PRiME run smarter, live longer

Research Brief

Power-efficient, Reliable, Many-core Embedded systems www.prime-project.org

Automatic generation of run-time management software from Event-B Models

PRiME *is a five year EPSRC funded research programme (2013-2018), in which four UK universities address the challenges of* ***power consumption*** *and* ***reliability*** *of future high-performance embedded systems utilising many-core processors.*

Formal RTM Design and Automatic Code Generation

In single or multi/many core embedded processor systems, **Run-Time Management** (RTM) software can be used to control energy hooks at run-time in order to minimise the energy consumption. Typically, such RTM systems are aware of application requirements and utilise workload prediction and machine learning algorithms to estimate the optimal configuration for energy saving.

In computer science, **formal methods** are mathematically based techniques for the specification and development of complex systems. By building a mathematically rigorous model of a system, it is possible to verify the system's properties in a more thorough fashion than empirical testing, and therefore improve the reliability and robustness of the design. **Event-B** is one such formal method for system-level modelling and analysis which can be used to create a formal model of the RTM.

Verifying the formal RTM design earlier in the development process – before implementation - can significantly reduce the cost and complexity of fixing any errors or failures. In addition, the formal model of a RTM system can be automatically translated into executable code to be run on the hardware. Automatic Code Generation (CG) reduces the effort of a hand-coded implementation and is portable across different architectures and Operating Systems (OSs).

Formal Model of Video Decoder RTM and associated Code Generation RTM system

Figure 1 shows the RTM within a cross-layer approach, interacting with the application, OS and hardware. Communication between layers is indicated by arrows. This shows the RTM for a video decoder application, where a frame can be seen as a task, and the deadline is obtained from the Frames Per Second (FPS) rate set by the video application. The frame workload needs to be known prior to its processing, then decisions on the power state (Voltage and Frequency, V-F) have to be taken so that they fulfil the constraint but take into account performance variations of the application.

To achieve this behaviour, PRiME's RTM algorithm works in two phases, Prediction and Decision Making (using Machine Learning). For each frame, the RTM first predicts the workload to be executed, and then it decides the V-F setting so that the predicted workload can finish execution before the frame deadline, set by the FPS.

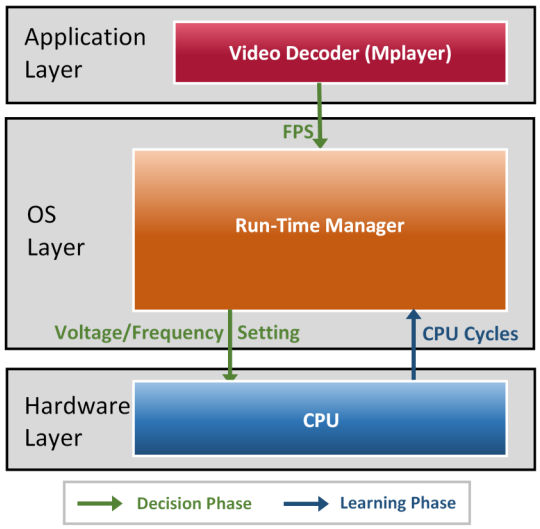

Figure 1. Cross-layer approach to RTM

PRiME has developed a formal approach towards automatic generation of RTM software for a video decoder application, starting from a verified formal model of the RTM. The formal model of the RTM system is developed using the Event-B formal modelling language and is verified using theorem proving and model checking.

Figure 2 illustrates the design architecture for Event-B modelling of the RTM for a video decoder application. Event-B refinement allows a model to be built gradually, starting with an abstract model and then introducing successive, more concrete refinements. The Event-B model of the RTM system comprises an abstraction level – which focuses on the main functionalities of the system - and two levels of

refinements - where the workload prediction and the reinforcement learning algorithm are introduced respectively. To manage the complexity of the final refinement and also to prepare the model for code generation, the model is decomposed into two sub-models: Controller and Environment. The Controller sub-model consists of properties of the RTM algorithms and the Environment sub-model represents the interfaces between the RTM and the application and hardware layers. Finally executable code is generated automatically from the final refinements of the sub-models.

