



Tutorial: TTool AMS Extension

Daniela Genius, Rodrigo Cortés Porto,
Ludovic Apvrille

FDL 2019





Outline

- ▶ TTool Introduction
 - ▶ Case study
- ▶ TTool AMS
 - ▶ Hands-on exercise 1
- ▶ Co-Simulation
 - ▶ Hands-on exercise 2

Design Methodology of an Embedded System

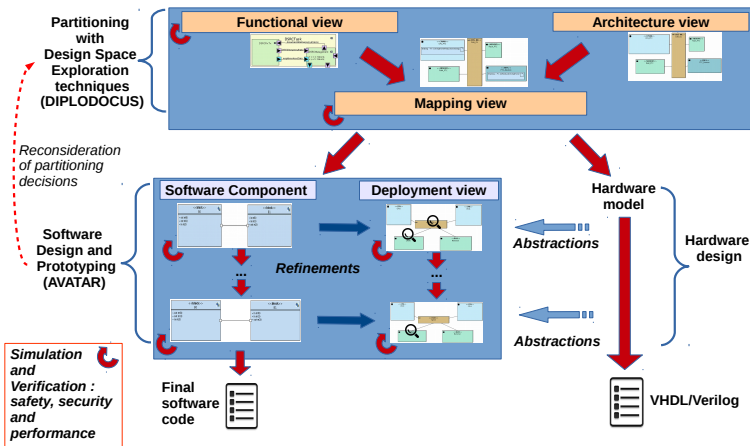
System Partitioning between HW and SW

- ▶ Functions → execution nodes
- ▶ Communications between functions → communication and storage nodes
- ▶ "System-level partitioning" with abstract models

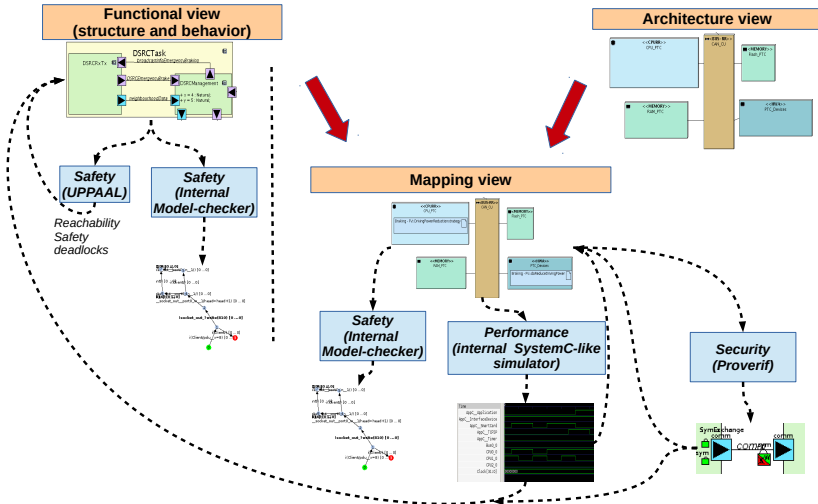
Software and hardware design

- ▶ Designed independently, Integration phase
- ▶ Problem: late Discovery of bad partitioning, difficult to iterate between partitioning and design
- ▶ TTool: model-based approach with close interaction between partitioning and software design

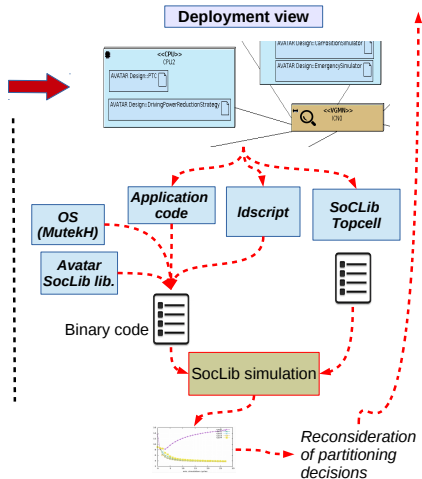
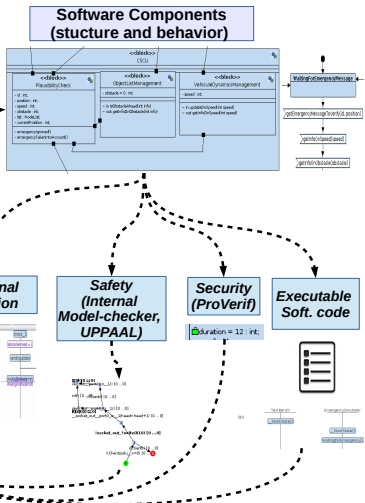
Overall Method



Partitioning Method

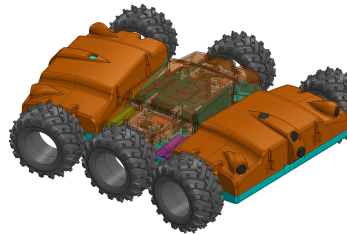


Software Design Method



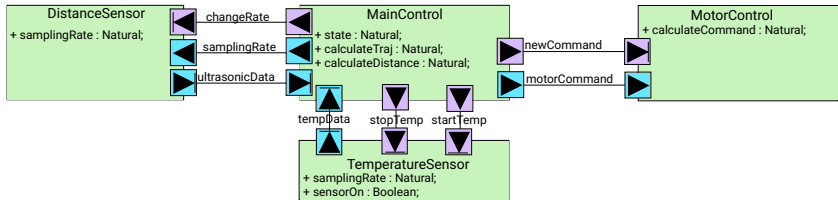
Case Study: Rover

Autonomous vehicle for disaster relief efforts (earthquake)

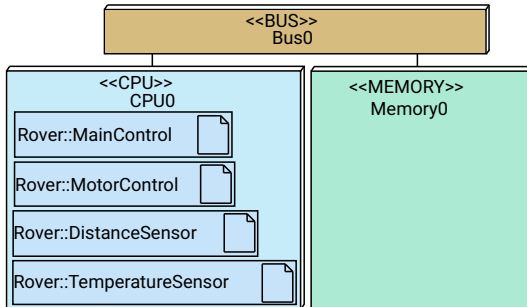


- ▶ Telemetric sensors to detect obstacles and navigate terrain autonomously
 - ▶ No obstacles in proximity → decrease sampling rate
 - ▶ Obstacle detected in close proximity → increase sampling rate
- ▶ Temperature and pressure sensors
- ▶ Avoid collisions → set time frame → impose maximal latency

Partitioning: Functional View



Partitioning: Mapping View





Partitioning: CPU Configuration

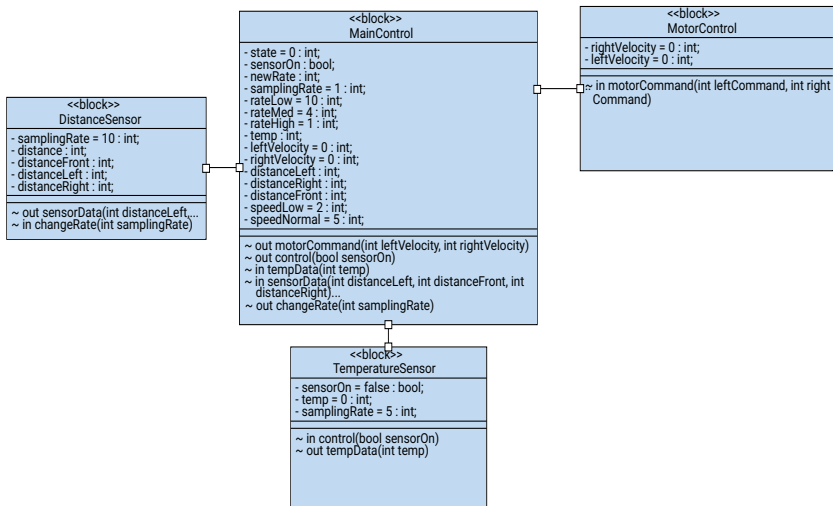
Simulation **Code generation**

CPU attributes

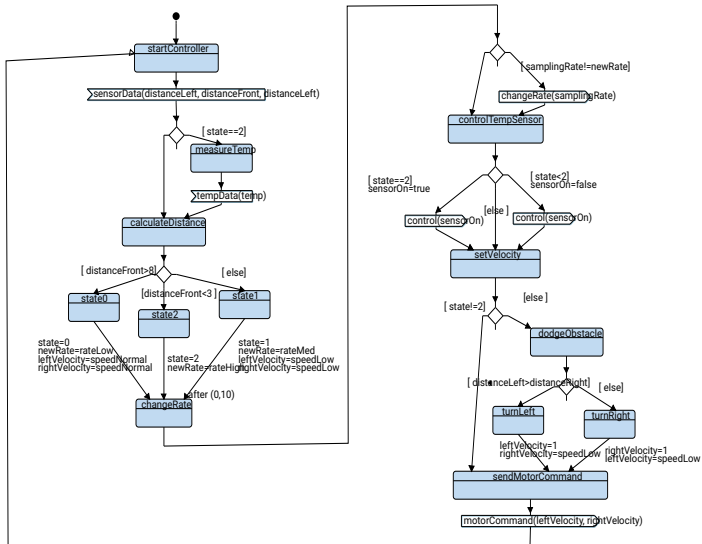
CPU name:	CPU0
Scheduling policy:	Round Robin ▼
Slice time (in microseconds):	10000
Nb of cores:	1
Data size (in byte):	4
Pipeline size (num. stages):	5
Task switching time (in cycle):	20
Mis-Branching prediction (in %):	2
Cache-miss (in %):	5
Go idle time (in cycle):	10
Max consecutive cycles before idle (i...):	10
EXECI execution time (in cycle):	1
EXECC execution time (in cycle):	1
Clock divider:	1

