Machine Learning applied to analog circuits verification

Frederico Coelho, Frank Sill, Janier Arias, Evandro Augusto

1 Inrodução

O objetivo geral deste projeto é o desenvolvimento de ferramentas para a verificação de circuitos analógicos. A ideia é que a partir do diagrama esquemático do circuito, uma rede neural artificial (RNA) seja capaz de identificar e caracterizar os diferentes componentes do circuito, e de prever seu funcionamento. Com isso a rede seria capaz de verificar se o circuito como um todo está funcionando como projetado.

Existem alguns problemas que precisam ser abordados para o desenvolvimento desta ferramenta. O primeiro deles é sobre a própria modelagem dos componentes do circuito. Devemos estudar uma forma de encontrar ou reconhecer os diferentes subcircuitos e suas funções de maneira que depois possamos modelá-lo para simular seu funcionamento. Outro problema no tocante à identificação destes componentes é saber se é possível extrair estas informações apenas dos *netlists* dos circuitos que se quer verificar. Também será necessário extrair parâmetros relacionados a componentes expecíficos dos circuitos e modelar os dados a serem utilizados no processo de aprendizado de máquina.

O primeiro passo é definir a classe de problemas que começaremos a abordar, quais as características que as soluções devem apresentar e como organizar os exemplos de circuitos para treinar as redes neurais. Inicialmente o problema é abordado no nível de conexão dos componentes elétricos em suas unidades mais básicas como resistores, transistores e capacitores. Iniciamos tentando identificar subcircuitos simples como um *espelho de corrente* em um circuito maior. Por enquanto a informação de *layout* dos subcircuitos é deixada de lado, para serem agregadas ao sistema de identificação de subcircuitos em uma etapa posterior a fim de melhorar o grau de acerto caso seja necessário.

2 Referencial teórico

2.1 VERIFICAÇÃO CIRCUITO ANALÓGICO

Segundo [10] O ato de projetar um circuito integrado consiste em traduzir a interpretação do projetista da especificação em algo a ser construído. Depois que o circuito é fabricado, suas características e funcionalidades devem estar em equivalência com o cirucuito planejado e o ato de investigar essa conformidade se chama teste.

Como pode-se perceber na figura 1, a ação de verificar ocorre como um fluxo de trabalho de sentido contrário ao de projeto o circuito. Isso porque, o engenheiro de verificação dispõe do projeto, e o seu objetivo é encontrar, no circuito projetado, os dados definidos na especificação. Quando esta fase estiver finalizada com sucesso, o circuito é enviado para a fabricação, seguida da execução de testes certificadores da conformidade do funcionamento da entidade fabricada com o circuito projetado. Estas duas fases são responsáveis pela continuidade do projeto por confirmarem o seu correto funcionamento.

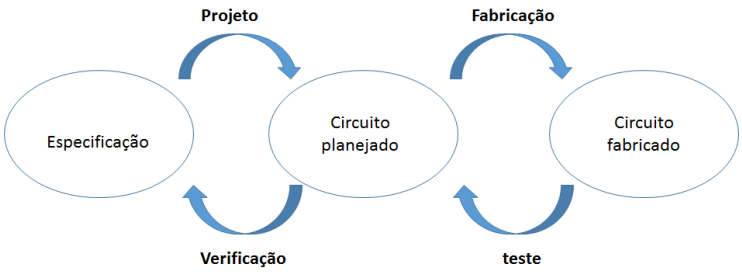


Figure 1: Fluxograma do Alternate test recomendado.

A validação do funcionamento dos circuitos analógicos não é uma tarefa fácil, pondendo envolver vários tipos de análises, como: na frequência, no tempo ou na escolha do ponto de operação dos transistores. Além disso, a escolha de uma técnica de verificação dependerá do propósito do circuito, se o mesmo é linear ou não-linear, das funcionalidades desempenhadas e da sua especificação[10].

2.1.1 Metodo direto de verificação

A verificação funcional do circuito é um tipo de certificação e tem como principal objetivo comprovar o funcionamento do circuito de acordo com sua especificação, ou seja nela se confirma a interseção entre os seguintes aspectos do circuito: a especificação do circuito; a sua implementação e a intenção do projeto[10]

exemplificado na figura abaixo[16].

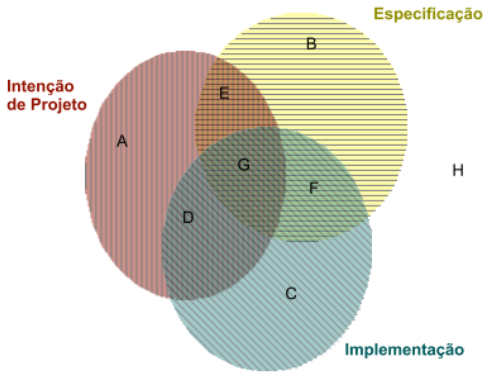


Figure 2: Representação dos universos das possibilidades de intenção do projetista, especificação do projeto e a sua implementação. Sendo A a inteção de projeto, B a especificação e C a implemnetação O principal objetivo da verificação funcional é garantir a conciliação destes três universos, ponto G.

Na verificação funcional existe vários tipos de tecnicas de verificação e elas podem ser classificadas em: técnicas baseadas em simulação, as formais ou as mistas. Nas técnicas baseadas em simulação, encontra-se o comportamento do circuito em diversas condições de operação através do uso de simuladores numéricos como spice6. Os resultados obtidos das simulações são analisados e confrontados com os dados da especificação. No caso das técnicas formais ou mistas não baseadas em simulação, os modelos matemáticos simbólicos dos circuitos são extraídos . Algoritmos formais de verificação são utilizados nestes modelos submetendo-os a exaustivos testes, assim são verificadas várias propriedades que foram abstraídas da especificação do circuito. [10].

