System-Level Modeling, Analysis and Code Generation: Object Recognition Case Study

Marius BOZGA, Research Engineer, VERIMAG
29/02/2012, Nürnberg, Germany
Co-Authors

- Ananda Basu, Saddek Bensalem, Marius Bozga
  *Joseph Fourier University, Grenoble, France*
- Julien Mottin, Francois Pacull
  *CEA-LETI Minatec, Grenoble, France*
- Athanasios Poulakidas, Aggelis Aggelis
  *Hellenic Airspace Industries, Athens, Greece*
Outline

▶ Design Flow for Parallel Software on Manycore
  ▶ Core Formalisms: DOL and BIP
  ▶ System-Level Modeling and Analysis
  ▶ Backend Code Generation Flow

▶ Case Study: HMAX Models Algorithm on P2012
  ▶ DOL and BIP Models
  ▶ System-Level Performance Results
  ▶ Code Generation Experiments on P2012
  ▶ Lessons learned

▶ Discussions
Design Flow Key Characteristics

- model-based design flow
- process network programming model in DOL
- support for design space exploration and automatic mapping using DOL
- allows for validation of functional correctness on the BIP software model
- fine-grain performance analysis on BIP system models
- backend code generation for P2012
Distributed Operation Layer - DOL

- Framework from ETH Zurich for programming parallel applications on manycore
- Design space exploration i.e., optimal mapping computation
- Selection through analytical performance evaluation methods such as Real-Time Calculus
Behavior Interaction Priority - BIP

- a component based framework from UJF/VERIMAG, Grenoble
- encompassing rigorous model based design
- supported by a rich toolbox for model transformation, simulation, formal verification and code generation
The BIP/DOL Design Flow in SMECY

- **D-Finder**
  - Software Model (BIP)
  - Functional verification

- **dol2bip**
  - Translation
  - The BIP/DOL Design Flow in SMECY

- **BIP/weaver**
  - Model transformation

- **BIP/simulator**
  - Performance analysis

- **System Model (BIP)**

- **BIP/backend code generator**
  - Code generation

- **P2012 compiler stxp70cc**
  - Application Executable(s)

- **Application Software Sequential Code**

- **KPN Application (C + XML)**

- **Static Mapping (XML)**

- **Abstract Platform (XML)**

- **P2012 platform**

- **Correctness results**

- **Performance results**

- **Hardware Dep. Software**

- **Application Code (CPU) (C with HdS calls)**

- **DOL**

- **ARTEMIS Joint Undertaking**
System-Level Modeling and Analysis

- Construct mixed SW/HW *System Models* from models of the application software, the target platform and the mapping

- System Models are constructed in a systematic way, by applying a **fixed number of transformations** on the software model e.g.,
  - splitting software channels,
  - breaking atomicity of read/write operations in processes,
  - inserting hardware-dependent software components…

- Model transformations **include extra-functional properties** from the target platform into the software model:
  - using a predefined library of BIP components
  - characterized by the HW architecture, OS, etc

- Model transformations **are correct-by-construction**:
  - preserving functional properties, no deadlocks are introduced
System Model Construction

Software Model (BIP)

System Model (BIP)

Hardware Model (BIP)
Backend Code Generation

System Model (BIP)

BIP/backend code generator

Glue Code (C)
- Main application routine
- Deployment of threads to cores, data to memories
- Data allocation: thread stacks; FIFO queues

Functional Code (C)
- Application tasks: local data + thread routines
- Communication: API calls provided by runtime

Runtime APIs (C)
- Thread Management, Memory allocation
- Communication, Synchronization
- available in P2012 SDK 2011.1

P2012 compiler stxp70cc

Application Executable(s)
P2012/MPARM
Glue Code Fragment

```c
#define NUM_PROCESS 3

/*** Forward declaration of process (slave) threads ***/
void* generator_ins_execute(void*);
void* consumer_ins_execute(void*);
void* square_ins_execute(void*);

/*** Process map table (global) ***/
process_map_t process_map [NUM_PROCESS] = {
  0, 1, generator_ins_execute,
  0, 2, square_ins_execute,
  0, 3, consumer_ins_execute
};

procid_t proc_id = {.cluster_id = p_map.cluster_id,
                    .core_id = p_map.core_id};
npl_printf("launching process in cluster (%i) core (%i)\n",
            p_map.cluster_id, p_map.core_id);

seg_alloc(get_L3_hid(), 1024, &stacks[proc_index]);
thread_create(&threads[proc_index], proc_id, p_map.start_routine,
              0, &stacks[proc_index], 0, p_map.label);

npl_printf("[Master] completed thread creation.\n");

/*** Wait for all process threads to finish ***/
while(joiner.thread_count > 0) {
```

Outline

- Design Flow for Parallel Software on Manycore
  - Core Formalisms: DOL and BIP
  - System-Level Modeling and Analysis
  - Backend Code Generation Flow

- Case Study: HMAX Models Algorithm on P2012
  - DOL and BIP Modeling
  - System-Level Performance Results
  - Code Generation Experiments
  - Lessons learned

- Discussions
HMAX Models Algorithm

- Application:
  - Contour detection
  - Object recognition
  - ...

