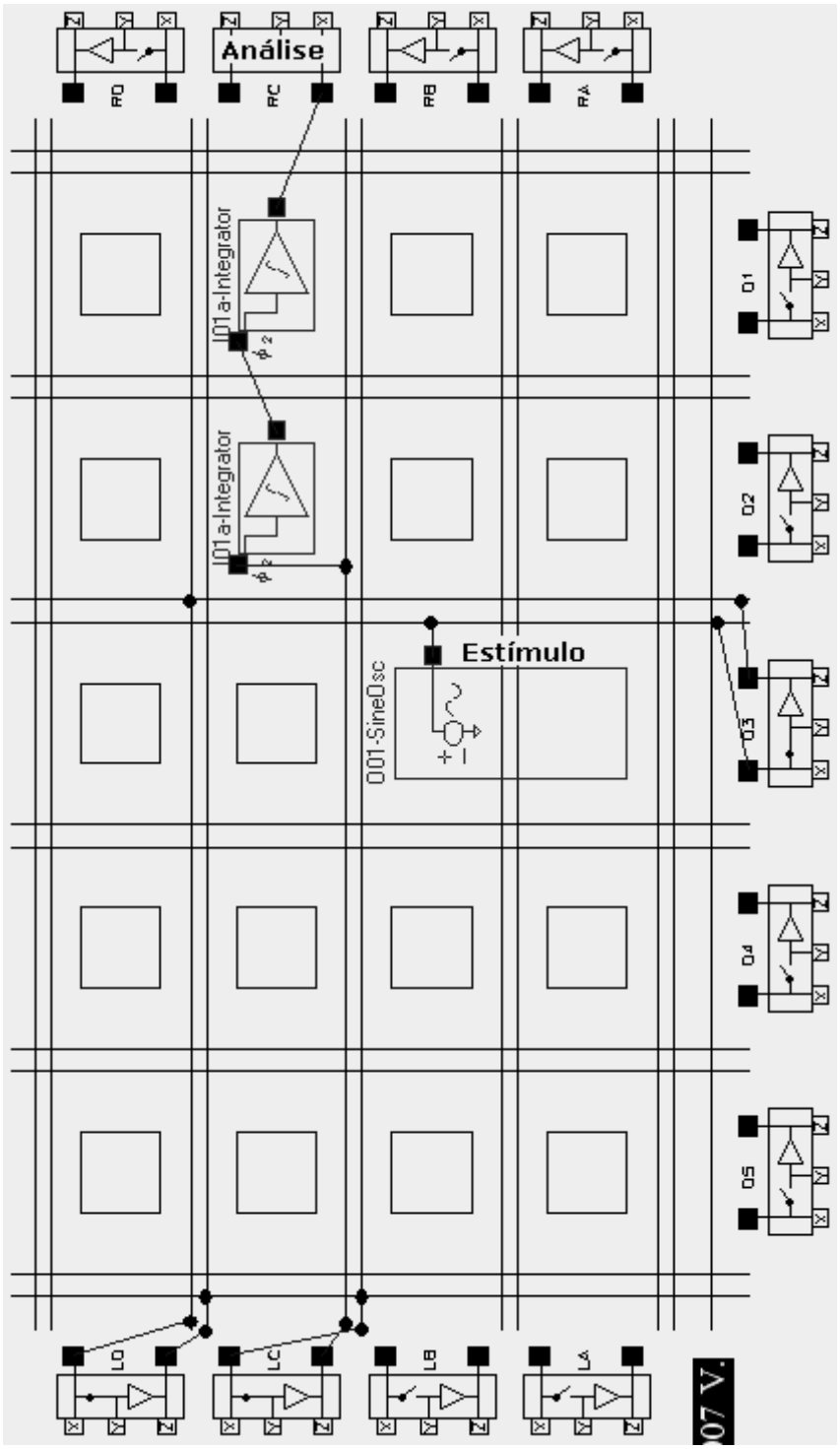
12ª CONFIGURAÇÃO DE TESTE



13ª CONFIGURAÇÃO DE TESTE

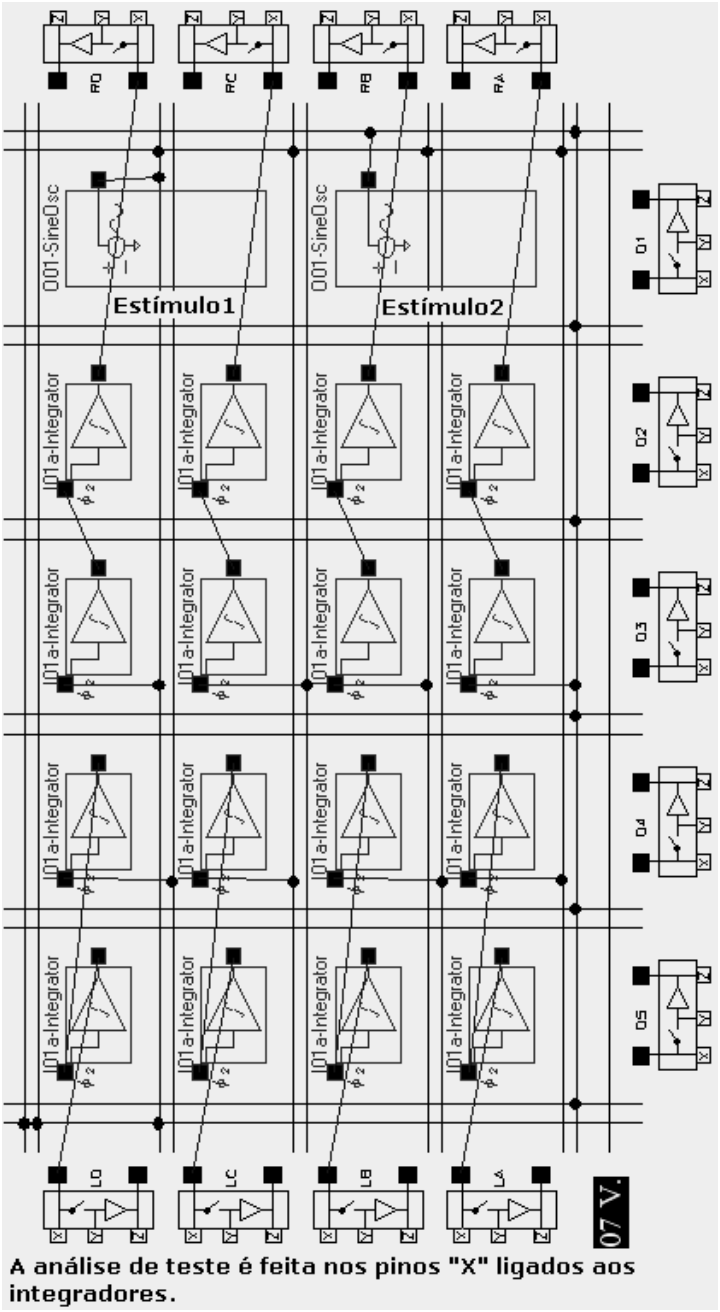


14ª CONFIGURAÇÃO DE TESTE

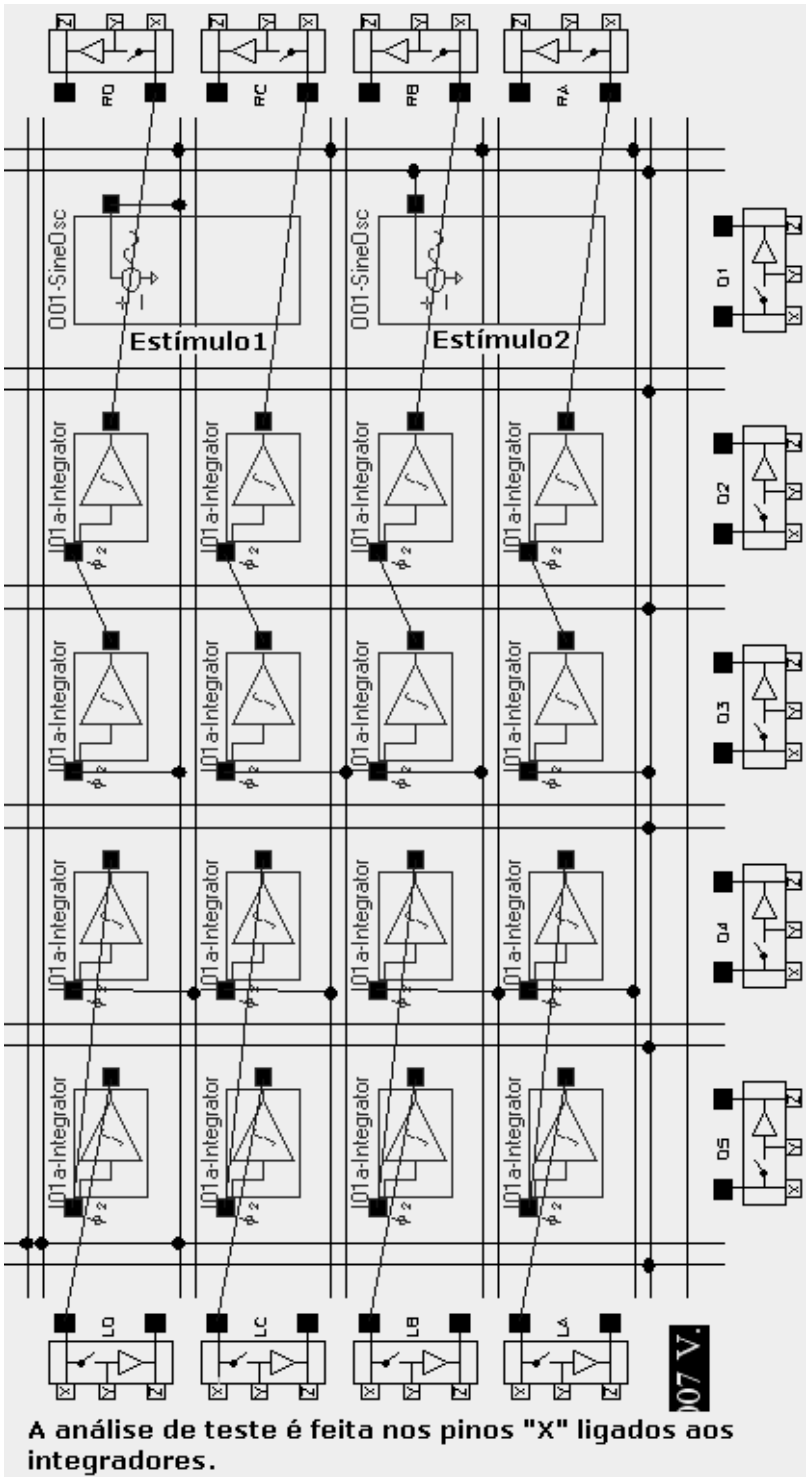


APÊNDICE H CTS DA REDE GLOBAL – FALHAS STUCK-ON – MULTI-ONDA – BIST