2.1.2 Metodo de verificação: teste alternativo

Como a prática padrão para testar circuitos analógicos de sinal misto e de Radiofrequência(RF) são testes baseados em especificações, que consiste em medir um a um dos desempenhos prometidos no datasheet e compara-los com suas especificações.O paradigma de teste alternativo foi proposto como uma substituição dos testes de especificação com o objetivo de simplificar amplamente o processo de teste e reduzir o custo do teste.[17]

O teste alternativo é uma estrutura de teste indireta e sua ideia básica é que as especificações do circuito analógico ou de Radiofrequência(RF) não sejam medidas diretamente, mas deduzidas de um conjunto de parâmetros "fáceis de medir", correlacionados com as especificações do circuito. Infelizmente, a relação

entre esses dois conjuntos de parâmetros é complexa e não pode ser simplesmente identificada com uma função analítica. A solução comumente implementada usa algoritmos de aprendizado de máquina. Mais precisamente, a idéia é aprender durante uma fase de treinamento o mapeamento entre medições indiretas e as especificações do circuito e usar apenas as medições indiretas de baixo custo para prever as especificações do dispositivo durante os testes de produção.[2][3][8]

O processo de teste é desenvolvido em duas etapas: uma etapa de aprendizado e uma etapa de teste. Durante o estágio de aprendizado, as especificações e assinaturas do circuito são medidas a partir de um conjunto de dispositivos de treinamento. Um algoritmo de aprendizado de máquina é treinado nos dois conjuntos de medidas para criar um modelo de mapeamento. No estágio de teste, apenas assinaturas são medidas para cada Dispositivo em Teste (DUT) e as especificações são inferidas usando o modelo de mapeamento obtido. [3]

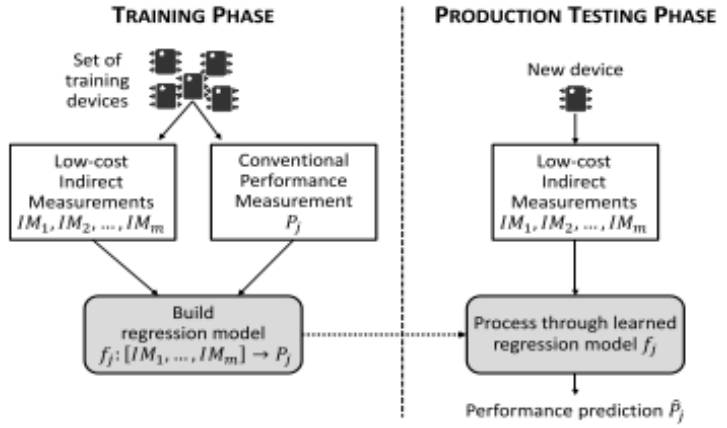


Figure 3: Fluxograma do Alternate test recomendado.

A metodologia proposta por [3] é uma combinação de seleção de características e design guiado de características. A seleção das mesmas pode ser feita por um algoritmo como o Principal Component Analysis(PCA), mas devido seu caráter linear, o método Wrapper approach, neste caso é o recomendado por levar em conta o não linearidade do espaço de assinaturas. Em [3] uma população de 2000 instâncias de um circuito LNA foi gerada usando a simulação de Monte Carlo e repartidas entre teste e treinamento, a fim de fazer o modelo de regressão, mapeando a correlação entre as assinaturas e a especificação. A estratégia de design guiado de características consiste em identificar informações relevantes que estão faltando do conjunto de assinaturas e propor novas assinaturas para cobrir essa infomação. Dito isto, em um processo de fabricação real, dificilmente tem-se acesso a todos os parâmetros físicos. Por outro lado, o processo de fabricação real já é modelado na maioria dos kits de design e

tem como ser acessado os parâmetros do processo no nível da simulação: essa é a base da simulação do processo de Monte Carlo. As variáveis de Monte Carlo podem ser rastreadas até os componentes físicos reais no DUT, graças aos modelos de transistores e elementos passivos. Assim é proposto explorar o conjunto de variáveis de Monte Carlo como se fossem assinaturas adicionais que poderíamos adicionar ao nosso conjunto inicial de assinaturas para treinar modelos de regressão. Como os parâmetros do processo são variáveis aleatórias independentes, simplesmente podemos executar adição gradual.

2.2 GRAFOS

Um grafo é um par (V, A) em que V é um conjunto arbitrário e A é um subconjunto de V (2) . Os elementos de V são chamados vértices e os de A são chamados arestas. Uma aresta como v, w será denotada simplesmente por vw ou por wv . Diremos que a aresta vw incide em v e em w e que v e w são as pontas da aresta. Se vw é uma aresta, diremos que os vértices v e w são vizinhos ou adjacentes. De acordo com nossa definição, um grafo não pode ter duas arestas diferentes com o mesmo par de pontas (ou seja, não pode ter arestas "paralelas"). Também não pode ter uma aresta com pontas coincidentes (ou seja, não pode ter "laços")[9]

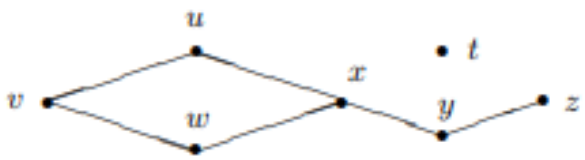


Figure 4: Esta figura é um desenho do grafo cujos vértices são t, u, v , w, x, y , z e cujas arestas são vw, uv , xw, xu, yz e xy .

Em muitas aplicações, uma operação crucial é a comparação entre dois objetos ou entre um objeto e um modelo ao qual o objeto pode estar relacionado. Quando as informações estruturadas são representadas por grafos, essa comparação é realizada usando alguma forma de correspondência de grafos(graph matching). O graph matching é o processo de encontrar uma correspondência entre os nós e as arestas de dois grafos que satisfaça algumas restrições (mais ou menos rigorosas), garantindo que subestruturas semelhantes em um gráfico sejam mapeadas para subestruturas similares no outro.[5]

