This submission addresses: Recent research advances in MoDES

Design of High-Performance Embedded System using Model Integrated Computing[1]

Sumit Mohanty and Viktor K. Prasanna
Dept. of Electrical Engineering
University of Southern California
Los Angeles, CA 90089

Abstract

Model Integrated Computing (MIC) promotes the use of models as a "backbone" of model-integrated system development. In this paper, we study the use of MIC to design a heterogeneous high-performance, low-power embedded system. Use of MIC enabled us to accommodate evolving performance requirements, target hardware specification, and mission scenarios while promoting design reuse. MILAN is a model-based integrated simulation environment for embedded system design based on the concept of MIC. While MILAN is capable of addressing a wide range of system design problems, in this paper, we focus on the problem of device selection for high-performance low-power embedded systems while satisfying given latency constraint. We illustrate the challenges in the design of such embedded systems and discuss various supports provided by MILAN through the design of an automated target detection and tracking system.

1. Introduction

An embedded system consists of an application, a target hardware, and satisfies a set of given performance and design constraints. Target hardware for an embedded system ranges from single device to multiple heterogeneous devices. Single device implementations use general purpose processor, DSP, or application specific integrated circuits (ASICs). On the other hand, multiple heterogeneous devices include one or more devices of different types (e.g. an FPGA and a processor) into single target hardware. For example, an Intel PXA 255 [16] based embedded system is a single device system while a multi-chip solution like the PASTA sensor stack [12] which integrates processors, micro-controllers, custom logics, sensors, and actuators. An embedded system design problem is an application mapping problem when the target hardware is already specified. In contrast, the embedded system design problem is a device selection and mapping problem when, instead of the target hardware, a set of possible target devices are specified. Device selection problem evaluates various combinations of target devices to identify a single or multiple device implementation of the target application that satisfies the given performance and design constraints. In this paper, we focus on the problem of device selection.

Target applications can be of many types such as signal processing, control, man-machine interface, etc. In this paper, we focus on signal processing applications that process a stream of input frames while meeting a given latency constraint for the processing of a single frame. We also focus on the specific performance constraint of minimizing energy dissipation while meeting the given latency constraint. Our focus is on the "next generation" low-power embedded systems with strict latency requirement. Examples of such systems include mobile base stations for software-defined radio [2], target detection and tracking systems [12][13], and space applications. Such embedded systems may not meet the volume requirement to consider ASIC as a feasible solution. So the design problem is to identify a hardware/software co-design solution that uses FPGAs and ISA-based processors. The tradeoff considered here is high throughput and relatively higher energy dissipation using FPGAs vs. comparatively lower throughput and energy efficient solution using general purpose processors. Additionally, DSPs also have the potential to provide a meet-in-the-middle solution. In addition, some applications are also associated with tasks with requirements such as floating-point arithmetic or control-intensive logic for which an ISA-based processor may be preferred over an FPGA. We focus on the design of embedded systems that have the above characteristics.

State-of-the-art devices support various features such as low-power operating modes, efficient shutdown and start-up, and dynamic voltage scaling that can be exploited during embedded system design to reduce energy dissipation [10].

[1] This work is part of the PARIS and PASTA project, supported by the DARPA Power Aware Computing and Communication (PAC/C) Program under contracts S76AX077X-62A9 and F30602-99-1-0529.

Duty cycle is the proportion of time a system is active. A system is active when it is processing input frames. Therefore, based on duty cycle specification execution of a system can be modeled as active and inactive phases. During inactive phases the system is not processing any data. However, during inactive phases, energy is dissipated due to leakage current and clock signal distribution [10]. If the duration of the inactive phases is significantly larger than that of the active phases, optimizing energy dissipation during inactive phases contributes significantly towards the overall energy efficiency [9]. Therefore, in order to fully exploit the energy saving features discussed above, we need to model and analyze the embedded system based on a period of execution that includes active and inactive phases. In order to reduce energy dissipation, it is attractive to consider the option of shutting-down the devices that are in inactive phase. However, this option has to be evaluated against the latency and energy required to start-up a device which may contribute towards a higher overall latency and energy dissipation.