 **Save and Close**  **Cancel**

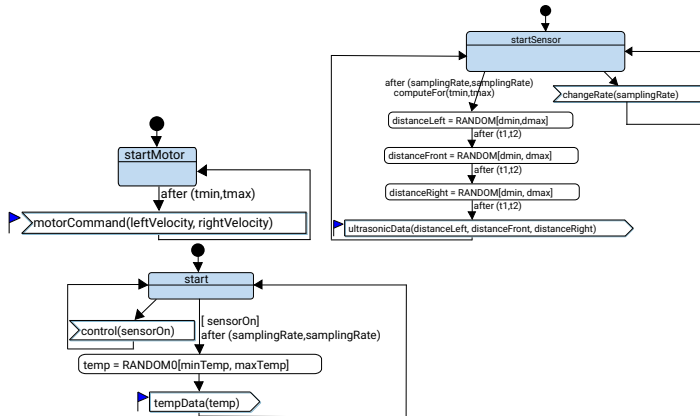
Software Design: Block Diagram



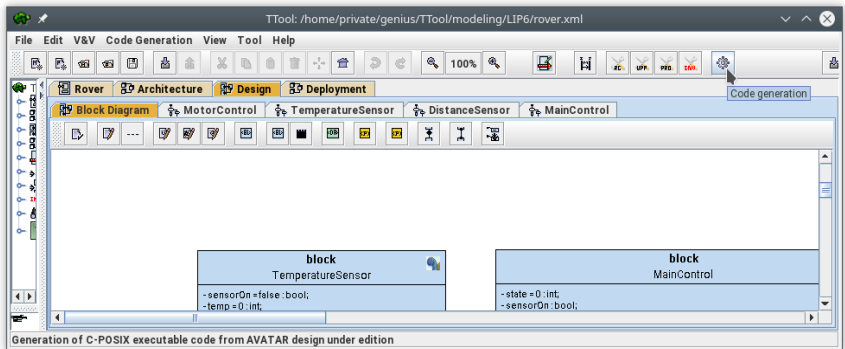
Software Design: State Machine Diagrams



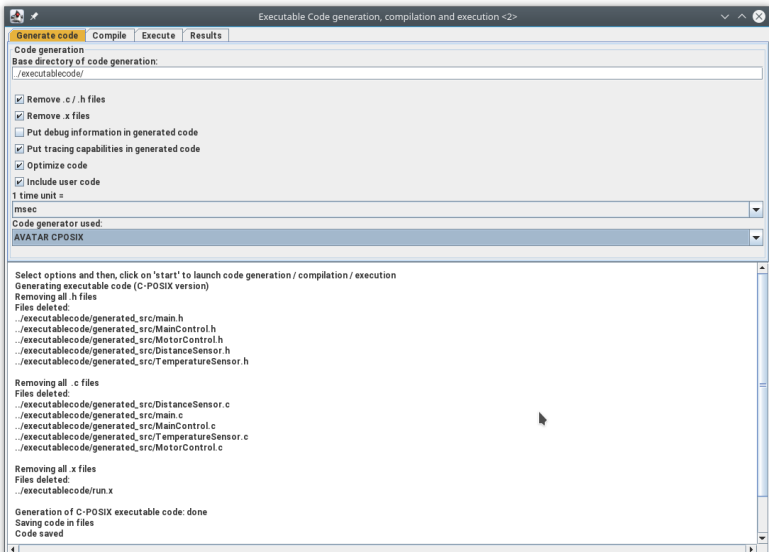
Software Design: State Machine Diagrams



C Posix code generation



C Posix code generation (contd.)



Compilation on local workstation

Generate code
Compile
Execute
Results

Execution

☐ Run code:

```
./executablecode/run.x
```

☒ Run code and trace events (if enabled at code generation):

```
./executablecode/run.x ./executablecode/trace.txt
```

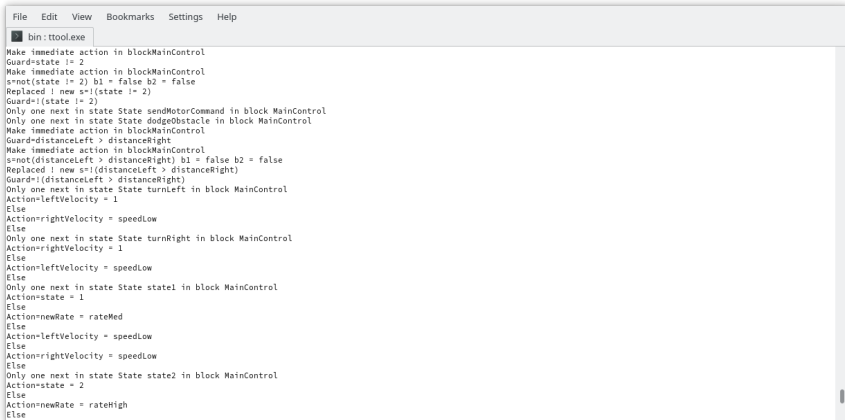
☐ Run code in soclib / mutekh:

```
make -C ./MPSoc/ runsoclib
```

Compiling executable code with command:

```
make -C ./executablecode/
make: Entering directory '/home/private/genius/TTool/executablecode'
echo Making directories
Making directories
mkdir -p ./lib
mkdir -p ./lib/generated_src/
mkdir -p ./lib/src/
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/generated_src/main.o -c generated_src/main.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/generated_src/MotorControl.o -c generated_src/MotorControl.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/generated_src/TemperatureSensor.o -c generated_src/TemperatureSensor.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/generated_src/DistanceSensor.o -c generated_src/DistanceSensor.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/generated_src/MainControl.o -c generated_src/MainControl.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/request.o -c src/request.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/message.o -c src/message.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/myerrors.o -c src/myerrors.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/debug.o -c src/debug.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/synchchannel.o -c src/synchchannel.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/asynchchannel.o -c src/asynchchannel.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/request_manager.o -c src/request_manager.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/random.o -c src/random.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/mytimelib.o -c src/mytimelib.c
./usr/bin/gcc -O1 -pthread -Wall -I. -I. -Isrc/ -Igenerated_src/ -o lib/src/tracemanager.o -c src/tracemanager.c
./usr/bin/gcc -O1 -pthread -ldl -rt -Wall -I. -I. -Isrc/ -Igenerated_src/ -o run.x lib/generated_src/main.o lib/generated_src/MotorControl.o lib/generated_src/TemperatureSensor.o
make: Leaving directory '/home/private/genius/TTool/executablecode'
generated_src/main.c: In function 'main':
generated_src/main.c:73: warning: implicit declaration of function 'activeTracingInConsole'; did you mean 'activeTracingInFile'? [-Wimplicit-function-declaration]
activeTracingInConsole();
^
activeTracingInFile
generated_src/DistanceSensor.c: In function 'mainFunc_DistanceSensor':
generated_src/DistanceSensor.c:14:7: warning: unused variable 'distance' [-Wunused-variable]
int distance = 0;
^
Compilation done
```

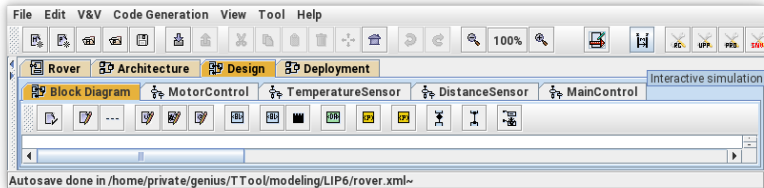

Software Components: Simulation



```
bin:ttool.exe

Make immediate action in blockMainControl
Guard=state != 2
Make immediate action in blockMainControl
s=not(state != 2) b1 = false b2 = false
Replaced ! new s=! (state != 2)
Guard=! (state != 2)
Only one next in state State sendMotorCommand in block MainControl
Only one next in state State dodgeObstacle in block MainControl
Make immediate action in blockMainControl
Guard=distanceLeft > distanceRight
Make immediate action in blockMainControl
s=not(distanceLeft > distanceRight) b1 = false b2 = false
Replaced ! new s=! (distanceLeft > distanceRight)
Guard=! (distanceLeft > distanceRight)
Only one next in state State turnLeft in block MainControl
Action=leftVelocity = 1
Else
Action=rightVelocity = speedLow
Else
Only one next in state State turnRight in block MainControl
Action=rightVelocity = 1
Else
Action=leftVelocity = speedLow
Else
Only one next in state State state1 in block MainControl
Action=state = 1
Else
Action=newRate = rateMed
Else
Action=leftVelocity = speedLow
Else
Action=rightVelocity = speedLow
Else
Only one next in state State state2 in block MainControl
Action=state = 2
Else
Action=newRate = rateHigh
Else
```

Software Components: Interactive Simulation



Software Components: Interactive Simulation

Terminate simulation and quit

Commands

Control **Save trace**

Nb of steps: 1 **x Step-by-Step**

Pending transactions

in Block DistanceSensor: Transition (delay=(1, 2), ...)

Simulation information

Status: **Stopped** Time: 10 Transactions: 12 Coverage: 13.0%

☐ Displayed blocks
 ☐ Latencies
 ☐ Randomness
 ☐ Asynch. msg

Options
☐ Blocks
 ☐ Variables
 ☐ Transactions
 ☐ Met states

☒ Animate UML diagrams
 ☐ Show AVATAR IDs on diagrams

☐ Show hidden state in sequence diag...
 ☒ Auto open active state machines

☒ Trace in sequence diagram
 # of transactions: Index of last trans...

☒ Auto execute empty transitions
 ☒ Auto enter states

UML Sequence Diagram:

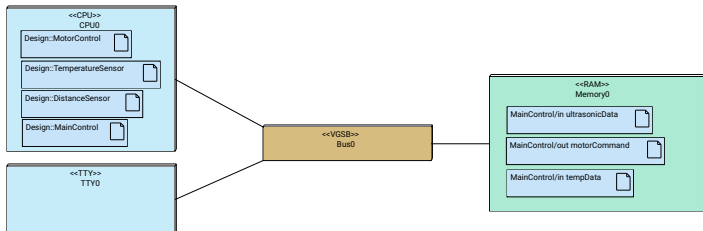
```

sequenceDiagram
    participant MC as MotorControl
    participant TS as TemperatureSensor
    participant DS as DistanceSensor
    participant Main as MainControl

    MC->>MC: startMotor
    TS->>TS: start
    DS->>DS: startSensor
    DS->>DS: 10
    DS->>DS: distanceLeft = 0
  
```

Run simulation for x commands. Works only if the simulator is 'ready'

Software Design: Deployment View





Prototype software components on destination platform

- ▶ Tasks mapped to model of target system
- ▶ Channels mapped to memory
- ▶ Generate software elements (tasks, main program) by model transformation: C Posix
- ▶ Hardware elements built from deployment information

Software Design: CPU configuration

CPU attributes

CPU name:	<input type="text" value="CPU0"/>
Nb Of IRQs :	<input type="text" value="6"/>
Nb of inst. cache ways:	<input type="text" value="8"/>
Nb of inst. cache sets:	<input type="text" value="4"/>
Nb of inst. cache words:	<input type="text" value="4"/>
Nb of data cache ways:	<input type="text" value="8"/>
Nb of data cache sets:	<input type="text" value="4"/>
Nb of data cache words:	<input type="text" value="4"/>
Index:	<input type="text" value="0"/>
Monitored:	<input type="button" value="VCI logger"/>

 Save and Close  Cancel

Virtual Prototype

The screenshot displays the TTool AMS Extension interface, which is a software development environment for system modeling and simulation. The main window shows a block diagram with several blocks, including "TestBlock", "GPSTrans", "FactorSensor", "SpeedSensor", and "CorrectnessChecker". The blocks are interconnected with lines representing data flow.