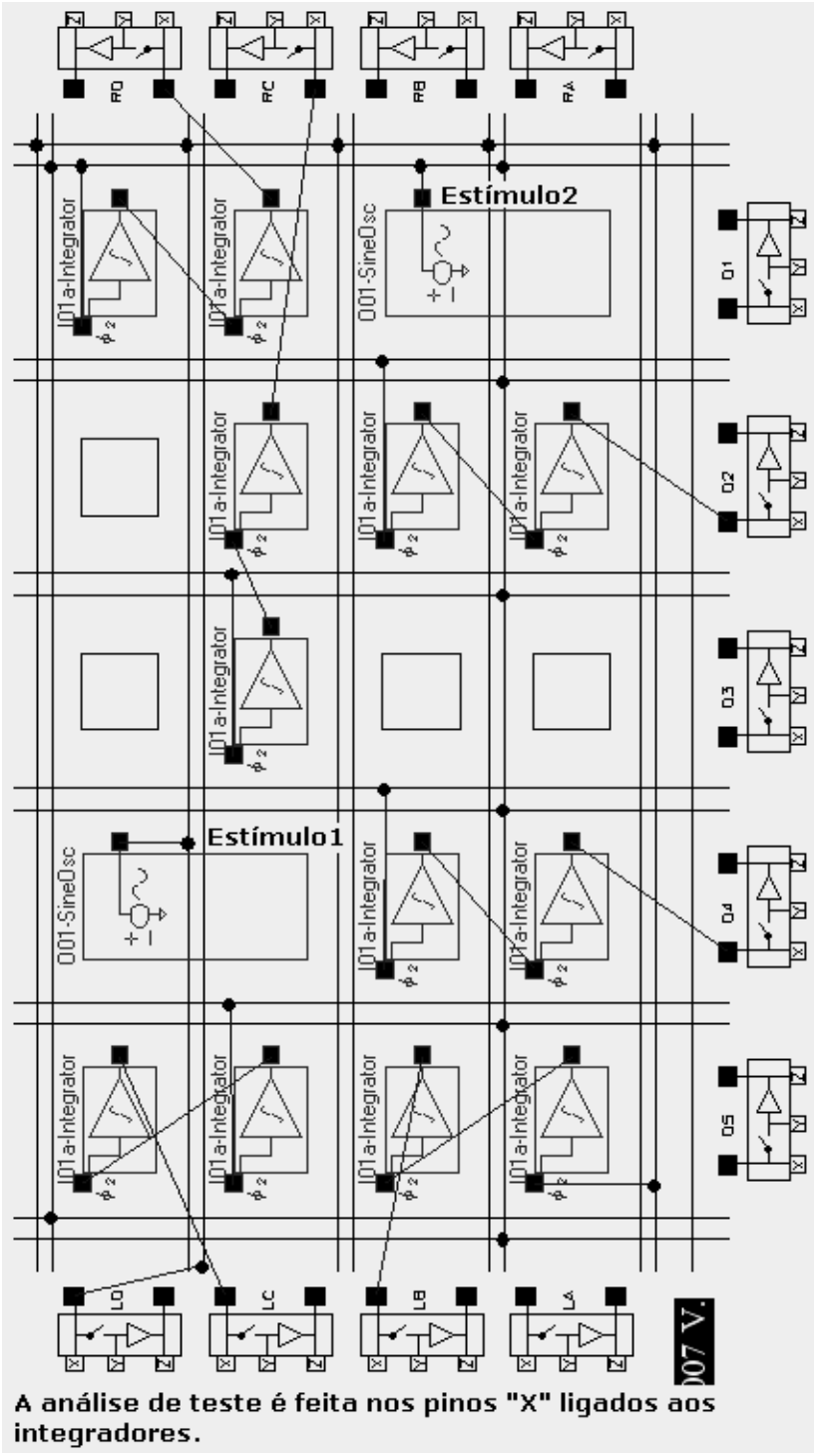
1ª CONFIGURAÇÃO DE TESTE



2ª CONFIGURAÇÃO DE TESTE

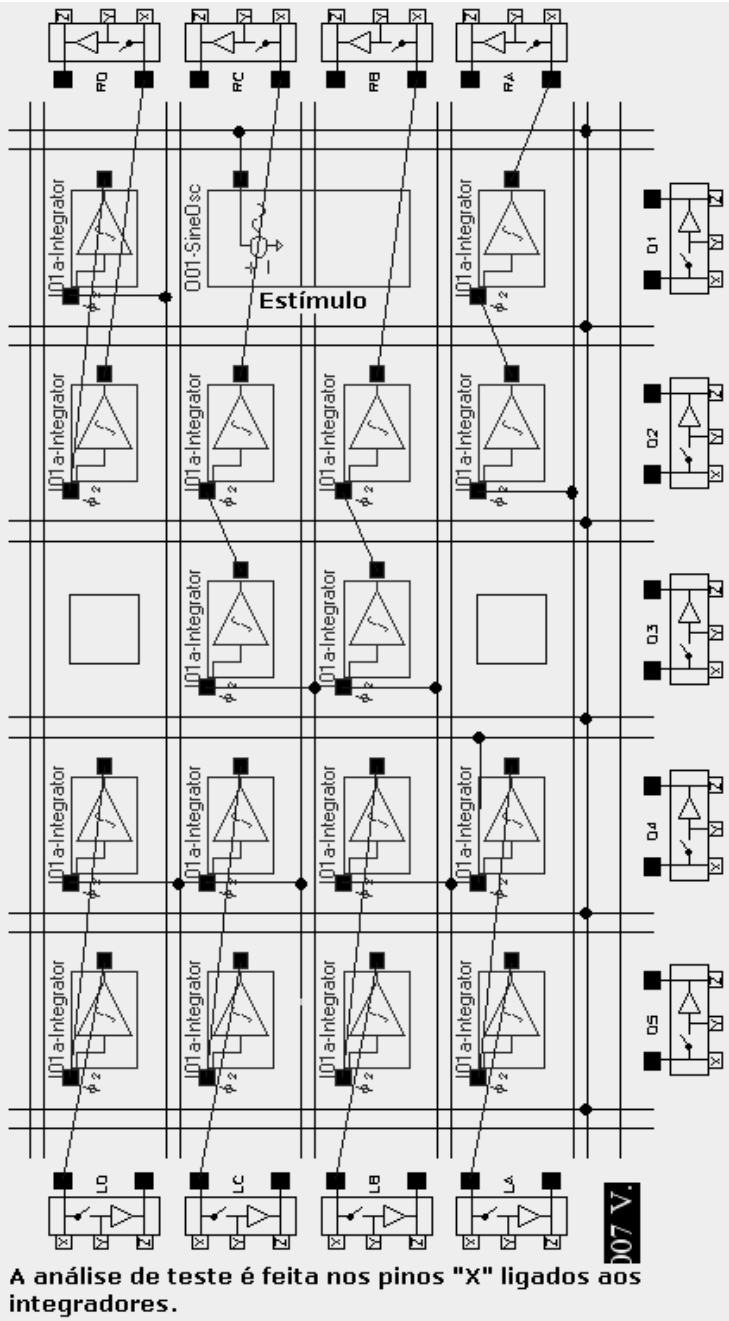


3ª CONFIGURAÇÃO DE TESTE

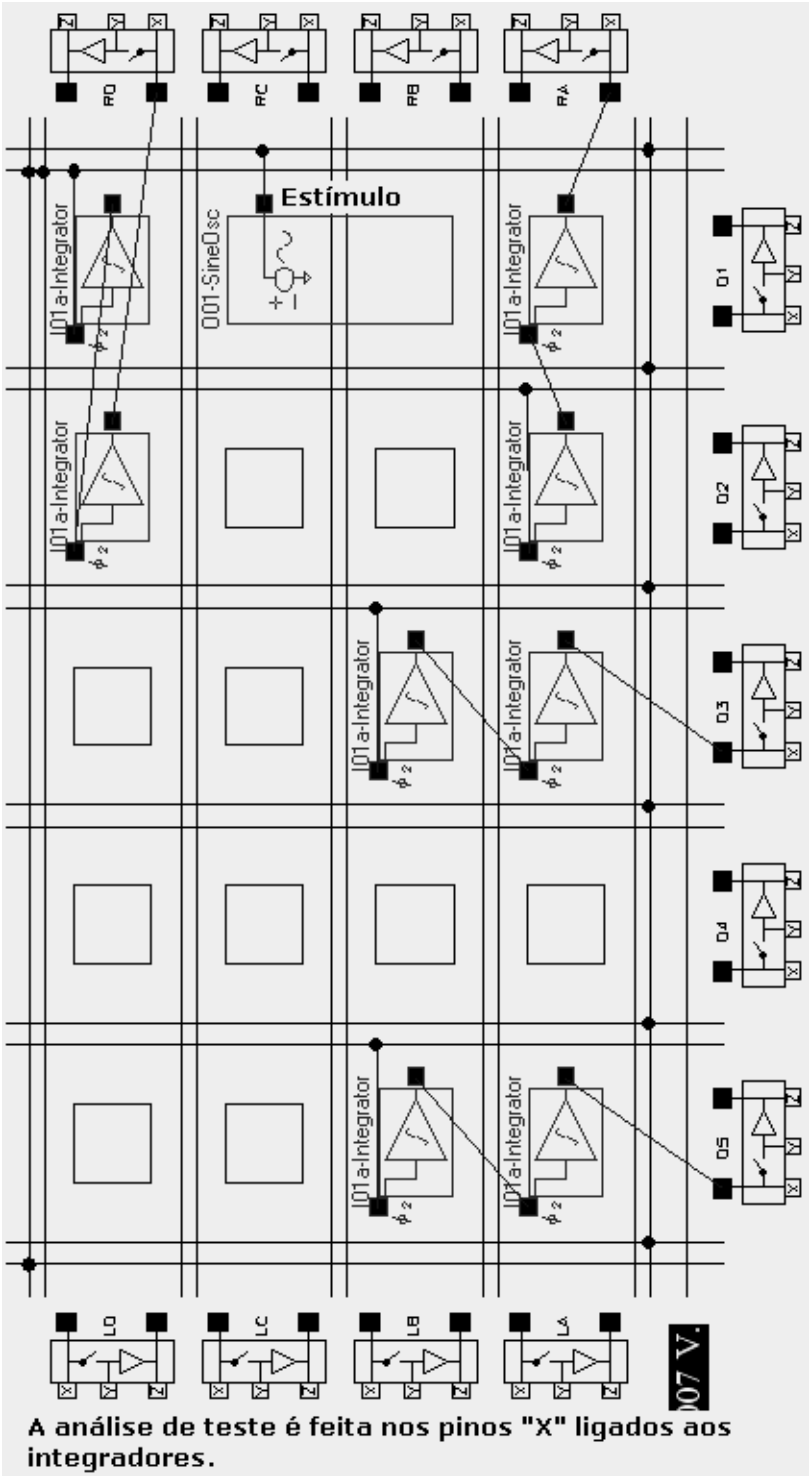


APÊNDICE I CTS DA REDE GLOBAL – FALHAS STUCK-ON – FONTE ÚNICA – BIST

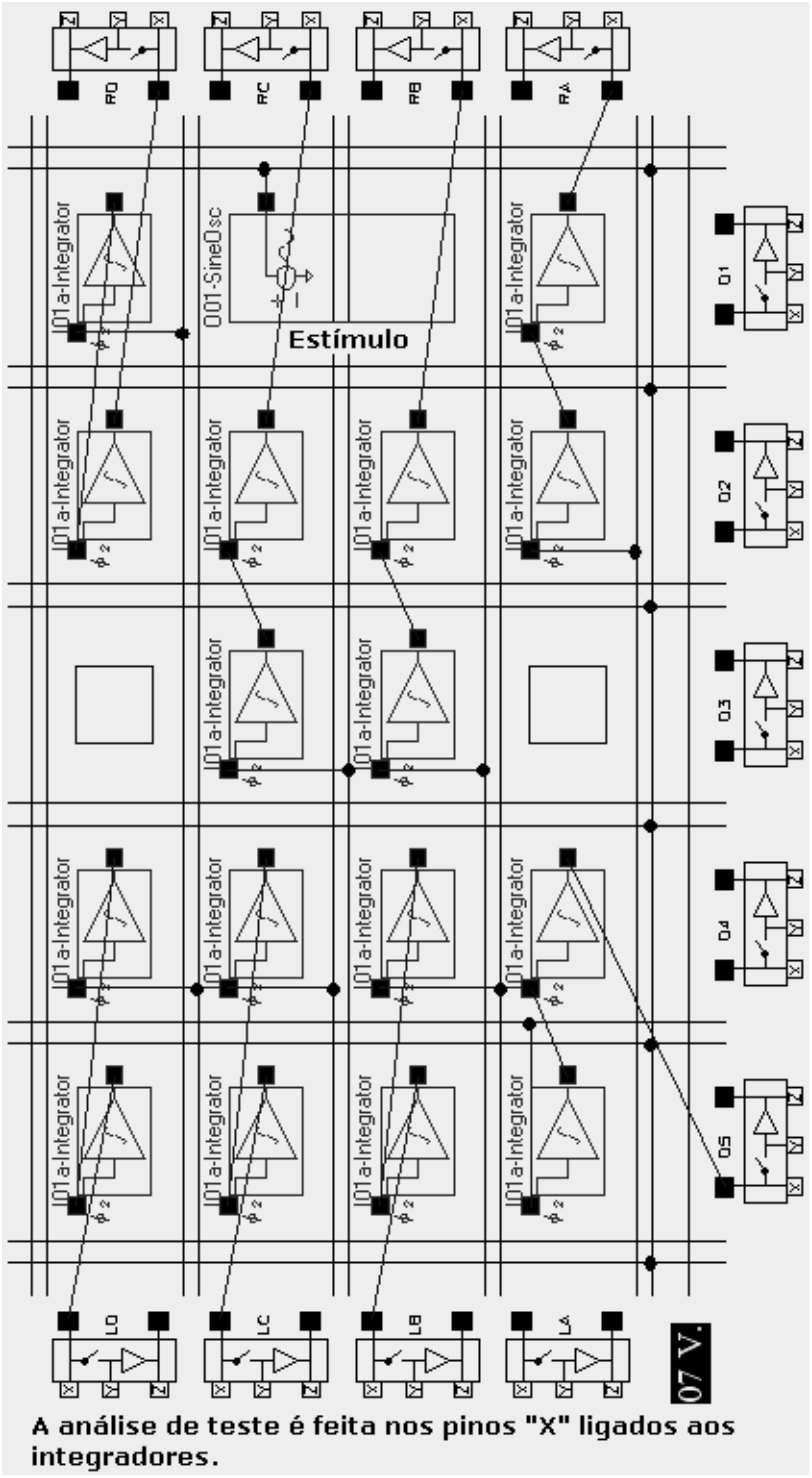
