TRAÇADO COMPLETO DE ROTAS EM PLACAS DE DIMENSÕES NÃO PRÉ-FIXADAS

HANS KURT EDMUND LIESENBERG

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO

CAMPINAS - SÃO PAULO
BRASIL

TRAÇADO COMPLETO DE ROTAS EM PLACAS DE DIMENSÕES NÃO PRÉ-FIXADAS

HANS KURT EDMUND LIESENBERG

Orientador

Prof.Dr. Nelson Castro Machado

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Outubro - 1980

UNICAMP

ABSTRACT

This paper presents a routing algorithm for printed circuit boards whose input is a description of the relative position of the components instead of the specific location of each component. Using two-layer boards with non-fixed dimensions, the algorithm always defines all the desired connections. It is also able to accept any component whose pins are on the boundary of a rectangle. The algorithm is non-cellular; the connections are established macroscopically on a grid with a small number of rows and columns, where each component corresponds to an intersection. This approach yields great savings in memory space and computing time. The spacing between components is initially assumed unlimited, capable of supporting any number of connections. The actual number of connections in the macroscopic cells is determined by the algorithm and only then the final location of each component is established, thus defining the dimensions of the board.

SUMÁRIO

1 - INTRODUÇÃO

Na primeira parte desta introdução é situado o problema de traçado de rotas. A segunda parte é dedicada à revisão bibliográfica e na terceira são delineadas as idéias básicas do algoritmo proposto.

1.1 - TRAÇADO DE ROTAS PARA CIRCUITO IMPRESSO

Nas últimas décadas a construção de equipamentos eletrônicos sofisticados caracterizou-se pela introdução e disseminação de circuitos impressos. Esta tecnologia consiste em utilizar uma base isolante(placa) na qual são fixados os componentes eletrônicos. A fixação dos componentes é feita através do alojamento de seus pinos em furos nesta base isolante. Tais furos estão revestidos de material condutor e por isso recebem o nome de furos metalizados. Chama-se de sinal a um conjunto de pinos eletricamente equivalentes, isto é, interligados por condutores ou "fios". Na tecnologia de circuito impressso esta interligação não é estabelecida por fios convencionais, mas por depósitos de material condutor sobre a base isolante. Obviamente, os condutores correspondentes a sinais distintos não podem se cruzar em uma mesma face da / placa ou

resultaria um "curto" entre estes sinais. Como nem sempre todas as interligações podem ser estabelecidas usando somente uma face da placa, os condutores são normalmente distribuidos em duas ou mais faces evitando-se assim os "cruzamentos". Denomina-se de"placa de dupla face": uma placa em que ambas as faces são utilizadas para deposição de condutores. Em alguns casos, em que se deseja alta densidade de componentes na placa, recorre-se às chamadas placas multi-face ou multi-camada. Nestas, utiliza-se um "sanduiche" de camadas isolantes entre as quais encontram-se as interligações. Neste caso, denomina-se "face" não apenas as duas faces externas, mas também a região entre duas camadas. Interligações que passam de uma face para outra, o fazem através de furos matalizados que garantem a continuidade elétrica do condutor. Assim sendo, furos metalizados podem se tornar necessários não apenas para alojar pinos de componentes, mas também, às vezes, simplesmente para permitir a um condutor passar de uma face a outra. (Fig. 1.1)

O problema do traçado de rotas consiste / em, especificadas as interligações desejadas entre pinos dos componentes e a localização destes na placa, determinar a distribuição espacial dos condutores respeitadas as restrições impostas / pela tecnologia de circuito impresso.

Um problema relacionado ao traçado é o da localização dos componentes que consiste em determinar a posição

de cada componente na placa de maneira a facilitar ou otimizar o posterior traçado de rotas.

No capítulo 2 é feita uma abstração para/ definir de maneira mais formal os conceitos envolvidos.

O surgimento dos circuitos integrados (componentes eletrônicos caracterizados por pequenas dimensões e grande número de pinos) provocou um aumento muito grande na densidade de conexões entre os componentes de uma placa de circuito impresso.O projeto manual detraçado de uma destas placas é uma tarefa / que pode passar de trivial a quase impossível à medida que o número de conexões aumenta. É perfeitamente normal encontrar-se hoje em dia, principalmente em computadores digitais, placas de duas ou mais faces com 50 a 100 circuitos integrados.O projeto manual pode levar até diversas semanas para ser completado, envolvendo um alto custo, o que justifica a necessidade de automação . Também as tarefas de documentação, depuração e introdução de modificações tornam-se mais rápidas e menos onerosas se se dispõe / de um sistema computarizado para auxiliar no projeto de placas de circuito impresso.

O traçado por computador não é uma atividade nova; investiga-se nesta área desde 1961(7) e já se dispõe de vasta bibliografia sobre o assunto, além de sistemas industriais, extremamente sofisticados, em operação. A quase totalidade dos / fabricantes de equipamentos digitais, projeta suas placas utilizando sistemas computarizados. Há, entretanto, um considerável /

vazio entre os métodos publicados e os sistemas industriais. A bibliografia normalmente descreve algoritmos de interesse teórico , mas cuja aplicação prática deixa muito a desejar em termos de eficiência ou devido às restrições ou generalizações que não podem / ser obedecidas em placas reais. Por outro lado, os sistemas industriais tem grande valor econômico e, quando não de uso privativo/ de um determinado fabricante, somente são disponíveis a nível de "caixas pretas" ou pacotes fechados sem nenhum detalhamento sobre a operação e estrutura dos métodos utilizados. Além disto, estes sistemas foram projetados para aplicações em indústrias desenvolvidas, altamente automatizadas e com grande volume de produção, o que, até certo ponto conflita com as necessidades de indústrias e universidades brasileiras em que as condições são exatamente opostas . Outra desvantagem de tais sistemas comercialmente disponíveis,além do alto custo, prende-se ao fato de que são normalmente fornecidos como parte de um conjunto que inclue necessariamente o "hardware" forçando a importação talvez desnecessária de computadores.

Na maioria dos algoritmos publicados, cada face é representada por um reticulado de passo fixo, isto é, um plano é dividido por um conjunto de retas verticais e horizontais com distância constante entre retas paralelas adjacentes. Chama-se de passo a esta distância constante e de célula ao menor / quadrado definido pelo reticulado, cujo lado mede um passo. O pas

so do reticulado é escolhido de maneira que na área de uma célula caiba um furo metalizado ou um segmento de conexão. Chama-se ainda de caminho entre duas células dadas denominadas extremos, a um conjunto de células adjacentes, ainda não comprometidas com outras interligações, que inclue os dois extremos e tal que cada célula/ não extremo é adjacente a no máximo duas outras célunas não extremo . A distância de um caminho é dada pelo número de células que o compõe. O problema de traçado de rotas reduz-se pois (no caso/ de placas de uma única face) a associar a cada par de pinos a ser interligado um par de células extremo e determinar um caminho entre estes dois extremos. Determinado este caminho, é trivial traçar um condutor que implementa a interligação desejada conforme ilustrado na Figura 1.2.

O algoritmo de LEE(7) publicado em 1961 é considerado o algoritmo clássico nesta área. Foi definido para / placas de uma face e interliga os pinos numa ordem pré-estabelecida. O algoritmo garante que: a) Se existem caminhos entre duas células do reticulado para uma dada configuração da placa, então um destes caminhos é encontrado; b) O caminho encontrado tem distância mínima. Segue-se uma rápida descrição do algoritmo.

Uma célula extremo é considerada origem e outra destino. A célula origem recebe o rótulo zero. As células / adjacentes a estas, não comprometidas e não rotuladas, são rotuladas com o valor dois. Continua-se o procedimento de rotulação

de maneira análoga até que não seja mais possível nenhuma rotulação ou até que a célula destino seja rotulada. Se a célula destino foi rotulada, então existe caminho mínimo e um dos tais caminhos pode ser determinado a partir da célula destino: Esta pertence ao cami - nho mínimo e é comprometida. Seja c a última célula comprometida / do caminho mínimo e r o seu rótulo. Escolhe-se então uma célula adjacente a c com rótulo r-1. Por construção, pelo menos uma destas / células sempre existe. Esta célula passa a ser a última célula determinada do caminho mínimo e é comprometida. Repete-se o procedimento até que a célula origem também seja comprometida (Fig.1.3). A vantagem deste algoritmo é sua simplicidade, sendo por este motivo usado em estudos teóricos como de AGRAWAL(8). Uma desvantagem,en - tre outras, é que, sob o aspecto computacional ele é ineficiente / tanto em espaço como em tempo. Como uma face é utilizada, uma solução sem insucessos (interligação não resolvida) raramente é obtida e correções manuais se tornam muito difíceis pois implicam na remoção de grande parte das conexões já concluídas. Uma extensão deste algoritmo para mais de uma face foi publicado por GEYER(9). Os algoritmos (9,10,11),que usam a mesma filosofia do algoritmo de LEE,são chamados celulares ou de labirinto.

Os algoritmos de traçados de rotas podem / ser calssificados segundo o seguinte critério:

traçado { em série { de labirinto / heurísticos ; em paralelo }

A característica dos algoritmos série é a resolução das conexões numa ordem pré-determinada.

Os algoritmos heurísticos (12,13,14,15) foram propostos visando melhorar a eficiência dos algoritmos de labirinto. Infelizmente, o aumento da eficiência é normalmente acompanhado pela perda da propriedade exibida pelo algoritmo de LEE de sempre encontrar um caminho quando o mesmo existe. A estratégia usada pelos algoritmos série heurísticos consiste basicamente no traçado de retas, a partir dos pontos a serem interligados e na tentativa de encontrar uma intersecção entre tais retas ou diretamente ou indiretamente através de retas auxiliares. Normalmente o número de tentativas para encontrar a intersecção é limitado. Esta técnica é exemplificada através de uma descrição sumária do algoritmo de HIGHTOWER(12).

Por cada ponto de um par a ser interligado é traçado um segmento de reta vertical ou horizontal. Os dois segmentos assim traçados são escolhidos de maneira que sejam perpendiculares entre si e que não cruzem uma conexão já resolvida ou pino. Se os dois segmentos se interceptarem, então a interligação destes dois pontos está determinada. Caso contrário, são traçadas perpendiculares (linhas de escape) aos segmentos já traçados e procura-se uma intersecção das últimas. O processo pode ser continuado traçando linhas de escape às anteriores até que uma intersecção seja encontrada ou até que não seja mais possível traçá-las.

Note-se que estes algoritmos adaptam-se naturalmente a placas de dupla face colocando-se os segmentos verticais em uma face e os horizontais na outra. Desta maneira evitam-se cruzamentos indesejáveis. A intersecção entre um segmento horizontal e um vertical é obtida, quando desejada, através de um furo metalizado.

Normalmente um algoritmo heurístico resolve 85% a 99% das conexões. As restantes são resolvidas manualmente.

Os algoritmos série resolvem as conexões / de maneira sequencial otimizando sempre a conexão. que está sendo considerada, independentemente daquelas que ainda devem ser resolvidas. Esta política de ótimo local não leva necessariamente a um ótimo global. Tentou-se contornar tal dificuldade ordenando-se as conexões a serem resolvidas segundo algum critério adequado. Na prática, (16,17) nenhum esquema de ordenação demonstrou ser satisfatório. Foi também proposta (8) a utilização de técnicas de retrocesso ("backtracking") para encontrar uma solução mais satisfatória.

Nos algoritmos paralelos as conexões são planejadas e ajustadas conjuntamente de modo a interferirem entre si o mínimo possível. Esta classe de algoritmos é em tese a ideal mas ainda não foi descrita na literatura uma implementação real / bem sucedida de um algoritmo paralelo de traçado de rotas.

1.2 - CARACTERÍSTICAS GERAIS DO ALGORITMO PROPOSTO

Os algoritmos tradicionais partem de uma prévia distribuição dos componentes sobre a placa (isto é: supõe previamente resolvido o problema de localização). Consequentemente, em tais algoritmos a dificuldade de traçar uma conexão é diretamente proporcional à densidade das conexões já traçadas. Isto torna as últimas conexões a serem resolvidas mais longas e até excessivamente longas quando não resulta em insucesso devido basicamente à falta de espaço na placa para alocar condutores.

A localização pré-fixada justifica-se quando os componentes são introduzidos na placa por máquinas automáticas (inserção automática). Isto, porém, só ocorre em linhas de produção automatizadas, o que não é a regra no país. Todos os algoritmos descritos na bibliografia apresentada supõem os componentes numa localização pré-fixada e não foi encontrada referência a algoritmos que não a utilizam.

Uma vantagem em não se utilizar localizações pré-fixadas está colocada na secção 3.3. É sempre possível encontrar uma solução sem insucessos numa placa de duas faces sem dimensões pré-fixadas. Torna-se desnecessária portanto, a fase de correção manual do traçado obtido automáticamente. Para garantir uma solução sem insucessos é necessário ainda impor que

os componentes tenham pinos somente nas bordas (ver secção 2.2 para uma definição rigorosa). Esta não é uma restrição indesejável, de vez que a quase totalidade dos componentes disponíveis no mercado a satisfazem.

A principal característica do algoritmo / proposto consiste em partir de uma relação cardeal de vizinhança entre os componentes (exemplo: o componente A é vizinho norte do componente B). As componentes formam pois uma rede retangular. Entre duas linhas ou colunas de componentes existem canais de capacidade não limitada para comportar conexões. Estes canais são chamados de canais comuns. Distingue-se ainda, uma segunda classe de canais, chamados canais próprios que interligam componentes adjacentes.

Esta situação pode ser convenientemente representada por um reticulado(Figura 1.4) chamado de macro-reticulado. As intersecções de retas do reticulado correspondem a componentes e intersecções de canais. Os lados do reticulado correspondem a segmentos de canais entre intersecções ou entre intersecção e componente. Só é determinado um espaço físico para cada conexão depois que é conhecido o número definitivo de conexões que a aresta vai comportar. Assim sendo, cada componente(ou melhor , cada linha e coluna de componentes) se afasta dos adjacentes na medida do necessário para acomodar todas as ligações desejadas. Ou seja, a placa se "expande" garantindo assim, uma solução com -

pleta do problema do traçado, sem insucessos. O preço a pagar por esta garantia de solução completa é, obviamente, a não garantia de dimensões pré-fixadas para a placa.

O algoritmo poderá ser considerado bem sucedido na prática se a expansão da placa for moderada, resultando em densidade de componentes semelhantes às obtidas com algoritmos tradicionais. Note-se que, se a heurística do algoritmo for adequada é mesmo possível obter altas densidades pois a placa só se expande na medida do necessário ao passo que nos algoritmos tradicionais / parte-se de componentes espaçados arbitrariamente de uma distância considerada adequada para reduzir os insucessos a um número aceitável.

Foi adotado uma linguagem informal e auto - explicativa do tipo ALGOL com passagem de parâmetro por referência para descrever de maneira independente de estruturas de dados específicas os algoritmos utilizados em todo o trabalho. São utilizados símbolos em português e matemáticos. Algumas partes são descritas em portugues corrente quando a descrição algorítmica é particularmente dependente da estrutura de dados escolhida para a implementação.

2.- NOMENCLATURA, DEFINIÇÕES E PROCEDIMENTOS BÁSICOS

Neste capítulo são introduzidos nomenclatura, símbolos e definições utilizados nos capítulos seguintes. Não serão repetidos os conceitos e definições estabelecidos na introdução.

São ainda apresentados alguns conceitos básicos que serão utilizados como primitivas no restante do trabalho.

2.1 - Grafos e Macro-Reticulado

2.1.1 - Nomenclatura e Definições

Seja G um grafo conexo.

V(G): conjunto de vertices de G

A(G): conjunto de arestas de G

Seja u,v ∈ V(G), então

a) (u,v) : aresta com extremos u e v

b) r(u,v): rótulo da aresta com extremos u e v. É ∞ se $\nexists$ (u,v)

c) I(u) : rótulo associado ao vértice u

d) g(u) : grau do vértice u ≡ número de arestas que incidem em u.

Subgrafo H de G $\Leftrightarrow$ V(H) $\subseteq$ V(G) e A(H) $\subseteq$ A(G).

ciclo: sequência não vazia finita $v_0a_1v_1a_2v_2\ldots v_{k-1}a_kv_k$ on - de

$v_i \in V(G)$ $(0 \leq i \leq k)$, $a_i \in A(G)$ $(1 \leq i \leq k)$, os extremos de a_i são v_i e v_{i-1}, $a_i \neq a_j$ $(i \neq j)$, $v_0 = v_k$ e $v_i \neq v_j$ $(i \neq j)$ e $i = 0 \equiv j \neq k$)

caminho: sequência não vazia finita $v_0a_1v_1\ldots v_{k-1}a_kv_k$ tal que $v_i \in V(G)$ $(0 \leq i \leq k)$, $a_i \in A(G)$ $(1 \leq i \leq k)$, os extremos de a_i são v_i e v_{i-1} $(1 \leq i \leq k)$ e $v_i \neq v_j$ $(i \neq j)$.

caminho mínimo entre u e v: o caminho cuja soma dos rótulos das arestas, que fazem parte do caminho, seja mínima em relação a todos os caminhos entre u e v.

d(u,v) : distância do caminho mínimo entre u e v.

um caminho $v_0a_1v_1\ldots a_kv_k$ PERTENCE a um subgrafo H de G se e somente se $\{v_i/0 \leq i \leq k\} \subseteq V(H)$ e $\{a_i/1 \leq i \leq k\} \subseteq A(H)$

sub-árvore T de G: subgrafo de G tal que $\forall t_1, t_2 \in V(T)$ existe um único caminho entre t_1 e t_2.

sub-árvore R(D,r) onde $r \in V(R)$, $D \subset V(R)$ e $r \notin D$ é uma sub-árvore do grafo G tal que, $\forall u \in D$, um caminho mínimo entre r e u em G pertence a R. Diz-se que R é a sub-árvore mínima em relação a raiz r.

grafo representativo do Macro-reticulado: já foi introduzido

no capítulo 1 o conceito de macro-reticulado. Torna-se conveniente,para possibiltar a utilização de representações e algoritmos de grafos no desenvolvimento / do trabalho, fazer corresponder ao macro-reticulado / um grafo, construido da seguinte maneira: enumera-se, no macro-reticulado, os canais verticais da esquerda para a direita a partir de um e os canais horizontais de baixo para cima também a partir de um. Cada intersecção é identificada pelo par [i,j] = [número do canal horizontal, número do canal vertical]. No grafo / representativo há um vértice correspondente a cada intersecção no macro-reticulado, designado pelo mesmo / par [i,j]. A cada segmento de canal entre duas intersecções no macro-reticulado corresponde uma aresta no grafo representativo. (Fig.2.1)

macro-célula de um componente: corresponde, no macro-reticulado, a um conjunto de doze segmentos de canal: os quatro segmentos de canal próprio que incidem no componente em questão e os oito segmentos de canal com que possuem uma intersecção em comum com um dos qua - tro primeiro segmentos(Fig.2.2) Os oito segmentos de canal comum são chamados de contorno da macro-célula.

Daqui em diante, sempre que não houver peri

de ambiguidade, passar-se-á a chamar simplesmente de macro-reticulado não somente o macro-reticulado propriamente dito como também seu grafo representativo.

$A_{\#\gamma}$: conjunto de arestas do subgrafo do macro-reticulado representantes das conexões do sinal γ .

$P_{\#\gamma}$: conjunto dos vértices do macro-reticulado que são extremos das arestas em $A_{\#\gamma}$.

H_i: conjunto de todas as arestas correspondentes a segmentos do canal horizontal i.

R_m: número de linhas de componentes no macro-reticulado, que corresponde ao número de linhas de macro-células.

R_n: número de colunas de componentes no macro-reticulado, que corresponde ao número de colunas de macro-células.

C : conjunto de vértices no grafo representativo do macro-reticulado correspondentes a componentes.

2.1.2 - PROCEDIMENTOS BÁSICOS

extf(c,b): função definida sobre o grafo representativo do macro-reticulado que retorna um vértice, dados um vértice c correspondente a um componente e b ϵ {0,1,2,3}. O vértice retornado é a outra extremidade da aresta / que incide no vértice c e que corresponde ao segmento de canal próprio que, no macro-reticulado, incide no

componente pela direção definida pela tabela abaixo:

b	direção
0	baixo para cima
1	direita para a esquerda
2	cima para baixo
3	esquerda para a direita

Uma ilustração da função é dada pela figura 2.3. A função é implementada pelo seguinte algoritmo:

```
início (* c é identificado por[c1,c2]*)
 extf: = se b = 2
   então [c1+1,c2]
   senão  se b = 1
    então  [c1,c2+1]
    senão se b = 0
      então [c1-1,c2]
      senão [c1,c2-1]
fim
```

extah(v,c): função definida sobre o grafo representativo do macro-reticulado que retorna um vértice, dados um vértice c correspondente a um componente e um vértice v correspondente a uma intersecção localizada sobre o contorno da macro-célula de c. O vértice retornado é o correspondente à primeira intersecção encon

trada quando se percorre o contorno da macro-célula do componente no sentido anti-horário a partir da intersecção correspondente a v. (Fig.2.4) A função é implementada pelo seguinte algoritmo:

```
início (* v é identificado por [v1,v2]
          e c é identificado por [c1,c2]*)
 exthab:=se v1 = c1
    então (se v2 < c2
      então [v1-1,v2]
      senão [v1+1,v2] )
    senão se v1 > c1
      então ( se v2 < c2
        então [v1-1,v2]
        senão [v1,v2-1] )
      senão se v2 > c2
        então [v1+1,v2]
        senão [v1,v2+1]
fim
```

seg(a,A): função definida sobre o grafo representativo do macro-reticulado que retorna uma aresta, dados um conjunto de arestas A correspondentes ao conjunto de segmentos de um canal no macro-reticulado e uma aresta a ∈ A. A aresta retornada corresponde ao segmento de canal sucessor do segmento de canal correspondente a a.