On the left, a "Braking - MV" window shows a list of simulation requests and responses, such as "Waiting request in inhibit", "No request inhibited -> looking for next", and "Waiting request in inhibit".

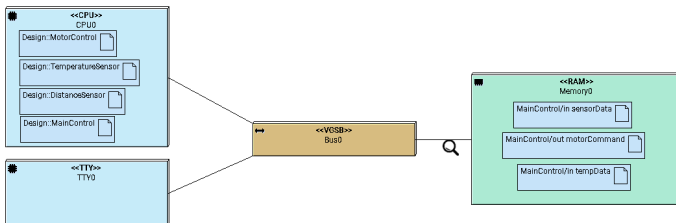
Below the main diagram, a "Simulation trace" window displays a timeline of events. The timeline shows various states and transitions, including "Waiting for environment init()", "Emergency generated()", "Emergency generated()", "Waiting for Emergency message()", "abstractCorrectness", "updateCorrectness()", "setCorrectness()", "DSRC Management", "PlausibilityCheck", "EmergencySensor", and "Waiting for Emergency()".

On the right, an "Executable Code generation, compilation and execution" window is open. It contains tabs for "Generate code", "Compile", "Execute", and "Results". The "Generate code" tab is active, showing a "Simulation trace" and a "Show trace from soclib file" section. The "Show trace from soclib file" section contains a file path: "file (Prog/soclib/soclib/platform/topcells/caba-vgrn-mutech_kernel_tutorial/vehclty".

At the bottom right, a terminal window shows the command prompt and the output of the "SystemCAS" command. The output includes the version number "2.9.2019" and the date "Last change : Dec 6 2011".

Performance Information

- ▶ Intercept traffic on interface between interconnect and memory
- ▶ Logging probes between interconnect, CPU and memories
- ▶ Cycle precise logging (memory access, cache miss, latency, buffer underflow/overflow)
- ▶ Confirm or correct assumptions made on partitioning level





Heterogeneous Extension Required

Embedded hardware is increasingly heterogeneous

- ▶ Digital/analog integrated circuits, sensors, actuators
- ▶ Robotics, automotive, medical

New Features

- ▶ Analog/mixed signal (AMS)
- ▶ Radio frequency (RF)

Increasing role of software

- ▶ Operating system, CPU, memory



Objective

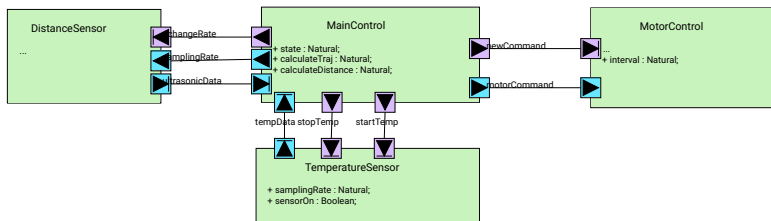
Extend functionality of TTool in order to generate virtual prototypes for embedded systems

- ▶ Composed of digital and analog hardware
- ▶ Able to run embedded software

How?

- ▶ Integrate the TDF models of AMS components with a digital MPSoC platform based on SoCLib components
- ▶ Time synchronization issues that may occur between the DE and TDF MoCs: implement solution at design level

Rover Revisited: Functional Model



- ⇒ Modeling of sensors not realistic (neither on partitioning level)
- ⇒ Requires better adapted representation to reflect analog/mixed signal components



Related Work

Non SystemC based

- ▶ Ptolemy II (Ptolemy.org 2014)
- ▶ METROPOLIS (Balarin et al. 2003)
- ▶ METRO II (Davare et al. 2007)
- ▶ Discrete Event System Specification (Zeigler 1976)
- ▶ Modelica (Elmqvist et al. 1999)

SystemC based

- ▶ HetSC (Herrera et al. 2007)
- ▶ HetMoC (Zhu et al. 2010)
- ▶ ForSyDe (Niaki et al. 2012)



SystemC AMS Extensions

- ▶ Standard describing an extension of SystemC with AMS and RF features (2010 v1.0, 2013 v2.0)
- ▶ Modeling formalisms:
 - ▶ Discrete Event (DE)
 - ▶ Timed Data Flow (TDF)
 - ▶ Linear Signal Flow (LSF)
 - ▶ Electrical Linear Networks (ELN)
 - ▶ Consortium includes Fraunhofer IIS, TU Vienna, EPFL, NXP, Infineon, LIP6

LIP6 in projects BeyondDreams, H-Inception



SystemC AMS Extensions

Building a modeling environment synchronizing DE and TDF is hard

- ▶ Choice of simulation method(s)
- ▶ Causality problems

Work at LIP6

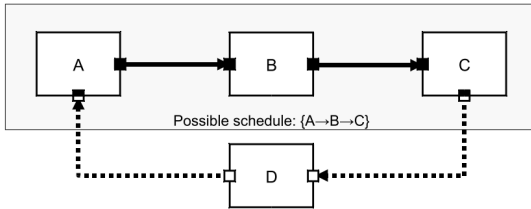
- ▶ Andrade et al. 2015
- ▶ Ben Aoun 2017
- ▶ Cortés Porto 2018



Discrete Event (DE) Model of Computation

- ▶ Based on temporal sequences of countable number of events
- ▶ Processes
 - ▶ Describe system behavior
 - ▶ Triggered by events or passing of time
- ▶ Events are sorted wrt. time stamps into event queue
- ▶ Scheduler determines process to execute at run time: dynamic scheduling
- ▶ SystemC simulation kernel based on DE MoC

Time Data Flow (TDF) Model of Computation



- ▶ Discrete-time model, continuous data as signals sampled in time
- ▶ Ports to connect TDF modules
- ▶ Converter ports to connect TDF to DE modules
- ▶ Static schedule computed before simulation



TDF module properties

- ▶ Module Timestep (T_m): Period of activation
- ▶ Port Timestep (T_p): Period of activation of a modules port
- ▶ Rate (R): Number of data samples read or write, annotated to a port
- ▶ Delay (D): Number of samples to be held per activation of the module
- ▶ Consistency of time step assignment and propagation
 - ▶ $T_m = T_{p_{in}} * R_{in} = T_{p_{out}} * R_{out}$



Validation

- ▶ Validation of TDF module and port attributes
- ▶ Timestep propagation: consistency check
- ▶ Computation of valid schedule



Static Schedule Computation

- ▶ Based on sequential scheduling algorithm (Lee, Messerschmitt, 1987)
- ▶ Deadlock due to feedback loops: suggest port delays to solve it (Cortés Porto RAPIDO 2019)
- ▶ Still without software, runs stand-alone (without SoCLib) with SystemC-AMS

Installation of TTool and SystemC-AMS



- ▶ Download TTool from <https://ttool.telecom-paristech.fr/download.html>
- ▶ Requires jdk 1.8 or higher

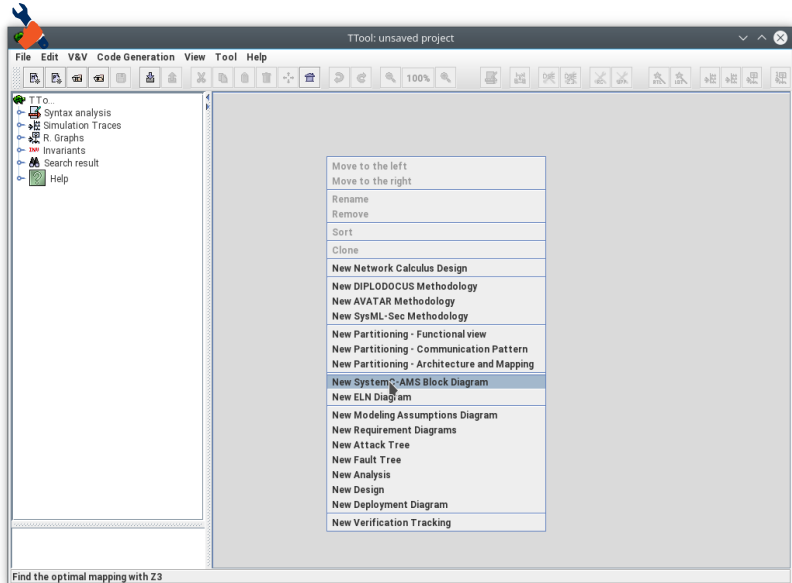
SystemC-AMS (necessary for compiling generated code)

- ▶ Download SystemC-AMS from Accellera website www.accellera.org
- ▶ Install SystemC-AMS

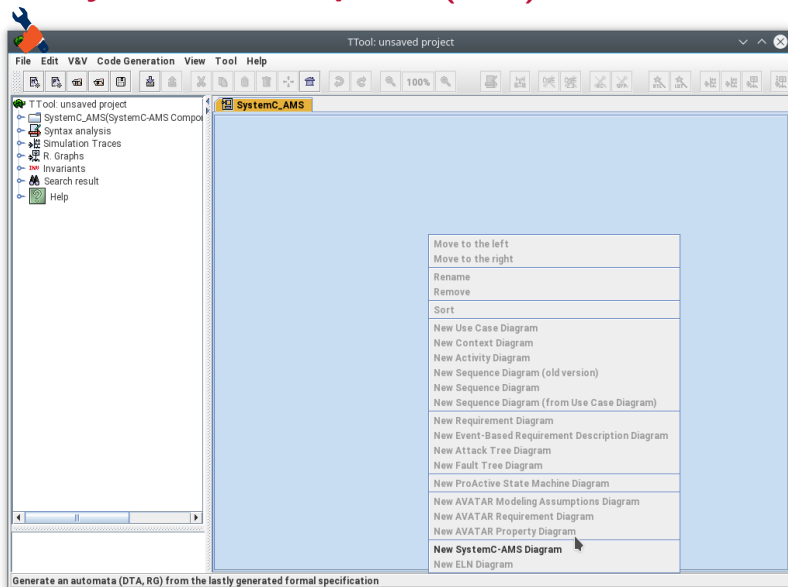
Environment

- ▶ Download `systemc-env.sh` from <ftp://ftp-asim.lip6.fr/outgoing/genius> (login: ftp, no password)
- ▶ `source ./systemc-env.sh`

