

MIT

# *L'Informatica verso il terzo millennio*



L'Informatica vista dalle università  
americane

*Diego Puppini, MIT*

diego@mit.edu

# Agenda

- Cosa stanno cercando di fare nelle università americane
- La cucina di domani
- Come cambierà il modo di leggere  
*pausa*
- Architetture avanzate
- Algoritmi randomizzati per architetture VLIW



# *L'informatica come appare al MIT*

- Chi c'è ad Informatica? Stallmann, Minsky, Papert, W3C, Legs Lab, Medialab, AI Lab
- Prima c'era Athena, ora c'è...



# Oxygen

- Sistema distribuito, con interfacce visive e vocali, accessibile in ambienti intelligenti o con dispositivi portatili - services and software objects must be named by intent, for example, "the nearest printer".
- Computing diffuso (*pervasive and free as air*)
- Obiettivo: permettere alle persone di fare di più facendo meno
- Oxygen technologies represent a bigger shift than those that led from the mainframe to the desktop.



# *Rationale*

In the future, computation will be freely available everywhere, like batteries and power sockets, or oxygen in the air we breathe. Computation will enter the human world, handling our goals and needs. Devices and people will communicate naturally, using speech and vision.

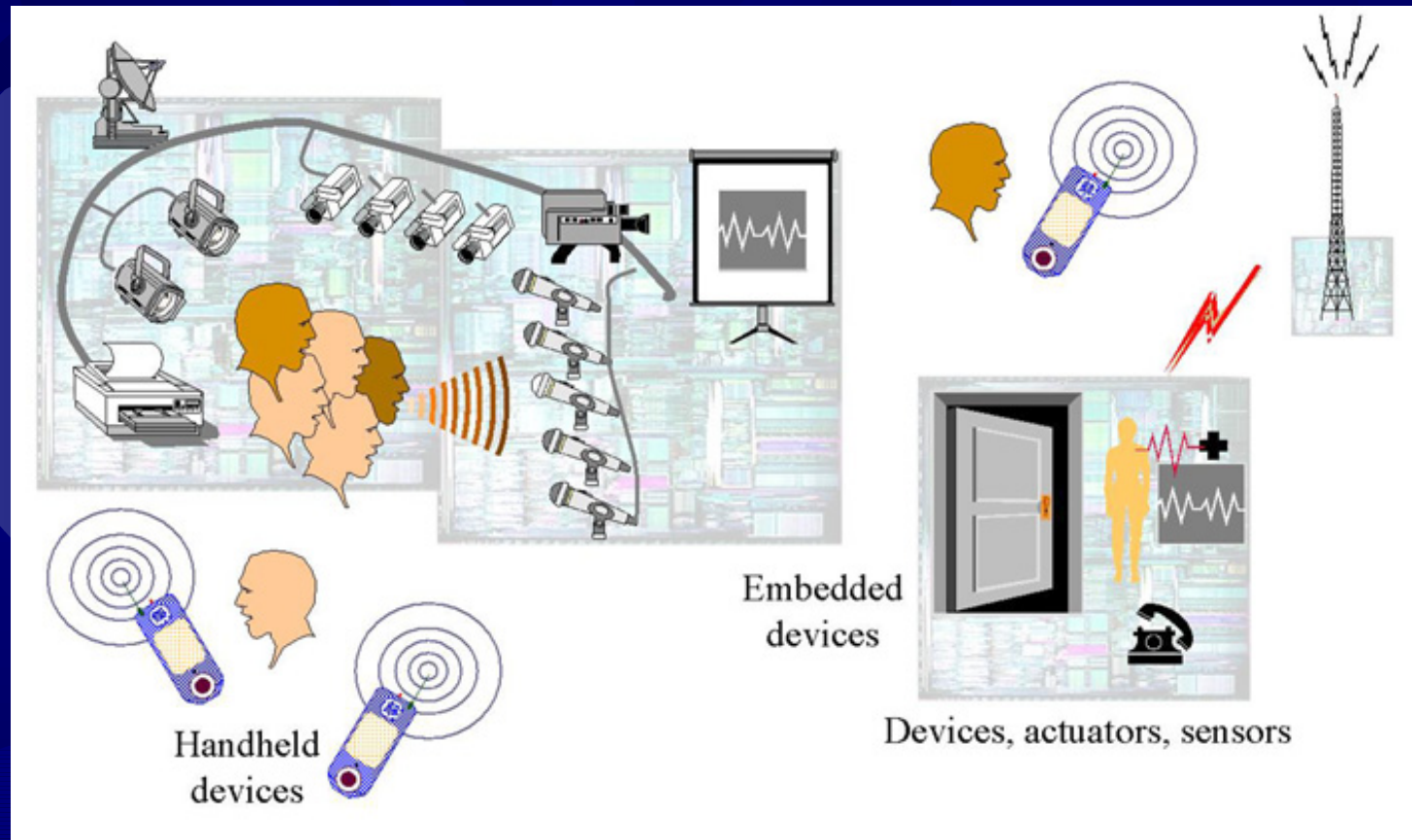
The Oxygen system must be pervasive, embedded, nomadic, eternal.



# *Oxygen user technologies*

- **Automation technologies**, they allow users to control doors or heating systems according to their tastes.
- **Collaboration technologies**, which provide support for recording and archiving speech and video fragments from meetings...
- **Knowledge access technologies**, which offer greatly improved access to information, customized to the needs of people, applications, and software systems.

