Síntese de layout

Floorplanning

DI-UFPE/GRECO

Manoel Eusebio de Lima

Síntese de layout - floorplanning

- O principal objetivo do floorplanning é a investigação relacionada a determinação relativa das posições dos módulos (unidades funcionais) de um circuito no sílicio.
- Esta fase do layout está diretamente relacionado ao problema de posicionamento e muitas das mesmas técnicas são usadas em ambos os problemas.
- Floorplanning pode ser interpretado como uma generalização do problema do posicionamento. No entanto, possui um pouco mais de flexibilidade, desde que módulos podem ter suas formas e direções alteradas, assim como, pinos podem ter suas posições modificadas visando a otimização do projeto.

Síntese de layout - floorplanning

- O problema do floorplanning pode ser formulado da seguinte forma, dadas restrições, tais como:
  - Um conjunto de módulos com geometria variável;
  - pinos com posições variáveis;
  - estimativa de atrasos e consumo de potência ;
  - restrições sobre a posição de alguns dos módulos e pinos;
  - restrições quanto a área total estimada e relação alturaxlargura;
  - uma netlist especificando quais pinos têm que ser interconectados;
  - estimativa de atraso por unidade de tamanho de interconexões;
  - restrições de atrasos estimado a nível de chip;
  - .....
- **Determine a geometria dos componentes, posições, relação alturaxlargura dos módulos e posição de pinos, tal que as restrições sejam satisfeitas e que a área total, tamanho de redes, atraso e potência dissipadas sejam minimizados.**

Síntese de layout - floorplanning

- A mais importante exceção com relação ao posicionamento dos módulos está no modelamento destes módulos. Algoritmos para floorplanning devem modelar as flexibilidades das células e qualquer condicionamento sobre tal flexibilidade.
- Em geral várias classes de células são usadas no flooplanning:
  - células rígidas - possuem seus layouts prontos na biblioteca dos sistema, onde todas as características da interface são conhecidas e fixas;
  - células flexíveis - não possuem forma fixa ainda e podem ser influenciadas pelos resultados do floorplanning.

Síntese de layout - floorplanning

- O problema do floorplannig é um problema NP-hard, ou seja, pelo menos tão hard quanto qualquer NP-completo.
- O número de possíveis soluções é muito grande para um determinado problema. Daí a necessidade de heurísticas baseadas em funções de custo como ferramenta para estimar floorplanning.
- Uma Função custo além de reduzir o tempo de pesquisa para possíveis soluções e sua introdução permite-nos selecionar bons resultados.
- A qualidade do floorplanning de um determinado circuito pode ser medida levando-se em conta critérios, tais como;
  - minimização de área
  - minimização das redes
  - maximização da routabalidade
  - minimização do atraso (delay) ou
  - combinação de dois ou mais critérios acima

Síntese de layout - floorplanning

Função custo (exemplo)

- Função custo cujos critérios são:
  - Área
  - Tamanho das interconexões
- **Custo = α x A + β x L**
  - α e β são parâmetros usualmente especificados pelo cliente.

O objetivo é encontrar um possível floorplanning que minimze a função custo acima.

Síntese de layout - floorplanning

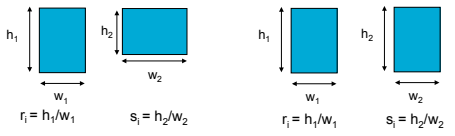
Problema:

- Dado um conjunto de módulos retangulares S = { 1,2,.., i, ., n };
- S_1 e S_2, uma partição de S, onde S_1 e S_2 são os conjuntos dos módulos com orientação fixa e flexível respectivamente;
- uma matriz conectividade $C_{nxn} = [c_{ij}]$, $1 \le i, j \le n$, onde c_{ij} indica a conectividade entre os módulos i e j;
- uma lista de n-triplas (A_1, r_1, s_1), ... (A_i, r_i, s_i), ... (A_n, r_n, s_n), onde;
  - A_i é a área do bloco i, ou seja, $A_i = w_i x h_i$ com w_i e h_i representando respectivamente a largura e altura do bloco i. r_i e s_i representam restrições (relação) quanto ao limite inferior e superior da **forma** do bloco i (*aspect ratio*)
  - $r_i \neq s_i$ se o bloco é flexível
  - $r_i = s_i$ se o bloco é rígido (forma não pode mudar)
- Dois números positivos p e q (p < q), os quais representam os limites de restrição superior e inferior da forma do retângulo que limita o chip (os blocos).

DI-UFPE/GRECO

Relação alturaxlargura

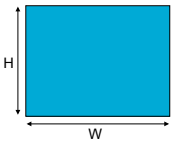
$r_i \neq s_i$ se o bloco é flexível $r_i = s_i$ se o bloco é rígido



$r_i = h_1/w_1$ $s_i = h_2/w_2$ $r_i = h_1/w_1$ $s_i = h_2/w_2$

Limites do chip

$p \le (H/W) \le q$



DI-UFPE/GRECO

Síntese de layout - floorplanning

Solução desejada

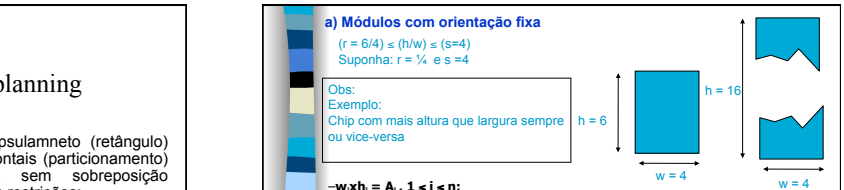
- Um floorplanning que permita o encapsulamneto (retângulo) sub-dividido por linhas verticais e horizontais (particionamento) que guarde todos os retângulos sem sobreposição (*overlapping*), considerando as seguintes restrições:
  - $w_i x h_i = A_i$, $1 \le i \le n$;
  - $r_i \le (h_i / w_i) \le s_i$ para todos os módulos i com orientação fixa (i é um elemento de S_1);
  - $r_i \le (h_i / w_i) \le s_i$ ou $1/s_i \le (h_i / w_i) \le 1/r_i$ para todos os módulos i com orientação livre (i é um elemento de S_2);
  - $x_i \ge w_i$ e $y_i \ge h_i$ $1 \le i \le j$, onde x_i e y_i são as dimensões do retângulo básico i, (todo retângulo é grande o bastante para acomodar o módulo i);
  - $p \le (H/W) \le q$, onde H e W são a altura e largura do retângulo que envolve os blocos.

DI-UFPE/GRECO

a) Módulos com orientação fixa

(r = 6/4) ≤ (h/w) ≤ (s=4)
Suponha: r = ¼ e s =4

Obs:
Exemplo:
Chip com mais altura que largura sempre ou vice-versa

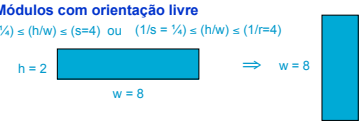


$w_i x h_i = A_i$, $1 \le i \le n$;

$r_i \le (h_i / w_i) \le s_i$ para todos os módulos i com orientação fixa (i é um elemento de S_1);

b) Módulos com orientação livre

(r = ¼) ≤ (h/w) ≤ (s=4) ou (1/s = ¼) ≤ (h/w) ≤ (1/r=4)



$w_i x h_i = A_i$, $1 \le i \le n$;

$r_i \le (h_i / w_i) \le s_i$ ou $1/s_i \le (h_i / w_i) \le 1/r_i$ para todos os módulos i com orientação livre (i é um elemento de S_2);

DI-UFPE/GRECO

Síntese de layout - floorplanning

- **Exemplo:**
  Suponha que desejemos posicionar 5 blocos rígidos com dimensões segundo a tabela abaixo em dada área:

Assumimos que todos os módulos têm orientações livres e que o chip possui a mesma área. Se a área é a função custo usada para medir a qualidade dos possíveis *floorplans*, então todos os 3 *floorplans* são igualmente bons.