Se não há sucessor, a função retorna a. Por sucessor de um segmento de canal entende-se o próximo segmento do mesmo canal quando se percorre o canal da / esquerda para a direita, se horizontal ou de baixo / para cima, se vertical (Fig.2.5) A função é implementada pelo seguinte algoritmo:

início

(* seja a=($[a_{11},a_{12}]$, $[a_{21},a_{22}]$) tal que
$a_{11} \leq a_{21}$ e $a_{12} \leq a_{22}$ *)

seg: = se $a_{11} = a_{21}$

então (se $\exists$ ($[a_{21},a_{22}]$, $[a_{21},a_{22}+1]$)

então ($[a_{21},a_{22}]$, $[a_{21},a_{22}+1]$)

senão a)

senão se $\exists$ ($[a_{21},a_{22}]$, $[a_{21}+1, a_{22}]$)

então ($[a_{21},a_{22}]$, $[a_{21}+1,a_{22}]$)

senão a

fim

primh(i) : função definida sobre o grafo representativo do macro-reticulado que retornava uma aresta dado o número de um canal horizontal. A aresta retornada corresponde ao segmento de canal mais à esquerda do canal horizontal i. Na figura 2.5 primh(3) = a_1. A fun

ção é implementada pelo seguinte algoritmo:

início

primh: = ([i,1] , [i,2])

fim

menorh(a,b): função lógica definida sobre o grafo representativo do macro-reticulado que tem como parâmetros duas arestas correspondentes a segmentos de um mesmo canal e retorna verdadeiro se e somente se o segmento correspondente a a estiver à esquerda do segmento correspondente a b. Segue-se o algoritmo que implementa a função:

início

(* a = ([k,a_1] , [k,a_2])

b = ([k,b_1] , [k,b_2]) *)

menorh:= $a_2 < b_2$

fim

ares(c,b): função definida sobre o grafo representativo do macro-reticulado que retorna uma aresta, dados um vértice c correspondente a um componente e b ϵ {0,1,2,3}. A aresta retornada incide no vértice e corresponde ao segmento de canal próprio que, no macro-reticulado incide no componente pela direção definida na tabela descrita na função extf. Uma ilustração da fun-

ção é dada pela figura 2.3. A função é implementada pelo seguinte algoritmo:

início

(* c =$[c_1,c_2]$ *)

ares:= se b = 2

então ($[c_1,c_2]$, $[c_1+1,c_2]$)

senão se b = 1

então ($[c_1,c_2]$, $[c_1,c_2+1]$)

senão se b = 0

então ($[c_1,c_2]$, $[c_1-1,c_2]$)

senão ($[c_1,c_2]$, $[c_1,c_2-1]$)

fim

det.bor(c,v): função definida sobre o grafo representativo / do macro-reticulado que retorna b ε {0,1,2,3} , dados um vértice c representante de componente é um vértice v que é extremo de uma das arestas incidentes em c. O valor de b depende da direção do segmento do macro-reticulado que corresponde a aresta (c,v) de acordo com a tabela seguinte:

direção	b
baixo para cima	0
direita para a esquerda	1
cima para baixo	3
esquerda para a direita	3

Uma ilustração da função é dada pela figura 2.6. A função é implementada pelo seguinte algoritmo:

início

(* c = $[c_1 , c_2]$

v = $[v_1 , v_2]$ *)

det.bor: = se $c_1 = v_1$

então (se $c_2 < v_2$ então 3 senão 1)

senão se $c_1 < v_1$ então 2 senão 0

fim

2.2 - MINI-RETICULADO E ESTRUTURA

2.2.1 - NOMENCLATURA E DEFINIÇÕES

As faces das placas são identificadas pelos números 0 e 1 (superior e inferior respectivamente). Nas definições abaixo k identifica uma face (k ε {o,1}) e c representa um determinado componente.

MR^c: mini-reticulado de c que é composto de dois reticulados um para cada face (Fig.2.7).

Os exemplos abaixo referem-se a figura 2.7.

$MR^{c,k}$: reticulado de c na face k.

Exemplo:

$MR^{c,0} = \{k_i / 1 \leq i \leq 12\}$ e $MR^{c,1} = \{k_i / 13 \leq i \leq 24\}$

$\underline{MR^{c,k}_{x,y}}$: célula do reticulado c na face k de coordenadas x e y, com relação a um sistema de coordenadas em que a célula de coordenadas 1,1 está a sudoeste, o eixo / dos x tem sentido norte e o eixo dos y tem sentido / leste.

Exemplo:

$M^{c,l}_{2,3} = k_{19}$

$\underline{m^c}$: número de linhas dos reticulados de c. Na figura $m^c = 3$.

$\underline{n^c}$: número de colunas dos reticulados de c.Na figura $m^c = 4$.

$\underline{\text{Seja}}$ $B^{c,k}_o = \{MR^{c,k}_{i,j} / i = 1,\ 1 \leq j \leq n^c\}$ = borda sul do componente c na face k;

$B^{c,k}_1 = \{MR^{c,k}_{i,j} / 1 \leq i \leq m^c,\ j = n^c\}$ = borda leste do componente c na face k;

$B^{c,k}_2 = \{MR^{c,k}_{i,j} / i = m^c,\ 1 \leq j \leq n^c\}$ = borda norte do componente c na face k;

$B^{c,k}_3 = \{MR^{c,k}_{i,j} / 1 \leq i \leq m^c,\ j = 1\}$ = borda oeste do componente c na face k

então

i) $B^c_i = B^{c,0}_i \cup B^{c,1}_i$ = borda i do componente c ;

ii) $B^{c,k} = B^{c,k}_0 \cup B^{c,k}_1 \cup B^{c,k}_2 \cup B^{c,k}_3$ = borda do componente c na face k

iii) $B^c = B^{c,0} \cup B^{c,1}$ = estrutura do componente

Exemplo :

$$B_0^{c,0} = \{k_1, k_2, k_3, k_4\}$$

$$B_2^{c,1} = \{k_{21}, k_{22}, k_{23}, k_{24}\}$$

$$B_3^{c} = \{k_1, K_5, k_9, k_{13}, k_{17}, k_{21}\}$$

Seja b',b" ϵ $B^{c,k}$, então b' e b" determinam dois subconjun - tos $S^{c,k}_{b',b"}$ e $S^{c,k}_{b",b'}$ de $B^{c,k}$. Ao conjunto $S^{c,k}_{b_1,b_2}$ pertencem b_1 e b_2 e todas as células de $B^{c,k}$ que são encontrados quando se percorre a borda de c, pela face k ,de b_1 até b_2 no sentido anti-horário.

Se b' = b", então $S^{c,k}_{b',b"} = S^{c,k}_{b",b'} = B^{c,k}$

Exemplo:

$$S^{c,0}_{k_3,k_{12}} = \{k_3, k_4, k_8, k_{12}\}$$

$$S^{c,0}_{k_{12},k_3} = \{k_{12}, k_{11}, k_{10}, k_9, k_5, k_1, k_2, k_3\}$$

As células da borda são enumeradas da seguinte maneira

i) $MR^{c,k}_{1,1}$ é a célula 0 ;

ii) a célula designada pelo número i+1 é a célula que sucede (adjacente) a célula de designação i no / sentido anti-horário.

Seja a = $MR^{c,k}_{i_1,j_1}$ e b = $M^{c,k}_{i_2,j_2}$ então $d(a,b)=|i_1-i_2|+|j_1-j_2|=$

= distância (de Manhattan) entre a e b

As células de B^c pertencem a um de dois subconjuntos chamados P^c e Q^c. $P^c \cap Q^c = \emptyset$ e $P^c \cup Q^c = B^c$

conjuntos de pinos $P^c = P^{c,0} \cup P^{c,1}$

$P_i^{c,k} \subseteq B_i^{c,k}$ $(0 \leq i \leq 3)$

$MR_{i,j}^{c,0} \in P^{c,0} \Leftrightarrow M_{i,j}^{c,1} \in P^{c,1}$

Se $p = \{M_{i,j}^{c,0}, M_{i,j}^{c,1}\} \subseteq P^c$ então diz-se que p é um pino do componente c.

Conjunto de passagens $Q^c = Q^{c,0} \cup Q^{c,1}$.

$Q_i^{c,k} \subseteq B_i^{c,k}$ $(0 \leq i \leq 3)$

(Fig. 2.8)

$W^{c,k}$: $\{\{b_1,b_2\}/b_1,b_2 \in B^{c,k}\}$ é um conjunto de vínculos se e somente se

i) $\forall \{x,y\} \in W^{c,k}$ $x \neq y$

ii) $\{b_1,b_2\} \in W^{c,k} \Rightarrow \forall\{x,y\} \in W^{c,k}$ ou $x,y \in S_{b_1,b_2}^{c,k}$ ou $x,y \in S_{b_2,b_1}^{c,k}$

iii) $\{x,y\} \in W^{c,k}$ e $x \in Q^{c,k} \Rightarrow \nexists \{b_1,b_2\} \in W^{c,k}/$ $(b_1 = x$ e $b_2 \neq y)$ ou $(b_2 = x$ e $b_1 \neq y)$

A definição de um conjunto de vínculos está ilustrada na figura 2.9. Note-se que se cada par de células de um conjunto de vínculos for unido por um segmento de reta, então não / há cruzamentos de retas e dois ou mais segmentos diferentes somente tem extremo comum se este extremo for um pino.

$W^c = W^{c,0} \cup W^{c,1}$

$W^c = K^c \cup L^c$ onde $K^c \cap L^c = \emptyset$

$K^c = K^{c,0} \cup K^{c,1}$ = conjunto de vínculos provisórios ($K^{c,k} \subseteq W^{c,k}$)

$L^c = L^{c,0} \cup L^{c,1}$ = conjunto de vínculos não provisórios ($L^{c,k} \subseteq W^{c,k}$)

Um sinal γ designado por $\#\gamma$ é uma lista de pares da forma (componente, pino). Um pino p do componente c é associado a um sinal γ se existe o par (c,p) na lista de $\#\gamma$.

$\sigma^{c,k}_{\gamma,b}$: conjunto de células na borda b e face k do componente c associado a γ.

$\sigma^{c}_{\gamma,b} = \sigma^{c,0}_{\gamma,b} \cup \sigma^{c,1}_{\gamma,b}$ = conjunto de pinos na borda b do componente c associados a γ

$\sigma^{c}_{\gamma} = \sigma^{c}_{\gamma,0} \cup \sigma^{c}_{\gamma,1} \cup \sigma^{c}_{\gamma,2} \cup \sigma^{c}_{\gamma,3}$ = conjunto de pinos do componente c associados a γ.

Um componente é acessível por uma borda b para um sinal $\#\gamma$ se e somente se ou $\sigma^{c,j}_{\gamma,b} \neq \emptyset$ ou $\{x,y\} \in K^{c,j} \cup L^{c,j}$ tal que $x \in \sigma_\gamma$ e $y \in Q^{c,j}_b$ onde $j = (b+1) \bmod 2$. Isto é, o componente é acessível pela borda b se existe um pino pertencente ao sinal associado na mesma borda ou se existe uma passagem na borda b e é possível chegar a um pino associado de outra borda através de um vínculo.

2.2.2 - PROCEDIMENTOS BÁSICOS

hh(x,f,c): função definida para células de borda de um componente c e retorna a célula adjacente a $x \in B^{c,f}$ no sentido horário. (Ver Fig. 2.10)

ah(x,f,c): análogo a hh no sentido anti-horário. (Ver Fig. 2.10)

coord(c,v,x,y): procedimento que retorna em x e y as coordenadas de $v \in B^{c}$.

borda(c,v): função que retorna o número da borda na qual se encontra a célula v do mini-reticulado de c.

pino(c,v): função que retorna vardadeiro se e somente se v é uma célula representativa de pino no mini-reticulado de c.

2.3 - LISTAS

2.3.1 - DEFINIÇÕES

Nos algoritmos são utilizadas listas cuja sintaxe é descrita pela gramática $(\{L,E\}, \{<,>, o\}, P, S)$ onde

P: L → < > | < E>

E → o|o E

O símbolo o está representando qualquer objeto manipulado pelos algoritmos, inclusive listas.

2.3.2 - PROCEDIMENTOS BÁSICOS

Os procedimentos que manipulam listas são:

vazio(I): função que retorna verdadeiro se e somente se a lista I não contém objetos

anexe(o,j) ou acrescente(o,j): procedimento que introduz o objeto o como último objeto de j.

retire (j): função que retorna o último objeto da lista j e retira o mesmo da lista.

primeiro (j): função que retorna o primeiro objeto de j.

segundo (j): função que retorna o segundo objeto de j.

3. - ALGORITMO PROPOSTO

Já foi mencionado em 1.3 que o algoritmo / proposto caracteriza-se por utilizar como entrada uma relação cardeal de vizinhança sobre os componentes. Esta relação determina uma rede de componentes, alguns dos quais, possívelmente vazios, isto é, de dimensão zero. Entre duas fileiras (linhas ou colunas) de componentes existem canais com capacidade ilimitada para comportar conexões. Os canais são de dois tipos: canais prórprios interligando componentes vizinhos e canais comuns entre duas fileiras de componentes.

A rede de componentes e canais formam um reticulado denominado macro-reticulado do qual é derivado um grafo: o grafo representativo (Figs. 1.4 e 2.1).

A primeira fase do algoritmo, descrito na secção 3.1, opera sobre o grafo representativo do macro-reticulado e nele estabelece as conexões de cada sinal na forma de subgrafo. A segunda fase, descrita na secção 3.2, designa espaços físicos na placa para as conexões obtidas na fase anterior.

3.1 - FASE I: Determinação das Conexões

A obtenção das conexões na forma de subgra-

fos baseia-se no algoritmo de DIJKSTRA (11), que determina o comprimento do caminho mínimo entre dois vértices de um grafo a cujas arestas está associado um peso ou custo positivo. As arestas correspondentes a segmentos de canais comuns são rotulados (peso associado) com valores maiores do que as arestas correspondentes a canais próprios a fim de que estes sejam ocupados prioritariamente, sempre que possível. Esta estratégia justifica-se pelo fato de a largura de um canal próprio ser fixa, dependendo somente das dimensões dos componentes interligados pelo canal. Portanto, é preferível ocupar estes canais, do que canais comuns cuja largura é diretamente proporcional a sua ocupação. Uma menor ocupação dos canais comuns resulta pois em placas de dimensões menores.

Conforme foi exposto em 2.2, um componente é representado por um conjunto de dois reticulados (um para cada fase da placa) chamado de mini-reticulado. As conexões no mini-reticulado são chamadas de intraligações e as conexões no macro-reticulado de interligações. Assim sendo, intraligações correspondem a ligações "sob" os componentes e interligações correspondem normalmente a ligações entre componentes distintos.

3.1.1 - INTERLIGADOR

O interligador determina as interligações para cada sinal, ou seja, determina subgrafos no grafo representati

vo do macro-reticulado. Não basta, entretanto, determinar as interligações sem considerar sistematicamente se será possível, em princípio, traçar mais tarde as interligações. Note-se que cada componente possue quatro bordas (norte, sul, leste, oeste) e, no grafo / representativo, incidem quatro arestas em cada vértice correspondente a um componente. Cada uma destas arestas corresponde a uma das quatro direções possíveis para se atingir um componente através de sua borda norte, sul, leste ou oeste. Como cada componente tem uma estrutura fixa, nem sempre é possível atingir qualquer de seus pinos a partir de qualquer borda. Dependendo da estrutura do componente e de quantas interligações já incidiram sobre o componente / em cada borda, é possível que uma, duas ou três bordas estejam totalmente comprometidas, obrigando o interligador a escolher um subgrafo mais "custoso" do ponto de vista das interligações, mas que / resultará numa incidência no componente através de uma borda à qual será possível obter intraligação. Assim sendo, ao interligador / compete também comprometer as estruturas dos componentes de acordo com as arestas do subgrafo incidentes nos vértices correspondentes/ a estes componentes.

Segue-se um esquema macroscópico do algoritmo que implementa o interligador :

início

inicialize o grafo representativo do macro-reticulado e os mini-reticulados;

```
para #γ:= 1 até nS faça (*nS: número de sinais *)
 início
  determine o conjunto D e o vértice raiz u0 ;
  inicialize pesos nos canais próprios ;
  determine um subgrafo
 fim
fim
```

A inicialização do grafo representativo do macro-reticulado consiste na construção do grafo e na atribuição de pesos PCH ou PCV às arestas correspondentes a segmentos de canais comuns horizontais ou verticais respectivamente. PCH e PCV são parâmetros fornecidos pelo usuário. Na inicialização dos mini-reticulados e consequentemente das estruturas as células representativas/ de pinos são diferenciadas das demais e todas são desvinculadas inicialmente ($L^C = \emptyset$). A diferenciação das células é feita atribuindo-se a cada célula representativa de pino o numero do sinal associado ao pino. A células não correspondentes a pinos será associado o número zero.

O conjunto D contém inicialmente todos os vértices representativos de componentes que possuem pinos associados a $\#\gamma$, com excessão de um destes vértices, arbitrariamente escolhido para raiz (u_0) ou vértice origem, a partir do qual será construído o subgrafo das conexões do sinal.

A inicialização dos pesos nas arestas correspondentes a segmentos de canais próprios e a determinação do sub

grafo compreendem quatro algoritmos distintos descritos mais detalhadamente no Apêndice B (algoritmos B1 a B4). B1 é o algoritmo de associação de pesos a canais próprios. Portanto, B1 chama preliminarmente o algoritmo B2 que estabelece, para cada vértice pertencente a $D \cup \{u_0\}$ vínculos provisórios de maneira a tornar acessível o maior número possível de bordas destes componentes. As arestas associadas aos segmentos de canal próprio que incidem numa borda acessível recebem como rótulo o valor 1.0 ou PPA ou PPO: Quando a borda contém um pino do sinal sendo pois trivialmente acessível, é utilizado o peso 1.0; quando a borda não contém um pino do sinal mas é acessível por ligação a pino do sinal localizado em borda adjacente, é utilizado o peso PPA; PPO é o peso utilizado quando a borda / não contém pino do sinal e soménte é acessível para ligação a pino do sinal localizado em borda oposta. PPA e PPO são parâmetros fornecidos pelo usuário. Normalmente PPO > PPA > 1.0 a fim de estabelecer a seguinte ordem de preferência de acesso da interligação ao componente: primeiro, pela borda a que pertence um pino; em seguida por borda adjacente e por último por borda oposta (Fig. 3.1)

As arestas incidentes em vértices correspondentes a componentes que não pertencem a $D \cup \{u_0\}$ ou associadas a segmentos incidentes em bordas não acessíveis recebem como rótulo o valor "infinito", o que garante que nunca serão escolhidas como parte da conexão.

Assim sendo, os únicos segmentos de canais

próprios que podem ser utilizados na determinação do sub-grafo que representa a conexão do sinal são os segmentos que incidem em componentes que tem pinos pertencentes ao sinal por bordas que permitirão (em princípio) intraligação ao pino. Note-se que todo componente, que tem pino pertencente ao sinal fará pelo menos um segmento de canal próprio acessível (aquele correspondente à borda a que pertence o pino, trivialmente acessível) e nunca ficará isolado. (Fig. 3.2)

Depois de determinados o conjunto D, a raiz u_0 e todos os pesos no grafo é determinado o subgrafo das interligações do sinal $\#\gamma$. Como um algoritmo para determinar as interligações baseado em sub-árvores do tipo Steiner é impraticável (definição e demonstração em E1 do Apêndice E) foi construído um algoritmo baseado em outro tipo de sub-árvores descritas em E2. O algoritmo/ B3 minimiza as interligações em relação à sub-árvore em construção. Se durante a construção da sub-árvore uma aresta incidente em vértice representativo de componente é encontrada, então a estrutura deste componente é comprometida (algoritmo B4): se a borda do componente na qual incide o segmento correspondente à aresta se tornou acessível por um vínculo provisório, então, a estrutura é comprometida/ pela transformação do vínculo provisório em não provisório; se a borda se tornou acessível devido a existência de um pino do sinal/ na mesma, então, nenhuma providência precisa ser tomada para comprometer a estrutura. Ao final do algoritmo B3 tem-se em $A_{\#\gamma}$ as ares

tas e em $P_{\#\gamma}$ os vértices da sub-árvore que representa as interligações do sinal γ.

3.1.2. - INTRALIGADOR

O intraligador (algoritmo B5) estabelece, para cada componente c , vínculos para as ligações pino a pino sempre que possível. Estes vínculos são automaticamente não provisórios e, portanto, adicionados a L^c. Em seguida, todos os vínculos não provisórios (vínculos em L^c) são transformados em conexões (intraligações) no mini-reticulado. O intraligador gera ainda duas listas Fl e F2 de falhas das ligações para as quais não foi possível determinar uma conexão no mini-reticulado do componente. Estas falhas são resolvidas como interligações, portanto, no grafo representativo do macro-reticulado, de maneira local em torno do vértice representativo do componente c pelo manipulador de falhas (secção 3.1.3). O algoritmo B5 chama os algoritmos B6 a B11, segundo a árvore ilustrada na figura 3.3.

O procedimento que estabelece vínculos entre pinos de um mesmo componente (B6) parte de uma sequência s crescente em relação a $d_{i,j}$ de n x(n-1)/2 elementos da forma $(p_i, p_j, d_{i,j})$ onde $p_i, p_j \in \sigma_\gamma^{c,o} = \{p_1, p_2, \ldots p_n\}$, $i < j$ e $d_{i,j}$ é a distância de Manhattan entre p_i e p_j. No algoritmo , s_k representa o k-ésimo elemento da sequência s e $s_{k,i}$ ($1 \leq i \leq 3$) a i-ésima coordenada do

k-ésimo elemento de s. Incialmente cada pino é associado a uma classe de equivalência distinta. Uma classe de equivalência é denotada por E((a)) onde a é o representante da classe. Sempre / que entre dois pinos é estabelecido um vínculo, as classes representadas por estes pinos são unidas. O algoritmo percorre a sequência s e para cada elemento s_k procura estabelecer um vínculo entre os dois pinos designados por s_k se tais pinos pertencem a classes/ distintas. A função lógica "possível.ligação.entre" (B7) é utilizada para estabelecer o vínculo. Se no final do procedimento B6 existir mais de uma classe de equivalência, então um pino de cada classe deve ser conectado externamente, isto é, interligado aos demais. As informações para tal são colocadas no conjunto de falhas Fl e posteriormente tratadas pelo manipulador de falhas.

