ESTRUTURAS DE DADOS E ALGORITMOS DE UM EDITOR GRAFICO PARA PROJETOS VLSI

J. A. S. BORGES*

O. V. S. PIRES**

RESUMO

ESTE TRABALHO TEM POR FINALIDADE DESCREVER ESTRUTURAS DE DADOS E ALGORITMOS UTILIZADOS POR UM EDITOR GRÁFICO HIERÁRQUICO PARA PROJETOS DE CIRCUITOS INTEGRADOS VLSI, DESENVOLVIDO NO NCE/UFRJ.

ABSTRACT

THIS WORK INTENDS TO DESCRIBE DATA STRUCTURE AND ALGORITHMS USED IN A HIERARCHICAL GRAPHIC EDITOR FOR INTEGRATED VLSI CIRCUITS DESIGN, DEVELOPED AT NCE/UFRJ.

* INFORMATICO (1980 - UFRJ); PROCESSAMENTO GRÁFICO; ESTAÇÕES DE CAD; UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, NÚCLEO DE COMPUTAÇÃO ELETRÔNICA , CAIXA POSTAL 2324, RIO DE JANEIRO, CEP 20001.

** GRADUANDO EM INFORMÁTICA (UFRJ); SOFTWARE BÁSICO; SISTEMAS OPERACIONAIS COMPILADORES; CAD PARA VLSI; UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, NÚCLEO DE COMPUTAÇÃO ELETRÔNICA, CAIXA POSTAL 2324, RIO DE JANEIRO, CEP 20001.

1. INTRODUÇÃO

Este trabalho descreve o funcionamento de um editor gráfico hierárquico para projetos de circuitos integrados VLSI, conforme especificado em (Borges-85).

Durante a concepção deste editor, tivemos como objetivo principal a fácil interação com usuário; achamos interessante que o EDCI pudesse dialogar com o projetista de maneira essencialmente gráfica. Para isto, ele se utiliza de duas telas: uma gráfica, onde se desenham as porções solicitadas do lay-out, e uma tela alfanumérica, onde são mostradas diversas informações sobre o estado atual da edição. Na tela gráfica, os elementos do circuito são representados por uma codificação de cores, correspondendo cada cor a uma camada específica utilizada na tecnologia de fabricação. As subcélulas são representadas por seus contornos, de modo a não sobrecarregar a imagem manipulada.

As funções que o EDCI é capaz de realizar podem ser divididas em duas categorias: funções que atuam sobre a hierarquia da montagem de células e funções que atuam sobre os elementos básicos do circuito. No primeiro caso, é possível verificar os conteúdos das células, alterar suas dimensões e posicionamento, removê-las, entre outras; a segunda categoria permite, entre outras, criar, remover e ajustar as posições dos componentes.

O EDCI permite que o projetista se posicione em um trecho qualquer do lay-out através de um cursor retangular representado por um contorno branco cujos movimentos básicos (deslocamento e ampliação) são comandados pelo operador através do teclado. A movimentação pode ser feita em modo lento ou rápido, possibilitando o percurso de todo um lay-out em tempo reduzido (Borges-83).

Para permitir o funcionamento do editor em máquinas de pequeno porte, tivemos que elaborar uma base de dados em memória secundária que conseguisse manter atualizadas todas as modificações incluídas durante o processo de edição, conservando na memória principal apenas as informações relativas a porção do lay-out sendo editada (usualmente, uma célula). Calcado nesta forma intermediária de armazenamento e sobre um mecanismo de transformações matriciais, o EDCI é muito eficiente e versátil no tratamento das informações gráficas.

2. DESCRIÇÃO DO BANCO DE DADOS

2.1. CIF e o formato DB/SYM

A linguagem CIF (Caltech Intermediate Format) é uma das linguagens mais utilizadas para descrição da geometria de Circuitos Integrados. Ela permite especificar para cada célula definida a combinação e posicionamento das diversas máscaras empregadas no processo de fabricação; a esses elementos chamamos boxes. Também é possível especificar subcélulas contidas em uma determinada célula, o que chamamos call. Ainda pa

ra efeito de extração e simulação de circuito, é interessante que se possa atribuir um nome a uma determinada posição dentro de uma célula, como descrito em (Telles & Souza-85); a estes pontos chamamos nós.