Dimensões dos módulos		
Módulo	*largura*	*altura*
1	*1*	*1*
2	*1*	*1*
3	*2*	*1*
4	*1*	*2*
5	*1*	*3*

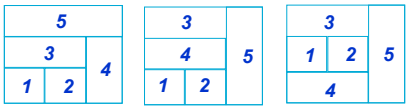
DI-UFPE/GRECO

Síntese de layout - floorplanning

Modelos de floorplans

Dimensões dos módulos		
Módulo	*largura*	*altura*
1	*1*	*1*
2	*1*	*1*
3	*2*	*1*
4	*1*	*2*
5	*1*	*3*

Observações:
- os 5 módulos são rígidos com orientação livre;
- Todos os três floorplans têm a mesma área



Observamos que todos os três floorplanning possuem a mesma área. Assim, se a função custo estivesse diretamente relacionada com a área resultante poderíamos dizer que os resultados acima são de mesma qualidade.

DI-UFPE/GRECO

Síntese de layout - floorplanning

Modelos de heurísticas para geração de *floorplans*

- **Construtivo**
  - Modelo que tenta construir uma solução a partir de uma semente (*seed*). Então outros módulos são selecionados (um por um) no tempo e adicionados ao resultado parcial. Este processo continua até que todos os módulos sejam selecionados.

  Exemplos:
  - *cluster growth*
  - particionamento e *slicing*
  - *connectivity clustering*
  - .....

Síntese de layout - floorplanning

- **Iterativo**
  - Neste modelo, o floorplanning começa de um resultado inicial e refina o *floorplanning*. Neste modelo, a solução inicial é perturbada, baseada em alguma heurística, até que uma solução adequada seja alcançada ou não haja mais melhoramento dos resultados obtidos.

  Exemplos:
  - *simulated annealing*
  - algoritmo genético
  - força direta com troca/relaxação
  - .....

Síntese de layout - floorplanning

- Exemplo: (algoritmo *Cluster Growth*)
  - **Neste estilo, o *flooplan* é construído no estilo "*greedy*", onde cada módulo individualmente é colocado em uma posição do *floorplan*.**
  - **Uma célula semente é escolhida entre os módulos e colocada em um dos cantos do floorplan (em geral canto esquerdo, embaixo).**
  - **Os demais módulos são então selecionados, um por vez, e adicionados ao *floorplan* inicial, enquanto o chip cresce diagonalmente, para a direita e para cima simultaneamente, preservando possíveis restrições impostas ao tamanho do chip.**
  - **Neste algoritmo utilizamos duas fases:**
    - **Determinamos a ordem na qual os módulos deveriam ser selecionados. Os módulos são organizados inicialmente na forma linear. "Aplicação do Ordenamento Linear"**
    - **Posteriormente aplicamos o algoritmo *Cluster Growth***

Síntese de layout - floorplanning

- Ordenamento Linear
  - Algoritmos para ordenamento linear de módulos em uma "*netlist*" visa ordenar estes módulos de forma a minimizar o número de redes que serão cortadas por qualquer linha vertical traçada entre quaisquer consecutivos par de módulos ordenados.
  - Esta técnica é bastante usada para a construção de um posicionamento inicial de módulos.