Therefore, the device selection problem, for an embedded system that meets the given latency requirements while minimizing overall energy dissipation for the system, is a non-trivial system design problem. In this paper, we discuss various support provided by MILAN (Model-based Integrated Simulation) and our experiences while solving the above problem.

MILAN is a model based integrated simulation environment for embedded system design and optimization through integration of various simulators and tools into a unified environment [1]. Using the MILAN environment, the designer formally models the target application, underlying hardware, and constraints (latency, energy dissipation, etc.) through a graphical interface provided by MILAN. The model information is translated through model interpreters into suitable input formats required by the integrated simulators. MILAN adopts Model Integrated Computing (MIC) as the core design technology [14]. The Generic Modeling Environment (GME) is a configurable graphical tool supporting MIC [4]. GME allows the designer to create domain-specific models. Every target system is specified in MILAN as a model. Model interpreters are the software components that translate the information captured in the models based on the input format required by the integrated tools and simulators. In this paper, we discuss various capabilities of MILAN that enables design of high-performance low-power embedded systems.

This paper is organized as follows. Next section provides a short description of the MILAN design environment and highlights various features that are used to solve the device selection problem. Section 3 demonstrates the use of MILAN for an automated target detection and tracking application. Section 4 discusses the capabilities and limitations of the MILAN environment in the context of solving the class of design problems presented in this paper. We conclude in Section 5.

2. MILAN: A Model-based Integrated Environment for Embedded System Design

MILAN design flow consists of modeling, performance estimation, and design space exploration. The user initiates the design process by modeling the application and the target architecture. Application modeling involves application specification as a data-flow graph with alternatives [3]. The functional specification of the target system specifies the structure of the data-flow graph and the choice of implementations specifies the alternatives. MILAN supports hierarchical modeling that enables hierarchical specification of the data flow graph making it easier to manage and analyze an application model.

Resource modeling involves modeling of the hardware. The modeling paradigm is based on the GenM model [8]. The user identifies key components and features of the target architecture that can be exploited for optimization and models them in MILAN [1]. In addition, the user also models various states and state transition costs associated with different components such as reconfigurable devices, processors supporting DVS, and power aware memories [1].

Finally, the user describes possible mappings for each task alternatives and different performance or compositional constraints that the system needs to satisfy [7]. A mapping is a relation between an application task and a target processing component. Performance constraints are based on the latency and energy dissipation requirements given as input. Design constraints capture requirements of mapping and composition of components in case of device selection. Constraints on composition restrict the composition of alternate processing components.

Once the complete system is modeled, the user invokes the design space exploration (DSE) tools. A DSE tool rapidly identifies a set of design that satisfies all the constraints. Currently, MILAN provides Design Space Exploration Tool, DESERT, as the primary DSE tool [11]. The output of DSE is evaluated using High-level Performance Estimator (HiPerE) tool [8], currently integrated into MILAN. HiPerE evaluates system-level energy dissipation and latency. In order to provide a rapid estimate, HiPerE operates at the task level abstraction of the application. In addition to the task execution cost, various other aspects considered by HiPerE for accurate performance estimation are data access cost, parallelism in the system, energy dissipation when a component is idle, and state transition cost. HiPerE also supports performance estimation based on duty cycle specification [9]. Using HiPerE, the designs are evaluated and compared manually to identify the final design.

2.1 Using MILAN for Device Selection for Design of High-Performance Low-Power Embedded Systems

Initially, using MILAN we model the target application as a hierarchical data flow graph (Figure 1). We assign several choice of implementation for each application task. Assuming that a task can be mapped onto a set of devices and implemented using a set of algorithms, each unique choice refers to a unique device and algorithm combination. In addition, if a device has several operating states (e.g. multiple operating voltages) each unique choice refers to (in addition to device and algorithm) a unique operating state. We use the MILAN resource model to specify the hardware devices available. Initially, we assume that the target hardware includes all the available devices. Figure 2 shows an instance of such a model, which consists of one DSP, two FPGAs, and two general purpose processors. MILAN supports specification of constraints that ensures valid combination of devices. We use such constraints to ensure that whenever a design is chosen only valid combination of devices are selected for mapping of the application tasks. For example, given a set of choices that includes two traditional processors, one such constraint can be “a valid system may contain one of the processors but not both".

