System-Level Design for Communication-Centric Task Farm Applications

Daniela Genius, Ludovic Apvrille
daniela.genius@lip6.fr
ludovic.apvrille@telecom-paristech.fr

ReCoSoC 2017
Context

Model-oriented Design of Complex Embedded Systems

- Frequently used for the **software development** of embedded systems
- Complex MPSoC architecture with interconnect-on-chip
  - Architecture might be hierarchical

Communication-centric Applications

- Examples: telecommunication, video streaming
- Not natively supported by MDE approaches
- Our goal: **analysis of critical performance properties** such as contention on the interconnect, cache effects, ...
Related Work

Virtual Prototyping

- Kumar/Hansson/Huisken/Corporaal 2007: FPGA
- Ptolemy II, METROPOLIS, SESAME, ARTEMIS
- SoCLib: SystemC based, no graphical front-end

System-Level Design

- Batori/Theisz/Asztalo 2007
- DiNatale/Chirico/Sindico/Sangiovanni-Vincentelli 2014
- Pedroza/Knorreck/Apvrille 2011: AVATAR
AVATAR

Automated Verification of reAl Time softwARe

- SysML-based environment
- Analysis and design of embedded software
- Safety and security properties modeling and verification
- Fully supported by the free software TTool
  - Edition of AVATAR models
  - Simulation and formal verification (Internal tools, UPPAAL, ProVerif)
  - Timed functional model
    - Hardware targets are ignored (CPUs, buses, etc.)
SoCLib and MutekH

Hardware Simulation Platform based on SoCLib

- Virtual prototyping of complex Systems-on-Chip
- Supports several models of processors, buses, memories
  - Example of CPUs: MIPS, ARM, SPARC, Nios2, PowerPC
- Two simulation models:
  - TLM = Transaction Level Modeling
  - CABA = Cycle Accurate Bit Accurate

Embedded Operating System: MutekH

- Natively handles heterogeneous multiprocessor platforms
- POSIX thread support
AVATAR

Model of SW components

Functional simulation and formal verification

Diagram edition, model transformation, model backtracing

C/POSIX code

C/POSIX code

C/POSIX code

Execution on local host

CABA simulation of HW and SW

Execution on target

Model of SW components

Deployment Diagram: HW architecture and mapping information
Multi-writer Multi-reader Paradigm
[Faure/GreinerGenius/2006]

- Any number of reader or writer tasks can access the channel
- Reader or writer tasks can be hardware or software tasks
- Implemented as software channels in shared memory
- Blocking or non-blocking read and write primitives (read/try read, write/try write)
▶ Task farm
▶ 8-byte descriptor
▶ bootstrap task
▶ input task
▶ classification task
▶ scheduling task
▶ output task
AVATAR Communication Channels

Asynchronous Semantics

- Blocking on read
- Blocking or non-blocking on write
- Lossy

Synchronous Semantics

- Difficult to implement for MPSoC, require central management
- broadcast (not implemented)
- private channel, security features (not implemented)
AVATAR Communication Channels (contd.)
AVATAR Communication Channels: Non-determism
Extending the Semantics

How to ensure the Task Farm property?

- Several readers/writers can access the channel in a non-deterministic fashion (provided by AVATAR semantics)
- Asynchronous AVATAR semantics
- Use hierarchical blocks to regroup channels
- Priority queues have \textit{non-blocking} read and write operations: \textit{try-read} and \textit{try-write}
AVATAR Model: Block Diagram

- PacketDesc
  - address : int;
  - date : int;
  - internal : bool;
  - TotalSize : int;

- Classification
  - packet : PacketDesc;
  - out queue_low(PacketDesc packet)
  - out queue_medium(PacketDesc packet)
  - out queue_high(PacketDesc packet)
  - in c0_to_queue_low(PacketDesc packet)
  - in c1_to_queue_low(PacketDesc packet)
  - in c2_to_queue_low(PacketDesc packet)
  - in c0_to_queue_medium(PacketDesc packet)
  - in c1_to_queue_medium(PacketDesc packet)
  - in c2_to_queue_medium(PacketDesc packet)
  - in c0_to_queue_high(PacketDesc packet)
  - in c1_to_queue_high(PacketDesc packet)
  - in c2_to_queue_high(PacketDesc packet)

- Scheduling
  - packet : PacketDesc;
  - in from_queue_low(PacketDesc packet)
  - in from_queue_medium(PacketDesc packet)
  - in from_queue_high(PacketDesc packet)
  - out to_scheduler0(PacketDesc packet)
  - out to_scheduler1(PacketDesc packet)

- Classif0
  - packet : PacketDesc;
  - out to_queue_low(PacketDesc packet)
  - out to_queue_medium(PacketDesc packet)
  - out to_queue_high(PacketDesc packet)