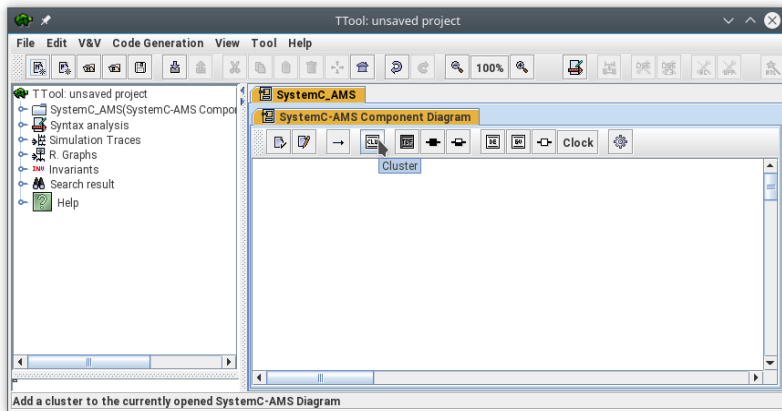
SystemC-AMS panel



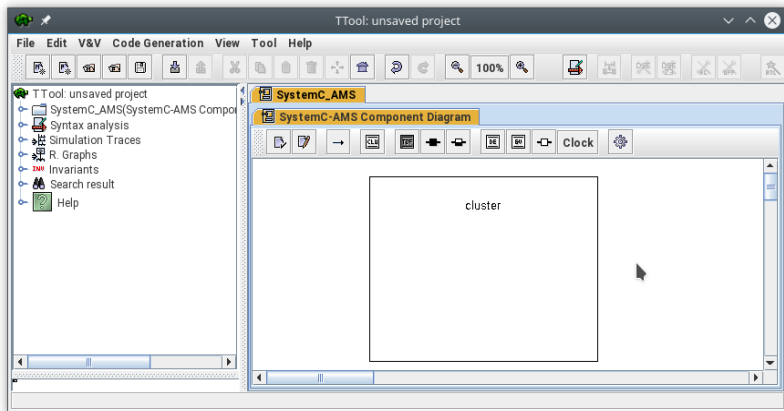
SystemC-AMS panel (ctd.)



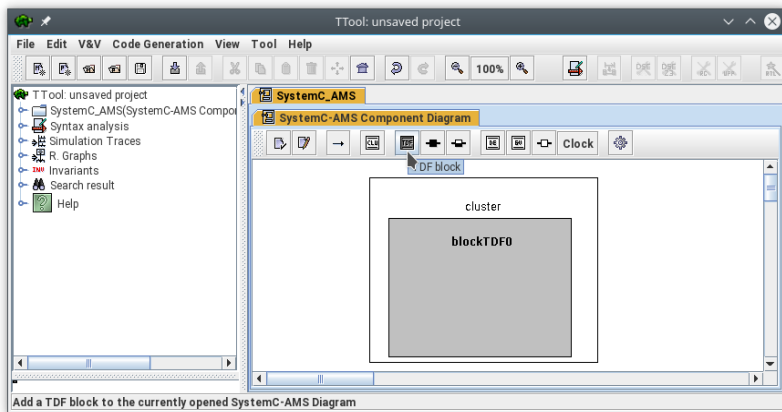
Inserting a Cluster



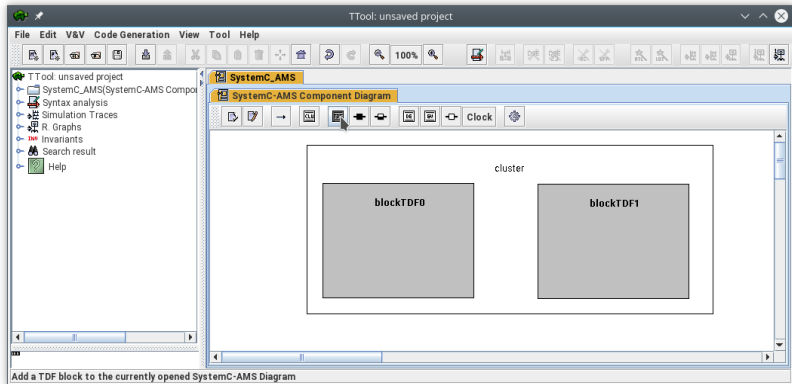
Inserting a Cluster (ctd.)



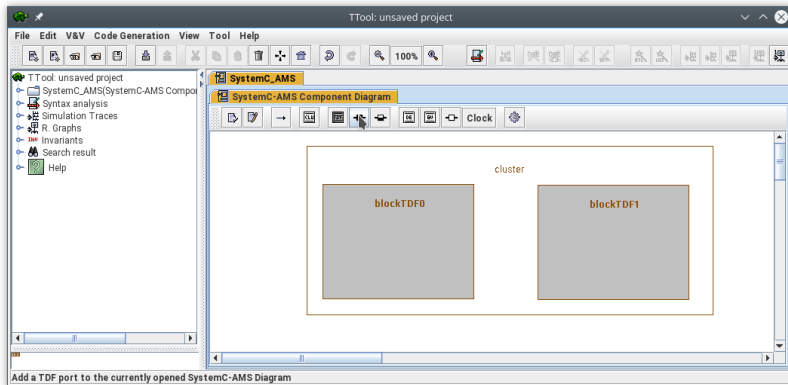
Inserting a Block



Inserting a Block (ctd.)



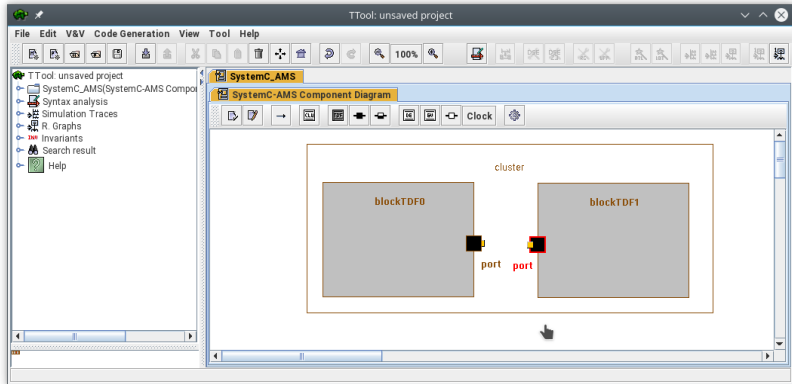
Configuring a Port



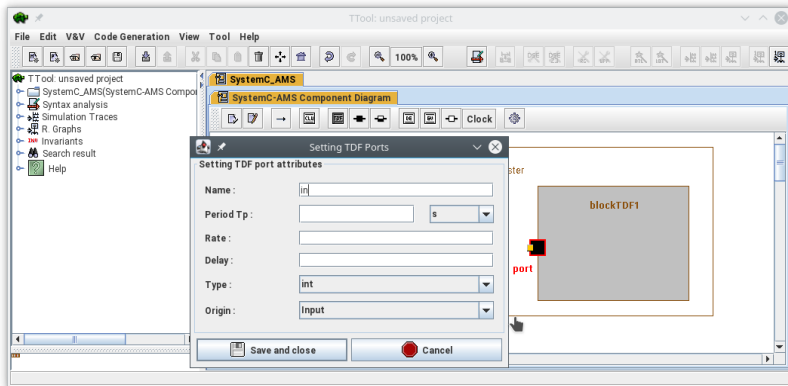
Configuring a Port (ctd.)



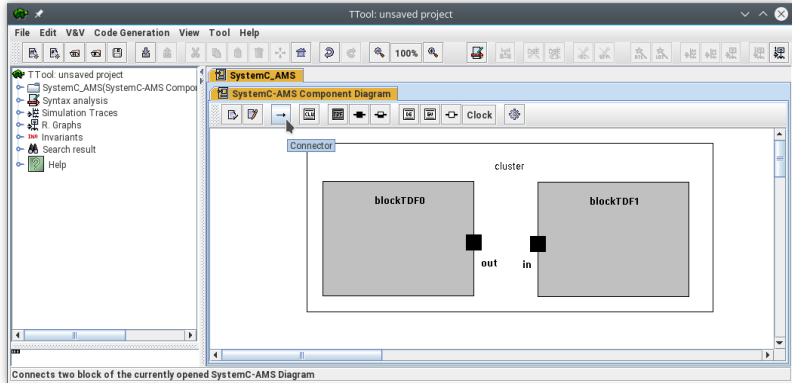
SystemC-AMS panel



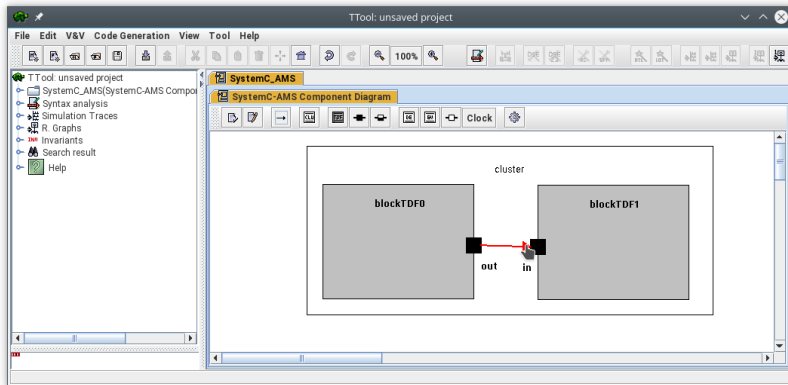
Configuring a Port (ctd.)



