EMICRO/SIM 2013
XV Escola de Microeletrônica Sul / 28º Simpósio Sul de Microeletrônica
Porto Alegre, 28 de abril a 3 de maio de 2013

Physical Design and CAD Tools

Marcelo Johann
www.inf.ufrgs.br/~johann
johann@inf.ufrgs.br

Outline

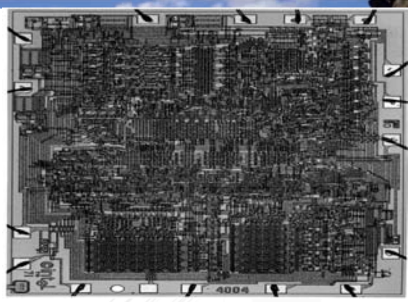
CAD Tools
- The evolution to Chip Design
- Physical Design Flow
- Problems and Solution Space
- Complexity, Graphs and Optimization

Problems and Algorithms
- Placement Algorithms
  SA, TA, Analytical, PEKO
- Routing Algorithms
  Maze Router and A*

Integrated Systems

Scale	Active devices	System Function	Year
SSI	1-100	Gates, OP-Amp, linear App.	1960
MSI	100-1.000	Registers, Filters	1965
LSI	1.000-10.000	Microprocessor, ADC	1970
VLSI	10.000-100.000	Memory, Computers, Signal Processor	1975
ULSI	100.000-50.000.000	Pentium IV.	2001
??SI	50.000.000-7bilion	Multi Core, complex integrated systems	2013

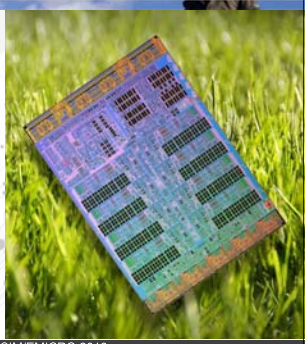
So we went from...



"Simple", "one-man" hand design in 1971

To this kind of designs (actually, old - 2004)

IBM Cell



4 years,
$400M design
400 engineers
234M transistors

People Clearly Need...

Tools

To handle:
- Billions of elements
- Associated complexity (macro and nano)
- Short time-to-market

Tools

A long time ago we discovered the first tools…

Tools

…tools started to get more sophisticated…

Tools

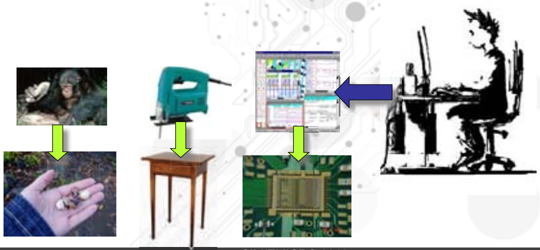
…more sophisticated and complex…

Tools

…and then came VLSI CAD tools!

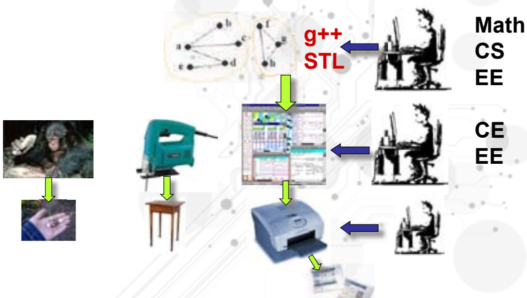
Tools

Designers use CAD tools to make Chips



Tools

And there is a chain of tools that make tools.



Microelectronics is...

Programming
Algorithms
Graph Theory
Optimization

g++
STL

Math
CS
EE

VLSI Design

CE
EE

Foundry

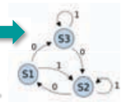
EE
Phy
Che

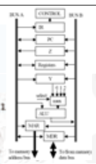
A Design Flow

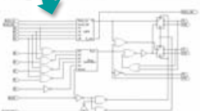
Designer makes initial description (ex: VHDL) and uses a methodology, a set of operations using tools to get the circuit:

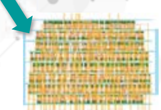
- High-Level Synthesis;
  - Algorithm to RTL
- Logic Synthesis:
  - RTL to mapped netlist
- Physical Synthesis:
  - Placement & Routing


```
process(clk,rst)
begin
if (rst='1') then
  r_reg<= (0=> '1', others =>'0');
elsif (clk'event and clk='1') then
  r_reg<= r_next;
end if;
end process;
r_next<= r_reg(0) & r_reg(3 downto 1);
q<= r_reg;
```









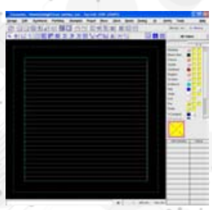
- High-Level Synthesis;
  - **Algorithm to RTL**
- Logic Synthesis:
  - **RTL to mapped netlist**
- Physical Synthesis:
  - **Placement & Routing**

Physical Design Flow Example

- Input
- Floorplanning
- Power Planning
- Placing Standard Cells
- First Optimization Phase
- Clock Tree Synthesis
- Post-CTS Optimization
- Final Routing
- Post-Route Optimization
- Ading Filler Cells
- Check/Verification
- Saving and using Hierarchy
- Chip Place & Route with Macro Blocks
- Simulation and Chip Assembly

