

Pakning og Floorplaning

David Pisinger

Efterår 2004

VLSI layout

Logisk design → global placement → detaljeret placement → rutning → fotografiske masker

- Hierarkisk opdeling nødvendig ved store VLSI kredse
- Floorplaning er et "mellemstadie" i VLSI-design
- Relativ placering, areal af byggeblokke kendes
- Bredde og højde kan stadig ændres

Floorplaning kritisk for hierarkisk proces

1

Oversigt

- Introduktion til pakning
 - 2D rektangulær pakning (slicing)
 - 2D rektangulær pakning (no-slicing)
- Introduktion til floor planing
- Eksempler på abstrakte representationer
- (Anvendelse i metaheuristikker)
- (Sammenligning af løsningsrum)

Litteratur:

- [1] P.N.Guo, C.K.Cheng, T.Yoshimura (1999),
"An O-tree representation of non-slicing floorplans and its applications
DAC-99, 268--273.
- [2] "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair"
H. Murata, K. Fujiyoshi, S. Nakatate, Y. Kajitani.
IEEE Transactions on CAD, vol. 15, no. 12, 1518-1524, 1996

2

Placement problem

circuit $C = (\mathcal{A}, \mathcal{M}, \mathcal{P}, \mathcal{N})$

- \mathcal{A} er placement areal
- \mathcal{M} er moduler
- \mathcal{P} er pins
- \mathcal{N} er netliste der definerer pins som skal forbindes

Vi antager

- Modul $m \in \mathcal{M}$ har dimensioner (w_m, h_m) koordinater (x_m, y_m) .

Minimer

- netlength (fixed areal)
- area + k netlength

Floorplaning

- areal af hvert modul kendes
- bredde-højde kan varieres indenfor givne grænser

Terminologi

- Modul = rektangel
- Placement = pakning

3

Pakning

Genstande skal arrangeres indenfor begrænset domæne

- genstande: stænger/rektangler/kasser
- pakke/udskære
- rektangulære/generelle

Da NP-hårdt: ingen universel algoritme

- Knapsack pakning
- 2D rektangulær pakning
- Container loading
- Pallet loading
- Bin packing
- Cutting stock
- Pakning med irregulære mønstre

ofte overlappende grupper

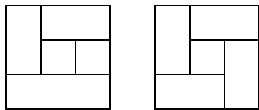
4

2D rektangulær pakning

- n rektangler w_j, h_j
- plade W, H
- anbring rektangler på pladen
- intet overlap
- ingen overskridelse af pladen

Versioner

- Slicing (guillotinesnit)/nonslicing (nonguillotine)
- Knapsack pakning/pak alle
- Minimere areal af plade
- Rektangler har variabel dimension (min/max aspect ratio)



5

2D rektangulær (slicing) pakning (Wong, Liu)

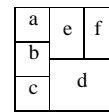
- repræsenteres ved "slicing tree"
 - xHy betyder x adskilles med horisontalt snit fra y
 - xVy betyder x adskilles med vertikalt snit fra y
- fra "slicing tree" til koordinater og dimensioner
- "slicing tree" som omvendt polsk notation

$$(((abH)cH)((eV)dH)V)$$

- Redundans i repræsentationen

$$(abH)cH \text{ or } a(bcH)H$$

- Søgerummets størrelse: $> n!2^n$



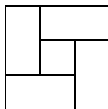
Fordele: lille løsningsrum, hurtig konvertering
Ulemper: fleste VLSI-design er non-slicing

6

2D rektangulær pakning (nonslicing)

Givet

- n rektangulære genstande w_j, h_j
- stort rektangel W, H
- knapsack fyldning: $p_j = w_j h_j = a_j$
- pak delmængde af rektangler så størst areal opnås
- ikke guillotine snit

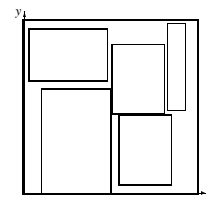


Kompleksitet vokser kraftigt: slicing \rightarrow nonslicing

7

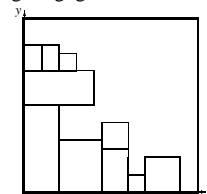
Idiotformel

Der findes altid en ækvivalent pakning hvor alle rektangler er flyttet længst muligt mod venstre og nedad.