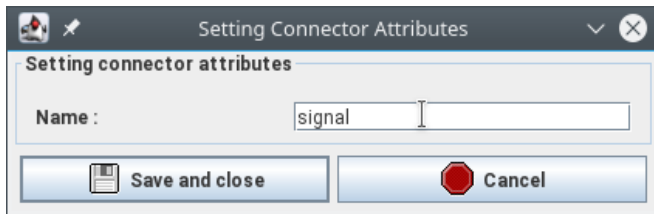
Inserting a Connector



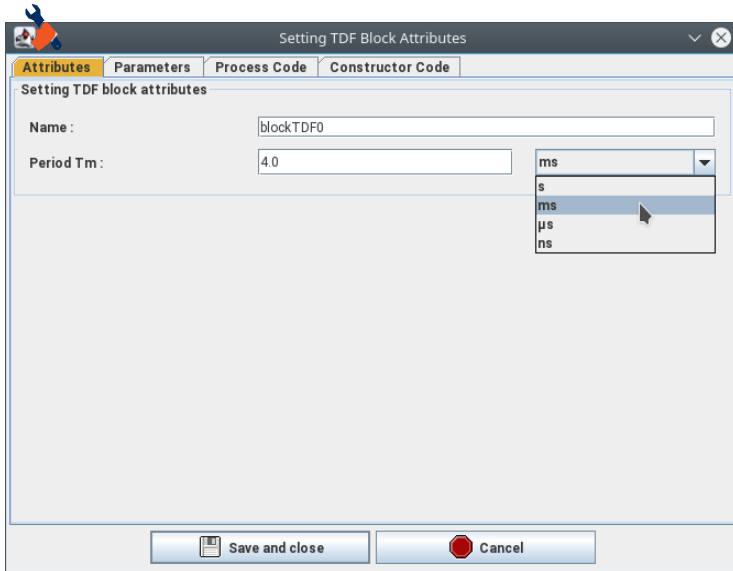
Inserting a Connector (ctd.)



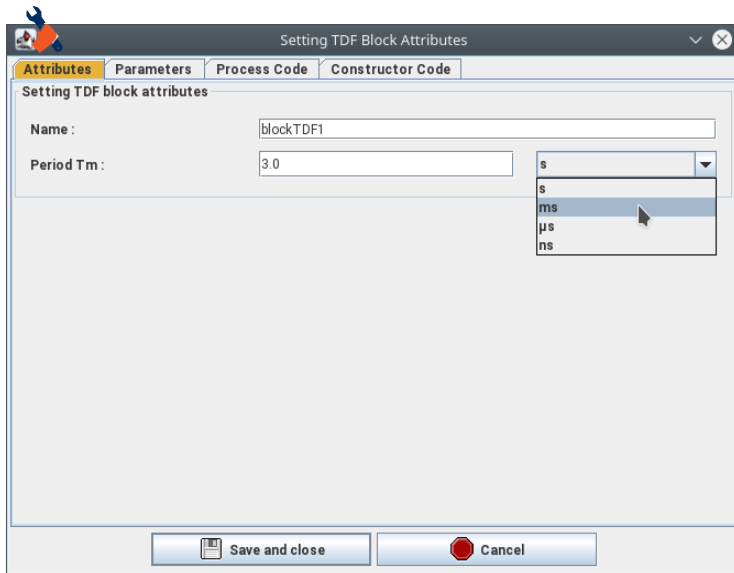
Naming a Connector



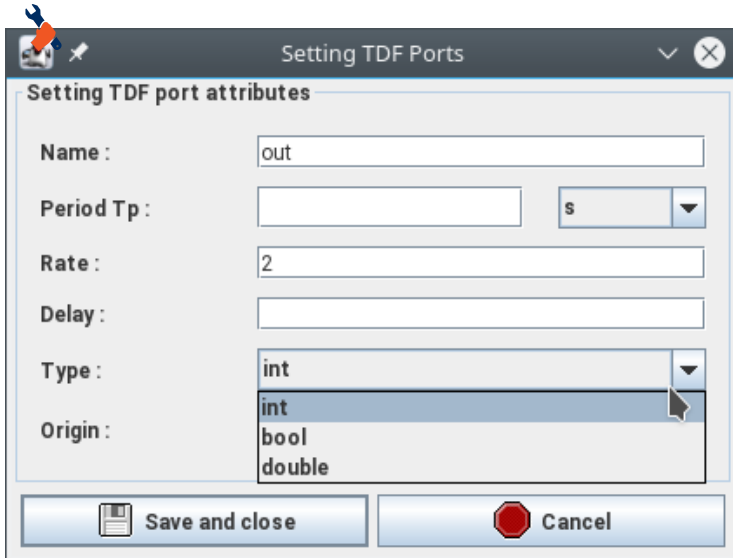
Parametrization of module attributes



Parametrization of module attributes (contd.)



Parametrization of port attributes



The image shows a screenshot of a software dialog box titled "Setting TDF Ports". The dialog box has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. Below the title bar, the main content area is titled "Setting TDF port attributes". It contains several input fields and a dropdown menu. The "Name" field is labeled "Name :" and contains the text "out". The "Period Tp" field is labeled "Period Tp :" and is empty, with a unit selector dropdown set to "s". The "Rate" field is labeled "Rate :" and contains the value "2". The "Delay" field is labeled "Delay :" and is empty. The "Type" field is labeled "Type :" and has a dropdown menu open, showing options "int", "bool", and "double". The "Origin" field is labeled "Origin :" and is empty. At the bottom of the dialog box, there are two buttons: "Save and close" and "Cancel".

Setting TDF Ports

Setting TDF port attributes

Name : out

Period Tp : s

Rate : 2

Delay :

Type : int

Origin :

int
bool
double

Save and close Cancel

Parametrization of port attributes (contd.)

Setting TDF Ports

Setting TDF port attributes

Name : out

Period Tp : \$

Rate : 2

Delay :

Type : int

Origin :

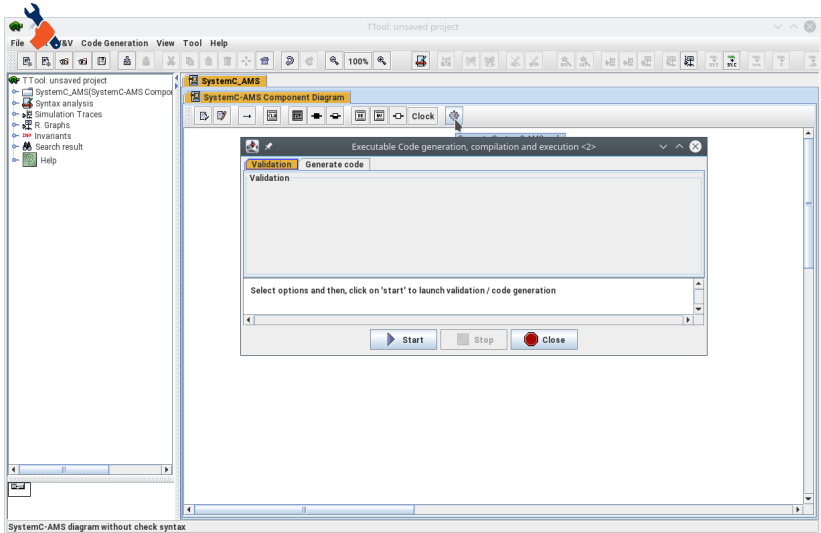
Input

Output

Save and close Cancel

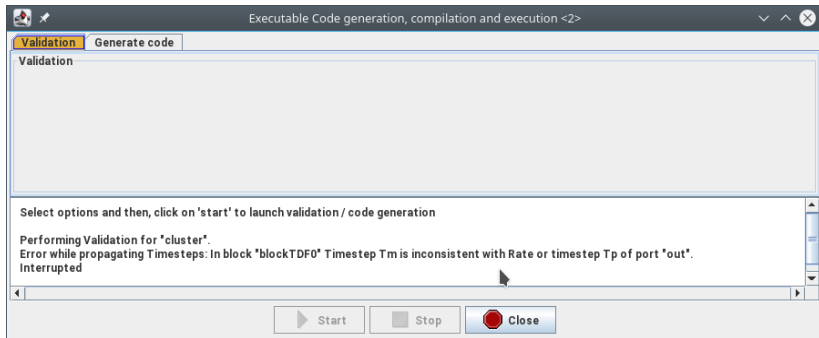
ToDo: Run schedulability test

First schedulability test



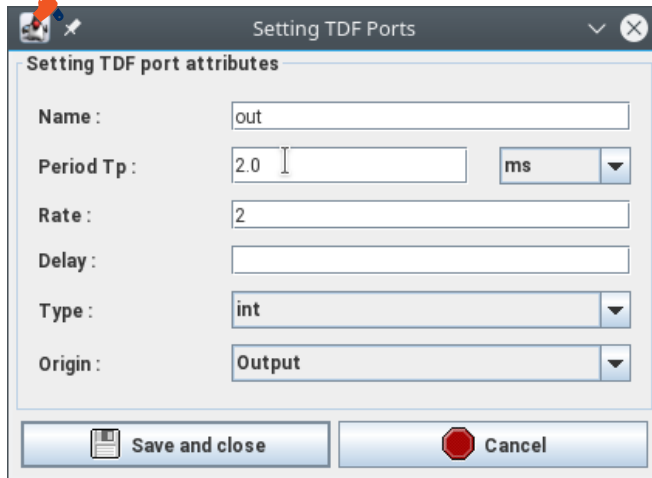

ToDo: Correct parameters

First schedulability test (contd.)



ToDo: Correct parameters

Correcting Parameters



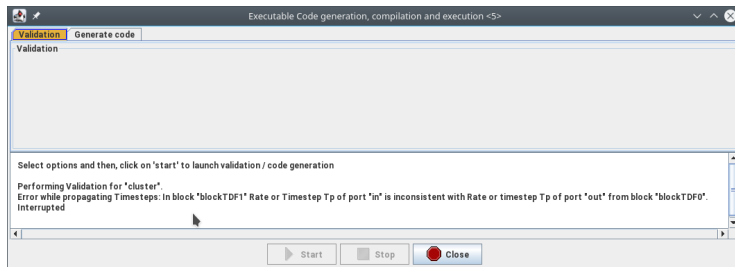
The dialog box titled "Setting TDF Ports" contains the following fields and controls:

- Name :** A text input field containing the value "out".
- Period Tp :** A text input field containing the value "2.0", followed by a unit selector dropdown menu currently set to "ms".
- Rate :** A text input field containing the value "2".
- Delay :** An empty text input field.
- Type :** A dropdown menu currently set to "int".
- Origin :** A dropdown menu currently set to "Output".

At the bottom of the dialog, there are two buttons: "Save and close" (with a floppy disk icon) and "Cancel" (with a red octagonal stop sign icon).

