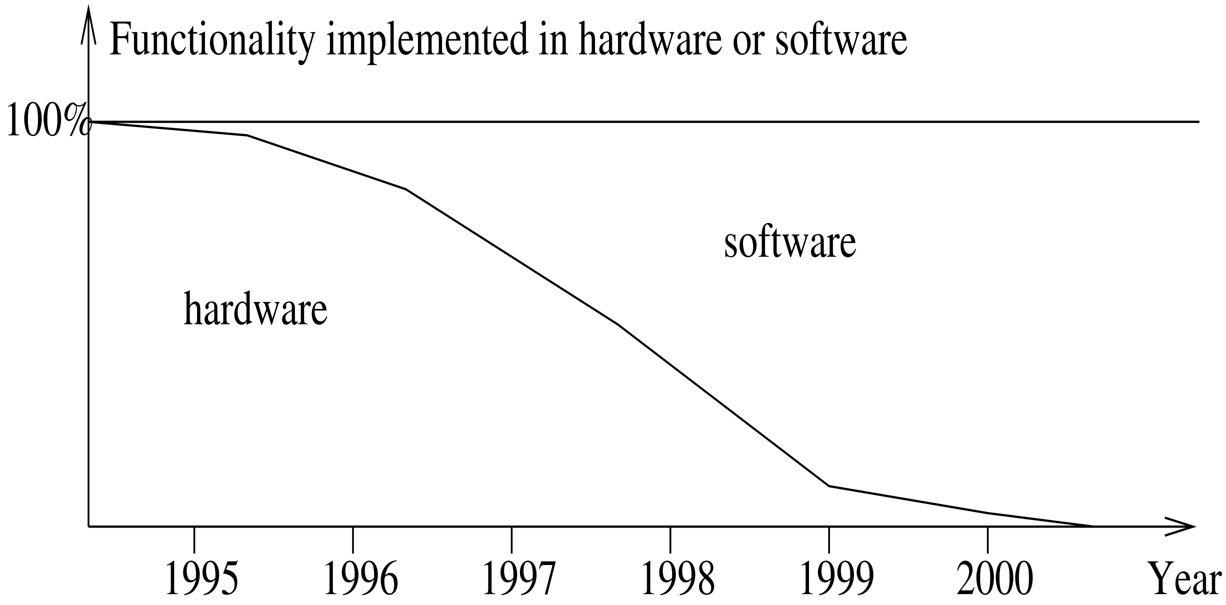
# 4. Hardware/Software Codesign

## 4.1 Purpose

Clear trend towards implementing most of the functionality in software (reason: flexibility):




Basic assumption: fixed functionality.

For actual applications: more and more demanding functionality.

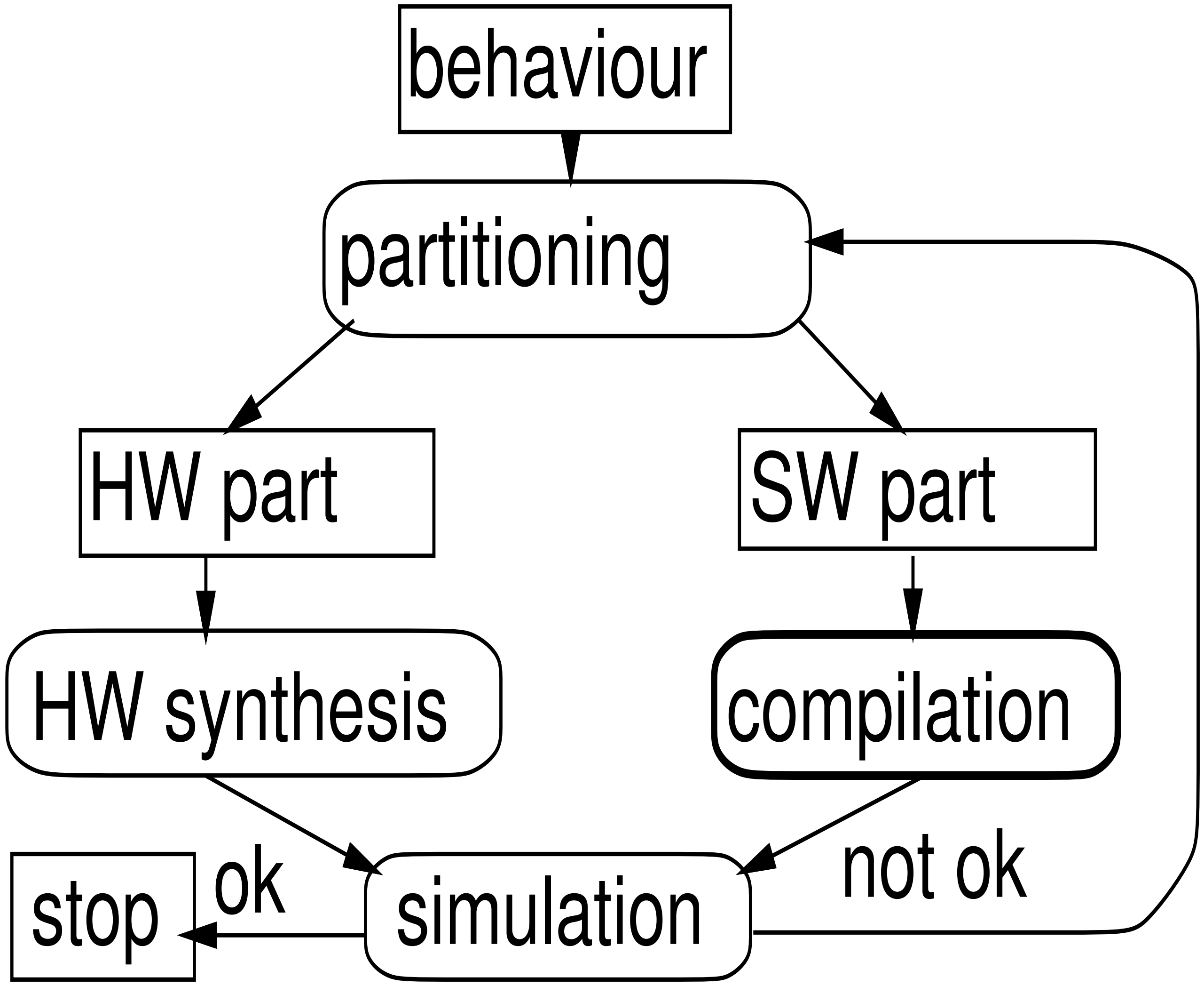
'By the time MPEG-n can be implemented in software, MPEG-n+1 will have been proposed.'