Note-se que o intraligador trabalha com os dois reticulados $MR^{c,0}$ e $MR^{c,1}$, que constituem o MR^c, correspondendo o primeiro à face superior. Os vínculos não provisórios colocados em L^c por B4 já contém uma face estabelecida, função da borda/ pelo qual se procedeu o acesso ao componente: Se o acesso se deu pela borda norte ou sul, o vínculo não provisório se referirá à face inferior; se pela borda leste ou oeste, à face superior . Os vínculos estabelecidos por B6 são entre pinos e portanto podem ser alocados a qualquer das duas faces. Primeiramente B6 tenta estabelecer o vínculo na face superior e em caso de insucesso é tentada a face inferior.

O procedimento "traçado" (B8) transforma - por intermédio do procedimento B9 vínculos em conexões (cami - nhos) no mini-reticulado comprometendo células com os sinais ao qual está associado o ou os pinos do vínculo. Uma célula está com prometida com um sinal quando o número do sinal foi atribuido à célula. Os procedimentos B10 e B11 são procedimentos auxiliares de B9. O algoritmo que estabelece os caminhos (B9) considera que duas células adjacentes possuem um lado ou um vértice em comum.

No capítulo 2 definiu-se caminho em função apenas de células com lado em comum. Isto corresponde a caminhos / com traços verticais e/ou horizontais. No intraligador, extende - se o conceito de caminho através da inclusão de adjacência por meio de vértice, o que permite, para as intraligações, não apenas traços verticais e/ou horizontais, mas também traços inclinados a ± 45^0 e/ou ± 135^0.

Os extremos de um vínculo para o qual não foi possível encontrar um caminho no mini-reticulado (falta de espaço) devem ser conectados externamente, isto é, por interligações. As informações para tal são colocadas no conjunto de falhas F2. Assim sendo há duas possíveis causas de falhas no intraligador: im - possibilidade de ligar pino a pino devido a interferência de ou - tras ligações ou devido à falta de espaço. Estes dois casos estão ilustrados na figura 3.4.

3.1.3 - MANIPULADOR DE FALHAS

O manipulador de falhas resolve no grafo representativo do macro-reticulado as conexões que estavam previstas/ na estrutura comprometida ou conexões entre pinos e que não puderam ser estabelecidas no mini-reticulado. Como no macro-reticulado não há limitação de espaço, estas conexões podem ser resolvidas sempre por interligações. Estas interligações são resolvidas de maneira/ local na macro-célula do componente em cujo mini-reticulado ocorreu a falha.

O algoritmo B12 resolve as falhas contidas no conjunto F1 e o algoritmo B13 as falhas contidas no conjunto F2.

3.2 - FASE II: Designação de Espaços

Depois de obtidas as conexões nos mini-reticulados e obtidos todos os subgrafos representantes das ligações / "externas" (interligações) dos sinais, resta dimensionar as áreas / compreendidas entre componentes e nelas distribuir as conexões referentes às ligações externas. A segunda fase é subdividida em três etapas. Na primeira os componentes são alinhados. O critério de alinhamento baseia-se nas dimensões dos componentes. Este alinhamento tem como resultado uma primeira distribuição dos componentes sobre a placa. Na segunda etapa são consideradas as arestas dos sub

grafos correspondentes a segmentos de canais comuns. Na terceira / são finalmente tratadas as arestas correspondentes a canais próprios. De acordo com os espaços designados às conexões nas etapas dois e três as fileiras de componentes são afastadas para criar o espaço necessário.

3.2.1. - DISTRIBUIÇÃO INICIAL DOS COMPONENTES

A distribuição inicial dos componentes na placa só leva em consideração as dimensões dos componentes. Os componentes são assentados em linhas imaginárias chamadas bases. Os componentes de uma mesma linha na rede são alinhados pela borda sul. Supondo-se que a enumeração das linhas da rede é feita em ordem / crescente de norte a sul, a distância entre uma base de uma linha / de componentes e a base da linha que a antecede é igual à maior altura de componentes nesta linha. Os componentes de uma mesma coluna são, por sua vez, alinhados pela borda oeste. Supondo-se que a enumeração das colunas é feita em ordem crescente de oeste a leste, a distância entre a base de uma coluna e a base da coluna sucessora é igual à maior largura de componente nesta coluna (Fig. 3.5).

Nas etapas seguintes as bases são afastadas quando necessário, para dar espaço às conexões das ligações externas.

Note-se que esta distribuição inicial dos

componentes implica em dimensionar cada linha e coluna pelo pior caso, isto é, pelo componente mais alto da linha ou mais largo da coluna, desperdiçando-se o espaço acima de componentes mais baixos ou à direita dos componentes mais estreitos. Assim sendo, esta abordagem dá melhores resultados quando as fileiras de componentes forem mais homogêneas, isto é; quando todos os componentes em uma mesma linha (coluna) tiverem aproximadamente a mesma altura (largura). Acredita-se que esta não é uma restrição demasiadamente forte: cabe ao usuário na fase (manual) de distribuição dos componentes no macro-reticulado, levar em conta o princípio acima, mesmo / que para tanto seja necessário combinar dois ou mais componentes físicos pequenos em um"componente" lógico que, para o algoritmo, equivale a um componente de dimensões mais próximas das dos demais , minimizando assim o espaço desperdiçado na placa.

Seria possível considerar uma distribuição inicial dos componentes sem linhas de base, acomodando da melhor / maneira possível componentes de diferentes dimensões. Esta distribuição, entretanto, não só é extremamente complexa de implementar , (envolve o clássico problema de "recorte de chapas") como leva a canais não retilíneos entre os componentes, o que dificulta sobremaneira a designação de espaços. Fica porém , aqui a sugestão para / futuro estudo.

3.2.2. - DESIGNAÇÃO DE ESPAÇOS PARA CANAIS COMUNS

Nesta secção é descrito o algoritmo que resolve as conexões associadas a arestas que correspondem a segmentos de canais comuns horizontais. O algoritmo para os canais verticais é análogo, exceto que uma maior densidade é possível depois da resolução das horizontais já que os extremos de cada conexão podem ser determinados com maior precisão.

Por "abrir, definir ou criar um novo espaço" entende-se a definição de uma nova linha (coluna) no reticulado representante da placa. O espaço entre duas linhas (colunas) de componentes é aberto abaixo (à esquerda) da base dos componentes ao norte(a leste). Os espaços seguintes são abertos abaixo (à esquerda) do último espaço aberto. A enumeração dos espaços é feita de acordo com ordem em que foram abertos. Por EH^i_j entende-se o j-ésimo espaço ao sul da linha base da i-ésima linha de componentes da rede (Fig. 3.6) Um espaço entre componentes de duas linhas na rede é chamado de espaço horizontal. O procedimento "abre.espaço.h (i,j) " ou "crie. espaço.h(i,j)" cria o espaço EH^i_j. Este procedimento é utilizado como primitivo no restante do trabalho.

O algoritmo B14 procura distribuir conexões de um determinado canal nos espaços já abertos neste canal. Se isto não for possível, novos espaços são criados, isto é: a placa é expandida.

Assim sendo, ao fim da aplicação do algoritmo B14 para os canais horizontais, cada aresta de um subgrafo de

interligação correspondente a segmento de canal próprio horizontal terá sido associada a um espaço específico aberto no canal. Con - forme ilustra a Figura 3.7, um mesmo espaço de um mesmo canal pode ser utilizado para abrigar mais de uma conexão, quando estas se localizam em porções disjuntas do canal. Duas porções disjuntas / podem se originar em um mesmo componente, pois já se dispõe (da fase I) informações sobre qual a célula da estrutura do componente / em que se origina a interligação. Nesta primeira fase, entretanto, não se dispõe ainda da alocação de espaços nos canais verticais e assim sendo, é necessário utilizar dois espaços horizontais distintos sempre que dois segmentos de conexões em um mesmo canal hori - zontal se originem de um mesmo canal vertical (Fig. 3.7). Já na aplicação do equivalente ao algoritmo B14 para os canais verticais é possível decidir se dois segmentos de conexões verticais que se originam de um mesmo canal horizontal são ou não disjuntos, pois / já se dispõe da alocação de espaços às porções horizontais(Fig.3.8)

Para simplificar a descrição do algoritmo , não foi considerada a restrição de distância mínima entre furos metalizados. Portanto, dois furos podem estar em células adjacentes. Na prática, este pode não ser o caso dependendo das dimensões escolhidas para células e furos. Assim sendo, foi implementada na realidade uma versão ligeiramente modificada do algoritmo B14 que impõe uma distância (de Manhattan) mínima igual a dois passos entre/ furos metalizados. Esta distância mínima é adequada para o caso

mais comum em que se utiliza o valor de um vigésimo de polegada para o passo.

3.2.3 - DESIGNAÇÃO DE ESPAÇOS PARA CANAIS PRÓPRIOS

Uma vez determinados espaços para os canais comuns(Fig. 3.8), resta determinar, em termos de reticulado, caminhos para implementar os segmentos de conexões representadas por arestas correspondentes a segmentos de canais próprios.

Novamente será descrito apenas a restrição das conexões nas regiões horizontais, isto é: nas regiões entre duas linhas de componentes. As conexões nas regiões verticais (isto é: entre duas colunas de componentes) são resolvidas de maneira análoga.

Para cada região entre dois componentes são resolvidas primeiramente os segmentos de conexões que interligam / um componente a um segmento de conexão de canal comum (já alocado) e a seguir as conexões entre os componentes. Denomina-se de região horizontal i,j a parte da região do canal horizontal i compreendida entre dois componentes c_N e c_S da coluna j separada por este canal i. Uma região destas é representada em cada face por uma submatriz de uma matriz designada por R, cujas linhas são enumeradas de -comp a comp+dim.canalh[i] e as colunas de -comp a 2 x comp, onde comp é o máximo das larguras de c_N e c_S e dim. canalh[i]

é o número de espaços já criados para o canal horizontal i. A matriz R foi assim dimensionada para poder conter a resolução do pior caso do maior número possível de conexões do canal próprio. O algoritmo B15 resolve as conexões de uma região horizontal i,j.

Inicialmente as linhas da submatriz representativa da região horizontal i,j é delimitada pelas linhas um e dim.canalh [i] e pelas colunas um e comp (Fig.3.9). Durante a resolução em R das conexões, estas dimensões podem aumentar. O acréscimo de uma linha significa o aumento de um espaço no canal horizontal i e o acréscimo de uma coluna o aumento de um espaço no canal / vertical j ou j-1,dependendo se a coluna foi adicionada pela direita ou pela esquerda à submatriz.

O algoritmo B16, chamdo por B15, inicializa a matriz R. Inicialmente é atribuido o valo zero a todos os elementos de R. A seguir, é atribuído aos elementos (ou células) de R , na face das horizontais, correspondentes aos espaços abertos e ocupados por conexões na região horizontal i,j o número que representa o sinal a que pertencem as conexões. O algoritmo B16, por sua vez, utiliza-se do procedimento auxiliar B17.

O algoritmo B18 ("liga.comp.canal") resolve em R com auxílio do algoritmo B19 as interligações de canal próprio a canal comum. Se isto não é possível através de um segmento de reta vertical(Fig.3.10), então isto é feito por três segmentos: um horizontal que liga dois verticais. Se na determinação destes segmentos o segmento horizontal não puder ser alocado na submatriz

representativa da região horizontal i,j , então o menor número de linhas possível é adicioanado à submatriz para possibilitar a alocação do segmento. O mesmo acontece com as colunas em relação ao segmento vertical que une o segmento horizontal com a conexão do canal comum. O algoritmo B19 é usado tanto pelo algoritmo B18 como pelo algoritmo B20 e atribui às células correspondentes aos segmentos de reta por ele determinados o número que representa o sinal ao qual pertence a conexão. Termina assim a ligação de componente a canais próprios. Resta finalmente, a ligação de componente a componente na mesma região horizontal, tarefa esta, implementada pelo algoritmo B20 ("liga.comp.comp"). Para tanto, B20 inicialmente determina cmin e cmax, que são respectivamente os índices das colunas mais à esquerda e mais à direita em que se localizam células das estruturas de c_N e c_S a serem interligadas. Em seguida, determina-se em R uma linha que tenha células não ocupadas desde a "primeira" coluna (isto é: a coluna -comp) até a coluna / cmax ou, se esta linha não existe, determina-se em R uma linha que tenha células não ocupadas desde a coluna cmin até a "última" coluna (isto é: a coluna 2 * comp). Uma destas linhas sempre poderá ser determinada expandindo-se, se necessário, a submatriz. Em seguida B20 utiliza B19 para estabelecer ligações desta linha a cada célula da estrutura de c_N e c_S a ser interligada; finalmente, atribui-se às células utilizadas desta linha o número do sinal comprometendo assim o segmento utilizado (Fig.3.11)

Determinadas todas as regiões é possível determinar de quanto devem ser afastados as linhas e colunas de componentes para acomodar as conexões na placa.

3.3 - O TRAÇADO É COMPLETO

A causa dos insucessos (ligações não completadas) nos algoritmos tradicionais, isto é: a falta de espaço devido a congestionamento de conexões, é contornada pelo algoritmo / proposto. Nas intraligações podem ocorrer falhas, devido à rigidez dos componentes que, limita as áreas nas quais as mesmas podem ser estabelecidas. Mas estas falhas são sempre resolvidas satisfatóriamente no grafo representativo do macro-reticulado. A resolução de todas as falhas de intraligações é garantida graças à restrição de que pinos só podem ocorrer na estrutura de um componente. Se os mesmos pudessem ocorrer em qualquer parte de um mini-reticulado, então um pino não na estrutura corre o perigo de ser isolado por intraligações. Se é vital o uso de componentes com pinos fora da estrutura, então placas de mais de duas faces são necessárias para manter a garantia de traçado completo.

Admitindo-se que os procedimentos apresentados são algoritmos de fato (isto é: sempre terminam), então um traçado completo é garantido, ou seja, todas as conexões são resolvi-

das . A própria especificação dos algoritmos utilizados demonstra este fato, se aceita a informalidade da linguagem utilizada e su - pondo que os programas foram completamente depurados.

4. - CONCLUSÕES E SUGESTÕES PARA FUTURA PESQUISA

A principal característica do algoritmo apresentado consiste na garantia de obtenção automática de um traçado sem insucessos, e, dentro dos limites da heurística do algoritmo, usando "mínima" área da placa. Isto elimina a fase posterior/ de alterações e correções manuais normalmente necessária para resolver insucessos quando se utiliza os algoritmos tradicionais / que não garantem traçado completo. Além disto, o sistema de pesos das arestas no garfo representativo no macro-reticulado permite a alteração dos mesmos durante a determinação dos subgrafos, o que torna o algoritmo bastante flexível. Se as arestas mais congestionadas recebem pesos maiores, então as interligações dos sinais ainda não resolvidos tendem a ocupar os canais menos congestionados . Isto, por sua vez, produz uma distribuição mais uniforme das interligações em sacrifício da "minimização" dos comprimentos das mesmas .

A maior limitação do algoritmo é a obtenção das dimensões da placa só no final da execução do algoritmo. Se a placa obtida é demasiadamente grande, então alguns componentes / devem ser retirados ou recolocados e os sinais redefinidos para se obter uma nova solução que resulte em uma placa de dimensões aceitáveis.

O algoritmo proposto foi implementado no DEC10 do Centro de Computação da UNICAMP por três programas em ALGOL60. Um breve manual de utilização dos sistema é apresentado no apêndice D. O maior programa ocupa 22K palavras e para a obtenção da maior placa no apêndice E foram necessários 4,5 minutos de CPU. Os diagramas das placas no apêndice E não representam o que se denomina de arte final (matriz de impressão) em circuito impresso. O objetivo destes diagramas é unicamente permitir a melhor visualização dos resultados.

De modo geral, pode-se afirmar que os resultados obtidos são extremamente favoráveis quanto a memória, tempo de execução e tamanho da placa obtida quando comparados com um algoritmo do tipo tradicional que já vinha sendo utilizado na UNICAMP. Para a maior placa do apêndice E, este algoritmo tradicional utilizou memória 12 % maior, tempo 280 % maior e área 160 % maior do que a do algoritmo proposto.

Apenas alguns aspectos desta nova maneira / de resolver traçados de circuito impresso foram explorados.

Não foi feito nenhum estudo mais detalhado de outros tipos de grafos para o macro-reticulado que introduzis - sem penalidades a certas classes de interligações para otimizar alguns aspectos como, por exemplo, a redução do número de furos metalizados.

Outros tipos de subárvores também podem resultar em melhores soluções. A estrutura de barramento ("bus") comumente usada em projetos de circuitos digitais, por exemplo, não é refletida pelas subárvores obtidas pelos algoritmos descritos no apêndice C.

A segunda fase do algoritmo assenta os componentes sobre linhas imáginárias e se for preciso afastar dois componentes então juntamente com os componentes são afastadas as duas fileiras nas quais se encontram os componentes. Além disto, modernamente utiliam-se cada vez mais componentes das mais diversas dimensões em uma mesma placa.

Estes dois fatos são responsáveis por um certo "desperdício" de espaço pelo algoritmo proposto. Uma maior compactação dos componentes e consequentemente abandono das linhas de base na segunda fase resultaria em canais não lineares mas pode originar placas de dimensões menores. Em alguns casos as dimensões também podem ser maiores do que as obtidas pelo algoritmo proposto já que seguidas mudanças de direção de um canal necessitam de mais espaço. Além disto, a complexidade de algoritmos para canais não lineares cresce bastante. De alguma maneira os canais não lineares deveriam ser refletidos pela estrutura do macroreticulado e consequentemente pelo grafo representativo.

As questões abordadas acima são possíveis áreas para futura pesquisa visando refinar a técnica de solução a-

qui introduzida. Torna-se ainda necessário, para completar um sistema prático, um subsistema de entrada e crítica de dados a serem apresentados ao traçador proposto, bem como um subsistema para o traçado de arte final, ocasião em que inúmeras otimizações/ locais podem ser introduzidas no traçado. Estes dois subsistemas/ já se encontram atualmente em fase de desenvolvimento por parte / de outros pesquisadores.

5. - RESUMO

É descrito um algoritmo de traçado de rotas em placas de circuito impresso que utiliza na entrada, em lugar da localização específica de cada componente na placa, apenas uma descrição de posição relativa dos componentes. O algoritmo / garante um traçado sem insucessos em placas de duas faces de dimensões não pré-fixadas. É aceito qualquer componente que possua todos os seus pinos na periferia do menor retângulo que os contém. O algoritmo não é celular. As conexões são estabelecidas de maneira macroscópica em um quadriculado de poucas linhas e colunas, onde os componentes estão reduzidos a pontos. Isto é vantajoso tanto do ponto de vista de espaço como de tempo. O espaço entre componentes não é limitado para permitir a passagem de um número / qualquer de conexões. Depois de conhecido o número de conexões / obtidas no quadriculado que passam entre os componentes são determinadas a localização definitiva de componentes e conexões e, consequentemente, as dimensões finais da placa.

BIBLIOGRAFIA

|1| GILBERT,E.N. & POLLAK,H.O. Steiner minimal trees. *SIAM J. Appl. Math.*,Philadelphia, *16*(1):1-29, jan. 1968.

|2| HANAN,M. On Steiner's problem with rectilinear distande. *SIAM J.Appl. Math.*, Philadelphia,*14*(2):255-65, mar.1966.

|3| HWANG,F.K. On Steiner minimal trees with rectilinear distance. *SIAM J. Apll. Math.*, Philadelphia,*30*(1):104-14,jan. 1976.

|4| GAREY,M.R. & JOHNSON,D.S. The rectilinear Steiner tree problem is NP-complet. *SIAM J. Appl. Math.*, Philadelphia,*32* (4):826-34,jun. 1974.

|5| BONDY,J.A. & MURTY,U.S.R. *Graph theory with applications*. Ontário, MacMillan,1976. 264p.

|6| ZDENEK,F. Routing problem. *CACM*, New York,*16*(9):572-4,set. 1973.

|7| LEE, C.Y. Algorithm for path connection and its applications. *IRE Trans.Eletronic Computer*, New York, *EC-10*,1961.

|8| AGRAWAL,P.& BREUER, M.A. Some theoretical aspects of algorithmic routhing. *In: Design automation workshop,14*,1977.

|9| GEYER,J.M. Connection routing algorithm for printed circuit boards. *IEEE Trans.Circ.theory*,New York,*CT-18*(1):95-100, jan.1971.

|10| SUTHERLAND,I.E. A method for solving arbitrary-wall mazes by computer. *IEEE Trans,Comp.*,New York, *C-18*(12) dez.1969.

|11| DE PEDRO,M.T. & GARCIA,R. M.Docil; an automatic system for printed circuit board designing. A board description language and an algorithm to connect a set of points, *In: Design automation workshop,14*,1977. p.174-81.

|12| HIGHTOWER,D.W. A solution to line-routing problems on continuous plane. *In: Design automation workshop*,1969.

|13| NEGRI,P.G. *Implementação e teste de um sistema para traçado de circuitos impressos*. Campinas,FEC/UNICAMP,1974. (Tese de Mestrado)

|14| LAGES,N.A. de C. SACCO; sistema automatizado de construção de circuitos otimizados. *In: SEMISH*.

|15| ARAMAKI,I.;KAWABATA,I. & ARIMOTO,K. Automation of etching-pattern lay-out. *CACM*, New York,*14*(11):720-30,nov.1971.

|16| ABEL, L.C. On the ordering of connections for automatic wire routing. *IEEE Trans. on Comp.*,New York,*C-21*,nov.1972.

|17| ADSHED,H.G. Optimizing automatic tracking of multilinear boards. *SIGDA Newsletter*, New York, *5*(3) set.1975.

|18| DIJKSTRA,E.W. A note on two problems in connexion with graphs. *Numerische mathematik*,Berlin,*1*:269-71,1959.