A correspondência exata de grafo é caracterizada pelo fato de que o mapeamento entre os nós dos dois gráficos deve preservar a aresta no sentido de que, se dois nós no primeiro gráfico estiverem vinculados por uma aresta, eles

serão mapeados para dois nós no segundo gráfico que estão vinculados por uma borda também. Na forma mais rigorosa de correspondência exata, graph isomorphism(isomorfismo de gráfico), essa condição deve se manter nas duas direções e o mapeamento deve ser bijetivo. Ou seja, uma correspondência um a um deve ser encontrada entre cada nó do primeiro gráfico e cada nó do segundo gráfico. Uma forma mais fraca de correspondência é o subgraph isomorphism(isomorfismo do subgrafo), que consiste em, dado grafos H e G determina se existe um isomorfismo (induzido) do subgrafo de H em G. caso um f exista, então se fala que f (H) é cópia de H em G.[11][5] Outra forma de correspondência é chamada de automorfismo que é uma forma de simetria em que o grafo é mapeado em si, ou seja um isomorfismo do grafo com ele mesmo.[14][4]

2.3 Algoritmos nauty e trace

O primeiro programa que conseguiu lidar estruturalmente com grafos regulares de centenas de vértices e grafos com grandes grupos de automorfismo foi o de McKay (1978b, 1980), que mais tarde ficou conhecido como nauty,que consiste em um conjunto de procedimentos de linguagem C muito eficientes para determinar o grupo automorfismo de um gráfico colorido em vértice, ele fornece essas informações na forma de um conjunto de geradores, o tamanho do grupo e as órbitas do grupo. Nauty também é capaz de produzir um isomorfo do gráfico marcado canonicamente, para auxiliar nos testes de isomorfismo. Foi a base do primeiro programa para gerar todos os gráficos de 11 vértices sem isomorfos e pode testar a maioria dos gráficos com menos de 100 vértices em menos de um segundo. O Nauty foi portado com sucesso para uma variedade de sistemas operacionais e compiladores C. Sendo a principal vantagem do nauty sobre os programas anteriores foi o uso inovador de automorfismos para podar a pesquisa. Outro programa importante se tratando de automorfismo foi Traces, o qual foi pioneiro em uma grande revisão da maneira como a árvore de pesquisa é varrida, o que demonstraremos para produzir grandes ganhos de eficiência.[14]. Em nauty, o valor de f(v) é um número inteiro e as regras de poda são aplicadas conforme cada nó é calculado. O Traces introduziu uma grande melhoria, definindo cada f (v) como um vetor em si. Os componentes primários de f (v) são os tamanhos e posições das células na ordem em que são criadas pelo procedimento de refinamento. $\emptyset(G, pi, v)$ torna-se assim um vetor de vetores, denominado trace (daí o nome “Traces”). A vantagem é que muitas vezes permite que a comparação de f (v) e f(v ’) seja feita enquanto o cálculo de v’ está apenas parcialmente completo. Isso torna o bom desempenho no Traces menos dependente da experiência do usuário do que no caso do nauty.

3 Metodologia

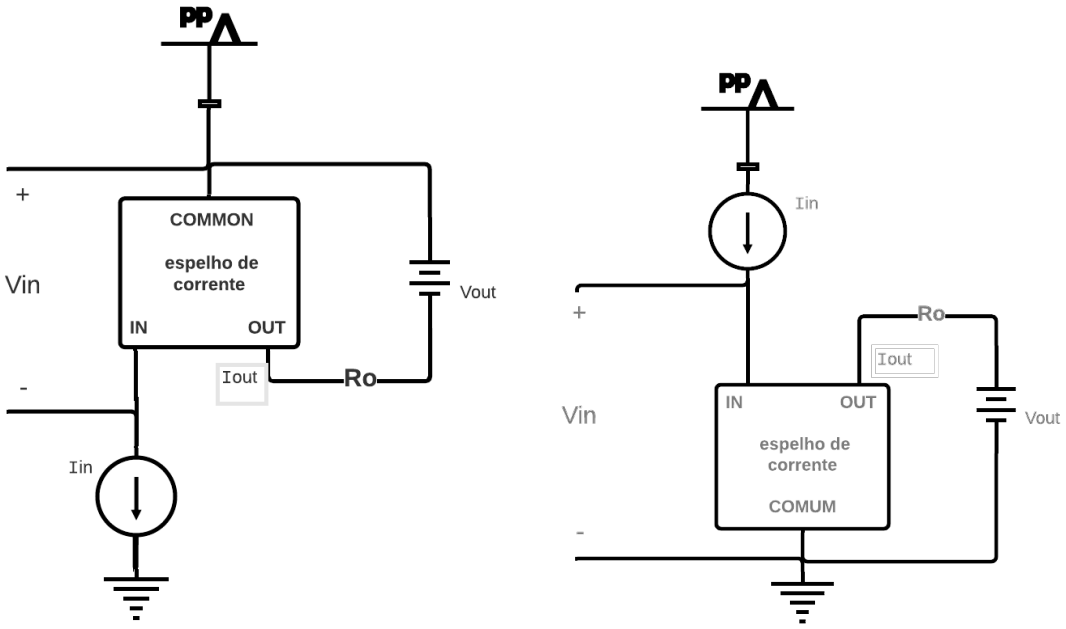
Deve explica e mostrar as arquiteturas de espelho de correntes que queremos identificar e sua representação em grafo.

3.1 Espelho de correntes

Os espelhos de correntes feitos com dispositivos ativos passaram a ser amplamente usados em sistemas analógicos integrados circuitos como elementos de polarização e como dispositivos de carga para estágios de amplificador. O uso de espelhos de correntes em polarização podem resultar em insensibilidade superior do desempenho do circuito a variações em fonte de alimentação e temperatura. Os espelhos de correntes são frequentemente mais econômicos do que os resistores em termos da área da matriz necessária para fornecer a corrente de polarização de um certo valor, particularmente quando o o valor necessário da corrente de polarização é pequeno. Quando usado como um elemento de carga em amplificadores de transistor, a alta resistência incremental do espelho atual resulta em alto ganho de tensão em baixas tensões de alimentação.

