Síntese de layout

Particionamento

DI-UFPE/GRECO

Manoel Eusebio de Lima

Síntese de layout - Particionamento

- Esta fase no projeto físico do circuito integrado (chip) representa a tarefa de decompor uma dado projeto em partes visando a redução de uma certa função objetivo, a qual pode ser bastante complexa dependendo da aplicação.
- As unidades funcionais do chip devem ser alocadas em partições nas quais as unidades guardam entre si uma certa relação, em função de uma função custo.
- Em um nível mais baixo, o *"particionamento"* é também usado para reduzir a complexidade do posicionamento, principalmente quando um grande número de unidades lógicas estão para ser distribuídas numa certa área de silício.

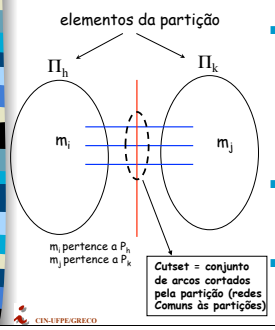
Síntese de layout - Particionamento

- Função custo
  - Considere M um conjunto de módulos (células) M={m_1, m_2, m_3,..... m_k}
  - Considere também N um conjunto de redes que conectam este módulos, N={n_1, n_2, n_3,..... n_k}
  - Cada rede n de N ($n_j \in N$) pode ser representada por um certo conjunto de módulos os quais possuem pelo menos um pino de n_j = {m_{1j}, m_{2j}, m_{3j},..... m_{kj}}
  - Cada rede por sua vez também possui um certo número de pinos P_j= {p_{1j}, p_{2j}, p_{3j},.....p_{kj}}, ou seja, conjunto de pinos de uma determinada rede n_j.
  - Cada módulo possui um certo número de pinos de diferentes redes.
  - Considere p(i) o conjunto de pinos de um módulo m_i.

Síntese de layout - particionamento

- O problema do particionamento é encontrar uma partição de M, Π={π_1,..... π_t}, $\pi_i \subset M$, $\cup \pi_i$=M, $\pi_i \cap \pi_j = 0$, i ≠ j, sujeito a restrições, tais como:
- Modelos de restrições:
  - O número de redes externas:
    - Minimizar a função $\mathbf{C(\Pi) = \Sigma\, c_{ij}}$, onde c_{ij} é o número de redes que conectam m_i a m_j, com $m_i \in \pi_h$ e $m_j \in \pi_k$, com k≠h, seja minimizado.
  - Limitar o tamanho dos módulos:
    - Em cada elemento da partição em função da área máxima permitida e não pelo número de elementos.
  - Estabelecer pesos às redes:
    - A função custo é função dos pesos das redes e não do número de redes.

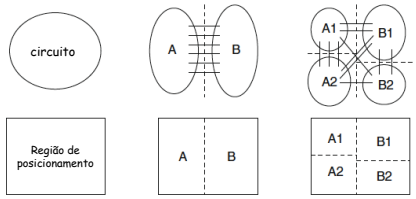
Síntese de layout - particionamento



- Em geral cada módulo m_x tem um tamanho tal que as restrições de capacidade são dadas em termos da soma dos tamanhos destes módulos (área) em cada elemento da partição, em vez do número de módulos.
- As redes podem também ter pesos distintos, em função do número de pinos, e seu tamanho.
- É desejável que haja um menor número possível de redes entre as partições.

Síntese de layout - particionamento

- Ideia:



Síntese de layout - Particionamento

- O problema de particionamento é um problema NP-completo. Algoritmos heurísticos têm sido propostos ao longo dos anos para se obter melhores resultados baseados em funções custo.
- Em geral estes algoritmos são divididos em dois grandes grupos:
  - Método construtivo
    - Não precisam em geral de uma solução inicial. O ponto de início do particionamento é simplesmente o conjunto de elementos sem particionamento. Tais como:
      - Bi-partitioning (Min-cut)
      - Force-Directed algorithm
  - Método iterativo com refinamento
    - Em geral estes algoritmos partem de uma partição inicial e a refinam a partição mediante processos iterativos baseadas em funções custo. Tais como:
      - Randon Interchanges
      - The kernighan-Lin Algorithm
      - Simulated Annealing

Síntese de layout - Particionamento

Algoritmo de particionamento

Min-cut

- Este algoritmo assume a disponibilidade de uma seqüência de r linhas de corte no circuito, as quais dividem os módulos em "*slots*".
- Características:
  - A idéia deste algoritmo é minimizar o número de linhas entre partições através de uma linha de corte, e uma vez especificado o corte e escolhidas as células dos dois lados da linha de corte, estas não poderão mais serem deslocadas para outros "*slots*".
  - A medida que as linhas de corte vão sendo colocadas as células vão sendo ajustadas aos "*slots*" em função de uma função custo (área, ...).
  - Por causa na natureza do algoritmo (*greedy-guloso*) a solução obtida não é garantida ser globalmente ótima.