→ We will consider the general case in which both hardware and software components will be required to implement a system.

Which part should be implemented in hardware, which part in software? → HW/SW partitioning.

## 4.2 HW/SW-Partitioning

General picture of HW/SW-Partititing



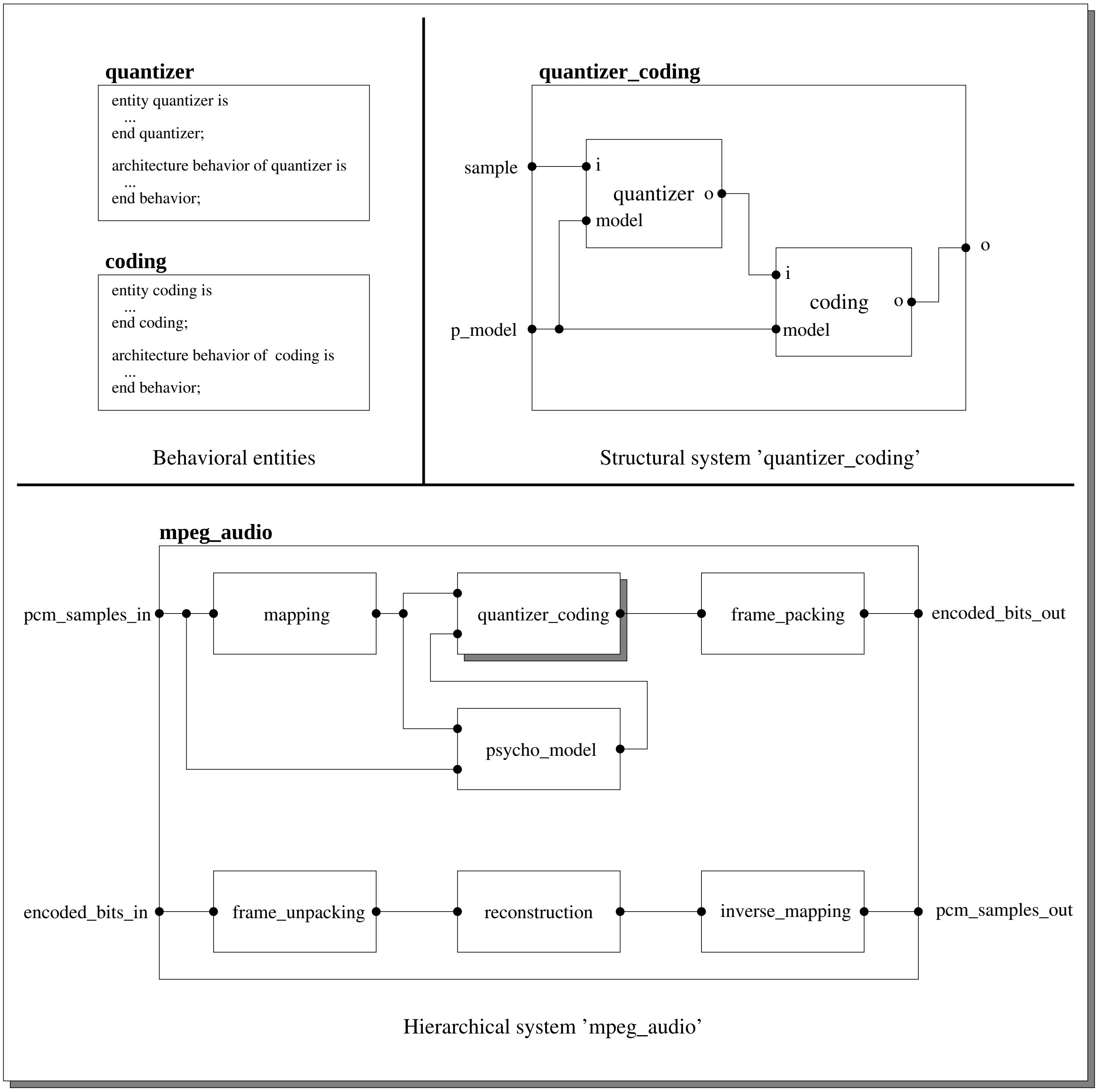

No perfect estimates of cost and performance of hardware or software implementations,