MILAN also supports addition or deletion of devices. Deletion of any device results in deletion of the model of the device and the connections with other device models. From the application model, for each task, the choices that correspond to the deleted device are also deleted. Finally, the constraints are modified to reflect the deletion of a device. In contrast, addition of devices extends the resource model. For each new device added, we also model the operating states of the device (Figure 2). Following resource modeling we identify the tasks that can be mapped onto the newly added device and the operating states (of the device) that can execute the mapped task and add choices to the application model accordingly. The constraints are also modified if the newly added device cannot be used with some other existing device. Resource model promotes reuse [9]. If a library of device models exists and contains the model for any of the target devices then we do not build the model and instead use the existing model from the library.

Another class of constraint supported by MILAN is the performance constraints. For example, if the design problem is to identify the design with lowest energy dissipation that satisfies a given latency constraint then the latency constraint is of the form “total latency of the application < 1000 ms”. This allows us to vary performance constraints by modifying the specification. Latency constraints can also be specified for a subset of tasks that are executed in a serial order [3]. This feature allows us to separately constrain to total allowable latency of groups of subtasks.

MILAN uses a hierarchical approach for design space exploration (DSE). The basic idea of hierarchical approach is a two-step DSE [9]. Step-I uses a heuristic based design space exploration technique to quickly evaluate a large design space and identify a set of design that satisfy the given performance and design constraints. Step-II uses a high-level performance estimator to evaluate the selected designs to identify the most energy efficient design. For the class of problems discussed in this paper, we use DESERT for Step-I and HiPerE for Step-II.

DESERT does not support duty cycle based design space exploration. However, DESERT is very efficient in applying design and performance constraints over extremely large design spaces. Our experience with DESERT shows that we can prune a design space with approximately $10^{20} \sim 10^{40}$ designs in order of minutes [7]. As our model specification includes all the available devices, multiple algorithms for implementing each task, and multiple operating states for each device, the design space is very large due to combinatorial expansion. Therefore, we use DESERT to apply the design and performance heuristics to identify only the designs that are valid and meet the given performance criteria. Our original focus is to identify energy efficient designs based on duty cycle specification that meets a given latency constraint while processing one input frame and uses a valid combination of target devices. DESERT allows us to select designs based on the last two criteria. We use HiPerE to evaluate the selected designs based on the third criteria, duty cycle specification, to identify the most energy efficient one. The designs rejected by Step-I are the ones which do not satisfy the given design constraints and the latency constraint. Therefore, our approach guarantees that none of the rejected designs would have satisfied the given performance requirement even when duty cycle is considered [9].

HiPerE supports three optimization techniques: I) shut down the devices and start up, II) move to a low power state, and III) leave as it is during inactive phases. HiPerE also supports an aggressive mode for optimization which when selected ensures that the optimization techniques I and II are applied only if the duration of the inactive phase allows moving to and from the shut down or low power state. In addition, duty cycle is evaluated based on the rate of input and latency of execution when the application processes one input frame. HiPerE derives the required input to evaluate different designs based on duty cycle specification from the application and resource models. Therefore, HiPerE (also DESERT) are integrated into the MILAN environment through appropriate model interpreters. Therefore, once the models, mappings, and the constraints are specified, the hierarchical design flow is completely automated. In the following section, we demonstrate the use of MILAN to design a target detection and tracking system.