Para se editar um lay-out com o EDCI a partir de sua descrição em CIF, foi necessária a construção de um programa pré-processador que convertesse esta descrição para um formato fixo mais facilmente manuseável para o editor, já que o texto livre não apresenta essas características.

Basicamente este conversor identifica os símbolos (células) definidos no texto fonte e gera dois arquivos que conterão as informações geométricas necessárias à edição do lay-out. Estes arquivos foram denominados DB e SYM.

Nosso pré-processador é capaz de identificar estes elementos e compactá-los de forma inteligível ao EDCI numa estrutura de dados que examinaremos a seguir.

2.2. Organização em disco

Para cada símbolo definido no texto fonte em CIF é gerado um registro correspondente no arquivo DB e um conjunto de registros no arquivo SYM.

No arquivo DB estão presentes as seguintes informações:

- Número e nome do símbolo;
- Dimensões do símbolo (bounding-box):
  - coordenadas do canto inferior esquerdo;
  - comprimento e altura do símbolo;
- Apontadores para a descrição do símbolo no arquivo SYM.

O registro inicial do arquivo DB é um cabeçalho contendo basicamente o número de símbolos guardados naquele banco de dados, a tecnologia de fabricação e o número de setores ocupados do arquivo SYM.

Neste último arquivo, distinguem-se quatro tipos de registros, cada um deles descrevendo um tipo de componente da célula, a saber:

- boxes:
  - camada (layer) do box;
  - coordenadas do centro;
  - comprimento e altura;
- calls:
  - número do símbolo chamado;
  - matriz de transformação da chamada;
- nós:
  - coordenadas dos nós;
  - tipo e camada do nó, conforme (Telles & Souza-85);

- nome do nó;

- texto de comentário.

O acesso à descrição do símbolo no arquivo SYM é feito através dos ponteiros existentes no arquivo DB. Eventualmente, alguns registros podem não existir, por exemplo, se o símbolo não contiver subsímbolos. Neste caso uma informação inválida é colocada no ponteiro correspondente.

2.3. Organização em memória

A estrutura de dados em memória principal compõe-se de cinco tabelas:

- tabela de símbolos:
  Esta tabela é preenchida no início da edição com o conteúdo do arquivo DB;

- tabela de boxes:
  É preenchida com a descrição dos boxes da célula sendo correntemente editada;

- tabela de calls:
  Contém as informações sobre as chamadas a símbolos contidas na célula-corrente;

- tabela de nós:
  É preenchida com a descrição dos nós da célula-corrente;

- vetor de texto:
  Contém o texto de comentário do símbolo corrente.

A tabela de símbolos permanece em memória durante toda a edição, só sendo regravada em disco no final do processo. Passo a passo, conforme os símbolos vão sendo editados, as informações das quatro tabelas restantes vão sendo trazidas e regravadas em disco, como apêndices do arquivo SYM. Para que fosse atenuado ao máximo o tempo dispendido em entrada e saída para preenchimento das tabelas, só regravamos um símbolo em disco se houver sido produzida alguma alteração em sua estrutura durante a edição.

3. INTERFACE ENTRE IMAGEM E BANCO DE DADOS

3.1. Representação de um símbolo na tela

Conforme já foi mencionado, as informações contidas na descrição de um símbolo que possuem correspondente gráfico são boxes e calls. Aqueles são representados por áreas retangulares coloridas e estes por seus contornos (bounding-boxes) desenhados em azul-claro.

A versão atual do EDCI está dedicada à tecnologia nMOS, existindo portanto uma cor associada a cada uma das cinco camadas existentes, conforme (Borges-83). Estas cores podem ser combinadas dando a impressão de superposição de boxes na tela gráfica.

