

# A Hierarchical Approach for Energy Efficient Application Design Using Heterogeneous Embedded Systems\*

Sumit Mohanty  
University of Southern California  
3740 McClintock Ave.  
Los Angeles, CA 90089  
smohanty@usc.edu

Viktor K. Prasanna  
University of Southern California  
3740 McClintock Ave.  
Los Angeles, CA 90089  
prasanna@usc.edu

## ABSTRACT

Several features such as reconfiguration, voltage and frequency scaling, low-power operating states, duty-cycling, etc. are exploited for latency and energy efficient application design using heterogeneous embedded systems. However, more choices during application design results in a large design space that must be traversed efficiently. In this paper, we propose a hierarchical methodology that integrates optimization heuristics, high-level performance estimators, and low-level simulators to enable efficient exploration of large design spaces. Our methodology is fast, robust against approximation errors due to high-level modeling, and can evaluate a much larger design space than an optimization heuristic only design space exploration technique. We have augmented MILAN, a model-based integrated simulation framework for embedded systems, to develop an environment to perform hierarchical design space exploration. Using our methodology for a beamforming application, we identify an energy-efficient mapping onto a heterogeneous embedded system while meeting a given latency constraint. We also demonstrate the use of our methodology in identifying an energy and latency efficient heterogeneous embedded system based on user-specified performance requirements for a personnel detection algorithm from a set of devices that includes FPGAs, DSPs, and traditional processors.

## Categories and Subject Descriptors

I.6.3 [Computing Methodologies]: Simulation and Modeling—*general, design space exploration*

## General Terms

Algorithms, Performance

\*This work is supported by the DARPA Power Aware Computing and Communication (PAC/C) Program under contract F33615-C-00-1633 monitored by Wright Patterson Air Force Base.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CASES'03, Oct. 30–Nov. 2, 2003, San Jose, California, USA.  
Copyright 2003 ACM 1-58113-676-5/03/0010 ...\$5.00.

## Keywords

design space exploration, energy efficiency, heterogeneous embedded systems, performance estimation

## 1. INTRODUCTION

Heterogeneous embedded systems integrate one or more processing components such as traditional processors, DSPs, and FPGAs and peripheral units such as memories, sensors, and actuators onto a single system. A heterogeneous embedded system can be a system-on-chip solution like Xilinx Virtex-II Pro [20] which integrates traditional processing cores into the reconfigurable fabric or a multi-chip solution like the PASTA sensor stack [15] which integrates processors, micro-controllers, custom logics, sensors, and actuators. Such heterogeneous systems support various power-aware capabilities/features such as reconfiguration, dynamic voltage and frequency scaling, low-power operating modes, efficient start up and shut down, among others. The ability to dynamically customize the architecture or the operating modes of different components to match the computation, data flow, and quality of service (QoS) requirements of an application has demonstrated significant performance benefits such as lower latency and energy dissipation [13]. As a result, heterogeneous systems are well-suited for latency and energy efficient implementation of compute intensive applications and thus are attractive for signal processing applications used in wired as well as wireless mobile devices.

Application design using heterogeneous embedded systems involves (device) selection of a suitable processing components and memory. Following device selection, mapping between individual application tasks and processing components, appropriate operating state for each mapping, and the schedule of execution are identified based on the performance requirements specified as an input. If the target system is already specified, application design does not involve device selection. With the availability of multiple implementation platforms such as FPGAs, traditional processors, and DSPs, a designer not only needs to identify suitable platforms but also appropriate hardware/software partitioning and mapping onto those platforms. In addition, other capabilities that play a significant role especially for energy efficient design are reconfiguration, dynamic voltage scaling, choice of low-power operating states, and device activation scheduling. However, a large number of choices during application design results in a large design space that must be traversed efficiently to identify the designs that meet

the performance requirements. In addition, due to complex inter-dependencies, the design parameters can have conflicting effect on different performance metrics making manual design and analysis impractical.

