

## Temporal Skeletons for Verifying Time

SigAda 2005  
2005-11-15

**Gustaf Naeser, Kristina Lundqvist  
and Lars Asplund**

Mälardalen University  
And  
MIT

**MRTC**  
MÄLARDALEN REAL-TIME  
RESEARCH CENTRE

**MÄLARDALENS HÖGSKOLA**

## Outline

- **Formal Methods**
- Hardware Monitoring

Application : Robocup – Soccer playing robots

SigAda 2005

## Assuring that a system works

- There are several strategies for ensuring that a developed system meets the safety demands that
  - Testing
  - Inspection
  - Reviewing
  - Simulation
  - *Formal verification*

SigAda 2005

## Formal verification

- Formal methods = mathematically based
  - The word *mathematics* often scares designers and developers
  - Likely to require domain experts
  - Complement to testing
- Two kinds
  - Theorem proving
  - *Model checking*

SigAda 2005

## The SafetyChip framework parts

SigAda 2005

## The Framework

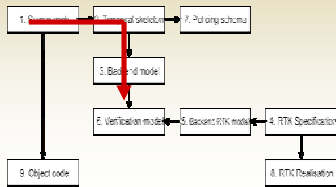
```

    graph TD
      1[1. Safety specs] --> 2[2. Temporal skeletons]
      2 --> 3[3. Backend model]
      3 --> 4[4. RTK Specification]
      4 --> 5[5. RTK Realisation]
      2 --> 6[6. Verification model]
      6 --> 7[7. Backend RTK model]
      7 --> 8[8. RTK Realisation]
      1 --> 9[9. Object code]
  
```

SigAda 2005

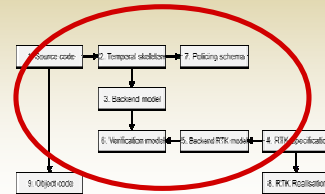
## Modelling

Can be visualised over the original source code



## Verification

The model and the RTK are used to verify the behaviour of the system



## The SafetyChip framework

- Desire to use FM in development of embedded high-integrity real-time systems
  - Must know properties of the software
  - Must know properties of the hardware
  - Reduce investment
    - » Between single computer and triple redundant
  - Reuse investment

## Ada Ravenscar source code and ASIS

- Source code parsed using the Ada Semantic Interface Specification
  - ASIS provides a parse tree
  - Additional information must be added (scopes are not available in ASIS)
  - ASIS can effectively be used to decide static properties of a program
  - Dynamic properties cannot be decided

## Intermediate code

- Intermediate code is used in order to reuse of the framework itself, i.e., to allow several front ends and back ends
  - Frontends: Input languages
    - » Ada, VHDL
  - Backends: Verification tools
    - » UPPAAL
- Automated transformation to and from the intermediate format!

## The Ada code for Task2

```

1 task Task2;
2 task body Task2 is
3   T_2 : Task_ID := 2;
4 begin
5   loop
6     PO.Open(T_2);
7     if (Global = 10) then
8       Global := 0;
9       Work(0.2, 0.3, T_2);
10    PO.Call(T_2);
11    end if;
12  end loop;
13 end Task2;
  
```

### Temporal skeletons

- Blocks of instructions that only consume time (full lined boxes)
- Blocks with instructions that "actively" change temporal behaviour (dashed boxes)

```

1 task Task2;
2 task body Task2 is
3   T_2 : Task_ID := 2;
4 begin
5   loop
6     PO.Open(T_2);
7     if (Global = 10) then
8       Global := 0;
9       Work(0.2,0.3, T_2);
10      PO.Call(T_2);
11    end if;
12  end loop;
13 end Task2;
  
```

SigAda 2005 16

### Temporal skeletons

- Skeletons presented to user contains only flow information that the user can change
- Must be manually extended with timing information
- Complexity and size only added by temporal instructions

```

1 task Task2;
2 task body Task2 is
3   T_2 : Task_ID := 2;
4 begin
5   loop
6     PO.Open(T_2);
7     if (Global = 10) then
8       Global := 0;
9       Work(0.2,0.3, T_2);
10      PO.Call(T_2);
11    end if;
12  end loop;
13 end Task2;
  
