SafetyChip
A Time Monitoring and Policing Device

Gustaf Naeser, Lars Asplund and Johan Furunäs
Mälardalen University

Outline
- Formal Methods
- Hardware Monitoring

Application: Robocup – Soccer playing robots

Hardware Monitoring and Policing
- How can the operation of a system during run time be
  - Observed
  - Modified
- Aids development
- The largest part of a system’s life is after deployment

Kinds of monitoring
- Intrusive
  - Changes the behaviour of the monitored system
    - Code is added to generate observable events
    - Monitor can be off target
    - Instructions are kept in the final system to keep the behaviour
- Non-intrusive
  - Does not change the systems behaviour
    - Harder to implement since it requires access to tap points “inside” the system

The SafetyChip framework parts

The Framework
A Ravenscar RTK
- Component based
- Allows timing analysis

Monitoring and Policing
Reuses the model from the verification

Non-Intrusive Fault-Tolerance
Reuses the model from the verification

Developments in hardware
- Field Programmable Gate Arrays (FPGA) is a programmable hardware combining properties from both hardware and software
  - Faster than SW, slower than HW (ASIC)
  - Easier to change than HW, harder than SW

RavenHaRT
- Hardware implemented RTK
  - Based on the Ravenscar profile of Ada95
  - Implemented using the Xilinx Virtex II Pro
- Deterministic run-time behavior
- Non-Deadlocking Inter-task communication

RTK
- Ravenscar profile supported
- Supports a more functionality at low cost
  - e.g., dynamic priorities
  - Multiple processors
- Component design to allow
  - Easy change
  - HW/SW locality in final system
- Implemented using generic templates
  - Allows the kernel to be tailored to each application
RTK components

- Ready Queue
- Delay Queue
- Protected Objects Queue
- Interrupt Queue (uses PO)
- Hardware
  - Processors with null tasks
  - clock
- Application
  - Tasks and protected objects

The Ada Code for task T1

```
with Task_T1;

task body Task_T1 is
  s : Int_Type := 0;
  rs : Real_Type := 0.0;
  run (s : Int_Type := 10);
  while Run_Time <= System_Time loop
    s := s + 1;
    if Run_Time <= System_Time then
      s := s + 1;
    end if;
    delay (rs);
    if Run_Time <= System_Time then
      s := s + 1;
    end if;
  end loop;
end Task_T1;
```

UPPAAL of Task T1

Timing of Task T1

Task T3

Behaviour for Task T3
RoboCup

- Continued

Team Aros

Aros disk – Camera

FPGA and Cameras

Block Diagram of the Vision System
Camera Block Diagram

IR-Ultra

IR / Ultra

No Synchronization

Searching

Synchronized - contact
IR/Ultra between 4 robots

Ultra sonic ‘radar’

Ir – Collision Avoidance

Gyro

Line up at start

Rotation

X1 = Omega * R
X2 = -Omega * R
Y1 = 0
Y2 = 0
X1-X2 = 2 * Omega * R
X1+X2 = 0
Y1+Y2 = 0
X1-X2 styr moton (kompensation)
### X-translation

<table>
<thead>
<tr>
<th>Condition</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 = a )</td>
<td>( X_1 = a )</td>
</tr>
<tr>
<td>( X_2 = a )</td>
<td>( X_2 = a )</td>
</tr>
<tr>
<td>( Y_1 = 0 )</td>
<td>( Y_1 = 0 )</td>
</tr>
<tr>
<td>( Y_2 = 0 )</td>
<td>( Y_2 = 0 )</td>
</tr>
<tr>
<td>( X_1 - X_2 = 0 )</td>
<td>( X_1 - X_2 = 0 )</td>
</tr>
<tr>
<td>( X_1 + X_2 = 2a )</td>
<td>( X_1 + X_2 = 2a )</td>
</tr>
<tr>
<td>( Y_1 + Y_2 = 0 )</td>
<td>( Y_1 + Y_2 = 0 )</td>
</tr>
</tbody>
</table>

### Y-translation

<table>
<thead>
<tr>
<th>Condition</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 = 0 )</td>
<td>( X_1 = 0 )</td>
</tr>
<tr>
<td>( X_2 = 0 )</td>
<td>( X_2 = 0 )</td>
</tr>
<tr>
<td>( Y_1 = a )</td>
<td>( Y_1 = a )</td>
</tr>
<tr>
<td>( Y_2 = a )</td>
<td>( Y_2 = a )</td>
</tr>
<tr>
<td>( X_1 - X_2 = 0 )</td>
<td>( X_1 - X_2 = 0 )</td>
</tr>
<tr>
<td>( X_1 + X_2 = 0 )</td>
<td>( X_1 + X_2 = 0 )</td>
</tr>
<tr>
<td>( Y_1 + Y_2 = 2a )</td>
<td>( Y_1 + Y_2 = 2a )</td>
</tr>
</tbody>
</table>

**Murphy**

- Arm \( 2(2+1) = 6 \)
- Back \( 3 \)
- Leg \( 2(3+1) = 8 \)
- Feet \( 2^2 = 4 \)
- Total \( = 21 \)