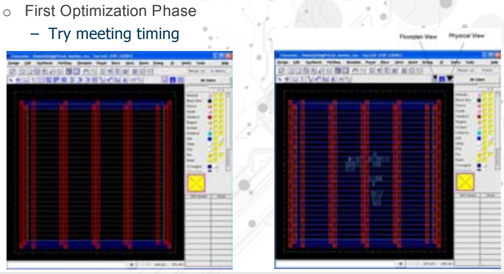
Input and Floorplanning

- Take the result from RTL/Logic Synthesis
  - mapped netlist
  - timing information
- Floorplanning
  - For modules, specify:
    - Aspect ratio
    - Core utilization
    - Core margins
    - Row spacing
  - For the chip
    - Placement of hard and soft blocks



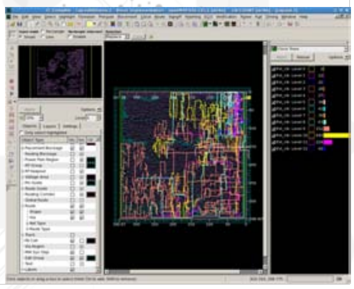
Power and STD Cells

- Power Planning
- Placing Standard Cells
- First Optimization Phase
  - Try meeting timing



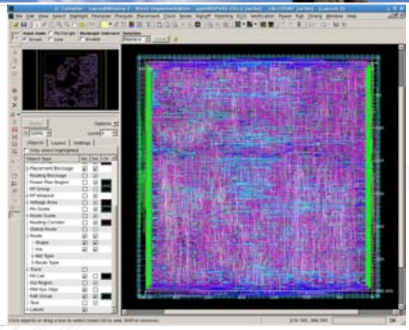
Clock Tree

- Clock Tree Synthesis
- Post-CTS Optimization



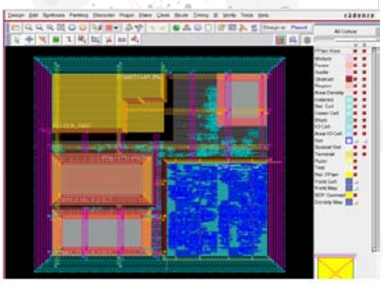
Routing

- Final Routing
- Post-Route Optimization



Additional Steps

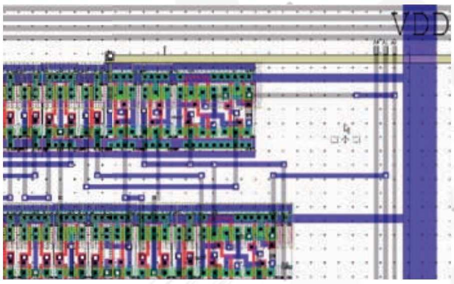
- Ading Filler Cells
- Check/Verification
- Saving and using Hierarchy
- Chip P&R with Macro Blocks
- Simulation
- Chip Assembly



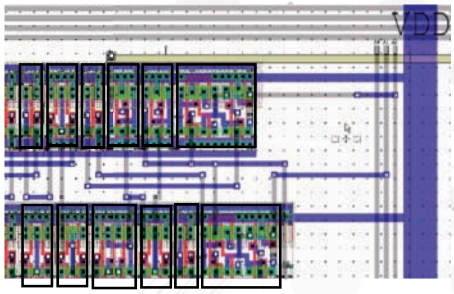
Now, how those tools work?

A L G O R I T H M S

The Placement Problem



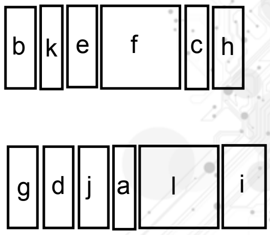
The Placement Problem



The Placement Problem

But why this way?

The Placement Problem

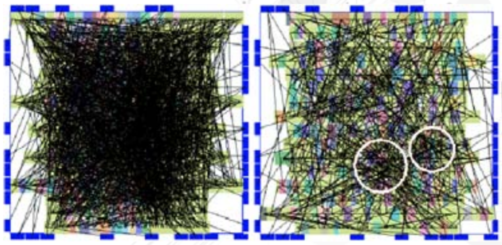


And not this way?

The "Rats Nest"

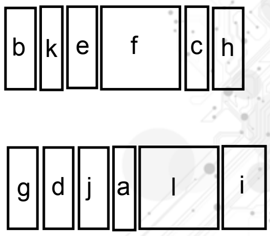
HPWL

Because in some dispositions cells to connect are a lot closer



The Placement Problem

Ok. Lets see how many options we have…



12 !

That's 479.001.600

for a circuit with only 12 cells!!!

Bad news - Complexity

Cobham's thesis

too small: not a concern ⟷ **do need a computer!**

function	n = 2	n = 6	n = 10	n = 20	n = 10^2	n = 10^3	n = 10^6
n	2	6	10	20	10^2	10^3	10^6
3n	6	18	30	60	$3 * 10^2$	$3 * 10^3$	$3 * 10^6$
$n \log_{10} n$	0.6	4.7	10	26	$2 * 10^2$	$3 * 10^3$	$6 * 10^6$
n^2	4	36	10^2	$4 * 10^2$	10^4	10^6	10^{12}
n^3	8	216	10^3	$8 * 10^3$	10^6	10^9	10^{18}
2^n	4	64	10^3	10^6	10^{30}	10^{301}	$> 10^{500}$
n!	2	720	$3 * 10^6$	$2 * 10^{18}$	$9 * 10^{157}$	$> 10^{500}$	$> 10^{500}$

Polynomial

Bad!