3. Illustrative Example

The design problem identifies an energy-efficient hardware for a target detection application [13] from a set of devices. Target detection is widely used in radar applications, surveillance videos, and sensor networks to predict position and velocity. The application consists of 6 tasks (Figure 1). The hardware needs to be selected from a set that consists of Xilinx Virtex-II, Actel ProASIC^{PLUS}, Intel PXA 255, PowerPC 405, and TI C6711 DSP. Figure 1 shows the hierarchical data flow modeling with a 6-task model at one level and 4 choices for a task at the second level. While we have 5 devices, the task has only 4 choices because one FPGA was not large enough to accommodate the required floating-point arithmetic. For example, *inverse* and *whiten* needs to be computed using floating-point arithmetic for which Actel ProASIC^{PLUS} is not a suitable choice. Such requirements are indicated by not allowing inverse and whiten to be mapped onto ProASIC^{PLUS} while modeling. Additionally, the 5 valid device combinations are, 1) Virtex-II only, 2) DSP only, 3) ProASIC + DSP, 4) ProASIC + PXA 255, and 5) ProASIC + PPC 405. The valid combinations are provided as constraints to MILAN. Figure 2 shows the device models and modeling of operating state of one device.

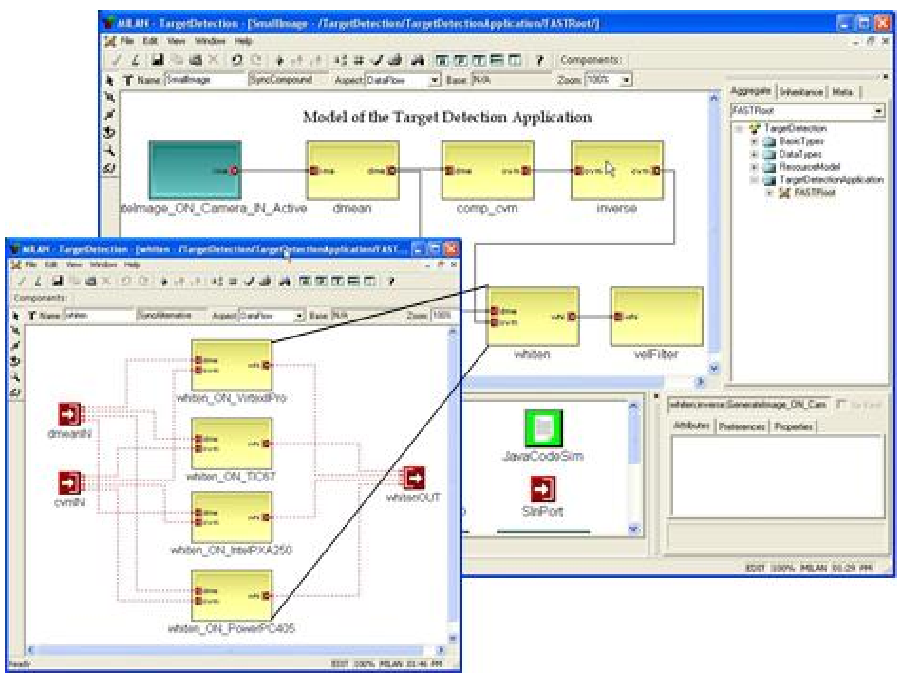

Figure 1: Hierarchical Application Model

Following modeling, we performed design space exploration using DESERT and HiPerE. Through several iterations using DESERT, energy constraint of < 900 milliJoules was chosen to have DESERT select 5 designs as output of Step-I. The size of the initial design space was approximately 73,300. Once 5 designs were identified by DESERT, we used HiPerE to perform Step-II to identify the best design that meets the duty-cycle requirements and dissipates the minimum energy. Figure 3 shows the HiPerE output for the 5 designs. The last two columns have the latency and energy estimates for the 5 designs for a given duty cycle specification. Figure 3 also shows the HiPerE input window that allows the designer to specify duty cycle. HiPerE supports a) evaluation for a period of time or a number of input frames, b) variable rate application where different tasks execute at a different rate, c) different rates of input, d) multiple optimization options for reduction of energy dissipation, e) evaluation of different memory modules, etc. HiPerE automatically extracts all required information from the model.