no perfect estimates of communication costs
$\rightarrow$ Iterations are required.

Let's consider an approach to hardware/software partitioning.

# 5.2 **COOL** (Codesign Tool) [Niemann, 1997/1998]

System Specification:

1. **Behaviour**: Connected components described in VHDL. Example: MPEG-Audio:

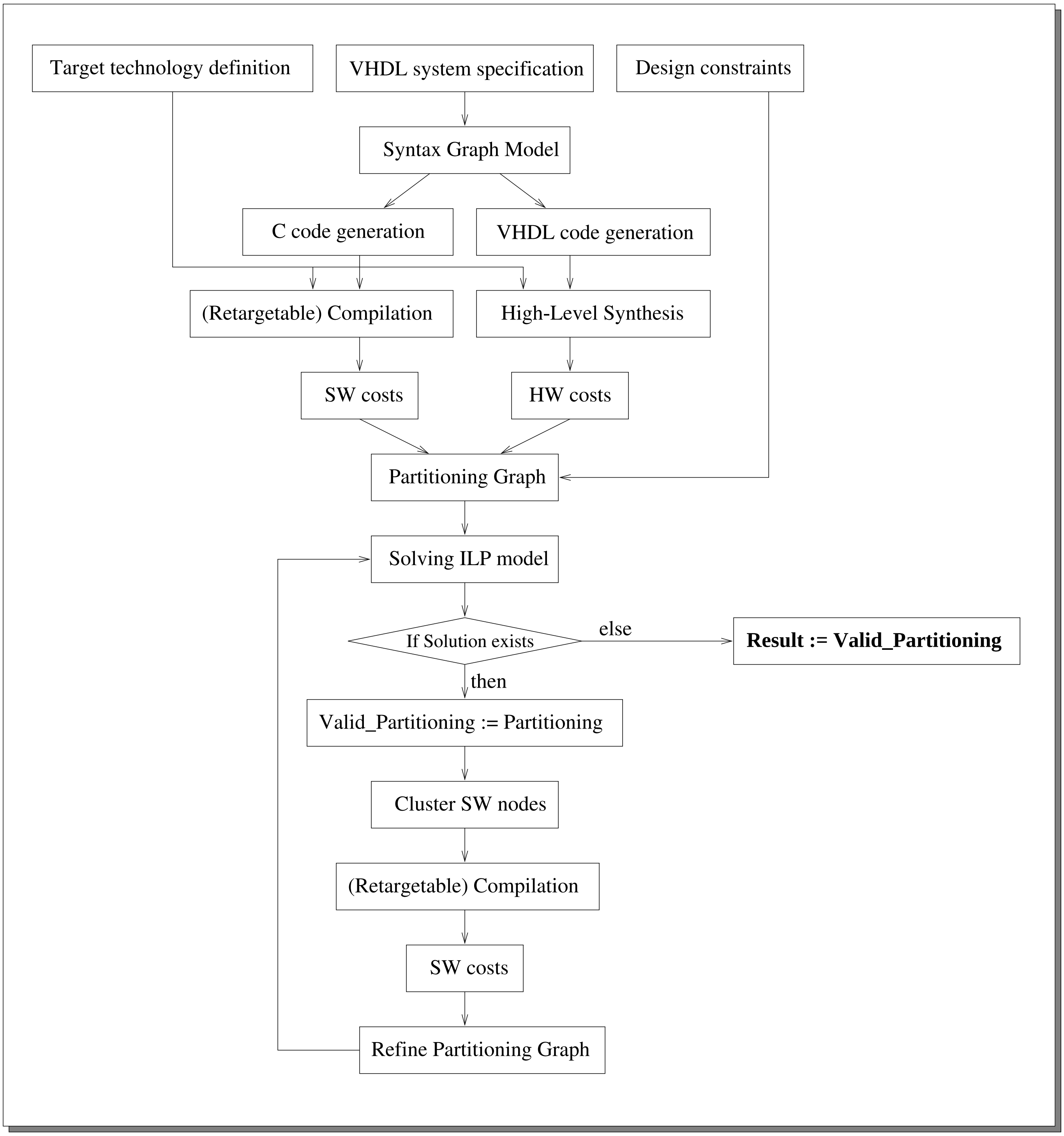



Top: Encoder for Audio-PCM; Output: MPEG audio data stream.