A P Ê N D I C E S

APÊNDICE A

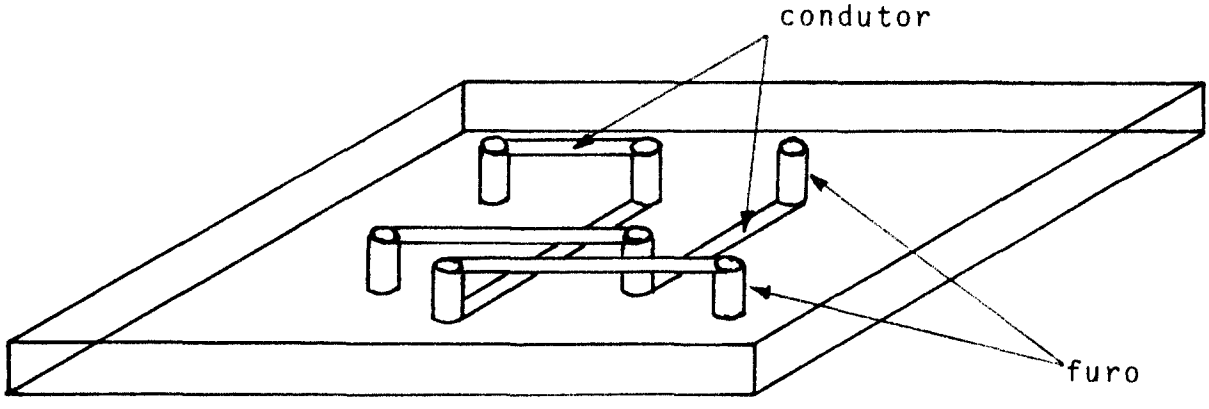


Figura 1.1 - Placa de duas faces (conexões)

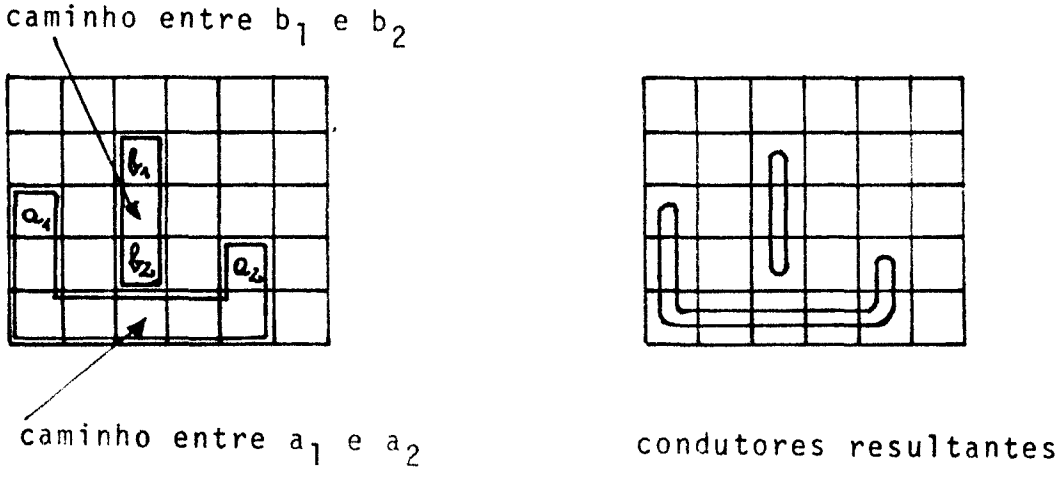


Figura 1.2

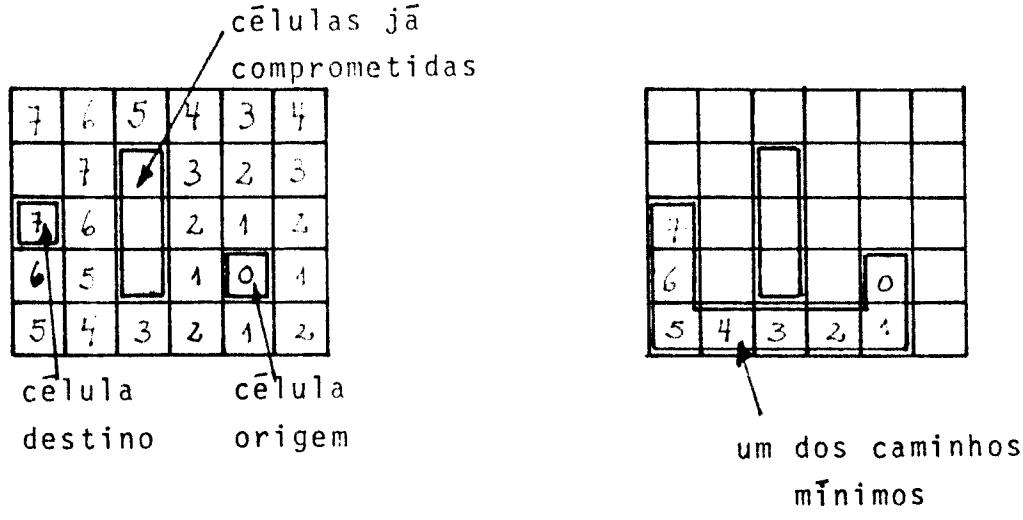


Figura 1.3

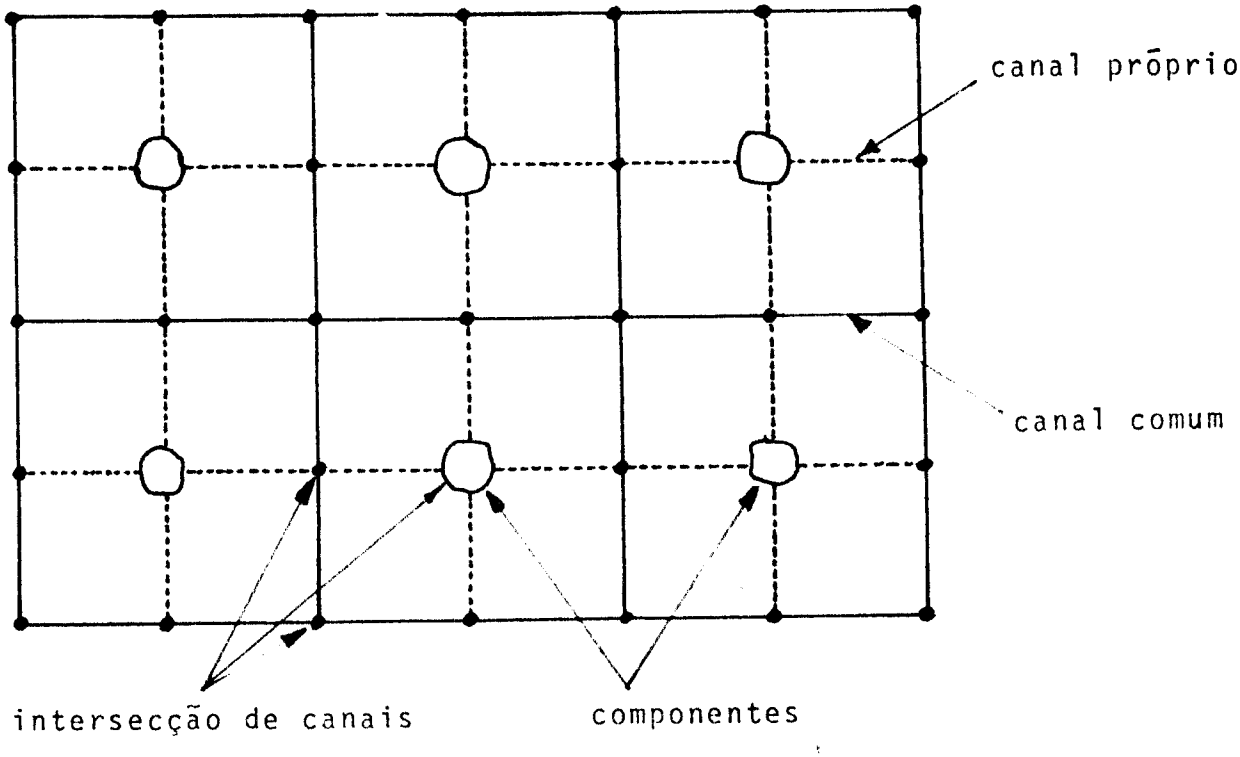
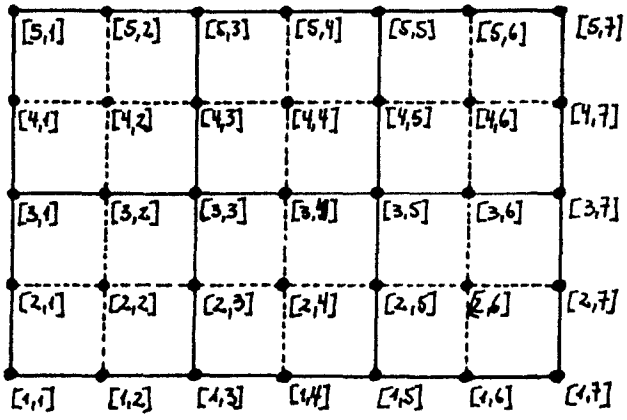
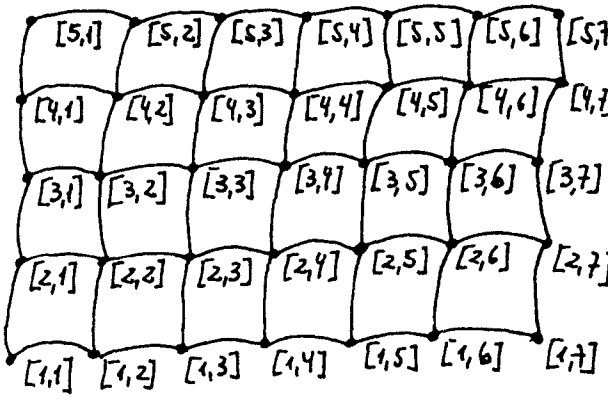


Figura 1.4 - Macro-Reticulado



macro-reticulado



grafo representativo

Figura 2.1 - Identificação de intersecções e vértices

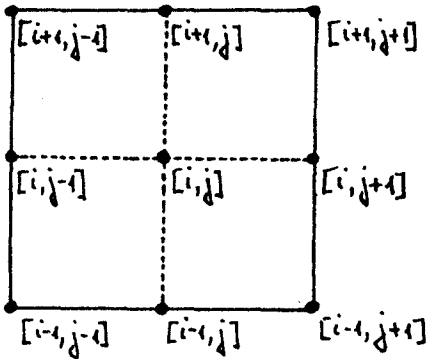
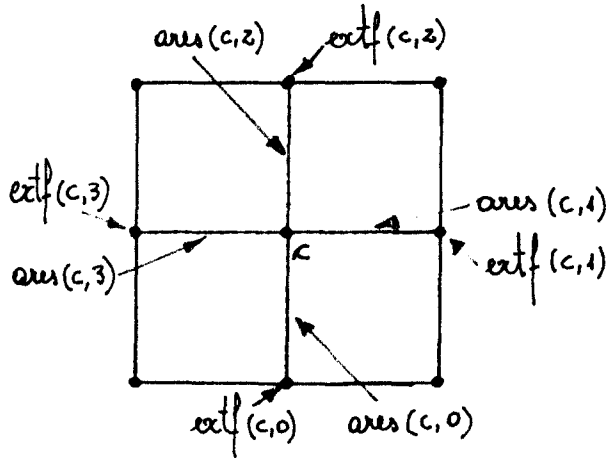
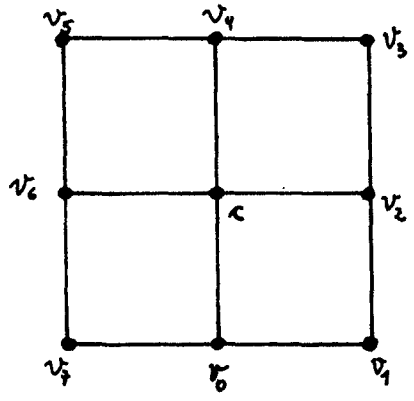


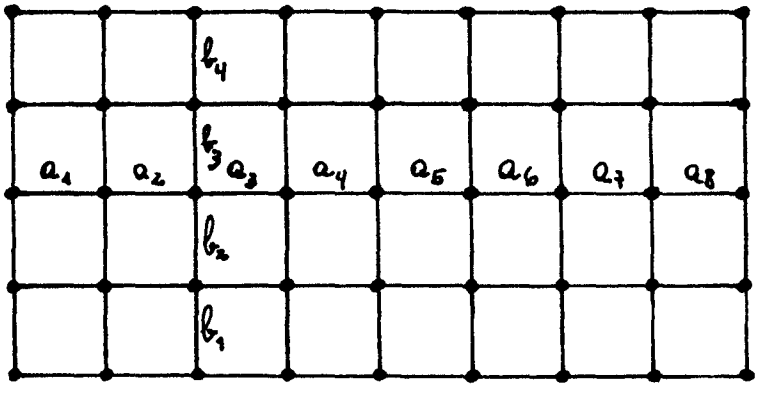
Figura 2.2 - Macro-célula do componente representado pela intersecção |i,j|





$extah(v_i, c) = v_{(i+1) \bmod 8}$

Figura 2.4 - Ilustração da função extah



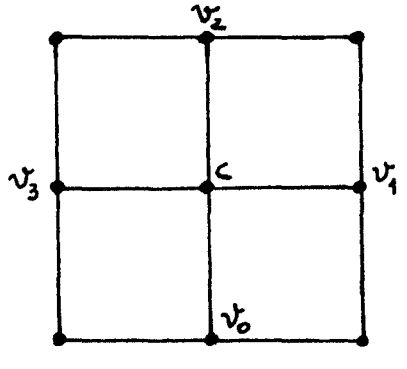
$A = \{a_i / 1 \leq i \leq 8\}$

$B = \{b_i / 1 \leq i \leq 4\}$

$$seg(a_i, A) = \begin{cases} a_{i+1} & \text{se } i < 8 \\ a_i & \text{se } i = 8 \end{cases}$$

$$seg(b_i, B) = \begin{cases} b_{i+1} & \text{se } i < 4 \\ b_i & \text{se } i = 4 \end{cases}$$

Figura 2.5 - Ilustração da função seg



$det.bor(c, v_0) = 0$

$det.bor(c, v_1) = 1$

$det.bor(c, v_2) = 2$

$det.bor(c, v_3) = 3$

Figura 2.6 - Ilustração da função det.bor

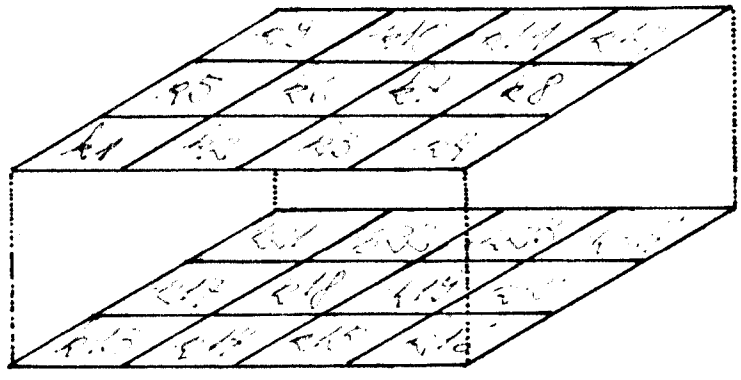
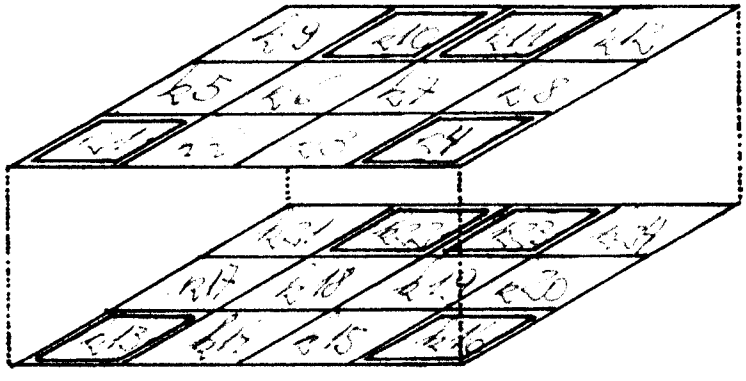


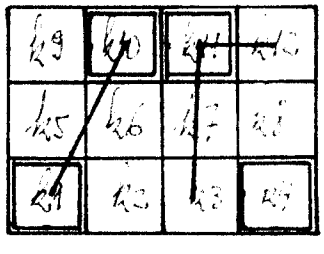
Figura 2.7 - Mini-reticulado



$P^{c,0} = \{k_1, k_4, k_{10}, k_{11}\}$

$Q^{c,1} = \{k_{14}, k_{15}, k_{20}, k_{24}, k_{21}, k_{17}$

Figura 2.8 - Pinos e Passagens



$W^{c,0} = \{\{k_1, k_{10}\}, \{k_{11}, k_3\}, \{k_{11}, k_{12}\}\}$

Figura 2.9 - Vínculos

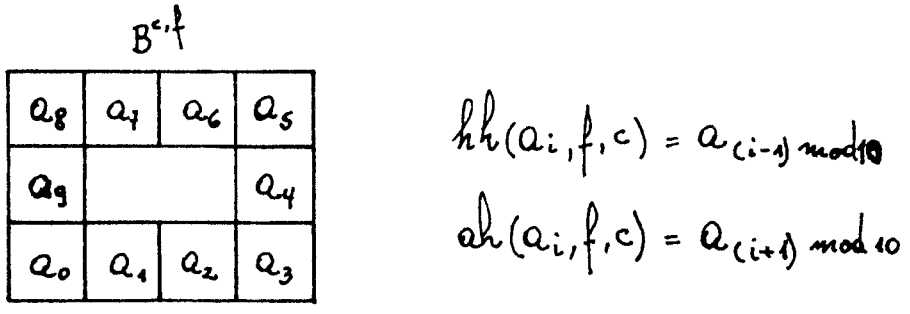


Figura 2.10 - Ilustração das funções hh e ah

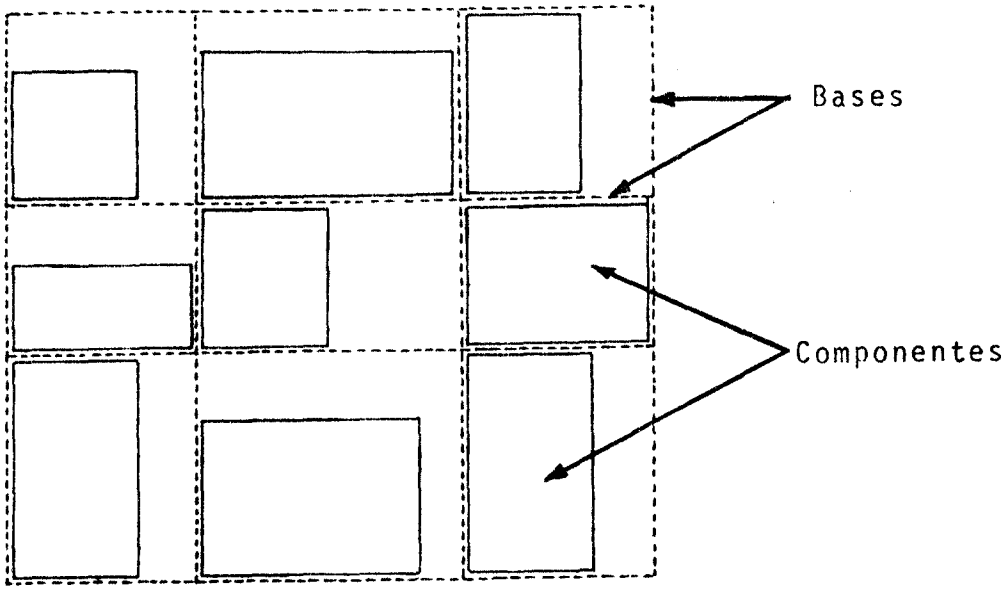


Figura 3.5 - Alinhamento Inicial de Componentes

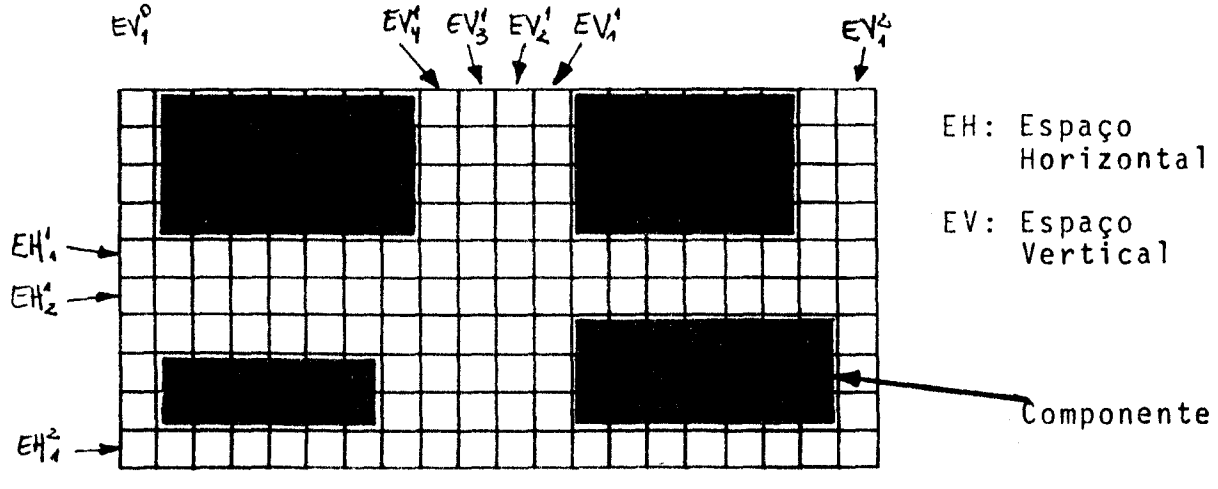


Figura 3.6

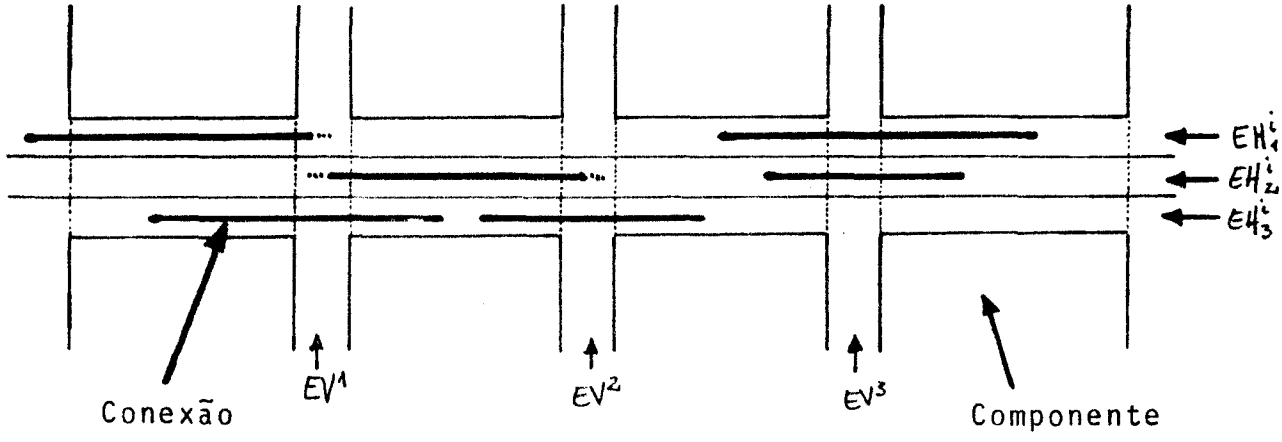


Figura 3.7

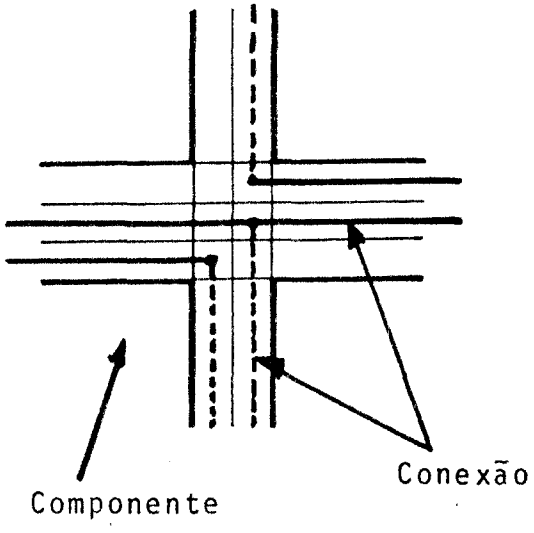
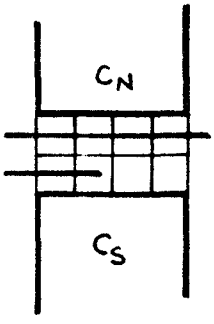


