Leiden Embedded Research Center

Prof. Dr. Ed F. Deprettere, Dr. Bart Kienhuis, Dr. Todor Stefanov
Leiden Embedded Research Center (LERC)
Leiden Institute of Advanced Computer Science
Leiden Univerity
{edd, kienhuis, stefanov}@liacs.nl
http://www.liacs.nl/lerc

Main problem

- Embedded is not merely 'small'.

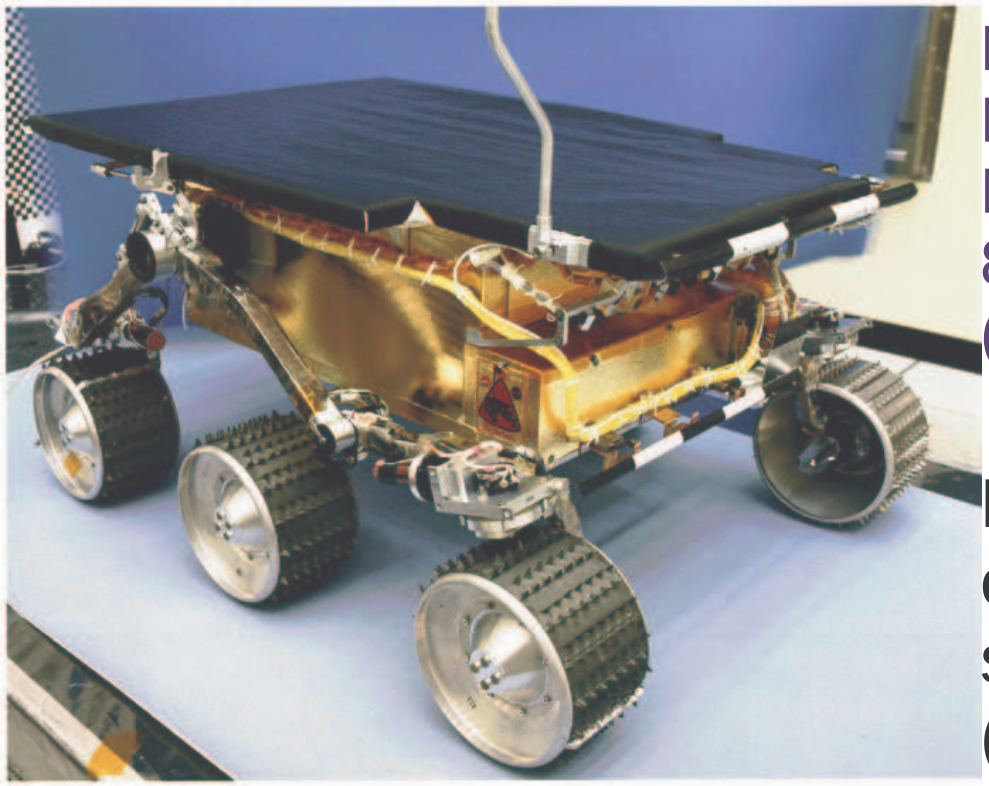
  A small, and a not so small example.

- Embedded means: *time matters*.

  Not 'fast', but determinate and predictable

  (Multi-threaded is neither of the three).