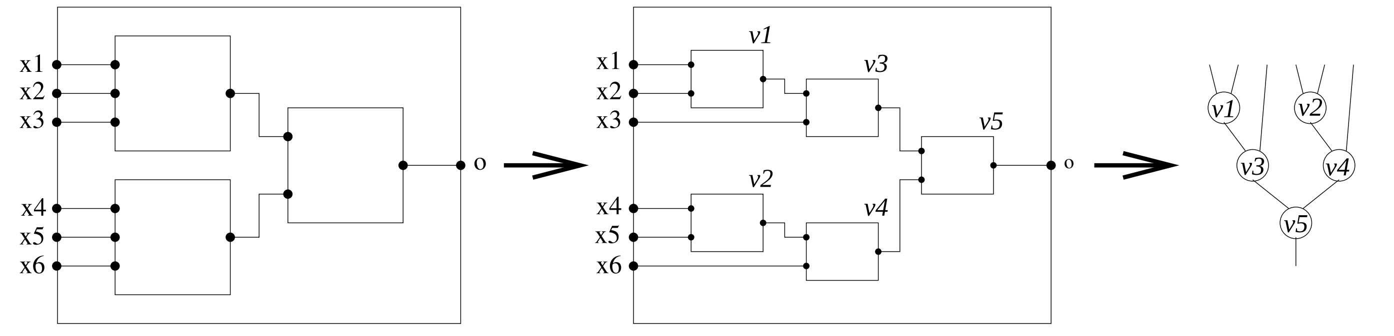
Bottom: Decoder. *Quantizer_coding* described as two components which in turn are described in VHDL.

2. **Target technology**: available processors, library of hardware components
3. **design constraints**: timing, area, memory sizes.

## Hardware/Software Partitioning



1. Translation into internal graph model.
2. Translation of VHDL into C.
3. Compilation of C programs for available target processors. Computation of program size, estimation of run time.
4. Synthesis of components in hardware. Computation of silicon area, estimation of run time.
5. Flattening of the hierarchy.
   Annotation with cost and performance values.




Hierarchical system

Flattened system

Partitioning graph

6. Generation of an integer programming (IP) model reflecting possible implementation options.
7. Solving IP model.

   Optimal with respect to selected cost function.

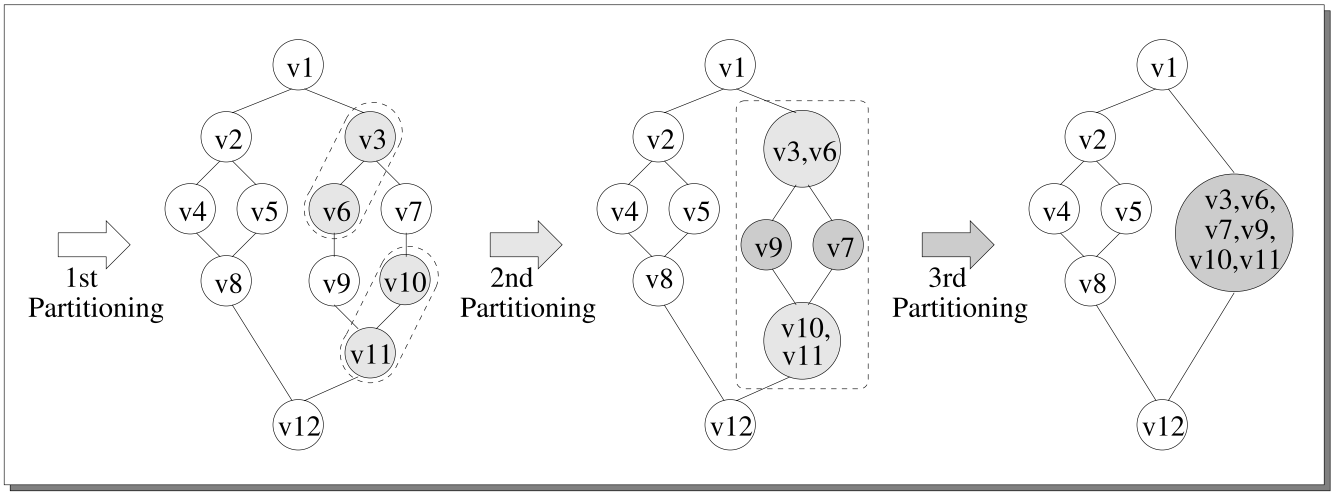
   $\rightarrow$ Iterations required (for example, because communication cannot be precisely estimated before synthesis is completed).

