From CoPop to CoWare: What are the challenges?

Paul Pop
Embedded Systems Laboratory
Computer and Information Science Dept.
Linköpings universitet

Codesign Environments

- Application domain
  - Architecture, communication
- Internal representation, underlying model
- Input & Output
- Design flow
  - Problems solved
  - Tools
  - Interfaces
Tools: At a Glance

<table>
<thead>
<tr>
<th>CoPop</th>
<th>COSMOS</th>
<th>CoWare</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Information</strong></td>
<td>Paul Pop et al., Embedded Systems Lab, Ljöpings universitet</td>
<td>A.A. Jerraya et al., System-Level Synthesis Group, TIMA, France</td>
</tr>
<tr>
<td><strong>Application Domain</strong></td>
<td>Automotive electronics, communication: time triggered protocol</td>
<td>Distributed and communicating systems, library of protocols</td>
</tr>
<tr>
<td><strong>Internal Representation</strong></td>
<td>Conditional process graphs Simple architecture model</td>
<td>SOLAR Simple architecture model</td>
</tr>
<tr>
<td><strong>Design Flow</strong></td>
<td>Incremental design process Scheduling Communication synthesis Mapping Architecture selection</td>
<td>Transformational codesign approach User guided partitioning</td>
</tr>
</tbody>
</table>

---

CoPop

- **General information**
  - Paul Pop et al., Embedded Systems Lab, Ljöpings universitet
  - http://www.ida.liu.se/~paupo

- **Application domain**
  - Automotive electronics, communication: time triggered protocol

- **Internal representation**
  - Conditional process graphs
  - Simple architecture model

- **Design flow: Incremental design process**
  - Scheduling
  - Communication synthesis
  - Mapping
  - Architecture selection
Design Flow

- Scheduling...
  Scheduling of processes and messages for distributed hard real-time applications with control and data dependencies in the context of a given communication protocol.

- Communication synthesis...
  Optimization of the parameters of the communication protocol so that the overall system performance is increased and the imposed timing constraints are satisfied.

- Mapping...
  Incremental Design Process

- Architecture modelling, selection...

Design Flow: System Specification using UML

[Diagram showing system specification using UML]
public void run()
{
    Message[] msgs = receive("cem&&ecm&&tcm&&abs&&ccont");
    // .. Extracting messages from array msgs
    boolean lowBattery = cemMsg.vbatt < 9;
    boolean atglpNotDrive = tcmMsg.atglp != Signals.DRIVE && tcmMsg.atglp != Signals.DRIVE_L;
    boolean allQualitiesGood = tcmMsg.atglpqf == Signals.GOOD && absMsg.vsqf == Signals.GOOD;
    boolean vsVsaDiffHigh = Math.abs(absMsg.vs - tcmMsg.vsa)/absMsg.vs > 0.05;
    boolean wrongSpeed = absMsg.vs < 35 || absMsg.vs > 200;
    if (!ccont || lowBattery || atglpNotDrive || !allQualitiesGood || vsVsaDiffHigh || wrongSpeed) {
        send("cca_error", null);
    } else {
        Message[] mmsgs = receive("cca_error||cca_is_button_pressed");
        cca = (Boolean) getDataFromMessageArray(mmsgs, "cca_is_button_pressed");
        if (cca == null) {
            send("cca_is_button_pressed", cemMsg);
        }
        cca = (Boolean) getDataFromMessageArray(mmsgs, "cca_is_button_pressed");
        if (cca == null) {
            send("cca_is_button_pressed", cemMsg);
        }
        send("ccf", cca);
        send("ccsp", cca);
    }
}

Vehicle cruise controller.
Modelled with a CPG of 32 processes and two conditions.
Mapped on 5 nodes: CEM, ABS, ETM, ECM, TCM.
Conditional Process Graph (CPG)

Subgraph corresponding to \( D \lor C \lor K \)
Event-Triggered vs. Time-Triggered

- **Event-triggered**: activation of processes and transmission of messages is done at the occurrence of significant events.
- **Time-triggered**: activation of processes and transmission of messages is done at predefined points in time.

<table>
<thead>
<tr>
<th></th>
<th>processes</th>
<th>messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>event-triggered</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>time-triggered</td>
<td>✗</td>
<td>✗ ✗</td>
</tr>
</tbody>
</table>
Hardware Architecture

- Hard real-time distributed systems.
- Nodes interconnected by a broadcast communication channel.
- Nodes consisting of: TTP controller, CPU, RAM, ROM, I/O interface, (maybe) ASIC.
- Communication between nodes is based on the time-triggered protocol.

- Bus access scheme: time-division multiple-access (TDMA).
- Schedule table located in each TTP controller: message descriptor list (MEDL).

Problem Formulation: Scheduling

Input
- Safety-critical application with several operating modes.
- Each operating mode is modelled by a CPG.
- The system architecture and mapping of processes to nodes are given.
- The worst case delay of each process is known.

Output
- Local schedule tables for each node and the MEDL for the TTP controllers.
- Delay on the system execution time for each operating mode, so that this delay is as small as possible.