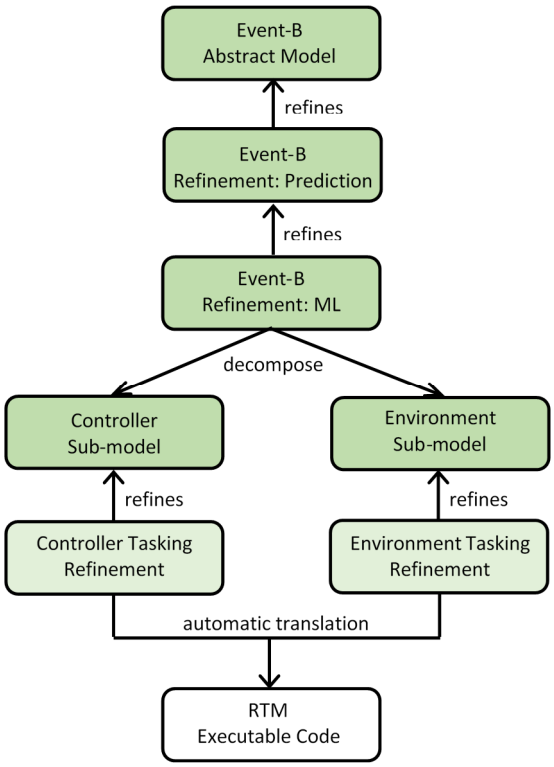

Figure 2: RTM design architecture

Experimental Results and Comparison

Experiments were first conducted on a BeagleBoard-xM (BBxM) embedded platform, which contains a TI OMAP DM3730 SoC with an ARM Cortex-A8 processor, running Linux Operating System 3.7.10. For the video decoder case study, we used FFMPEG5 libraries, running H.264 videos of VGA resolution (640x480) for 720 frames, which at 23.976 FPS is 30.03 seconds.

The automatically generated RTM system has been successfully integrated into an embedded platform as a Linux governor, and provides up to 4% improvement in energy consumption compared to Linux's default "ondemand" governor, (see figure 3).

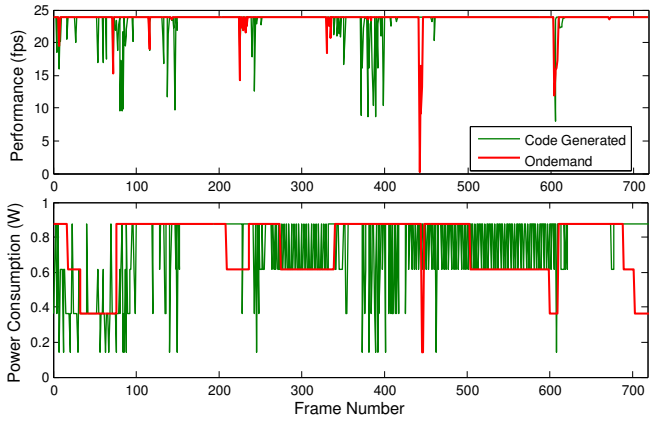

Figure 3. Power saving compared to Linux ondemand

Future Work: Independent CG RTM system

The Code Generated RTM of the video decoder has been used on two further platforms: Xilinx Zynq and Xeon Phi and work is progressing to further optimise the Event-B model. This work will be followed by optimisation of the RT algorithms themselves. Any optimisation will be added as a refinement to the Event-B model and the new CG RTM will again be compared with the state-of-the-art.

Subsequently, different control knobs, monitors and different RT algorithms will be modelled and the automatic CG process will be applied to them. This will lead to the development of independent CG RTM systems.

More information

Visit the PRiME programme web site, including the opportunity to sign-up for programme updates, or contact the PRiME Collaboration Manager (including industry liaison):

Gerry Scott
Email: gerry.scott@soton.ac.uk
Tel: +44 (0)23 8059 2749

www.prime-project.org