ACTORS

• Adaptyivity and Control of Resources in Embedded Systems
• EU FP7 STREP project
  – 2008-2010
  – Coordinated by Ericsson (Johan Eker)
  – Lund University, TU Kaiserslautern, Scuola Superiore Sant'Anna di Pisa, EPFL, AKAtch, Evidence
• Dataflow programming and adaptive resource management using resource reservation techniques on Linux-based multi-core platforms

ACTORS: Dataflow Modeling

• Dataflow programming with actors (Hewitt, Kahn, etc)
  – Parallelism explicit
  – Strict semantics provides foundation for analysis and model transformation
• CAL Actor Language (UC Berkeley, Xilinx) [http://opendf.org](http://opendf.org)
  – Part of MPEG/RVC
A small example

```text
actor Add(A, B) => Out:
    action A[i], B[j] => Out[i+j] end

actor Delay(A) => Out:
    A: action => Out[v] end
    schedule fun x:
        w1: A(x) => w1;
        w2: w1 => x;
    end
end
```

Real-life examples

```text
network Sum(A) => Out:
    entities
        add = Add();
        delay = Delay();
    structure
        A: action => Out[A];
        add: Out => delay: Out;
        delay: Out => add: Out;
        end
end
```

CAL Transformations

- Several possibilities to transform dataflow programs
  - Split actors to better exploit parallelism
  - Merge actors to reduce communication and synchronization overhead
  - Merge tokens ("vectorization") to reduce firing overhead
  - Split tokens to make data parallelism explicit

This presentation: **Actor merging**

Dynamic CAL Applications

- Most actors have a data- or time-dependent behavior
  - Static analysis and scheduling impossible
    - Run-time best-effort scheduling
  - Some actors have a static behavior
    - The corresponding sub-network can be analyzed and scheduled
Actor Merging

- **Sub-problems**
  - Actor classification (Carl von Platen)
  - Finding statically schedulable sub-networks and generate the schedule (Karl-Erik Årzén)
  - Merge the actors in the sub-networks (Anders Nilsson)
- User interaction required → GUI (Karl-Erik Årzén)

Classification

- **Abstract interpretation**
  - Abstract interpretation on the XLIM level
  - Classifies actors as
    - Static
      - SDF, CSDF
      - port signatures
      - which action to fire
    - Dynamic
      - Data-dependent behaviour
      - Timing dependent

Generating Schedules

- **Subproblems:**
  - Find the repetition vector for the network
    - The number of firings of each actor
    - ILP problem
  - Find an admissible schedule that that respects the repetition vector and optimizes some suitable criterion:
    - Minimize total buffer requirements
    - Minimize the largest buffer required
    - NP-complete nonlinear integer problem

Finding Repetition Vector: SDF

- Incidence matrix:
  \[
  G = \begin{bmatrix}
  2 & -3 & 0 \\
  0 & 1 & -2 \\
  \end{bmatrix}
  \]
- Repetition vector \( q \) given by smallest integer solution to \( Gq = 0 \)
- Solution:
  \[
  q = \begin{bmatrix}
  3 \\
  2 \\
  1
  \end{bmatrix}
  \]

Finding a Schedule: SDF

- Lee & Messerschmitt’s method for finding a PASS (Periodic Admissible Sequential Schedule)
- **Algorithm:**
  1. Find the repetition vector
  2. Form a list of all actors in the network
  3. For each actor in the list, schedule the actor if it is fireable, trying each node once (respect the balance equation)
    \[
    b(n+1) = b(n) + Gv(n)
    \]
  4. Stop if each node has been scheduled the correct number of times
  5. Go to 3

Finding Schedules

- Repetition Vector \( q = \begin{bmatrix}
  3 \\
  2 \\
  1
  \end{bmatrix} \)
- **Schedules:**
  - AABABC
    - Max buffer required: 4
  - AAABBC
    - Max buffer required: 6
Finding Repetition Vector: CSDF

\[ P_i = \text{least common multiple of the lengths of all the token patterns of an actor} \]

\[ \sigma_i = \text{sum of tokens in pattern} \]

\[ p_i = \text{pattern length} \]

\[ T = \begin{bmatrix} P_i \sigma_i & \ldots & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix} \rightarrow Tq = 0 \rightarrow P_iq = \text{Actor repetition vector} \]

Finding a Schedule: CSDF

• Previous approach still applicable

• Algorithm:
  1. Find the repetition vector
  2. Form a list of all actors in the network
  3. For each actor in the list, schedule the actor if it is fireable, trying each node once. Take the cyclo-static consumption rates and production rates into account in the balance equation.
  4. Stop if each node has been scheduled the correct number of times
  5. Go to 3

Implementation

• JaCoP - Constraint Programming Solver
  – Krzysztof Kuchinski, CS, Lund
  – Java
  – http://www.jacop.eu/

• Problem formulated in miniZinc
  – Declarative, equation-oriented language for specification of finite-domain optimization problems
  – Translated into JaCoP

• Problem formulated directly in JaCoP
  – Java code

Implementation

• NP-complete problem
  – Timeout necessary

• Optimization criteria currently supported
  – Find all schedules (within the timeout)
  – Find the schedule(s) that minimizes the total internal buffer requirements
  – Find the schedule(s) that minimizes the size of the largest internal buffer

A schedule that minimizes the maximum buffer: ABCAAABCBAAAACBCABCAABC
Maximum buffer = 5

SDF actors a special case of CSDF \(\rightarrow\) Only CSDF scheduling implementing Mixed SDF/CSDF sub-networks handled
Actor Merging

- On the XLIM/XDF level
- Steps:
  - Merge actions
  - Name mangling
  - Replacement of internal FIFO buffers with state variables (ring buffers or scalars)
  - Merge action schedules

Graphical User Interface

- Java Swing
  - JGo Graphical Library

Demo

- MPEG 4 Simple Profile Video Decoder
- Static subnetwork involving 5 actors
  - 1-dimensional inverse discrete cosine transform (idct1d)
- Merging gives an 18% speedup measured in terms of frames per second
- The speedup for the idct1d network in isolation is 300%

Results

Summary

- Dataflow programming natural for media streaming applications on multicore platforms
- Static dataflow applications allows extensive off-line analysis and optimization
- A model compiler form actor merging has been developed within the ACTORS project
- 20-300% speedup depending on application

Questions?