3.2. Associação entre o Banco de Dados e a Tela

Ao símbolo correntemente sendo editado associamos uma transformação matricial que permite fazer correspondência entre o sistema de coordenadas do símbolo e o sistema de coordenadas de tela gráfica. À medida que se caminha em profundidade na hierarquia dos símbolos, as transformações vão sendo compostas de modo a refletirem a correspondência atual entre desenho/tela.

Assim como determinada transformação permite fazer correspondência entre as coordenadas do símbolo e as coordenadas da tela, o contrário é conseguido através da aplicação da inversa da transformação, que mapeia pontos da tela em pontos do símbolo.

3.3. Matrizes de Transformação

O formalismo descrito a seguir encontra-se em (Newman & Sproull-79).

As transformações associadas a símbolos são representadas por matrizes 3 x 3, sendo possível através delas: transladar um símbolo, espelhá-lo em relação ao eixo X ou Y e girá-lo de 90, 180 ou 270 graus.

Matematicamente, seja P = (x,y) um ponto do plano. Vamos representá-lo através da matriz:

$$[\, x \;\; y \;\; 1 \,]$$

O ponto resultante da aplicação de uma transformação representada pela matriz T ao ponto P é dado por P x T.

Assim, se quisermos transladar a origem do sistema para o ponto (u, v), a matriz de transformação correspondente será:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ u & v & 1 \end{vmatrix}$$

e o produto P x T valerá

$$[x+u \;\; y+v \;\; 1]$$

que representa o ponto $(x+u, y+v)$.

Para espelhamento em relação a Y, X e rotação de t graus (t = 90, 180 ou 270 para geometria Manhattan), as matrizes são as seguintes:

$$\begin{vmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} \cos(t) & -\text{sen}(t) & 0 \\ \text{sen}(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Composições de transformações são obtidas multiplicando-se as matrizes que as representam. Desta forma, se a um símbolo está associada a transformação T e este símbolo chama outro com transformação U, a transformação associada ao segundo quando

ele for editado será U x T.

Na implementação, consideramos o fato de que a terceira coluna das matrizes valeria sempre 0 0 1, o que nos permitiu representá-las por um vetor de 6 elementos, reduzindo consideravelmente o tempo de acesso a cada um deles.

3.4. Stack de chamadas

Via de regra, a edição no EDCI começa sempre do símbolo fundamental, identificado no banco de dados como sendo o de número 0. A este símbolo associa-se uma transformação tal que as coordenadas de seu canto mínimo venham a coincidir exatamente com as coordenadas (0,0) da tela. Se (xmin, ymin) são as coordenadas do canto mínimo, não é difícil ver que a transformação será:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xmin & -ymin & 1 \end{vmatrix}$$

Uma das funções do EDCI (chamada Entra em Símbolo) permite, estando em um determinado nível, passar a editar uma subcélula chamada pelo símbolo corrente. Terminada a edição da subcélula, o projetista pode querer retornar ao símbolo de nível imediatamente anterior. Estes fatos sugerem que as informações sejam empilhadas durante o transcurso da hierarquia. Em geral, empilhamos o seguinte:

- número do símbolo corrente;
- transformação atual;
- limites de movimentação do cursor;
- escala atual de edição;
- posição atual do cursor.

Estas informações são suficientes para retornar um nível na hierarquia quando assim for desejado.

Outras funções que manipulam a hierarquia estão descritas em (Borges-85).

4. ALGORITMOS MAIS IMPORTANTES

O processo de edição torna-se suficientemente complexo na medida em que as imagens gráficas devem acompanhar as alterações realizadas sobre a estrutura de dados que representa a célula; isto implica em redesenhos constantes da tela gráfica, para que o desenho reflita a realidade das estruturas de dados.

Das diversas funções implementadas no EDCI, selecionamos algumas cujos algoritmos apresentam aspectos interessantes tanto do ponto de vista gráfico quanto algoritmico. Eles foram calcados sobre uma base de primitivas cuidadosamente elaboradas para permitir eficiência nas operações solicitadas. Dentre estas primitivas, podemos citar:

- preencher as tabelas de boxes, calls, nós e texto, o que corresponde a trazer as informações do disco;
- regravar estas informações em disco quando necessário;
- empilhar e desempilhar informações sobre o contexto atual de edição;
- aplicar a transformação atual a um ponto;
- aplicar a transformação atual a um box;
- multiplicar e inverter matrizes;
- desenhar um box na tela;
- desenhar um contorno na tela;
- apagar uma região da tela.