Figura 3.8



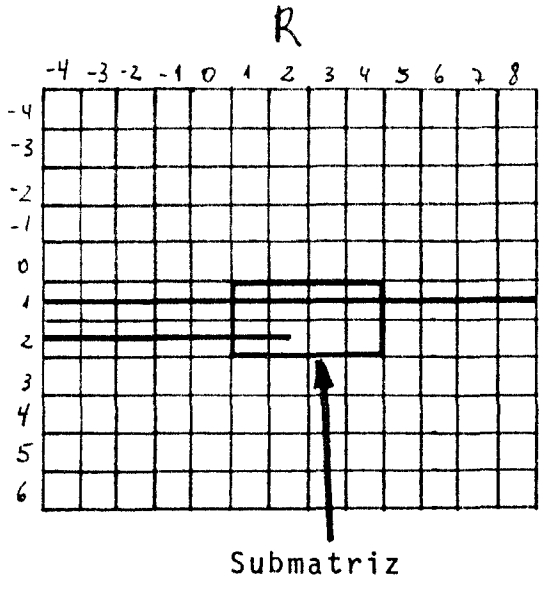
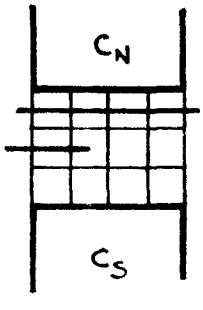


Figura 3.9



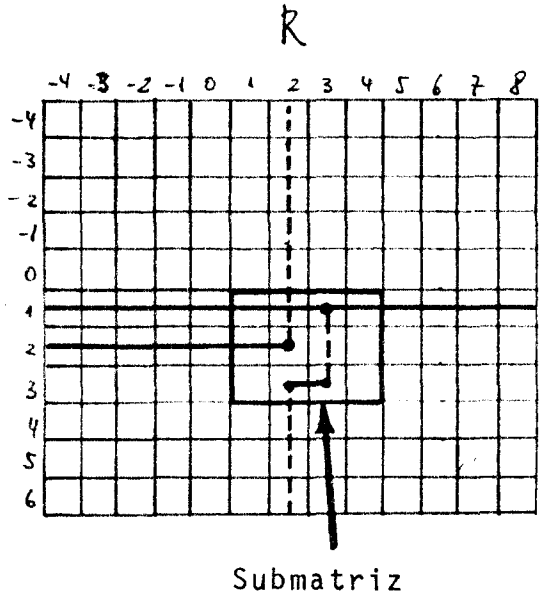


Figura 3.10

APÊNDICE B

PROCEDIMENTO B1

Procedimento de iniciação de pesos nas arestas de canais próprios.

```
início
 para ∀c ∈ C faça  (*C: conjunto dos vértices representativos de
                        componentes*)
  início
   se c ∈ D ∪ {u0}
    então início
      K^c := ∅;
      torne.acessível (c);
      para b:= 0 até 3 faça
       início
        f:= (b+1) mod 2;
        se σ^c_{γ,b} ≠ ∅ ou ∃(v,z) ∈ L^{c,f} / v ∈ σ^{c,f}_γ e z ∈ Q^{c,f}_b
         então r(ares(c,b)):= 1
         senão r(ares(c,b)):= ∞
       fim
      fim
    senão para b:= 0 até 3 faça r(ares(c,b)):= ∞
  fim
fim
```

PROCEDIMENTO B2

```
procedimento torne.acessível (c);
 início
  b: = 0 ;
  enquanto b < 4 faça
   início
    l:=(b+l) mod 2  ;
    se σ^{c,I}_{γ,b} = ∅  e Q^{c,I}_b ≠ ∅
     então início
       bb:=(b+l) mod 4;
       possível:= σ^{c,I}_{γ,bb} ≠ ∅ ;
       se possível
        então início
          ca:= elemento de B^{c,I}_b ∩ B^{c,I}_{bb}   ;
          determine p ∈ σ^{c,I}_{γ,bb} tal que d(p,ca) é mínima ;
          y:=p ;
          enquanto y ∈ B^{c,I}_{bb}  faça
           início
            se ∃ z/ {y,z} ∈ L^{c,I} ∪ K^{c,I},z ∈ S^{c,I}_{p,y} e |S^{c,I}_{p,z}| é mínimo
             então y:= z
             senão y:=hh(y,c,I)
           fim;
          possível := y ∈ B^{c,I}_b;
          enquanto y ∉ Q^I_b e possível faça
           início
```

```
  se  ∃ x/{x,y} ε L^{c,I} ∪ K^{c,I},x ε S^{c,I}_{p,y} e |S^{c,I}_{p,x}| é mínimo
   então y:= x
   senão y:= hh(y,c,I) ;
  possível:= y ε B^{c,I}_b
 fim ;
z:= y ;
resolveu:= falso  ;
enquanto possível e  não resolveu faça
 início
  se   x/{x,y} ε L^{c,l}   K^{c,l},x ε S^{c,l}_{p,y} e |S^{c,l}_{p,x}| é mínimo
   então início
     se  y ε Q^{c,I}_b
      então início
        y:= hh(y,c,I)    ;
        possível:= y ε B^{c,I}_b
       fim
      senão possível:= falso
    fim
   senão se y ε Q^{c,I}_b
     então início
       resolver:= verdadeiro ;
       x:= y    ;
       enquanto y ≠ z faça
        início
```

```
                x:= ah(x,c,I)
                se  ∃ v/{v,x} ∈ K^{c,I}
                 então início
                   K^{c,I}:= (K^{c,I} \ {{v,x}}) ∪ {{v,y}}   ;
                   y:= x
                  fim
                 senão se ∃ v/{v,x} ∈ L^{c,I}
                   então início
                     L^{c,I}:= (L^{c,I} \ {{v,x}}) ∪ {{v,y}}   ;
                     y:= x
                    fim
               fim;
               K^{c,I}:= K^{c,I} ∪ {{p,z}}
             fim
            senão início
              y:= hh(y,c,I)  ;
              possível:= y ∈ B_b^{c,I}
             fim
        fim
  fim;
se não possível
 então início
   bb:=(b-1) mod 4   ;
   possível:= σ_{γ,bb}^{c,I} ≠ ∅   ;
```

se possível

então início

ca:= elemento de $B_b^{c,I} \cap B_{bb}^{c,I}$;

determine p ∈ $\sigma_{\gamma,bb}^{c,I}$ tal que d(p,ca) é mínimo ;

y:=p ;

enquanto y ∈ B_{bb}^{I} faça

início

se ∃ z/{y,z} ∈ $L^{c,I} \cup K^{c,I}$, z ∈ $S_{p,y}^{c,I}$ e $|S_{p,z}^{c,I}|$ é mínimo

então y:= z

senão y:= ah(y,c,I)

fim ;

possível:= y ∈ $B_b^{c,I}$;

enquanto y ∉ $Q_b^{c,I}$ e possível faça

início

se ∃ x/{x,y} ∈ $L^{c,I} \cup K^{c,I}$, x ∈ $S_{p,y}^{c,I}$ e $|S_{p,x}^{c,I}|$ é mínimo

então y:= x

senão y:= ah(y,c,I);

possível:= y ∈ $B_b^{c,I}$

fim ;

z:= y ;

resolveu := falso ;

enquanto possível e não resolveu faça

início

se x/{x,y} ∈ $L^{c,I}$ $K^{c,I}$, x ∈ $S_{p,y}^{c,I}$ e $|S_{p,x}^{c,I}|$ é mínimo

```
então início
  se y ∈ Q_b^{c,I}
   então início
     y:= ah(y,c,I) ;
     possível:= y ∈ B_b^{c,I}
    fim
   senão possível:= falso
  fim
senão se y ∈ Q_b^{c,I}
   então início
     resolveu:= verdadeiro   ;
     x:= y ;
     enquanto y ≠ z faça
      início
       x:= hh(x,c,I)  ;
       se ∃ v/{v,x} ∈ K^{c,I}
        então início
          K^{c,I}:= (K^{c,I} \ {{v,x}}) ∪ {{v,v}}   ;
          y:= x
         fim
        senão se ∃ v/{v,x} ∈ L^{c,I}
          então início
            L^{c,I}:=(L^{c,I} \ {{v,x}}) ∪ {{v,y}}   ;
            y:= x
```

```
                    fim
                fim ;
                K^{c,I} := K^{c,I} ∪ {{ p,z }}
            fim
          senão início
            y:= ah(y,c,I) ;
            possível:= y ε B_b^{c,I}
          fim
      fim
    fim;
se não possível
 então início
    bb:=(b+2) mod 4;
    T:= ∅;
    encontrou:= falso ;
    enquanto σ_{γ,bb}^{c,I} ≠ ∅ e não encontrou faça
     início
      p:= elemento de σ_{γ,bb}^{c,I} ;
      σ_{γ,bb}^{c,I} := σ_{γ,bb}^{c,I} \ {p} ;
       T:= T ∪{p} ;
       x:= se ∃z/{p,z} ε K^{c,I} ∪ L^{c,I} e S_{z,p}^{c,l} ⊂ B_b^{c,I} e
          |S_{z,p}^{c,I}| é mínimo
        então z senão ah(p,c,I) ;
       enquanto x ∉ B_b^{c,I} e x ≠ p faça
```

início

se $\exists$ v/{v,x}ϵ $K^{c,I} \cup L^{c,I}$ e $S^{c,I}_{p,v} \supset S^{c,p}_{p,x}$

então determine v tal que{v,x} $\epsilon K^{c,I} \cup L^{c,I}$ e

$S^{c,I}_{p,v} \supset S^{c,I}_{p,x}$ e $|S^{c,I}_{v,p}|$ é mínimo

senão v:= ah (x,c,I) ;

x:= v

fim ;

se x ϵ $B^{c,I}_{b}$

então início

y:= se $\exists$ z/{p,z} ϵ $K^{c,I} \cup L^{c,I}$ e $S^{c,I}_{p,z} \supset B^{c,I}_{b}$ e

$|S^{c,I}_{p,z}|$ é mínimo

então z senão hh(p,c,I) ;

enquanto y $\notin$ $B^{c,I}_{b}$ faça

início

se $\exists$ v/{v,y} ϵ $K^{c,I} \cup L^{c,I}$ e $S^{c,I}_{v,p} \supset S^{c,I}_{y,p}$

então determine v tal que {v,y} ϵ $K^{c,I} \cup L^{c,I}$ e

$S^{c,I}_{v,p} \supset S^{c,l}_{y,p}$ e $|S^{c,I}_{p,v}|$ é mínimo

senão v:= hh(x,c,I) ;

x:= v

fim ;

se y $\neq$ x

então início

y:= hh(y,c,I) ;

```
enquanto y ≠ x e não encontrou faça
 início
  se  ∃ v/{y,v} ∈ K^{c,I} ∪ L^{c,I} e |S^{c,I}_{p,v}| é mínimo
   então y:= v
   senão se y ∈ Q^{c,I}_b
     então início
       encontrou:= verdadeiro ;
       solução:= y ;
       distância:= d(p,y) ;
       y:=hh(y,c,I) ;
       enquanto y ≠ x e distância > d(p,y) faça
        início
         se  ∃ v/{y,v} ∈ K^{c,I} ∪ L^{c,I} e |S^{c,I}_{p,v}| é
           mínimo
          então y:= v
          senão se y ∈ Q^{c,I}_b
            então início
              distância:= d(p,y) ;
              solução:= y;y:= hh(y,c,I)  ;
             fim
            senão y:= hh(y,c,I)
        fim
       K^{c,I}:= K^{c,I} ∪ {{p, solução}}
      fim
     senão y:= hh(y,c,I)
```

```
                    fim
                  fim
                fim
              fim;
            σ^{c,I}_{γ,bb} := σ^{c,I}_{γ,bb} ∪ T
           fim
        fim
      fim;
    b := b+1
  fim
fim
```

PROCEDIMENTO B3

PROCEDIMENTO QUE DETERMINA UMA SUB-ÁRVORE NO GRAFO

```
início
 P#γ :={u0 }  ;
 A#γ := ∅     ;
 enquanto D ≠ ∅ faça
  início
   para ∀ v ∈ P#γ faça l(v):= 0   ;
   para ∀ v ∈ P̄#γ faça l(v):= ∞   ;
   S:= { u0 }  ;
   u:= u0    ;
   objetivo:= falso;
   enquanto não objetivo faça
    início
     ul:=u ;
     para ∀ v ∈ S̄ faça l(v):= min { l(v) ,l(v)+r(v,u ) }  ;
     redefina u tal que l(u)= min { l(v) }   ;
                                 v ∈ S̄
     se u ∈ D
      então início
        objetivo:= verdadeiro  ;
        D:= D \ { u }     ;
        w:= u  ;
        compromete (w,det.bor(w,ul));
        enquanto w ∉ P   faça
```

```
            início
             ul:= w  ;
             P#γ:= P#γ U {w}   ;
             determine v tal que l(v) = l(w)-r(w,v)  ;
             A#γ:= A#γ U {(w,v)};
             w:= v
            fim  ;
           se w ε C  então compromete (w,det.bor(w,ul))
          fim
         senão S:= S U {u}
      fim
    fim
fim
```

PROCEDIMENTO B4

```
procedimento compromete (c,b);
 início
  se σ^c_{γ,b} = ∅
   então início
     I:= (b+1) mod 2  ;
     determine v1 e v2/ v1 ∈ Q_b^{c,I}  e v2 ∈ σ_γ^{c,I} e {v1,v2} ∈ K^{c,I} ;
     K^{c,I} := K^{c,I} \ {{v1,v2}}   ;
     L^{c,I} := L^{c,I} ∪ {{v1,v2}}
    fim
 fim
```

PROCEDIMENTO B5

INTRALIGADOR

```
início
 para ∀ x ∈ C faça
  início
   para γ:= 1 até nS  faça pino.a.pino (x, γ) ;
   traçado (x)
  fim
fim
```

PROCEDIMENTO B6

```
procedimento pino.a.pino (c,#γ ) ;
 início
  n:= |σ_γ^{c,0}|    ;
  se n > 1
   então início
     para i:= 1 até n faça E((p_i)):= {p_i}     ;
     q:= n    ;
     k:= 0    ;
     enquanto k < n *(n-1)/2 e q > 1 faça
      início
       k:= k+1      ;
       se (se E((s_{k,1})) ≠ E((s_{k,2}))
         então possível. ligação. entre (c,s_{k,1},s_{k,2})
         senão falso
        então início
          E((s_{k,1})):= E((s_{k,1})) ∪ E((s_{k,2}))   ;
          q:= q-1
         fim
      fim ;
     se q > 1
      então início
        I_1:= < >   ;
        para i:= 1 até n faça
         início
```

```
            se (se vazio (Il)
              então verdadeiro
              senão E((p_i)) ≠ E((e)) ∀ e em Il)
             então anexe (p_i,Il)
           fim;
         Fl:= Fl ∪ {< x  #γ Il> }
       fim
    fim
fim
```

PROCEDIMENTO B7

```
função lógica possível. ligação.entre (c,pl,p2) ;
 início
  possível:= falso ;
  f:= 0 ;
  enquanto não possível e f < 2 faça
   início
    x:= se ∃{pl,v} ∈ L^{c,f}/v ∈ S^{c,f}_{pl,p2} e |S^{c,f}_{v,p2}| é mínimo
       então v senão ah(pl,f,c) ;
    enquanto x ≠ p2 e x ∈ S^{c,f}_{pl,p2} faça
     início
      x:= se ∃{x,v} ∈ L^{c,f} /|S^{c,f}_{x,pl}| é mínimo
         então v senão ah(pl,f,c)
     fim;
    se x = p2
     então início
       possível:= verdadeiro ;
       L^{c,f}:= L^{c,f} ∪ {{pl,p2}}
      fim
     senão f:= f+1
   fim;
  possível. ligação.entre:= possível
fim
```

PROCEDIMENTO B8

```
procedimento traçado (c);
 início
  f:= 0 ;
  enquanto f < 2 faça
   início
    v2:= elemento de B0^c,f ∩ B3^c,f ;
    v1:= ah(v2,f,c);
    enquanto v1 ≠ v2 faça
     início
      se ∃ v/{v,v1} ∈ L^c,f ,v ∈ S^c,f_v1,v2 e |S^c,f_v1,v| é mínimo
       então início
         L^c,f := L^c,f \ {{v,v1}} ;
         trace (v1,v,f,c) ;
        fim
       senão v1:= ah(v1,f,c)
     fim;
    f:= f+1
   fim
 fim
```

PROCEDIMENTO B9

```
procedimento trace (orig, dest) ;
 início
  It:=  << orig dest >>  ;
  orig:= ah(orig,c,f) ;
  enquanto não vazio (It) faça
   início
    enquanto orig ≠ dest faça
     início
      se ∃ v/{v,orig} ∈ L^{c,f},v ∈ S^{c,f}_{orig,dest} e |S^{c,f}_{orig,v}| é mínimo
       então início
         L^{c,f}:= L^{c,f} \ {{v,orig}} ;
         anexe ( < orig v > ,It)
         dest:= v
        fim
       senão orig:= ah(orig,f,c)
     fim;
    Il:= retire(It);
    dest:= retire (Il);
    orig:= retire (Il);
    coord(c,orig,xl,yl);
    coord(c,dest,x2,y2);
    kl:= 0 ;
    enquanto kl < 2 faça
     início
```

```
x0:= x1 ;
y0:= y1 ;
k:= 0 ;
I:=  << x1 y1 >> ;
I1:= < > ;
próximo (x1,y1,x2,y2,x,y,k) ;
achou:= falso ;
enquanto k ≠ 0 e não achou faça
 início
  x1:= x;
  y1:= y;
  anexe (k,I1) ;
  k:= 0 ;
  enquanto (x1 ≠ x0 ou y1 ≠ y0) e MR^{c,f}_{x1,y1} ≠ #γ faça
   início
    se k = 0 então MR^{c,f}_{x1,y1}:= #γ ;
    próximo (x1,y1,x2,y2,x,y,k) ;
    se k = 0
     então início (* volta *)
       MR^{c,f}_{x1,y1} := 0 ;
       I2:= retire(I)
       y1:= retire(I)
       x1:= retire(I2)
       k:= retire (I1)
```

```
      fim
     senão se MR(x,y)^(c,f) = 0 ou MR(x,y)^(c,f) = #γ
       então início (* avança *)
         anexe (k,Il);
         anexe ( < xl yl >,I) ;
         k:= 0 ;
         xl:= x;
         yl:= y
        fim
   fim;
  se xl ≠ x0 ou yl ≠ y0
   então achou:= verdadeiro
   senão próximo (xl,yl,x,y,k)
 fim;
se xl = x0 e yl = y0
 então início
   F2:=F2 ∪ {< c #γ orig dest >} ;
   kl:= 2
 fim
 senão se xl≠ x2 ou yl ≠ y2
   então início
     kl:= kl+l;
     x:= xl; xl:=x2; x2:= x ;
     y:= yl; yl:=y2; y2:= y ;
```

```
            fim
           senão k:= 2
      fim
     se não vazio (It)
      então início
        Il:= retire (It) ;
        anexe (Il,It)    ;
        dest:= retire (Il)
       fim
   fim
fim
```

PROCEDIMENTO B10

```
procedimento próximo (x1,y1,x2,y2,x,y,k) ;
 início
  determine x e y tal que (x=x1 ou x=x1+1 ou x=x1-1)
   e (y=y1 ou y=y1+1 ou y=y1-1) e d(MR^{c,f}_{x,y}, MR^{c,f}_{x2,y2}) é mínimo ;
  achou:= falso  ;
  se k= 0
   então início
     se MR^{c,f}_{x,y} = 0 ou MR^{c,f}_{x,y} =     #Y
      então início
        k:= 1 ;
        achou:= verdadeiro
       fim
    fim ;
  se k < 2 e não achou
   então início
     vizinho (x1,y1,x,y, verdadeiro, x3,y3) ;
     se x3 > 0 e x3 <=n^c  e y3 > 0 e y3  <=m^c
      então início
        se M^{c,f}_{x3,y3} = 0 ou M^{c,f}_{x3,y3} =     #Y
         então início
           k:= 2 ;
           achou:= verdadeiro ;
           x:= x3 ;
           y:= y3
```

```
          fim
        fim
    fim;
  se não achou
   então início
     vizinho (xl,yl,x,y, falso, x3,y3) ;
     se x3 > 0 e x3 <= n^c e y3 > 0 e y3 < = m^c
      então início
        se MR^{c,f}_{x3,y3} = 0 ou MR^{c,f}_{x3,y3} = ≠ Y
         então início
           k:= 3  ;
           achou:= verdadeiro ;
           x:= x3 ;
           y:= y3
          fim
       fim
    fim;
  se não achou então k:= 0
fim
```

