Scheduling for VLIW

now with more RANDOMIZATION!

Agenda

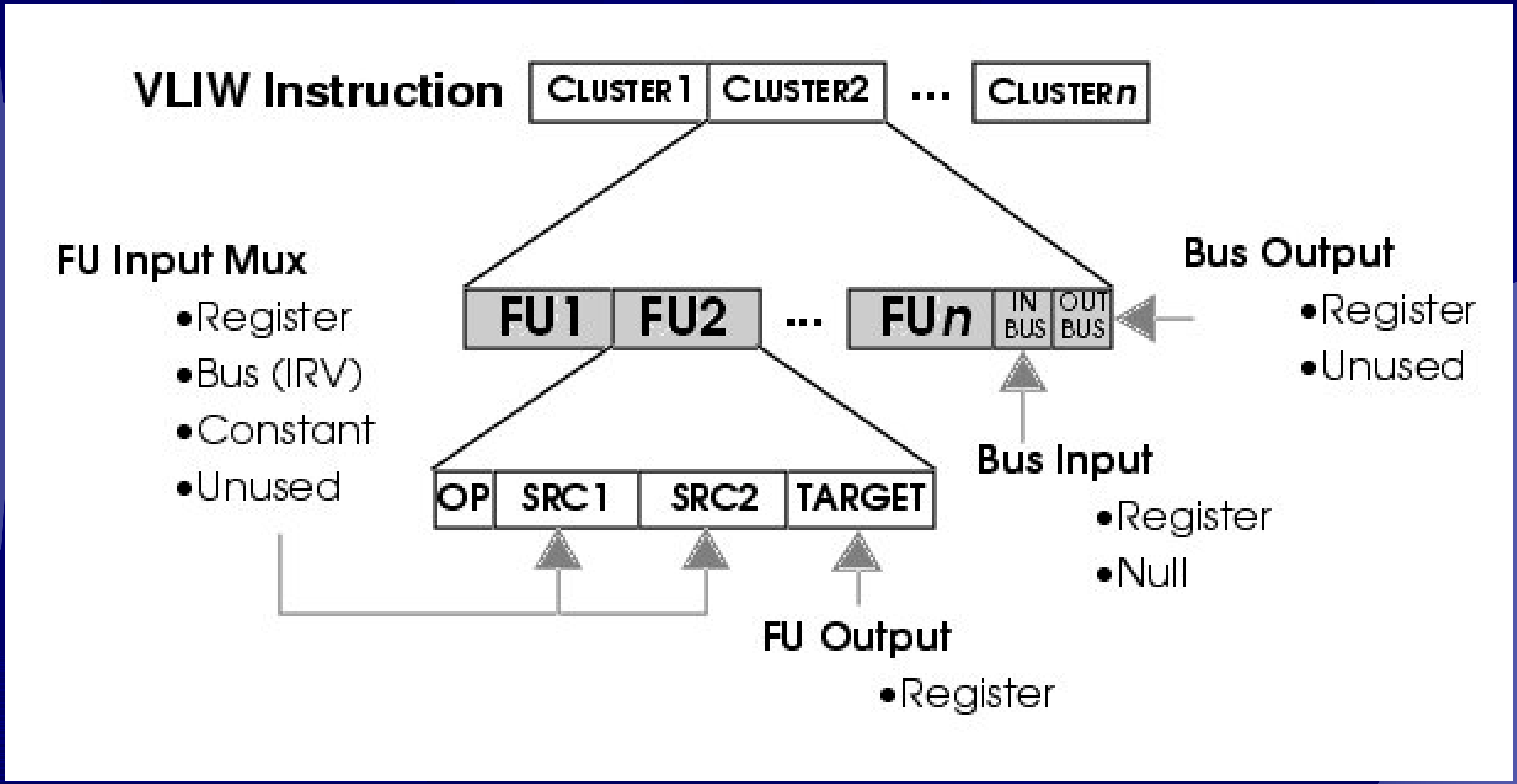
- What’s VLIW? What’s clustered VLIW?
- Scheduling with Integer Programming
- Scheduling for Clustered VLIW
- My favourite one
- Future directions