There exists many design approaches to address the above issues. Each approach targets an application design problem domain (henceforth referred to as problem domain). A problem domain refers to all application design problems specific to a class of applications and hardware. For example, mapping the class of applications that can be modeled as linear array of tasks onto reconfigurable devices is a problem domain. Different design approaches can be classified into two major categories. The first category is *optimization heuristics*. Approaches that fall under this category are based on a high-level abstraction (model) of a problem domain and allow development of provably optimal solutions. Such high-level models are a mathematical abstraction of the application behavior and hardware characteristics without the underlying implementation details (and therefore runtime complexities). While such approaches can quickly identify an optimal solution, they are sensitive to errors introduced due to approximations during high-level modeling. The second category is *simulation* based design space exploration. Approaches that fall under this category use simulators that are based on execution models suitable for cycle-accurate or register-transfer level simulation. While simulations provide reliable results in terms of accuracy, design space exploration using such an approach consumes a significant amount of time due to low simulation speeds.

In contrast, our approach, hierarchical methodology for design space exploration, integrates best of both worlds. Our methodology consists of two steps. The first step uses optimization heuristics that operate on the initial design space and prune it to a smaller set of designs based on the performance requirements. The second step uses high-level estimation tools that operate on the designs identified in the first step. A high-level estimation tool operates at a higher level of abstraction than a typical simulator such as cycle-accurate, register-transfer level, or even instruction-set simulators. The precise definition of “level” for a high-level estimation tool depends on the target problem domain. The high-level estimation tools used in our methodology utilize available low-level simulators to estimate performance. However, the use of a high-level estimator allows us to reuse low-level simulation results and considerably reduces the number of low-level simulations that need to be performed.

Many design tools are available for a specific target device [3, 17, 20]. For example, Xilinx EDK and System Generator for Simulink [20] are design tools for Xilinx Virtex series of FPGAs. However, in our case the target device is not specified. We start with an application specification and a set of performance requirements. Performance requirements can be a hard latency requirement or energy dissipation minimization or minimization of energy dissipation subject to a latency constraint. Typical applications in such category include high performance signal processing applications which need to be deployed in power-constrained environments [15, 18]. We do not assume an underlying architecture and instead evaluate a set of available components, typically commercial-off-the-self (COTS), to identify an architecture that minimizes energy dissipation while meeting the latency requirements. In addition to architecture, we also identify the corresponding mapping of the application

onto the architecture. We are not aware of any commercially available tool that can handle such a wide range of devices.

We have enhanced the MILAN framework to support hierarchical design space exploration. MILAN is a Model based **I**ntegrated **s**imu**L**atio**N** framework for embedded system design and optimization through integration of various simulators into a unified environment [10]. MILAN already supports modeling of applications as a synchronous data flow (SDF) graph, modeling of the various capabilities of the heterogeneous devices, an ordered-binary decision diagram based design space exploration tool (DESERT), and a high-level performance estimator (HiPerE) for heterogeneous embedded system [12]. We have enhanced MILAN by adding a dynamic programming based N-optimization tool (see Section 2), augmenting HiPerE with capabilities to support memory, sensors, actuators, and duty-cycle (see Section 3) while estimating performance. We have also developed a graphical front-end to HiPerE to enable interactive design space exploration in the second step of our methodology to allow efficient performance trade-off analysis of the designs identified by the optimization heuristics.

The paper is organized as follows. The following section discusses the hierarchical design space exploration (DSE) methodology. The implementation of our methodology in MILAN is discussed in Section 3. Section 4 provides two examples demonstrating the use of our methodology. Some related works are discussed in Section 5 followed by conclusion in Section 6.

## 2. HIERARCHICAL DESIGN SPACE EXPLORATION

We focus on energy-efficient design of signal processing applications using heterogeneous embedded systems. Typical signal processing applications such as MPEG decoder and encoder, software defined radio, image and acoustic signal processing for automated target recognition and tracking, etc. can be modeled using synchronous data flow graphs [8]. Hence, our focus is on the class of applications that can be modeled using SDF graphs. A simplified version of SDF is a linear data flow which models an application as an ordered set of tasks where each task can have at most one input and one output. Due to a simple and regular structure, linear data flow is well suited for formal algorithmic analysis and optimization. We also focus on developing specialized solutions for linear data flow graphs.

In our previous work, we have defined a model called GenM for system-on-chip architectures [11]. GenM models dynamic voltage/frequency scaling for the processor, power states for the memory, and reconfiguration for the reconfigurable logic (FPGA). Additional details of the GenM model can be found in [11]. For our purpose, we have extended GenM to include multiple processors and FPGAs. Our target hardware resources are the class of heterogeneous embedded systems that can be modeled using the extended GenM model.