Um espelho de corrente é um elemento com pelo menos três terminais, conforme mostrado na figura 5 . O comum terminal está conectado a uma fonte de alimentação, e a fonte de corrente de entrada está conectada à entrada terminal. Idealmente, a corrente de saída é igual à corrente de entrada multiplicada por uma corrente desejada ganho. Se o ganho for unitário, a corrente de entrada é refletida na saída, levando ao nome atual espelho. Sob condições ideais, o ganho do espelho de corrente é independente da frequência de entrada, e a corrente de saída é independente da tensão entre a saída e os terminais comuns. Além disso, a tensão entre os terminais de entrada e comuns é idealmente zero porque isso condição permite que toda a tensão de alimentação apareça na fonte de corrente de entrada, simplificando seu design em nível de transistor. Mais de um terminal de entrada e / ou saída às vezes são usados. Na prática, os espelhos reais de corrente no nível do transistor sofrem muitos desvios deste ideal comportamento. Por exemplo, o ganho de um espelho de corrente real nunca é independente da entrada frequência[12].

1. Um dos desvios mais importantes é a variação da corrente de saída do espelho de corrente com mudanças na tensão no terminal de saída. Este efeito é caracterizado pela resistência de saída de pequeno sinal, Ro do espelho de corrente. O modelo equivalente ao Norton da saída do espelho de corrente inclui Ro em paralelo com uma fonte de corrente controlada pela corrente de entrada. A resistência de saída afeta diretamente o desempenho de muitos circuitos que usam espelhos atuais. Por exemplo,



(a) positive supply. (b) Ground.

Figure 5: Diagrama de blocos para espelhos de correntes (a) positive supply (b) ground

a taxa de rejeição de modo comum do diferencial amplificador depende diretamente desta resistência, assim como o ganho dos circuitos de carga ativa. Aumentar a resistência de saída reduz a dependência da corrente de saída na saída tensão e, portanto, é desejável. De um modo geral, a resistência de saída aumenta em circuitos práticos quando a corrente de saída diminui. Infelizmente, diminuindo a saída a corrente também diminui a velocidade máxima de operação. Portanto, ao comparar a saída resistência de dois espelhos de corrente, eles devem ser comparados em correntes de saída idênticas.

2. Quando a fonte de entrada de corrente é conectada ao terminal de entrada de um espelho de corrente real, ele cria uma queda de tensão positiva, VIN, que reduz a tensão disponível na entrada fonte atual. Minimizar o VIN é importante porque simplifica o design da entrada fonte de corrente, especialmente em aplicações de baixo fornecimento. Para reduzir o VIN, os espelhos atuais às vezes tem mais de um terminal de entrada.

3. Quando a fonte de entrada de corrente é conectada ao terminal de entrada de um espelho de corrente real, ele cria uma queda de tensão positiva,

VIN, que reduz a tensão disponível na entrada fonte atual. Minimizar o VIN é importante porque simplifica o design da entrada fonte de corrente, especialmente em aplicações de baixo fornecimento. Para reduzir o VIN, os espelhos de correntes às vezes tem mais de um terminal de entrada.

Os espelhos de correntes que serão utilizados nos circuitos deste trabalho serão compostos apenas por transistores mosfet canal n ou canal p, mas apesar de serem utilizados apenas esses dois tipos de transistores são muitas configurações diferentes de espelhos de correntes que podem ser criados, segue como exemplo a figura 6 .

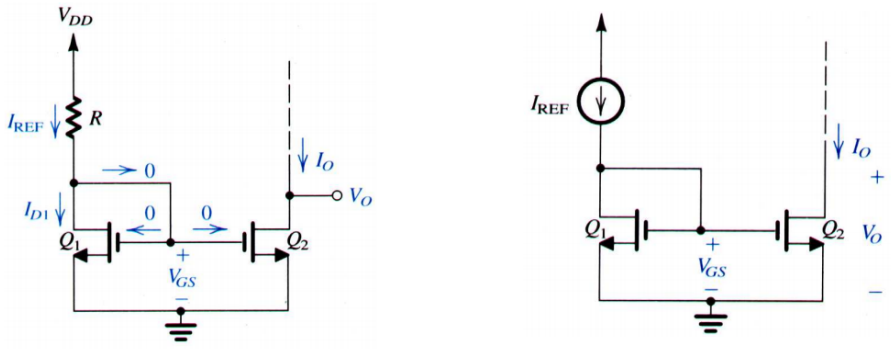


Figure 6: Espelhos de corrente com Mosfest.[1]

3.2 Representação dos circuitos através da netlist

Tópico sobre a representação dos circuitos com uma netlist. Tem de explicar o que é uma netlist, mostrar exemplos de uma parte de uma netlist, etc.

No projeto eletrônico, uma netlist é uma descrição textual da conectividade de um circuito eletrônico. [13] [15] Em sua forma mais simples, uma netlist consiste em uma lista dos componentes eletrônicos em um circuito e uma lista dos nós aos quais eles estão conectados [13], geralmente utilizada em ferramentas de simulação analógica.

A estrutura, complexidade e representação das netlists podem variar consideravelmente, mas o propósito fundamental de cada netlist é transmitir informações de conectividade. Netlists geralmente fornecem nada mais do que instâncias, nós e talvez alguns atributos dos componentes envolvidos. [4] Se expressarem muito mais do que isso, geralmente são considerados uma linguagem de descrição de hardware, como Verilog ou VHDL, ou uma das várias linguagens projetadas especificamente para entrada em simuladores.