Note
- Processes scheduled with static cyclic non-preemptive scheduling, and messages according to the TTP.
Design Flow: Scheduler

Scheduling and Communication Synthesis

24 ms

\[
\begin{array}{llllll}
S_1 & S_2 & m_1 & m_2 & m_3 & m_4 \\
\text{Round 1} & \text{Round 2} & \text{Round 3} & \text{Round 4} & \text{Round 5} \\
\end{array}
\]

22 ms

\[
\begin{array}{llllll}
S_1 & S_2 & m_1 & m_2 & m_3 & m_4 \\
\text{Round 1} & \text{Round 2} & \text{Round 3} & \text{Round 4} \\
\end{array}
\]

20 ms

\[
\begin{array}{llllll}
S_1 & S_2 & m_1 & m_2 & m_3 & m_4 \\
\text{Round 1} & \text{Round 2} & \text{Round 3} \\
\end{array}
\]
Problem Formulation: Mapping

Input
- A set of existing applications modelled using process graphs.
- A current application to be mapped modelled using process graphs.
- Each process graph in the application has its own period and deadline.
- Each process has a potential set of nodes to be mapped to and a WCET.
- The system architecture is given.

Output
- A mapping and scheduling of the current application, so that:
  Requirement a: constraints of the current application are satisfied and minimal modifications are performed to the existing applications.
  Requirement b: new future applications can be mapped on the resulted system.

Notes
- Hard real-time applications
- Static cyclic scheduling of processes and messages
- Time-triggered protocol, TDMA
**Incremental Design Process**

- **Version N-1**
  - **Future applications**
  - **Current applications**
  - **Existing applications**

- **Map and schedule** so that the future applications will have a chance to fit.

- **No modifications** are performed to the existing applications.

---

**COSMOS**

- **General information**
  - A.A. Jerraya et al., System-Level Synthesis Group, TIMA, France
  - [http://tima.imag.fr/SLS/](http://tima.imag.fr/SLS/)

- **Application domain**
  - Distributed and communicating systems, library of protocols

- **Internal representation**
  - SDL -> SOLAR -> C/VHDL
  - Simple architecture model

- **Design flow:** transformational codesign approach
  - User guided partitioning
Problems with Automation

- To be successful, very restrictive.
- Lack of a universal estimation method, no realistic evaluation procedure
- No correspondence (to the user, no explanation) between the initial specification and the resulting architecture.
- The designer already has a good solution in mind when it starts working.
- The design is of complex systems is iterative.

Design Flow

Traditional codesign flow

Transformational codesign flow
User-Guided Transformational Partitioning

SOLAR

- Transformations work on the SOLAR model
- Internal design representation
  - Extended Finite State Machines for behaviour
  - Remote procedure call for communication
    - High level communication concepts
      - Channels
      - Shared global variables
      - Can model complex communication protocols
**Basic concepts:** State table, Design unit, Channel unit

**Transformations: Step 1**

Transformational partitioning, step 1: functional specification
Transformations: Step 2

- Functional decomposition (partitioning)
  - Transform behaviours (state tables)
  - Split, Merge, Move, Flat

Transformational partitioning, step 2: functional decomposition

Transformations: Step 3

- Structural reorganization (mapping, allocation)
  - Distributes the behaviours to design units (processors)
  - Split, Merge, Move, Map, Flat

Transformational partitioning, step 3: structural reorganization
Transformations: Step 4

- Communication transformation (communication synthesis)
  - From high level primitives to protocols from a library
  - Map, Merge

Example: Robot Arm Controller
Example: Answering Machine

Functional Decomposition: Split

- **Split**
  - Decomposes a sequential machine into a set of submachines.
  - Each resulting machine can be placed into a different partition.
  - Control signals and wait states have to be added.
**Functional Decomposition: Merge**

- **Merge**
  - Groups a set of sequential machines into a unique machine.
  - Sharing of resources.

**Functional Decomposition: Move**

- **Move**
  - Transforms the hierarchy.
  - Can be used to move code from a software partition to a hardware partition.
Structural Reorganization: Split

- **Split**
  - Works on the behaviour of parallel processes (state tables) to split them into a set of independent modules (design units).
  - Shared data converted to abstract channels.

Structural Reorganization, Cont.

- **Merge**
  - Groups a set of modules into a new design unit.
  - Clusters the modules that will be assigned to the same processor.

- **Move**
  - Moves a design unit in the hierarchy.
  - Used to prepare a merge operation.

- **Map**
  - Decides the hardware or software implementation.

- **Flat**
  - Structural flattening operation on the hierarchy.
Communication Transformation

Structural Reorganization: Split

- **Split**
  - Works on the behaviour of parallel processes (state tables) to split them into a set of independent modules (design units).
  - Shared data converted to abstract channels.
General information
- D. Verkest et al., IMEC, CoWare
  - http://www.coware.com

Application domain
- Heterogeneous systems on a chip

Internal representation
- Data model
  - Processes, ports, protocols, threads, terminals, channels
- Communication: Remote Procedure Call

Design flow
- Simulation
- Hardware/Software partitioning, mapping
- Synthesis

Design Flow
Example: Spread-Spectrum Pager

- Spread-spectrum pager
  - DSP

- Codesign
  - Conceptual specification
  - Partitioning and mapping
  - Communication selection
  - Implementation of components
  - Interface synthesis
Example: Functional Specification

- Each component corresponds to a process implementing a specific function of the pager.
- This functional decomposition determines the initial partitioning.
- The arrows in between the processes indicate communication via RPC.

Example: Architectural Description

- Interconnection of processes
- Processes are processor implementations
- Structural VHDL
Example: Partitioning

Implementation of Components

Encapsulated C process on an Encapsulated processor
What are the challenges?

- What is it needed to make the tools useful?
  - Usable (user interface)
  - Interoperability
  - End-user modifiability
  - Collaboration
  - Support the cognitive processes of the user
    - Visual representations of models
    - Workflow
    - Help systems
    - Computational design critics