### 2.1 Key Ideas of Our Methodology

In order to discuss our methodology, we define *parameter coverage* as a metric to compare different design space exploration (DSE) techniques. A set of parameters are associated with each application design problem domain. A parameter can be a variable or a constant. For example,

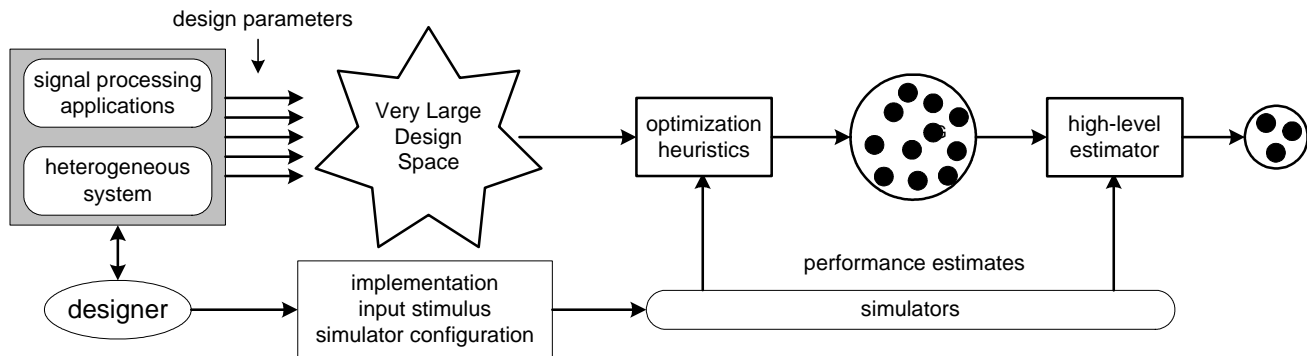


Figure 1: Hierarchical design space exploration

a linear array of tasks and a processor supporting dynamic voltage scaling is an application design problem domain. For this domain, the parameters are task execution cost for each voltage setting, operating voltages of the processor, voltage scaling cost, mapping choices, cost of data transfer between tasks, over-all latency, and energy dissipation. For a given problem domain, there exists several optimization heuristics, high-level estimators, and simulators that can be used to perform design space exploration. For a given solution, be it an optimization heuristic or an estimation/simulation tool, parameter coverage refers to the set of parameters that are considered by each solution. Higher parameter coverage refers to a larger set of parameters and results in higher accuracy but can potentially be time consuming during DSE. A low-level simulator is an example of a performance estimation tool with high parameter coverage. In contrast, optimization heuristics, due to high-level of abstraction, tend to have lower parameter coverage.

Given an application design problem domain, hierarchical design space exploration is defined as a two step process (Figure 1). The first step uses a suitable optimization heuristic for the problem domain that can generate a set of designs meeting the given constraints. The second step consists of a suitable high-level performance estimator that can evaluate the performance of any design that is a potential solution for the given problem domain. We assume that the high-level performance estimators have a higher parameter coverage than the optimization heuristics. Therefore, the high-level estimation tool can cover the parameters not included in the model used by the optimization heuristic due to approximations. Hence, our methodology can explore a larger design space than an optimization heuristic only DSE scheme. Hierarchical design space exploration assumes the availability of appropriate simulators to estimate the model parameters required by the optimization heuristics and the high-level estimators. We also assume that appropriate input such as implementations in scripts or languages supported by the simulators, input stimulus, and simulator configurations are available to perform simulation.