Problem Classes

Classe P

- Problemas para os quais existe um algoritmo determinístico em tempo polinomial que os solucionam exatamente

Problem Classes

Classe NP

- Problemas para os quais existe um algoritmo em tempo polinomial que **verifica** se uma **dada** solução é a correta

Classe NP-Completo ⊂ NP

Classe NP-Difícil: ao menos tão difícil quanto NP

Complexity and Classes

P ≠NP ?

Existem problemas cuja resposta pode ser verificada em tempo polinomial, que não possam ser resolvidos em tempo polinomial?

Complexity and Speed

Both are important in practice

Complexity
- When the problem's instance size increases

Speed
- When the problem's instance size is bounded

More bad news:

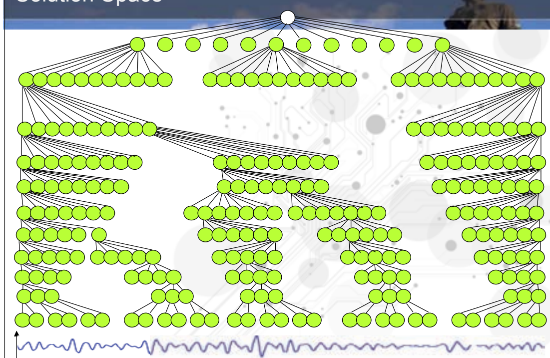
- Most CAD problems are NP-complete or NP-Hard
- So, How do we solve them???

Restricted cases

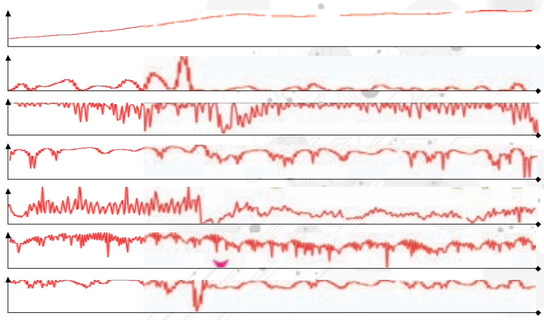
Suboptimal solutions

Good Optimization algorithms

Solution Space



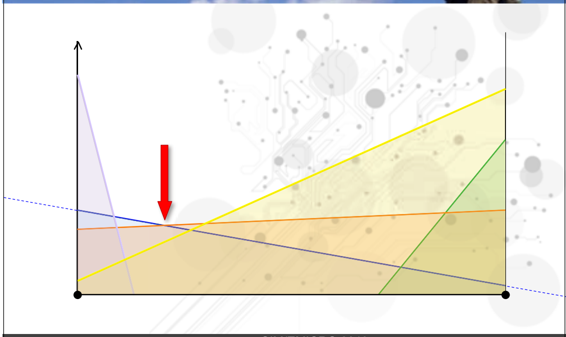
How is this Cost Function?



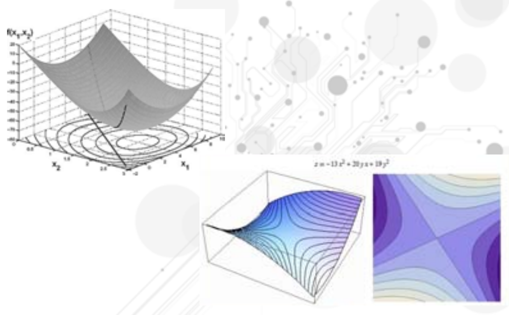
Many Optimization Algorithms

- Otimização Convexa (I've got a model!)
  Pesq. Oper., Progr. linear, quadrática, geométrica
- Mecanismos de Busca
  Guloso, Backtrack, Local, Exaustiva, Branch-and-bound, Programação Dinâmica, Tabu, GRASP
- Algoritmos de Otimização Randomizados ou Meta-Heurísticas
  SA, **TA**, GA, MA, AC

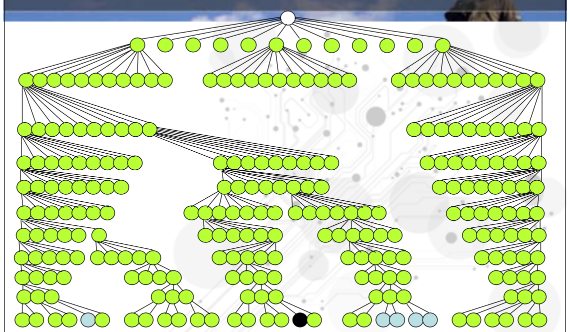
Linear Models



Quadratic Models



Search Procedures



Guloso, Backtrack, Local, Exaustiva, Branch-and-bound, Programação Dinâmica, Tabu, GRASP

Randomized Algorithms

What?

- Use a random number generator to sample or select decisions

Why?