PROCEDIMENTO B11

```
procedimento vizinho (x1,y1,x,y, esquerda,x3,y3) ;
 início
  x3:= x ;
  y3:= y ;
  se x1 = x
   então x3:= se y > y1
     então (se esquerda então x3-1 senão x3+1)
     senão (se esquerda então x3+1 senão x3-1)
   senão se y1 = y
     então y3:= se x > x1
       então se esquerda então y3+1 senão y3-1
       senão se esquerda então y3-1 senão y3+1
     senão se x > x1
       então início
         se y > y1
          então início
            se esquerda
             então x3:= x1
             senão y3:= y1
           fim
          senão se esquerda
            então y3:= y1
            senão x3:= x1
       fim
```

```
        senão se y > y1
          então início
            se esquerda
              então y3:= y1
              senão x3:= x1
           fim
          senão se esquerda
            então x3:= x1
            senão y3:= y1
fim
```

PROCEDIMENTO B12

MANIPULADOR DE FALHAS I

```
início
 enquanto Fl ≠ ∅ faça
  início
   Il:= elemento de Fl ;
   Fl:= Fl \ {Il} ;
   I2:= retire (Il);
   #γ := retire (Il);
   c:= retire (Il) ;
   para i:= 0 até 3 faça bor [i]:= falso ;
   enquanto não vazio (I2) faça
    início
     p:= retire (I2) ;
     b:= borda (p,c) ;
     bor [b]:= verdadeiro ;
     A #γ := A #γ ∪ {ares (c,b)}  ;
     P #γ := P #γ ∪ {extf(c,b)}
    fim;
   de:= se (bor[3] e bor[0] e bor[1] ) ou
         ( não (bor[2] ) e bor [3] e bor [0])
         então 3
         senão min {i/ 0 ≤ i ≤ 3  e bor [i] é verdadeiro}  ;
   a:= de;
   b:=(a+1) mod 4;
```

```
    enquanto b ≠ de faça
     início
      se bor [b] então a:=b ;
      b:= (b+1) mod 4
     fim ;
    de:= extf(c,de) ;
    a:= extf(c,a) ;
    enquanto de ≠ a faça
     início
      v:= extah(de,c) ;
      A #γ := A #γ ∪ {(de,v)} ;
      P #γ := P #γ ∪ {v} ;
      de:= v
     fim
   fim
fim
```

PROCEDIMENTO B13

MANIPULADOR DE FALHAS II

```
início
 enquanto F2 ≠ ∅ faça
  início
   I1:= elemento de F2 ;
   F2:= F2 \ {I1} ;
   dest:= retire(I1);
   orig:= retire(I1);
   #γ:= retire (I1);
   c:= retire(I1);
   b1:= borda(c,dest);
   b2:= borda(c,orig);
   se não pino(c,dest)
    então A#γ :=A γ\{ares(c,b1)}
    senão se não pino(c,orig)
      então A#γ:=A#γ \{ares(c,b2)} ;
   se b1 > b2
    então início
      b:=b1 ;
      b1:=b2;
      b2:=b ;
     fim ;
   se (b1= 0) e (b2=3)
    então início
```

```
        b1:= 3 ;
        b2:= 0
       fim ;
    v1:= extf(c,b1) ;
    v2:= extf(c,b2) ;
    enquanto v1 ≠ v2 faça
     início
      v:= extah(v1,c) ;
      A #γ:= A #γ ∪ {(v1,v)}  ;
      P #γ:= P #γ ∪ {v};
      v1:= v
     fim
  fim
fim
```

PROCEDIMENTO B14

```
início
  (* designação de espaços para as conexões nos canais comuns
       horizontais *)
  para i:= 0 até Rm faça
   início
    al:= primh(i) ;
    cont:= 0 ;
    repita
      a:= al ;
      al:= seg(al,H_i) ;
      para #Y:= 1 até nS faça
       início
        se a ϵ A_#Y
         então início
           c:= a ;
           repita
             b:= c ;
             c:= seg(b,A_#Y ) ;
           até b = c ;
           j:= 0;
           achou:= falso ;
           enquanto não achou e j < cont faça
            início
             j:= j+1 ;
```

```
livre:= verdadeiro;
I1:= < > ;
enquanto não vazio (EH^i_j) e livre faça
 início
  I2:= retire (EH^i_j) ;
  acrescente (I2,I1);
  b2:= retire (I2);
  a2:= retire (I2);
  livre:= menorh(b,a2) ou menorh(b2,a)
 fim;
enquanto não vazio (I1) faça
 início
  I2:= retire (I1) ;
  acrescente(I2,EH^i_j)
 fim ;
se livre
 então início
   achou:= verdadeiro;
   acrescente (< #γ  a b > , EH^i_j)
  fim
fim
se não achou
 então início
  cont:= cont +1 ;
  crie.espaçoh(i,cont) ;
  EH^i_cont :=< < #γ a b >>
```

```
            fim
          fim
        fim
      até a = al ;
    dim.canalh [i] := cont
  fim
fim
```

PROCEDIMENTO B15

```
procedimento região.h(i,j) ;
 início
  SRH_i := < > ;
  inic.sub(i,j) ;
  liga.comp.canal ;
  liga.comp.comp ;
  salvar.região
 fim
```

PROCEDIMENTO B16

```
procedimento inic.sub(i,j)
 início
  zerar a matriz R ;
  f:= 0;
  v:= vértice do canal comum i na coluna j da rede;
  a:= aresta do canal i que tem v como extremo "oeste";
  b:= aresta do canal i que tem v como extremo "leste";
  c:= aresta do canal próprio j que tem v como extremo "sul";
  cnorte:= extremo "norte"de c;
  csul:= extremo "sul" de d;
  Ii:= 1; ci:=1;Is:= dim.canalh|i|; cs:= comp,
  ncomp:= comprimento do componente cnorte;
  scomp:= comprimento do componente csul;
  nI:= lista das células da borda sul de cnorte pertencentes a um
       determinado sinal, e  que tem ligação com conexão "externa".
       Os elementos de nI são listas de dois elementos. O primeiro
       é a identificação do sinal e o segundo a coluna do mini-re-
       ticulado na qual se encontra a célula;
  sI:= listas das células, pertencentes a um determinado sinal   e
       que tem ligação com conexão "externa", da borda norte de
       csul;
  para k:= 1 até Is faça
   início
    Il:= < > ;
    achou:= falso
```

```
    enquanto não vazio (EH_k^i) e não achou faça
     início
      I2:= retire (EH_k^i);
      acrescente(I2,I1);
      b1:= retire(I2);
      a1:= retire(I2);
      #γ:= retire(I2);
      det.de.aa(pertence);
      se pertence
       então início
          achou:= verdadeiro;
          para j1:= de até aa faça R[f,k,j1]:= #γ
         fim
     fim
    enquanto não vazio (I1) faça
     início
      I2:= retire (I1);
      acrescente (I1,EH_k^i)
     fim
  fim
fim
```

PROCEDIMENTO B17

```
procedimento det.de aa(pertence);
 início
  pertence:= falso;
  se al ≤ a
   então início
     se bl≥ b
      então início
        pertence:= verdadeiro;
        de:= - comp;
        aa:= 2* comp
       fim
      senão se bl= a
        então início
          pertence:= verdadeiro;
          de:= - comp;
          se ∃I3 em nI/primeiro(I3)= #γ e segundo(I3) é máximo
           então início
             aa:= segundo (I3);
             se  ∃I3 em sI/primeiro(I3) = #γ e segundo(I3) é máximo
              então início
                se segundo (I3) > aa
                 então aa:= segundo (I3)
               fim
            fim
```

```
          senão início
            determine I3 em sI/primeiro (I3)=#γ e segundo(I3) é
                                                        máximo;
            aa:= segundo(I3)
           fim
        fim
  fim
senão se b = al
  então início
    pertence:= verdadeiro ;
    aa:= 2*comp;
    se ∃I3 em nI/primeiro(I3)= #γ e segundo(I3) é mínimo
     então início
       de:= segundo(I3);
       se ∃I3 em sI/primeiro(I3)= #γ e segundo(I3) é mínimo
        então início
          se segundo(I3) < de
           então de:= segundo(I3)
          fim
       fim
      senão início
        determine I3 em sI/primeiro(I3)= #γ e segundo(I3) é mínimo;
        de:= segundo (I3)
       fim
   fim
```

PROCEDIMENTO B18

```
procedimento liga.comp.canal;
 início
  para k:= 1 até Is faça
   início
    I1:=  < > ;
    achou:= falso ;
    enquanto não vazio (EH_k^i) e não achou faça
     início
      I2:= retire (EH_k^i) ;
      acrescente(I2,I1);
      b1:= retire (I2);
      a1:= retire (I2);
      #γ:= retire(I2);
      det.de.aa(pertence);
      se pertence
       então início
         achou:= verdadeiro;
         j2:= comp;
         j3:= 1;
         I3:= < > ;
         enquanto não vazio (nI) faça
          início
           I4:= retire (nI);
           I5:= I4;
```

```
    j1:= retire(I4);
    #Y1:= retire(I4);
    se  #Y= #Y1
     então liga(k,j1,-comp,j4)
     senão acrescente (I5,I3);
    se j4  < j2 então j2:= j4;
    se j4  > j3 então j3:= j4;
   fim;
enquanto não vazio (I3) faça
 início
   I4:= retire (I3);
   acrescente(I4,nI)
 fim;
enquanto não vazio (sI) faça
 início
   I4:= retire (sI);
   I5:= I4 ;
   j1:= retire (I4);
   #Y1:= retire (I4);
   se  #Y=  #Y1
     então liga (k,j1,comp+dim.canalh[i], j4)
     senão acrescente(I5,I3);
   se j4 < j2 então j2:= j4;
   se j4 > j3 então j3:= j4;
```

```
          fim;
        enquanto não vazio (I3) faça
         início
          I4:= retire(I3) ;
          acrescente(I4,sI)
         fim ;
        se j2 > de e de > -comp
         então início
           para j4:= de até j2-1 faça R[f,k,j4]:= 0
          fim;
         se j3 < aa e aa < 2*comp
          então início
            para  j4:=j3+1 até aa faça R[f,k,j4]:= 0
           fim
        fim
      fim;
    enquanto não vazio (I1) faça
     início
      I2:= retire (I1);
      acrescente(I2,EH_k^i)
     fim
  fim
fim
```

PROCEDIMENTO B19

```
procedimento liga (If,ks,Is,cf);
 início
  desv:= 0; cf:=ks;Iii:=Is;
  inc:= se Is = -comp então 1 senão -1 ;
  enquanto Iii ≠ If+inc e R[1,Iii,cf]= 0 faça Iii:= Iii+inc;
  Iii:= Iii-inc;
  se Iii = If
   então início
     Iii:=Is ;
     R|1,Iii,cf|:= #γ;
     enquanto Ii ≠ If faça
      início
       Iii:= Iii+inc;
       R[1,Iii,cf]:= #γ;
      fim;
     FRHi,j:=FRHi,j ∪  {< If,cf >}       (*conjunto de furos na
          região i,j*)
     se If > Is então  Is:=If ;
     se If < Ii então  Ii:=If ;
    fim
   senão início
     achou:= falso; imp[1]:= falso; imp[-1]:= falso ;
     nova:= verdadeiro;
     enquanto não achou faça
```

```
início
 desv:= desv + 1; incl:= 1; cont:= 0 ;
 enquanto  cont < 2 faça
  início
   cf:= ks + desv * incl ;
   imp[incl]:= se imp[incl]
     então verdadeiro
     senão se cont = 0
       então cf > aa
       senão cf < de,
  se não imp[incl]
    então início
      se nova
       então início
         possível:= R[0,Iii,ks]= 0;
         jl:= ks ;
         enquanto possível e jl ≠ cf faça
          início
           jl:=jl + incl ;
           possível := R[0,Iii,jl] =0 ;
          fim
        fim
       senão possível:= R[0,Iii,cf]= 0;
      se possível
```

```
então início
  il:= Iii
  possível:= R[1,il,cf] = 0;
  enquanto possível e il ≠ If faça
   início
    il:=il+inc;
    possível:= R[1,il,cf] = 0
   fim;
  se possível
   então início
     achou:= verdadeiro ;
     cont:= 2;
     il:= Is;
     R[1,il,ks] := #X;
     enquanto il ≠ Iii faça
      início
       il:=il+inc ;
       R[1,il,ks]:= #Y
      fim ;
     FRH_i,j := FRH_i,j ∪ { <Iii,ks >} ,
     jl:= ks ;
     R[0,Iii,jl]:=    #Y;
     enquanto jl ≠ cf faça
      início
       jl:=jl+incl;
```

```
                R[0,Iii,jl]:= #γ
               fim ;
              FRH_{i,j}:= FRH_{i,j}  ∪  {<Iii,cf>} ;
              il:= Iii;
              R[1,il,cf]:=  #γ;
              enquanto il ≠ If faça
               início
                il:=il+inc;
                R[1,il,cf]:=  #γ
               fim ;
              FRH_{i,j}:= FRH_{i,j} ∪  {<If,cf >};
              se If > Is então Is:= If;
              se If < Ii então Ii:= If ;
              se Iii > Is então Is:= Iii ;
              se Iii < Ii então Ii:= Iii ;
              se cf > cs então cs:= cf;
              se cf < ci então ci:= cf;
             fim
          fim
         senão imp[incl]:= verdadeiro
       fim;
     cont:= cont+1 ;
     incl:= -1
   fim ;
```

```
        se imp[1] e imp[-1]
         então início
           nova:= verdadeiro ;
           imp[1]:= falso ; imp[-1]:= falso ;
           Iii:= Iii-inc
          fim
         senão nova := falso
       fim
    fim
fim
```

PROCEDIMENTO B20

```
procedimento liga.comp.comp ;
 início
  I1:= < > ;
  final:= vazio (nI) e vazio (sI) ;
  enquanto não final faça
   início
    se não vazio (nI)
     então início
       I2:= retire (nI);
       j1:= retire (I2);
       #Y:= retire (I2);
       LN:={j1};
       I3:= < >;
       enquanto não vazio (nI) faça
        início
         I2:= retire (nI);
         I4:= I2 ;
         j1:= retire (I4);
         #Y1:= retire (I4);
         se  #Y= #Y1
          então LN:=LN ∪ {j1}
          senão acrescente (I2,I3)
        fim;
       nI:=I3;
```

```
    LS:= ∅
   fim
  senão início
    LN:= ∅ ;
    I2:= retire (sI);
    j1:= retire (I2);
    #Y:= retire (I2);
    LS:= {j1}
   fim;
I3:= < > ;
enquanto não vazio (sI) faça
 início
   I2:= retire (sI);
   I4:= I2;
   j1:= retire (I2);
   #Y1:= retire(I2);
   se  #Y= #Y1
    então LS:= LS ∪ {j1}
    senão acrescente (I4,I3)
 fim;
sI:=I3;
se LN  ≠ ∅
 então início
   se LS ≠ ∅
```

```
    então início
      maxn:= max(LN);minn:= min(LN);
      maxs:= max(LS);mins:= min(LS);
     fim
    senão início
      maxn:= max(LN);maxs:= maxn;
      mins:= min(LN);mins:= minn;
     fim
  fim
 senão início
   maxs:= max(LS); maxn:= maxs;
   mins:= min(LS); minn:= mins;
  fim;
de:= min({minn,mins});
aa:= max({maxn,maxs});
il:= 0 ;
achou:= falso;
enquanto não achou faça
 início
  il:= il+1;
  jl:= - comp;
  possível := R[0,il,jl]= 0 ;
  enquanto possível e jl < aa faça
   início
    jl:= jl+1 ;
```

```
  possivel:= R[0,il,jl] = 0
 fim;
se possível
 então início
   achou:= verdadeiro ;
   de:= - comp
  fim;
 senão início
   jl:= de;
   possível:= R[0,il,jl]= 0;
   enquanto possível e jl < 2* comp faça
    início
     jl:= jl+1;
     possível:= R[0,il,jl]= 0
    fim ;
   se possível
    então início
      achou:= verdadeiro;
      aa:= 2*comp
     fim
 fim;
j2:= aa;j3:=de;
para ∀jl ∈ LN faça
```

```
      início
        liga (il,jl,-comp,j4);
        se j4  < j2 então j2:= j4;
        se j4  > j3 então j3:= j4;
       fim;
      para ∀jl ∈ LS faça
       início
        liga(il,jl,comp+dim.canalh[i],j4) ;
        se j4 < j2 então j2:=j4;
        se j4 > j3 então j3:=j4;
       fim;
      para jl:=j2 até j3 faça R[0,il,jl]:=  #Y
     fim;
    final:= vazio (nI) e vazio(sI)
  fim
fim
```

PROCEDIMENTO B21

```
procedimento salvar.região;
 início
  acrescente ( <Ii ci Is cs R > , SRH_i )
 fim
```

APÊNDICE C

APÊNDICE C

SUBÁRVORES DE CUSTO MÍNIMO

A primeira parte do algoritmo proposto determina subárvores no grafo representativo do macro-reticulado. Neste apêndice demonstra-se em C1 que o problema de obter a subárvore de custo mínimo em um grafo é NP-completo ou exponencial. Portanto, desconhece-se um algoritmo viável para resolvê-lo. Este fato justifica o uso de algoritmos heurísticos que resultem em subárvores satisfatórias. Em C2 são apresentados dois desses algoritmos heurísticos; ambos são polinomiais de ordem n^3 onde n é o número de vértices do grafo.

C1 - Subárvore do Tipo Steiner de Custo Mínimo

Definições:

Árvore Mínima de Steiner(1) para os pontos $P_1, P_2, \ldots, P_n$ no plano é uma árvore que interconecta estes pontos usando retas tal que o comprimento total seja o mínimo possível. A fim de conseguir o comprimento mínimo, a árvore mínima de Steiner poderá conter outros vértices (pontos de Steiner) além de P_1, $P_2, \ldots, P_n$. Assume-se que os pontos de Steiner não introduzem custo adicional.

Distância Retilínea ou de Manhattan(2,3)

entre dois pontos P_1 e P_2 de coordenadas (x_1,y_1) e (x_2,y_2) respectivamente é designada por $d(P_1,P_2)$ e é definida como:

$$d(P_1,P_2) = |x_1-x_2| + |y_1-y_2|$$

O conceito de árvore de Steiner de custo / mínimo é aqui estendido a subárvores de um grafo através da seguinte definição:

Subárvores do Tipo Steiner de Custo Mínimo: Seja G um grafo conexo com um conjunto de vértices V(G) e um conjunto de arestas A(G) . A cada aresta V(G) está associado um custo positivo e seja $V' \subseteq V(G)$. Uma subárvore T de G cujo conjunto de vértices contém V' ($V' \subseteq V(T)$) e cuja soma dos custos das arestas em A(T) é mínima é chamada de subárvore do tipo Steiner de custo mínimo induzida em G por V'.

Teorema:

O problema da determinação da subárvore do tipo Steiner de custo mínimo é NP-completo ou exponencial.

Demonstração:

Considere-se o problema da árvore retilínea de Steiner: trata-se de obter uma árvore de comprimento mínimo em um reticulado retangular de passo fixo que contenha um determinado conjunto P de vértices (intersecções) utilizando somente retas horizontais e verticais. Este problema é NP-completo (4).

Basta pois, para demonstrar o teorema, re-

duzir o problema da subárvore do tipo de Steiner de custo mínimo ao problema da árvore retilínea de Steiner.

Ao reticulado retangular de passo fixo R associa-se um grafo G cujos vértices derivam das intersecções entre retas verticais e horizontais do reticulado e cujas arestas derivam dos segmentos de reta entre as intersecções. Às arestas associa-se um peso proporcional igual ao passo do reticulado. Seja $P = \{P_1, P_2, \ldots, P_n\}$ um conjunto de intersecções em R, ao qual corresponde em G o conjunto de vértices $V' = \{V_1, V_2, \ldots, V_n\} \subseteq V(G)$. A subárvore do tipo de Steiner de custo mínimo em G reduzida por V' corresponde em R à árvore retilínea de Steiner relativo ao conjunto P.

C2 - Subárvores Alternativas

Nesta secção subentende-se por um grafo G um grafo conexo com rótulos associados às arestas, que representam o custo ou a distância de se passar de um de seus extremos a outro através desta aresta. A seguir apresenta-se o algoritmo de Dijkstra que determina a distância do menor caminho de um vértice u_0 do grafo a qualquer outro (5,18). A notação é a mesma dada em 2.1.1.

início

$I(u_0) := 0$;

para todo $v \in V(G)$ e $v \neq u_0$ faça $I(v) := \infty$;

$S := \{u_0\}$;

$u := u_0$;

```
enquanto  S̄ ≠ ∅  faça
 início
  para todo v ∈ S̄ faça I(v):= min {I(v),I(u)+ r(v,u)}  ;
  redefina u tal que I(u) = min {I(v)} ;
  S:= S ∪ {u}
 fim
fim
```

Ao fim deste algoritmo foi atribuido a cada vértice um rótulo que corresponde à distância mínima desse vértice ao vértice u_0. Partindo desta rotulação dos vértices é também possível determinar facilmente um caminho mínimo de um vértice qualquer a u_0.