```

SigAda 2005 17

### UPPAAL automata

- UPPAAL used for verification
- Skeletons are automatically translated to the tool's TA
- TA optimisation to reduce state space

SigAda 2005 18

### Full UPPAAL Graph

SigAda 2005 19

### VHDL Translator

```

1 BEGIN
2 PROCESS(Resetn, Clock)
3 BEGIN
4   IF Resetn = '0' then
5     y <= A;
6   ELSIF (Clock'EVENT AND Clock = '1') then
7     CASE state IS
8       WHEN idle =>
9         state <= read;
10        WHEN read=>
11          IF (RxAv=1) then
12            ReadB <= 0;
13            end if;
14          IF (IP2Bus_ack='1' AND Bus2IP_CS='1') then
15            Bus2IP
16            txdata;
17            txdata;
18            txdata;
19            txdata;
20            end if;
21          IF (Bus2IP_LoadA
22            state <=
23            ELSE
24            state <=
25            end if;
26          WHEN re
27          IF (TxB
28            LoadA
29            state<=
30            state<=
  
```

- Enable formal verification of legacy components

SigAda 2005

### RoboCup

- 1997 Deep Blue beat Kasparov in chess
- In 50 years robots are going to beat the humans in Soccer
- Different Leagues

SigAda 2005 21

**RoboCup**

### Mini League




- A video camera above the field
- Radio communication 433MHz
- Two motors
- Rotating disc as kicker
- Field the size of table tennis table

SigAda 2005 22


**RoboCup**

### Sony Legged Robots



**RoboCup**

### Middle Sized League

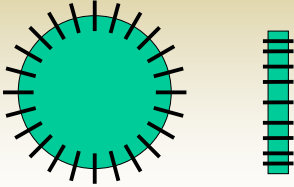


- Field size 5m\*8m
- Robot size < 50 (45) height < 80 cm weight < 80 kg!!
- No global sensors
- Communication OK
- Server
- Colour scheme robots, field, goals and ball