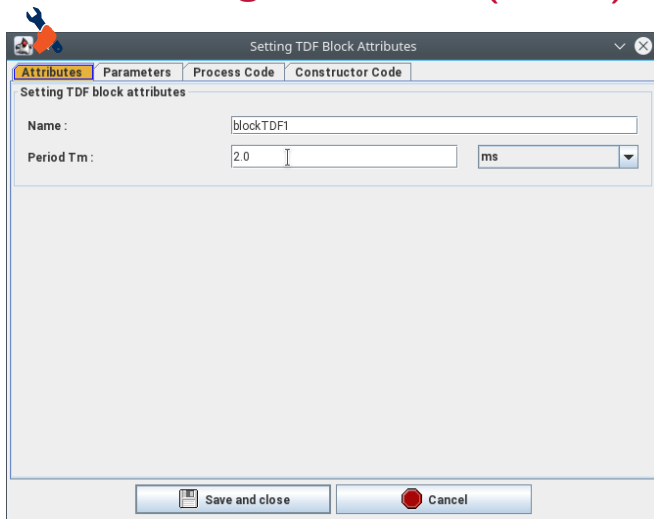
ToDo: Correct parameters

Correcting Parameters (contd.)



ToDo: Correct parameters

Correcting Parameters (contd.)



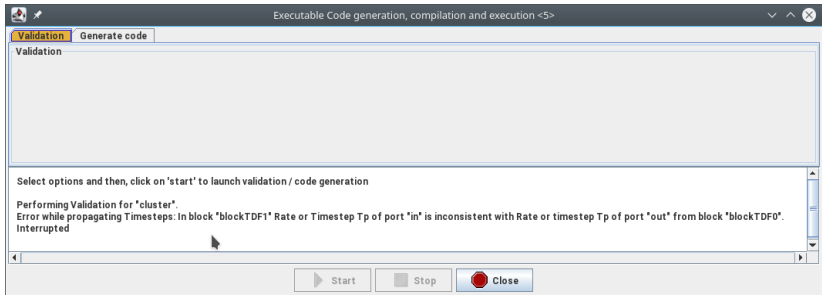
The image shows a software dialog box titled "Setting TDF Block Attributes". It has four tabs: "Attributes" (selected), "Parameters", "Process Code", and "Constructor Code". The "Attributes" tab contains the following fields:

- Name :** A text box containing "blockTDF1".
- Period Tm :** A text box containing "2.0" and a unit dropdown menu set to "ms".

At the bottom of the dialog, there are two buttons: "Save and close" (with a floppy disk icon) and "Cancel" (with a red stop icon).

ToDo: Correct parameters

Correcting Parameters (contd.)



ToDo: Validate current TDF cluster



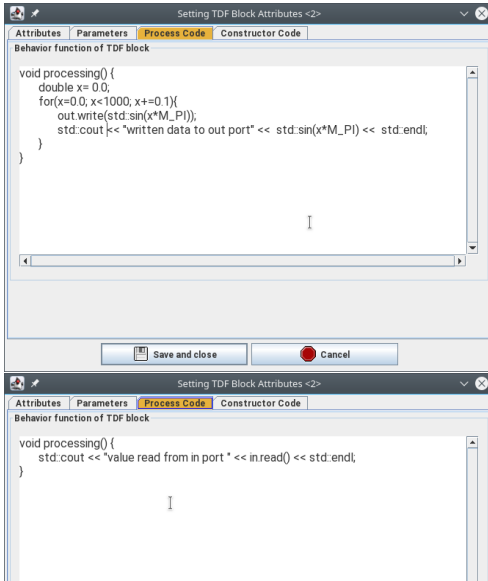
Adding a Processing Function

- ▶ Analog components are most often unique
- ▶ Hypothesis: *processing()* function given manually

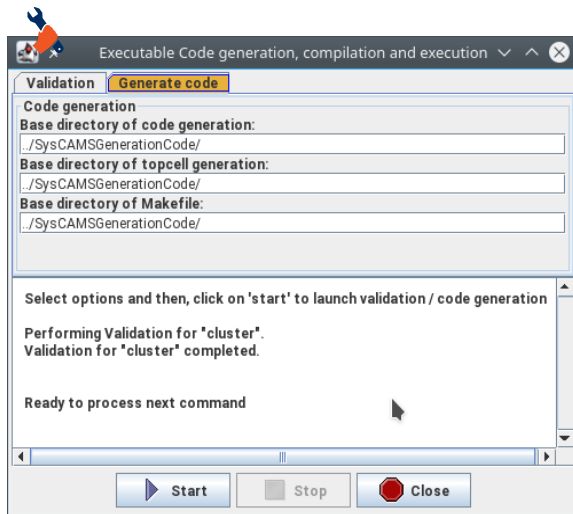
ToDo: Write a processing function which writes to output port/read from input port

Example: sine function (change port type to double)

Adding a Processing Function: Example

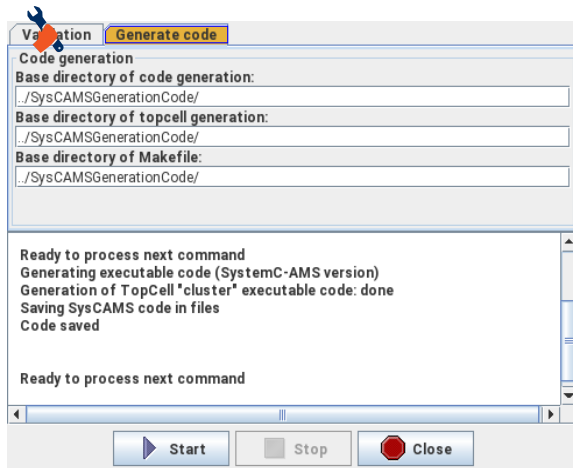


Validation



ToDo: Generate SystemC-AMS code

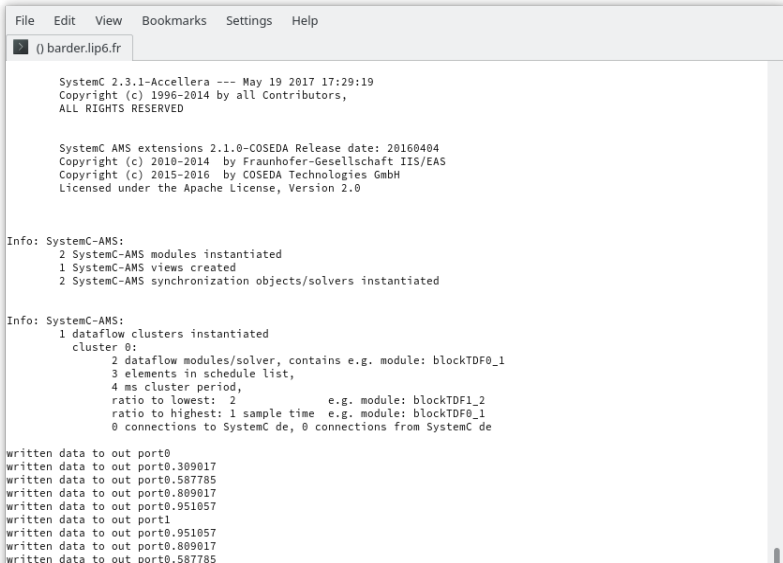
Code Generation



ToDo: Have a look at generated code in your TTool/SysCAMSGenerationCode directory

ToDo: Run the simulation

Trace File



```
File Edit View Bookmarks Settings Help
() border.lip6.fr

SystemC 2.3.1-Accellera --- May 19 2017 17:29:19
Copyright (c) 1996-2014 by all Contributors,
ALL RIGHTS RESERVED

SystemC AMS extensions 2.1.0-COSEDa Release date: 20160404
Copyright (c) 2010-2014 by Fraunhofer-Gesellschaft IIS/EAS
Copyright (c) 2015-2016 by COSEDa Technologies GmbH
Licensed under the Apache License, Version 2.0

Info: SystemC-Ams:
  2 SystemC-Ams modules instantiated
  1 SystemC-Ams views created
  2 SystemC-Ams synchronization objects/solvers instantiated

Info: SystemC-Ams:
  1 dataflow clusters instantiated
    cluster 0:
      2 dataflow modules/solver, contains e.g. module: blockTDF0_1
      3 elements in schedule list,
      4 ms cluster period,
      ratio to lowest: 2 e.g. module: blockTDF1_2
      ratio to highest: 1 sample time e.g. module: blockTDF0_1
      0 connections to SystemC de, 0 connections from SystemC de

written data to out port0
written data to out port0.309017
written data to out port0.587785
written data to out port0.809017
written data to out port0.951057
written data to out port1
written data to out port0.951057
written data to out port0.809017
written data to out port0.587785
```



Co-Simulation

Problem: Simulation of TDF and DE parts

- ▶ Simulation in two (or more) different domains

Idea: SoCLib components can be (re)used

- ▶ SystemC is a set of C++ class libraries
- ▶ Co-simulation (SystemC-AMS and SystemC simulation motor)
- ▶ SoCLib digital components are considered DE modules

What is new in our approach?

- ▶ Validation of schedule and causality issues **before** code generation (Cortés Porto 2018)
- ▶ Full-system simulation (SW running under OS on digital part)

Virtual Prototyping with SoCLib

- ▶ SoCLib: Open platform for virtual prototyping of multi-processors System on Chip (MP-SoC)
- ▶ Started in 2006 as an ANR project with 11 research labs and 6 industrial partners: LIP6, Bull, CEA-LETI, INRIA, STMicro, ...
- ▶ Public domain library of SystemC models of hardware components
- ▶ Operating systems: NetBSD, MutekH, GIET, ALMOS, ...
- ▶ Different levels of modeling and simulation
 - ▶ Transaction Level (TLM)
 - ▶ Transaction Level with Time (TLM-T)
 - ▶ Cycle Accurate Bit Accurate (CABA)





MP-SoC Virtual Prototype Generation

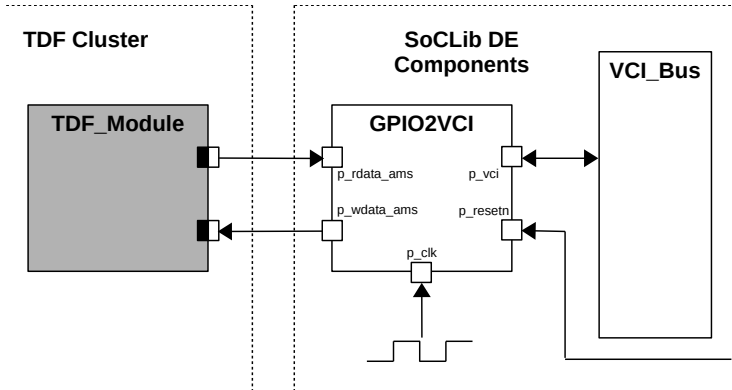
- ▶ SoCLib: Shared memory paradigm
 - ▶ Based on Virtual Component Interconnect (VCI) protocol
 - ▶ Initiators issue requests (e.g. CPUs)
 - ▶ Targets respond to requests (e.g. RAM)

