Parallelizing Equation-Based Models for Simulation on Multi-Core Platforms by Utilizing Model Structure

Martin Sjölund Mahder Gebremedhin Peter Fritzson

Programming Environments Laboratory (PELAB)
Department of Computer and Information Science
Linköping University

2013-07-04

Part I

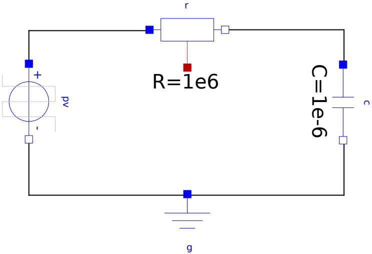
Background

Equation-Based Modelling and Simulation

- Declarative and acausal – describe the problem instead of the solution
- Symbolic manipulations
- Numerical methods and solvers

- An equation-based object-oriented language
- Primarily used to model and simulate (multi-domain) physical systems
- Easiest explained through an example

RC Circuit Example: Different Representations




```
model RC
 import Modelica.Electrical.Analog.Basic;
 import Modelica.Electrical.Analog.Sources;
 Basic.Capacitor c;
 Basic.Resistor r;
 Basic.Ground g;
 Sources.PulseVoltage pv;
equation
 connect(pv.n,g.p);
 connect(r.p,pv.p);
 connect(g.p,c.n);
 connect(r.n,c.p);
end RC;
```

RC Circuit Example: Results

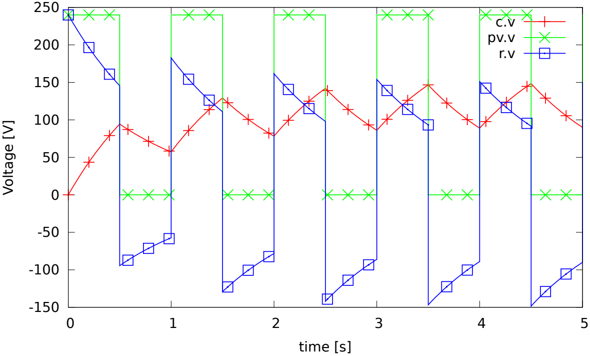


Figure: Simple RC circuit simulation with input square wave pv.v, capacitor voltage c.v, and resistor voltage r.v.

Sorting and Matching

- There exists many valid solution paths
  - $v = r * i$
  - $i = v/r$
  - $r = v/i$
  - ...
- We will not use an integrated approach
  - Normal sorting+matching, then parallelize
  - Parallelize, then sort+match

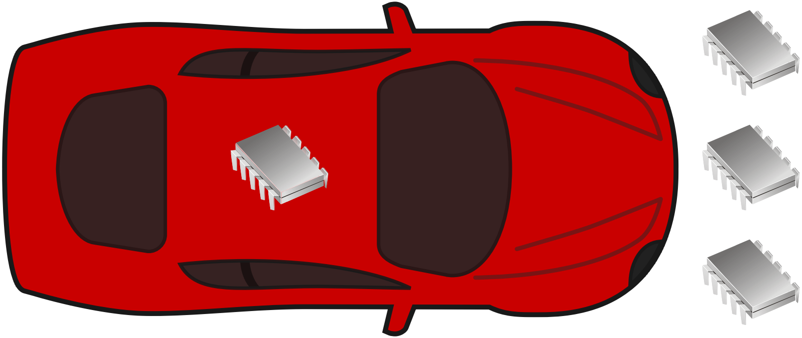
OpenModelica

- Open source Modelica tool
- Open Source Modelica Consortium (OSMC)
  - Many partners
  - Main development at Linköping University
- Used for research prototypes and industrial products