## 2.2 Advantages of Our Methodology

The primary difference between our version of an optimization heuristic and the traditional version is generation of a set of designs as opposed to a single optimal design. The set of designs consists of the designs that meet the given performance constraints or are the N-optimal solutions. N-

optimal solutions refer to  $N$  best designs based on a single performance metric such as latency or energy dissipation. An N-optimization heuristic is a technique that generates N-optimal solutions for an application design problem domain. By ensuring that we have a set of good designs as opposed to one optimal design, we increase the chances of finding the *real-optimal* design from the set even when approximated high-level models are used. An optimal design is the best design identified (based on the performance requirements) by the optimization heuristic using the underlying approximated high-level model. A real-optimal design is the design that is optimal when the designs are implemented using hardware and performance is measured. For ease of comparison, we assume that the most detailed low-level simulator available is accurate and can be used to identify the real optimal solution. We also assume that the errors induced by approximations are marginally low when compared with the actual performance values. The advantages of hierarchical design space exploration are as follows:

- robust against approximation errors due to high-level abstractions (models) used by the optimization heuristics
- reduces the number of simulations necessary when compared against simulation based design space exploration
- combines the speed of optimization heuristic based DSE and the higher accuracy and parameter coverage of simulation based DSE
- designer can combine different optimization heuristics and high-level estimators to suite the need of target application design problem domain

In the following, we perform an experiment to demonstrate the robustness of our hierarchical design space exploration methodology against approximation errors.

## 2.3 Robustness Against Approximation Errors

Our objective in this experiment is to demonstrate the effect of error on design space exploration and the advantages of a hierarchical design methodology over both optimization heuristic and simulation based techniques. The design space exploration problem we consider is “given a linear array of tasks and a reconfigurable device, a set of configurations for

**Table 1: Robustness Against Approximation Errors**

size of design space	256	3125	46656	823543	observations
(tasks, configurations)	(4,4)	(5,5)	(6,6)	(7,7)	(latency)
up to	4535	6363	7625	7889	real-optimal design
10%	4535	6823	7625	8044	optimal design using dynamic prog.
error	4535	6363	7625	7889	optimal design using our method
	Yes	Yes	Yes	Yes	was real-optimal in the N-optimal set?
up to	4691	6031	7406	8325	real-optimal design
15%	4895	6075	7651	8354	optimal design using dynamic prog.
error	4691	6031	7499	8424	optimal design using our method
	Yes	Yes	No	No	was real-optimal in the N-optimal set?
up to	4705	6160	6849	8669	real-optimal design
20%	5117	6230	7109	9370	optimal design using dynamic prog.
error	4988	6230	6849	9116	optimal design using our method
	No	No	Yes	No	was real-optimal in the N-optimal set?
up to	4755	5850	6680	8567	real-optimal design
25%	4876	6305	6851	8800	optimal design using dynamic prog.
error	4755	5850	6783	8790	optimal design using our method
	Yes	Yes	No	No	was real-optimal in the N-optimal set?

each task, time taken to execute each task in each configuration, reconfiguration cost for each pair of configurations, find a mapping of the tasks and configurations with the minimum latency”.

We compare the well known dynamic programming based solution using assembly-line scheduling algorithm [4] and our hierarchical approach that uses an N-optimal solution for the linear array mapping problem and HiPerE (see Section 3). In the experiment, we introduce a certain amount of error in the latency estimates and compare the optimal design identified by both the approaches with the real optimal design. The real optimal design is identified based on the estimates without any error using an exhaustive search of the complete design space. The latency estimates for task execution and reconfigurations are generated randomly. By introducing random errors, we are able to simulate approximation errors encountered while modeling using a high-level abstraction. We also verify if the real-optimal design was included in the N-optimal (we chose  $N = 20$  for our experiments) solutions identified by step one of our methodology. We experiment using 4 different applications (linear array of tasks) containing 4, 5, 6, and 7 tasks each and 4, 5, 6, and 7 numbers of configuration for each task respectively. Table 1 shows the results. We evaluate each instance of the linear array of tasks with 10%, 15%, 20%, and 25% approximation errors. The latency values shown in the table is based on the actual latency estimates for the designs. However, the designs are identified based on the latency estimates with error.

As shown in Table 1, out of the 16 experiments, the dynamic programming is able to identify the real-optimal design only in 2 cases whereas using our methodology we are able to identify the real-optimal design in 9 cases. Even when our methodology fails to identify the real-optimal design, the optimal designs identified by our method is on average 2.7% (max 6%, min 1.2%) worse than the real-optimal design. In contrast, the same, for results obtained using dynamic programming solution is on average 3.8% (max 8.7% and min 0.34%). While we do not evaluate the time needed for simulation based DSE, the size of the design space indi-

cates the magnitude of the simulation time needed to perform all the simulations.