SigAda 2005 24


**RoboCup**

### Swedish Wheels

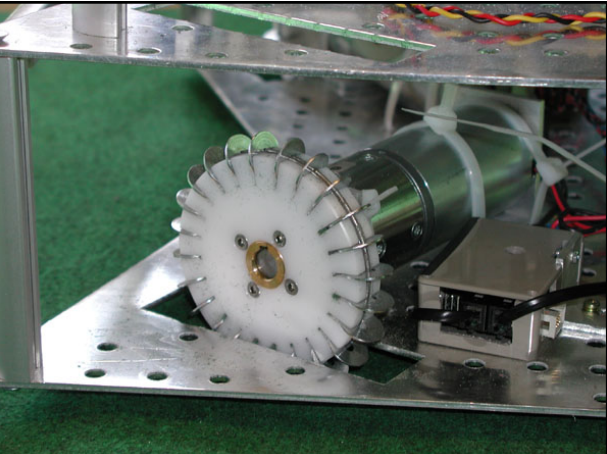


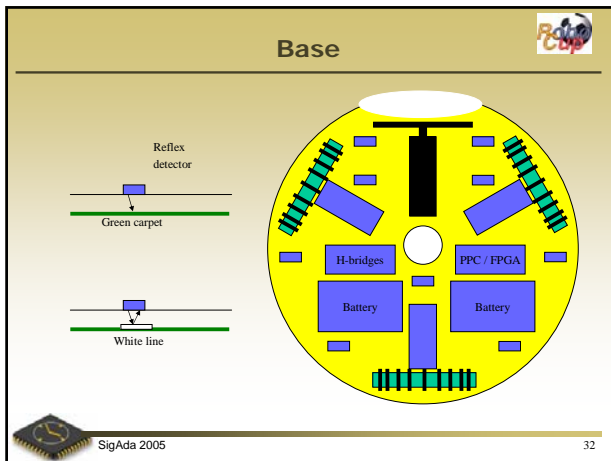
SigAda 2005 28

**RoboCup**

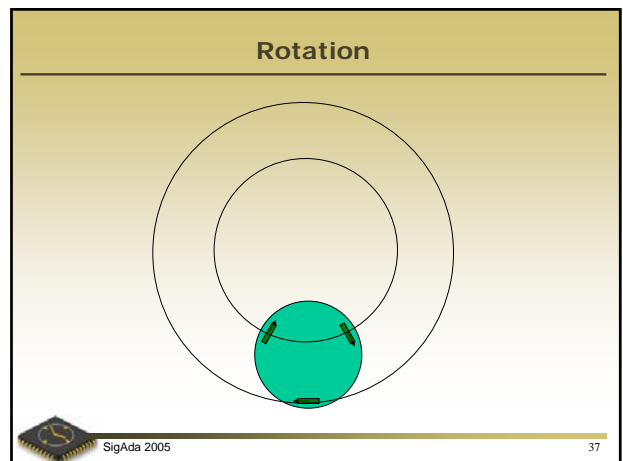
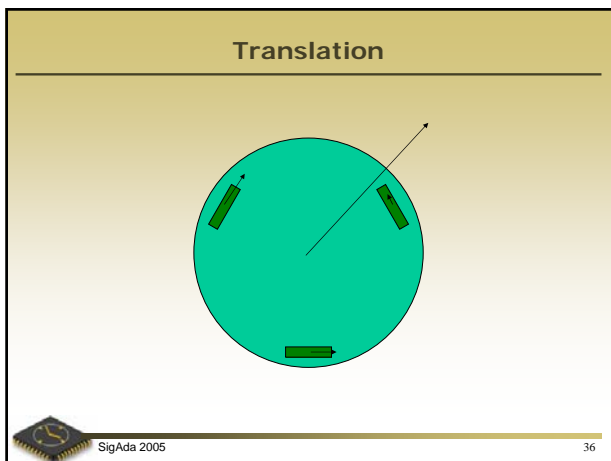
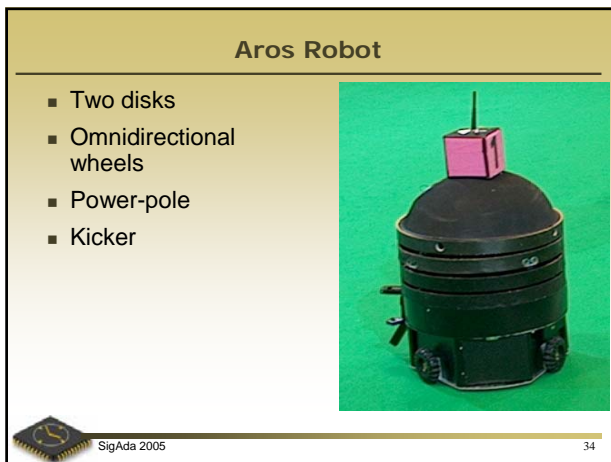


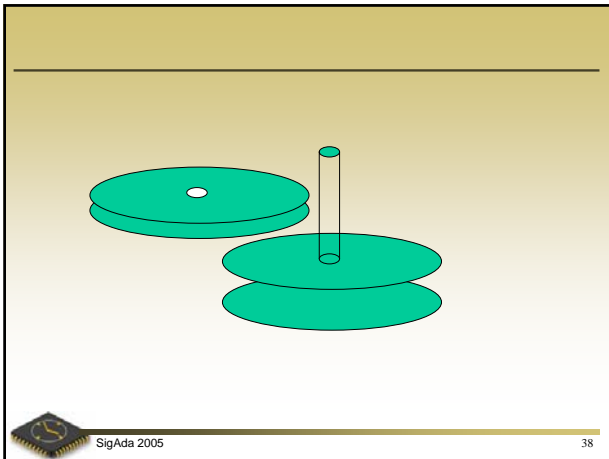
SigAda 2005 29





- ### Embedded System
- Software on the robots are Ada-95
    - Ravenscar profile
    - FPGA for all IO; VHDL
  - Software in the coach computer is full Ada
  - Communication between server and robots are by using RadioLAN
  - Communication inside the robot is by using an Optical CAN-bus
- SigAda 2005 33