Síntese de layout - floorplanning

Exemplo de Algoritmo para geração de floorplans (Cluster growth)

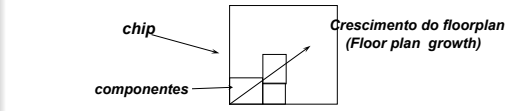
Algorithm Linear_Ordering (by Kang 1983)
S: Conjunto de módulos;
Order: Seqüência de módulos ordenados; (inicialmente vazio)
Begin
  Seed:= Selecione módulo Seed (primeiro módulo)
  Order:= [Seed]
  S:= S - [Seed]
  Repeat
    ForEach módulo m de S Do
      Compute o ganho para os módulos selecionados;
      $gain_m$:= no. de redes terminadas por m - no. de redes iniciadas por m
    End ForEach;
    Selecione o módulo de maior ganho;

continua

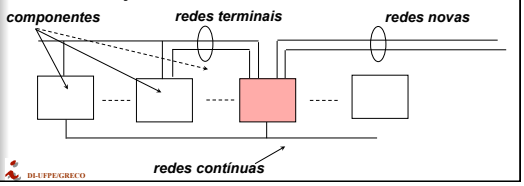
Síntese de layout - floorplanning

(Algoritmo - Linear_Ordering)
  If existe um empate no valor do ganho Then
    Selecione o módulo que termina um maior número de redes
  Elself existe um empate no valor do ganho Then
    Selecione o módulo que tem o maior número de redes contínuas
  Elseif existe um empate no valor do ganho Then
    Selecione o módulo com menos conexões
  Else Quebre o resto das redes como desejado;
  Endlf
  Order := [!Order, m*]; (junte m* a seqüência ordenada)
  S := S- [m*]
Until S = ϕ
End.

Síntese de layout - floorplanning



Classificação das redes durante o ordenamento linear



Algoritmo *Cluster_Growth*

- **Algoritmo *Cluster_Growth***

S: Conjunto de todos os módulos

Begin

Order:= Linear_Ordering (S)

Repeat

nextmodule := b onde Order = [b,!rest];

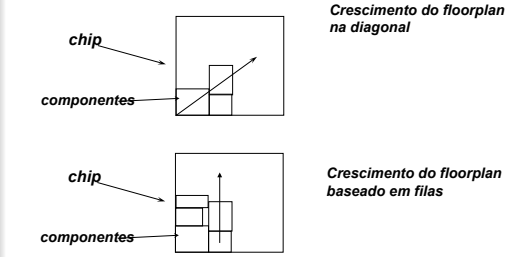
Order := rest;

Selecione a posição de b no chip que resultará no mínimo incremento da função custo; (O custo pode ser função do contorno do *floorplan* parcial, tamanho e forma de b, e tamanho de conexões)

Until Order = ϕ

End.

Algoritmo Cluster_Growth



Crescimento do floorplan na diagonal

Crescimento do floorplan baseado em filas

Exemplo:

- **Considere a netlist com 6 células [C_1, C_2, C_3, C_4, C_5, C_6] e 6 redes**
  - $N_1 = \{C_1, C_3, C_4, C_6\}$
  - $N_2 = \{C_1, C_3, C_5\}$
  - $N_3 = \{C_1, C_2, C_5\}$
  - $N_4 = \{C_1, C_2, C_4, C_5\}$
  - $N_5 = \{C_5, C_2, C_6\}$
  - $N_6 = \{C_3, C_6\}$
- Considere que todas as células são rígidas mas têm orientações livres.
- Procedimento:
  1. Escolha da célula semente

     A escolha da primeira célula pode ser randômica ou àquela que tem maior conexões de I/O e/ou conexões com as demais células do circuito. No exemplo, a célula semente (*seed*) será a célula C_1, célula com maior número de conexões no circuito.

Ordenamento Linear

Passo	Célula	Redes novas	Redes terminais	ganho	Redes contínuas
0	C_1^*	N_1, N_2, N_3, N_4	0	- 4	----------

* No passo 0 escolhemos C_1 para semente por um dos módulos com maior número de redes ligadas a ele (maior connectividade).