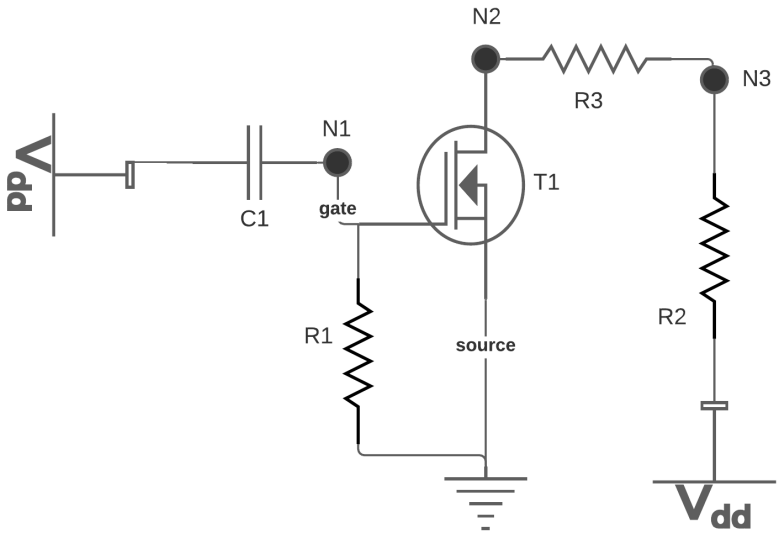


Figure 7: Circuito e netlist

3.3 Visão geral

tópico explicando como vamos usar os grafos pra resolver o problema

Neste trabalho estamos considerando que toda informação do circuito vem da especificação do mesmo através de uma netlist e o objetivo é criar uma algoritmo capaz de identificar se existe estruturas espelhos de correntes na netlist, a quantidade e a localização dessas estruturas também. A figura 8 está exemplificando as etapas que foram feitas a fim de se chegar neste objetivo.

Possuindo a especificação do circuito em forma de netlist temos que modelar essa informação de modo que se consiga resolver o problema para isso temos que construir um algoritmo que seja capaz de ler essa netlist que é um arquivo .txt, após a leitura deste arquivo temos que tratar os dados de forma que possibilite fazer uma varredura nos dados procurando por estruturas especificas de espelho de correntes e para isso foi eleito a representação da netlist em um grafo como já foi tratado no tópico 3.1, os principais fatores que contribuíram para essa escolha é a possibilidade de achar estruturas isomórficas em grafos, os algoritmos nauty e o traces vistos na sessão 2.3 são bons exemplos de algoritmos que conseguem fazer o trabalho de encontrar estruturas isomórficas dentro de grafos, o que faz muito sentido em um circuito, pois o mesmo circuito pode ser construído de diversas formas diferentes e continuar possuindo as mesmas características, outro motivo é a existência de outra gama de algoritmos de graph matching,

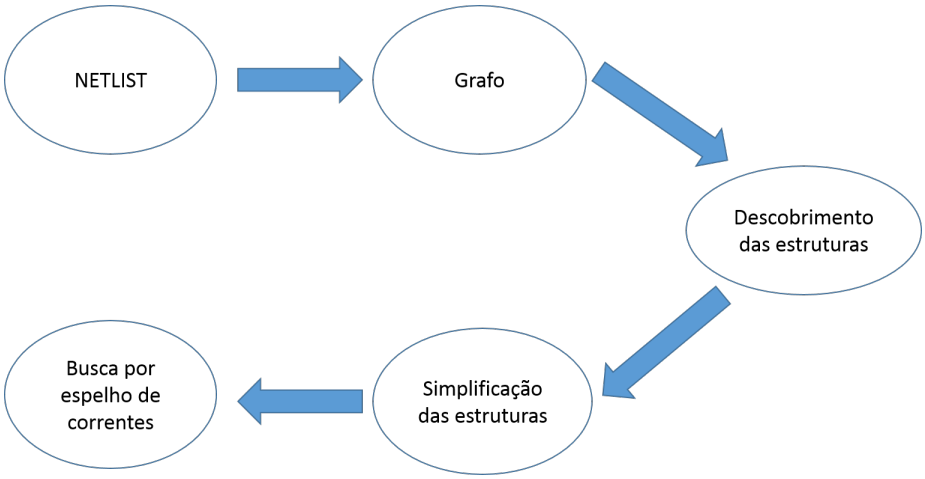


Figure 8: Simplificação do problema

que consistem em percorrer um grafo e encontrar estruturas específicas dentro do mesmo. Sendo assim o descobrimento de estruturas totalmente equivalentes, ou que possuem isomorfismo se torna viável, mas que para isso tudo aconteça as estruturas do circuito tem que serem bem modeladas da netlist para o grafo de forma que não seja perdida nenhuma informação e isso só irá ocorrer se as três estruturas principais que são os resistores, capacitores e transistores conservem todas as suas características o que será discutido no tópico 3.4. Neste trabalho todo circuito irá conter apenas resistores, transmissores e capacitores, como mostrado na figura 7 quando descrito em netlist cada linha equivale uma dessas estruturas, sendo assim ao transformar a netlist em grafo coloca-se o pesos diferentes nos nós dependendo da estrutura para diferencia-las, isso será melhor explicado no tópico 3.4. Na etapa de simplificação das estruturas é feito uma simplificação de resistores e capacitores em serie e em paralelo facilitando que a busca por espelho de correntes que é o principal objetivo deste trabalho.

3.4 descobrimento das estruturas

Deve ter um tópico explicando direitinho essa conversão de netlist para grafo mostrando exemplos e explicando certinho como representamos cada componente, os pesos das arestas pra identificar os terminais dos elementos, etc.

No item descobrimento de estruturas você deve elaborar mais. Tem de explicar como identificaremos os modelos de espelhos de corrente no circuito maior, etc

melhorar a parte sobre a representação do circuito com os grafos

A primeira etapa foi escolher um meio de representação das estruturas dos circuitos de forma que se tornasse possível dentro dela a identificação de múltiplos circuitos específicos(exemplo: espelho de correntes, resistores e capacitores em série e paralelo). A principio o circuito analógico é descrito em forma de netlist de acordo com a seguinte sintaxe.

Netlist Sintaxe:

RESISTOR: R12 (net14 net18) R=73E4

- Tipo do elemento: R(resistor)
- Nome do elemento: R12
- Conexões (pino1 , pino2): que são (net14 net18)
- Parâmetro: resistência = 7.3E4

CAPACITOR: C65 (net24 net23) C=6.7E-8

- Tipo de elemento: C (capacitor)
- Nome do elemento: C65
- Conexões (pin1, pin2): to nets net24, net23
- Parâmetro: capacitância = 6.7E-8

MOSFET: T64 (net24 net13 net13 0) nfet L=7.93E-6 W=0.52E-6

- Tipo de elemento: T (transistor)
- Nome do elemento:T64
- Conexões: (drain, gate, source, bulk): to nets net24 net13 net13 0
- Tipo: nfet
- Paramêtros: Length = 7.93E-6, Width = 0.52E-6

Como já explicado no tópico 3.2, a netlist é a descrição textual do circuito e a forma em que a informação do circuito é descrita pela netlist é mostrado acima em *Netlist sintaxe*, sendo assim pode-se perceber que já é possível a identificação de todos os transistores, capacitores e resistores, pois cada linha da netlist equivale a um destes componentes com todas as informações que precisamos para identificar a localização do mesmo, no caso já é descrito pela netlist entre quais pinos(nós) estes componentes se encontram, assim como as informações de resistência, capacitância, length(comprimento) e width(largura) dos componentes.