Síntese de layout - particionamento

- **Algoritmo Min-cut(N,n,C);**

(N é a área reservada para o layout.
n é o número de células a serem colocadas.
n_0 é o número de células em um determinado slot.
C é a matrix de conectividade).

Begin
IF(n menor ou igual a n_0) **Then**
place-células(N,n,C)
Else Begin
(N_1, N_2) <- linha de corte(N);
(n_1, c_1), (n_2, c_2) <- partição(n,C);
Call Min-cut (N_1, n_1, c_1);
Call Min-cut (N_2, n_2, c_2);
EndIf;
End.

A idéia no algoritmo Min-cut é achar cortes no circuito que minimizem o número de redes através destes cortes e o número de células (ou área) em cada partição.
Este algoritmo de particionamento também é usado para problema de posicionamento.

Algoritmo - Kernighan-Lin Fiduccia-Mattheyses

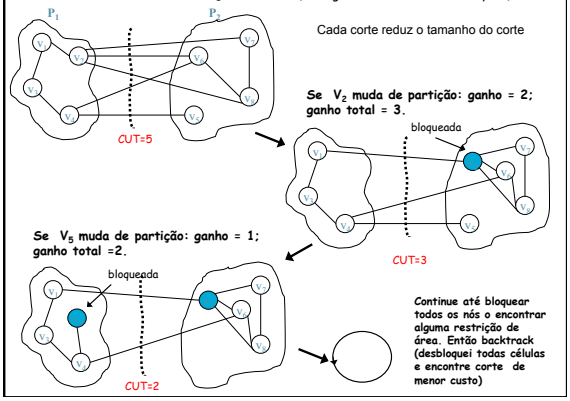
- KLFM é um algoritmo melhoramento interativo (*iterative improvement*) que não dá nenhuma orientação sobre como construir o particionamento inicial.
- Como se poderia esperar, há muitas maneiras de construir um particionamento inicial, e o método escolhido tem um impacto sobre os resultados.
- O método mais simples para gerar uma partição inicial é o método randômico, ou seja, sugere-se um corte inicial, randomicamente, para se iniciar a pesquisa pelo melhor corte (*cutsize*) do particionamento.
- Em um sistema com N células a complexidade do algoritmo KLFM é $O(N^2)$.

Algoritmo - Kernighan-Lin Fiduccia-Mattheyses

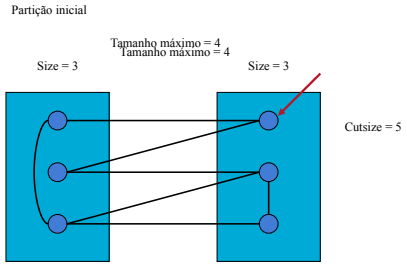
- Criar particionamento inicial;
  While *cutsize* é reduzido {
  While existirem movimentos válidos {
  Utilize estruturas de dados tipo "*bucket*" para encontrar nós desbloqueados, em cada partição, os quais na maioria das vezes melhora o *cutsize* quando muda-se para outra partição;
  Mova qualquer um dos dois nós que mais melhora o *cutsize* enquanto não atingir o limite superior do tamanho de partição;
  Bloquear nó movido;
  Atualize redes conectadas a nós que se moveram, e nós conectados a essas redes;
  } ***endwhile***;
  Backtrack para o ponto com *cutsize* mínimo encontrado e salve o resultado;
  Desbloquear todos os nós;
  } ***endwhile***;

www.ee.washington.edu/faculty/hauck/Thesis/Chapter10.ps

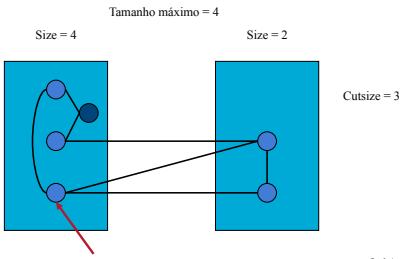
Como é definida a linha de corte: algoritmo KLFM (Kernighan-Lin Fiduccia-Matheyses) ?