Konsekvens: endeligt løsningsrum

Hvis rutning tages i betragtning, gælder idiotformel ikke



Pakninger med denne struktur kaldes *normaliserede* (compact)

8

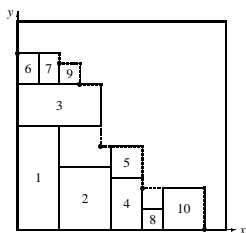
Ordning

For enhver pakning findes en ordning af rektanglerne, så pakningen altid sker ovenover eller til højre for allerede pakkede rektangler.

Lad x_j, y_j være start-kordinater for rektangel j .
Definer $i \preceq j$ hvis $x_i \leq x_j + w_j$ og $y_i \leq y_j + h_j$.

- refleksiv: $i \preceq i$
- transitiv: $i \preceq j$ og $j \preceq k$ medfører $i \preceq k$

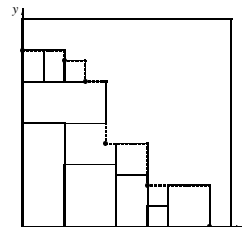
\preceq preordning



9

Branch-and-bound algoritme

Branch: kun punkter længst venstre/nede betragtes
Bound: ordning fastlægger en "konvolut" af brugt plads

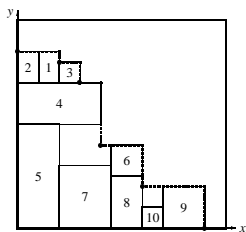


Rekursiv algoritme: allerede pakket I

- hvis bedre fyldning: gem løsning.
- find hjørnepunkter $C(I)$, areal $A(I)$
- backtrack hvis $C(I) = \emptyset$
- backtrack hvis konvolut har mistet for meget areal
- placer på skift hvert ledigt rektangel j til hvert hjørne $c \in C(I)$.
- kald rekursivt med $I \cup \{j\}$.

10

Bestemmelse af $C(I)$ og $A(I)$



- ordning: $\max y, x$, $O(n \log n)$
- find konvolut i $O(n)$ tid
- bestem hjørnepunkter
- slet hjørner, hvor ingen rektangler passer ind
- find omsluttet areal

11

Størrelse af løsningsrum

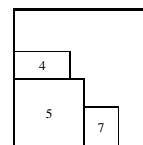
Ordning af rektangler: $n!$

- 1: anbring 1 af n rektangler på 1 position
- 2: anbring 1 af $n - 1$ rektangler på 1 af 2 positioner
- 3: anbring 1 af $n - 2$ rektangler på 1 af 3 positioner
- ...
- n : anbring 1 af 1 rektangler på 1 af n positioner

Størrelse: $O(n!n!)$

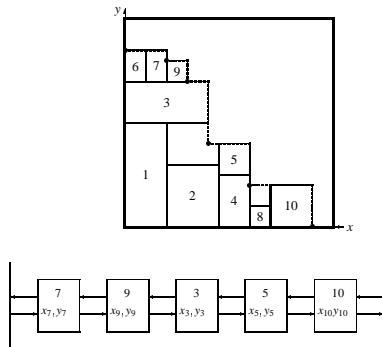
Redundans (ækvivalente pakninger)

- Symmetri (2 spejlinger, 180 grader rotation)
- Samme mønster kan fremkomme på flere måder



12

Hurtigere 2D pakning



- Recursion indsætter rektangler i samme orden som eksterne rektangler.
- For hvert placeret rektangel, find de rektangler som nu er usynlige.
- Fjern dominerede rektangler
- Anvend 2D algoritme på disse til at finde areal

Når anbragt n rektangler, kan højst $n - 1$ rektangler være blevet fjernet ved dominans.

