Process Network Models for Embedded System Design based on the Real-Time BIP Execution Engine*

Fotios Gioulekas¹, Peter Poplavko², Panagiotis Katsaros¹,³, Pedro Palomo⁴

¹Aristotle University of Thessaloniki
²Mentor®, A Siemens Business
³Information Technologies Institute, CERTH
⁴Deimos-Space S.L.U.

*Research supported by the ESA project (2015-2016) “Schedulability Analysis Techniques and Tools for Cached and Multicore Processors (MoSaTT-CMP)”, contract No. 4000111814/14/NL/MH
Outline

- Model-Based Design (MBD) for real-time embedded systems
- Rigorous Design-Flow of real-time embedded systems based on Fixed Priority Process Networks (FPPNs)
- Design of FPPNs using ESA’s TASTE toolset
  - Model Transformation (TASTE2BIP)
  - Schedulability analysis
- Case-Study: Guidance Navigation & Control Application executed with BIP RTE on 4-Core LEON4FT NGMP platform
- Future work & Discussion
MBD for Real-Time Embedded Systems - I

- Model-Based Design flow systematically involves domain-specific models (DSMs)
  - Application behavior
  - HW/SW partitioning
  - Mapping onto an architecture

- Analysis of system’s nonfunctional properties (e.g. task execution times, memory footprint, schedulability) is based on DSMs throughout the design process e.g. by model checking, simulation, analytical methods

- Enables early verification and performance estimation
MBD for Real-Time Embedded Systems - II

- Architecture-centric approach
  - Via model transformations the system’s non-functional properties are analyzed and described with appropriate tools (e.g. AADL language)
  - Schedulability is based on assumptions for the temporal and concurrency properties of computations, comm. and synch. (e.g. priority based preemption)

- Synchronous languages (e.g. Esterel, Lustre)
  - Suitable for formal design, verification & code generation of reactive systems (e.g. flight control)
  - Program reacts in a sequence of logical clock ticks and computations within a tick are instantaneous (reaction to stimuli within strict time bounds)

References:
MBD for Real-Time Embedded Systems - III

- Synchronous languages (e.g. Esterel, Lustre) lack appropriate concepts for task parallelism and timing-predictable scheduling on multiprocessors
- Ptolemy II and PeaCE support design based on Models of Computation (MoCs) and subsequent code-generation, however schedulability aspects are often ignored
- Rigorous model-based design flow aims at a system implementation derived from high-level models by applying a sequence of semantics-preserving transformations
- Proposed Rigorous MBD:
  - Usage of a new MoC called FPPN which is appropriate for timing-aware modeling at the early design steps
  - Task Schedulability on multi-cores

---

Fixed Priority Process Networks (FPPNs) - I

- FPPNs combine streaming and reactive control processing defined by two directed graphs
  - (Possibly cyclic) graph \((P, C)\), whose nodes \(P\) are processes and edges \(C\) are channels for pairs of communicating processes that define a dataflow direction, i.e. from the writer to the reader
  - Graph \((P, FP)\) is the functional priority directed acyclic graph (DAG) with edges defining a functional priority relation between processes ensuring its functional determinism => precedence constraint on task execution
Fixed Priority Process Networks (FPPNs) - II

- “X” sporadic process generates values, the “Square” process calculates the square of the received value and the “Y” periodic process serves as a sink for the squared value
- Periodic process is annotated by its period
- Sporadic process is annotated by its minimal inter-arrival time
- Two types of non-blocking inter-process channels
  - FIFO (mailbox) has a semantics of a queue
  - Blackboard (shared variable) remembers the last written value that can be read multiple times
- The arc depicted above the channels indicates the functional priority relation FP (higher to lower)
Rigorous Design-Flow of real-time embedded systems based on FPPNs - I

- **Our method**: “TASTE – to – BIP” design flow
- Capture FPPN in architectural design framework (TASTE) and perform model transformation into expressive formal language BIP and use it for refinement towards final implementation
  - **Input**: (i) application requirements (FPPN model), (ii) platform requirements
  - **Output**: implementation on the target platform using BIP run-time environment

Rigorous Design-Flow of real-time embedded systems based on FPPNs - II

- **Step 1 Architectural design**
  - The functional code (software behavior) is implemented and the requirements are mapped to an architectural model (i.e., TASTE I-V): Application is decomposed into FPPN processes
  - Establish the dependencies between processes in the form of functional priorities and data channels