Part II

Parallelisation of Equation-Based Models

Single-Core Solution



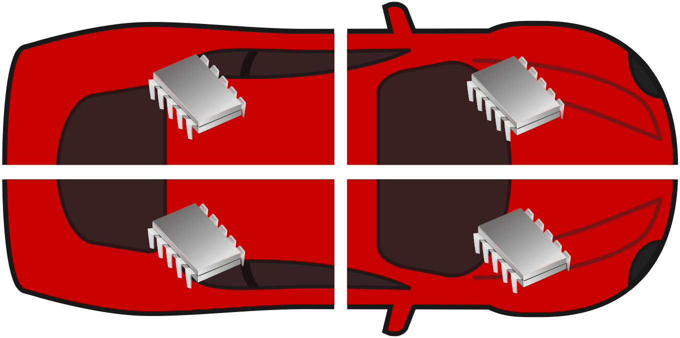
Where Should You Parallelise?

- Expression level
- Equation blocks
- The ODE solver
- The simulation itself

Where Should You Parallelise?

- Expression level
- Equation blocks
- The ODE solver
- The simulation itself

Multi-Core Solution



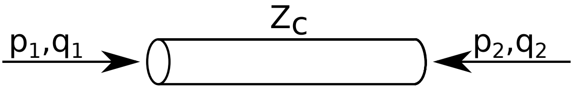


Figure: Transmission line components calculate wave propagation through a line using a physically correct separation in time.

Implementing the delay line: Delay

```
left.c = delay(right.c + 2*Zc*right.q, T);
right.c = delay(left.c + 2*Zc*left.q, T);
```

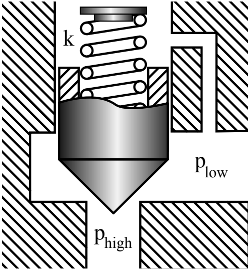


Figure: A pressure relief valve is designed to protect a hydraulic system by opening at a specified maximum pressure.

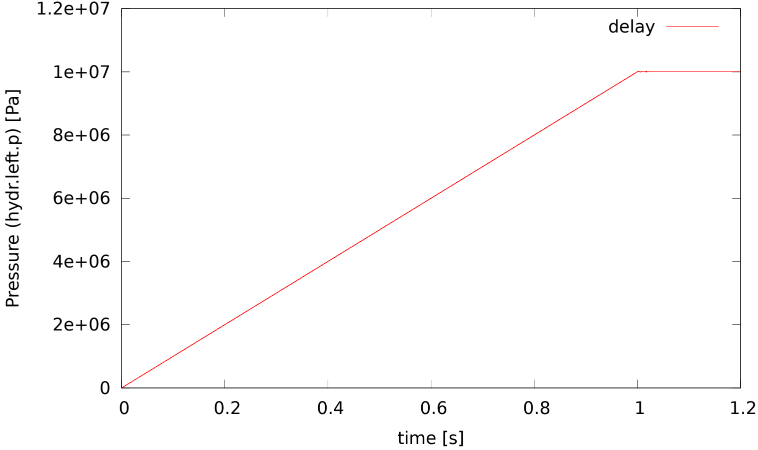


Figure: Pressure increases until the reference pressure of 10 MPa is reached, where the relief valve opens.

Preparing for Parallelization

- Thread pooling in OpenMP
- Thread safety in the run-time system
- Partitioning the system into parallel parts

$$\begin{bmatrix} * & & & & \\ * & * & & & \\ & X & * & & \\ & & * & * & \\ & & * & & * \end{bmatrix}$$

(a) Regular system

$$\begin{bmatrix} * & & & & \\ * & * & & & \\ & & * & & \\ & & * & * & \\ & & * & & * \end{bmatrix}$$

(b) TLM system

$$\begin{bmatrix} * & \\ * & * \end{bmatrix} \begin{bmatrix} * & & \\ * & * & \\ * & & * \end{bmatrix}$$

(c) Partitioned systems

Figure: Adjacency matrices in lower triangular form.