**SafetyChip**  
A Time Monitoring and Policing Device

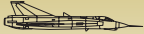
SigAda 2005  
2005-11-16

**Gustaf Naeser, Lars Asplund and  
Johan Furunäs**


Mälardalen University

**Outline**

- Formal Methods
- **Hardware Monitoring**



Application : Robocup – Soccer playing robots



SigAda 2005 42

**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

SigAda 2005 43


### 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


SigAda 2005
44


### The SafetyChip framework parts



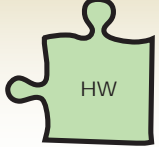
Application




FM



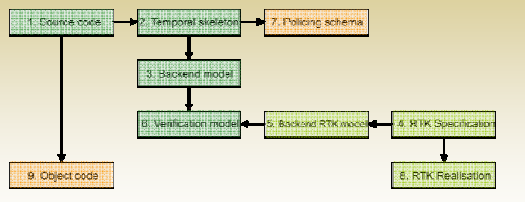
RTK



HW



SigAda 2005
45

### The Framework



```

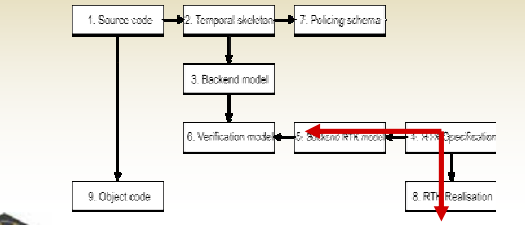
    graph TD
      1[1. Source code] --> 2[2. Temporal skeleton]
      1 --> 9[9. Object code]
      2 --> 3[3. Backend model]
      2 --> 7[7. Policing schema]
      3 --> 6[6. Verification model]
      6 --> 5[5. Backend RTK model]
      5 --> 4[4. RTK Specification]
      4 --> 8[8. RTK Realisation]
  
```


SigAda 2005
46

### System on Chip


A Ravenscar RTK

- Component based
- Allows timing analysis



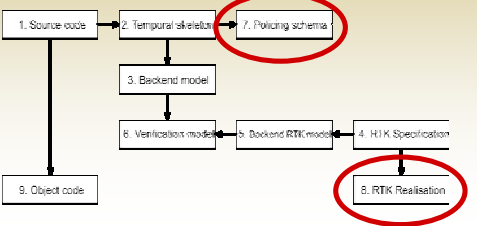
```

    graph TD
      1[1. Source code] --> 2[2. Temporal skeleton]
      1 --> 9[9. Object code]
      2 --> 3[3. Backend model]
      2 --> 7[7. Policing schema]
      3 --> 6[6. Verification model]
      6 --> 5[5. Backend RTK model]
      5 --> 4[4. RTK Specification]
      4 --> 8[8. RTK Realisation]
  
```


SigAda 2005
47


### Monitoring and Policing

Reuses the model from the verification

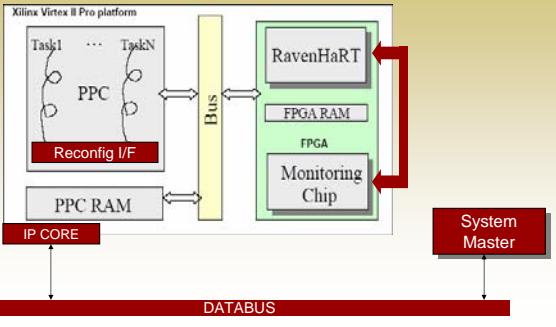


```

    graph TD
      1[1. Source code] --> 2[2. Temporal skeleton]
      1 --> 9[9. Object code]
      2 --> 3[3. Backend model]
      2 --> 7[7. Policing schema]
      3 --> 6[6. Verification model]
      6 --> 5[5. Backend RTK model]
      5 --> 4[4. RTK Specification]
      4 --> 8[8. RTK Realisation]
  