8. Iterative Improvement

Nodes mapped onto the same processor are merged.

New cost and performance values are computed for the new node.

Partitioning will be repeated.




*The first partitioning iteration results in 4 software nodes ($v_3$,$v_6$,$v_{10}$,$v_{11}$). The nodes $v_3$, $v_6$ and $v_{10}$, $v_{11}$ are clustered.*

*After the second iteration it is now possible to execute $v_7$,$v_9$ on the processor, so the new cluster contains $v_3$,$v_6$,$v_7$,$v_9$,$v_{10}$,$v_{11}$.*

*In the third iteration no more nodes can be moved from hardware to software.*

# Some of the equations of the IP-model

## Decision variables

$$x_{i,j,k} = \begin{cases} 1 : & \text{if } v_i \text{ will be mapped onto instance } j \text{ of hw component } k \\ 0 : & \text{otherwise} \end{cases}$$

$$Y_{i,k} = \begin{cases} 1 : & \text{if } v_i \text{ will be mapped onto processor } k \\ 0 : & \text{otherwise} \end{cases}$$

## Constraints:

Every node must be implemented.

No node can implemented more than one node at a time.

Timing constraints must be respected
(this includes communication time).

## Costs

The total cost for processors and hardware has to be minimized.

## Output

*Bindings* between nodes, hardware components and time steps.

## Optimal and heuristic methods

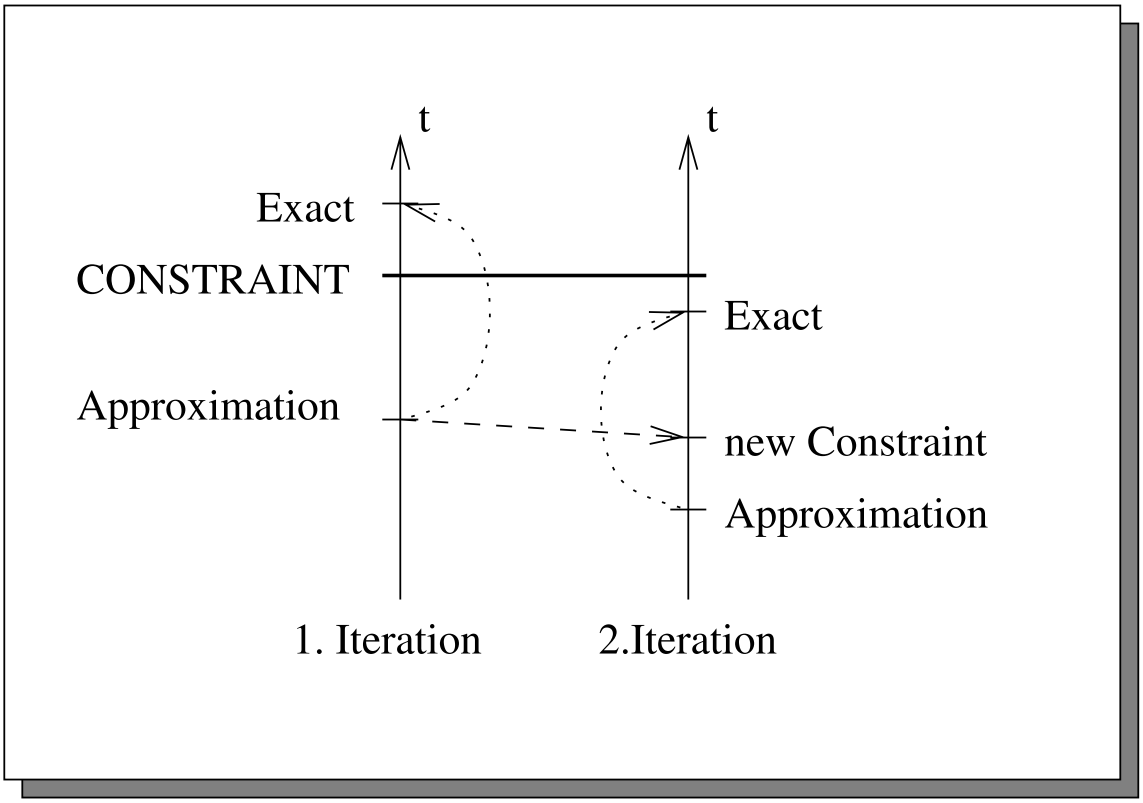
IP model includes the problem of scheduling computations.

Resource constrained scheduling is NP-complete for almost all interesting cases.

→ run times of optimal method grow quickly.

→ heuristic method uses two steps:

1. Partitioning with using estimated schedules
2. optimal scheduling for given partitioning.
3. If timing constraint is violated, then repeat step 1, using a tighter timing constraint.