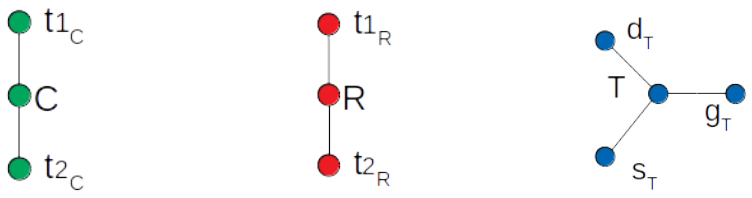
Porém ao tentar identificar estruturas de componentes maiores(e.g estruturas de resistores e capacitores em série e paralelo, espelhos de correntes) se torna um grande desafio, a solução encontrada para a identificação de tais estruturas maiores foi transformar a netlist do circuito em um grafo. A figura 9 é um exemplo de uma netlist valida que pode ser lida e modelada pelo algoritmo.

Depois desenhar o grafo dessa netlist

```
C1 (net6 net5) C=8.2E-5
R2 (net1 net8) R=7.0E4
T3 (net2 net4 net4 0) pfet L=3.90E-6 W=5.24E-6
R4 (net4 net2) R=6.0E5
C5 (net11 net3) C=7.8E-12
T6 (net6 net1 net1 0) pfet L=7.76E-6 W=6.84E-6
R7 ( net8 net1)  R=9.5E3
T8 ( net1 net1 net10 0)  nfet  L=9.69E-6 W=0.73E-6
T9 ( net10 net5 0 0)  nfet  L=9.69E-6 W=0.73E-6
T10 ( net5 net5 0 0)  nfet  L=9.69E-6 W=0.73E-6
T11 ( vdd! net1 net5 0)  nfet  L=9.69E-6 W=0.73E-6
T12 (net4 net6 net6 0) nfet L=6.30E-6 W=5.94E-6
T13 (net12 net4 net4 0) nfet L=7.99E-6 W=9.50E-6
T14 (net4 net11 net11 0) nfet L=8.82E-6 W=5.62E-6
```

Figure 9: Exemplo de Netlist de entrada

Quando o CV é lido da *netlist*, é gerado um grafo para representá-lo. Inicialmente, estamos trabalhando com circuitos que contenham apenas transistores, resistores e capacitores. Cada componente do circuito é carregado e modelado conforme mostrado na Figura 10.



(a) Capacitor. (b) Resistor. (c) Transistor.

Figure 10: Modelos de grafos para cada componente do circuito.

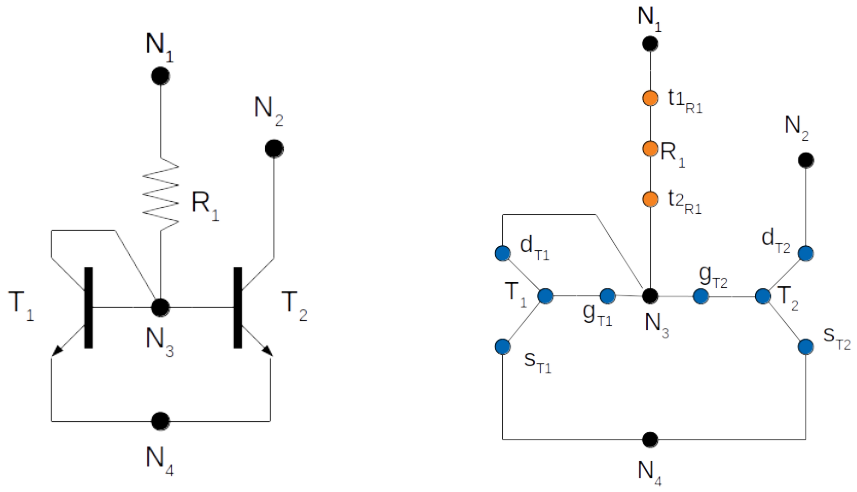
Cada elemento é modelado por um nó que representa seu tipo (C para capacitor, R para resistor e T para Transistor), ou seja, é o nó central que representa o elemento do circuito em si. Cada terminal do elemento é também representado por um nó. No caso dos capacitores e resistores teremos dois terminais representados pelos nós t1 e t2 e no caso dos transistores teremos três nós rep-

resentando os terminais *gate*, *source* e *drain*. No caso dos transistores cada terminal tem uma função específica, e dependendo de como eles estão conectados ao CV eles podem caracterizar subcircuitos de comportamentos distintos. Sendo assim cada aresta que conecta os terminais a cada elemento terão pesos diferentes para facilitar a sua comparação. As arestas (conexões) do nó resistor aos seus nós terminais receberão o peso 4 enquanto que as conexões do nó capacitor aos seus terminais receberão o peso 5. A conexão do terminal S do transistor receberá o peso 3, a conexão do gate receberá o peso 2 e a conexão do drain o peso 1. As diferentes arquiteturas dos subcircuitos a serem encontrados no CV também são modelados desta forma.

Uma vez modelados os circuitos e subcircuitos utilizamos o algoritmo VF2 [6, 7] para teste de isomorfismo em grafos para identificar se existem estruturas no CV semelhantes às estruturas dos sub circuitos. No nosso caso inicial estamos procurando diferentes arquiteturas de um espelho de corrente. O teste é feito levando em consideração os tipos de nós (se são nós centrais do tipo C,R ou T, e os pesos das arestas de conexões entre os terminais de cada elemento. Sendo assim, por exemplo, um transistor conectado a um resistor através do seu terminal *gate* só encontrará correspondência no CV se lá existir um resistor conectado ao terminal *gate* de um transistor.