Algoritmo C21: Determina uma subárvore mínima em relação à uma raiz r induzida por um conjunto de vértices D. Seja $r \in V(G)$, $D \subset V(G)$ e $r \notin D$.

```
1 início
2  P:= {r}  ;
3  I(r):= 0;
4  para todo v ∈ V(G) e v ≠ r faça I(v):= ∞ ;
5  S:= {r} ;
6  u:= r
7  B:= ∅ ;
8  enquanto D ≠ ∅ faça
9   início
```

```
10    para todo v ∈ S̄ faça I(v):= min {I(v),I(v)+r(u,v)} ;
11    redefina u tal que I(u) = min  {I(v)} ;
                                v∈S̄
12    se u ∈ D
13     então  início
14       D:=D \{u} ;
15       w:= u ;
16       enquanto w ∉ P faça
17        início
18         P:= P ∪ {w} ;
19         determine v tal que I(v) = I(w)-r(w,v) ;
20         B:= B ∪ {(w,v)} ;
21         w:= v
22        fim
23      fim
24    S:= S ∪ {u} ;
25   fim
26 fim
```

Fundamentalmente foi adicionado ao algoritmo de Dijkstra o trecho da linha 12 a 23 no qual é determinado o caminho mínimo do vértice de D encontrado a um dos vértices da subárvore. Os vértices e arestas deste caminho mínimo são adicionados / aos conjuntos de vértices P e de arestas B respectivamente. Depois da execução do algoritmo tem-se em P todos os vértices e em B todas

as arestas da subárvore mínima em relação a raiz r induzida por D.

Algoritmo C22: Variante do algoritmo precedente para obter uma subárvore mínima em relação à subárvore já construída induzida por D . Neste caso determina-se um vértice do conjunto D mais próximo à subárvore em construção e incorpora-se o caminho mínimo entre este vértice e a subárvore à mesma. Seja $u_0 \in V(G)$, $D \quad V(G)$ e $u_0 \notin D$.

```
1  início
2   P:= {u0} ;
3   B:= ∅
4   enquanto D ≠ ∅ faça
5    início
6     para todo v ε P faça I(v):= 0;
7     para todo v ε P̄ faça I(v):= ∞;
8     S:= {u0} ;
9     u:= u0  ;
10    objetivo:= falso
11    enquanto não objetivo faça
12     início
13      para todo v ε S̄ faça I(v):= min{I(v),I(v)+r(v,u)} ;
14      redefina u tal que I(u) = min {I(v)}  ;
                                  v∈S̄
15      se  u ε D
16       então início
17         objetivo:= verdadeiro ;
```

```
18          D:= D ∖ {u} ;
19          w:= u ;
20          enquanto w ∉ P  faça
21           início
22            P:= P ∪ {w};
23            determina v tal que I(v) = I(w)-r(w,v) ;
24            B:= B ∪ {(w,v)};
25            w:= v
26           fim
27         fim
28        senão S:=S∪{u};
29      fim
30    fim
31 fim
```

A diferença básica deste algoritmo (C22) em relação ao anterior (C21) é o trecho entre a linha 6 e a linha 10. / Toda vez que um vértice em D ainda não incorporado à subárvore é encontrado, os vértices em P (conjunto dos vértices da subárvore já determinados) são remarcados com o rótulo zero e os demais com o rótulo ∞ sendo o processo de rotulação novamente iniciado a partir do vértice u_0. Isto tem como resultado o estabelecimento, sempre que um vértice de D é encontrado, do caminho mínimo deste vértice à subárvore já existente sendo este caminho mínimo incorporado à subárvore (linhas 17 a 26). O único caminho na subárvore (pois não existem

ciclos) de u_0 a este vértice (o último vértice em D encontrado) / não é necessariamente mínimo em G. No final da execução deste algoritmo tem-se em P todos os vértices e em B todas as arestas da subárvore.

APÊNDICE D

APÊNDICE D

MANUAL DO USUÁRIO DO

SISTEMA DE TRACADO DE ROTAS

PARA CIRCUITO IMPRESSO

I - INTRODUCAO.

OS PROGRAMAS DO SISTEMA BASEIAM-SE NOS ALGORITMOS DESCRITOS NA TESE DE MESTRADO "TRACADO COMPLETO DE ROTAS EM PLACAS DE DIMENSOES NAO PRE-FIXADAS", UNICAMP-IMECC, 1980, E FORAM DESENVOLVIDOS NO DEC-10 DO CENTRO DE COMPUTACAO DA UNICAMP. O ALGORITMO NAO REQUER UMA POSICAO PRE-FIXADA DOS COMPONENTES, MAS UTILIZA-SE DE UMA REDE DE COMPONENTES, COM CANAIS DE CONEXOES DE CAPACIDADE "ILIMITADA" ENTRE OS MESMOS, CHAMADA DE MACRO-RETICULADO.

O SISTEMA CONSISTE ATUALMENTE DE QUATRO PROGRAMAS:

P1: OBTEM PARA CADA SINAL UM CONJUNTO DE SEGMENTOS DE CANAIS NO MACRO-RETICULADO REPRESENTANTE DE CONEXOES "EXTERNAS" (ENTRE COMPONENTES), E AS AREAS SOB OS COMPONENTES (MINI-RETICULADOS) COM AS RESPECTIVAS CONEXOES DESIGNADAS PARA ESTAS AREAS.

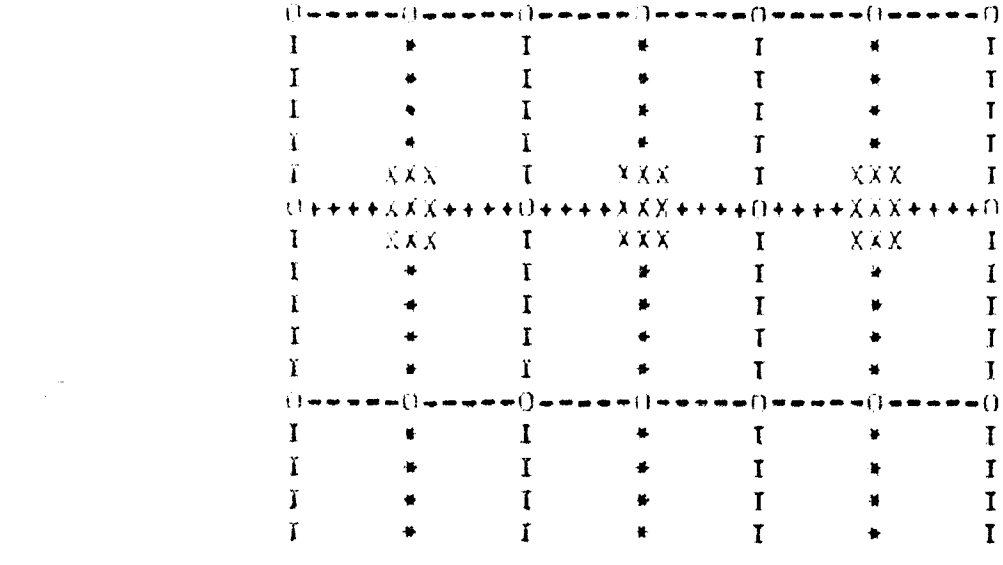
P2: DETERMINA ESPACOS PARA AS CONEXOES DO MACRO-RETICULADO E, CONSEQUENTEMENTE, AS DIMENSOES FINAIS DA PLACA.

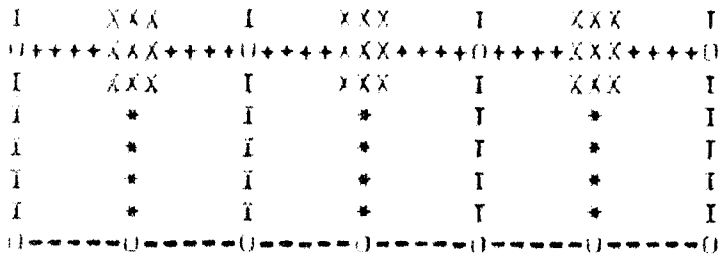
P3: CONSTROE A PLACA, A PARTIR DAS INFORMACOES OBTIDAS ANTERIORMENTE, USANDO UMA REPRESENTACAO DE RETICULADO DE PASSO FIXO E GERA UM CODIGO PARA CADA CELULA REPRESENTANDO OS SEGMENTOS DE CONEXOES E POSSIVEL FURO COMPORTADO PELA CELULA.

P4: INTERPRETA O CODIGO CELULAR E GERA O ARQUIVO DE COMANDOS PARA O PLOTADOR CAI430, QUE PERMITE DESENHAR O TRACADO PARA FINS DE VISUALIZACAO.

II - DESCRICAO INICIAL DA PLACA.

O ALGORITMO PRODUZ PLACAS DE DUAS FACES. NA FACE DENOMINADA ZERO (SUPERIOR) ENCONTRAM-SE OS CANAIS HORIZONTAIS E NA FACE UM (INFERIOR) OS CANAIS VERTICAIS. PARA UMA REDE DE DUAS LINHAS POR TRES COLUNAS DE COMPONENTES O MACRO-RETICULADO SERIA:





LEGENDA:

CANAIS HORIZONTAIS COMUNS: -
CANAIS HORIZONTAIS PROPRIOS: +
CANAIS VERTICAIS COMUNS: I
CANAIS VERTICAIS PROPRIOS:*

```
      XXX
COMPONENTE: XXX
      XXX
```

INTERSECCAO DE CANAIS (EXCETO COMPONENTE): O

OS COMPONENTES SAO IDENTIFICADOS DE ACORDO COM A SUA ENUMERACAO POR LINHA DA ESQUERDA PARA A DIREITA NO MACRO-RETICULADO. A PRIMEIRA LINHA E' A LINHA MAIS ACIMA. AS BORDAS DA PLACA SAO ENUMERADAS NO SENTIDO ANTI-HORARIO SENDO A BORDA INFERIOR A BORDA ZERO, A BORDA A DIREITA A BORDA UM, ETC.

A REPRESENTACAO DE CADA TIPO DE COMPONENTE E' DADA POR UM CONJUNTO DE DOIS RETICULADOS (UM PARA CADA FACE DA PLACA) DE PASSO FIXO CHAMADO DE MINI-RETICULADO. A LOCALIZACAO DOS PINOS E' DADA POR COORDENADAS (LINHA,COLUNA) SENDO QUE O ALGORITMO EXIGE OS PINOS NA BORDA DO MINI-RETICULADO. OS PINOS DOS COMPONENTES SAO ENUMERADOS DE ACORDO COM A SUA OCORRENCIA NA LISTA DE PINOS DADA NA DESCRICAO DO TIPO DO COMPONENTE. UM PINO OCORRE SIMULTANEAMENTE NA MESMA LINHA E COLUNA NOS DOIS RETICULADOS. CELULAS NAO OCUPADAS POR PINOS PODEM OU NAO SER OBSTRUIDAS PARA PASSAGEM DE CONEXAO. AS BORDAS DOS MINI-RETICULADOS SAO ENUMERADAS DE MANEIRA ANALOGA AS BORDAS DA PLACA SENDO A ENUMERACAO DE LINHAS E COLUNAS NO MINI-RETICULADO FEITA DE ACORDO COM O ESQUEMA ABAIXO. SUPOE-SE A EXISTENCIA DE M LINHAS E N COLUNAS.

```
  1 2 3 ... N
M             M
.             .
.             .
.             .
3             3
2             2
1             1
  1 2 3 ... N
```

CADA COMPONENTE E' DESCRITO ATRAVES DA ESPECIFICACAO DE SEU TIPO E ORIENTACAO. HA' QUATRO POSSIVEIS ORIENTACOES SENDO CADA UMA REPRESENTADA POR UM NUMERO ENTRE ZERO E TRES. ORIENTACAO ZERO SIGNIFICA QUE O COMPONENTE OCORRE NA PLACA DA MESMA FORMA COMO ESTA' NA DESCRICAO DO TIPO. ORIENTACAO UM SIGNIFICA QUE A REPRESENTACAO DADA PELO TIPO E' GIRADA POR 90 GRAUS NO SENTIDO ANTI-HORARIO, DOIS SIGNIFICA UM GIRO DE 180 E TRES UM GIRO DE 270 GRAUS.

OS SINAIS SAO DADOS POR UMA LISTA DE PARES (NUMERO DE COMPONENTE, NUMERO DE PINO) CHAMADOS PONTOS, ONDE NUMERO DE COMPONENTE E NUMERO DE PINO SAO A IDENTIFICACAO DE COMPONENTE E PINO RESPECTIVAMENTE DE ACORDO COM A ENUMERACAO DOS MESMOS.

III - OS PROGRAMAS E SEUS ARQUIVOS.

O PROGRAMA "P1" TEM COMO ENTRADA O ARQUIVO "PLACA" DESCRITO EM IV QUE CONTEM A ESPECIFICACAO DA PLACA (TIPO DE COMPONENTES USADOS, SINAIS, ETC). O PROGRAMA DETERMINA UM CONJUNTO DE SEGMENTOS DE CANAIS NO MACRO-RETICULADO PARA CADA SINAL. O PESO ASSOCIADO A UM SEGMENTO DE CANAL NO MACRO-RETICULADO REPRESENTA O CUSTO OU DIFICULDADE DE PASSAR UMA CONEXAO POR ESTE SEGMENTO. O PROGRAMA MINIMIZA O CUSTO DE ACORDO COM UM DE DOIS CRITERIOS ESCOLHIDOS PELO USUARIO: O PRIMEIRO MINIMIZA O CUSTO DAS CONEXOES EM RELACAO 'A ORIGEM DO SINAL E O SEGUNDO EM RELACAO AO CONJUNTO DE SEGMENTOS EM CONSTRUCAO. OS ARQUIVOS GERADOS PELO PROGRAMA SAO:

"RELAT.TRA": CONTEM INFORMACOES SOBRE O DESEMPENHO DE ALGUMAS ROTINAS DO PROGRAMA.

"CONEX.DEF": CONTEM OS CONJUNTOS DE SEGMENTOS DE CANAIS REPRESENTATIVOS DOS SINAIS.

"CONEX.LNK": CONTEM OS APONTADORES PARA O INICIO DAS ARESTAS DE CADA SINAL EM "CONEX.DEF".

"MINI.PAR": CONTEM O NUMERO DE LINHAS E COLUNAS DA REDE DE COMPONENTES, A DIMENSAO DOS COMPONENTES, O NUMERO DE SINAIS E O NUMERO DE PALHAS SOB UM COMPONENTE RESOLVIDAS EXTERNAMENTE NUMA MESMA BORDA DO COMPONENTE.

"MINI.RET": CONTEM OS MINI-RETICULADOS.

O PROGRAMA "P2" USA COMO ENTRADA OS SEGUINTES ARQUIVOS, QUE APOS A LEITURA SAO DELIDOS: "MINI.PAR", "CONEX.LNK" E "CONEX.DEF". DE ACORDO COM A INFORMACAO CONTIDA NESTES ARQUIVOS, CRIA ESPACOS OU REGIOES ENTRE OS COMPONENTES E DISTRIBUE AS CONEXOES NESTES ESPACOS. ESTE PROGRAMA GERA OS SEGUINTES ARQUIVOS:

"LIGH.PRO": CONTEM OS ESPACOS OU PARTE DESTES ESPACOS CRIADOS ENTRE COMPONENTES VIZINHOS DE UMA MESMA LINHA NA REDE DE COMPONENTES.

"LIGV.PRO": ANALOGO PARA COMPONENTES VIZINHOS NUMA MESMA COLUNA.

"LIGM.PRO": CONTEM OS ESPACOS OU PARTE DOS ESPACOS ORIGINARIOS DOS ENTRONCAMENTOS DE CANAIS VERTICAIS COM CANAIS HORIZONTAIS.

"ONDE.PLA": E' DESCRITO EM VI E CONTEM INFORMACOES QUE POSSIBILITARAO UMA POSTERIOR CONCATENACAO DAS DIVERSAS REGIOES DA PLACA.

O PROGRAMA "P3" USA COMO ENTRADA OS SEGUINTES ARQUIVOS, QUE APOS A SUA LEITURA SAO DELETADOS EXCETO O PRIMEIRO: "ONDE.PLA", "MIGI.ROT", "LIGH.PRO", "LIGV.PRO" E "LIGI.PRO". O PROGRAMA FAZ A CONCATENACAO DAS DIVERSAS REGIOES E GERA O ARQUIVO "CODIGO.PLA" DESCRITO EM V.

O PROGRAMA "P4" INTERPRETA O CODIGO EM "CODIGO.PLA" E GERA OS COMANDOS PARA O PLOTADOR.

OS PROGRAMAS DEVEM SER EXECUTADOS DA SEGUINTE FORMA:
.RUN PROG
ONDE PROG E' O NOME DO PROGRAMA.

AS INSTRUCOES DE COMO USAR O PLOTADOR PODEM SER OBTIDAS ATRAVES DO COMANDO
.HELP EAI

IV - DESCRICAO DO ARQUIVO "PLACA".

OS ARQUIVOS NAO TEM UM FORMATO RIGIDO, EXCETO O ARQUIVO "CODIGO.PLA". NA CONSTRUCAO DO ARQUIVO "PLACA" O USUARIO PODE, PORTANTO, ARRANJAR A SEQUENCIA DOS DADOS DE MANEIRA CONVENIENTE OBJETIVANDO UMA MAIOR CLAREZA NA INFORMACAO QUE DESCREVE A PLACA.

OS SETE PRIMEIROS NUMEROS (SEIS INTEIROS E O ULTIMO REAL) SAO:
1)RM: NUMERO DE LINHAS NA REDE DE COMPONENTES;
2)RN: NUMERO DE COLUNAS NA REDE;
3)S: NUMERO DE SINAIS;
4)P: NUMERO DE PONTOS;
5)C: NUMERO DE BORDAS EM QUE SE DESEJA PROIBIR A UTILIZACAO DO CANAL COMUM JUNTO 'A BORDA.
6)T: NUMERO DE TIPOS DISTINTOS DE COMPONENTES;
7) FATOR SOMADO AO PESO DE UM SEGMENTO DE CANAL COMUM TODA VEZ QUE ELE E' INTEGRADO A UMA CONEXAO. SE FOR MAIOR DO QUE ZERO, OS SEGMENTOS MAIS OCUPADOS TENDEM A SER CADA VEZ MENOS ESCOLHIDOS NA CONSTRUCAO DAS LIGACOES DOS SINAIS AINDA NAO RESOLVIDOS E DESTA MANEIRA CONSEGUE-SE UMA DISTRIBUICAO MAIS UNIFORME DAS CONEXOES.

O RESTANTE DO ARQUIVO E' CONSTRUIDO POR SETE TABELAS QUE SERAO DESIGNADAS DE T1 A T7. A TABELA T1 CONTEM C NUMEROS INTEIROS QUE REPRESENTAM AS BORDAS EM QUE SE DESEJA PROIBIR O USO DO CANAL COMUM JUNTO 'A MESMAS.

A TABELA T2 E' CONSTITUIDA DE S ELEMENTOS DE TRES ITENS. O PRIMEIRO ITEM E' ZERO SE PARA A OBTENCAO DO CONJUNTO DE SEGMENTOS E' DESEJADA A UTILIZACAO DA MODALIDADE "DISTANCIA

MINIMA EM RELACAO 'A ORIGEM" E UM SE E' DESEJADA A UTILIZACAO DA MODALIDADE "DISTANCIA MINIMA EM RELACAO AO CONJUNTO DE SEGMENTOS EM CONSTRUCAO". O SEGUNDO E' O APONTADOR PARA O INICIO DA LISTA DOS PONTOS DO SINAL NA TABELA T3 E O TERCEIRO ITEM E' O TAMANHO DESTA LISTA DE PONTOS DO SINAL.

A TABELA DE PONTOS T3 E' CONSTITUIDA DE P ELEMENTOS DE DOIS ITENS. CADA ELEMENTO REPRESENTA UM PONTO, OU SEJA, UM PAR (NUMERO DE COMPONENTE, NUMERO DE PINO). T3 E' COMPOSTA DAS LISTAS DE PONTOS DE CADA SINAL SENDO A PRIMEIRA LISTA A LISTA DO SINAL DE NUMERO UM, A SEGUNDA DO SINAL DE NUMERO DOIS, ETC.

A SEGUIR VEM A TABELA T4 DE DESCRITORES DOS COMPONENTES. A TABELA POSSUE RM*RN ELEMENTOS DE DOIS ITENS. O PRIMEIRO ITEM REPRESENTA O TIPO DO COMPONENTE E O SEGUNDO A ORIENTACAO.

A TABELA SEGUINTE (T5) TEM T+1 ELEMENTOS DE UM UNICO ITEM. OS T PRIMEIROS ELEMENTOS CONTEM APONTADORES PARA O INICIO DA DESCRICAO DE CADA TIPO NA TABELA T6. O ULTIMO ELEMENTO DEVE SER IGUAL AO NUMERO DE ELEMENTOS DA TABELA T6 MAIS UM.

A DESCRICAO DE CADA TIPO DE COMPONENTE SE ENCONTRA NA TABELA T7. O PRIMEIRO ELEMENTO DE CADA DESCRICAO CONTEM O NUMERO DE LINHAS E DE COLUNAS DO MINI-RETICULADO. O SEGUNDO O NUMERO DE PINOS E O NUMERO DE OBSTRUCOES. SEGUEM-SE OS ELEMENTOS COM AS COORDENADAS DOS PINOS. CADA ELEMENTO CONTEM AS COORDENADAS DE UM PINO, OU SEJA, DA CELULA DA BORDA QUE CORRESPONDE AO PINO. SEGUEM-SE OS ELEMENTOS COM AS LOCALIZACOES DAS OBSTRUCOES. UM ELEMENTO DE LOCALIZACAO DE OBSTRUCAO TEM COMO ITENS O NUMERO DA FACE (ZERO OU UM) EM QUE OCORRE, O NUMERO DE LINHA E O NUMERO DE COLUNA.
EXEMPLO: UM "CHIP" DE 14 PINOS SEM CELULAS OBSTRUIDAS TEM A SEGUINTE DESCRICAO.

```
    7       13
    14      0
    1       1
    1       3
    1       5
    1       7
    1       9
    1       11
    1       13
    7       13
    7       11
    7       9
    7       7
    7       5
    7       3
    7       1
```

O COMPONENTE VAZIO SERIA POIS DEFINIDO POR

```
    0       0
    0       0
```

POR ULTIMO TEM-SE CINCO NUMEROS REAIS. O PRIMEIRO E' O PESO PCH INICIALMENTE ASSOCIADO AOS SEGMENTOS DE CANAIS COMUNS HORIZONTAIS; O SEGUNDO (PCV) E' O PESO ANALOGO A PCH MAS PARA OS SEGMENTOS DE CANAIS COMUNS VERTICAIS; O TERCEIRO E' A SOMA DOS DOIS PRIMEIROS MAIS UM NUMERO QUE SIGNIFICA A PENALIDADE DE SE PASSAR UMA CONEXAO POR UM SEGMENTO HORIZONTAL SEGUIDO DE UM SEGMENTO VERTICAL OU VICE-VERSA, ISTO E': PENALIDADE POR UMA MUDANCA DE DIRECAO, POIS IMPLICA EM FURO METALIZADO; O QUARTO E' O PESO PPA ASSOCIADO A UM SEGMENTO DE CANAL PROPRIO, QUANDO E' POSSIVEL UMA CONEXAO PELA BORDA NA QUAL INCIDE O SEGMENTO DE CANAL, MAS O PINO A SER INTERLIGADO SE ENCONTRA NUMA BORDA ADJACENTE; O QUINTO E' O PESO PPO ASSOCIADO A UM SEGMENTO DE CANAL PROPRIO QUANDO, NAS MESMAS CONDICOES, O PINO SE ENCONTRA NA BORDA OPOSTA.