## 2.4 Design Flow for Hierarchical Design Space Exploration

In order to use hierarchical design space exploration, the designer needs to follow the design flow as described in the following (Figure 2). Once the application design problem domain is identified, the designer needs to identify a suitable optimization heuristic and a high-level performance estimation tool. Given a specific application design problem, the designer needs to define models for the application and the target hardware or hardware choices. A mapping model, containing feasible mappings, is defined using the application and hardware model. In addition, performance constraints for latency and energy dissipation are also specified in the model. These performance constraints are defined such that a set of designs can be chosen using the optimization heuristic. Following model definition and constraint specification, the optimization heuristic is used to identify the designs that meet the performance constraints. The selected designs are evaluated using the high-level performance estimator to identify the design that meets the performance requirements of the design. Note the difference between performance requirements and performance constraints. Performance requirement is specified as part of the design problem specification and refers to the requirements that the resulting design needs to satisfy. Performance constraints are specified as input to certain type of optimization heuristics (e.g. DESERT) such that a set of designs can be selected using the first step. In contrast, N-optimization heuristics do not need performance constraints.

We have enhanced the MILAN embedded system design environment to implement our methodology. In the following section, we discuss the enhancements and highlight various support that MILAN provides to implement our methodology.

### 3. HIERARCHICAL DESIGN SPACE EXPLORATION USING MILAN

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 [10]. Using the MILAN environment, the designer formally models the target application, underlying hardware, and constraints (latency, throughput, energy dissipation, etc.) through a graphical interface provided by MILAN. The models are stored in a model database. 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 [5, 7]. The Generic Modeling Environment (GME) is a configurable graphical tool suite supporting MIC [5]. GME allows the designer to create domain-specific models [10]. A metamodel (modeling paradigm) is a formal description of model construction semantics. Once the metamodel is specified by the user, it can be used to configure GME itself to present a modeling environment specific to the problem domain. 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.

#### 3.1 Overview of the MILAN Design Flow

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 [7]. The functional specification of the target system specifies the structure of the data-flow graph and the choice of implementations specifies the alternatives. For multi-rate applications, while modeling an application in MILAN, the designer can specify the rate of execution for each application task. MILAN supports hierarchical modeling that enables hierarchical specification of the data flow graph making it easier to manage and analyze an application model. Functional simulation and verification may be done iteratively with application modeling [7]. To enable functional simulation, MILAN supports generation of high-level source code in C and Matlab and integration of functional simulators [7]. 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 simulators to estimate performance.

Resource modeling involves modeling of the target architecture. The modeling paradigm is based on the GenM model [11]. The user identifies key components and features of the target architecture that can be exploited for optimization and models them in MILAN. 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 [12].

Finally, the user describes possible mappings for each task alternatives and different performance or compositional constraints that the system needs to satisfy [11, 12]. 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 in-

put. 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. 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”. 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. MILAN is a simulator integration environment. 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.

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 [12]. Our experience with DESERT shows that we can prune a design space with approximately  $10^{20} \sim 10^{40}$  designs in order of minutes [12]. DESERT uses symbolic methods based on Ordered Binary Decision Diagrams (OBDDs) for constraint satisfaction [7]. We have also integrated other techniques based on dynamic programming to provide additional DSE options to the designer. One such technique targets applications that can be modeled as a linear array of tasks (discussed in Section 3.3). Given a linear array of tasks, execution cost of each task on a target reconfigurable device, and reconfiguration cost, the technique can identify a set of mapping with minimum latency or energy. The output of DSE is evaluated using High-level Performance Estimator (HiPerE) tool, 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. Our results for signal processing applications show that HiPerE estimates are within 8% of the estimates using low-level simulations [11, 12]. Using HiPerE, the designs are evaluated and compared manually to identify the final design.

MILAN, in its current form, supports modeling of applications using synchronous data flow graphs and heterogeneous embedded systems using the extended GenM model [7, 11]. Hence, MILAN is suitable for our domain of interest which is design of signal processing application using heterogeneous embedded systems. In addition, MILAN already integrates DESERT and HiPerE [11]. However, MILAN did not support hierarchical design space exploration because the parameter coverage for both DESERT and HiPerE were the same. In addition, MILAN models were not capable of storing a set of designs (output of step one) and therefore there was no easy method of applying step two of our methodology in MILAN. Further, there were no support for generating N-optimal solutions for any class of application design problems. Hence, we augmented MILAN in many ways to implement and evaluate the hierarchical design exploration methodology. We discuss them in the following.