# *Come sara'*



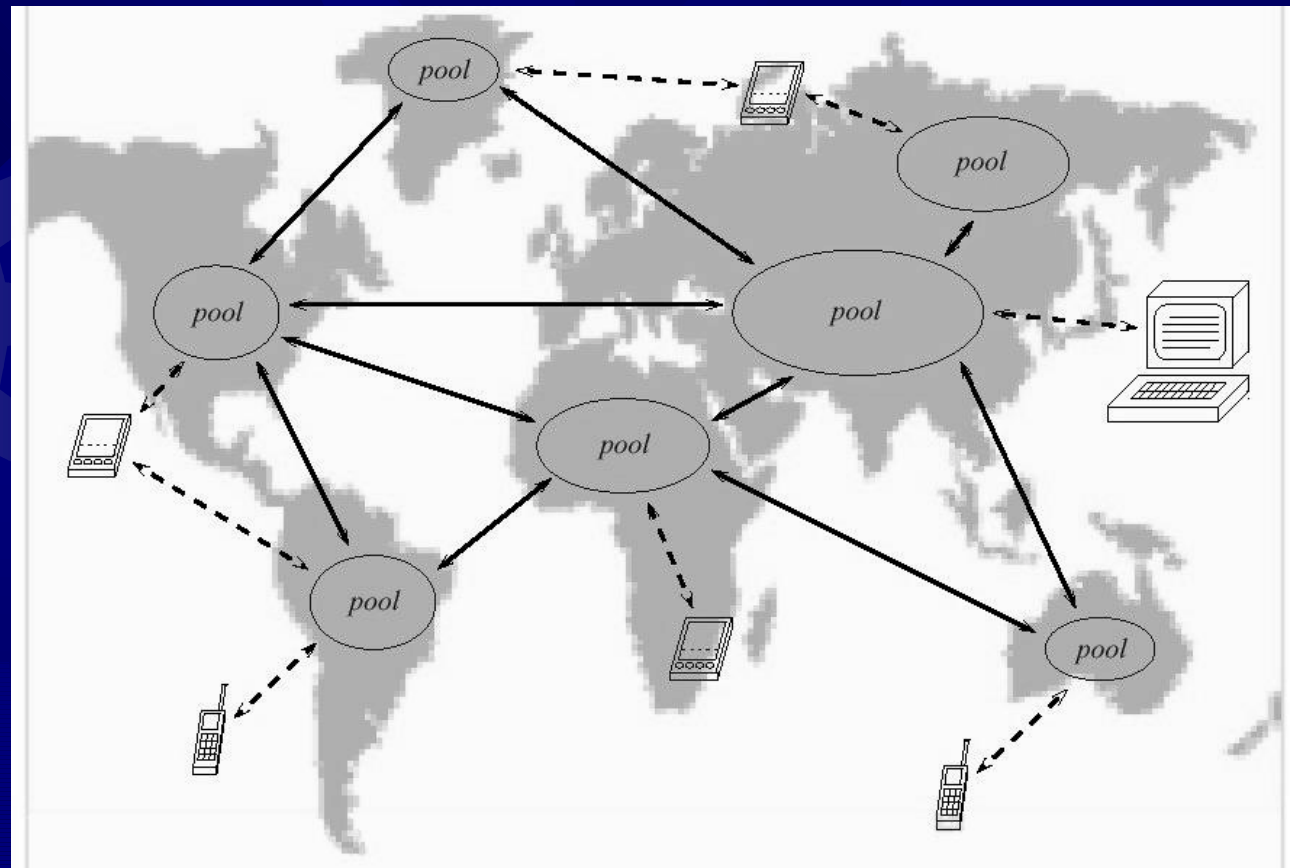


# *Progetti di ricerca*

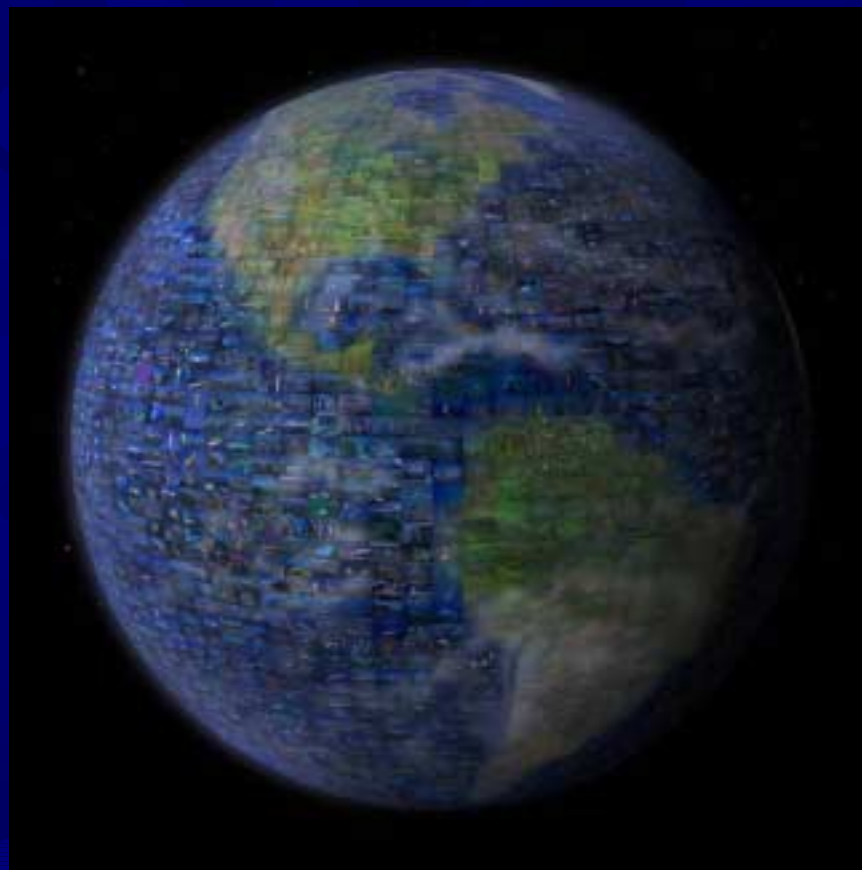
- Intelligent Room
- Speech Recognition
- Haystack (knowledge access)
- RAW Architecture
- SCALE Compiler
- Robotics
- Artificial Intelligence



# *Ci provano anche a Berkeley...*



# *Benvenuti al Medialab*



April 18, 2001

Scheduling for VLIW

10

# *Benvenuti alla cucina del terzo millennio*

- Computer nel frigo e nel forno
- Il frigorifero sa se manca qualcosa e la ordina a [homeruns.com](http://homeruns.com)
- Il forno controlla la cottura
- Telecamere controllano la preparazione dei cibi e correggono
- La cucina aiuta a controllare diete e calorie
- Monitor ovunque, la cucina come ufficio

# *La cantina al Medialab*

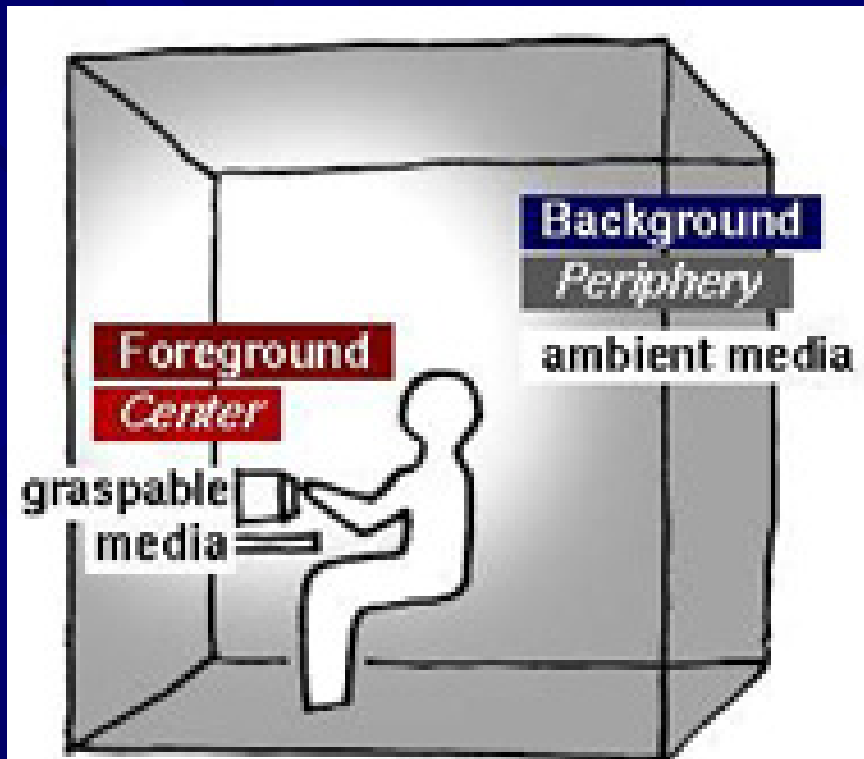


April 18, 2001

Scheduling for VLIW

12

# *Ambiente intelligente*



*drawing: Hiroshi Ishii*



# *La lettura nel terzo millennio*

April 18, 2001

Scheduling for VLIW

14



## *Come leggeremo...*

- Allo Xerox/PARC studiano e cambiano i modi della lettura
- Classificazione delle modalità di lettura oggi
- Aumentare il libro
- Le nuove tecnologie come nuovi supporti per la lettura

# *I modi della lettura*

	Sincrono	Asincrono
Verticale	Slides	Cartelloni
Orizzontale	Appunti	Libro
	Volontario	Inconscio
Statico	Giornale	Maglietta
Volatile	Web	Banner



# *Hyperbolic Reader*



- Tecnologia Inxight: i dati diventano più piccoli al bordo
- Il contesto è sempre visibile
- Permette di esplorare uno spazio senza perdersi (Internet?)

# *Speeder Reader*



- Le parole sono mostrate una alla volta ad alta velocità
- Velocità di lettura maggiore
- E' possibile accelerare per aumentare il flusso di informazione

# *Reading-Eye Dog*



- Sostituto del nonno che racconta le favole
- Gli occhi sono scanner + OCR che leggono il testo presentato
- Legame affettivo del pubblico giovane

# *Walk-in Comix*



- Fondere architettura e letteratura
- Leggere con tutto il corpo
- Lo spazio della narrazione diventa lo spazio fisico

# *Listen Reader*



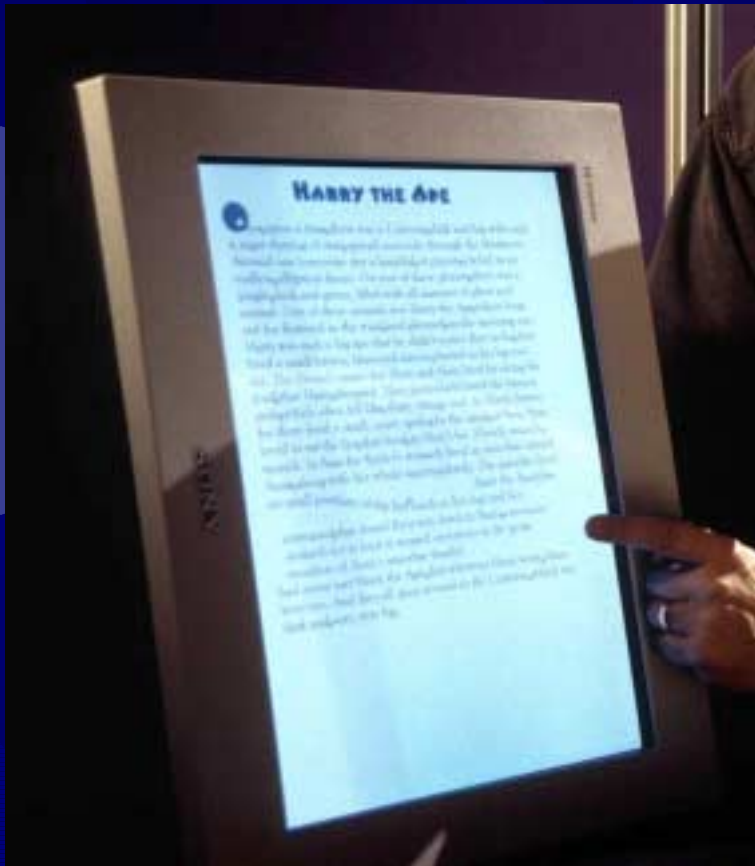
- Libro aumentato
- Le pagine sono arricchite di sensori, che fanno suonare il testo
- Aumenta l'immersione offerta dal libro

# *Tilty Table*



- Permette di navigare il testo: quando inclinata, il testo scorre
- Browser di informazione su grandi dimensioni
- Coinvolgimento fisico nella lettura