A small Embedded System



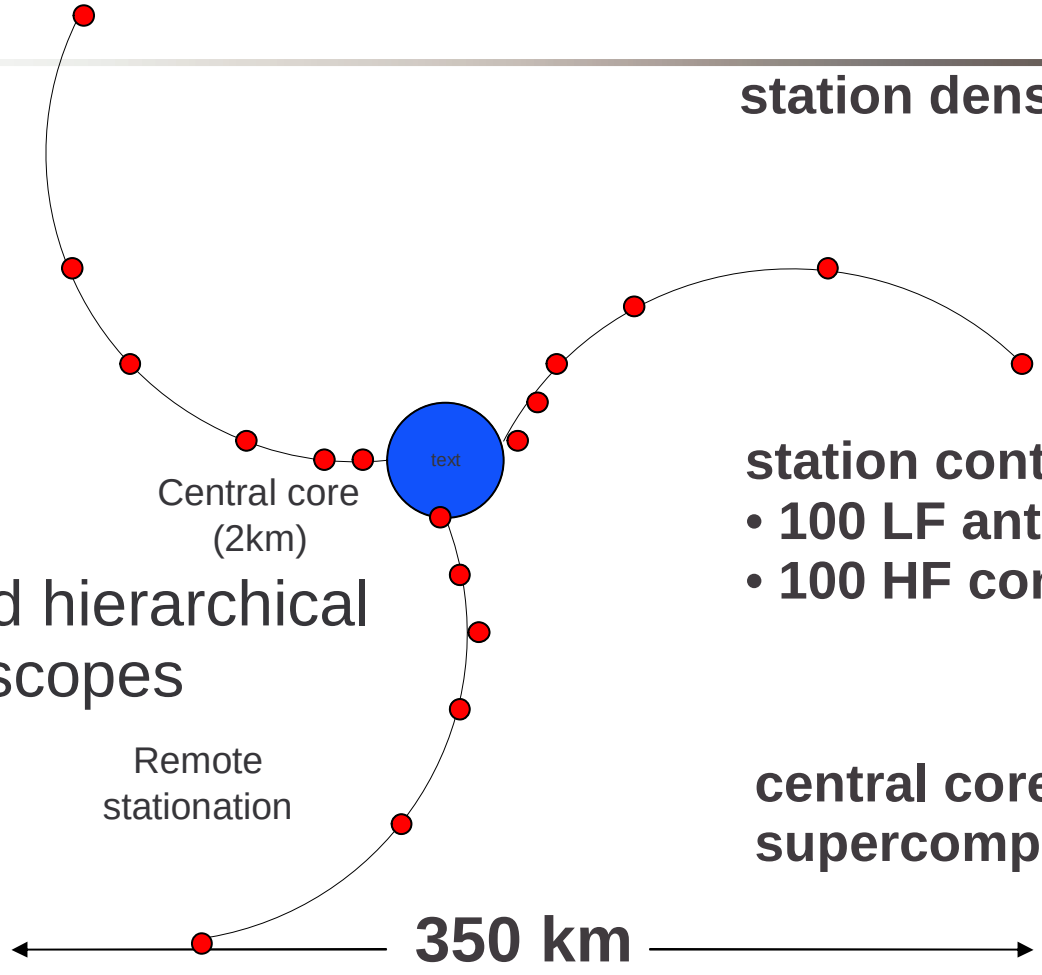
NASA's Mars Sojourner Rover (July 8, 1997)
Microprocessor:
8-bit Intel 80C85
(produced 1977-1990)

Rover *senses* the environment (input signals), *decides on* (computes) its *actions* (response) in *real time*.

iPhone. (Small?)

ARM11 core
Vector Floating Point co-processor (3D-graphics)
SIMD integer CPU (2,1 MIPS)
DMA .45mW/MHz

A not so small Embedded System



station density high in core

- LOFAR
- SKA

Distributed hierarchical radio telescopes

Central core (2km)

Remote stationation

station contains
- **100 LF antennas**
- **100 HF compound antennas**

central core contains supercomputer

350 km

Embedded is What?

An Embedded System is an information processing system that is:

- *application domain specific* (*not* general purpose)
- *tightly coupled* to its *environment*

Examples of *application domains* are: automotive, multimedia.

Environment: type and properties of input/output information.

Tightly coupled: environment *dictates* what the system's *response* behavior must be. (*"ES cannot synchronize with environment"*)

Embedded Systems are *reactive and real-time*

Time Matters.

Computer Science has been concerned about *abstraction*.