Specificity of our approach to co-simulation

- ▶ Hypothesis: (Digital) MP-SoC is initiator
- ▶ TDF clusters are VCI targets for the SoCLib initiators

Generic adaptor component (RAPIDO2019)

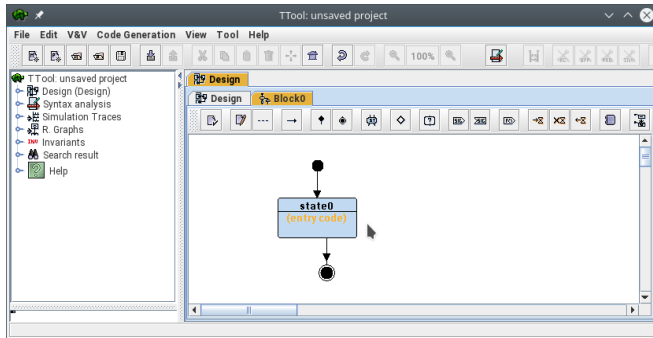
- ▶ Interface between TDF clusters and SoCLib components
- ▶ Follows SoCLib's writing rules for cycle-accurate components
- ▶ Represented in Deployment Diagram



Embedded SW design and development



- ▶ State Machine Diagrams:
 - ▶ Functionality of software tasks as timed automates
 - ▶ Generated code this way is correct-by-construction, but:
 - ▶ Code entered manually into the state blocks





Installation from provided Virtual Machine (Required for Tutorial 2)



- ▶ Download and install VirtualBox for your operating system from <https://www.virtualbox.org/>
- ▶ Download the TTool virtual machine file from <ftp://ftp-asim.lip6.fr/outgoing/genius/TToolVM.tgz>
Attention, file size 5835180 KB
- ▶ Unpack the archive
- ▶ Start VirtualBox and import the provided TTool virtual machine:
 - ▶ Select menu *File Import Appliance*
 - ▶ Browse to the downloaded .ova file
 - ▶ Follow the wizard using the default settings
 - ▶ Once the import is finished, start the machine (password = *TToolVM*)

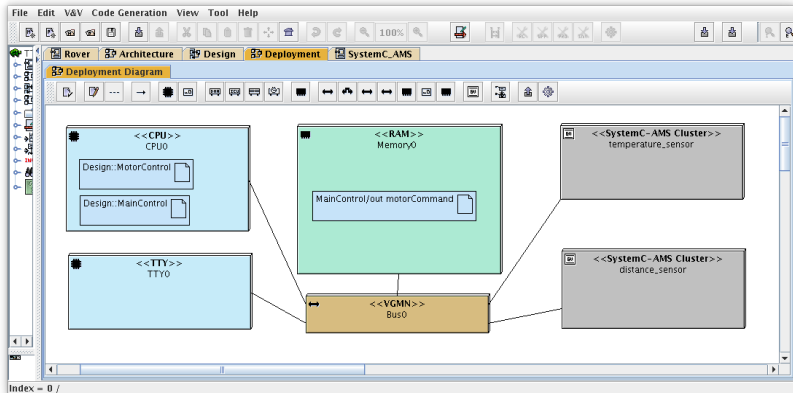


Installation from provided Virtual Machine (Required for Tutorial 2)

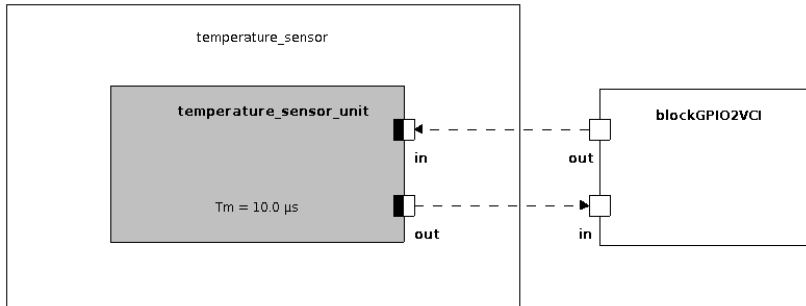


- Download the tutorial example from
`ftp:://ftp-asim.lip6.fr/outgoing/genius/rover_ams.xml`

Deployment Diagram with GPIOs (gray)

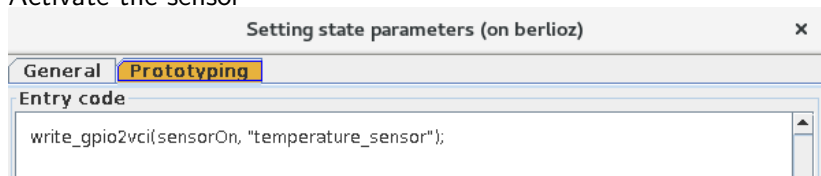


Temperature Sensor

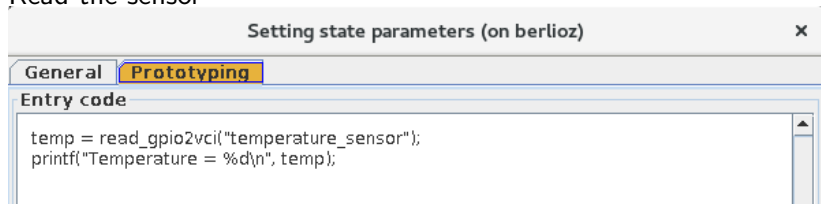


Temperature Sensor

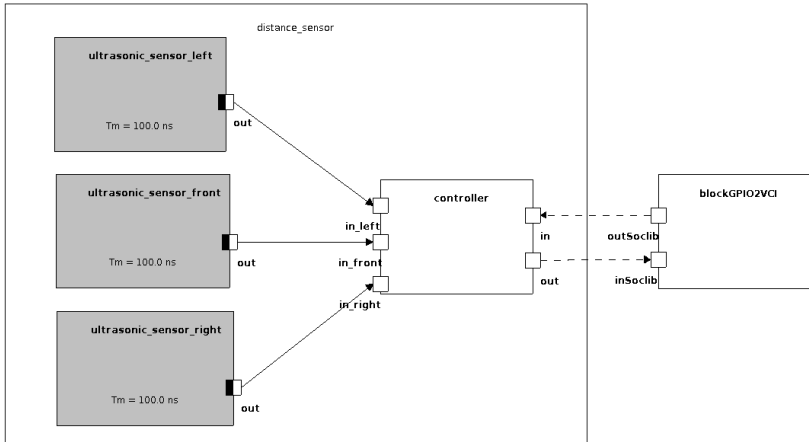
Activate the sensor



Read the sensor



Distance Sensor



Distance Sensor



Setting state parameters (on berlioz) x

General Prototyping

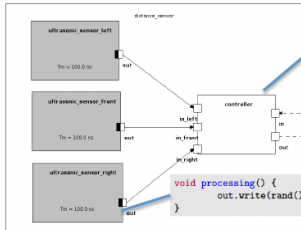
Entry code

```
write_gpio2vci(1, "distance_sensor");
distanceFront = read_gpio2vci("distance_sensor");
printf("distanceFront = %d\n", distanceFront);

write_gpio2vci(0, "distance_sensor");
distanceLeft = read_gpio2vci("distance_sensor");
printf("distanceLeft = %d\n", distanceLeft);

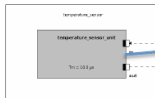
write_gpio2vci(2, "distance_sensor");
distanceRight = read_gpio2vci("distance_sensor");
printf("distanceRight = %d\n", distanceRight);
```

Code and Processing Function



```
void read_sensor() {
    if(in.read() == 0) {
        out.write(in_left.read());
    }
    else if(in.read() == 1) {
        out.write(in_front.read());
    }
    else if(in.read() == 2) {
        out.write(in_right.read());
    }
}
```

```
void processing() {
    out.write(rand() % 12);
}
```

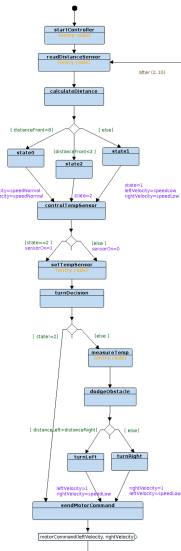


```
void processing() {
    if(in.read() != 0) {
        out.write(rand() % 30);
    }
    else {
        cout << "Temp sensor is off. @ " << this->get_time() << endl;
    }
}
```

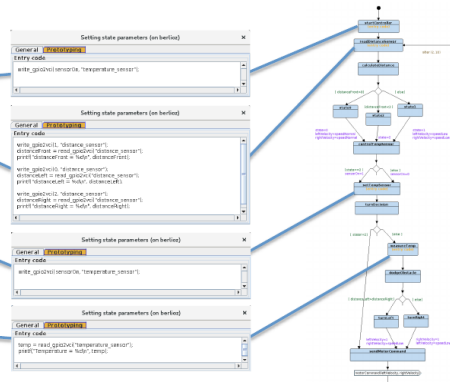
Embedded SW Design and Development

Entry Code in the Control State Machine

- ▶ State machine slightly simplified
 - ▶ Less channels to read/write
 - ▶ SystemC-AMS current version does not allow rate change at run time

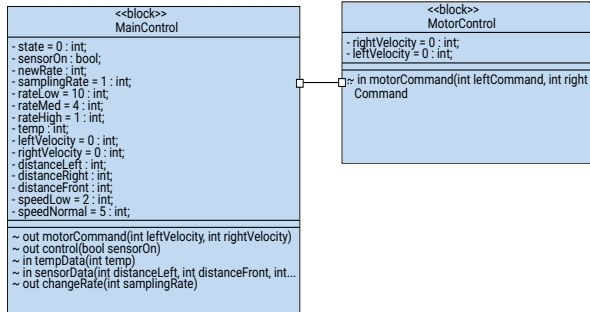


Entry code

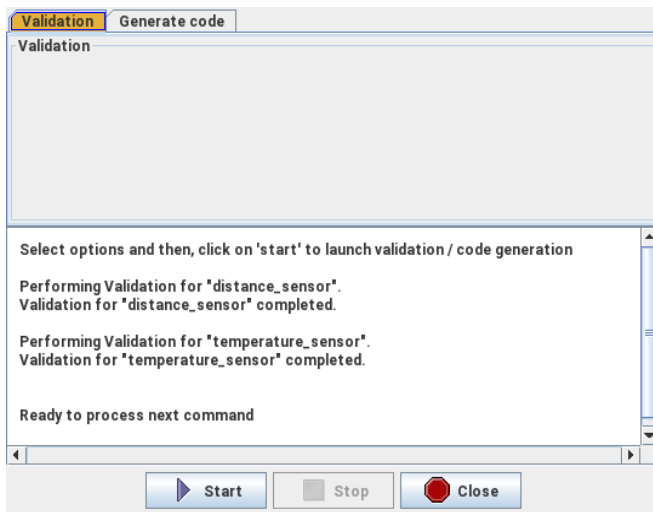
[illegible]