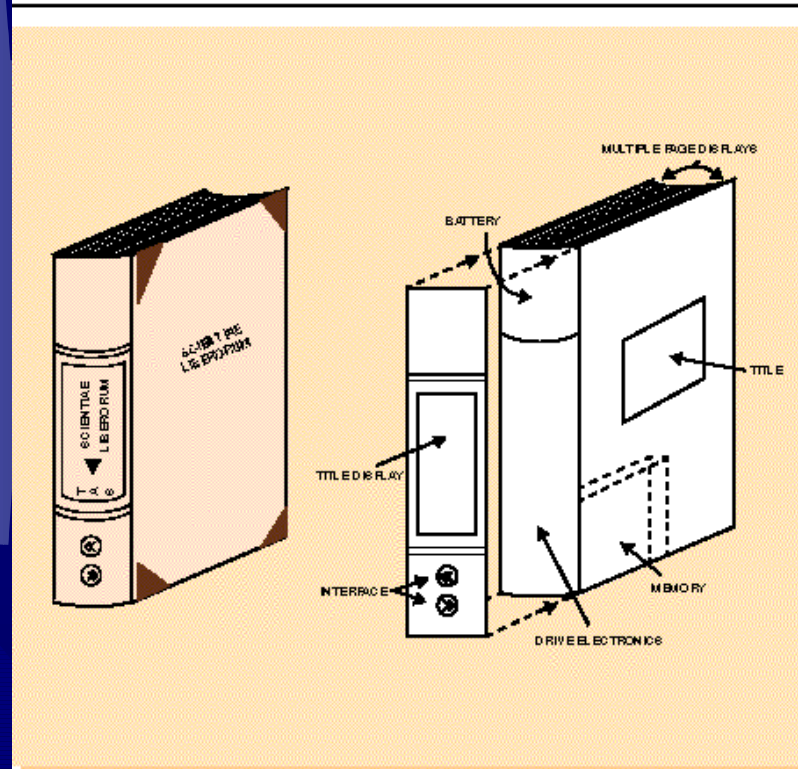
# Liquid Text



- Piccoli hyperlink nel testo: il testo si sposta per fare spazio alle nuove informazioni
- Viene mantenuto il contesto
- Annidamento dell'informazione diventa annidamento spaziale

# Il libro di domani

Figure 1 Schematic of the electronic book



- Inchiostro controllabile elettricamente (e-ink)
- Le pagine possono rappresentare le pagine di ogni testo
- Scaricare letteratura sul libro
- Animazioni a 20Hz, pagine multimediali



Figure 1 Schematic of the electronic boot

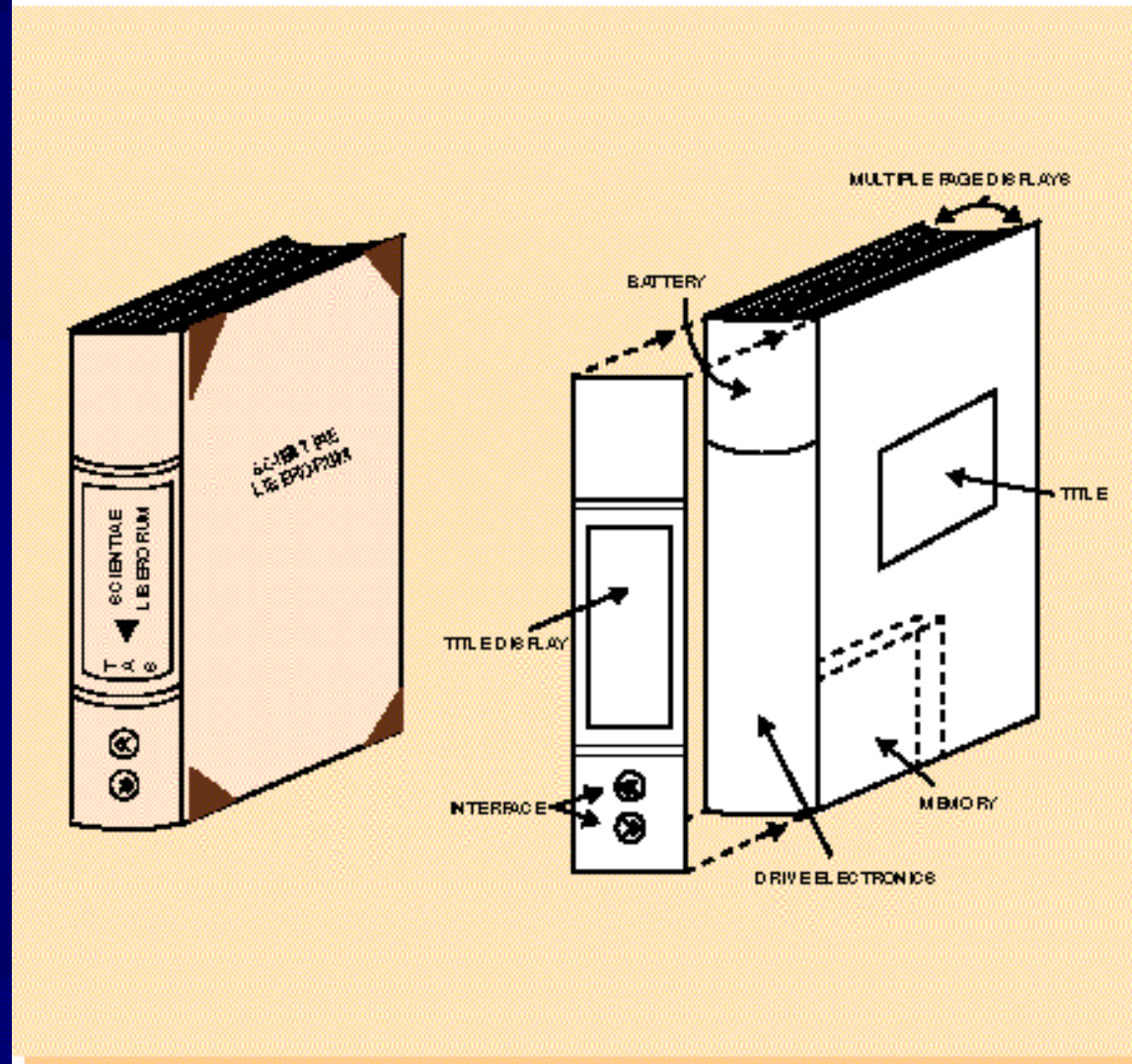
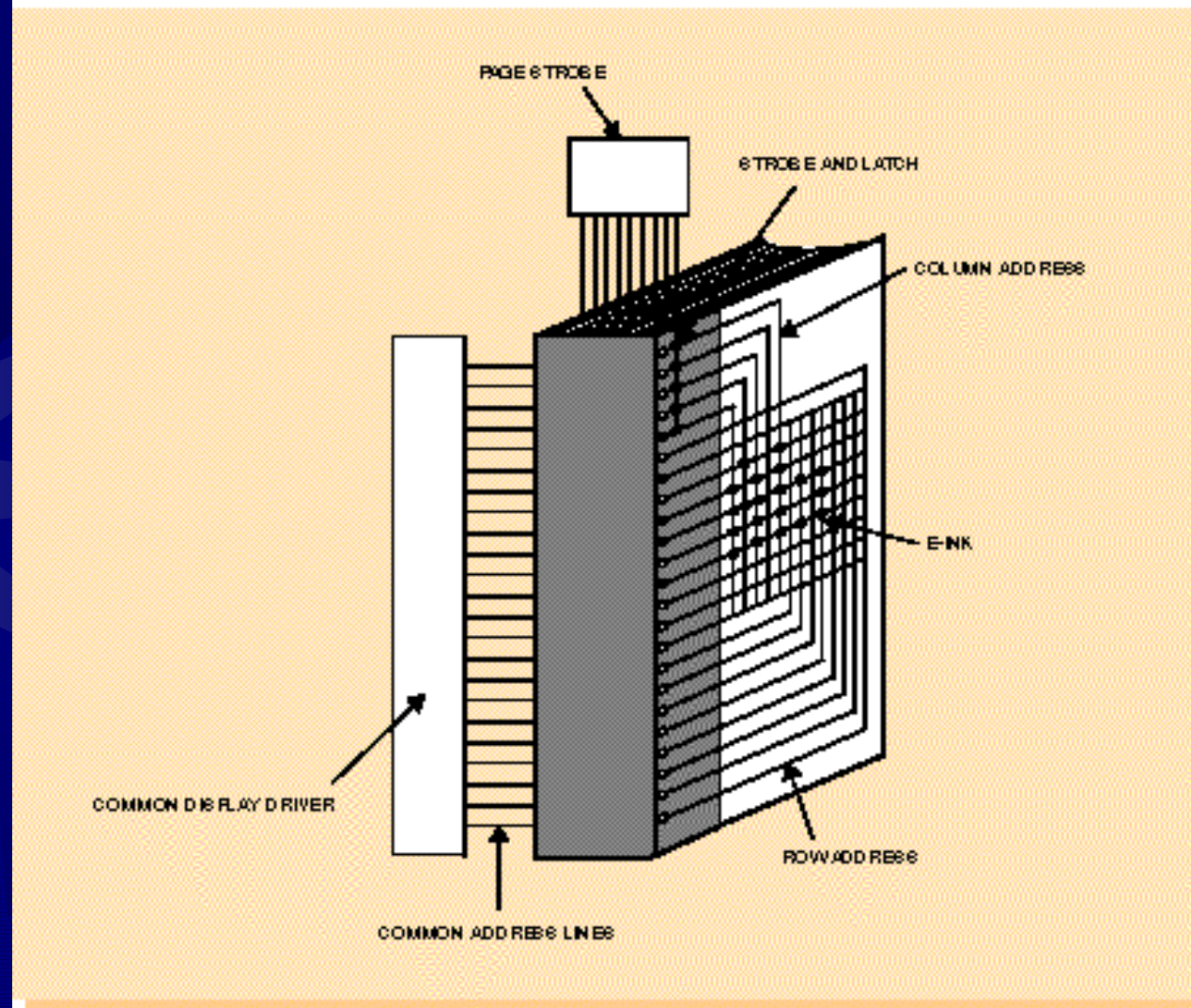


Figure 5 Schematic of drive logic and pagelayout for electronic book



# *Morale*

- Nessuno leggerà mai un libro sul suo palmare...
- ... ma nuove forme di testo si affiancheranno ad esso.