- Better efficiency

Probabilistic, Las Vegas, Monte Carlo

Randomized Example 1

Identifying the repeated element

- Array half filled with a repeated number
- Any deterministic algorithm requires at least (n/2)+1 comparisons
- An algorithm that randomly chooses two elements terminates with *O(log n)* attempts with high probabillity

Randomized Example 2

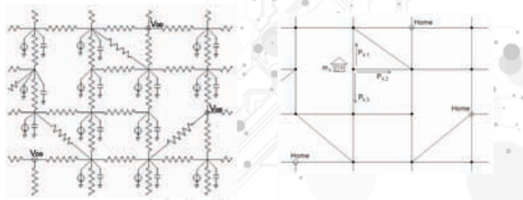
Computing π

```
int main (int argc, char ** argv)
  {
  long experiments, total = 1000000;
  long inside = 0;

  for (experiments=0 ; experiments < total; ++experiments )
    {
    double x = (double) rand()/(double) RAND_MAX;
    double y = (double) rand()/(double) RAND_MAX;
    if (sqrt(x*x+y*y) <= 1)
      ++ inside;
    if ((experiments%10000)==0)
      printf("With %ld experiments PI = %lf\n",
              experiments,(double)4*inside/experiments);
    }
  }
```


Randomized Example 3

Power supply analysis

- Haifeng Qian, Sani R. Nassif, Sachin S. Sapatnekar. *Random Walks in a Supply Network*, **DAC 2003**



Randomized Meta-Heuristics

Meta-Algorithms that work for many optimization problems

- **SA** - Simulated Annealing
- **TA** - Threshold Accepting
- **GA** - Genetic Algorithm
- **MA** - Memetic Algorithm
- **Many others more recently…**

Let's put together...

Placement Problem

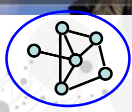
Simulated Annealing

…and we need a graph

Graph Theory

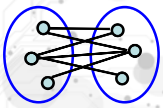
Graph

A Set and a set or Pairs



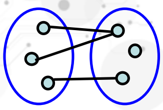
Relation

Each element of the pair belongs to one of the sets



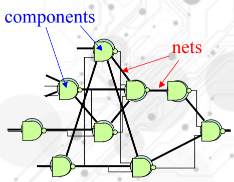
Function

There is a single element from the second set associated to each element of the first set



Representation

Circuit represented as a Graph *G=(V,E)*



Simulated Annealing in 512 bytes

Marcelo Johann

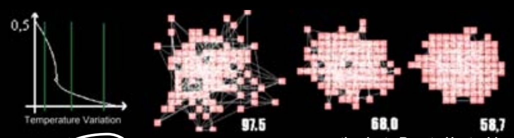
EMICRO2004

Microelectronics School

BRAZIL

SBC GME UFRGS

Annealing

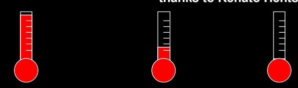


thanks to Renato Hentschke

Perturbation

Cost

Reject / Accept



Three basic functions: Temperature Schedule (above), Cost (wire length) and Perturbation (changes in current solution).

S.A. in 512+ bytes

Looks crazy, but works!

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

A big number!!!

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Loop *a* times
Read *a* from *stdin*
Exchange *k[a]* with *k[b]*

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

x,y position of each node

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```

S.A. in 512+ bytes

source and target of each arc

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Input graph reading: number of cells and wires, number of rows and list of wires (node node)

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Input graph reading: number of cells and wires, number of rows and list of wires (node node)
initial placement

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Cost function: adds the length of all arcs in the graph

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Temperature scheduling

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Perturbation: **exchange** a random selected pair of cells

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```

S.A. in 512+ bytes

Accepting criteria (inverted)
Reject if the cost is higher and the probability is not satisfied

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512+ bytes

Probabilistic: inversely proportional to the temperature

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


S.A. in 512 bytes

Take off cost reports to get your 512-byte SA placer!

```
#include<stdio.h>
#include<math.h>
#define z 9999
#define F(a)for(i=0;i<a;++i)
#define R(a)scanf("%d",&a);
#define T(k)d=k[a];k[a]=k[b];k[b]=d;
long a,b,c=0,i,v,e,d,l,X[z],Y[z],E[9*z],G[9*z];double t;
w(){int i;d=0;F(e)d+=abs(X[E[i]]-X[G[i]])+abs(Y[E[i]]-Y[G[i]]);return d;}
main(){R(v)R(e)R(a)F(v){X[i]=i/a;Y[i]=i%a;}F(e){R(E[i])R(G[i])}
for(t=z*a;c++<z*9;t*=.9){F(1){a=rand()%v;b=rand()%v;l=w();T(X)T(Y);
if(d=(w()-l)>0)if(rand()%z>z*exp(-d/t)){T(X)T(Y)}}}
F(v)printf("%d %d %d %d\n",i,X[i],Y[i],w());}
```


Now “Threshold Accepting”

Simulated annealing does:

```
While (temp*=0.99 > finalvalue)
  {
  make_perturbation();
  if ( newcost < actualcost  || rand() < e^(-deltacost / temp) )
     accept();
  else
     reject();
  }
```


Threshold Accepting

Threshold Accepting is solely:

```
While (threshold*=0.99 > finalvalue)
  {
  make_perturbation();
  if ( newcost < actualcost + threshold )
     accept();
  else
     reject();
  }
```

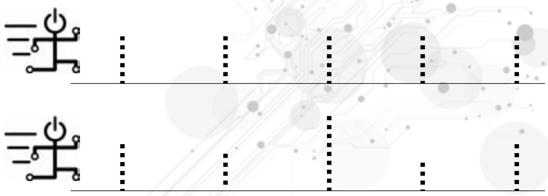
It is easier **and more efficient**

but Why???