```


SigAda 2005
48

### Non-Intrusive Fault-Tolerance



Xilinx Virtex II Pro platform

Task1 ... TaskN  
PPC  
Reconfig I/F  
PPC RAM


Bus

RavenHaRT  
FPGA RAM  
FPGA  
Monitoring Chip

IP CORE

System Master

DATABUS


SigAda 2005
51

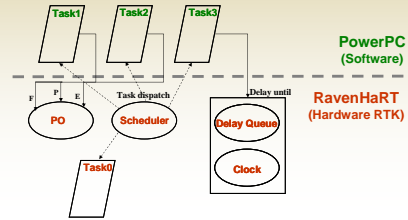
## Developments in hardware

- 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

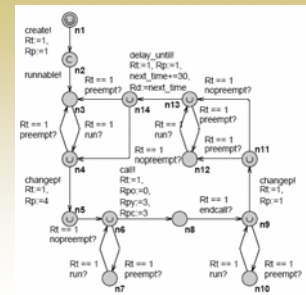
- Ready Queue
- Delay Queue
- Protected Objects Queue
- Interrupt Queue (uses PO)
- Hardware
  - Processors with null tasks
  - clock
- Application
  - Tasks and protected objects

## The Ada Code for task T1

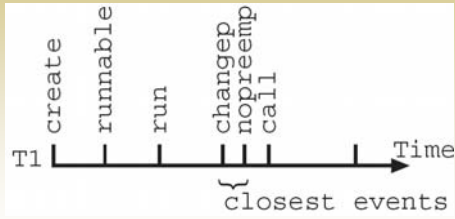
```

1 task T1;
2 task body T1 is
3   T_ID : Task_ID := 1;
4   Next_Time : Time := Start_Time
5     + To_Time_Span(3.0);
6 begin
7   loop
8     PO.P(T_ID);
9     Next_Time := Next_Time + To_Time_Span(3.0);
10    delay until Next_Time;
11  end loop;
12 end T1;
    
```

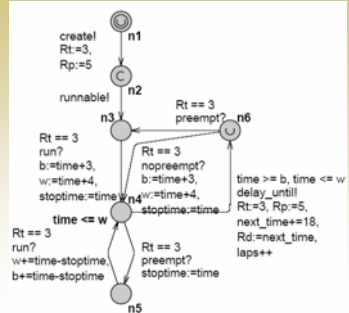
## UPPAAL of Task T1



### Timing of Task T1



### Task T3



### Behaviour for Task T3

Address	Branch	Call / Node	BCET	WCET
00	00	node (n1)	0	0
01	02	runnable	0	0
02	03	period	18	18
03	00	node (n1)	0	0
04	05	run	0	0
05	00	node (n1)	3	4
06	07	delay_until	3	4
07	00	node (n1)	0	0
08	03	preempt	0	0
09	10	nopreempt	0	0
10	05	period	18	18