Clustered VLIW

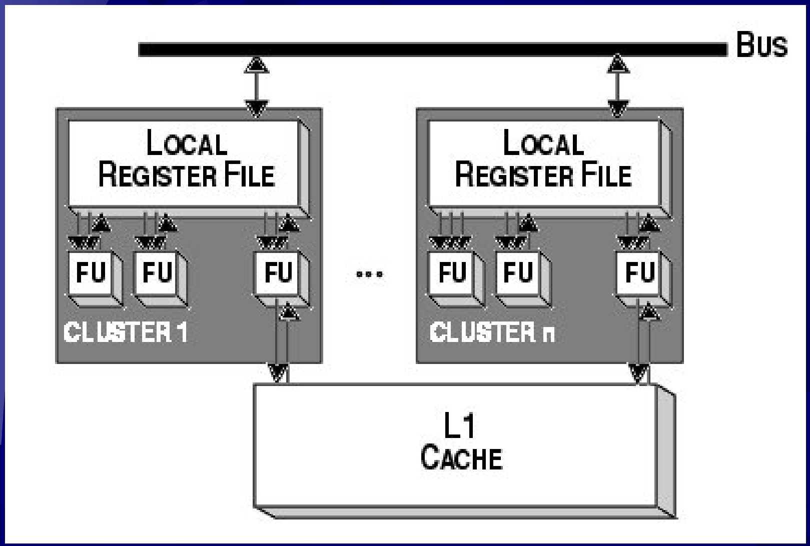
What is VLIW

- 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


VLIW Instruction
CLUSTER1
CLUSTER2
...
CLUSTERn
FU Input Mux
•Register
•Bus (IRV)
•Constant
•Unused
FU1
FU2
...
FUn
IN BUS
OUT BUS
Bus Output
•Register
•Unused
OP
SRC1
SRC2
TARGET
Bus Input
•Register
•Null
FU Output
•Register