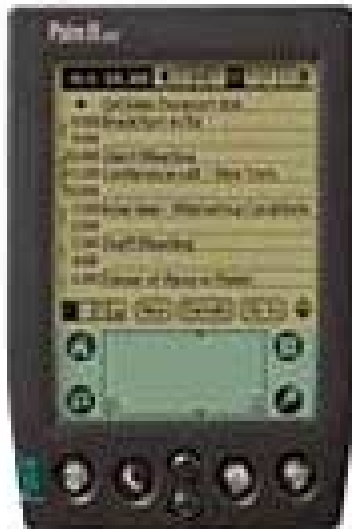
*pausa*

April 18, 2001

Scheduling for VLIW

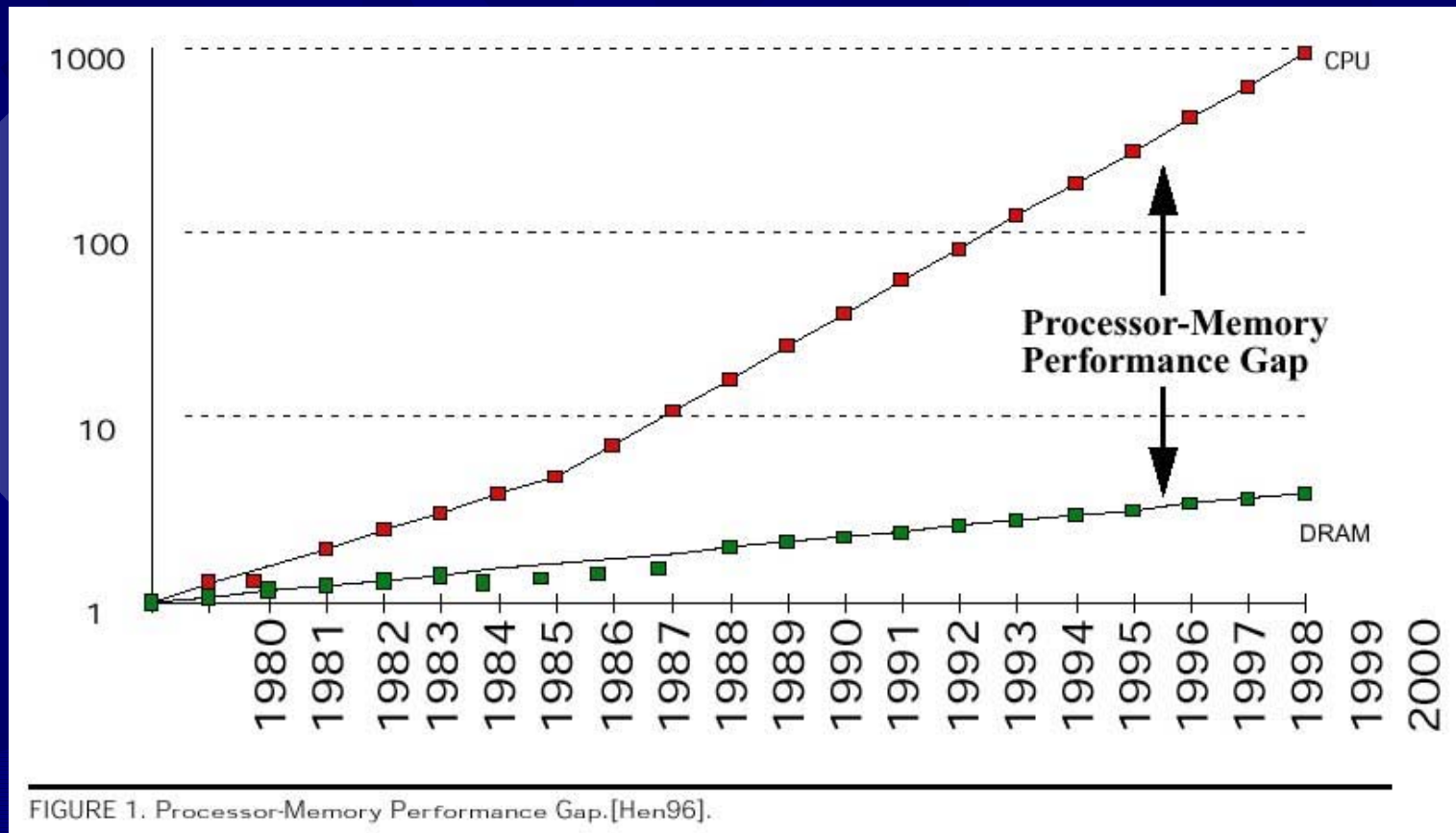
28

# *Verso 1 miliardo di transistor su chip*



- Come sfruttare lo spazio
- Crescente gap memoria-processore
- Applicazioni multimediali con architettura streaming
- Miglior utilizzazione dell'energia per dispositivi portatili