1ª CONFIGURAÇÃO DE TESTE



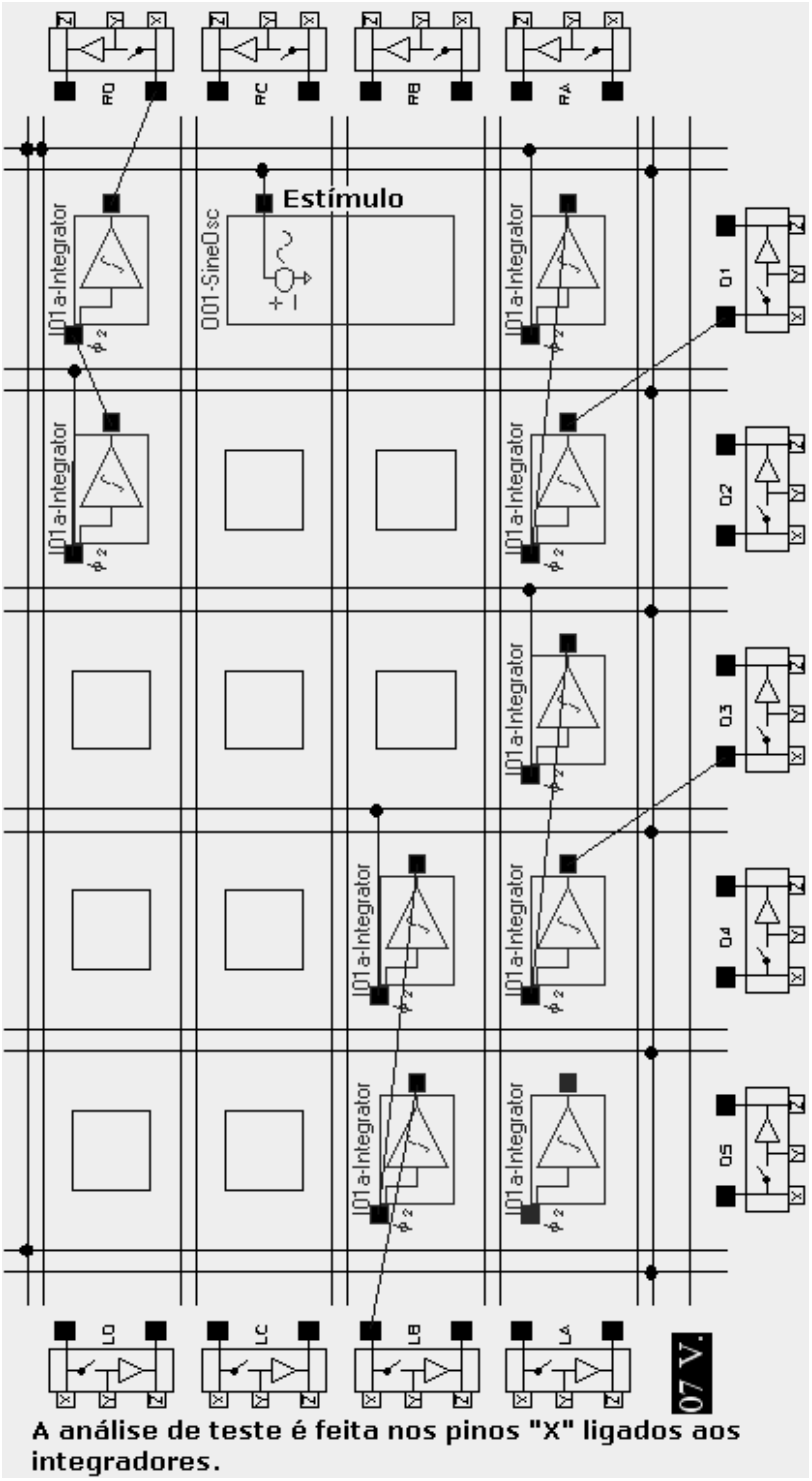
2ª CONFIGURAÇÃO DE TESTE



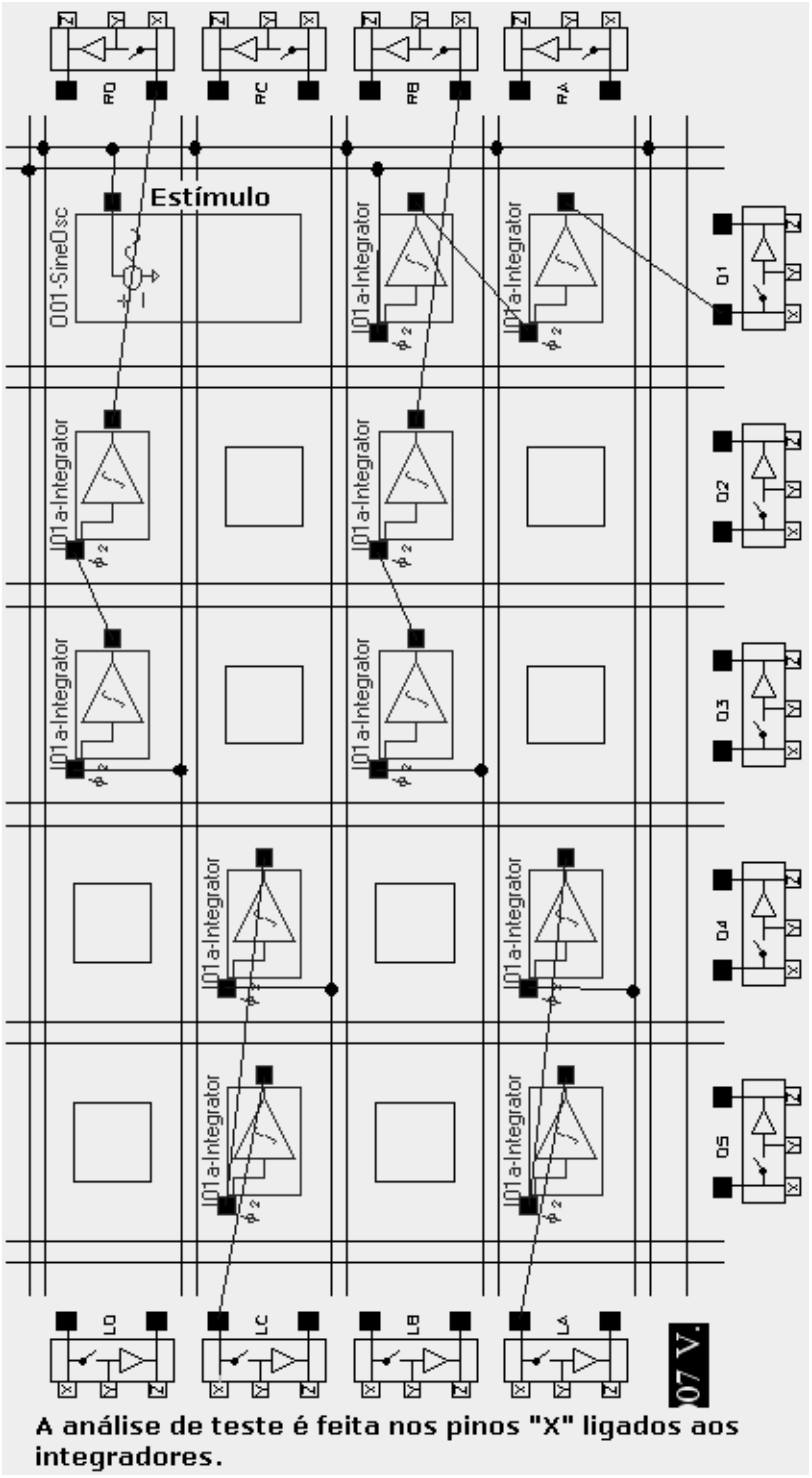
3ª CONFIGURAÇÃO DE TESTE



4ª CONFIGURAÇÃO DE TESTE

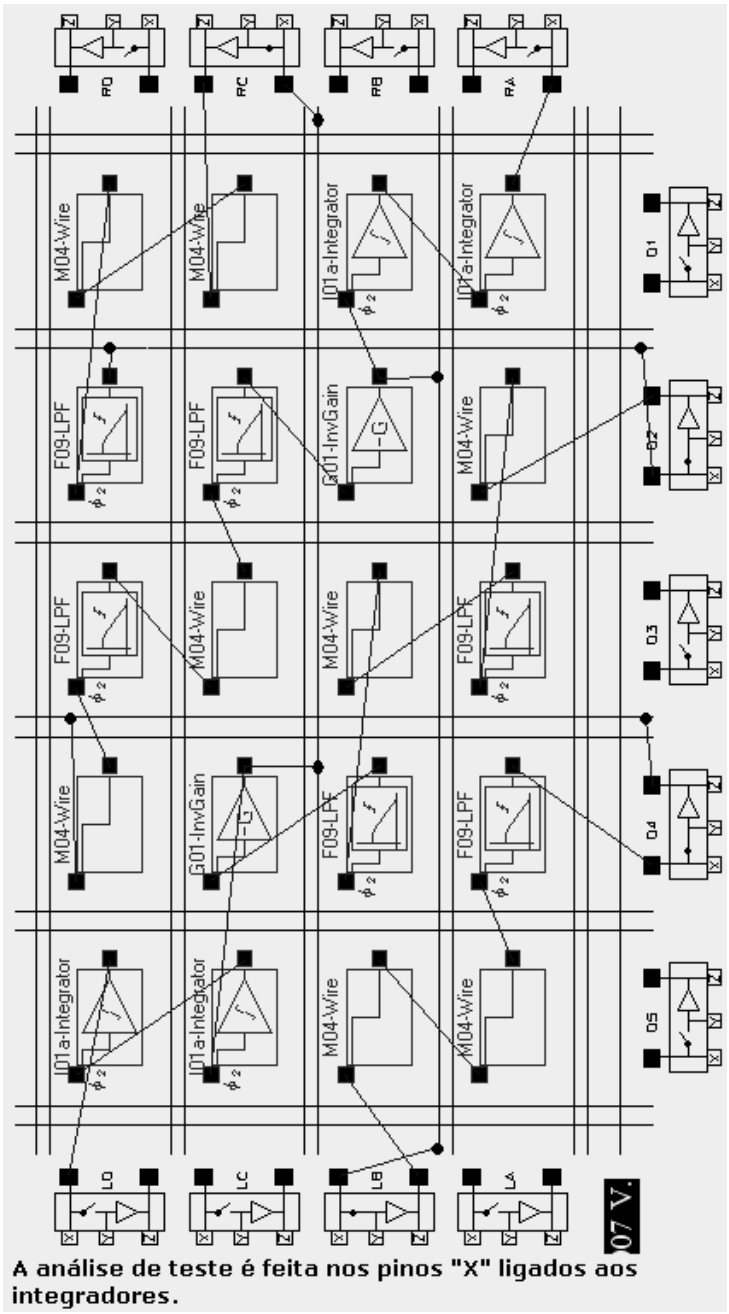


5ª CONFIGURAÇÃO DE TESTE

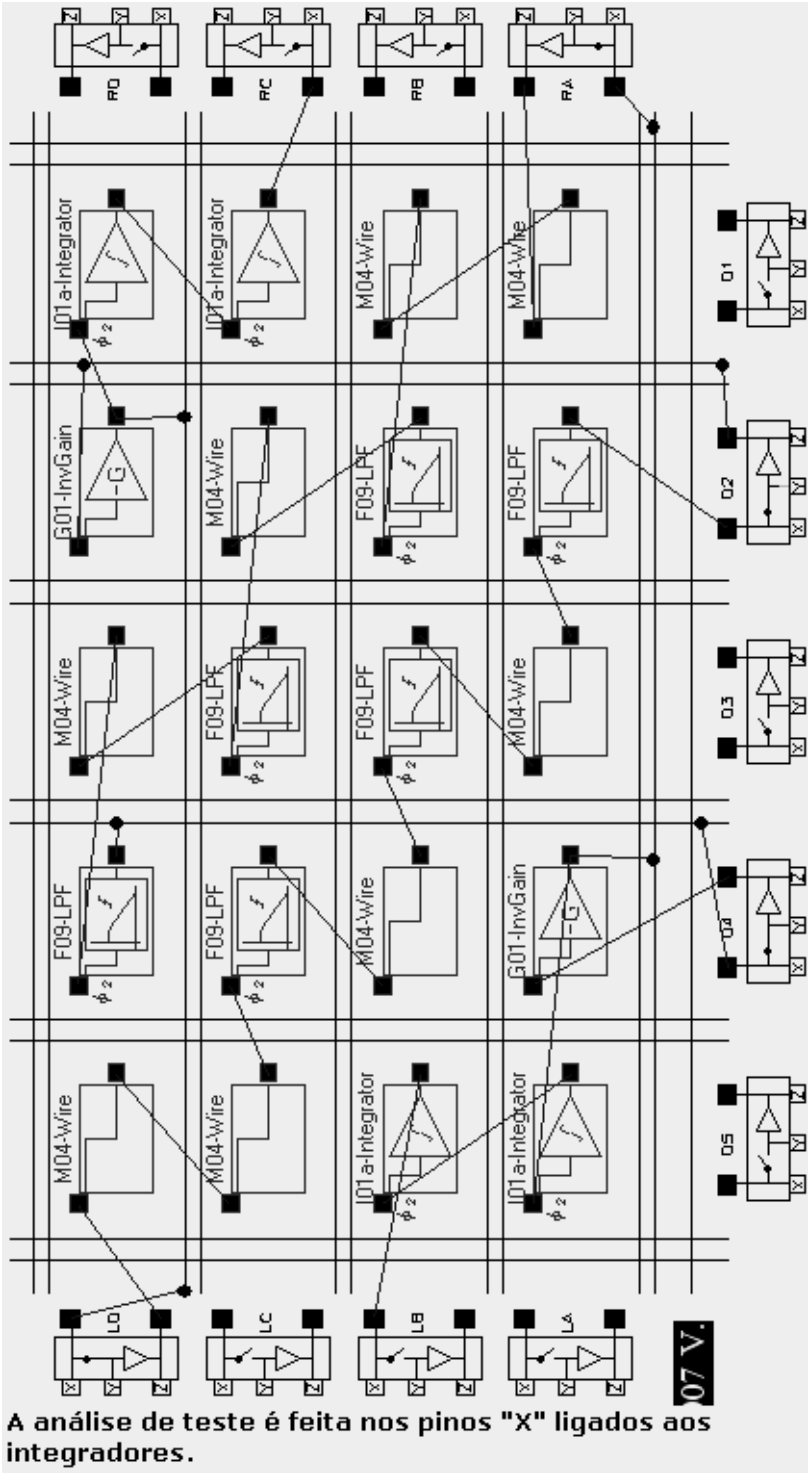