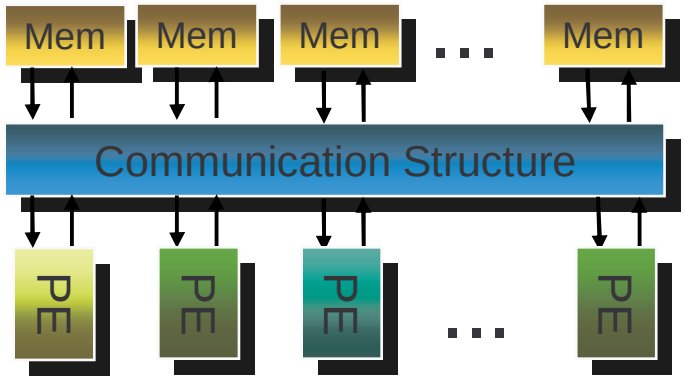
An *instruction set,* a *register file*, and a *program counter*, is all we need to execute whatever application(s) we can dream of.

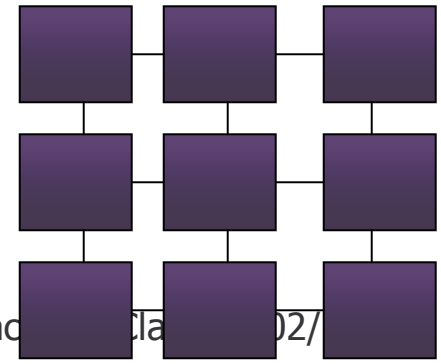
True. But, no programming language (except assembly) considers *timing requirements* and *constraints.*

Modern languages and computers make it impossible to tell how long it will take to execute a piece of code.

Main problem

How to program heterogeneous multiprocessor on chip





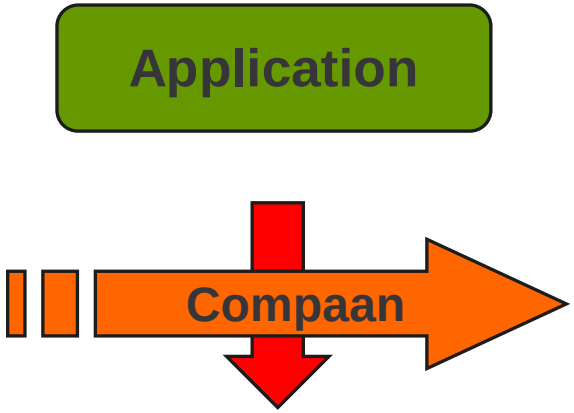
Example 1

EASY to specify

Sequential
Application Specification

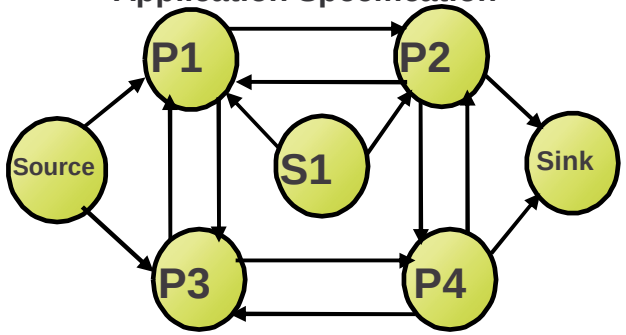
```
for j = 1:1:N,
  [x(j)] = Source1( );
end
for i = 1:1:K,
  [y(i)] = Source2( );
end
for j = 1:1:N,
  for i = 1:1:K,
    [y(i), x(j)] = F( y(i), x(j) );
  end
end
for i = 1:1:K,
  [Out(i)] = Sink( y( I ) );
end
```