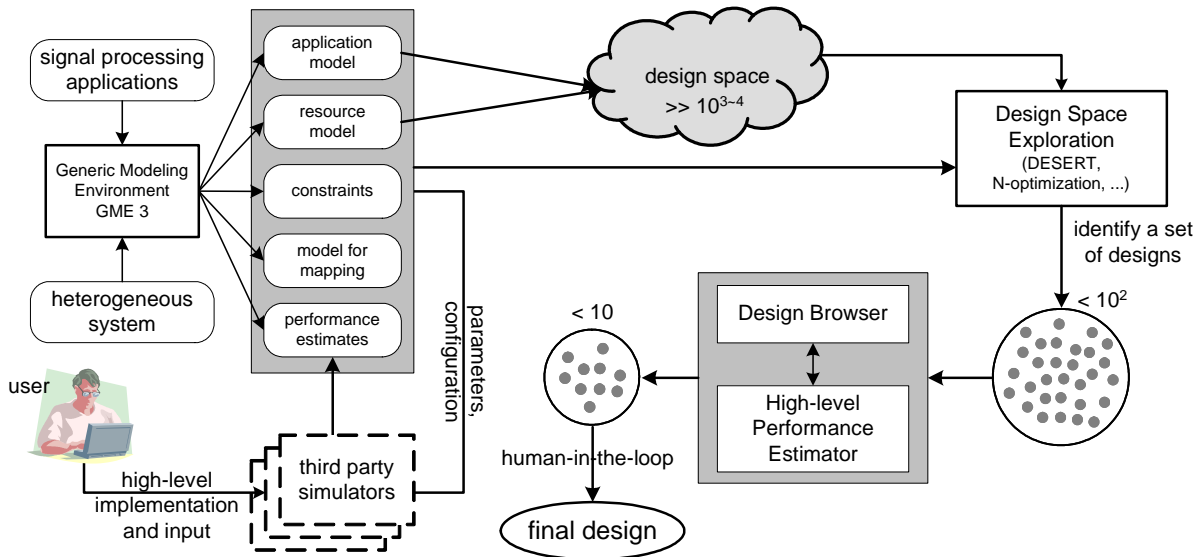


Figure 2: Design flow using MILAN

### 3.2 High-level Performance Estimator (HiPerE)

HiPerE is a high-level performance estimator based on the GenM model [11]. Application specification is provided as a SDF graph to HiPerE. HiPerE combines component specific performance estimates (as specified in the mapping information) through interpretive simulation to derive system-level performance estimates. Additional details about HiPerE can be found in [11]. For hierarchical design space exploration, we enhanced HiPerE to support multiple processors, reconfigurable devices, memories while estimating performance. In addition, support for duty-cycle while estimating performance was also added. Duty-cycle in the context of application execution refers to the proportion of time during which a component, device, or system is operated. Support for duty-cycle includes being able to estimate performance for a length of time or number of execution instances while taking into account, start up and shut down cost, idle energy dissipation, and rate of input.

In addition, a duty-cycle aware estimator needs to support applications with multi-rate execution. An application modeled as a set of tasks is said to be multi-rate if different tasks have different rate of execution. A multi-rate application needs to adapt based on the input or environment condition. In the second illustrative example (Section 4.2), co-variance matrix is updated once every constant number of samples to adapt to the changing signal environment [18]. Hence, we enhanced HiPerE to estimate performance of different execution instances based on rate of execution of individual tasks. The second illustrative example discussed in this paper (Section 4.2) deals with a multi-rate application.

HiPerE also addresses other important issues related to the design of heterogeneous embedded systems. MILAN facilitates seamless integration of different simulators and invocation all integrated simulator from a single modeling environment. However, due to the complexity of the target heterogeneous systems, either there is a lack of appropriate system-level simulator that can simulate the complete system, or if such a simulator exists it is prohibitively ex-

pensive in terms of simulation time. HiPerE addresses both the above issues through integration of individual simulation results and rapid performance estimation [11].