Block Diagram

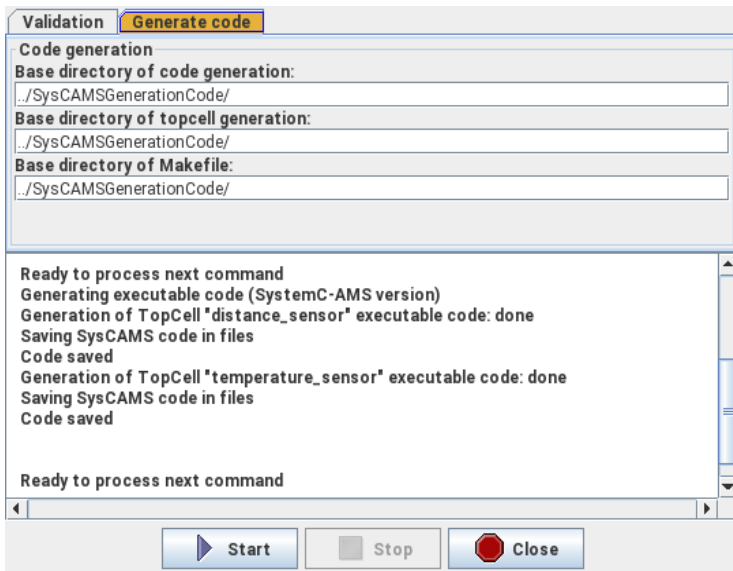
Sensors are no longer represented



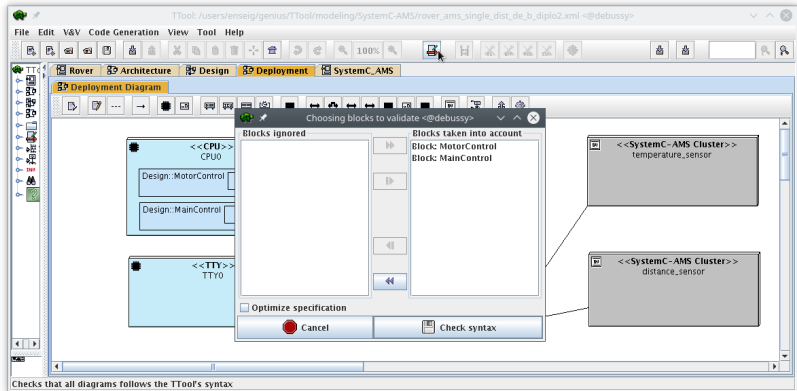
Validation of Schedulability and Causality



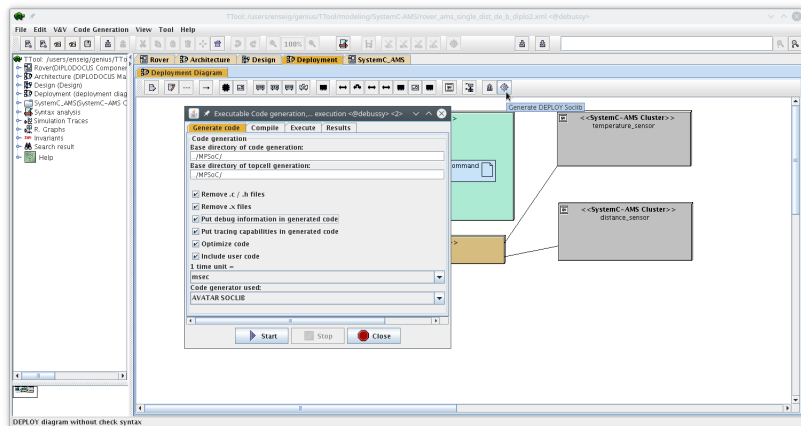
SystemC-AMS Code Generation



Syntax Analysis

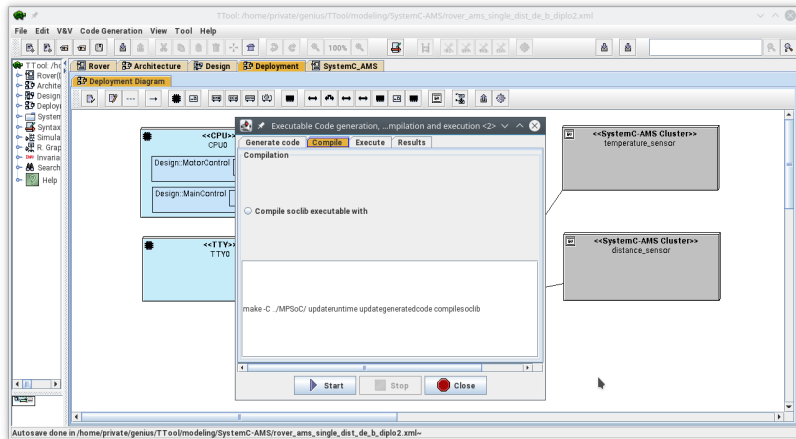


Code Generation (SoCLib)



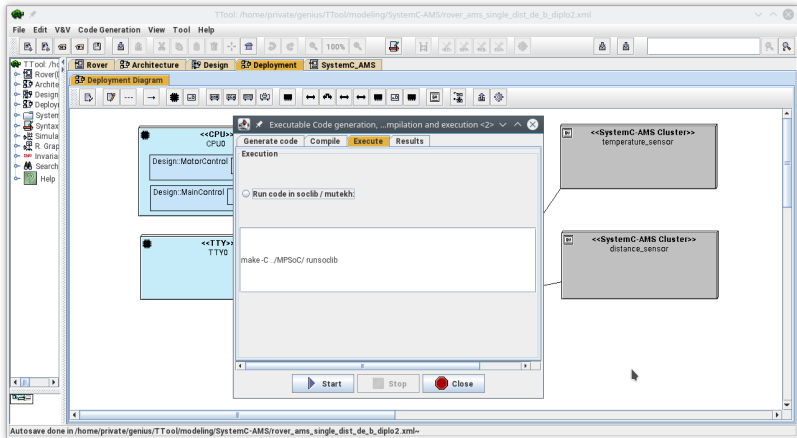
Compilation (Application code)

Requires cross-compiler (e.g. for MIPS32, ARM, ...)



Compilation (SoCLib Platform)

Source the configuration file `systemc-env.sh` (in `ttool/bin`)



In TTool/MPSoC Make `compilesoclib`
Make `allsoclib`

Simulation: Reading the Distance

```
vci_multi_tty0 (on berlioz)
```

```
DT> addToRequestQueue
DT - MotorControl -> Waiting for request!
DT - MotorControl -> Releasing mutex
DT - MainControl -> -> (====) Entering state + startController
DT - MainControl -> -> (====) Entering state + readDistanceSensor
distanceFront = 2
distanceLeft = 7
distanceRight = 9
DT - MainControl -> -> (====) Entering state + calculateDistance
DT - MainControl -> -> (====) Entering state + choice__0
DT - MainControl -> Locking mutex
DT - MainControl -> Mutex locked
DT - MainControl -> Going to execute request
DT - MainControl -> No request selected -> looking for one!
```

Simulation: Measure the Temperature

```
vci_multi_tty0 (on berlioz)

DT - MainControl -> -> (=====) Entering state + setTempSensor
DT - MainControl -> -> (=====) Entering state + turnDecision
DT - MainControl -> -> (=====) Entering state + choice__2
DT - MainControl -> Locking mutex
DT - MainControl -> Mutex locked
DT - MainControl -> Going to execute request
DT - MainControl -> No request selected -> looking for one!
DT> Starting loop
DT> immediate
DT - MainControl -> counting executable requests
DT> Counting requests=: 1
DT> At least one pending request is executable: 1
DT> selectedIndex=: 1
DT> Getting request at index: 0
DT> Selected request of type: a
DT> Removing original req
DT> Considering request of type: a
DT - MainControl -> Request selected!
DT - MainControl -> Mutex unlocked
DT - MainControl -> -> (=====) Entering state + measureTemp
Temperature = 21
DT - MainControl -> -> (=====) Entering state + dodgeObstacle
DT - MainControl -> -> (=====) Entering state + choice__3
DT - MainControl -> Locking mutex
```



Conclusion

- ▶ Generation of standalone SystemC AMS code from SysML-like diagrams
- ▶ Successful integration of SystemC-AMS and SoCLib modules in order to run Software under a (lightweight) Operating System
- ▶ TDF models validated at design level before code generation
- ▶ Rover: require Rate change at run time in TDF blocks



Future Work

- ▶ Creation of specific state blocks for communication
- ▶ Advanced methods for solution of synchronization issues
- ▶ Modeling “Real-world” applications (EchOpen project) necessitates extensions
- ▶ Library of parametrizable SystemC-AMS blocks
- ▶ Feedback of simulation results (latencies, cache misses etc.) to higher modeling levels
- ▶ Design Space Exploration, with automatically suggested modifications
- ▶ More detailed performance profiles (cache misses, latencies, processor workload, buffer fill state etc.)



Links

TTool: ttool.telecom-paristech.fr

SoCLib: www.soclib.fr

Accellera Systems Initiative: www.accellera.org

References



Genius, D., Li, L. W., Apvrille, L.: Model-Driven Performance Evaluation and Formal Verification for Multi-level Embedded System Design, Model-Driven Engineering and SW Development, Porto, Portugal (2017)



Genius, D., Li, L. W., Apvrille, L.: Multi-level Latency Evaluation with an MDE Approach, Model-Driven Engineering and Software Development, Funchal, Portugal (2018)



Cortés Porto, R., Genius, D., Apvrille, L.: "Modeling and Virtual Prototyping for Embedded Systems on Mixed-Signal Multicores." Proceedings of Rapid Simulation and Performance Evaluation: Methods and Tools, ACM (2019)



Genius, D., Cortés Porto, R., Apvrille, L.: A Tool for High-Level Modeling of Analog/Mixed Signal Embedded Systems. In: 7th International Conference on Model-Driven Engineering and Software Development (2019)