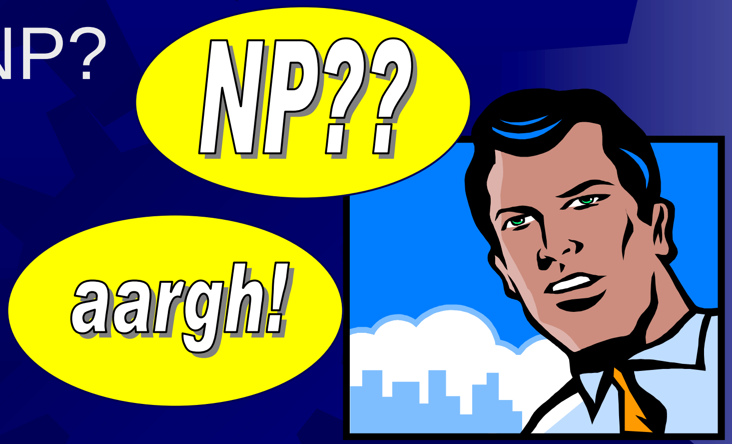
Clustered VLIW



- 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

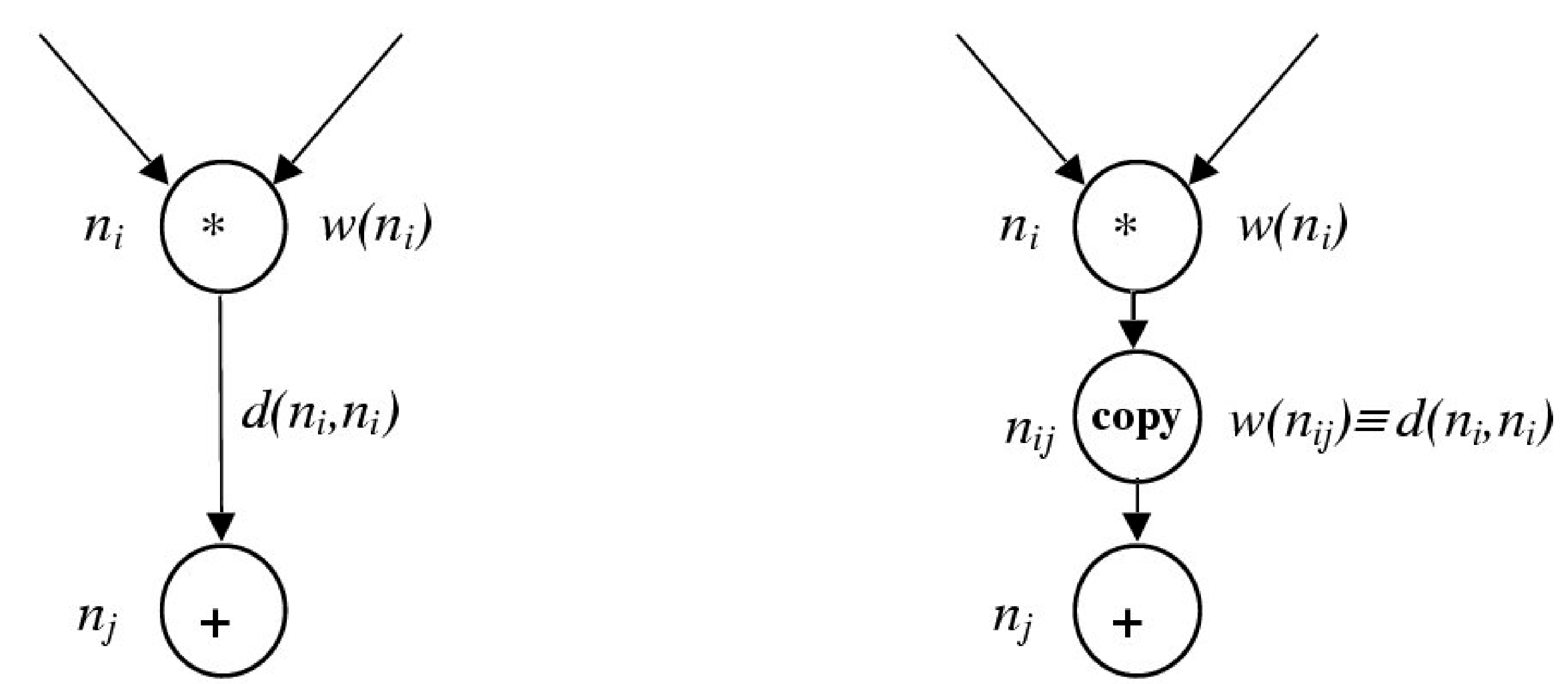
- Partitioning!
- We have to manage copies
- Ideally, instruction selection, partitioning, register allocation and scheduling should be done simultaneously!
- ...ever heard about NP?



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

Scheduling Taxonomy

- NP-hard (ρ-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}, ΣC_i, lateness
- For VLIW: P/R | prec | 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.

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

Integer Programming (2)

- Special case $R2||C_{max}$, can be mapped to a quadratic problem, with a randomized expected 1.2752-approx (hyperplane technique).

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

Optimal Scheduling!

- Optimal Instruction Scheduling Using Integer Programming: *K. Wilken et al.*

- $1|prec|C_{max}$, 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 ρ-approx for $P||C_{max}$
  - is it enough for our purpose? NO!

Semidefinite Programming

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

- $R||\Sigma w_i C_i$
  - convex quadratic programming relaxation, 3-approximation: assign jobs to machine with int. programming, relaxed to quadratic prog., then randomized rounding
- $R|r_{ij}|\Sigma w_i C_i$
- good direction, but not yet enough

Scheduling for VLIW

now with more RANDOMIZATION!

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!