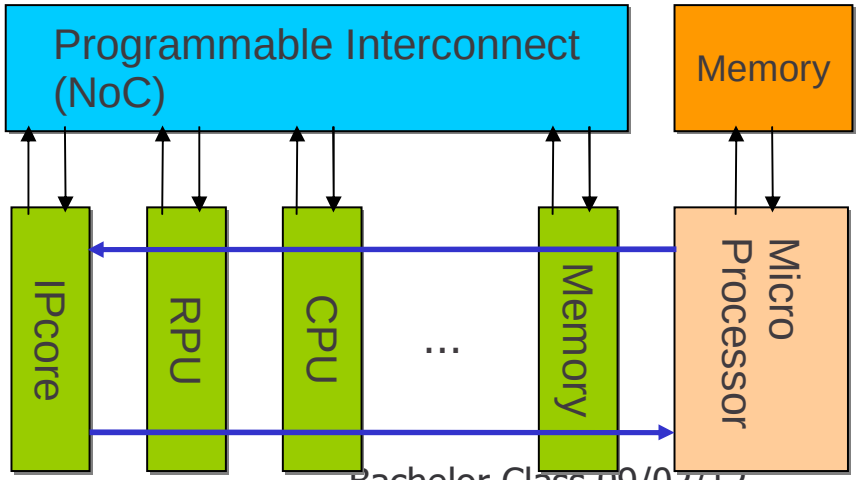
DIFFICULT to map



DIFFICULT to specify

Parallel (Compaan PN)
Application Specification

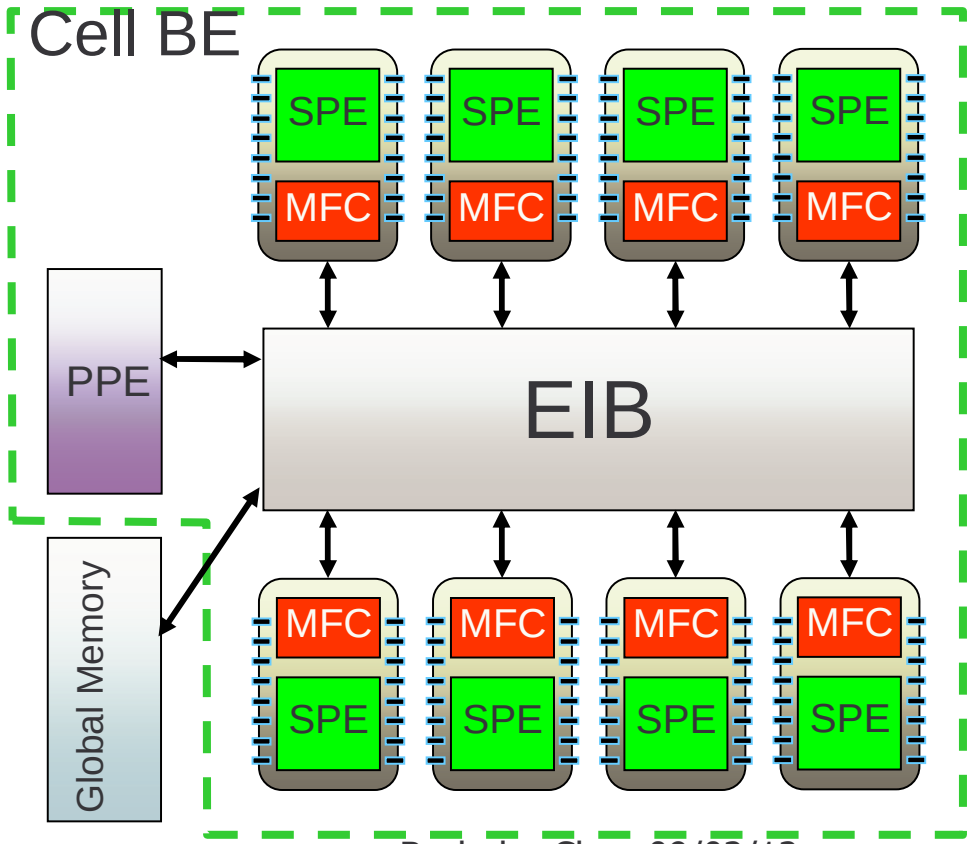




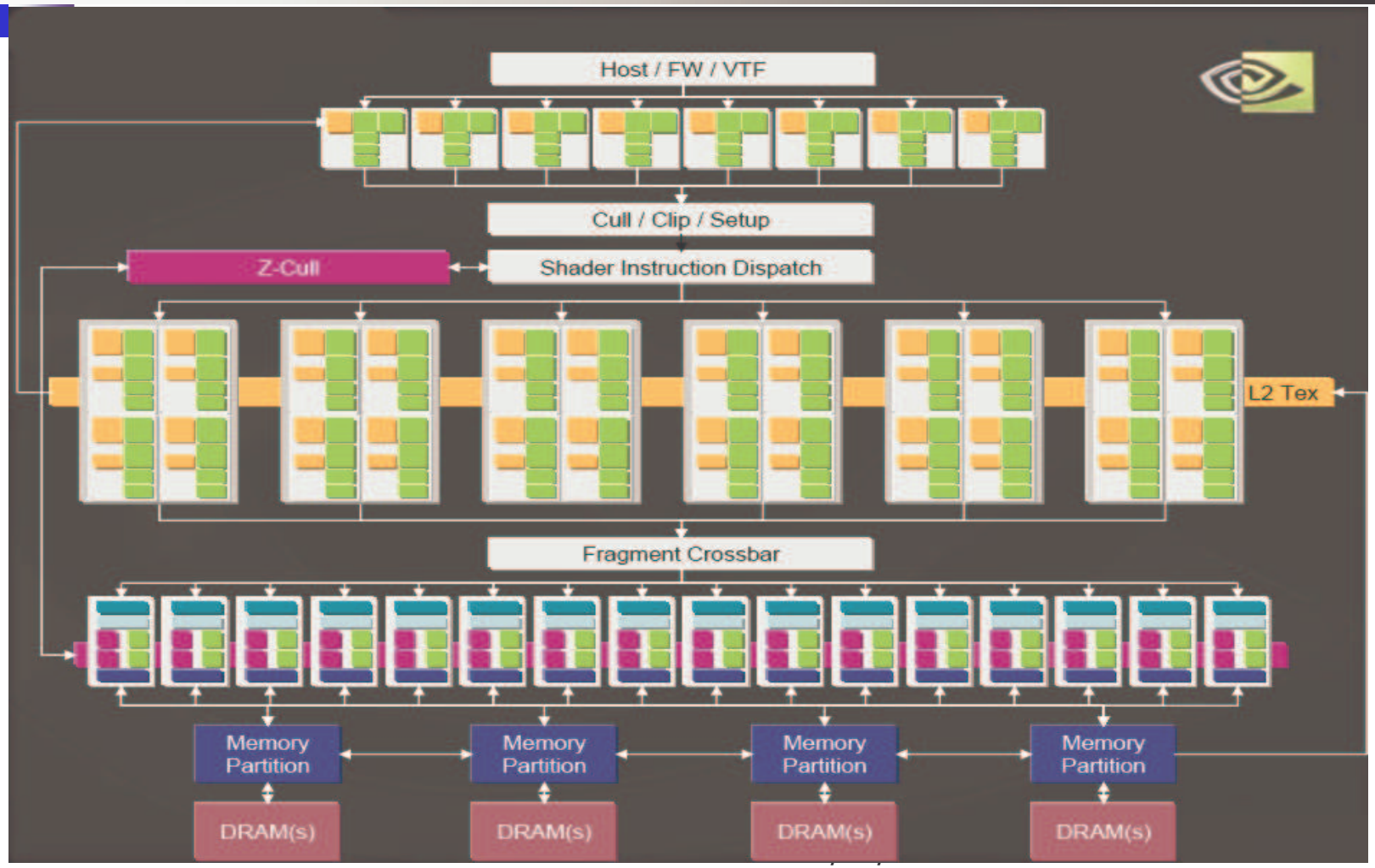
EASY to map

Example 2 Cell

IBM Cell BE (play station 3)



Example 3 nVIDIA GPU



Example 4 Multi-Core

- Desktop computer as framework controlling communication, ensure concurrency
- Dispatch processes to different platforms, container thread remains on host machine
- Correctness checking
- Use input from *pn*, ESPAM
- Goal: integrate