Threshold Accepting

1) Integrating this constant acceptance leads to a similar decreasing probability of bad moves

2) It is easier to advance:



Threshold Accepting

Results **1** - **10** of about **1,870,000** for **"**simulated annealing**"**. (**0.26** seconds)

Results **1** - **10** of about **28,000** for **"**threshold accepting**"**. (**0.18** seconds)

Results **1** - **10** of about **145** for **"**bachelor acceptance**"**. (**0.26** seconds)

Using S.A. And T.A.

Temperature (or threshold)
- Initial identification, adaptive schedule, for how long, low temperature…

Neighborhood
- Check for valid perturbations

Cost shape
- Stratification, global behavior, …

Other issues
- Heuristic moves, windowing…

ALTERNATIVAS

Are there other ways of doing placement?

Of course!

Quadratic Placement

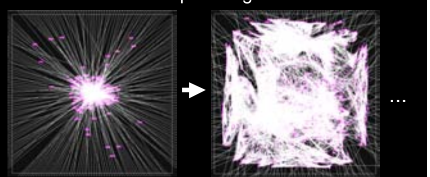
Clustering
Quadratic Placement
Global Legalization
Detailed Place
Detailed Legalization
Placement Refinement

$$\Phi(\mathbf{x},\mathbf{y}) = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}\left[\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right]^2 = \underbrace{\frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(x_i - x_j)^2}_{\Phi(\mathbf{x})} + \underbrace{\frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(y_i - y_j)^2}_{\Phi(\mathbf{y})}$$

$$\Phi(\mathbf{x}) = \begin{bmatrix}\mathbf{x}_m^T & \mathbf{x}_f^T\end{bmatrix}\begin{bmatrix}\mathbf{Q}_{mm} & \mathbf{Q}_{mf} \\ \mathbf{Q}_{fm} & \mathbf{Q}_{ff}\end{bmatrix}\begin{bmatrix}\mathbf{x}_m \\ \mathbf{x}_f\end{bmatrix} \xrightarrow{\text{derivative}} \mathbf{Q}_{mm}\mathbf{x}_m = -\mathbf{Q}_{mf}\mathbf{x}_f$$

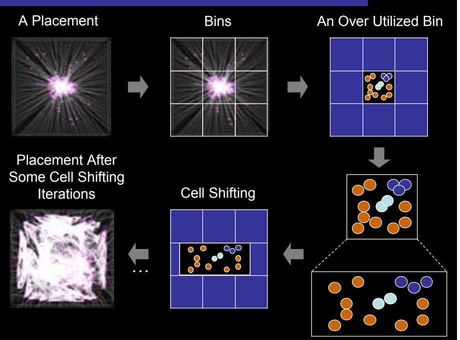
Linear System of Equations

cell spreading



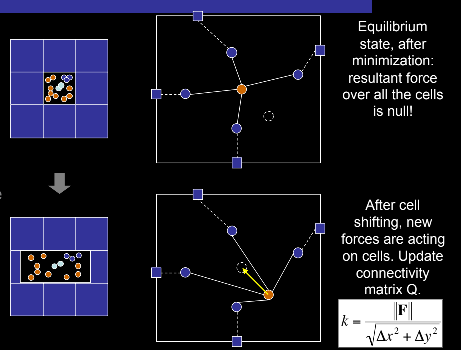
Cell Shifting

Clustering
Quadratic Placement
Global Legalization
Detailed Place
Detailed Legalization
Placement Refinement

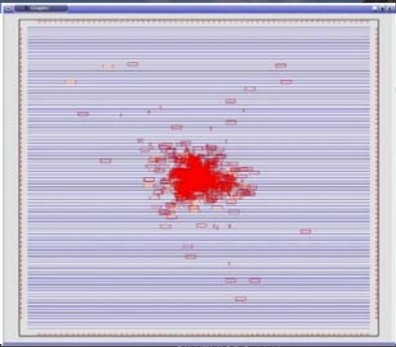


Adding Spreading Forces

Clustering
Quadratic Placement
Global Legalization
Detailed Place
Detailed Legalization
Placement Refinement



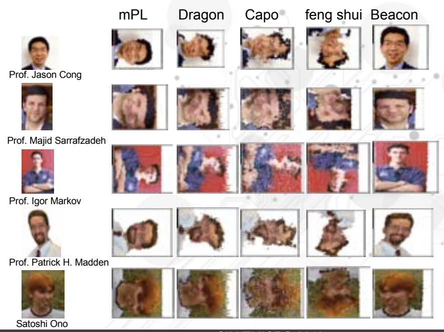
Analytical Placement - FastPlace



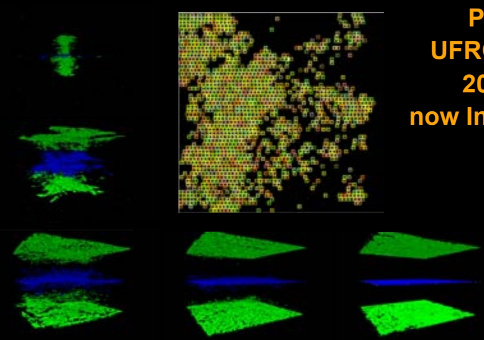
PEKO

- **Qual seria a solução ótima?**
- Jason Cong (UCLA) 2002
  - "Inventa" circuitos com posicionamento ótimo
  - Respeita qualquer distribuição de redes
- Falhas
  - Circuitos são malhas, sem globais
  - Peko killer e outros trabalhos;
- Placement Contest no ISPD2005
  - Patrick Madden mostra visualmente a qualidade dos posicionadores em circuitos Peko

