An automatic parallelization tool for embedded systems, based on hierarchical task graphs

Motivation

- Using multicore systems can...
  - reduce CPU frequencies / exec. time
  - save energy
  - enable further optimizations
- Most embedded system applications are written in sequential C
  - Automatic parallelization beneficial

Problems

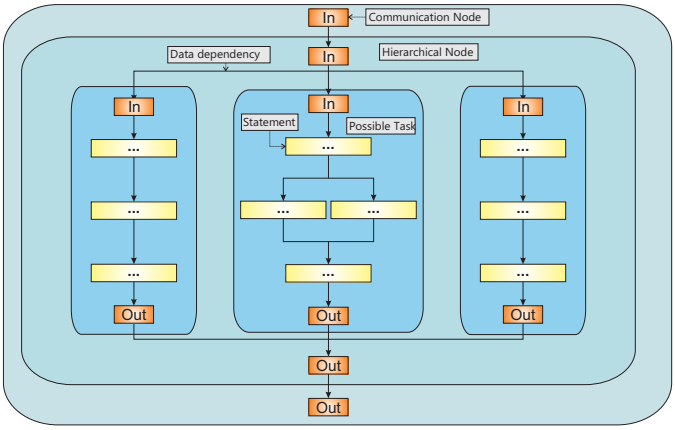
- Splitting program into tasks manually is...
  - error prone
  - time consuming
- Doing this automatically...
  - is a very complex problem with a very large search space
- Limitations of parallelization techniques of high-perfomance computing

Idea

- Focus on characteristics of embedded system applications and architectures
- Reduce search space by introducing hierarchy to the task graph model
- Clustering of each hierarchical block can be handled isolated of other nodes
- Use integer linear programming (ILP) formulation of problem to find best solution in each hierarchical block

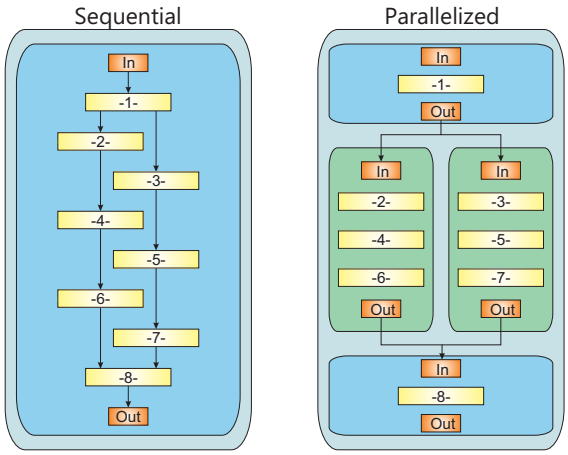
Tool Characteristics

Internal Model: Hierarchical Task Graph



- Sequential C-Code is translated automatically into hierarchical task graphs
- Elements of hierarchical task graphs:
  - Hierarchical node i.e. loops / conditional statements
  - Simple nodes for expression statements like 'a = 0;'
  - Data dependencies for communication
  - In/out node for every hierarchical node → encapsulates communication
- Advantages of model:
  - Each node can be optimized without having detailed knowledge about other nodes
  - Subgraphs are cycle free

Parallelization step



- Parallelization is done bottum-up in hierarchy
- Each node is transformed in ILP formulation to find possible tasks
- Objective function: minimization of longest (most expensive) path from communication in- to communication out-node
- ILP generates a sequential part before and after a parallel section
- ILP is solved several times for a different max. number of tasks
- Value of objective function is used as execution time of this node for the parallelization step of the parent node
- Tool is capable of generating only a restrictive number of tasks
- Results are combined to find solutions for parent nodes

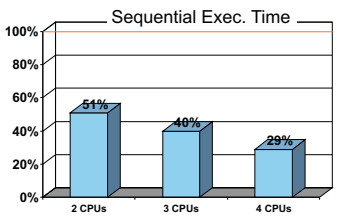
Experimental Results

- Results measured with MPARM simulator
  - Default configuration with shared memories used
  - Implementation of parallelization done using ATOMIUM tools from IMEC
  - Speedup up to 1,94 with 2 cores
  - Speedup up to 2,55 with 3 cores
  - Speedup up to 3,94 with 4 cores
- Approach has shown that it finds reasonable solutions
- Fast hierarchical approach

Benchmarks

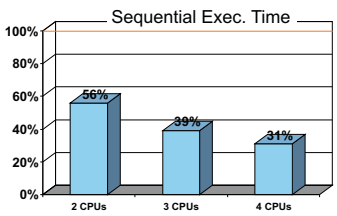
- Compress (UTDSP)

#Tasks	#Par. Tasks.	Speedup
2	2	1,94
3	3	2,51
4	4	3,94



- Boundary Value Problem (image processing)

#Tasks	#Par. Tasks.	Speedup
2	2	1,78
3	3	2,55
4	4	3,19



Daniel Cordes | Peter Marwedel | Web: www.icd.de, www.mnemee.org | Email: Lastname@icd.de
Informatik Centrum Dortmund | Joseph-von-Fraunhofer-Straße 20 | 44227 Dortmund | Germany