APÊNDICE J CTS DA REDE LOCAL – FALHAS STUCK-OPEN E STUCK-ON – BIST

1ª CONFIGURAÇÃO DE TESTE

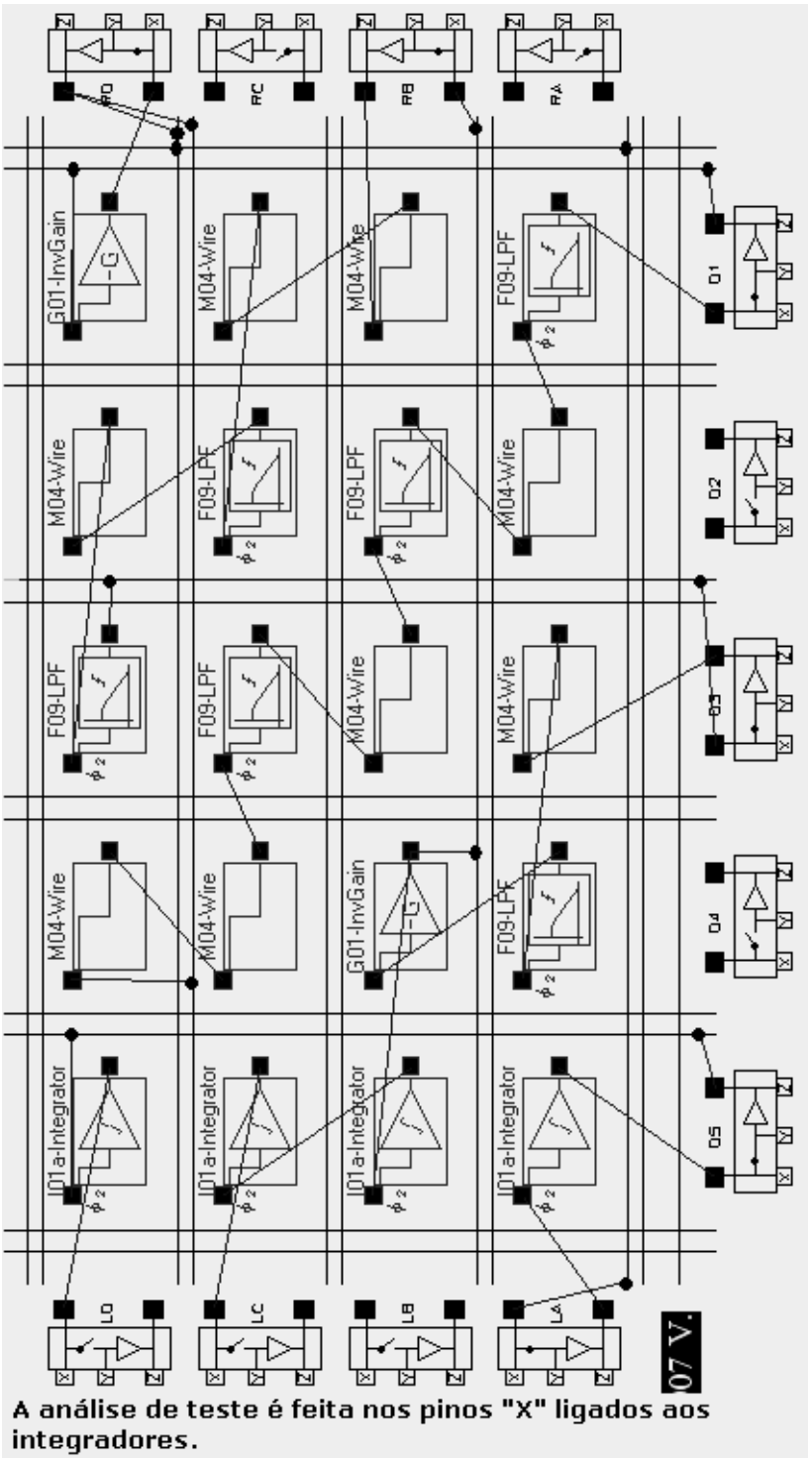


2ª CONFIGURAÇÃO DE TESTE

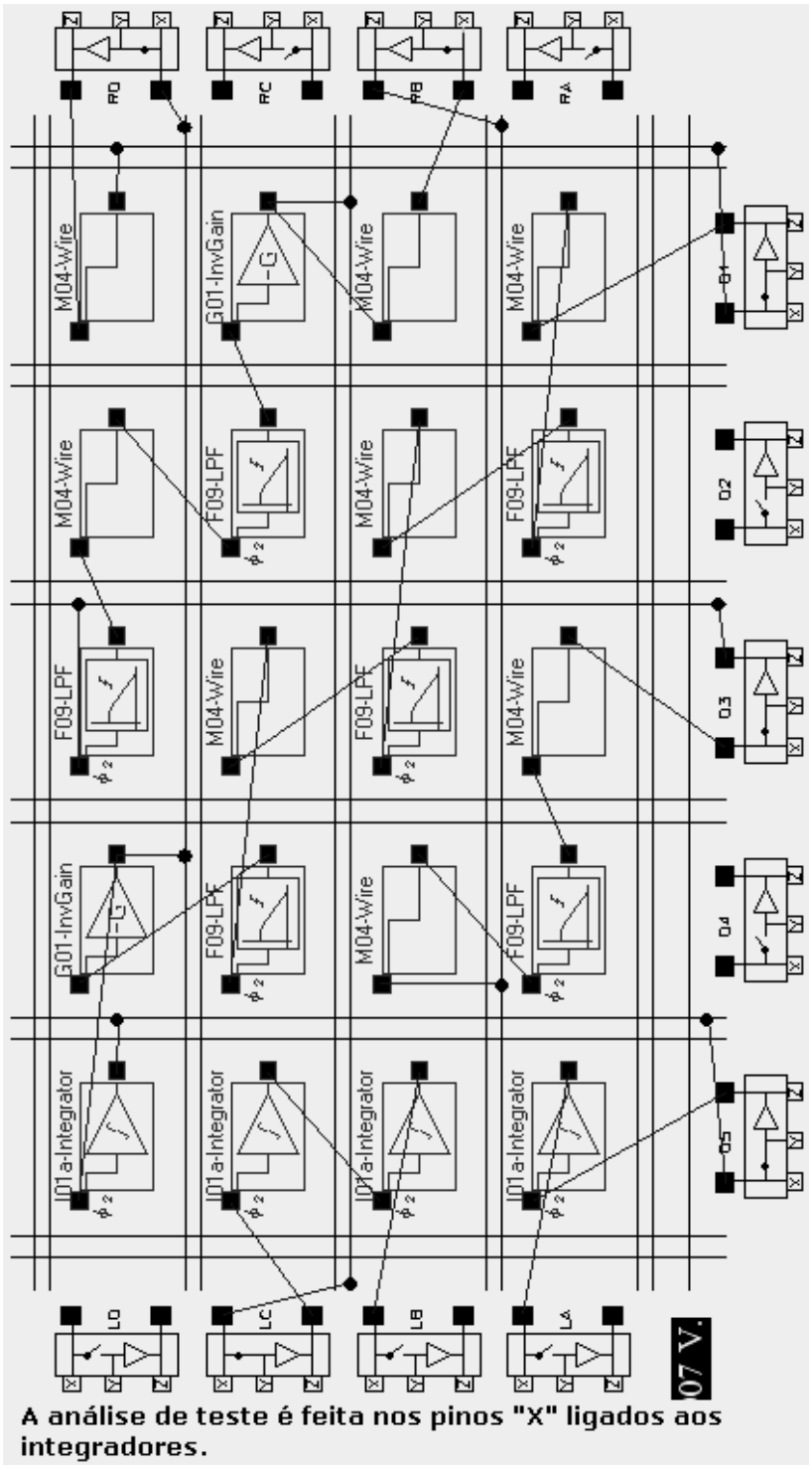


A análise de teste é feita nos pinos "X" ligados aos integradores.