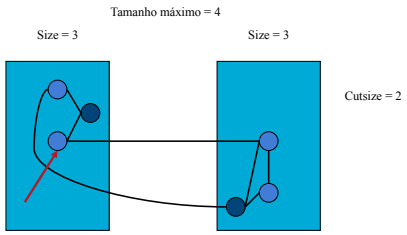
Algoritmo KLFM



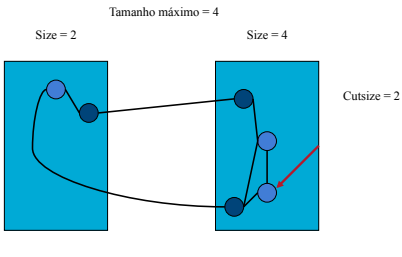
Algoritmo KLFM



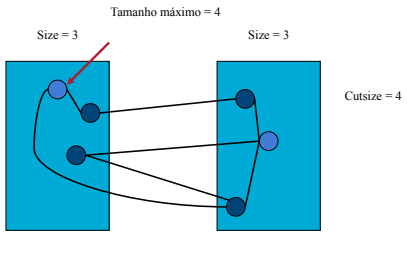
Algoritmo KLFM



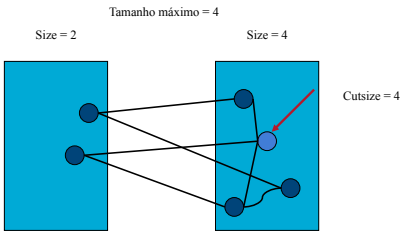
Algoritmo KLFM



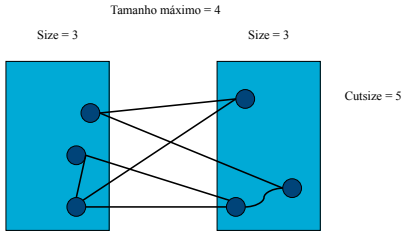
Algoritmo KLFM



Algoritmo KLFM

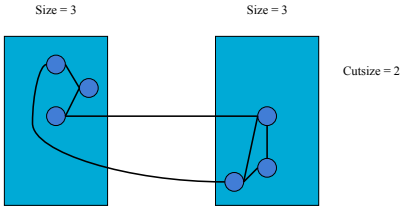


Algoritmo KLFM



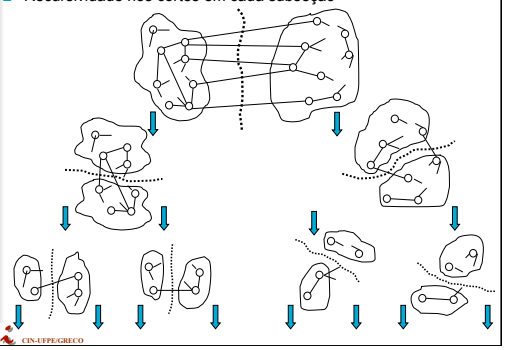
Algoritmo KLFM

Backtrack até encontrar o menor *cut size*. Então, desbloqueie os nós, e repita o processo a partir desta configuração até encontrar um *cut size* menor. Se não encontrar pare o processo e aceite a configuração encontrada



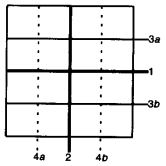
Algoritmo Min-cut

- Recursividade nos cortes em cada subseção



Algoritmo Min-cut

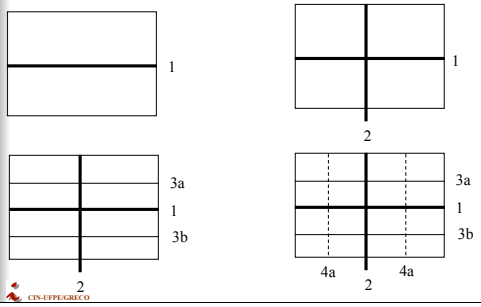
- Estilos de seqüência de linhas de corte (Breuer 1977)
  - Procedimento de particionamento por quadratura
    - O layout é dividido em quatro unidades com duas linhas, uma vertical e outra horizontal, ambas passando pelo centro.
    - A divisão acima continua aplicada a cada quadrante do layout até que o layout todo seja partido em *slots*.