# Memory-proc. gap

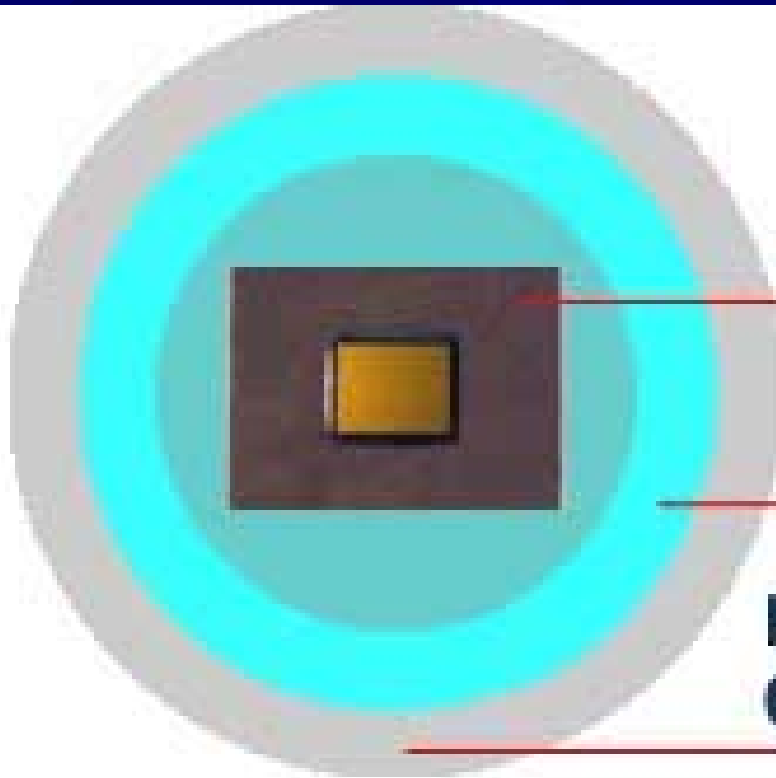




# *Alcune proposte*

- Crusoe
- IRAM
- RAW
- SMT

# *Crusoe*



**VLIW CPU**

**Code Morphing™ Software**

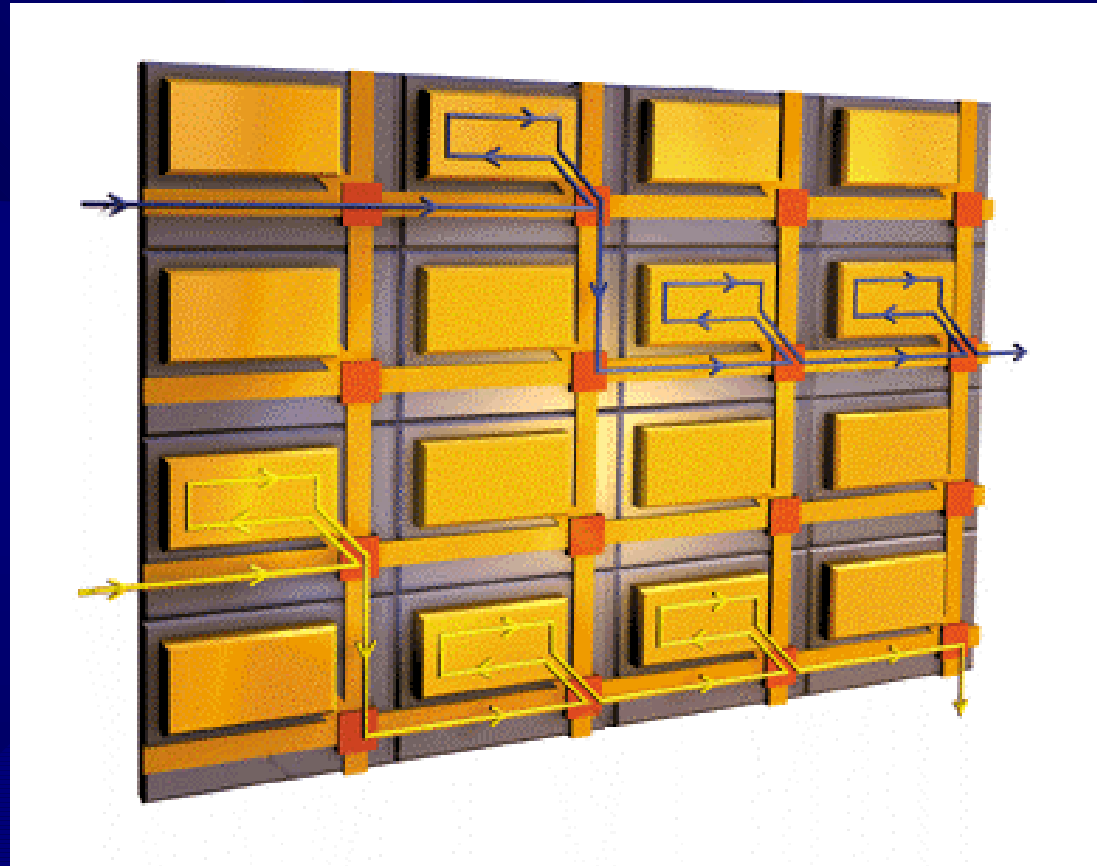
**PC, Internet Applications  
Operating Systems**



# *Crusoe*

- Processore VLIW (semplificato rispetto x86)
- Emulatore x86 a software (Code Morphing)
- Cache di codice tradotto
- Codice ottimizzato a run-time
- Migliore consumo energetico
- Minor costo di design, verifica e aggiornamento

# *RAW, design modulare*





# Scheduling for VLIW

How with more  
RANDOMIZATION

April 18, 2001

Scheduling for VLIW

35

# *Agenda*

- What's VLIW? What's clustered VLIW?
- Scheduling with Integer Programming
- Scheduling for Clustered VLIW
- My favourite one
- Future directions



# *Clustered VLIW*

April 18, 2001

Scheduling for VLIW

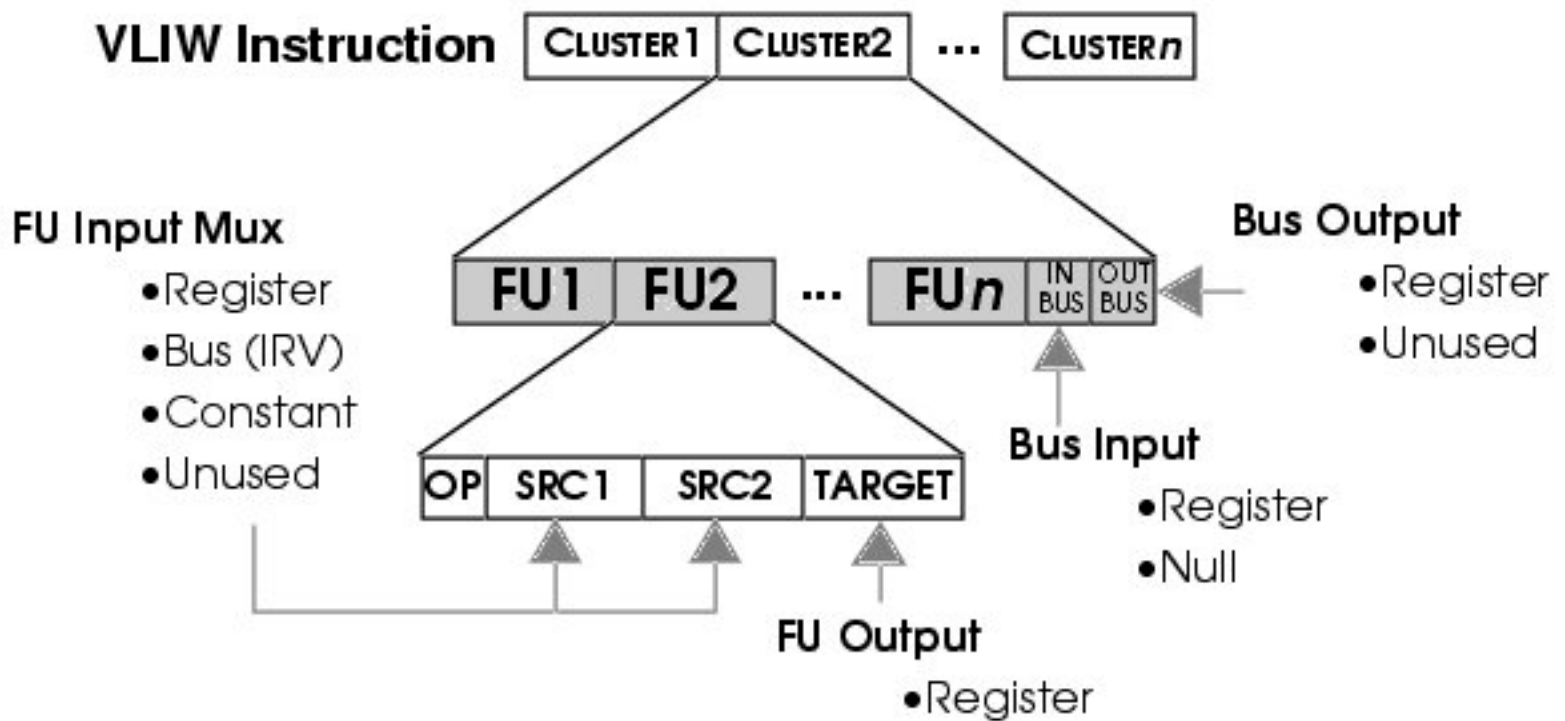
37



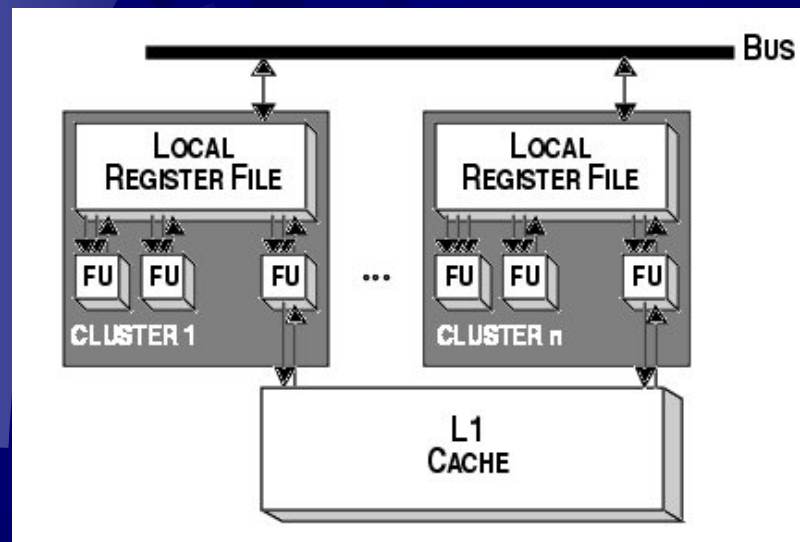
# *What is VLIW*

- Very Long Instruction Word
  - Explicit control of all the functional unit
  - Compiler is responsible for scheduling
  - Reduced hardware support
- Rationale
  - Simpler design
  - Reduced design and test cost
  - Low power consumption
  - Higher clock frequency

# VLIW



# Clustered VLIW



- Minimize the impact of a high number of R/W ports in the reg. file
- Functional units can use just local register
- Copies among the clusters, as needed



# Compiling Problems

- Partitioning!
- We have to manage copies
- Ideally, instruction selection, partitioning, register allocation and scheduling should be done simultaneously!
- ...ever heard about NP?

NP??