3.5 Simplificação de estruturas

Ao se carregar o netlist de um circuito, algumas estruturas podem ser associadas de forma a se promover a simplificação dos circuitos e, consequentemente, facilitar a tarefa do sistema em identificar sub-circuitos. Para exemplificar esta situação, imagine que a Figura 11a mostra uma arquitetura simples de um espelho de corrente a ser encontrado em um circuito maior (que está sendo verificado). A Figura 11b mostra a representação desta arquitetura por um grafo.



(a) Exemplo de arquitetura de um espelho de corrente.

(b) Representação do circuito da Figura 11a em forma de grafo.

Figure 11: Representações de um espelho de corrente.

Se o sistema está tentando identificar um circuito *espelho de corrente* com a arquitetura como especificada na Figura 11a, em um circuito maior, representado pela Figura 12, ele pode indicar a existência de dois possíveis subcircuitos que representam um espelho de corrente. Esses possíveis subcircuitos estão representados nos grafos da Figura 13 e ocorrem por causa dos resistores R_1 e R_2 em paralelo. Sendo assim, a ferramenta proposta deve ser capaz, se possível, de fazer simplificações nos circuitos de maneira a eliminar ou minimizar este tipo de problema onde a ferramenta deveria arbitrar qual dos dois grafos realmente representa um espelho de corrente. As simplificações mais simples de se fazer são as associações série e paralelo de resistores e capacitores do circuito. Para isso desenvolvemos uma heurística baseada no isomorfismo de grafos para a identificação inicialmente de resistores e capacitores em série ou paralelo.

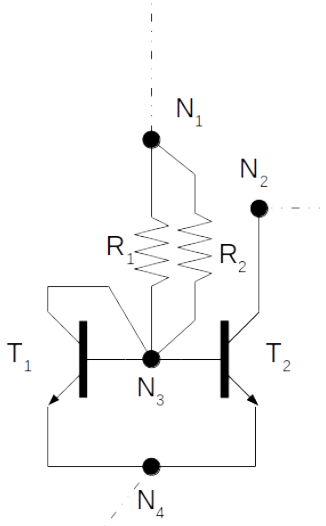
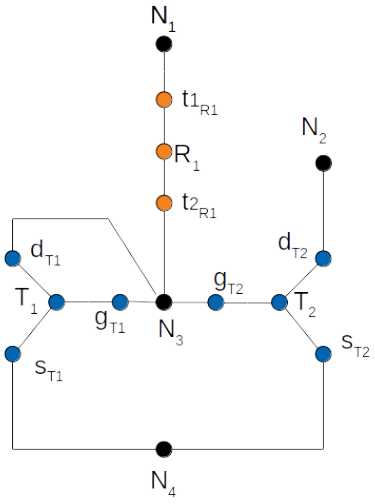
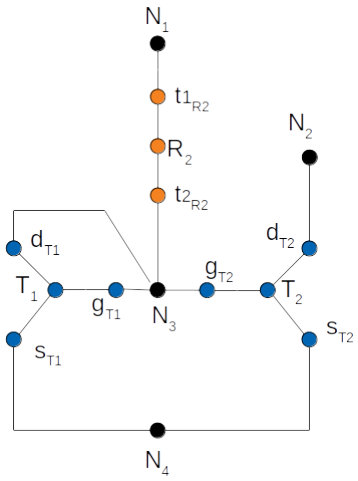


Figure 12: Representação de parte de um circuito maior. As linhas tracejadas indicam a continuação do circuito.



(a) Primeira representação possível de um espelho de corrente.

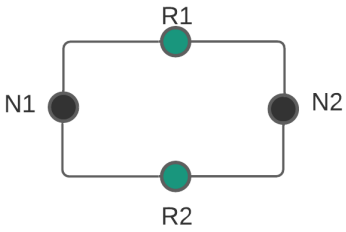


(b) Segunda representação possível de um espelho de corrente.

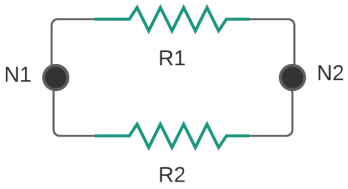
Figure 13: Possíveis circuitos de espelho de corrente no circuito da Figura 12.

3.6 método se simplificação

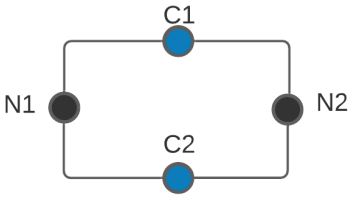
Basicamente são definidas grafos padrão exemplificados pelas figuras 14 de capacitores e resistores em série e paralelo, que são comparadas à estrutura do grafo do circuito que se quer simplificar.



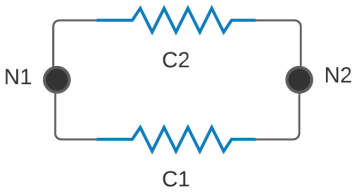
(a) Estrutura de resistência em paralelo representada em grafo.



(b) Resistência em paralelo



(c) Estrutura de capacitores em paralelo representada em grafo.



(d) Capacitores em paralelo.

(e) Estrutura de resistência em paralelo representada em grafo.

(f) Resistência em série

(g) Estrutura de capacitores em paralelo representada em grafo.

(h) Capacitores em série.

Figure 14: Representação estruturas para simplificação.