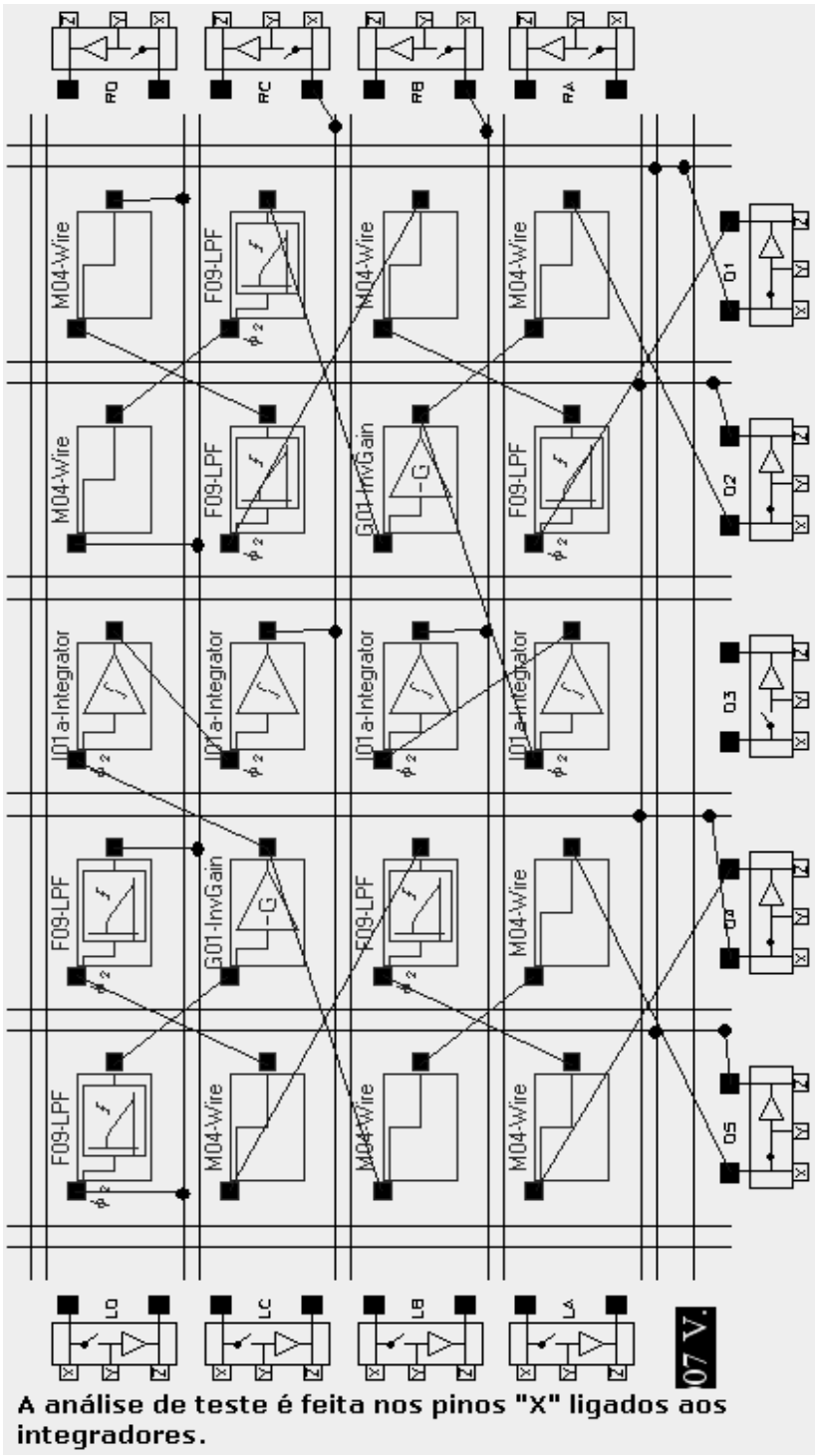
3ª CONFIGURAÇÃO DE TESTE



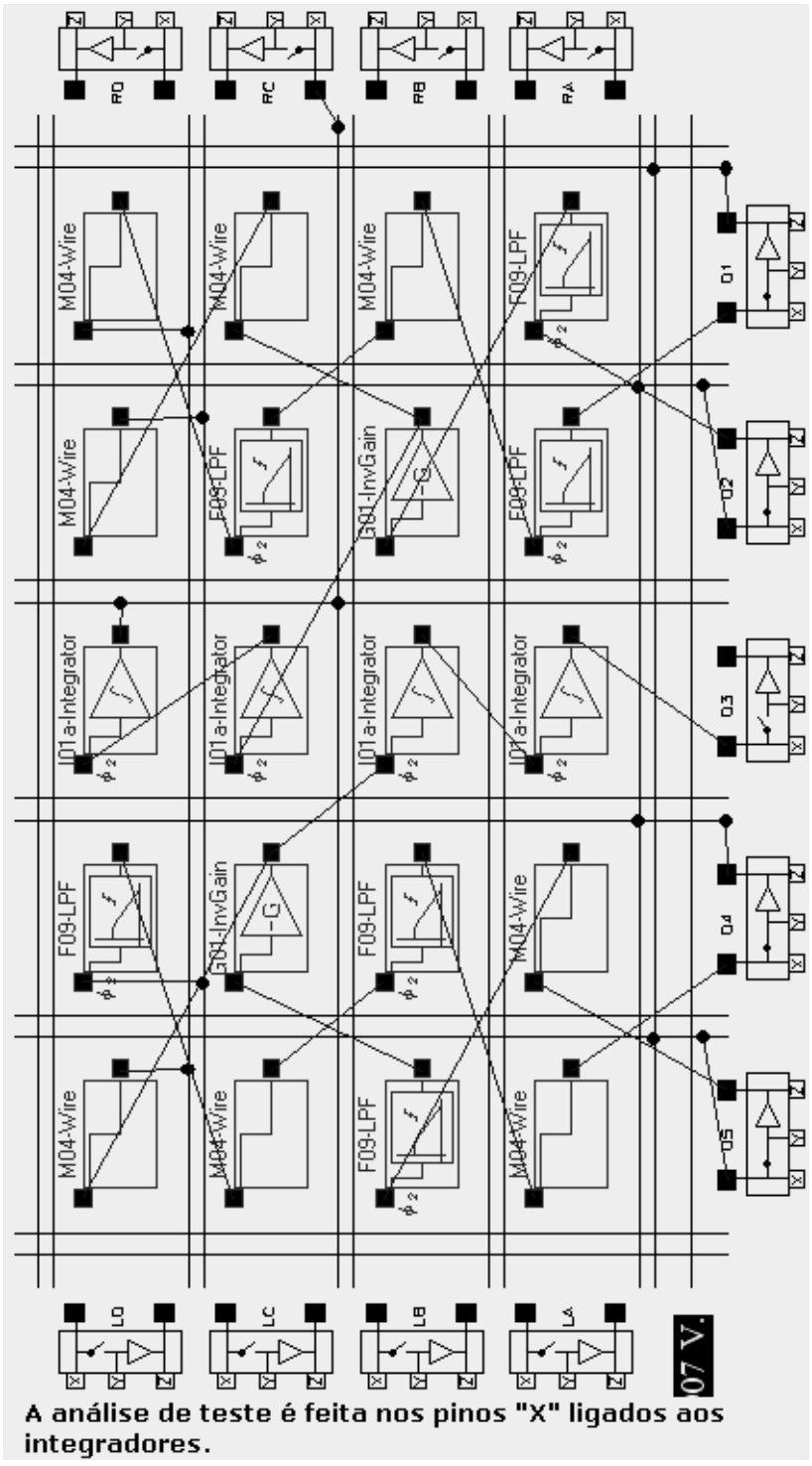
4ª CONFIGURAÇÃO DE TESTE



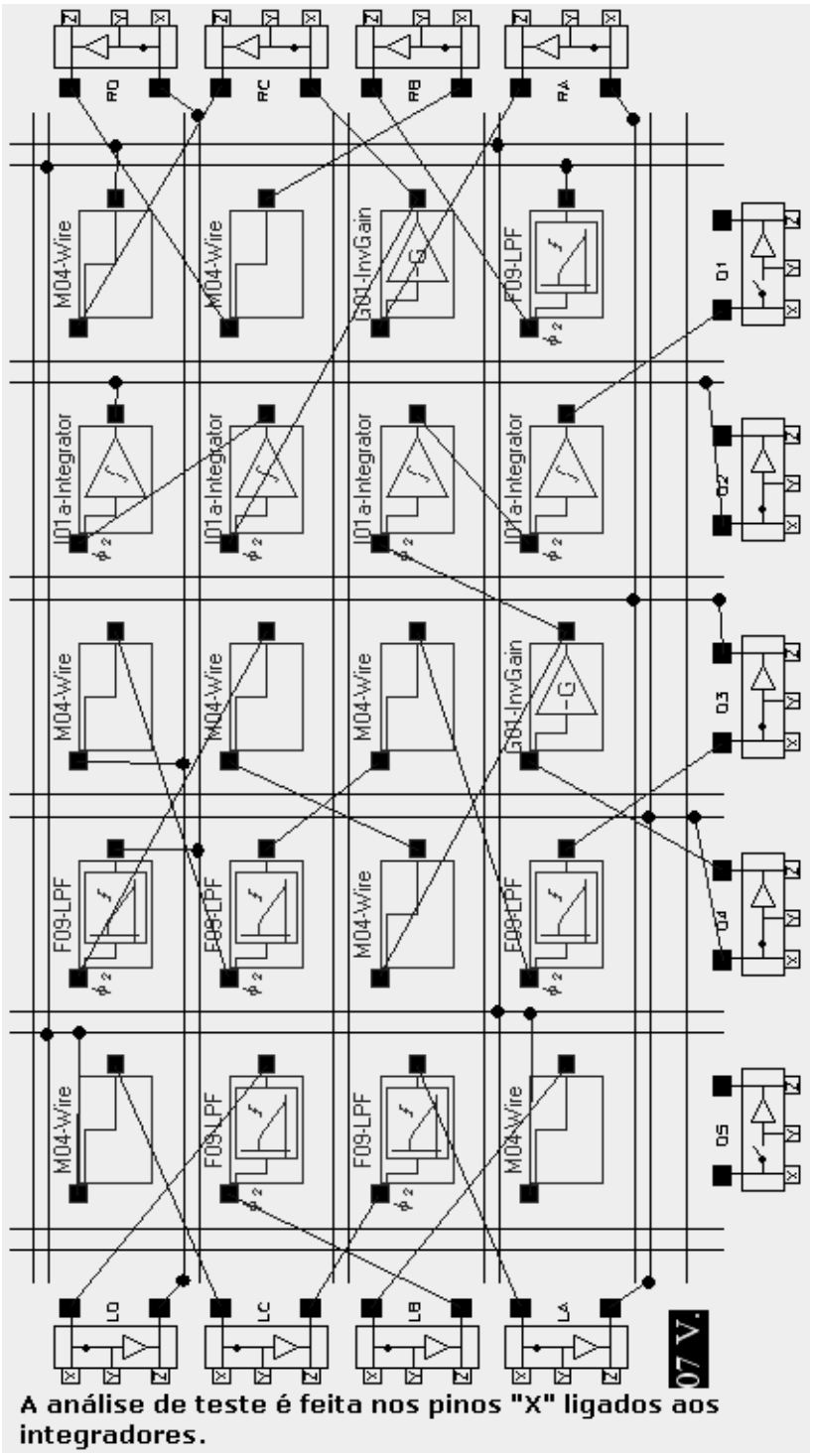
5ª CONFIGURAÇÃO DE TESTE



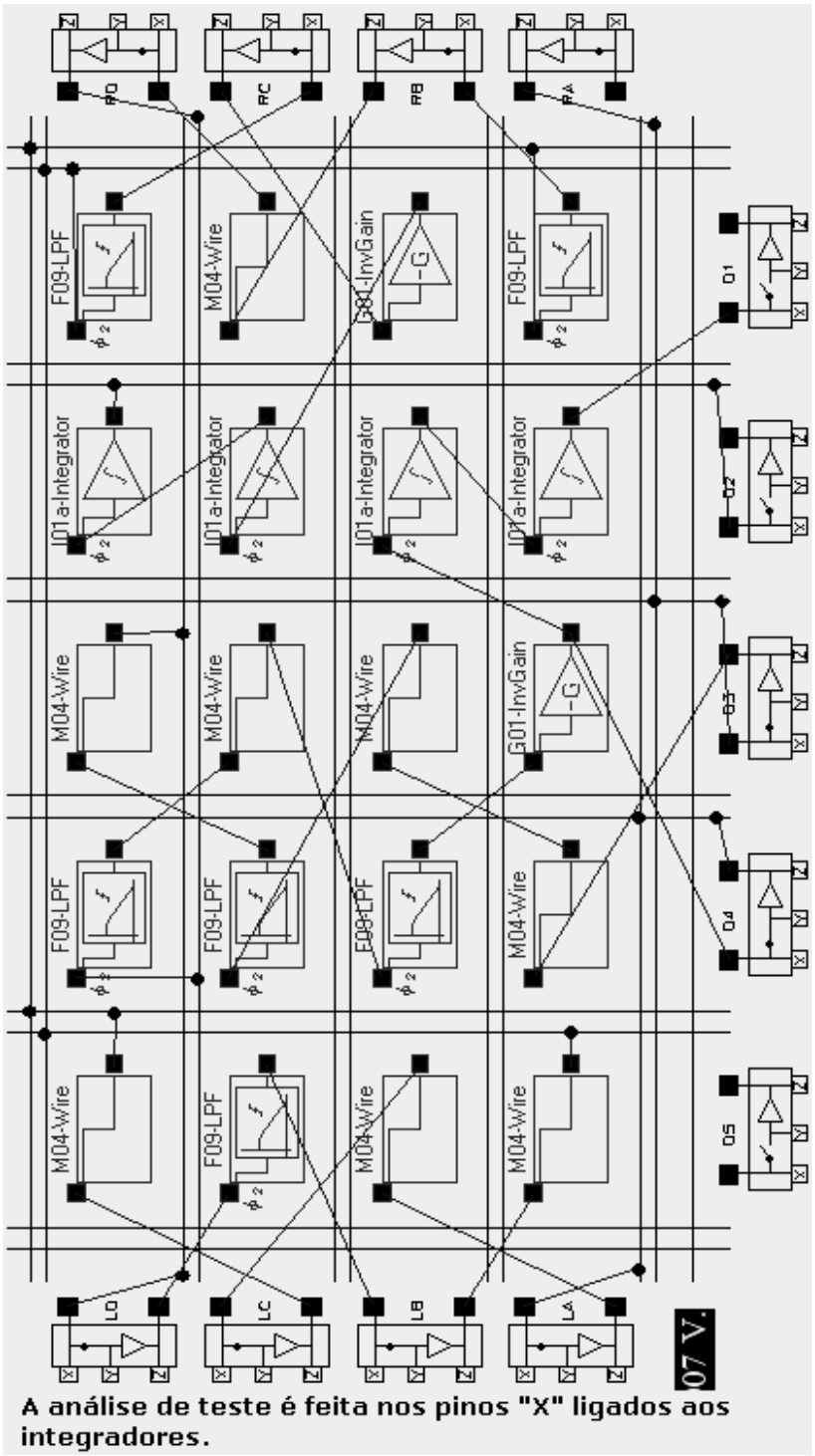