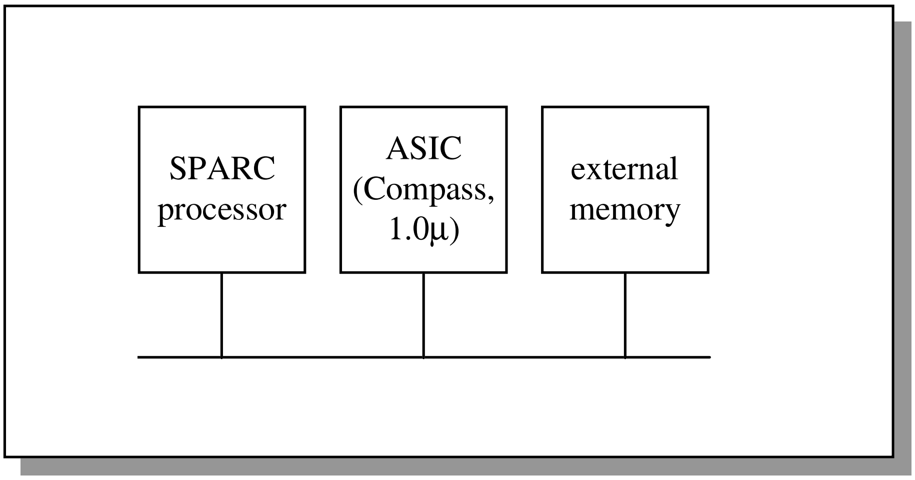


## Results

Examples:

- $n$-Band-Equalizer, $n \leq 7$,
- *audiolab* (mixer, fader, echo, equalizer, balance)
- an MPEG audio encoder (layer II).

Target architecture: SPARC, ASIC, memory, Bus:




Model includes interface costs and sharing

IP-solver: OSL (IBM).

Comparison of results:

- deviation from optimal result
- run time

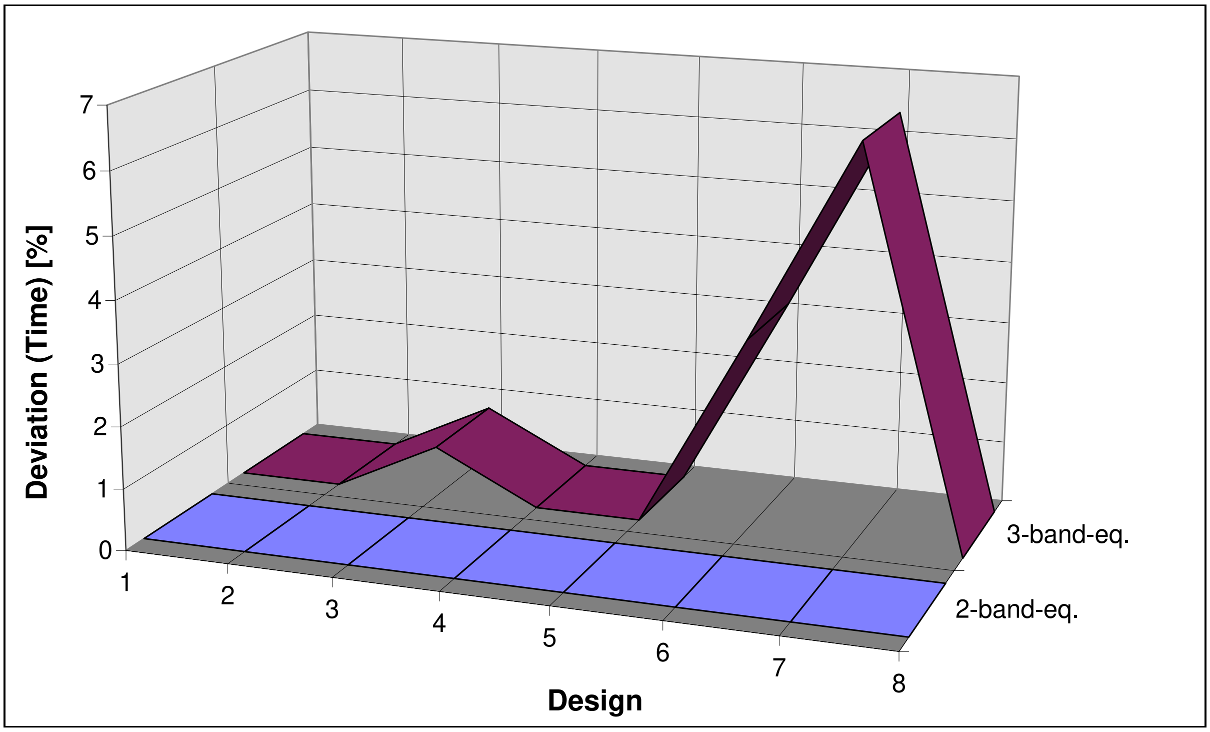
# Optimal vs. heuristic method

- Optimal method: 1. Mapping for minimal area, 2. minimal execution time.
- Heuristic method: 1. Mapping for minimal area, 2. search of legal schedules.