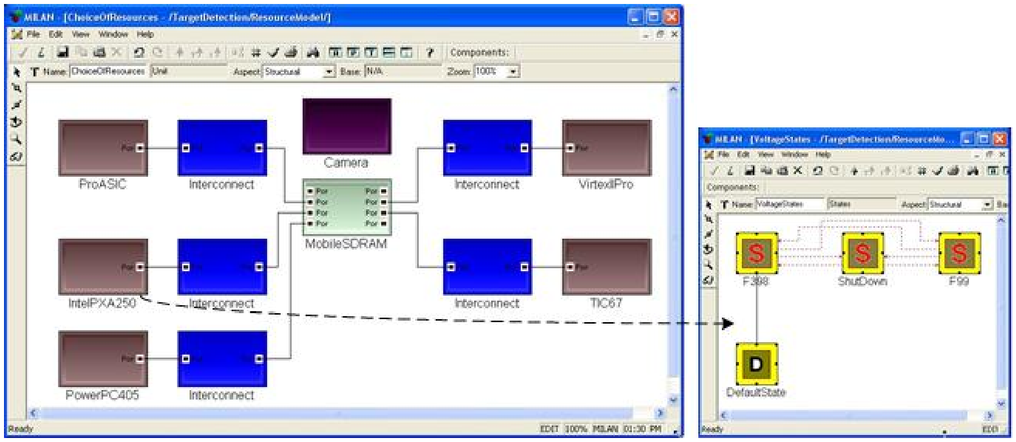

Figure 2: Multi-level Resource Model

Using MILAN we identified the most energy efficient device for different duty cycle specifications. For example, for a sample duty cycle specification, rate of CVM computation and inverse as 2 (once every 2 cycles) and input rate as 0.5 Hz. (1 input frame per 2 seconds), the design that uses the Actel ProASIC^{PLUS} and the TI DSP was identified as the most energy efficient one. In contrast, for input rate of 2 and 3 Hz, the design that uses the Virtex-II FPGA was identified as the most energy efficient one. DSE using our framework was performed using a PC with an 848 MHz Pentium III. The use of an optimization heuristic in Step-I allows us to evaluate the above design space in less than a minute. Step-II, evaluating 5 designs, took about 1 minute.

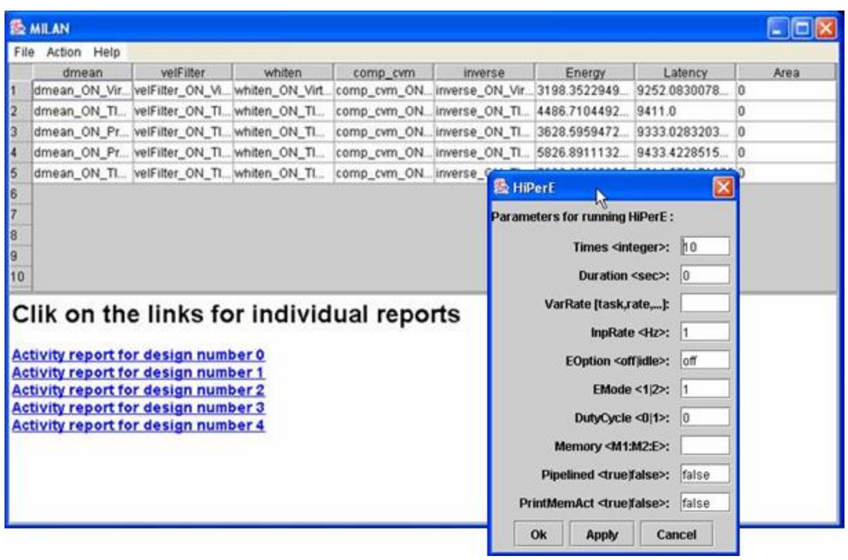

Figure 3: Input window and Output of HiPerE

4. Discussion

The MILAN design environment is a publicly available open source software[2]. Applying the concept of MIC, MILAN provides an extensible and integrated environment for modeling and simulator and tool integration. As discussed earlier, MILAN supports application and resource modeling and mapping and constraint specification. MILAN also supports functional simulation. Functional simulation and verification may be done iteratively with application modeling [3]. To enable functional simulation, MILAN supports generation of high-level source code in C and Matlab and integration of functional simulators [3][6]. MILAN also supports specification of input stimulus at the source tasks and output processing logic at the sink tasks. These capabilities are also exploited for simulation using integrated

[2] MILAN can be downloaded at http://www.isis.vanderbilt.edu/Projects/milan/

simulators to estimate performance. However, the source code generation capability of MILAN needs specification, by the user, of high-level implementation of the choices associated with each task. This can be a tedious job but we feel it is fair because whenever a designer needs to evaluate an implementation at some point high-level source code based implementation is necessary to correctly estimate performance.