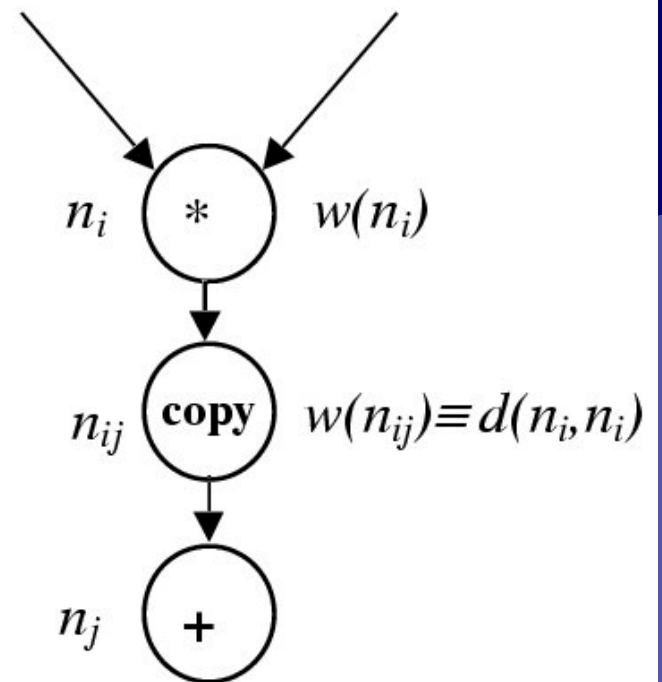
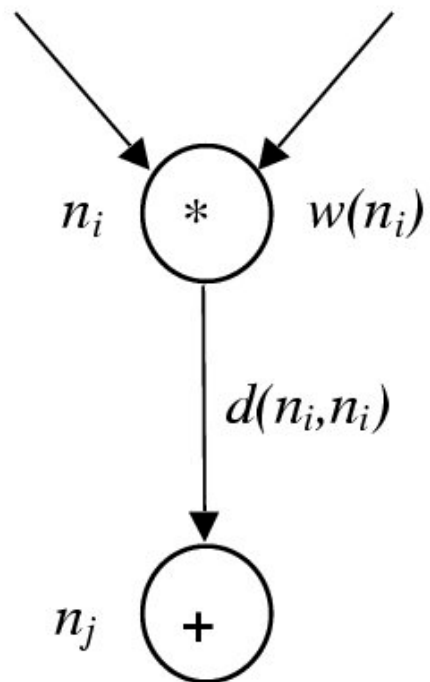
aargh!



# Copies

- Transparent: managed by HW, the processor will stall if operands are not available (lost hidden cycles)
- Non transparent: compiler has to schedule explicit *copy* operations
- Full connectivity/Routing (higher delay)
- Bandwidth (1-2 words/cycle)

# Copies (2)





# *Scheduling with Integer Programming*

April 18, 2001

Scheduling for VLIW

44

# *Scheduling Taxonomy*

- NP-hard ( $\rho$ -approximation)
- Classification:  $\alpha \mid \beta \mid \gamma$ 
  - $\alpha \Rightarrow$  type machine: 1, P, R
  - $\beta \Rightarrow$  constraints: release date, preemption, precedences, due dates
  - $\gamma \Rightarrow$  goal:  $C_{\max}$ ,  $\Sigma C_i$ , lateness
- For VLIW: P/R  $\mid$  prec  $\mid$   $C_{\max}$

# Integer Programming

- Scheduling constraints are mapped to integer linear programming constraints. E.g. decision procedure for  $R|pmtn|C_{max}$ ; then relaxed to linear programming, and rounded.

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & \sum_{i=1}^m x_{ij} = 1 \quad \forall j \\ & \sum_{j=1}^n p_{ij} x_{ij} \leq D \quad \forall i \\ \text{and} & x_{ij} \in \{0, 1\} \quad \forall i, j \end{array}$$

# Integer Programming (2)

- Special case R2||C<sub>max</sub>, can be mapped to a quadratic problem, with a randomized expected 1.2752-approx (hyperplane technique).

$$\begin{array}{ll} \text{minimize} & Z \\ \text{subject to} & Z \geq \sum_j w_j \sum_{i=-1}^0 \left( \frac{1 + v_i v_j}{2} p_{ij} + \right. \\ & \left. + \sum_{k \prec_i j} \frac{v_j v_k + v_i v_j + v_i v_k + 1}{4} p_{ik} \right) \\ \text{and} & v_{-1} \cdot v_0 = -1 \\ \text{and} & v_i \in S_n \quad \forall j \in J \cup \{0, -1\} \end{array}$$

# Optimal Scheduling!

- Optimal Instruction Scheduling Using Integer Programming:  
*K. Wilken et al.*
- $1|\text{prec}|C_{\max}$ , with max 3-cycle latency, single-issue!!
- Integer programming problem
- DAG simplification (partitioning, redundant edge elimination, linearization)
- Compilation time +14%, all the blocks are optimally-scheduled
- Really strong simplification!!!
- Best known result for optimal scheduling so far!



# Approximated Problems

- Scheduling algorithms, *D. Karger et al.*
- Bad luck:
  - polynomial 2-approximation for  $R||C_{\max}$
  - provably no better approx. than  $2/3$  unless  $P=NP$
- there is a poly  $\rho$ -approx for  $P||C_{\max}$ 
  - is it enough for our purpose? NO!

# Semidefinite Programming

- Convex Quadratic and Semidefinite Programming Relaxations in Scheduling, *M. Skutella*

- $R || \sum w_i C_i$

- convex quadratic programming relaxation, 3-approximation: assign jobs to machine with int. programming, relaxed to quadratic prog., then randomized rounding

- $R |r_{ij}| \sum w_i C_i$

- good direction, but not yet enough



# *Scheduling for VLIW*

**HOW WITH MORE  
RANDOMIZATION**

April 18, 2001

Scheduling for VLIW

51

# *Multiflow BUG*

- Estimates, from the bottom, when a specific instruction can be executed -- greedy!
- Functional units are the limit
- Ignore copies and register pressure
- Local cost: delay of scheduling; Global cost: critical path



## *New Ideas!*

HOW WITH MORE  
RANDOMIZATION

- Feedback among partitioner and scheduler
- Iterative descent algorithm
- Heuristics to determine local cost

### Examples:

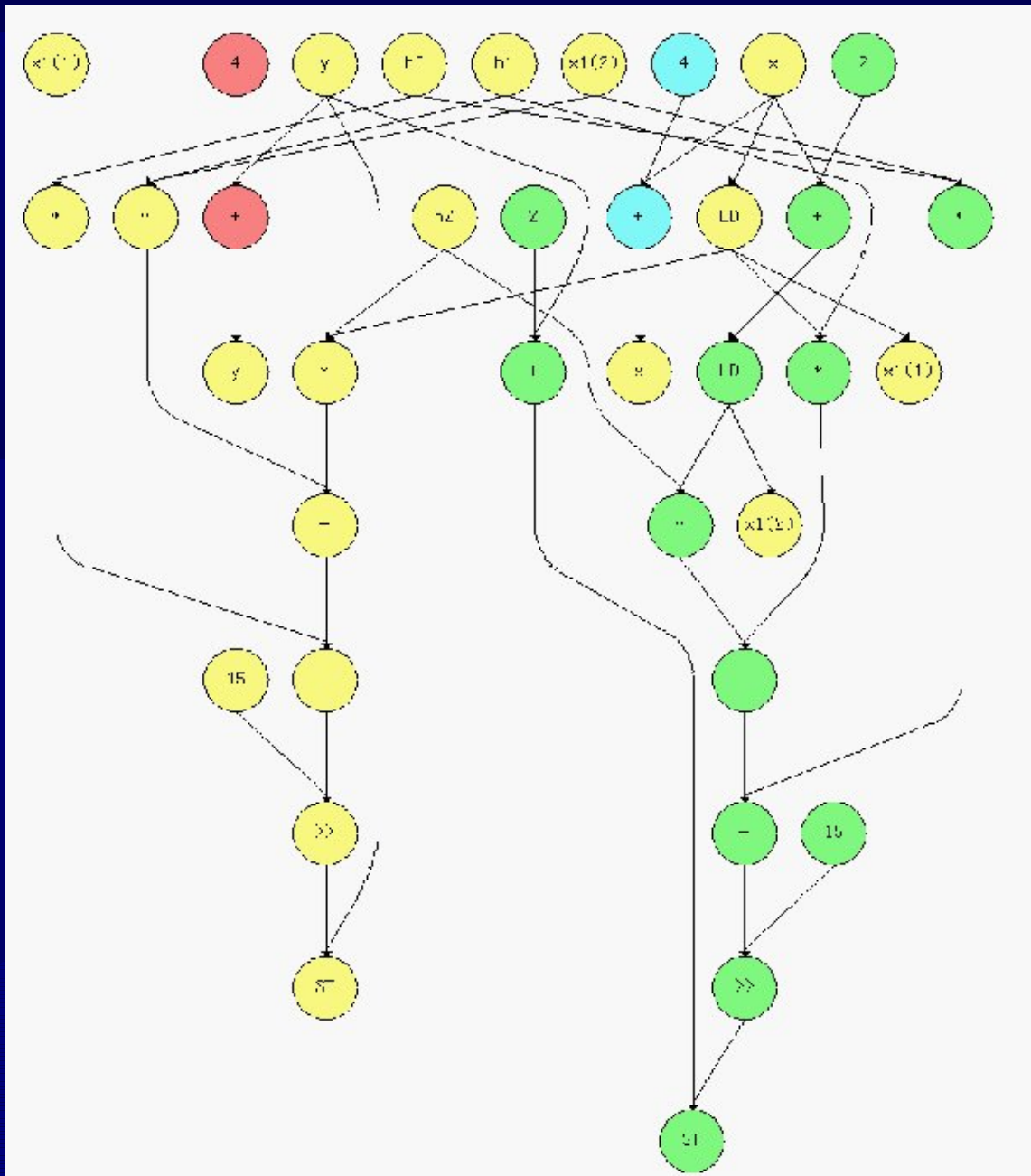
- Simulated annealing
- Unified Assign-Schedule
- Semidefinite programming
- Desoli's