- **Step 2 Model transformation**
- **Step 3 Functional simulation of RT-BIP model**
- **Step 4 Worst Case Execution Time (WCET) Estimation**
- **Step 5 Schedulability analysis & timing simulation**
- **Step 6 Code generation for the BIP RTE**
- **Step 7 Performance analysis on the target platform**
Rigorous Design-Flow of real-time embedded systems based on FPPNs - II

- **Step 1** Architectural design
- **Step 2** Model transformation
  - FPPN model transformation into BIP according to the FPPN execution semantics in [1, 2]
  - If WCETs are known, the task graph is also generated
    - [if (Task Graph exists) goto Step 5]
- **Step 3** Functional simulation of RT-BIP model
- **Step 4** Worst Case Execution Time (WCET) Estimation
- **Step 5** Schedulability analysis & timing simulation
- **Step 6** Code generation for the BIP RTE
- **Step 7** Performance analysis on the target platform

TASTE 2 BIP model transformation principle

TASTE-IV architectural model

Set of functional blocks communicating via SW interfaces

void SQR_Init() {
    index = 0;
}

void SQR_Execute() {
    XIF_Read(&x, &x_valid);
    if (x_valid) {
        y = x * x;
        YIF_Write(&y);
    }
    index = index + 1;
}

BIP model

Network of communicating timed automata
Design of Fixed Priority Process Networks (FPPNs) using ESA’s TASTE toolset - I

**INPUTS**

- InterfaceView.xml
  
  *(TASTE-IV functional model)*

- TASTE-IV C functional Code
  
  *(TASTE-IV generated folders that contain the C code files per process and channel)*

- offsets.dat
  
  *(process_name=offset time in ms in each row if offset is not zero)*

**OUTPUTS**

- fppn folder
  
  - fppn.xml
    
    *(BIP model definition)*

  - generator.out
    
    *(logging info)*

  - fppn_tg.jobs
    
    *(number of jobs per process within H: TASK GRAPH DESCRIPTION)*

  - fppn_tg.jobsprocs
    
    *(job characteristics per process and job arc pairs: TASK GRAPH DESCRIPTION)*

- src folder
  
  - BIP C++ code
  
  - temp_asn1.h
    
    *(ASN1 data type definitions)*
Design of Fixed Priority Process Networks (FPPNs) using ESA’s TASTE toolset - II

- **FPPNClass** attribute: the type of FPPN entities (e.g. blackboard, periodic process)
- The **Priority** attribute is an integer, which dictates the priority index of the process (priority order in the network)
The **FPPNClass** attributes **mailbox** and **blackboard** are used for data channels.

Each channel declares two **provided interfaces** for ‘read’ and ‘write’, while the processes that access the channel have respective **required interfaces**.

**DataChannelSize** represents the minimum size of the data type (in bytes) communicated via the channel.

**DataChannelLength** is defined in mailbox channel determining the length of the FIFO.
Rigorous Design-Flow of real-time embedded systems based on FPPNs - II

- Step 1 Architectural design
- Step 2 Model transformation
- **Step 3 Functional simulation of RT-BIP model**
  - Generated BIP model is functionally tested on a workstation
- Step 4 Worst Case Execution Time (WCET) Estimation
- Step 5 Schedulability analysis & timing simulation
- Step 6 Code generation for the BIP RTE
- Step 7 Performance analysis on the target platform
Rigorous Design-Flow of real-time embedded systems based on FPPNs - II

- Step 1 Architectural design
- Step 2 Model transformation
- Step 3 Functional simulation of RT-BIP model
- **Step 4 Worst Case Execution Time (WCET) Estimation**
  - The probabilistic measurement-based timing analysis in [1] is used to guarantee safe probabilistic bounds
- Step 5 Schedulability analysis & timing simulation
- Step 6 Code generation for the BIP RTE
- Step 7 Performance analysis on the target platform

Rigorous Design-Flow of real-time embedded systems based on FPPNs - II

- Step 1 Architectural design
- Step 2 Model transformation
- Step 3 Functional simulation of RT-BIP model
- Step 4 Worst Case Execution Time (WCET) Estimation
- **Step 5 Schedulability analysis & timing simulation**
  - The task graph is generated and given as input to a static scheduler
  - The schedule obtained from the scheduler is translated into input for the online-scheduler model in BIP, which implements resource management (by enforcing task ordering and other constraints)
    - [if (! schedulable) iterate Steps 1 to 4]
- Step 6 Code generation for the BIP RTE
- Step 7 Performance analysis on the target platform
The “split” task appends two small data items to the two output channels

Tasks “A” and “B” read the data

All tasks have the same periodic scheduling window, with period and deadline being 25ms
In the derived task graph, every task is represented by a job.
The arrival times $A_i$ and deadlines $D_i$ for all jobs are the same.