Floorplanning

Definition

- En floorplan opdeler chip'en i rektangulære rum med horisontale eller vertikale segmenter
- Floorplanning er en "abstrakt repræsentation"
- En floorplan angiver topologien (relative position af rektangler)

Direkte repræsentation

- Svært at lave lokalsøgnings algoritmer
- Svært at eliminere symmetrier (stort søgerum)
- Nemt at beregne pris af rutning
- Ikke nødvendigt at beregne positioner eller areal

Floorplan

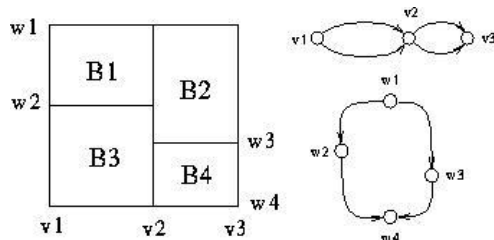
- Nemt at lave lokalsøgnings algoritmer
- Nemmere at eliminere symmetrier (mindre søgerum)
- Svært at beregne pris af rutning
- Floorplan skal konverteres til layout

Floorplan Optimization

- Mellem-tilstand i VLSI-design
- Relativ placering af moduler kendt
- Areal af moduler kendt
- Dimensioner af moduler variable
- (Forskellige implementationer af hvert modul)

Repræsentation

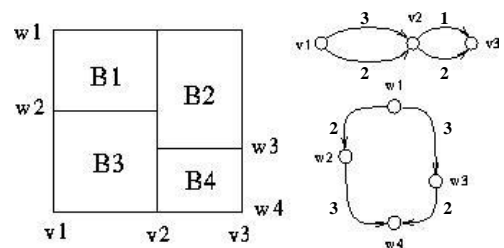
- To grafer $G = (V, E)$ og $H = (W, F)$
- (Vertical constraint graph) (Horizontal constraint graph)
- Knuder i V og W er "vægge"
- Kanter i E og F er moduler
- Kantvægte svarer til bredde af moduler
- Ekstreme knuder (North, South, East, West)



Floorplan og pakning er hinandens "duale".

Floorplanning med dimensioner

- Find længste vej $W \rightarrow E$
- Find længste vej $S \rightarrow N$



Abstrakte repræsentationer, floorplan

- Corner block list
- O -tree
- B^* -tree
- Transitive Closure Graph
- Sequence Pair

Focus på

- Repræsentation
- Hvor stort er løsningsrum
- Hvor hurtigt kan repræsentation konverteres til pakning

17

Corner Block List (Hong, Huang, Cia, Gu, Dong, Cheng, Gu)

- Mosaik floorplan har ingen "huller"
- Ækvivalens m.h.t. segment flytning
- To "T-er" må ikke mødes i samme punkt

En "corner block" (CB) er det rektangel der findes i øvre højre hjørne

$$\begin{array}{c} \text{SLICING} \\ \text{FLOORPLAN} \end{array} \subseteq \begin{array}{c} \text{MOSAIC} \\ \text{FLOORPLAN} \end{array} \subseteq \begin{array}{c} \text{NON-SLICING} \\ \text{FLOORPLAN} \end{array}$$

18

Corner block list: Sletnings algoritme

Sletnings-algoritme:

- Givet floorplan
- Så længe der er rektangler i floorplan
- Fjern CB
- Hvis CB's nedre venstre hjørne er " \perp ": flyt venstre segment mod højre ad chip'en og træk tilhørende T-kryds med segmentet
- Hvis CB's nedre venstre hjørne er " \lrcorner ": flyt nedre segment op ad chip'en og træk tilhørende T-kryds med segmentet

Lemma: Efter hver sletning af CB i en mosaik floorplan er tilbageværende graf stadigvæk en mosaik floorplan

19

Corner block list: Konstruktions algoritme

Foretag det modsatte af sletnings-algoritme.