# First Example of Iterative Descent

- Instruction assignment for clustered VLIW DSP compiler: a new approach, *G. Desoli*
- Instruction DAG, enriched with explicit copy nodes (latency $\geq 0$ )
- Determine subcomponents of DAG (if needed, iterate to determine the best size, from  $\infty$ )
- Good size for subcomponents needs to be found iterating, DAG sub-components are determined heuristically: from the bottom, along the critical path (given maximum size and depth)
- Problem: find a k-partition of nodes (sub-components) so to minimize the schedule length  $L$  on the architecture
- Create an initial schedule and partition

# *Partitioning and Clustering*

- Good heuristics for the choice of the initial partition: copy-cost matrix
- Integer programming problem -> simplified to a greedy load balancing problem
- Improve the clustering by moving subcomponents (heuristics: copy-cost matrix)
- Subcomponent to move: for every  $i$ , recompute matrix after moving  $i$ ; for every  $i, j$ , recompute matrix after swap
- Start the descent, with expected schedule length  $L$  as metric
- Heuristic for  $L$ : simple list scheduler, register pressure, resource allocation, number of copies





# *Results*

- Up to 50% faster than BUG,
- No more than 2 times slower compiling

# *People went crazy!*

- Partitioned Schedules for Clustered VLIW Architectures, *M. Fernandes et al.*
- Using queues as register file... memory of dataflow architectures...
- “... *using copy operations does not increase significantly the number of queues required ... do not change the machine configuration required to schedule most of the loops ...*”



*My favourite one!*

April 18, 2001

Scheduling for VLIW

59

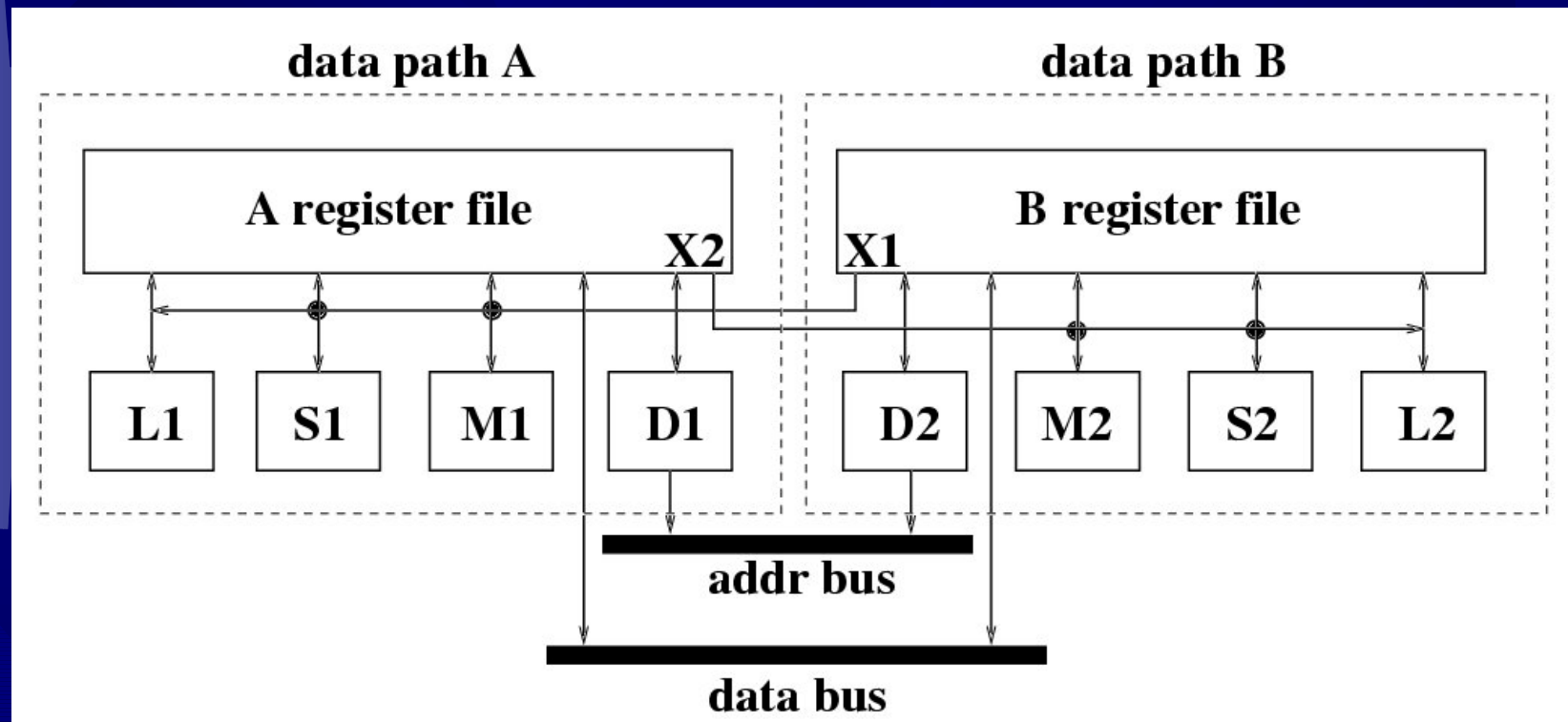
# *Rationale*

- Critical path heuristics (BUG) are good if CPL is close to the optimal schedule
- Scheduling and clustering needs to be coupled
- Iterative search of the optimal solution

# Simulated annealing

- Instruction scheduling for Clustered VLIW DSPs, *R. Leupers*
- Feedback from partitioner to scheduler!
- Simplified: 2 clusters! Partition  $P: V \rightarrow \{A, B\}$
- Schedule:  $F: V \rightarrow \{L, S, M, D\}$ ,  $C: V \rightarrow \mathbf{N}$
- Two interleaved phases: partitioning and scheduling
- Simulated annealing avoid local minima
  - theorem: with logarithmic cooling, global optimum is found

# Model architecture



```
input: DFG  $G$  with  $N$  nodes;  
output:  $P$ : array[1.. $N$ ] of {0, 1} ;  
var  
  int i,r,cost,mincost;  
  float T;  
begin  
  T = 10;  
   $P :=$  RANDOMPARTITIONING();  
  mincost := LISTSCHEDULE( $G,P$ );  
  while T > 0.01 do  
    for i = 1 to 50 do  
      r := RANDOM(1, $n$ );  
       $P[r] := 1 - P[r]$ ;  
      cost := LISTSCHEDULE( $G,P$ );  
      delta := cost - mincost;  
      if delta < 0 or RANDOM(0,1) < exp(-delta/T)  
        then mincost := cost;  
        else  $P[r] := 1 - P[r]$ ;  
        end if  
    end for  
    T = 0.9 * T;  
  end while  
  return  $P$ ;  
end algorithm
```

```

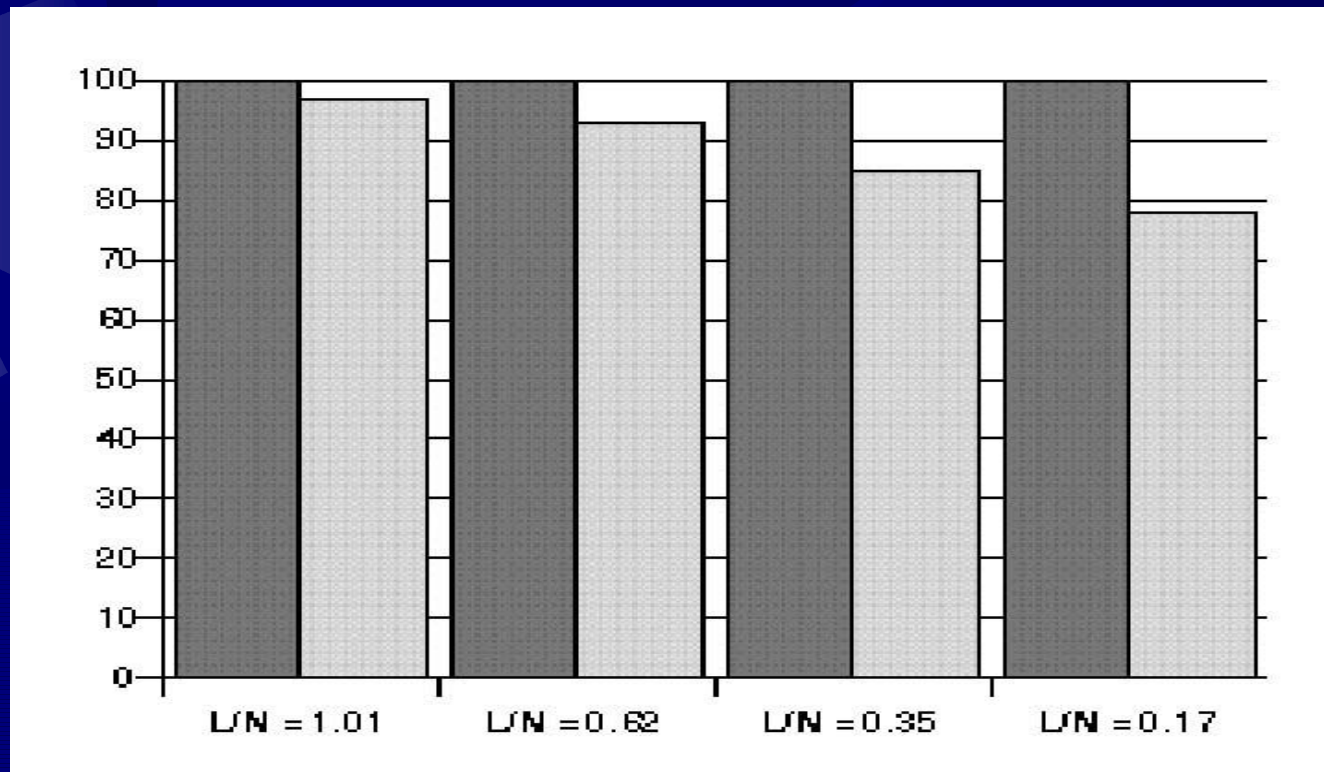
algorithm SCHEDULENODE
input: current schedule  $S$ , node  $m$ , partitioning  $P$ ;
output: updated schedule  $S$  containing node  $m$ ;
var  $cs$ : control step number;
begin
   $cs := \text{EARLIESTCONTROLSTEP}(m) - 1$ ;
  repeat
     $cs := cs + 1$ ;
     $f_m := \text{GETNODEUNIT}(m, cs, P)$ ;
    if  $f_m = \emptyset$  then continue; /* try next  $cs$  */
    if ( $m$  has an argument on a different cluster) then
      CHECKARGTRANSFER();
      if (at least one transfer impossible) then continue;
      else TRYSCHEDULETRANSFERS();
    until ( $m$  has been scheduled);
    if ( $m$  is a LOAD instruction) then
      DETERMINELOADPATH( $m$ );
    end if
    if ( $m$  is a CSE with more than 2 uses) then
      INSERTFORWARDCOPY( $S, m$ );
    end if
    return  $S$ ;
  end algorithm