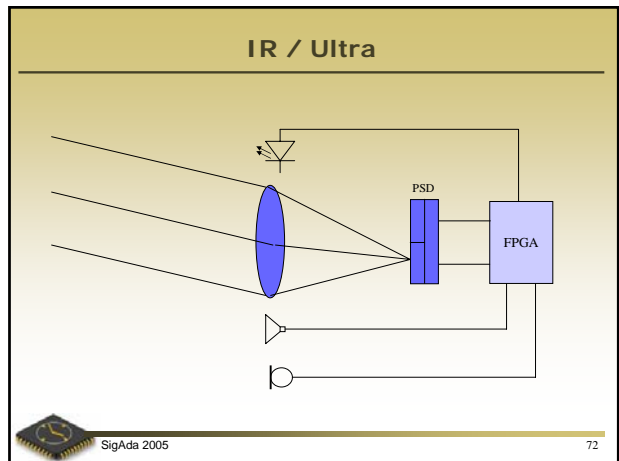
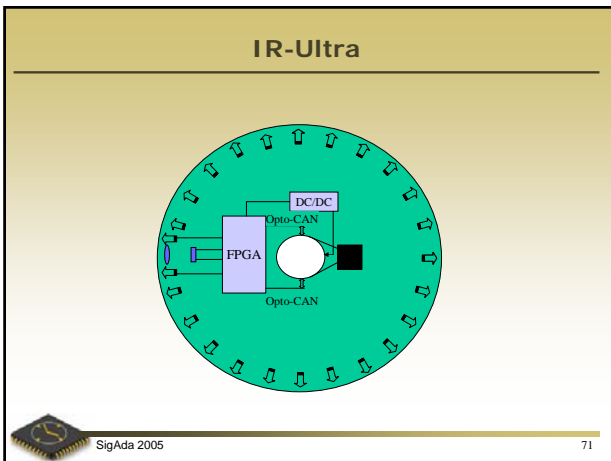
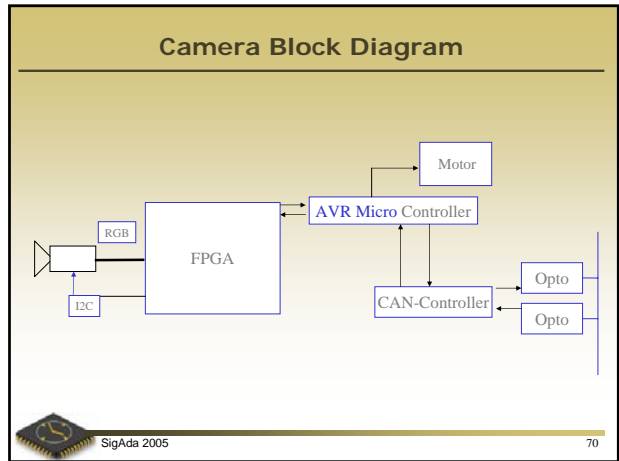
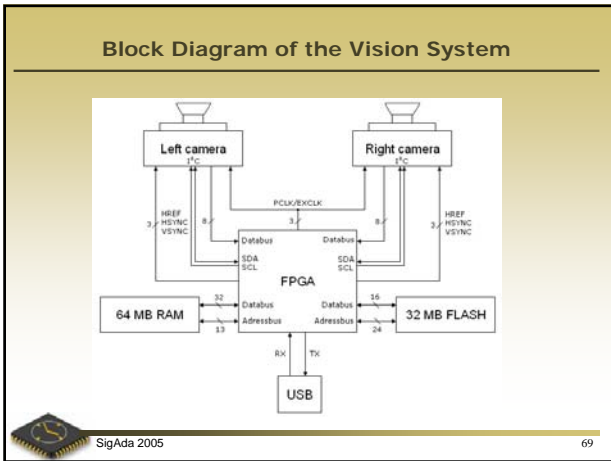
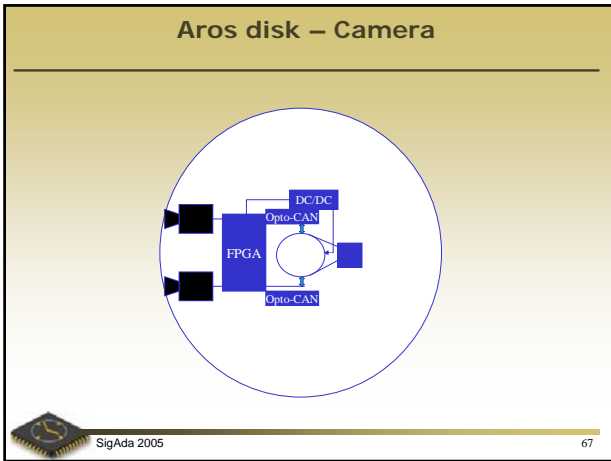