Feng Shui Suboptimality Gallery



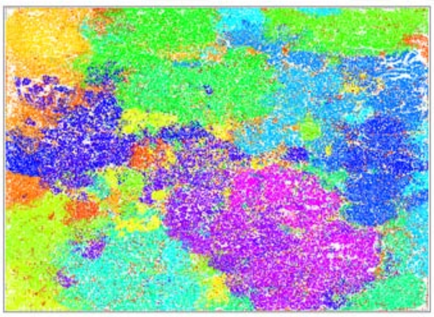
Renato Hentschke's ZPlace



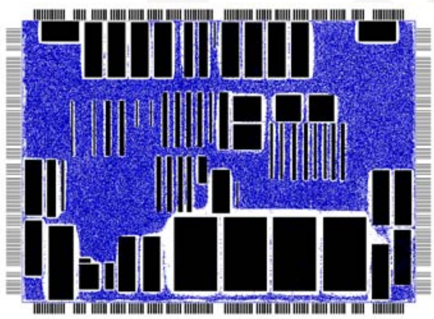
PhD
UFRGS
2007
now Intel

Flach's PlaceDL and Colors

MsC UFRGS 2010, PhD ongoing



PLaceDL and the "Rock and Sand" problem



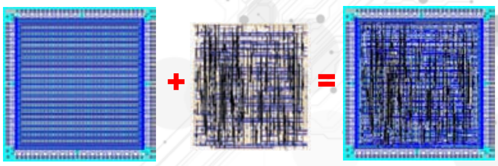
Placement and Floorplanning

From Igor Markov's webpage

The Routing Problem

After placement

- Lets see how we can add connections to the cells so that we build a real circuit



Why Routing still Important Today?

Many technology/design issues

- Scaling…
- Timing…
- Buffering…

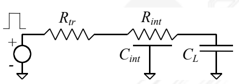
Why Routing still Important Today

Atraso de porta (T_{gate}):

- pode ser reduzido com o reprojeto da porta

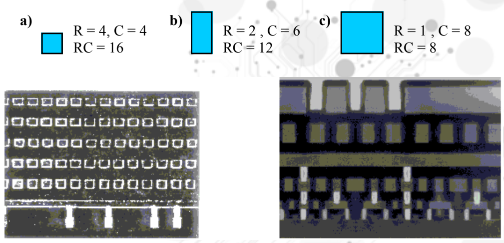
Atraso de interconexão (T_{int}):

- exige o reprojeto das próprias interconexões



$$T_{50\%} = T_{gate} + T_{int}$$
$$T_{gate} = R_{tr}\,(0.7\,C_{int} + 0.7\,C_L)$$
$$T_{int} = R_{int}\,(0.4\,C_{int} + 0.7\,C_L)$$

Reverse Scaling



Camadas têm características diferentes

Routing Algorithms

- **genéricos**: podem ser aplicados a muitos problemas diferentes de roteamento;

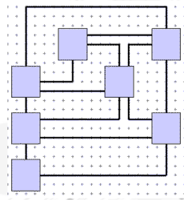
  ex: ***maze routers*** = *breadth-first search*
- **restritos**: exploram características ou restrições particulares de um problema para encontrar soluções com maior eficiência;

  ex: **roteamento de canal**

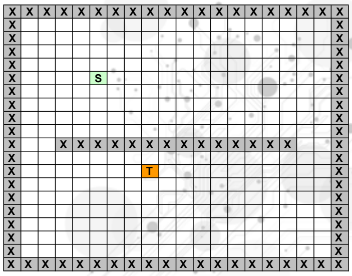
  caixas de conexão

  roteamento planar ...