4.1. Desenho de um símbolo na tela

Esta função é chamada no EDCI toda vez que um símbolo deva ser desenhado na tela submetido e uma determinada transformação.

```
Procedimento MostraSímbolo (S, T)
S - Número do Símbolo
T - Matriz de Transformação
Início
    Traz o símbolo S do disco
    Para cada box do símbolo
        Aplica T ao box, obtendo (bx, by, bdx, bdy)
        Desenha (bx, by, bdx, bdy) na tela

    Para cada call do símbolo
        Seja M a matriz associada à chamada
        Seja N o símbolo chamado
        U = M x T
        Aplica U ao bounding-box de N, obtendo (bx, by, bdx, bdy)
        Desenha o contorno (bx, by, bdx, bdy)
Fim.
```

4.2. Entrada em Símbolo

Esta operação consiste em fazer com que o processo de edição passe a se realizar sobre uma subcélula do símbolo atual, o que corresponde a descer um nível na hierarquia.

```
Procedimento EntraSímbolo
Início
    Seja T a transformação atual
    Seja S o símbolo que está sob o cursor
    Seja M a matriz de chamada do símbolo S
    Se o símbolo atual foi alterado
```

```
        Guarda-o em disco
    Empilha o contexto atual
    U = M x T
    Aplica U ao bounding-box de S, obtendo (bx, by, bdx, bdy)
    Apaga a região (bx, by, bdx, bdy)
    Recalcula limites de movimentação do cursor
    MostraSímbolo (S, U)
    T = U
    Posiciona o cursor no canto inferior esquerdo do símbolo S
Fim.
```

4.3. Saída de Símbolo

É o processo inverso ao de entrada em símbolo. Acrescentamos a facilidade de o conteúdo do símbolo abandonado continuar ou não a ser visível.

```
Procedimento SaiDeSímbolo
Início
    Pergunta ao Usuário: "Quer recobrir o símbolo ?".
    Guarda a resposta em resp.
    Se o símbolo atual foi alterado
        Guarda-o em disco.
    Desempilha o contexto de edição.
    Se resp foi afirmativa
        Apaga a região que estava sendo editada.
    Redesenha (S, T, região anterior).
Fim.
```

4.4. Redesenho de uma região

O processo de atualização da tela gráfica muitas vezes se reflete em apagar um trecho que tenha sido alterado e redesenhar somente aquele trecho do símbolo. Este procedimento difere de MostraSímbolo apenas no fato de que somente a porção indicada é redesenhada.

```
Procedimento Redesenha (S, T, R)
S - Número do símbolo.
T - Matriz de Transformação.
R - Retângulo (x, y, dx, dy) delimitando o trecho a redesenhar.
Início
    Traz o símbolo S do disco.
    Para cada box do símbolo
        Aplica T ao box, obtendo (bx, by, bdx, bdy).
        Se (x, y, dx, dy) intercepta (bx, by, bdx, bdy)
```

```
            Desenha o box (bx, by, bdx, bdy).
    Para cada call do símbolo
        Seja M a matriz da chamada.
        Seja N o símbolo chamado.
        U = M x T.
        Aplica U ao bounding-box de N, obtendo (bx, by, bdx, bdy).
        Se (x, y, dx, dy) intercepta (bx, by, bdx, bdy)
            Desenha o contorno (bx, by, bdx, bdy).
Fim.
```

4.5. Expansão de Símbolo

A operação de expansão no EDCI consiste em visualizar um símbolo e seus subsímbolos recursivamente.