now 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

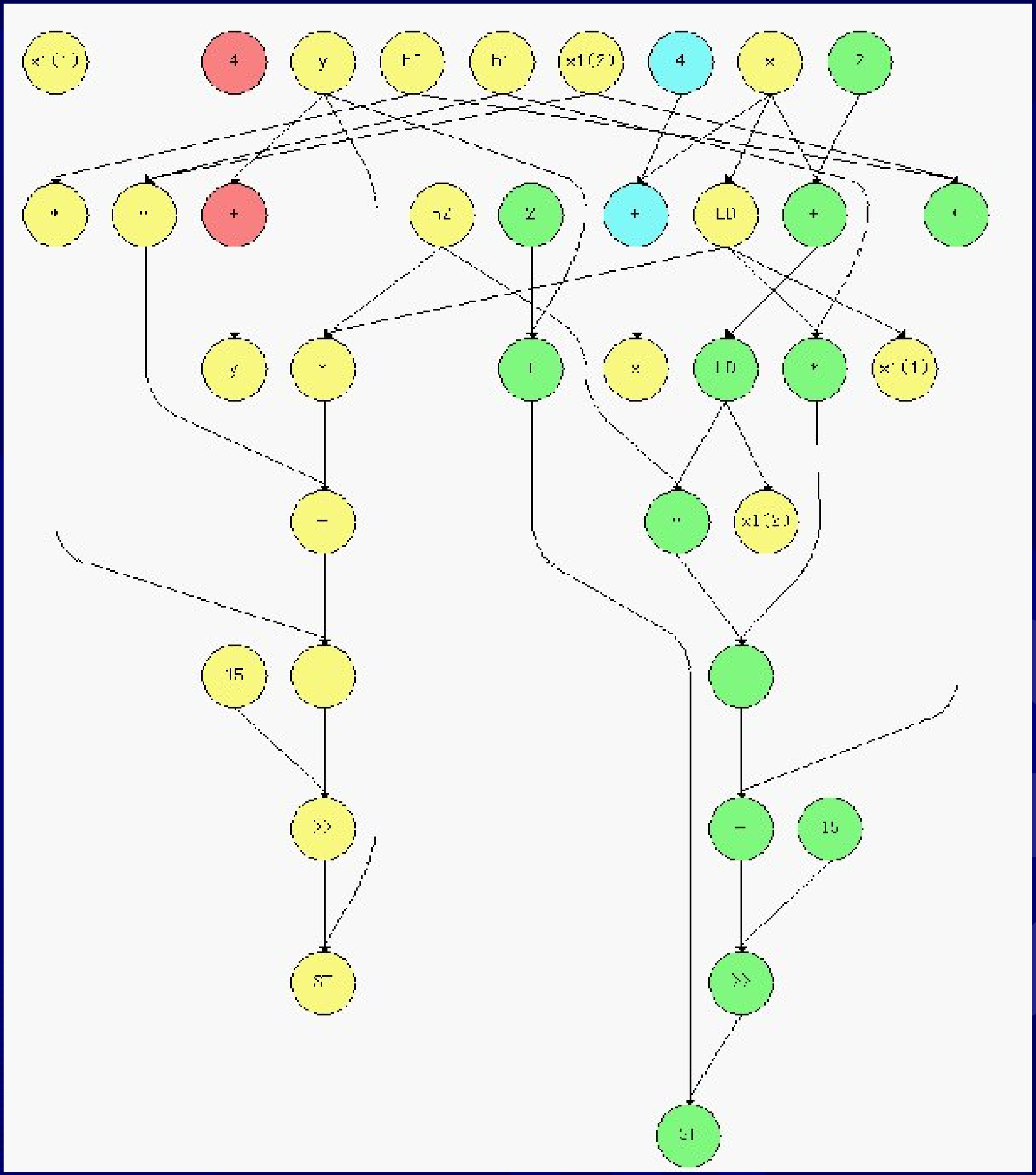
First Example of Iterative Descent

- Instruction assignment for clustered VLIW DSP compiler: a new approach, *G. Desoli*

- Instruction DAG, enriched with explicit copy nodes (latency≥0)
- Determine subcomponents of DAG (if needed, iterate to determine the best size, from ∞)
- 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!