Before we can perform design space exploration, we need to populate the design space described above using the performance estimates for all the mappings specified in the model. Performance estimates can be obtained in any manner; simulation, estimation, past experiences. MILAN is a simulator integration environment [1]. Hence, if appropriate simulators are integrated, MILAN has the capability to perform automatic simulation (using specified implementation and sample input) and update the model for mapping using the simulation results. Therefore, the accuracy of the results produced by MILAN is highly dependent on the types of simulators integrated. While MILAN allows integration of simulators, many simulators require compilers (e.g. FPGA compilers [5][15]) to generate the executable code. MILAN can potentially integrate compliers as well. However, we have not explored this option a lot and not all commercial compilers (and simulators) can be integrated into a publicly distributable environment.

5. Conclusion

We discussed a Model Integrated Computing based design environment that is used to design high-performance low-power embedded systems through device selection and mapping. Extensive use of models allows us to accommodate evolving performance requirements, target hardware specification, and mission scenarios while promoting design reuse. Our design environment can model the target devices, application, mapping between the application tasks and the devices, and performance and design constraints and configure the devices based on the performance characteristics and different operating states. The design environment also allows us to compose solutions (designs) and analyze them to identify suitable designs based on the performance requirements.

Acknowledgement: We acknowledge Akos Ledeczi, James R. Davis, and Sandeep Neema of Institute for Software Integrated Systems, Vanderbilt University for their contributions to the MILAN project.

References

[1] A. Agrawal, et. al, "MILAN: A Model Based Integrated Simulation for Design of Embedded Systems," Language Compilers and Tools for Embedded Systems, 2001.
[2] C. Dick, “The Platform FPGA: Enabling the Software Radio" SDR Forum.
[3] A. Ledeczi, J. Davis, S. Neema, and A. Agrawal, “Modeling Methodology for Integrated Simulation of Embedded Systems," ACM Transactions on Modeling and Computer Simulation, January 2003.
[4] Generic Modeling Environment (GME). http://www.isis.vanderbilt.edu/projects/gme/
[5] Mentor Graphics FPGA Advantage. http://www.mentor.com/fpga-advantage/
[6] Model-based Integrated Simulation. http://milan.usc.edu/
[7] S. Mohanty, V. K. Prasanna, S. Neema, J. R. Davis, "Rapid Design Space Exploration of Heterogeneous Embedded Systems using Symbolic Search and Multi-Granular Simulation," Intl. Conf. On Language Compilers and Tools for Embedded System, 2002.
[8] S. Mohanty and V. Prasanna, "Rapid System-Level Performance Evaluation and Optimization for Application Mapping onto SoC Architectures," IEEE Intl. ASIC/SOC Conference, 2002.
[9] S. Mohanty and V. Prasanna, “A Hierarchical Approach for Energy Efficient Application Design Using Heterogeneous Embedded Systems," Compilers, Architecture, and Synthesis for Embedded Systems, 2003.
[10] T. Mudge, “Power: A First Class Design Constraint,” Computer, vol. 34, no. 4, April 2001.
[11] S. Neema, "System Level Synthesis of Adaptive Computing Systems," Ph.D. Dissertation, Vanderbilt University, Department of Electrical and Computer Engineering, May 2001.
[12] Power Aware Sensing Tracking and Analysis. http://pasta.east.isi.edu/
[13] P. Singer, ``The Optimal Detector," SPIE Conference: Signal and Data Processing for Small Targets, 2002.
[14] J. Sztipanovits and G. Karsai, “Model-Integrated Computing," Computer, Volume: 30 Issue: 4, April 1997.
[15] Xilinx Virtex-II and Xilinx System Generator for DSP. http://www.xilinx.com/
[16] XScale: Intel PXA255 Processors. http://www.intel.com/design/intelxscale/