A expansão é utilizada principalmente quando se deseja visualizar todo o layout de um chip sendo editado, por exemplo.

```
Procedimento Expande (S, T)
S - Número do símbolo a expandir.
T - Matriz de Transformação.
Início
    Traz o símbolo S do disco.
    Para cada box do símbolo
        Aplica T ao box, obtendo (bx, by, bdx, bdy).
        Desenha o box (bx, by, bdx, bdy).
    Para cada call do símbolo
        Seja N o símbolo chamado.
        Seja M a matriz da chamada.
        U = M x T.
        Expande (N, U).
Fim.
```

4.6. Apagamento de boxes

A função de apagamento de boxes permite remover todos os boxes que estejam situados sob a região delimitada pelo cursor. A complexidade deste algoritmo reside no fato de que, ao se apagar uma porção de um box, podem surgir de zero a quatro novos boxes distintos.

```
Procedimento ApagaBox (C)
C - Conjunto de cores a apagar.
Início
    Seja T a transformação atual e I a sua inversa.
    Seja S o símbolo atual.
```

```
    Seja R = (xcur, ycur, dxcur, dycur) as coordenadas do cursor.
    Aplica I ao cursor obtendo (x, y, dx, dy).
    Para cada box do símbolo atual
        Seja L o layer do box.
        Se Cor (L) pertence a C
            Sejam (xb, yb, dxb, dyb) as coordenadas do box.
            Se (x, y, dx, dy) intercepta (xb, yb, dxb, dyb)
                Remove o box da tabela.
                Acrescenta os boxes complementares.
    Apaga a região do cursor R.
    Redesenha (S, T, R).
Fim.
```

4.7. Seleção de Janela

É possível durante o processo de edição que se queira visualizar regiões do lay-out em escala ampliada ou mesmo visualizar o lay-out de todo um chip. Tendo em vista as limitações das dimensões da tela, torna-se necessária uma função que possibilite selecionar o trecho do lay-out a ser mostrado. Isto se reflete numa alteração da matriz de transformação corrente, sendo conseguido pelo algoritmo abaixo:

```
Procedimento SelecionaJanela (X, Y)
(X, Y) - Ponto a partir do qual o lay-out será mostrado.
Início
    Seja T a transformação atual.
    Seja U a transformação que translada a origem para (X, Y)
```

$$U = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X & -Y & 1 \end{vmatrix}$$

```
    T = T x U.
Fim.
```

5. CONCLUSÃO

A versão atual do EDCI, já em funcionamento no NCE desde o início de 1985, foi programada em linguagem C para executar em um mini-computador PDP 11/70 com sistema operacional Unix.

Quanto a tamanho e desempenho, o EDCI é um programa de aproximadamente 5.000 linhas que necessita de 128 Kbytes de memória para execução. Seu rápido tempo de resposta a comandos, mesmo com alto índice de utilização da máquina, deve-se principalmente ao fato de as rotinas básicas largamente solicitadas pelo editor haverem sido elaboradas de maneira a funcionarem o mais eficientemente possível, evitando-se pontos de estrangulamento no fluxo normal de execução. Além disso, foram utilizados re-

cursos de bufferização promovidos pelo Unix para entrada e saída de dados que permitiram tornar quase instantâneo o processo de atualização de imagens.

Deve ser enfatizada a questão da portabilidade: por haver sido programado em C de forma paramétrica e modular, o EDCI é um programa que pode ser facilmente adaptado para outros equipamentos, eventualmente micro-computadores, com um mínimo de alterações.

6. BIBLIOGRAFIA

Borges, J. A. S. - Ferramentas Gráficas para o Projeto de Circuitos Integrados - Anais do Congresso Nacional de Informática, 1983.

Borges, J. A. S. - Editores Gráficos para Projetos de Circuitos Integrados - Submetido à Comissão de Seleção do II Simpósio Brasileiro de Concepção de Circuitos integrados.

Teles, A. A. S. et Souza, L. F. P. - EXTRAI - Extrator de Circuitos Integrados NMOS - Submetido à comissão de seleção do II Simpósio Brasileiro da Concepção de Circuitos Integrados.

Newman, W. et Sproull, R. - Principles of Interactive Computer Graphics - 2nd. edition - Mac Graw Hill - 1979