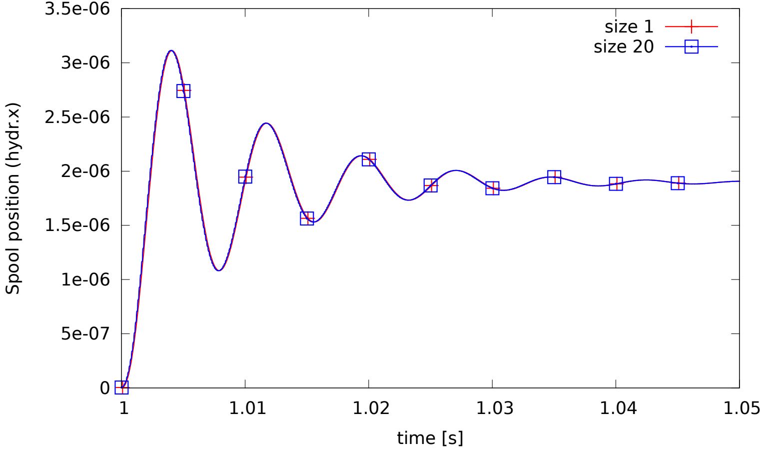


Figure: Comparison of spool position using a volume split into more segments.

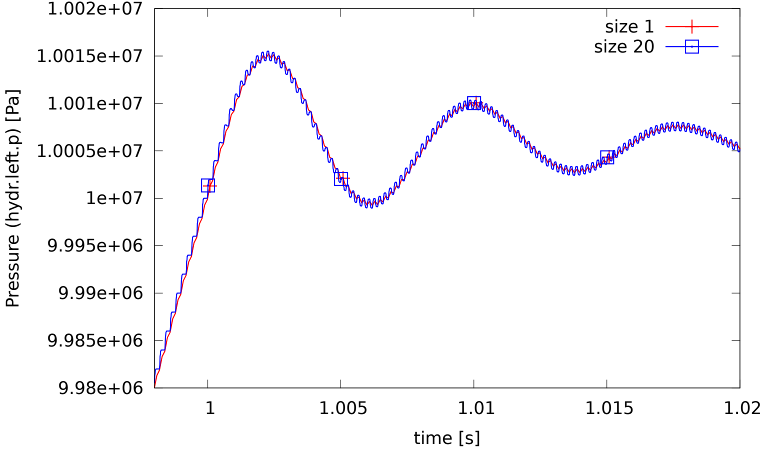


Figure: Comparison of system pressure using a volume split into more segments.

Performance Improvements from Parallel Simulation

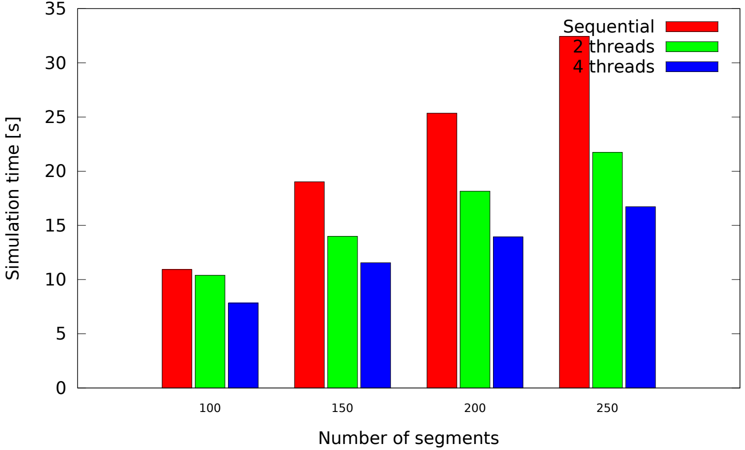


Figure: Simulation time vs. number of segments.

Part III

Conclusion

Summary

- TLM has been used to introduce parallelism in an example model
- Running models in parallel gives performance improvement using OpenMP

Future Work

- Optimise operators (delay) for parallel simulation
- Merge small tasks
- Schedule parallel tasks better
- Add more parallelisation schemes
- Add profiling for parallel simulation (waiting times, etc)

Linköping University

expanding reality