Example: 2- and 3-band equalizer

Different *timing constraints*
(everything in hardware .. everything in software)

Speed of generated designs:

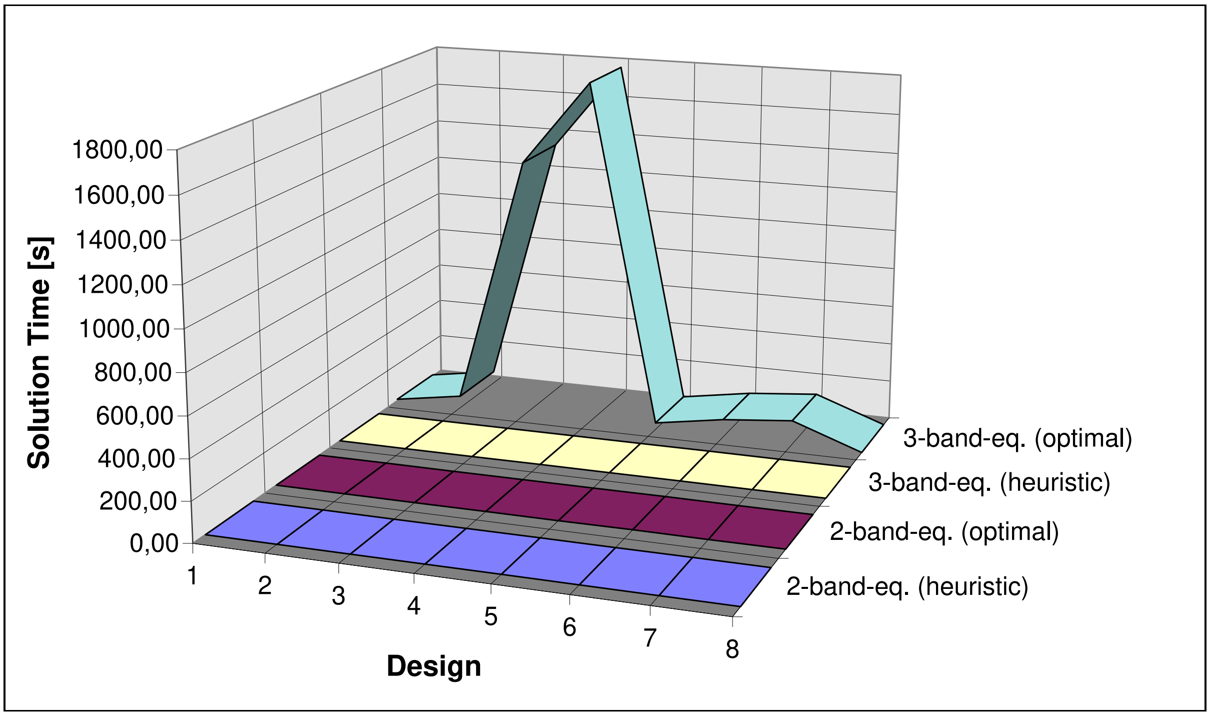



Max. deviation (speed): 6,4%.
Average deviation (speed): 1%.