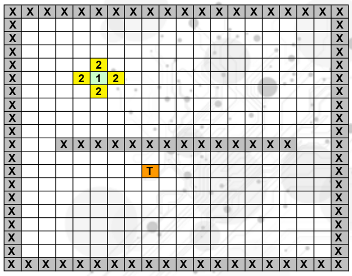
The MAZE Router



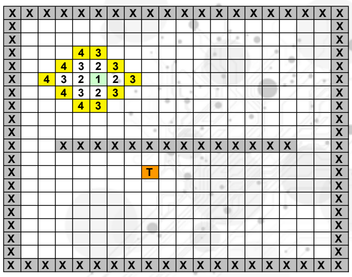
The MAZE Router



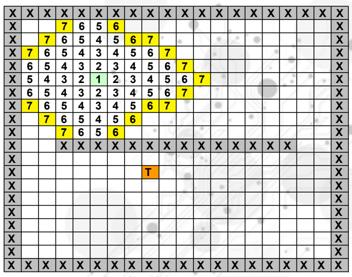
The MAZE Router



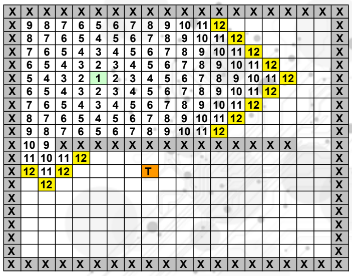
The MAZE Router



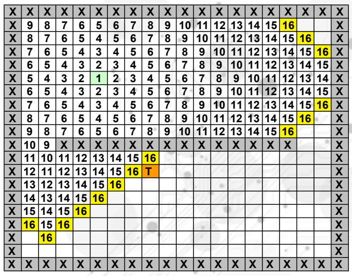
The MAZE Router



The MAZE Router



The MAZE Router



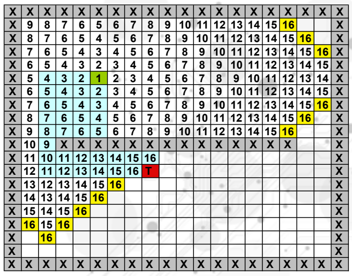
Maze Router's Expansion

g++

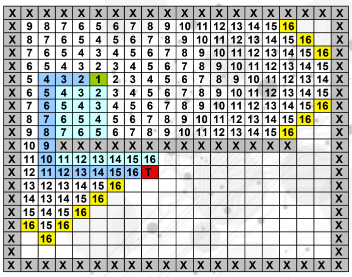
```
#include <iostream>
#include <queue>
using namespace std;
#define SIDE 20
#define PLACE(x,y) ((y)*SIDE+(x))
#define WEST(n) (n-1)
#define EAST(n) (n+1)
#define NORTH(n) (n-SIDE)
#define SOUTH(n) (n+SIDE)
char Space[SIDE*SIDE];
void init (void)
  {
  for (int i=0; i<SIDE*SIDE; ++i)
    Space[i] = ' ';
  for (int i=0; i<SIDE; ++i)
    {
    Space[PLACE(i,0)] = 'X';
    Space[PLACE(i,SIDE-1)] = 'X';
    Space[PLACE(0,i)] = 'X';
    Space[PLACE(SIDE-1,i)] = 'X';
    }
  for (int i=3; i<SIDE-3; ++i)
    Space[PLACE(i,10)] = 'X';
  }
```

```
void print(void);
int bfs(int source, int target)
  {
  queue<int> q;
  q.push(source);
  while (!q.empty())
    {
    int n = q.front();
    q.pop();
    if (n==target)
      return 1;
    if (Space[n] != 'X')
      {
      Space[n] = 'X';
      print();
      getchar();
      q.push(WEST(n));
      q.push(EAST(n));
      q.push(NORTH(n));
      q.push(SOUTH(n));
      }
    }
  return 0;
  }
```


The MAZE Router



The MAZE Router



Maze Router's Tracking

g++

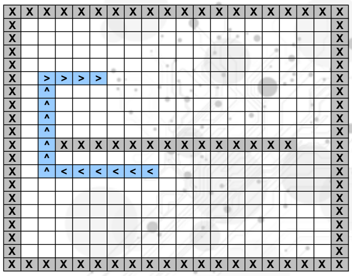
1-You have to store the node and where it came from in the queue using a pair<int,int>

2-Instead of Space, use Expansion[n] to store where the nodes came from and Taken[n] to represent obstacles and routes

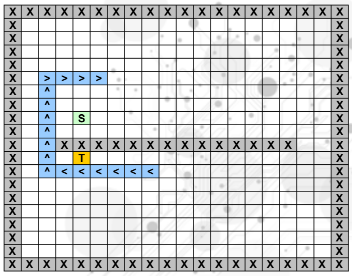
4-Write backtrack(t,s) from target to source following Expansion[n]

```
typedef pair<int,int> Ref;

char Expansion[SIDE*SIDE]; // 0
char Taken[SIDE*SIDE]; // ' '

int bfs(int source, int target)
  {
  queue<Ref> q;
  q.push(Ref(source,0));
  while (!q.empty())
    {
    int n = q.front().first;
    int p = q.front().second;
    q.pop();
    if (n==target)
      return 1;
    if (Expansion[n] == 0 &&
      Taken[n] == ' ')
      {
      Expansion[n] = p;
      q.push(Ref(WEST(n),n));
      . . . }
    }
  return 0;
  }
```