Rationale

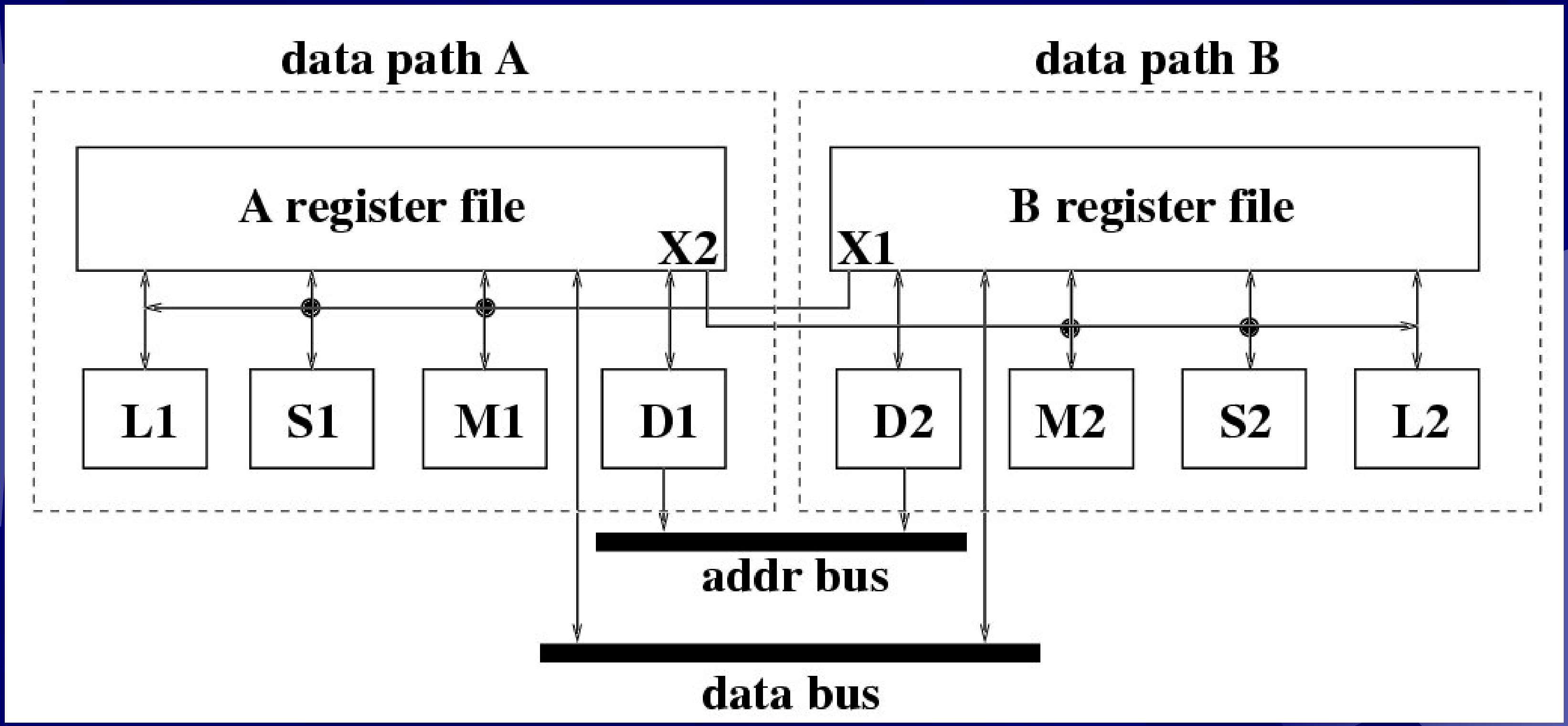
- Critical path heuristics (BUG) are good if CPL is close to the optimal schedule
- Scheduling and clustering needs to be coupled
- Iterative search of the optimal solution

Simulated annealing

- Instruction scheduling for Clustered VLIW DSPs, *R. Leupers*

- Feedback from partitioner to scheduler!
- Simplified: 2 clusters! Partition P: V-> {A,B}
- Schedule: F: V->{L,S,M,D}, C: V-> **N**
- Two interleaved phases: partitioning and scheduling
- Simulated annealing avoid local minima
  - theorem: with logarithmic cooling, global optimum is found