- Givet rækkefølge af rektangler
- For hvert rektangel givet orientering (horisontal, vertikal)
- For hvert rektangel givet hvor mange rektangler der ligger t.h. eller f.o. (antal T-kryds)

Køretid $O(n)$.

Kodning (S, L, T)

- rækkefølge af rektangler $S = (f, c, e, g, b, a, d)$
- orientering ($0 = \lrcorner, 1 = \perp$) af rektangler $L = (0, 0, 1, 1, 0, 0)$
- antal T-kryds t.h. eller f.o. 0-termineret $T = (0, 0, 10, 10, 0, 10)$

Pladsforbrug:

S : n heltal hvert $\log n$ bit langt

L : $n - 1$ bits

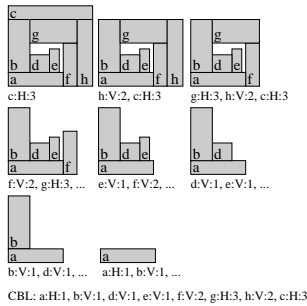
T : $2n - 3$ bits

Samlet: $n(3 + \log n)$ bit

Antal kombinationer $O(n!2^{3n-3}/n^{3/2})$

20

Corner block list: Eksempel



Corner block list: metaheuristik

Simuleret udglødning

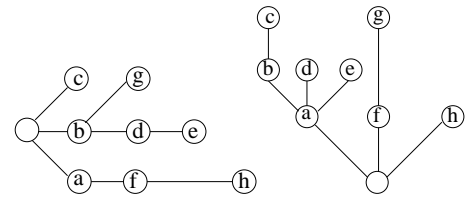
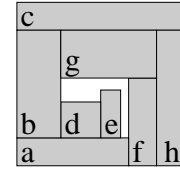
- ombyt rækkefølge af to rektangler i S
- ret en bit i L
- ret en bit i T

For hvert skridt: konstruer den tilhørende floorplan i $O(n)$ tid
evaluer objektfunktion

O-tree (Guo, Cheng, Yoshimura)

Repræsenteret ved to træer:

- Knuder: rektangler
- Kanter: "venstre" eller "under"



Konvertering:

- O-tree \rightarrow Pakning: søgning $W \rightarrow E, S \rightarrow N$ med contour (konvolut).
- Pakning \rightarrow O-tree: bredde-først søgning $W \rightarrow E, S \rightarrow N$

O-tree

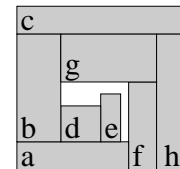
Fordele:

- normaliseret (kompakt)
- transformation i $O(n)$ tid
- repræsentation fylder $n(2 + \lceil \log n \rceil)$ bits
- søgerum $O(n!2^{2n-2}/n^{1.5})$

Lokalsøgning

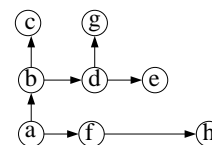
- Flyt knude i O-tree
- Konverter til pakning
- Udregn areal
- Evaluer i henhold til lokalsøgnings framework

B^* -tree (Chang², Wu²)



Horisontalt (x-kordinater) og vertikalt (y-kordinater) B^* -tree

en-til-en sammenhæng



Pakning til horisontalt B^* -tree

- rod svarer til rektangel i nedre venstre hjørne
- hver knude har to børn (nemt at implementere)
- hvis knude j er venstre barn af knude i så rektangel j til højre for rektangel i (og rører)
- hvis knude j er højre barn af knude i så rektangel j over rektangel i (og rører)
- dybde-først søgning: først venstre så højre deltræ

B*-tree

B* træ til pakning

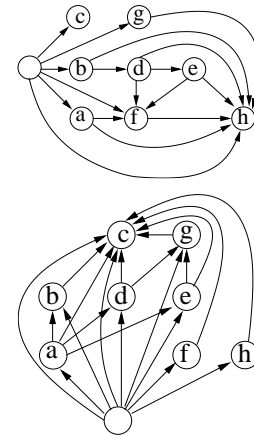
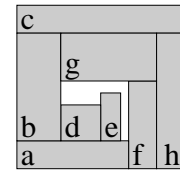
- rod har $x_r = 0$
- hvis knude j venstre barn af knude i så $x_j = x_i + w_i$.
- hvis knude j højre barn af knude i så $x_j = x_i$.