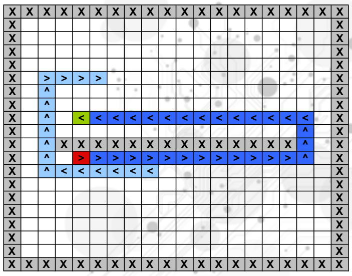
The MAZE Router



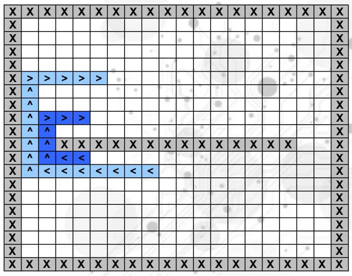
The MAZE Router



The MAZE Router

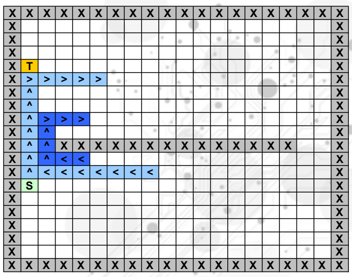


The MAZE Router



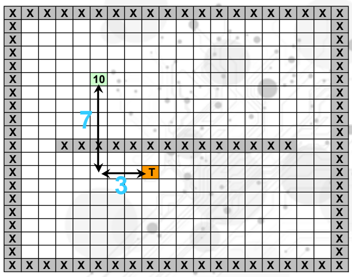
We can try another ordering…

The MAZE Router



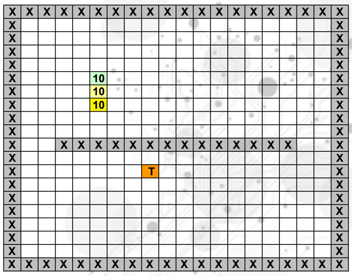
…or another ordering…

The A* ALgorithm



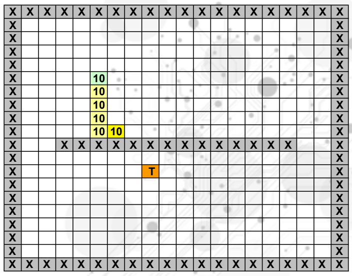
h = estimation function

The A* Algorithm



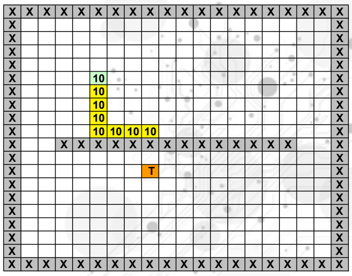
Min f = g + h **Max g**

The A* Algorithm



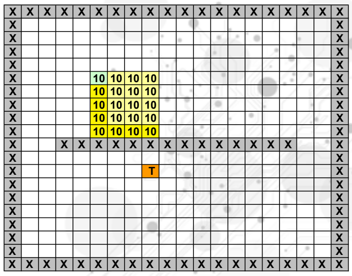
Min f = g + h **Max g**

The A* ALgorithm



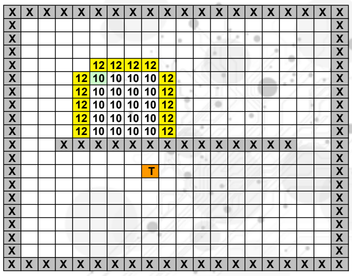
Straingth path from S to T

The A* ALgorithm



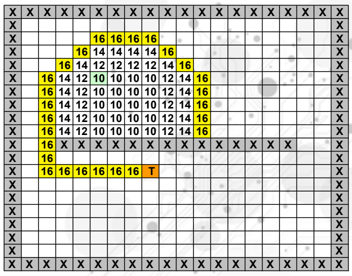
Degrees of freedom

The A* Algorithm



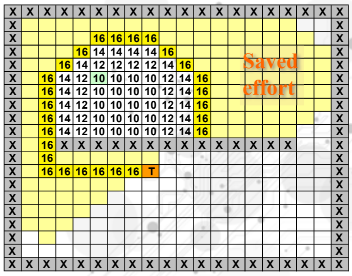
Maze router-like with obstacles

The A* Algorithm



Maze router-like with obstacles

The A* Algorithm



So, Maze Router...

has:

Nice Properties

- It always finds a path if there is one
- It always returns the shortest possible path

but:

Advantages turn to be problems…

- The paths might be too complicated
- We rarely need “THE” optimal path…

On the other hand...

Maze Routers are known for having

3 Big Problems

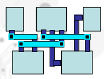
- Memory
- Speed
- Ordering

But today:

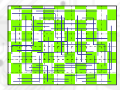
- We have plenty of memory, in general
- Speed can be addressed by A*, Hannan Grids
- Ordering can be dealt with by **NEGOTIATION**

Routing Scopes/Applications

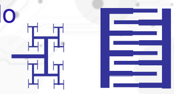
- Roteamento detalhado



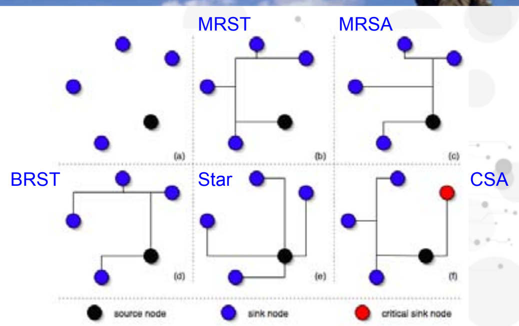
- Roteamento global



- Roteamento especializado

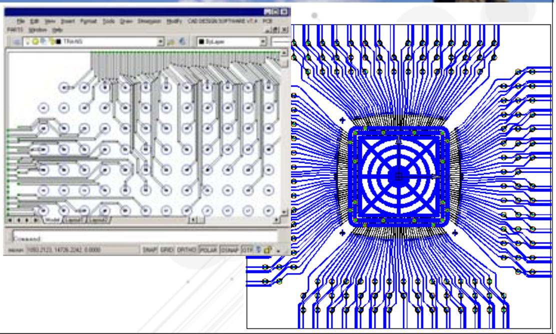


(Steiner) Tree Topologies



Clock Routing

Package Routing



Other CAD tools

- Síntese - agregam informação ao projeto
- Extração (análise)
- Simulação
- Verificação Formal
- Otimização
- Visualização
- Processamento para fabricação
- Projeção para novas tecnologias (tech CAD)

EMICRO/SIM 2013
XV Escola de Microeletrônica Sul / 28º Simpósio Sul de Microeletrônica
Porto Alegre, 29 de abril a 3 de maio de 2013

Thank you!

Physical Design and CAD Tools

Marcelo Johann
www.inf.ufrgs.br/~johann
johann@inf.ufrgs.br