Passo

1 { Retiro C_1 }

$N_1 = \{\underline{C_1}, C_3, C_4, C_6\}$
$N_2 = \{\underline{C_1}, C_3, C_5\}$
$N_3 = \{\underline{C_1}, C_2, C_5\}$
$N_4 = \{\underline{C_1}, C_2, C_4, C_5\}$
$N_5 = \{C_5, C_2, C_6\}$
$N_6 = \{C_3, C_6\}$

Passo	Célula	Redes novas	Redes terminais	ganho	Redes contínuas
1	C_2	N_5	0	-1	N_3, N_4
	C_3	N_6	0	-1	N_1, N_2
	C_4^*	0	0	0	N_1, N_4
	C_5	N_5	0	-1	N_2, N_3, N_4
	C_6	N_5, N_6	0	-2	N_1

Maior ganho

Extraímos C_4 por esta possuir o maior ganho.

Ordenamento Linear

Passo

2 { Retiro C_1, C_4 }

$N_1 = \{\underline{C_1}, C_3, \underline{C_4}, C_6\}$
$N_2 = \{\underline{C_1}, C_3, C_5\}$
$N_3 = \{\underline{C_1}, C_2, C_5\}$
$N_4 = \{\underline{C_1}, C_2, \underline{C_4}, C_5\}$
$N_5 = \{C_5, C_2, C_6\}$
$N_6 = \{C_3, C_6\}$

Passo	Célula	Redes novas	Redes terminais	ganho	Redes contínuas
2	C_2	N_5	0	-1	N_3, N_4
	C_3	N_6	0	-1	N_1, N_2
	C_5^*	N_5	0	-1	N_2, N_3, N_4
	C_6	N_5, N_6	0	-2	N_1

Maior nº de redes contínuas

Temos três células com o mesmo ganho, C_2, C_3 e C_5. Extraímos C_5 por esta possuir o maior número de redes contínuas.

Ordenamento Linear

Passo

3 { Retiro C_1, C_4, C_5}

$N_1 = \{\underline{C_1}, C_3, \underline{C_4}, C_6\}$

$N_2 = \{\underline{C_1}, C_3, \underline{C_5}\}$

$N_3 = \{\underline{C_1}, C_2, \underline{C_5}\}$

$N_4 = \{\underline{C_1}, C_2, \underline{C_4}, \underline{C_5}\}$

$N_5 = \{\underline{C_5}, C_2, C_6\}$

$N_6 = \{C_3, C_6\}$

Passo	Célula	Redes novas	Redes terminais	ganho	Redes contínuas
3	C_2^*	0	N_3, N_4	+2	N_5
	C_3	N_6	N_2	0	N_1
	C_6	N_6	0	-1	N_1, N_5

Maior ganho

Extraímos C_2 por esta possuir o maior ganho.

DI-UFPE/GRECO

Ordenamento Linear

Passo

4 { Retiro C_1, C_4, C_5, C_2}

$N_1 = \{\underline{C_1}, C_3, \underline{C_4}, C_6\}$

$N_2 = \{\underline{C_1}, C_3, \underline{C_5}\}$

$N_3 = \{\underline{C_1}, \underline{C_2}, \underline{C_5}\}$

$N_4 = \{\underline{C_1}, \underline{C_2}, \underline{C_4}, \underline{C_5}\}$

$N_5 = \{\underline{C_5}, \underline{C_2}, C_6\}$

$N_6 = \{C_3, C_6\}$

Passo	Célula	Redes novas	Redes terminais	ganho	Redes contínuas
4	C_3^*	N_6	N_2	0	N_1
	C_6	N_6	N_5	0	N_1

Escolha arbitrária célula com menor índice

Temos duas células com o mesmo ganho, C_3 e C_6. Escolhemos C_3 por esta possuir o menor índice.