Fordele

- Normaliseret
- Med statisk datastruktur kan søge, indsætte i $O(1)$ tid
- Arealforbrug kan udregnes incremental
- Søgerums størrelse $O(n!2^{n-2}/n^{1.5})$

TCG (Lin, Chang)

Transitive Closure Graph



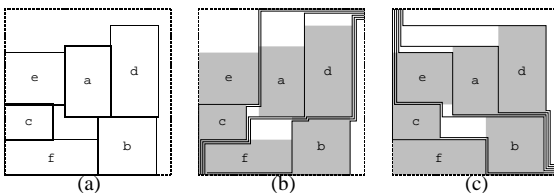
Konvertering: $O(n^2)$

Sequence pairs (Murata, Fujiyoshi, Nakatake, Kajitani)

- Enhver topologi af rektangler a, b, c, d, e, f kan repræsenteres ved to permutationer $\langle e, c, a, d, f, b \rangle, \langle f, c, b, e, a, d \rangle$

Størrelse af løsningsrum $O(n!n!)$

Enhver pakning kan repræsenteres som sequence pairs



(a) Placemet repræsenteret ved

positiv sekvens: $A = \langle e, c, a, d, f, b \rangle$

negativ sekvens: $B = \langle f, c, b, e, a, d \rangle$

(b) Positive step-linier

(c) Negative step-linier

Sequence pairs: Fra kodning til layout

Konstruer en constraint graf for floorplan

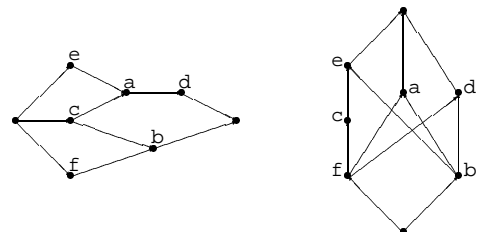
- Givet liste $(\Gamma_+, \Gamma_-) = (\langle d, c, a, f, b, e \rangle, \langle f, c, e, d, a, b \rangle)$
- Hvis $\langle \dots x_i \dots x_j \dots \rangle, \langle \dots x_i \dots x_j \dots \rangle$ så x_i til venstre x_j
- Hvis $\langle \dots x_i \dots x_j \dots \rangle, \langle \dots x_j \dots x_i \dots \rangle$ så x_i over x_j
- N, S, E, W forbindes til respektive rektangler
- F.eks. W forbindes til alle rektangler der ikke har et rektangel t.v.

Grafen vil have størrelse $O(n^2)$.

Constraint graf omdannes til faktisk layout i lineær tid.

Eksempel

Sequence pair $(A, B) = (\langle e, c, a, d, f, b \rangle, \langle f, c, b, e, a, d \rangle)$



Redundante kanter er slettet.

Sequence pairs: Metaheuristik

Simuleret udglødning

- repræsentation: (Γ_+, Γ_-)
- ombyt rækkefølge af to rektangler i Γ_+
- ombyt rækkefølge af to rektangler i både Γ_+ og Γ_-
- ombyt højde og bredde af rektangel

For hvert skridt: konstruer den tilhørende floorplan i $O(n^2)$ tid
evaluer objektfunktion

29

Forbedringer af Sequence pair

- Takahashi [10]: $O(n \log n)$ to derive enclosing rectangle
- Tang [11] $O(n \log n)$ time to derive enclosing rectangle and coordinates
- Tang [12] $O(n \log \log n)$ time using advanced data structures
- Pisinger [9] $O(n \log \log n)$ and semi-normalized