A SECCAO VII DESTE MANUAL APRESENTA ALGUNS CRITERIOS QUANTO AOS VALORES A SEREM ESCOLHIDOS, NA PRATICA, PARA OS DIVERSOS PESOS QUE DEVERAO SER SEMPRE REAIS OU ZERO.

EXEMPLO DE DESCRICAO DE PLACA DE DOIS COMPONENTES: O PRIMEIRO COMPONENTE TEM TRES PINOS E O SEGUNDO DOIS. EXISTEM DOIS SINAIS: O PRIMEIRO E' O PINO UM E DOIS DO PRIMEIRO COMPONENTE MAIS O PINO UM DO SEGUNDO; O SEGUNDO SINAL SAO OS DOIS PINOS RESTANTES.

```
1
2
2
5
0
2
0.0
1          1          3
1          4          2
1          1
1          2
2          1
1          3
2          2
1          0
2          0
1
6
10
3          3
3          0
1          1
2          3
3          1
3          1
2          0
1          1
3          1
2.0
1.8
4.4
```

1.2
1.5

V - DESCRICAO DO ARQUIVO "CODIGO.PLA"

A PLACA RESULTANTE DO TRACADO E' REPRESENTADA POR DOIS RETICULADOS (UM PARA CADA FACE) DE PASSO FIXO. NORMALMENTE O VALOR ESCOLHIDO E' DE UM VIGESIMO DE POLEGADA. A CADA CELULA DE CADA RETICULADO E' ASSOCIADO UM CODIGO DE ACORDO COM OS SEGMENTOS DE CONEXAO E COM OS FUROS PRESENTE NA CELULA.

PARA O CODIGO SAO NECESSARIOS DEZ "BITS". O "BIT" MENOS SIGNIFICATIVO E' AQUI IDENTIFICADO COMO O "BIT" UM E O MAIS SIGNIFICATIVO COMO O "BIT" DEZ. O "BIT" DEZ, QUANDO IGUAL A UM, SIGNIFICA FURO METALIZADO E O "BIT" NOVE, QUANDO UM, FURO DE PINO. OS OITO "BITS" MENOS SIGNIFICATIVOS REPRESENTAM A EXISTENCIA OU NAO DOS OITO POSSIVEIS SEGMENTOS DE CONEXAO, QUE UMA CELULA PODE COMPORTAR.

UM SEGMENTO DE CONEXAO TEM UM DE SEUS EXTREMOS NO CENTRO DA CELULA E O OUTRO NA BORDA DA CELULA NUMA DAS SEGUINTES DIRECOES: NORTE(1), NORDESTE(2), OESTE(3), SUDOESTE(4), SUL(5), SUDESTE(6), LESTE(7) E NORDESTE(8), E A CADA UMA DESTAS DIRECOES E' ASSOCIADO O NUMERO DADO. PARA CADA SEGMENTO DE CONEXAO COMPORTADO NA CELULA O "BIT" RESPECTIVO E' UM.

OS DOIS PRIMEIROS NUMEROS DO ARQUIVO "CODIGO.PLA" SAO O NUMERO DE LINHAS (PM) E O NUMERO DE COLUNAS (PN) DO RETICULADO. PARA EFEITO DE ENUMERACAO DE LINHAS E COLUNAS, A LINHA MAIS AO NORTE E' A LINHA UM, A LINHA MAIS AO SUL E' A LINHA PM, A COLUNA MAIS A OESTE E' A COLUNA UM E A COLUNA MAIS A LESTE E' A COLUNA PN. SEGUEM-SE P1*PN NUMEROS REPRESENTANTES DOS CODIGOS DA FACE ZERO DADOS POR LINHA E FINALMENTE OUTROS PM*PN NUMEROS REPRESENTANTES DOS CODIGOS DA FACE UM TAMBEM DADOS POR LINHA.

VI - DESCRICAO DO ARQUIVO "ONDE.PLA".

ESTE ARQUIVO CONTEM INFORMACAO SOBRE A LOCALIZACAO FINAL DE COMPONENTES E CANAIS NA PLACA.

INICIALMENTE TEM-SE TRES NUMEROS INTEIROS:
1)RM: NUMERO DE LINHAS NA REDE DE COMPONENTES;
2)RN: NUMERO DE COLUNAS NA REDE;
3)S: NUMERO DE SINAIS.

SEGUE-SE UMA MATRIZ M1 DE RM*2+1 LINHAS POR RN COLUNAS ENUMERADA POR LINHA. A LINHA RM*2+1 CONTEM SOMENTE ZEROS. O ELEMENTO M1[I,J] CONTEM A ALTURA DO COMPONENTE NA LINHA I/2 E COLUNA J DA REDE SE I E' PAR. SE I E' IMPAR, CONTEM A DIFERENCA ENTRE O VALOR MAXIMO DA LINHA I+1 DE M1 E O VALOR EM M1[I+1,J].

SEGUE-SE UMA MATRIZ M2 DE RM LINHAS POR RN*2+1 COLUNAS ENUMERADA POR LINHAS. A COLUNA 1 CONTEM SOMENTE ZEROS. O ELEMENTO M2[I,J] CONTEM O COMPRIMENTO DO COMPONENTE NA LINHA I E COLUNA J/2 DA REDE SE J E' PAR. SE J E' IMPAR, ENTAO CONTEM A DIFERENCA ENTRE O COMPRIMENTO MAXIMO NA COLUNA J-1 E O VALOR EM M2[I,J-1].

SEGUEM-SE DOIS VETORES V3 E V4 DE RM+1 POSICOES. A SOMA DE V3[I] COM V4[I] REPRESENTA A ALTURA DO ESPACO OCUPADO PELAS CONEXOES ENTRE A LINHA I E I-1 NA REDE.

POR ULTIMO APARECEM DOIS VETORES V5 E V6 DE RN+1 POSICOES. A SOMA DE V5[J] COM V6[J] REPRESENTA O COMPRIMENTO DO ESPACO OCUPADO ENTRE A COLUNA J E J-1 DA REDE DE COMPONENTES.

VII - DIRETRIZES E RECOMENDACOES.

O OBJETIVO DESTE CAPITULO E' AUXILIAR O USUARIO NA DEFINICAO DO ARQUIVO "PLACA" PARA EXPLORAR CERTAS CARACTERISTICAS DO ALGORITMO E OBTER UM MELHOR RESULTADO NO TRACADO. O PROGRAMA NAO FAZ A CONSISTENCIA DESTE ARQUIVO E COMPETE, PORTANTO, AO USUARIO CERTIFICAR-SE DE QUE A DEFINICAO DA PLACA ESTA' CORRETA.

QUANTO MAIS HOMOGENEA FOR A LARGURA (ALTURA) DOS COMPONENTES DE UMA MESMA COLUNA (LINHA), MELHOR SERA' O RESULTADO TRACADO. SUGERE-SE AGRUPAR COMPONENTES PEQUENOS E CONSIDERA'-LOS COMO SENDO UM UNICO COMPONENTE. NAO DEVE SER ESQUECIDO QUE OS PINOS DEVEM ESTAR NA BORDA DO MINI-RETICULADO.

RECOMENDA-SE O USO DA MODALIDADE "DISTANCIA MINIMA EM RELACAO 'A ORIGEM" SOMENTE EM SINAIS EM QUE O COMPRIMENTO DAS CONEXOES EM RELACAO A UM PINO (PONTO) E' CRITICA.

PARA DETERMINAR OS PESOS DOS SEGMENTOS DE CANAIS DEVE SE TER EM MENTE QUE OS SEGMENTOS DE CANAIS PROPRIOS TEM PESO UM SE O SEGMENTO INCIDE NA BORDA NA QUAL SE ENCONTRA UM PINO A SER INTERLIGADO E PESO "INFINITO" QUANDO NAO E' ACESSIVEL.

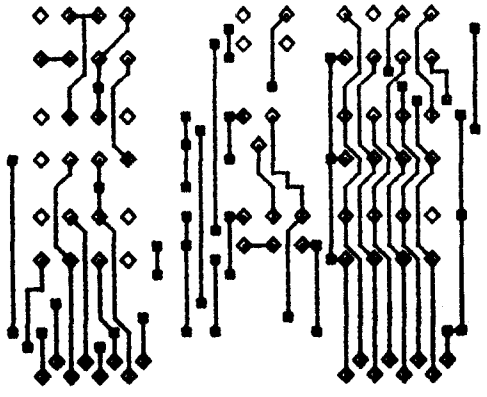
O PROGRAMA P2 PRIMEIRAMENTE ABRE ESPACOS PARA AS CONEXOES HORIZONTAIS E DEPOIS PARA AS VERTICAIS. CONSEQUENTEMENTE NAO E' POSSIVEL DETERMINAR COM PRECISAO A EXTREMIDADE DE UMA CONEXAO HORIZONTAL, QUE INTERCEPTA UMA CONEXAO VERTICAL, O QUE TORNA A OCUPACAO DOS ESPACOS ENTRE DUAS FILEIRAS HORIZONTAIS DE COMPONENTES MAIS BAIXA DO QUE OS ESPACOS ENTRE FILEIRAS VERTICAIS. POR ESTE MOTIVO O RESULTADO DO TRACADO GERALMENTE E' MELHOR SE OS PESOS DOS SEGMENTOS DE CANAIS HORIZONTAIS SAO LIGEIRAMENTE MAIORES DO QUE OS PESOS DOS SEGMENTOS DE CANAIS VERTICAIS.

****

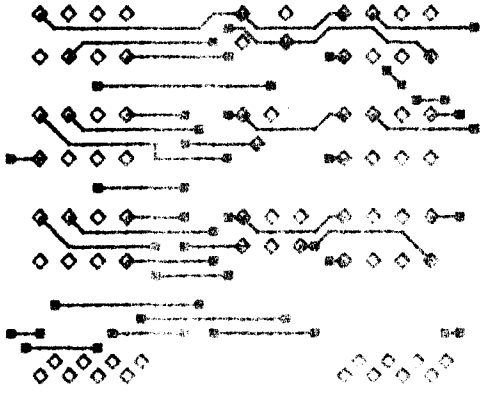
APÊNDICE E

APÊNDICE E

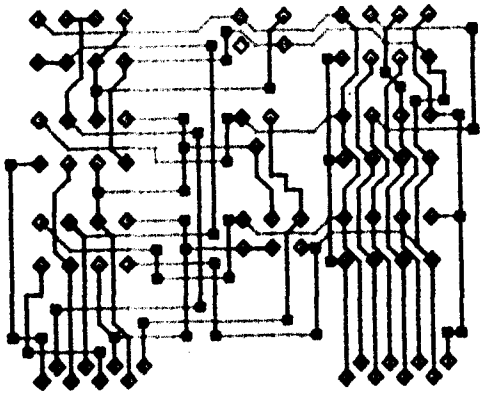
Placa I: 11 componentes



Face Inferior

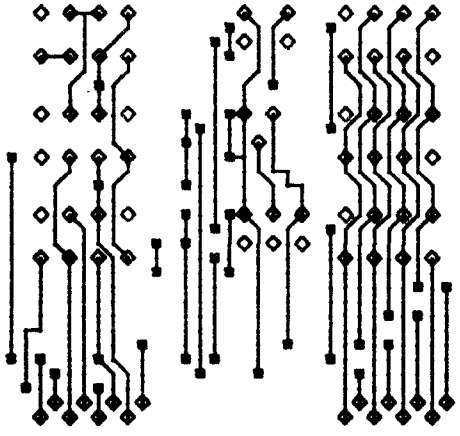


Face Superior

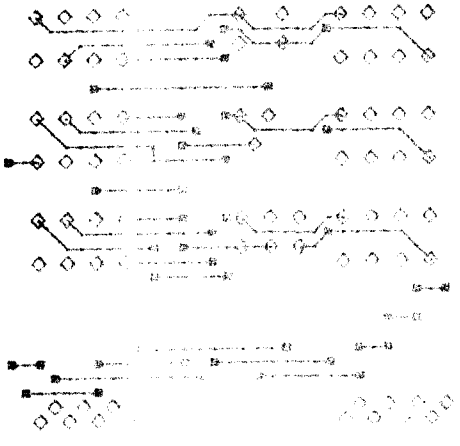


Ambas

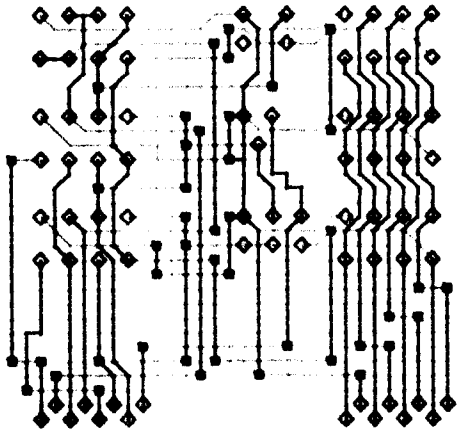
Placa 2: Mesma especificação da Placa 1 com os sinais resolvidos em outra ordem



Face Inferior

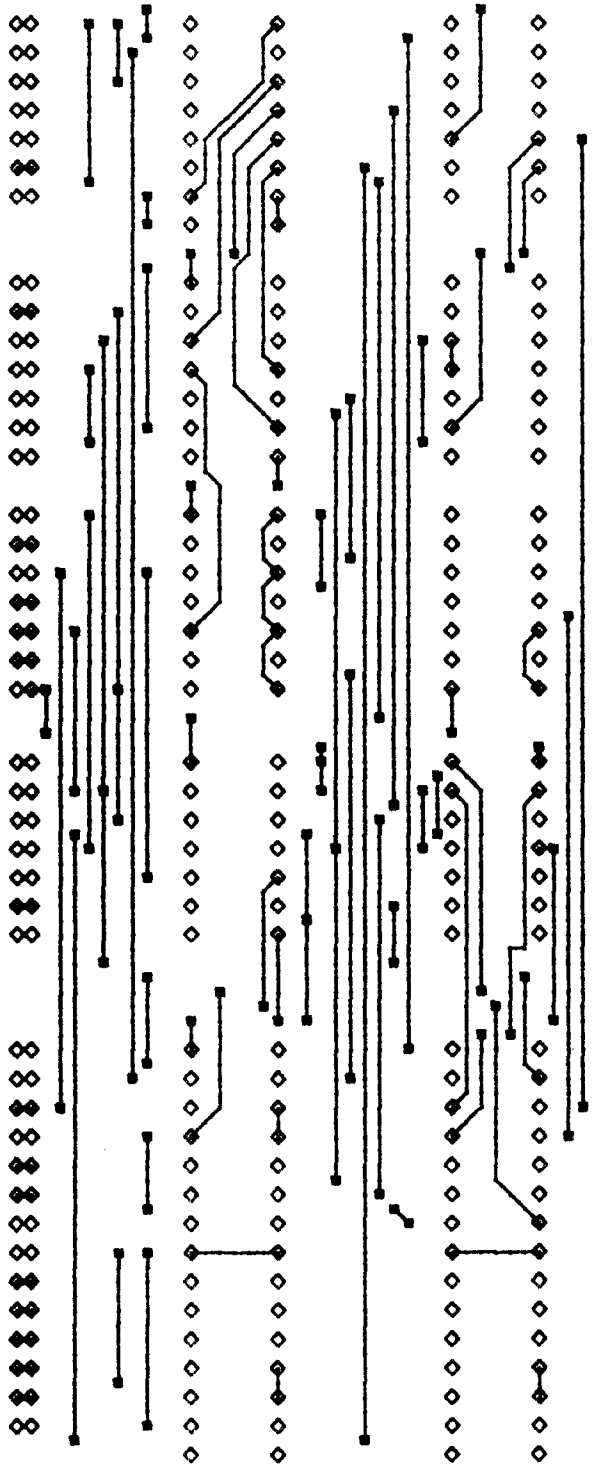


Face Superior

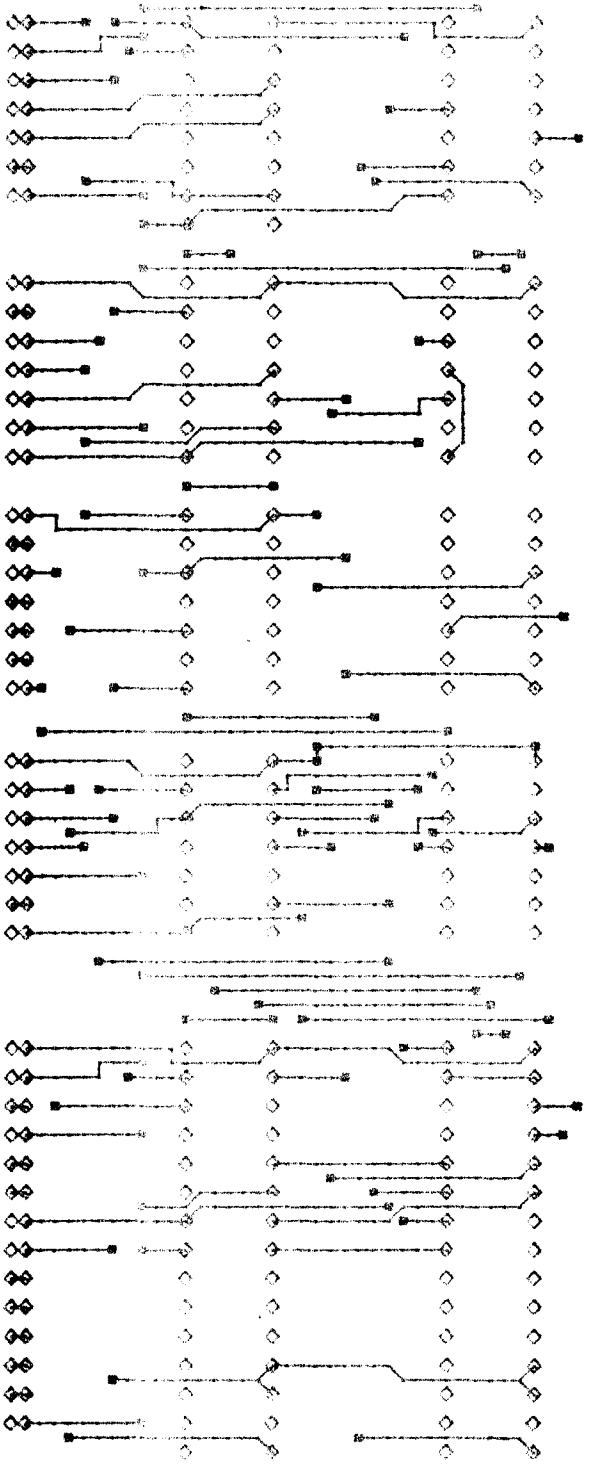


Ambas

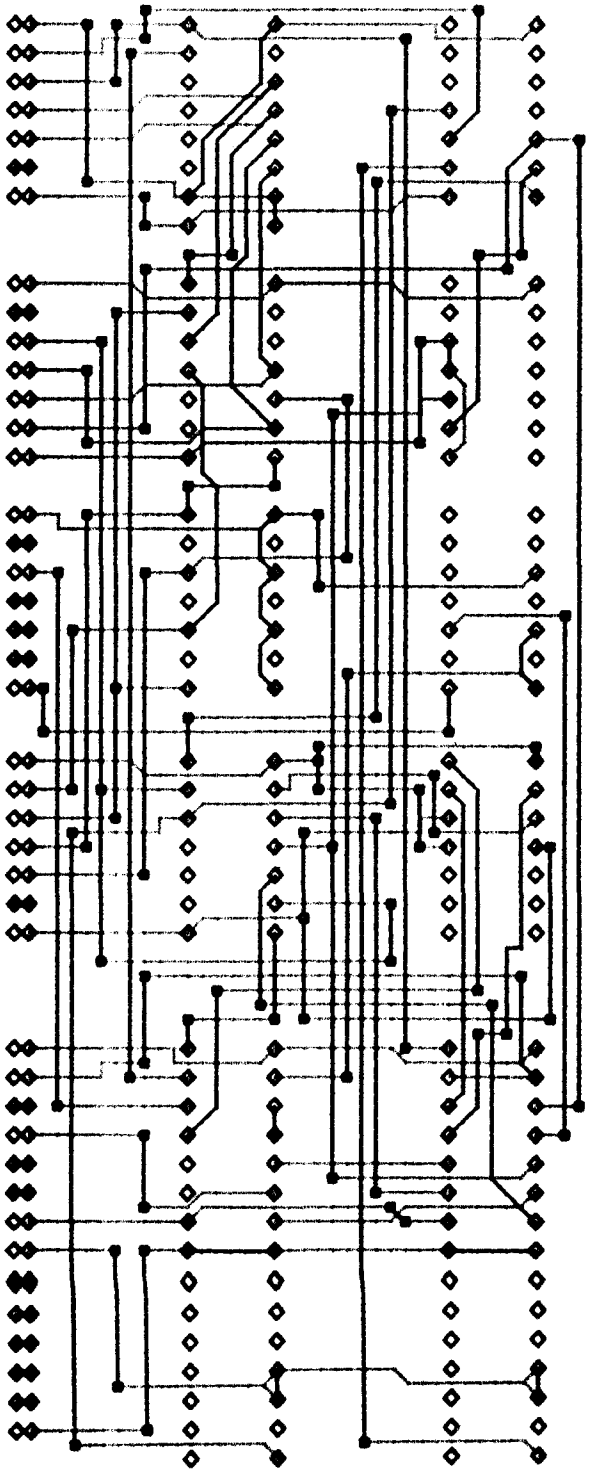
Placa 3: 18 componentes



Face Inferior



Face Superior



Ambas