Model architecture

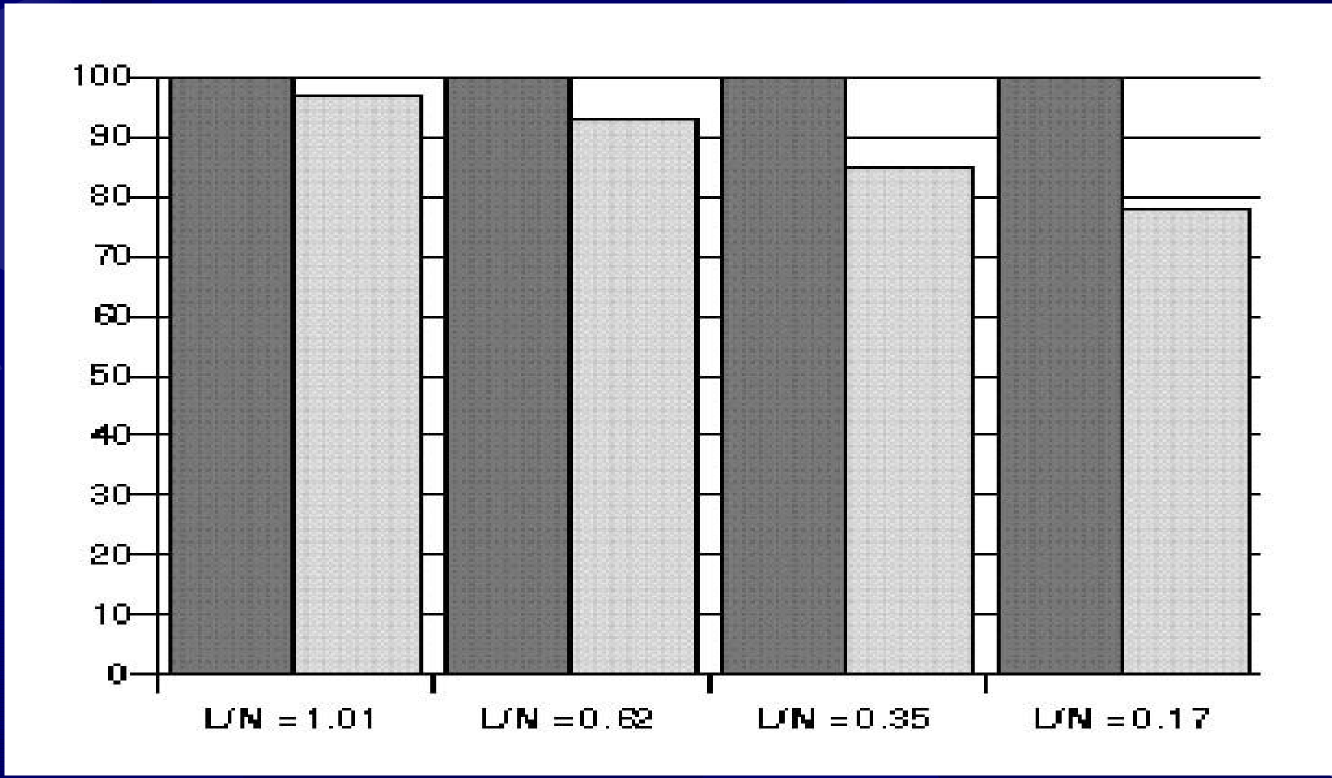


```
input: DFG G with N nodes;
output: P: array[1..N] of {0, 1} ;
var
   int i,r,cost,mincost;
   float T;
begin
   T = 10;
   P := RandomPartitioning();
   mincost := ListSchedule(G,P);
   while T > 0.01 do
      for i = 1 to 50 do
         r := Random(1,n);
         P[r] := 1 - P[r];
         cost := 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 := EARLIESTCONTROLSTEP(m) - 1;