- **Método interessante para circuitos com uma densidade de roteamento grande no centro e estilo gate array.**

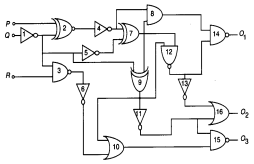
Síntese de layout - particionamento

- **Algoritmo Min-cut(N,n,C);** (particionamento por quadratura)



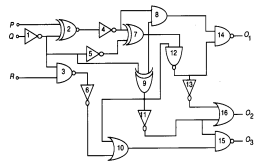
Algoritmo Min-cut

- Exemplo (particionamento por quadratura)



Inicialmente
Suponha área reservada para o layout = N
Número de células a serem alocadas (n) = 16

Particionamento



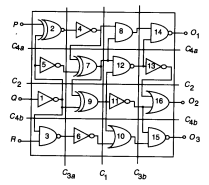
Matriz conectividade

Célula	1	2	3	4	.....	16	Σ
1	0	1	1	0	.....		3
2	1	0	0	1	.....		1
3	1	2	0	0	.....		
4	0	1	0	0	.....		
.....							
16	0	0	0	0	.....		3

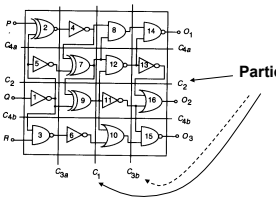
n = 16 (número de células a serem colocadas) ⇓ Particionamento por quadratura

2	4	8	14
5	7	12	13
1	9	11	16
3	6	10	15

c_{4a}, c_{4b}, c_2, c_{3a}, c_1, c_{3b}



Algoritmo Min-cut



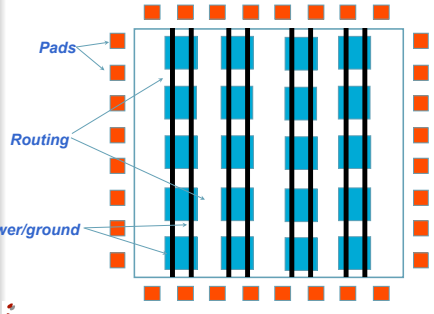
Particionamento por quadratura

$N_P = \{P, C_2\}$
$N_1 = \{C_1, C_2, C_3, C_5, C_9\}$
$N_4 = \{C_4, C_7, C_8\}$
$N_7 = \{C_7, C_8, C_9, C_{12}\}$
$N_{10} = \{C_{10}, C_{15}\}$
$N_{13} = \{C_{13}, C_{16}\}$
$N_{16} = \{C_{16}, O_3\}$

$N_Q = \{Q, C_1\}$
$N_2 = \{C_2, C_4\}$
$N_5 = \{C_5, C_7\}$
$N_8 = \{C_8, C_{14}\}$
$N_{11} = \{C_{11}, C_{15}, C_{16}\}$
$N_{14} = \{C_{14}, O_1\}$

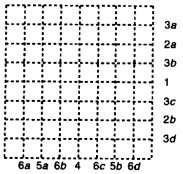
$N_R = \{R, C_3\}$
$N_3 = \{C_3, C_6\}$
$N_6 = \{C_6, C_{10}\}$
$N_9 = \{C_9, C_{10}, C_{11}, C_{12}\}$
$N_{12} = \{C_{12}, C_{13}, C_{14}\}$
$N_{15} = \{C_{15}, O_3\}$

Possível estilo de projeto
Gate Array - Block-cell (MPGA/FPGA)



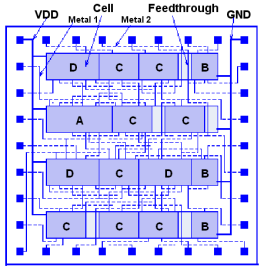
Algoritmo Min-cut

- Bi-seção (Bisection)
  - O layout neste modelo é repetidamente dividido em metades iguais por linhas de corte horizontal produzindo segmentos horizontais. Esta divisão continua até que cada uma segmento horizontal é uma linha. As células são colocadas em linhas.
  - Depois, cada linha é bi-seccionada verticalmente até que sub-regiões contenham um *slot*.



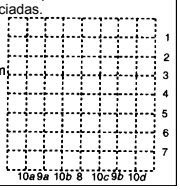
- **Este método é bom para implementação de circuitos que usam o estilo de projeto *standard cells***

Projeto *standard cells*



Algoritmo Min-cut

– Slice/Bisection
  - Neste método as *n* células do circuito são divididas usando-se uma linha de corte c1 entre dois conjuntos *k* e (*n-k*) de células, tal que o corte é minimizado.
  - As primeiras células *k* obtidas são colocadas na fila, no topo do circuito, ou na fila na base do circuito.
  - O procedimento é então aplicado as restantes (*n-k*) células, dividindo-as em (*n-2k*) e assim por diante, sempre colocas em linhas até todas as células serem referenciadas.
  - As células são então atribuídas a colunas usando bi-seção vertical.
  - **Método recomendado para células com alta interconexão com periféricos.**



Algoritmo Min-cut

- Referências
  – Kernighan-Lin Mincut Heuristic (1970)
    - Complexidade logarítmica: $O(N^2 \log N)$
  – Extension by Fiduccia and Mattheyses (1982)
    - Complexidade Linear: O(e)