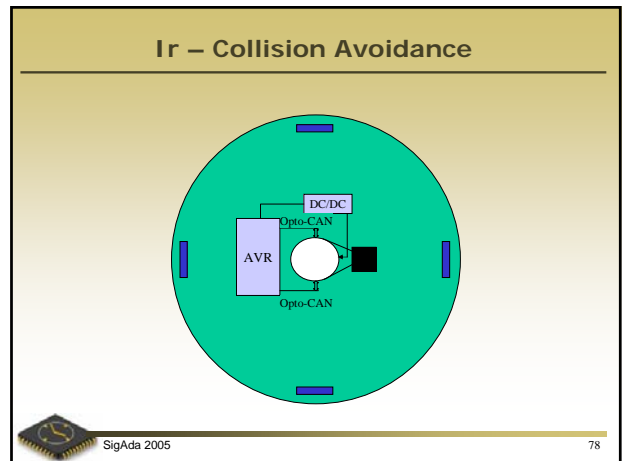
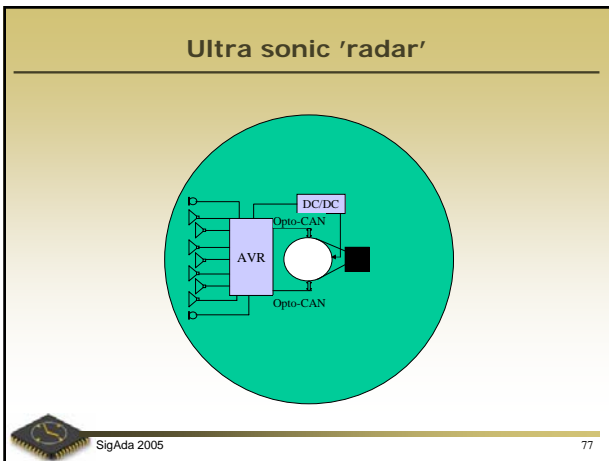
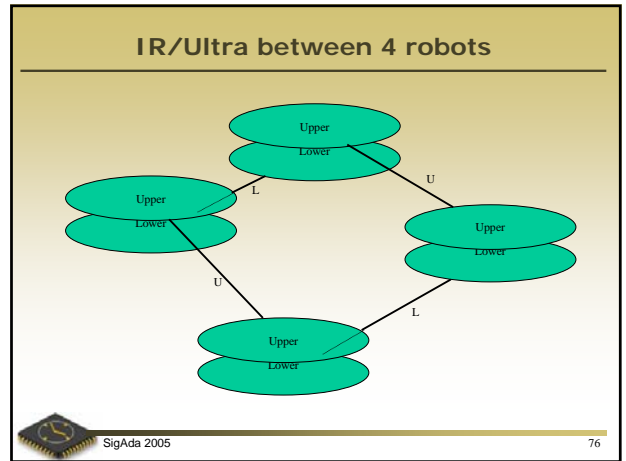
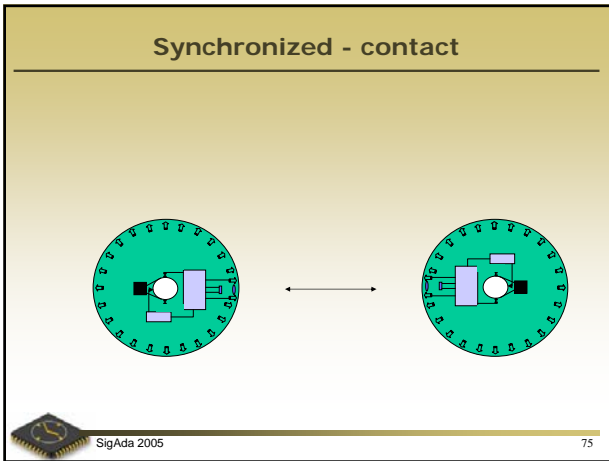
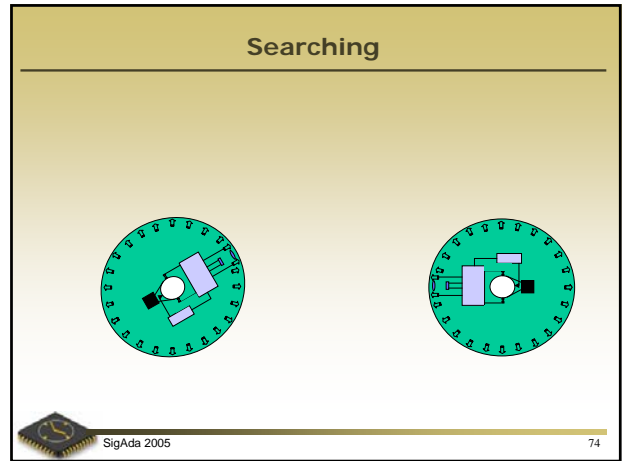
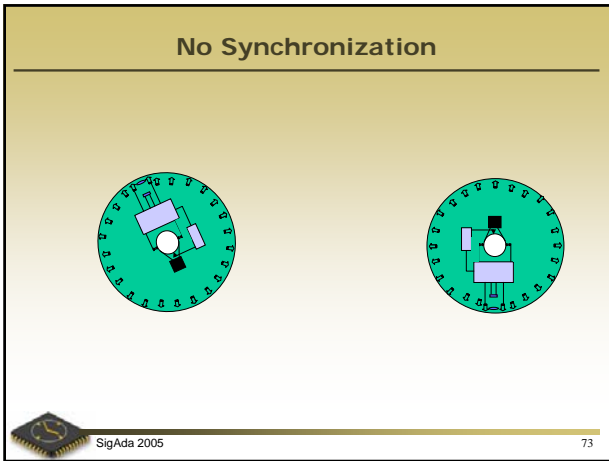
### RoboCup