Jobs are annotated by WCETs.
$\delta$ is the worst-case cost of a single transition in the BIP automata components.

$J_i: A_i = 0, D_i = 25 \text{ ms}, \delta = 1 \text{ ms}$
Schedulability analysis and code generation for the BIP-RTE - III

- The offline scheduler takes into account the cost of BIP transitions by BIP-RTE and the execution time of jobs and generates the schedule for the online scheduler.

- Core 0: BIP-RTE
- Core 1: Task split & Task A
- Core 2: Task B
Rigorous Design-Flow of real-time embedded systems based on FPPNs

- Step 1 Architectural design
- Step 2 Model transformation
- Step 3 Functional simulation of RT-BIP model
- Step 4 Worst Case Execution Time (WCET) Estimation
- Step 5 Schedulability analysis & timing simulation
- Step 6 Code generation for the BIP RTE
  - The joint application/scheduler model is compiled by the RT BIP compiler and linked with the BIP-RTE
- Step 7 Performance analysis on the target platform
Rigorous Design-Flow of real-time embedded systems based on FPPNs - II

- Step 1 Architectural design
- Step 2 Model transformation
- Step 3 Functional simulation of RT-BIP model
- Step 4 Worst Case Execution Time (WCET) Estimation
- Step 5 Schedulability analysis & timing simulation
- Step 6 Code generation for the BIP RTE

- Step 7 Performance analysis on the target platform
  - Validation by performance analysis is essential towards identifying possible excessive delays, due to resource starvation cases
  - The executable runs on the target platform on top of the real-time operating system (RTEMS-SMP)
  - Tools are used (e.g. gprof) that trace/monitor the software performance on the target platform
    - [if (excessive delays found) goto Step 1]
Case-Study: GNC application - I

- Demonstrate the execution of the GNC application with BIP RTE on 4-Core LEON4FT embedded platform [1]
- We adapt the FPPN, Task Graph and BIP models to explore parallelism in comparison to [2]
  - the potential for such an exploratory approach is inherent in the FPPN model and remains transparent to the application designer until the final steps of our rigorous design flow

Case-Study: GNC application - II

Guidance Navigation & Control (GNC) is on-board spacecraft application that controls the movement of the vehicle by processing the data of the corresponding sensors and controller

- **Data Input Dispatcher Task**: signals each time new data (Mission and Vehicle Management, Inertial Measurement Unit, Global Positioning System) is available (reads, decodes, dispatches) - pre-computed and stored in static-memory buffers as C arrays

- **Guidance Navigation Task**: executes the guidance and navigation algorithms

- **Control FM Task**: performs the control and flight management algorithms

- **Control Output Task**: sends the outputs of the GNC to the Dynamics Kinematics and Environment module (DKE). The output data consists of the geodetic altitude, the longitude, the mach and the dynamic pressure values
Case-Study: GNC application - III

- TASTE-IV FPPN model
- Functional priorities were assigned based on the specification
The WCET values were estimated by profiling the application’s execution under BIP RTE (In progress to use Statistical tools described in [1]).
Real-Time Execution of the GNC App. on 4-Core LEON4FT NGMP - I

- Real-time execution on LEON4FT
  - P1: Data Input Dispatcher
  - P2: Control FM
  - P3: Control Output
  - P4: Guidance Navigation
  - P20: BIP-RTE Engine

- P4 and P3 processes skip their first job execution

- 11th job of P1 is executed in parallel with 1st job of P4
  - P4 and P1 access to the buffered mailbox concurrently
  - P4 reads the first 10 valid IMU frames stored while P1 writes the 11th frame
Real-Time Execution of the GNC App. on 4-Core LEON4FT NGMP - II

- all activities in the current period end by time slightly less than 40ms after the period start
- job in P4 requires more time to be completed than non-pipelined version in [1]=> no throughput improvement (P1, P4 interference)
- improving the implementation of the mailbox and letting P4 ten data items in one call to the `read’ interface instead of issuing 10 calls as it is done now

Discussion & Future Work

- Rigorous Design Flow for the FPPN MoC using TASTE2BIP transformation and model refinement for schedulability
- Validated in real application for multi-core embedded platform
- Future work:
  - Support distributed multi-core platforms
  - Support for preemptive scheduling by BIP/BIP RTE
  - WCET measurement tool integration
  - Support additional languages included in TASTE e.g. ITU-T SDL and Simulink
Thank you

TASTE2BIP tool download:
http://www-verimag.imag.fr/Time-Critical-Applications-on-Multicore
Or
Google for
Time Critical Multicore Verimag