30

Oversigt (fra forfattere til B^* -tree)

Sequence-pair

- + håndterer non-slicing struktur
- + meget fleksibel repræsentation (pre-placed, etc.)
- stort løsningsrum
- ikke normaliseret [forbedret version findes]
- transformation til placement i $O(n^2)$ tid [forbedret $O(n \log \log n)$]
- kan ikke håndtere bløde moduler direkte

O-tree

- + håndterer non-slicing struktur
- mindre fleksibel end sequence-pair
- + mindre løsningsrum
- + normaliseret
- + transformation til placement i $O(n)$ tid
- + færre bits til at repræsentere
- træ-struktur er irregulær, svær at implementere
- begrænset antal indsættelsespositioner

B^* -tree

- + håndterer non-slicing struktur
- + fleksibel repræsentation (pre-placed, soft, rectilinear)
- + mindre løsningsrum
- + normaliseret
- + transformation til placement i $O(n)$ tid
- + færre bits til at repræsentere
- + areal-forbrug kan udregnes incremental
- mindre fleksibel end sequence pair

31

Diskussion

Hvilken repræsentation skal man vælge?

- Alle 2D-pakninger skal helst kunne genereres
- Få redundante repræsentationer
- Tidsforbrug: flytte modul, rotere modul
- Konvertering til konkret pakning
- Ekstra krav: bløde moduler, faste moduler

32

References

- [1] Y.C.Chang, Y.W.Chang, G.M.Wu, S.W.Wu (2000), "B*-trees: a new representation for non-slicing floorplans", DAC-2000, 458–463.
- [2] N.Christofides, C.Whitlock (1977), "An Algorithm for Two-Dimensional Cutting Problems", *Operations Research* **25** 30–44.
- [3] P.N.Guo, C.K.Cheng, T.Yoshimura (1999), "An O-tree representation of non-slicing floorplans and its applications", DAC-99, 268–273.
- [4] X.Hong, G.Huang, Y.Cai, J.Gu, S.Dong, C-K.Cheng, J.Gu (2000), "Corner block list: An effective and efficient topological representation of non-slicing floorplan", *ICCAD 2000*, San Jose, California.
- [5] S. Martello, D. Pisinger, D. Vigo (2000), "The Three-Dimensional Bin Packing Problem", *Operations Research* **48** 256–267.
- [6] S. Martello, D Vigo (1998), "Exact Solution of the Two-Dimensional Finite Bin Packing Problem", *Management Science* **44** 388–399.
- [7] H.Murata, K.Fujiyoshi, S.Nakatake, Y.Kajitani (1996), "VLSI module placement based on rectangle-packing by the sequence pair". *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems* **15** 1518–1524.
- [8] Y.Pang, C.K.Cheng, T.Yoshimura (2000), "An enhanced perturbing algorithm for floorplan design using the O-tree representation", ISPD-2000, 168–173.
- [9] D. Pisinger (2002), "From sequence pair to semi-normalized placement in $O(n \log \log n)$ time"
- [10] T.Takahashi (1996), "An algorithm for finding a maximum-weight decreasing sequence in a permutation, motivated by rectangle packing problem", *Technical report of IEICE, VLD96*, vol VLD96, no 201, 31–35.
- [11] X.Tang, R.Tian, D.F.Wong (2000), "Fast Evaluation of Sequence Pair in Block Placement by Longest Common Subsequence Computation", *Proceedings of DATE 2000 (ACM)*, Paris, France, 106–110.
- [12] X.Tang, D.F.Wong (2001), "FAST-SP: a fast algorithm for block placement based on sequence pair", *Asia and South Pacific Design Automation Conference*.
- [13] P.van Emde Boas (1977), "Preserving order in a forest in less than logarithmic time and linear space" *Information processing letters*, **6**, 80–82.