```



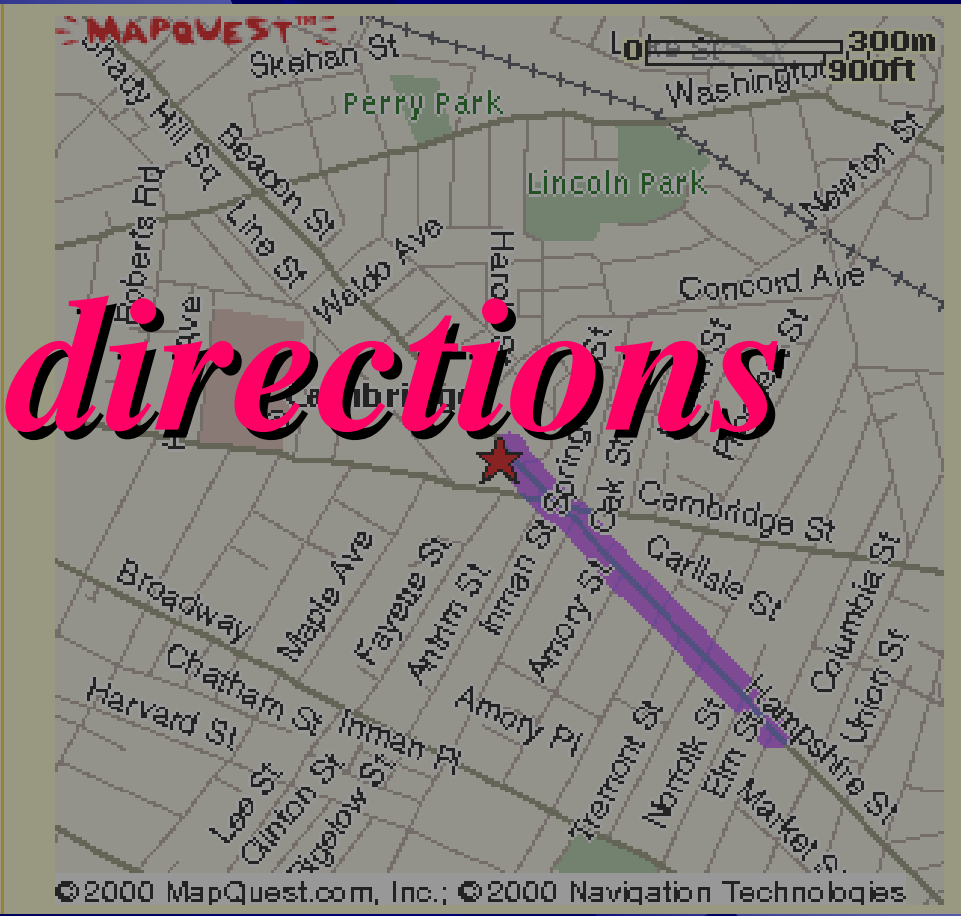
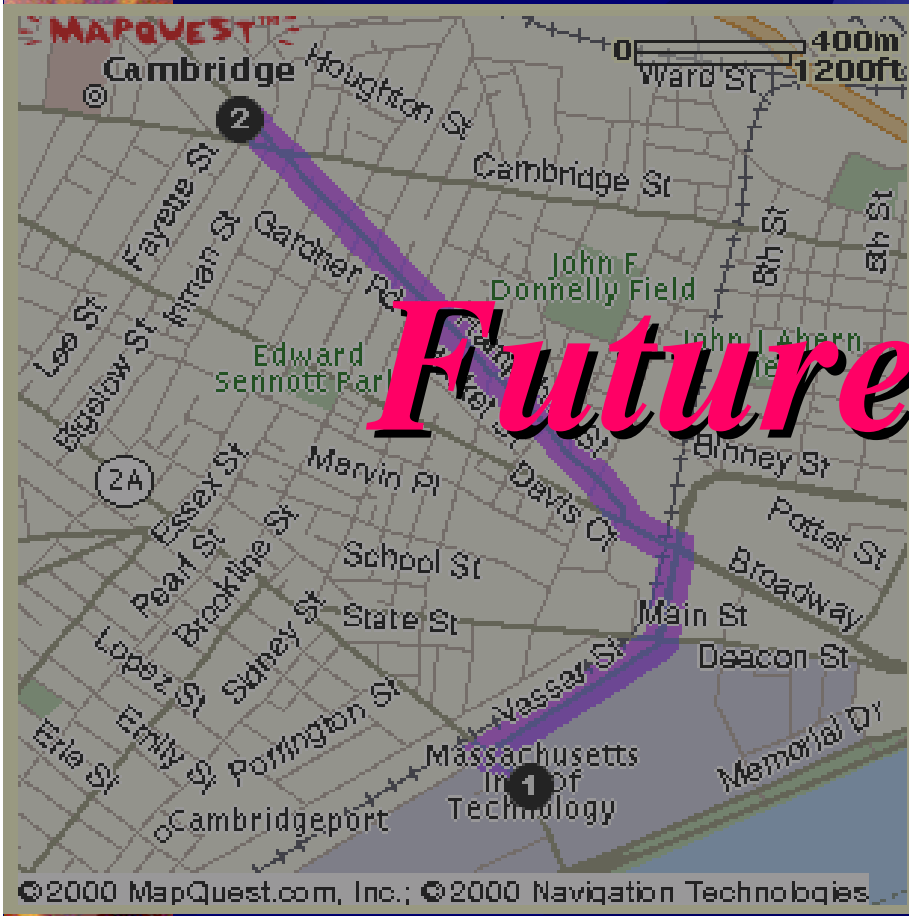
# Results

- Very good when DFG size  $\gg$  CPL

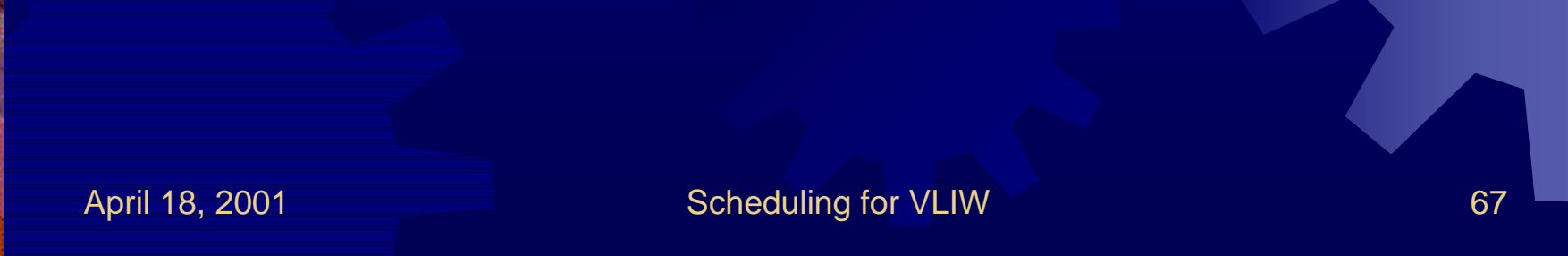


## *Results (2)*

- Code size 10% larger
- 10 seconds for 100-node blocks



# *Future directions*



# *Future directions*

- Simulated annealing looks like the best option
  - Improve the choice heuristics (Desoli's)
  - Can it scale?
- Stronger math. model for approximate solution for integer programming
- Is there any hope for optimal scheduling?

*optimal??*

*aargh!*





*Grazie!*

Prego!!!

Ciao da Riccardino.