  repeat
    cs := cs + 1;
    f_m := GETNODEUNIT(m, cs, P);
    if f_m = ∅ 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

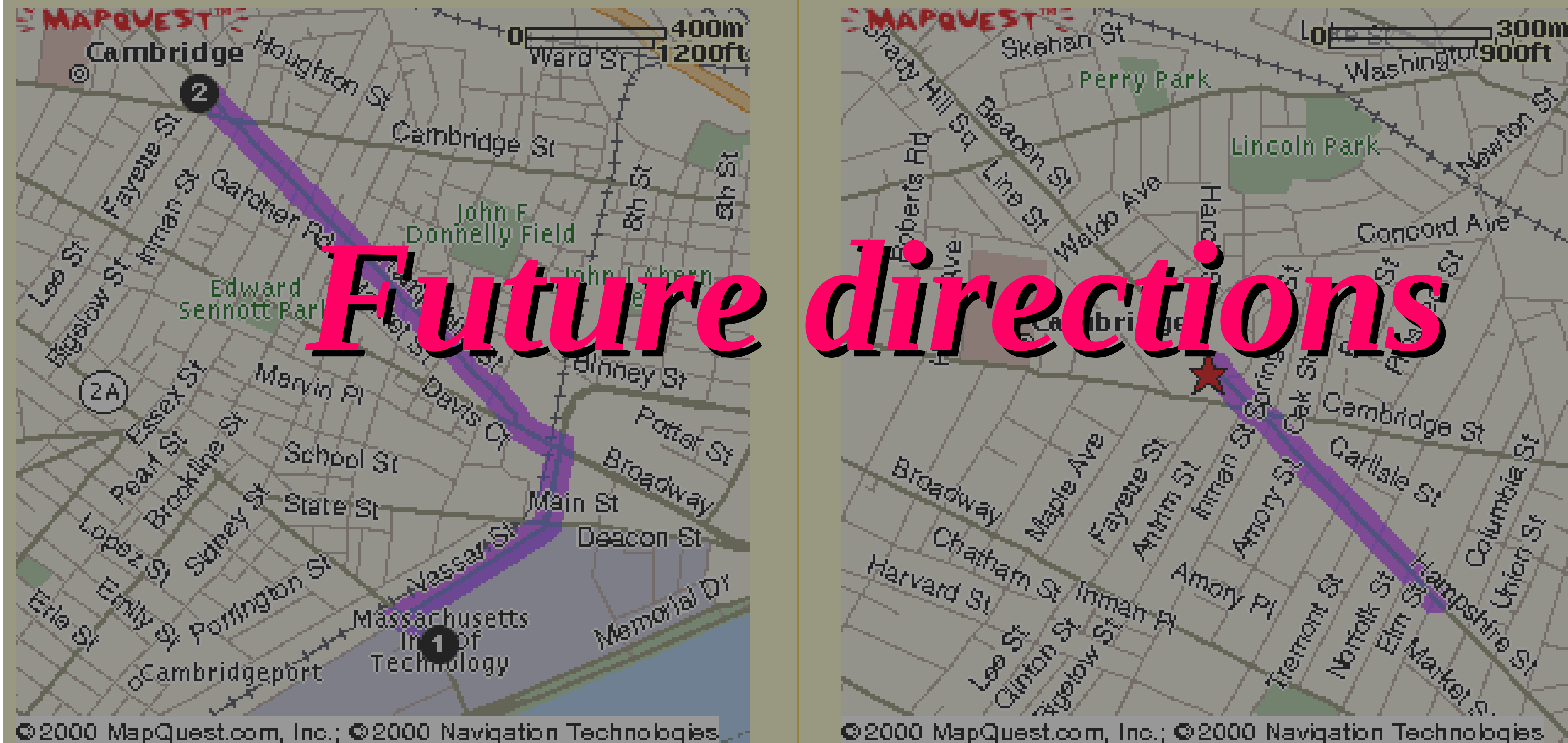
- Very good when DFG size >> CPL



Results (2)

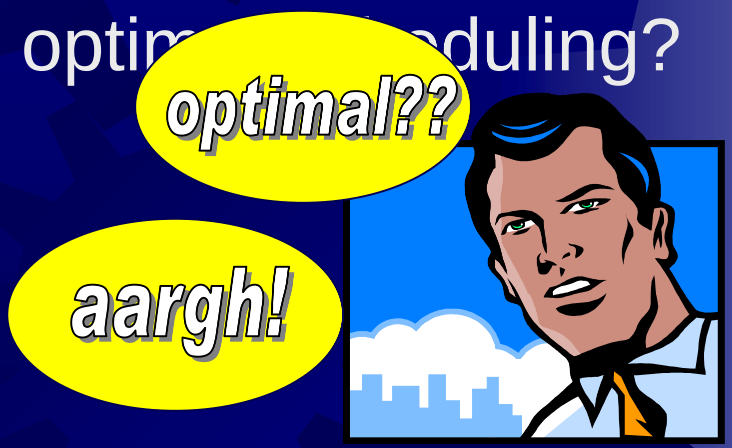
- Code size 10% larger
- 10 seconds for 100-node blocks

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 optim… eduling?



Are you still up?

Thank you!