- Classif1
  - packet : PacketDesc;
  - out to_queue_low(PacketDesc packet)
  - out to_queue_medium(PacketDesc packet)
  - out to_queue_high(PacketDesc packet)

- Classif2
  - packet : PacketDesc;
  - out to_queue_low(PacketDesc packet)
  - out to_queue_medium(PacketDesc packet)
  - out to_queue_high(PacketDesc packet)
AVATAR Model: Timed Automata

Prototyping

Contribution

Code Generation

Experiments

Conclusion

A VATAR Model: Timed Automata

 Dispatch
to_scheduler1(packet)
to_scheduler0(packet)

from_queue_low(packet)
from_queue_medium(packet)
from_queue_high(packet)

Waiting

Low

Medium

High

queue_low(packet)

queue_medium(packet)

queue_high(packet)
AVATAR Model: Deployment Diagram
1. **Libavatar** Runtime for SoCLib, implements AVATAR operators

2. **DDSyntaxChecker** checks syntax of deployment diagrams and identifies their elements

3. **AVATAR2SOCLIB** translates AVATAR blocks into C POSIX tasks, generates main program

4. **TopcellGenerator** generates SystemC top cell

5. **LdscriptGenerator** generates linker script
Top Cell Generation

- SystemC instanciation of components, netlist, table associating memory segments, charging of code
- Some components transparent to the user (interrupts, simulation infrastructure)
#define CHANNEL0 __attribute__((section("section_channel0")))
#define LOCK0 __attribute__((section("section_lock0")))

...  
pthread_t thread__Classif0;
pthread_t thread__Classif1;
pthread_t thread__Classif2;
pthread_t thread__Sched0;
pthread_t thread__Sched1;
...

struct mwmr_s *channels_array_Classif[3];
...

ptr = malloc(sizeof(pthread_t));
thread__Classif2 = (pthread_t)ptr;
atr_t = malloc(sizeof(pthread_attr_t));
 pthread_attr_set_affinity(atr_t, 2);
Code Generation: Extract from Topcell

```c
maptab.add(Segment("cram0", 0x1000000, 0x800, IntTab(2), true));
maptab.add(Segment("uram0", 0x10200800, 0x800, IntTab(2), false));
maptab.add(Segment("cram1", 0x20000000, 0x800, IntTab(7), true));
maptab.add(Segment("uram1", 0x20200800, 0x800, IntTab(7), false));
...

callback::VciRam<vci_param>Memory1("Memory1", IntTab(2), maptab);
callback::VciRam<vci_param>Memory0("Memory0", IntTab(7), maptab);
...

data_ldr.load_file(std::string(kernel_p) + ";.data;.channel0;...
.channel14;.cpudata;.contextdata");
```
Code Generation:
Extract from ldscript

uram0 (RWAL) : ORIGIN = 0x10280000, LENGTH = 0x80000

cram0 (RWAL) : ORIGIN = 0x10000000, LENGTH = 0x80000

... .channel0 : {*(section_channel0)} > uram0
.channel14 : {*(section_channel14)} > uram1
.lock0 : { *(section_lock0) } > uram0

... .lock14 : { *(section_lock14) } > uram1
Experimental Setup

- CABA (Cycle Accurate/Bit Accurate) level simulation
- *PowerPC405* processors
- MutekH
- SYSTEMCASS
Using TTool/SoCLib

Prototyping
Contribution
Code Generation
Experiments
Conclusion

Institut Mines-Telecom, Paris Sorbonne Universités
System-Level Design
Using TTool/SoCLib: Channel Configuration and Code Generation Dialogues

Setting signal association (sur debussy)

- Adding signals
- Managing Signals
  - Classification.queue_low(PacketDesc)
  - Classification.queue_medium(PacketDesc)
  - Classification.queue_high(PacketDesc)

- Add Signals
- Remove signals

Executable Code generation

- Code generation
  - Base directory of code generation code SoClib:
    - //executablecode/
  - Base directory of code generation code TopCell:
    - //executablecode/

- Remove .c/.h files
- Remove .x files
- Remove .cc files
- Put debug information in generated code
- Put tracing capabilities in generated code
- Optimize code
- Include user code
- 1 time unit =

- Code generator used:
  - AVATAR TOPCELL/SoCLIB

Start... Stop... Close...
Experiments

Duration of a transfer in the priority queue (SoCLib simulation cycles)
Experiments

Fill state of the priority queues
Conclusion

Limitations

- AVATAR cannot model application specific co-processors
- Better capture latencies and automatically evaluate them during the prototyping stage
- Support more interconnects (CAN, clustered etc.)

Perspectives

- Increase simulation speed (TLM?)
- Performance evaluation tools
- Long term: Design Space Exploration with low-level simulations
Questions?

To try/download TTool:

   ttool.telecom-paristech.fr

To try/download SoCLib/MutekH:

   www.soclib.fr
   www.mutekh.fr