- Continued

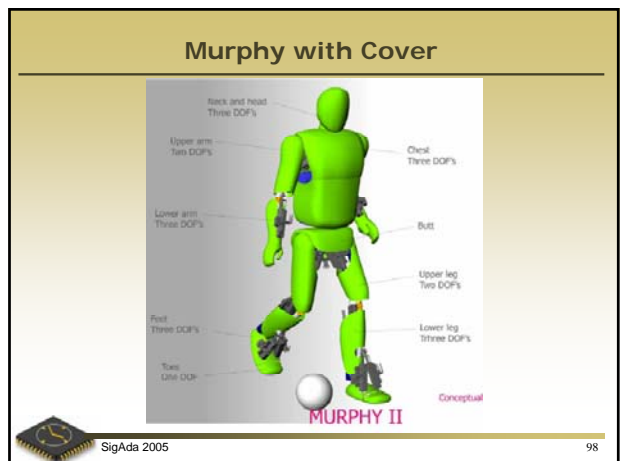
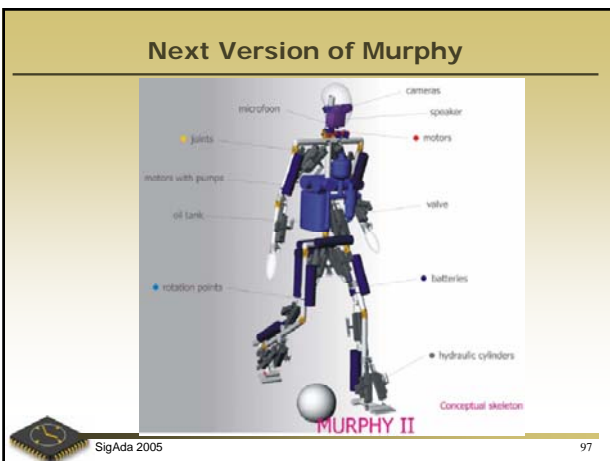
### Team Aros



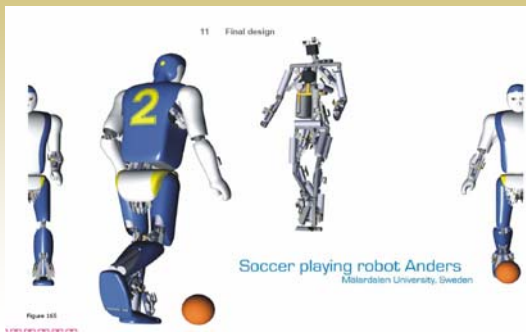






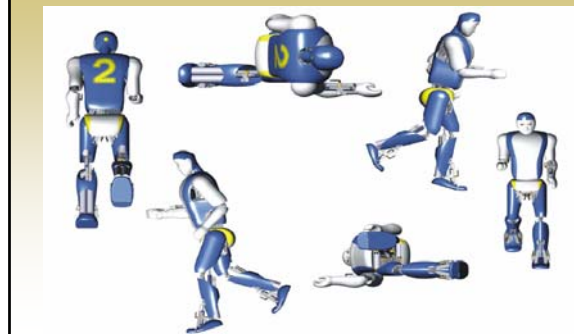


## New Name = Anders??



SigAda 2005

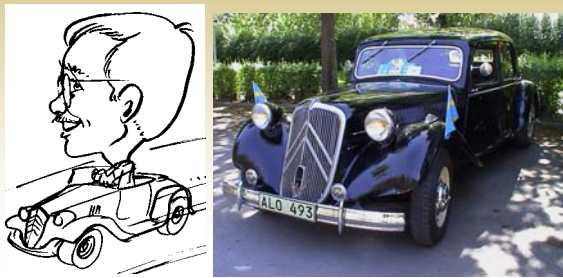
99



SigAda 2005

100

## The End



SigAda 2005

101