6ª CONFIGURAÇÃO DE TESTE



7ª CONFIGURAÇÃO DE TESTE

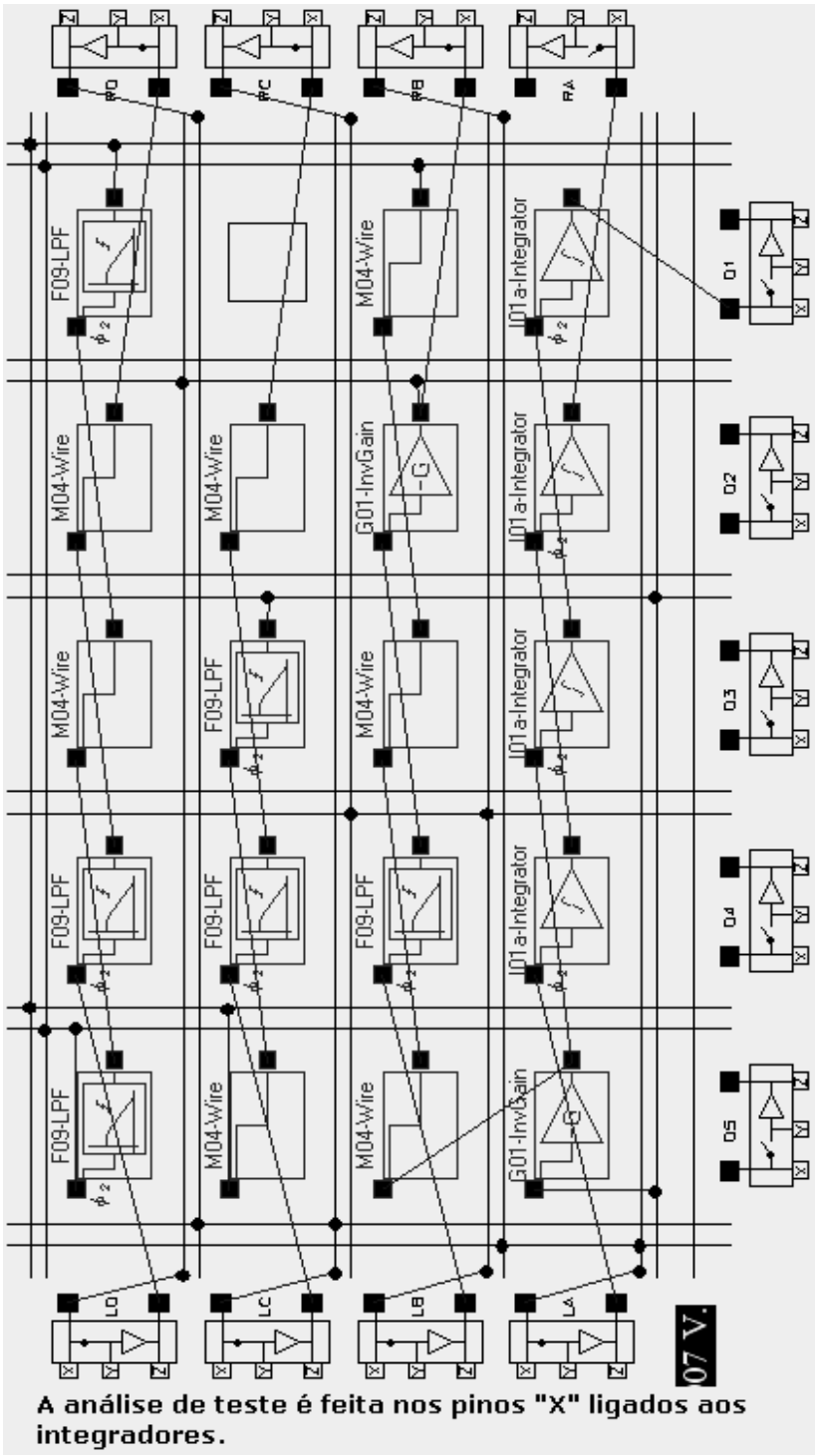


8ª CONFIGURAÇÃO DE TESTE

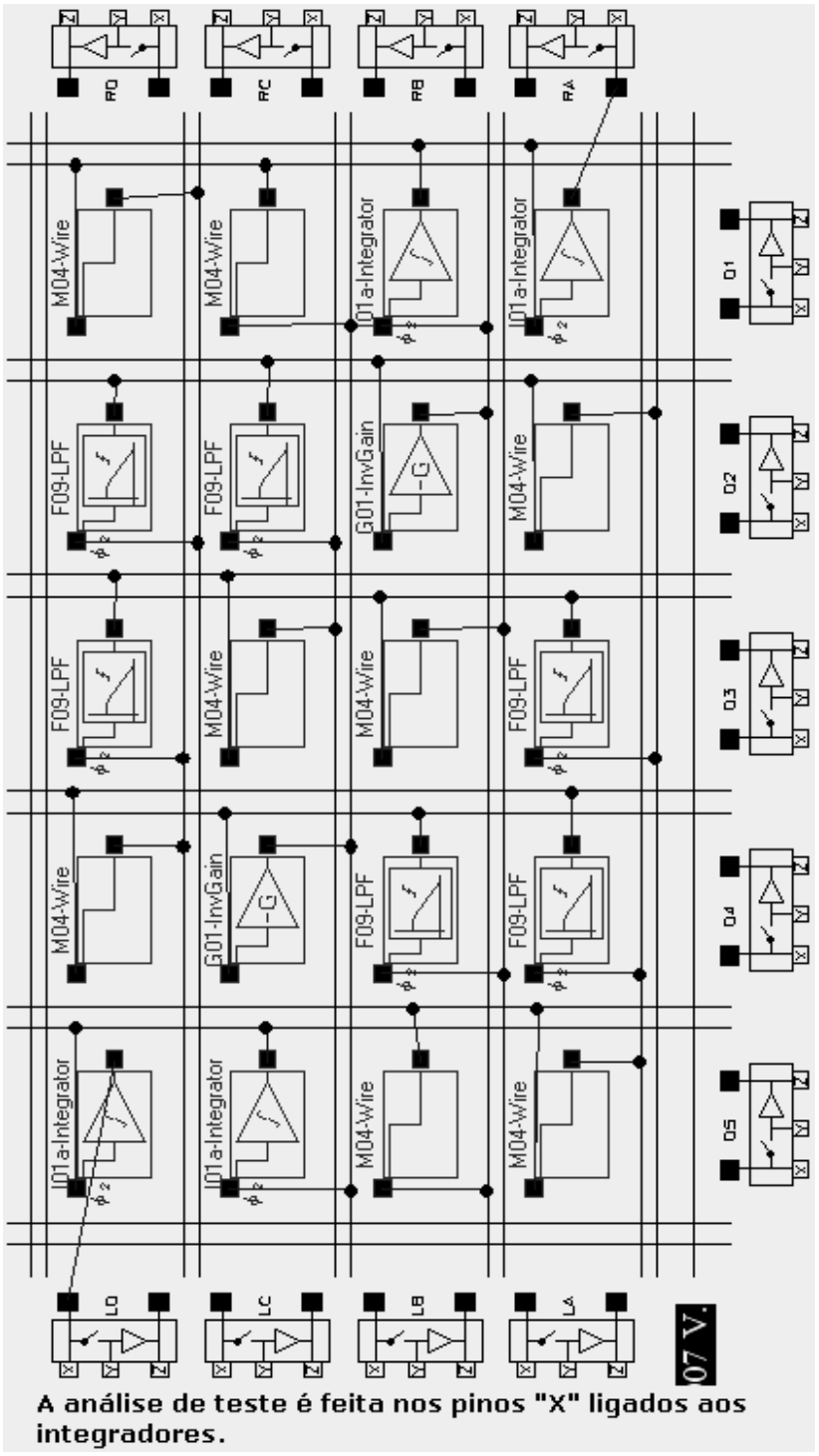


A análise de teste é feita nos pinos "X" ligados aos integradores.