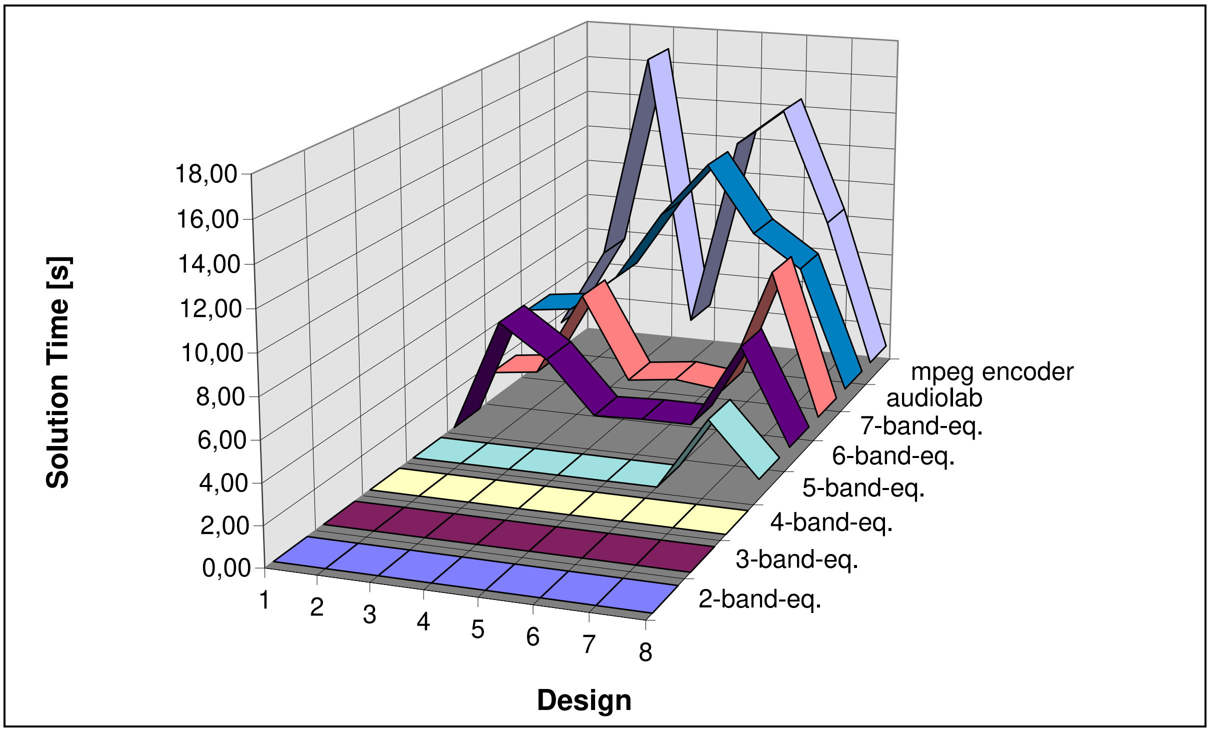
Deviation (area): 0

# Time required for partitioning

Heuristic vs. optimal method:

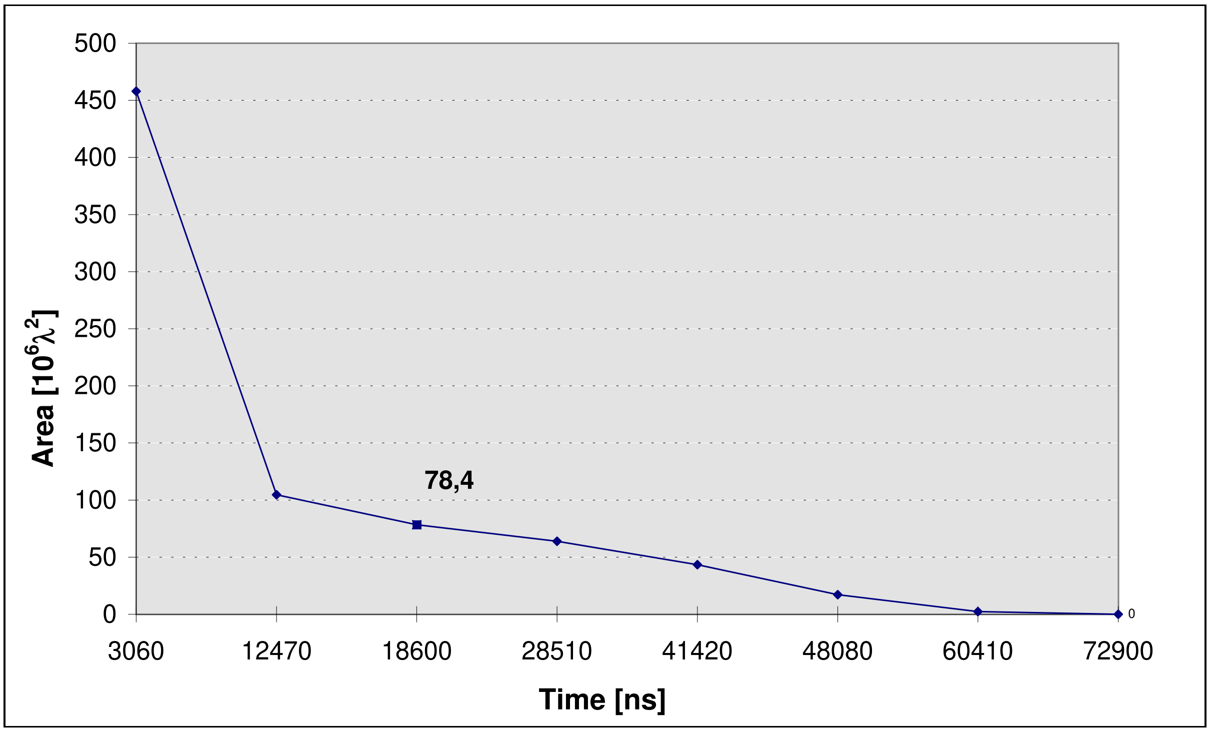



Run times, heuristic method:



## Tradeoff: area vs. speed

Example: *audiolab*, 8 different *timing constraints*




Everything in software: 72900 ns.

Everything in hardware: 3060 ns, $457,9 * 10^6\lambda^2$.

Lowest cost for given sample rate
(44.1 kHz $\sim$ 22675 ns): $78.4 * 10^6\lambda^2$, 18600 ns, .

## Other partitioning strategies:

- Everything mapped to software, move to hardware until timing constraints are met.
- Everything mapped to hardware, move to software as long as timing constraint is met.
- No automation (current practice in industry)

**4.3 Cosimulation** Skipped in this course.