Encontrando essas estruturas, elas se tornaram candidatas a serem simplificadas no grafo da *netlist*. As candidatas para simplificação em paralelo sejam elas entre resistores ou capacitores podem ser imediatamente simplifcadas, mas caso elas sejam candidatas a simplificação sejam em série, elas precisam passar por uma segunda análise que é a checagem do nó central, ou seja o nó que interliga as estrutras, se o mesmo não possuir nenhuma aresta extra poderá ser simplicado,

3.7 Busca por espelho de correntes

Utilizando a teória de descobrimento de estruturas tratado no tópico 3.2, neste tópico irá ser mostrado o passo a passo de como se busca estrutura de espelhos de corrente:
1) Circuito é lido da netlist gerando um grafo, os pesos de cada nó desse grafo vária de acordo com os tipos de elementos: resistores, trasmissores e capacitores.
2) Subcircuito, no caso estruturas de espelho de corrente, são lidas gerando subgrafos, cujos os nós também terão pesos.
3) Uma vez modelado os circuitos e subcircuitos em grafo é utilizado o algoritmo VF2 [6, 7], ele irá percorrer o Grafo do circuito procurando os subcircuitos de espelhos de correntes desejados

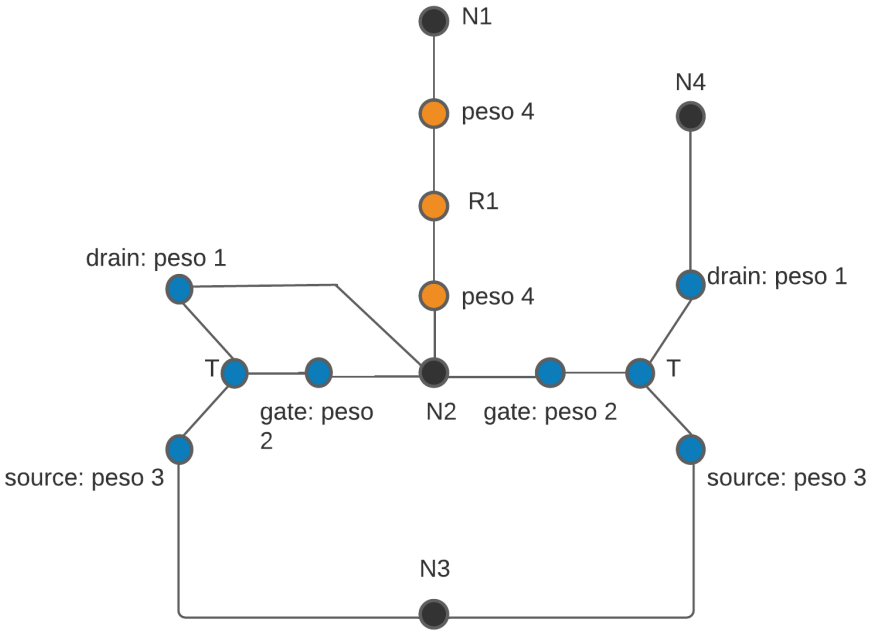


Figure 15: Exemplo de espelho de corrente representado em grafo com seus respectivos pesos

Com a figura 15 fica mais fácil exemplificar o passo 3, o algoritmo VF2 [6, 7] recebe como entrada o grafo do circuito todo, junto a um subgrafo de espelho de corrente no caso estamos pegando como exemplo o da figura 15, sendo assim o algoritmo começa a procurar está estrutura do espelho de corrente dentro do circuito levando em consideração as estruturas que são isomorfas, isso se torna possível principalmente pelo fato de cada nó possuir um peso que caracteriza o tipo de estrutura por exemplo se um nó com peso 2 estiver conectado a um nó peso 4, o algoritmo consegue saber que existe um resistor qualquer conectado a

um gate de um transistor

4 Experiments

4.1 Circuit simplification and CM identification

In this first experiment 20 circuits with near 50 elements each were generated containing 2 CMs maximum.

Table 1 shows the description os the generated circuits where #E is the number of elements in the circuit, #CM(T) specifyies the number of CM in the circuit and the types inside parentheses, #PR is the number of resistors conected in parallel, #SR is the number of resistors conected in series, #PC is the number of capacitors conected in parallel, and, #SC is the number of capacitors conected in series.

Table 1: Circuits description for experiment 1

Circuit	#E	#CM(T)	#PR	#SR	#PC	#SC
0	50	1(2)/1(4)				
1	30	1(1)/1(3)				

Pedir ao Evandro para Conferir esta tabela visualmente

References

[1] Kenneth C. Smith Adel S. Sedra, editor. *Microeletrônica*. Pearson Prentice Hall, 5th edition, 2007.

[2] Haithem Ayari, Florence Azais, Serge Bernard, Mariane Comte, Vincent Kerzerho, Olivier Potin, and Michel Renovell. Making predictive analog/RF alternate test strategy independent of training set size. *Proceedings - International Test Conference*, pages 1–9, 2012.

[3] Manuel J. Barragan and Gildas Leger. A procedure for alternate test feature design and selection. *IEEE Design and Test*, 32(1):18–25, 2015.

[4] Jeffrey H. Dinitz Charles J. Colbourn, editor. *Handbook of Combinatorial Designs*, volume 2, chapter 6, page 777. CRC Press, 2006.

[5] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[6] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004.

[7] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159, 2001.

[8] H. El Badawi, F. Azais, S. Bernard, M. Comte, V. Kerzerho, and F. Lefevre. Use of ensemble methods for indirect test of RF circuits: Can it bring benefits? *LATS 2019 - 20th IEEE Latin American Test Symposium*, (1), 2019.

[9] P Feofiloff, Y Kohayakawa, and Y Wakabayashi. Uma Introdução Sucinta à Teoria dos Grafos. Technical report, 2009.

[10] Adauto Luis Tadeo Bernardes FONSECA. Metodologia de Verificação Funcional para Circuitos Analógicos. page 119, 2009.

[11] Karam Gouda and Mosab Hassaan. A fast algorithm for subgraph search problem. 01 2012.

[12] Paul R. Gray. *Analysis and Design of Analog Integrated Circuits*. Wiley Publishing, 5th edition, 2009.

[13] Randy Holt. Schematic vs. netlist: A guide to pcb design integration.

[14] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[15] Aviral Mittal. Netlist.

[16] Andrew Piziali. *Functional Verification Coverage Measurement and Analysis*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[17] Haralampos-G. Stratigopoulos. Machine learning applications in IC testing. In *2018 IEEE 23rd European Test Symposium (ETS)*, volume 2018-May, pages 1–10. IEEE, may 2018.