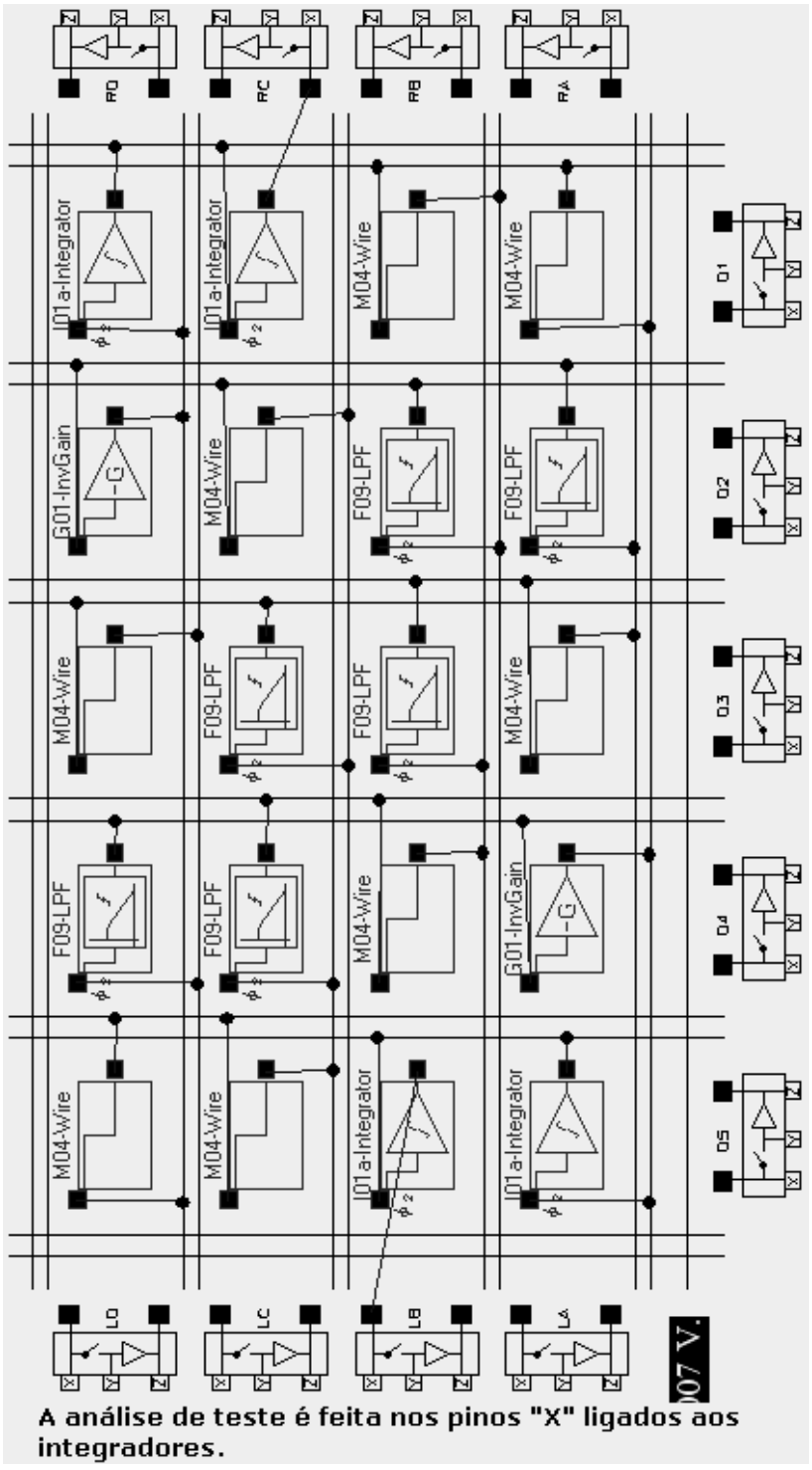
9ª CONFIGURAÇÃO DE TESTE



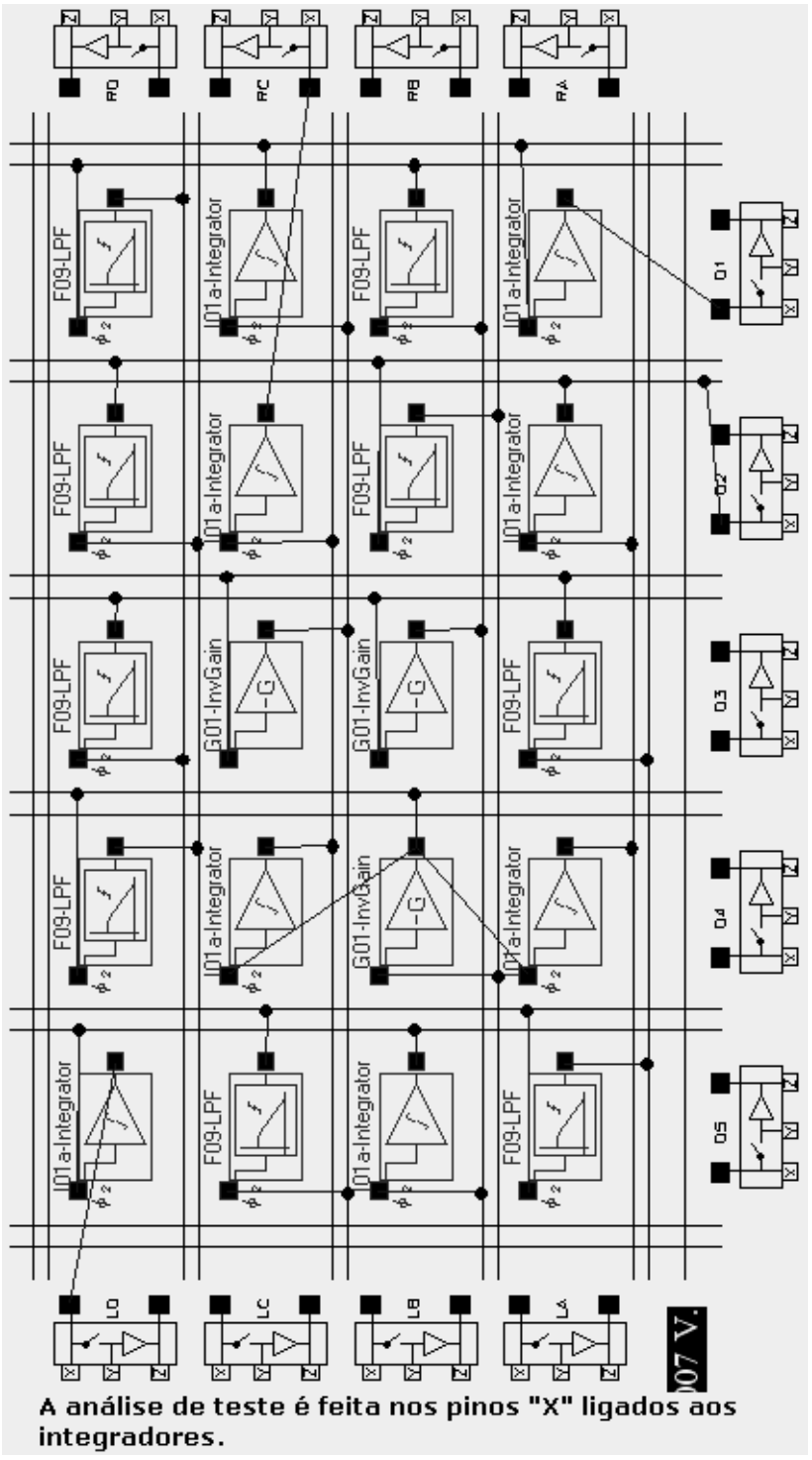
10ª CONFIGURAÇÃO DE TESTE



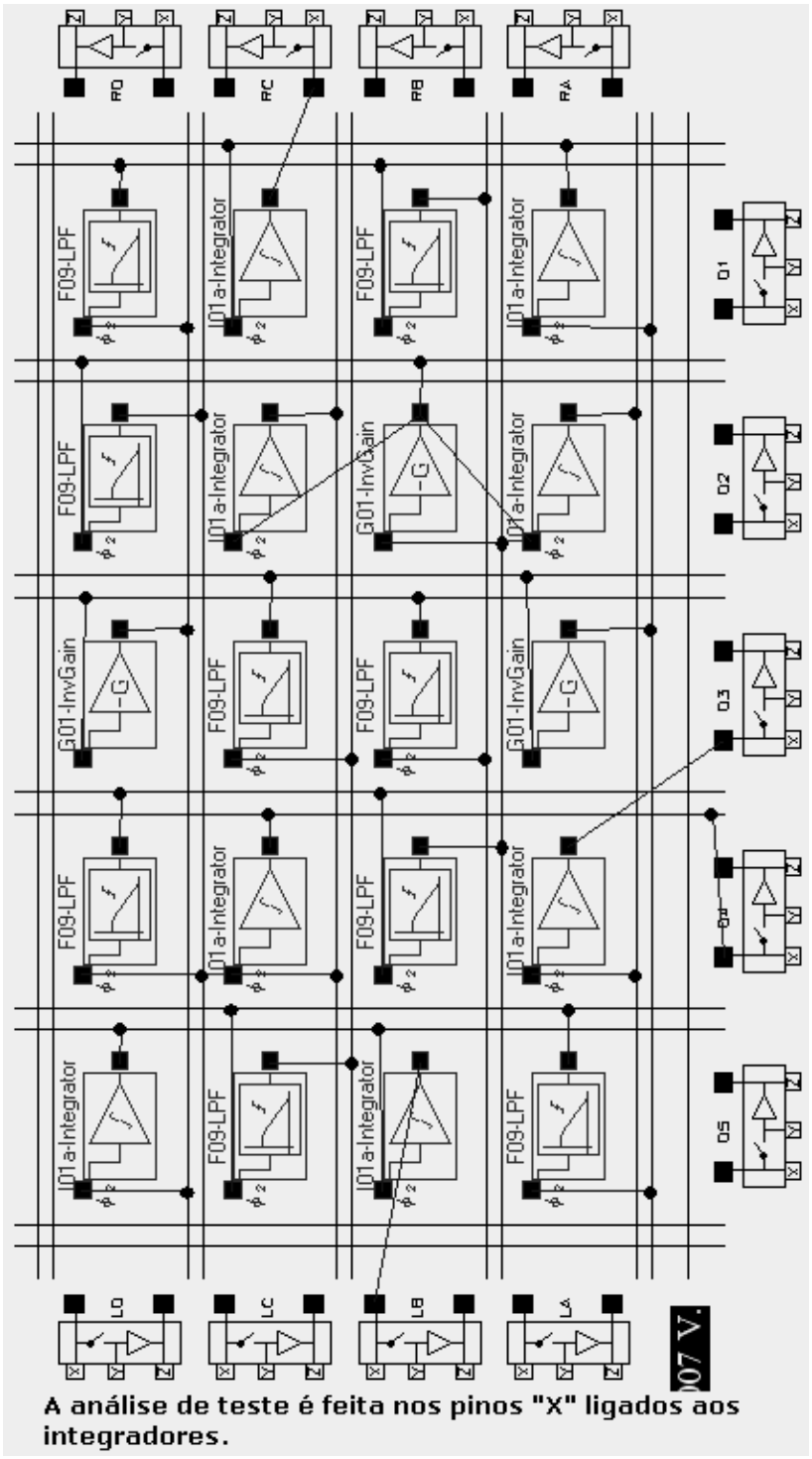
11ª CONFIGURAÇÃO DE TESTE



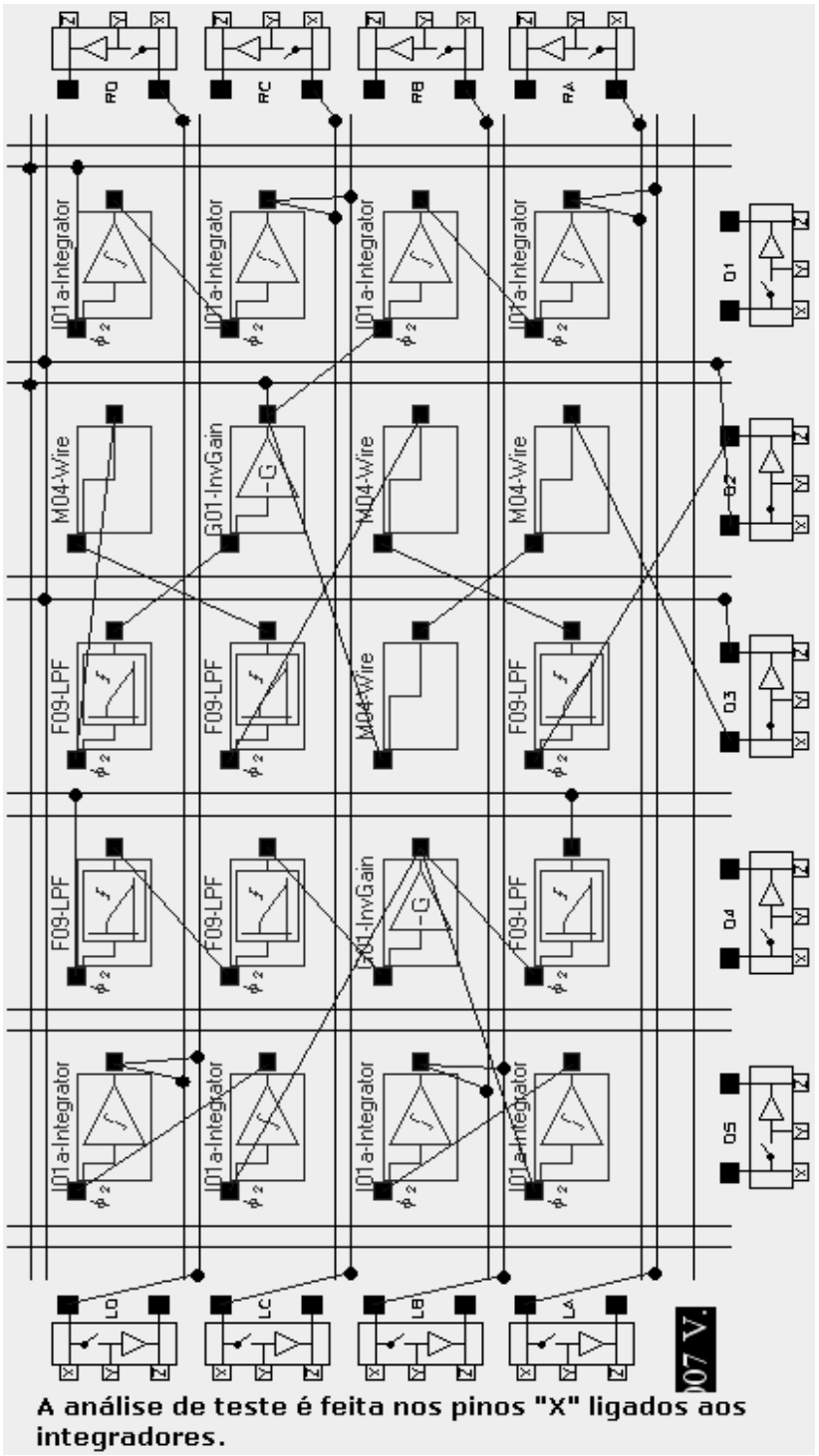
12ª CONFIGURAÇÃO DE TESTE