- Successive layer computation

- Feature computation at layer $n$ is based on layers $1..n-1$

- Case study: Layer 1
  - Feature = 2D-Gabor Filter, Max Filter
HMAX Models Algorithm

- Legacy C++ sequential implementation: ~1000 LOC
- Image processing: 2 layers, 13 features
  - First layer: 2D-Gabor and Max filters on 12 image subsamples
- Implementation based on OpenCV library
  - JPEG decompression
  - Subsampling
- Parallelism can be exploited at several levels
  - Coarse-grain: feature-level
  - Fine-grain: pixel-level
P2012 Cluster [STMicroelectronics]

- SDK 2011.1
  - GEPOP virtual platform, functional simulator
  - TLM virtual platform, cycle-accurate simulator

- Features
  - Single cluster
  - 1 CC, 16 PE
  - 256 Kb cluster-shared memory
HMAX/P2012 Modeling in DOL

DOL Application:
25 processes, 45 FIFOS

P2012 cluster
16 cores, 2 Buses, 2 memories

PE1 PE2 ... PE16

LIC

BUS

L3 MEM

TCDM 256 Kb

System Level Modeling, Analysis and Code Generation: Object Recognition Case Study
System-Level Analysis

▶ System Model in BIP for the S1 layer
  ▶ 2 mappings: all FIFOs to L1, respectively L3
  ▶ 125 components, 1500 connectors, 13000 lines of BIP code

▶ GFilter Execution Time(s)
  M-cycles

▶ S1 Execution Time(s)
  M-cycles

▶ speedup : 2.5
▶ platform utilization: 15.6%
Code Generation Experiments

- Run on **GEPOP** platform
  - Legacy implementation & distributed implementation: functional equivalence
  - GEPOP runs really fast: few seconds, similar to legacy application
  - No performance evaluation in 2011.1 SDK

- Run on **TLM** platform
  - Floating point support not fully available in 2011.1 SDK for TLM platform
  - Simulation time around 5 minutes
  - Memory allocation errors
Lessons Learned

- **Cluster (L1) memory is a critical resource on P2012**
  - current DOL model exhibiting only coarse-grain parallelism is not the best fit for the P2012
  - in the flow, allocate only FIFOs and threads stacks...
  - fine grain memory management is needed

- **Interaction with host-side application is crucial**
  - images cannot be loaded at runtime from files
  - preload images in memory at compile time, big size images have been manually reduced

- **Integration of legacy code is not as easy ...**
  - C/C++ is used everywhere: DOL, BIP, P2012 but...
  - P2012 stxp70cc is a C compiler (not C++),
  - (manual) code rewriting has been used to replace OpenCV
HMAX/P2012 Ongoing Work

- Focus on parallelization of 2D-Gabor and Max filters, instead of parallelization of the whole application

- Better exploitation of P2012 resources, both processing elements and memories

- Port on the latest P2012 SDK which features multi-cluster support and faster simulation and performance estimation tools
Design Flow for Parallel Software on Manycore
- Core Formalisms: DOL and BIP
- System-Level Modeling and Analysis
- Backend Code Generation Flow

Case Study: HMAX Models Algorithm on P2012
- DOL and BIP Modeling
- System-Level Performance Results
- Code Generation Experiments
- Lessons learned

Discussions
Discussions

- The DOL/BIP Design Flow:
  - allows generation of a correct-by-construction model of mixed HW/SW system from its application, a description of the architecture and a mapping (single semantic framework)
  - separates SW and HW design issues, parameterized by design choices related to resource management (scheduling, memory size and execution times)
  - completely automated and supported by tools
- System Model as a basis for performance evaluation and verification of extra functional properties
- Experimental results shows the flow is tractable and allow for design space exploration to determine optimal solutions
Perspectives for Future Work

- Extension to **other programming models** for the application software and **richer hardware** architecture models (DMA, NoC, etc.)
- Use **statistical model checking** for validation of several applications running on multi-core architectures
- Improve **backend code generation**, mainly with respect to memory management

- Toolbox downloadable from 
  
  [http://www-verimag.imag.fr/BIP-System-Designer.html](http://www-verimag.imag.fr/BIP-System-Designer.html)
Thank you for your attention!

Marius Bozga & Marius.Bozga@imag.fr