DI-UFPE/GRECO

Ordenamento Linear

2. Seguir o algoritmo passo-a-passo escolhendo adequadamente as células. Observe tabela abaixo:

Passo	Célula	Redes novas	Redes terminais	ganho	Redes contínuas
0	C_1^*	N_1, N_2, N_3, N_4	----------	- 4	----------
1	C_2	N_5	----------	-1	N_3, N_4
	C_3	N_6	----------	-1	N_1, N_2
	C_4^*	----------	----------	0	N_1, N_4
	C_5	N_5	----------	-1	N_2, N_3, N_4
	C_6	N_5, N_6	----------	-2	N_1
2	C_2	N_5	----------	-1	N_3, N_4
	C_3	N_6	----------	-1	N_1, N_2
	C_5^*	N_5	----------	-1	N_2, N_3, N_4
	C_6	N_5, N_6	----------	-2	N_1
3	C_2^*	----------	N_3, N_4	+2	N_5
	C_3	N_6	N_2	0	N_1
	C_6	N_6	----------	-1	N_1, N_5
4	C_3^*	N_6	N_2	0	N_1
	C_6	N_6	N_5	0	N_1

Maior ganho

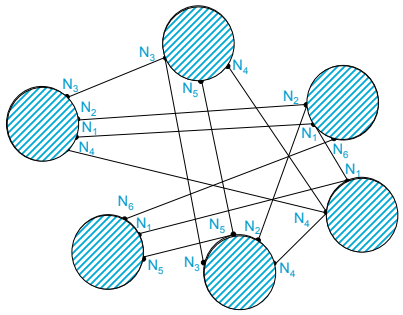
Maior no. redes contínuas

Maior ganho

Escolha arbitrária célula com menor índice

DI-UFPE/GRECO

Distribuição de redes entre circuitos



DI-UFPE/GRECO

Algoritmo Cluster_Growth

- **Algoritmo Cluster_Growth**

S: Conjunto de todos os módulos

Begin

Order:= Linear_Ordering (S)

Repeat

nextmodule := b onde Order = [b,!rest];

Order := rest;

Selecione a posição de b no chip que resultará no mínimo incremento da função custo; (O custo pode ser função do contorno do floorplan parcial, tamanho e forma de b, e tamanho de conexões)

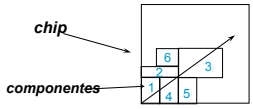
Until Order = ϕ

End.

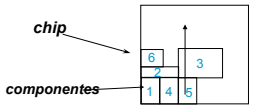
DI-UFPE/GRECO

Algoritmo Cluster_Growth

- Resultado do ordenamento linear [C_1, C_4, C_5, C_2, C_3, C_6]



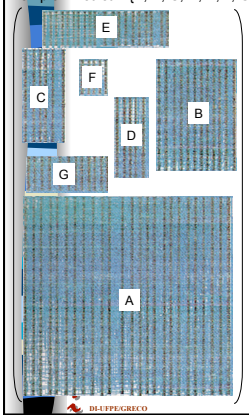
Crescimento do floorplan na diagonal



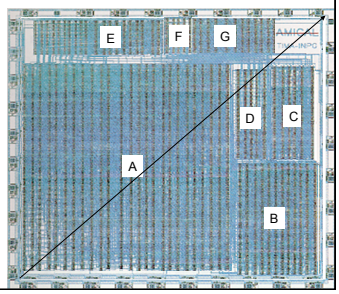
Crescimento do floorplan baseado em filas fila-por-fila

DI-UFPE/GRECO

Exemplo - Netlist = {A, B, C, D, E, F, G}



1. Particionamento
2. Flooplanning/posicionamento
3. Roteamento



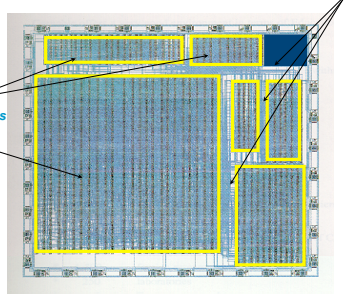
DI-UFPE/GRECO

Síntese de layout - floorplanning

Exemplo de floorplan

canais para roteamento

unidades funcionais



DI-UFPE/GRECO