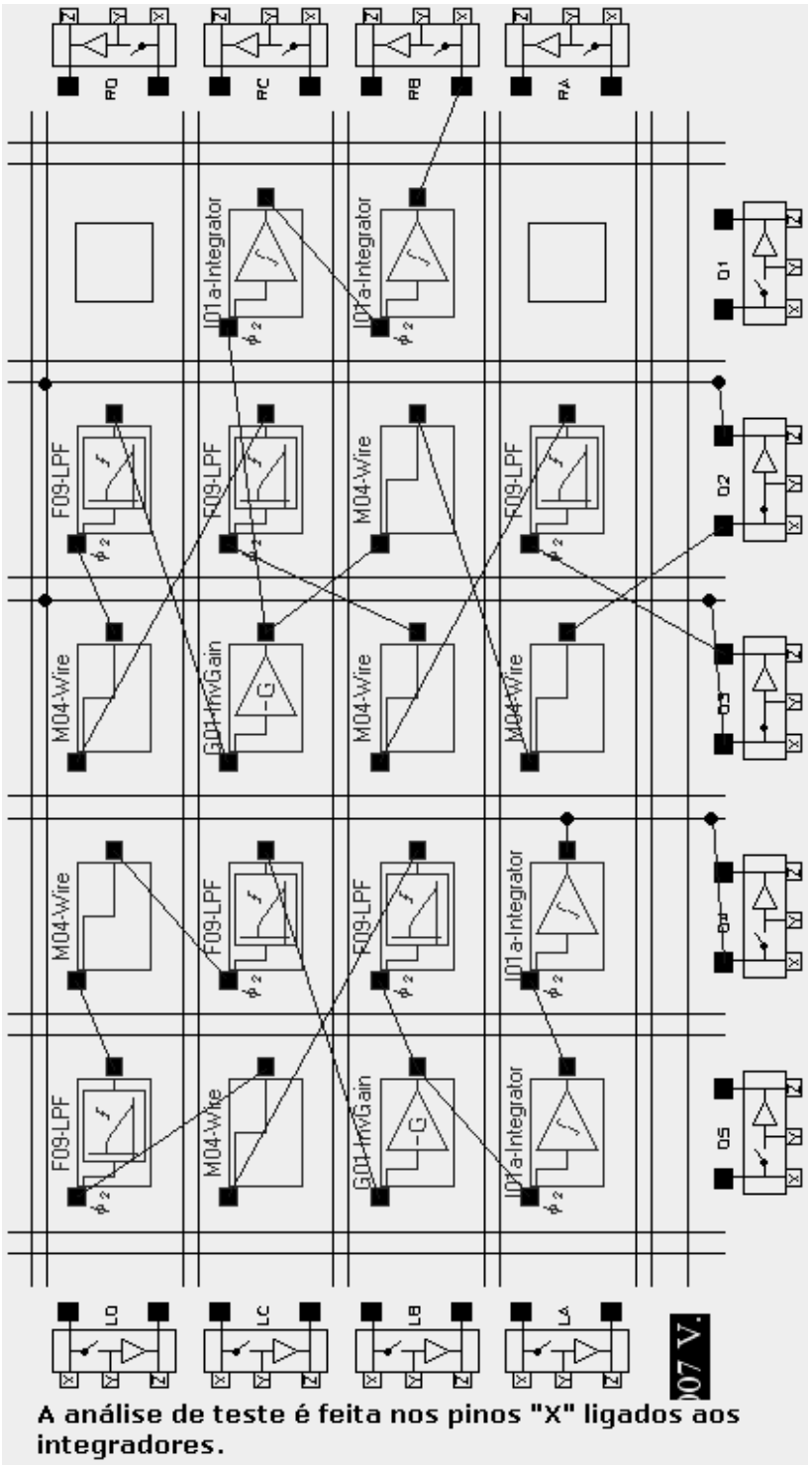
13ª CONFIGURAÇÃO DE TESTE



14ª CONFIGURAÇÃO DE TESTE

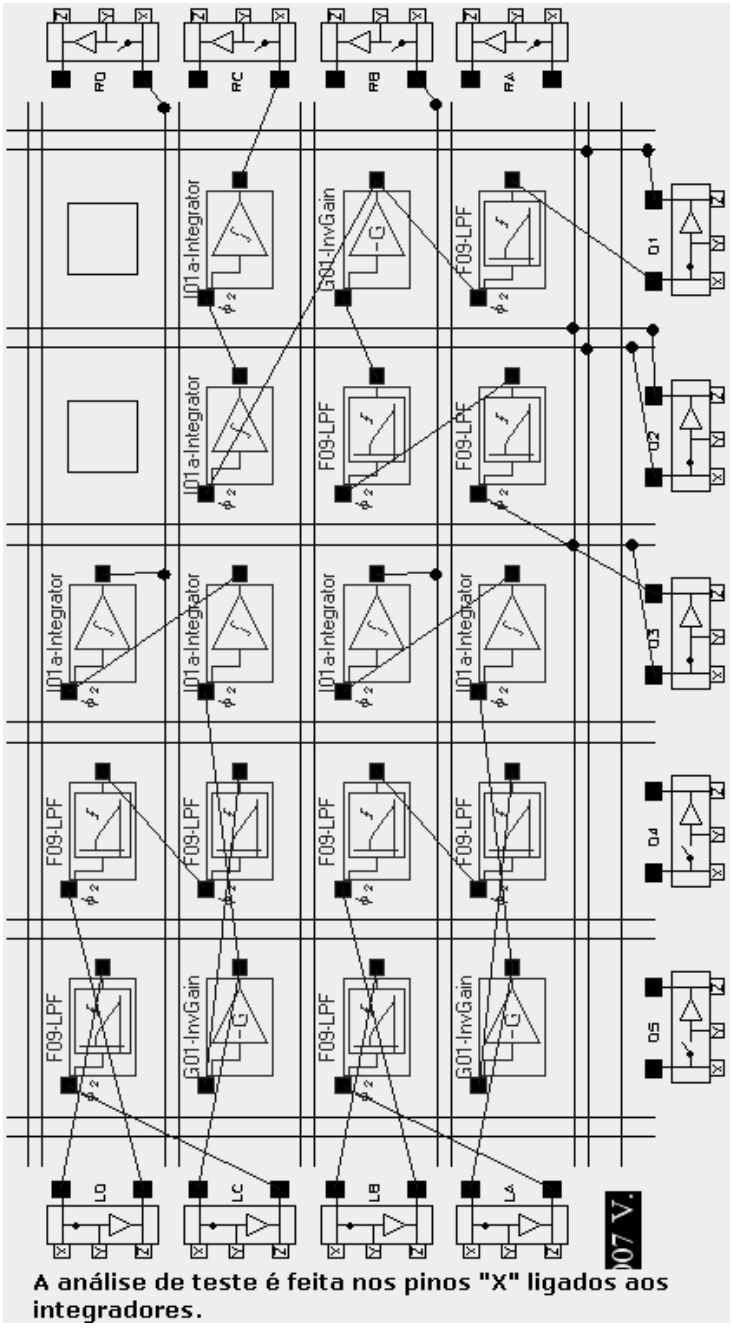


15ª CONFIGURAÇÃO DE TESTE



APÊNDICE L CTS DOS I/OS – FALHAS STUCK-OPEN E STUCK-ON E PARAMÉTRICAS– BIST

1ª CONFIGURAÇÃO DE TESTE



2ª CONFIGURAÇÃO DE TESTE