### 3.3 Dynamic Programming based N-Optimization Heuristic

We have enhanced MILAN to integrate an N-optimization heuristic that identifies N-optimal solutions for a class of applications that can be modeled as a linear array of tasks. A linear array of tasks consists of an ordered set of tasks with each task having at most one input or output. There is only one source task (one output and no input) and one sink task (one input and no output). The application is to be mapped onto a target device that supports multiple operating states. An operating state can be a configuration in case of reconfigurable devices or an operating voltage (frequency) in case of processors supporting dynamic voltage (frequency) scaling. Each task can be executed in different operating states. Hence, each task is associated with a unique performance estimate (energy dissipation and latency) for each operating state. Transitions between the operating states are associated with transition costs (energy dissipation and latency) which depend on the source and destination states. In the above scenario, a design is referred to as a set of operating states; one operating state per task. Performance of a design is the sum of the execution cost of each task in the corresponding operating state and the state transition cost between task executions (Figure 3). The N-optimization problem can be formally defined as, given a linear array of tasks, a device supporting multiple operating states, and performance estimates for task executions and state transitions, find a set of designs with  $N$  lowest values for latency or energy dissipation. The proof of complexity and correctness are provided below.

Let the set of  $n$  tasks be  $T'_1 T'_2 \dots T'_r$  and the set of  $m$  possible operating states be  $S_1 S_2 \dots S_m$ . Our focus is to minimize energy dissipation while solving the N-optimization problem. Minimization of latency can also be analyzed in the same fashion. We use dynamic programming to identify

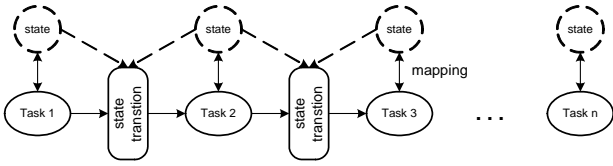


Figure 3: Linear array of tasks and state transition

N-optimal solutions. In the solution space of the optimization problem considered, a sequence of configurations for a given sequence of tasks may occur as part of a solution to multiple larger sequences. Dynamic programming is utilized to compute the optimal solution for the complete sequence of tasks by using solutions to smaller subsequences. Once N-optimal solutions for executing up to task  $T_i$  is determined, the energy dissipation for executing up to task  $T_{i+1}$  can be determined. This approach is used recursively to compute the N-optimal solution. We define  $min_N()$  as a function that given a set of values identifies N smallest values.  $min_N()$  is a selection problem and can be solved in  $O(n)$  time if the number of inputs is  $n$  [4].

*Theorem:* Given a sequence of tasks  $T'_1 T'_2 \dots T'_r$ , and  $m$  possible states  $S_1 S_2 \dots S_m$ , N-optimal solutions (sequence of states for executing these tasks) with minimum energy dissipation can be computed in  $O(rm^2N)$  time.

*Proof:* We use the dynamic programming approach to compute the N-optimal solutions. We define the energy dissipation values for N-optimal solutions for execution up to task  $T'_i$  ending in state  $S_j$  as  $E_{ijk}$  where  $k = 1 \dots N$ . Let  $E_{ij}$  be a set of N values  $E_{ij1} \dots E_{ijN}$ . We initialize the E values as  $E_{0jk}, \forall j : 1 \leq j \leq m$  and  $\forall k : 1 \leq k \leq N$ .

We assume that N-optimal solutions for executing up to task  $T'_i$  ending in all possible states,  $S_i$ , are computed. Now for each of the possible states ( $S_j \in S$ ) in which we can execute  $T'_{i+1}$ , we have to compute N-optimal solutions for sequence of states ending in that state  $S_j$ . This is computed as follows.

$$E_{i+1j} = \min_N(\{V_{kl} \mid V_{kl} = e_{i+1j} + E_{ikl} + q_{kj}\})$$

$$\forall k : 1 \leq j \leq m, \forall l : 1 \leq l \leq N \forall j : 1 \leq j \leq m$$

Thus we examine all possible ways to execute  $T_{i+1}$  once we have finished executing  $T_i$ . If each set of values  $E_{ik}$  is N-optimal then the values  $E_{i+1j}$  is also N-optimal.

Computation of each N-optimal set of values takes  $O(mN)$ . Since there are  $O(rm)$  sets of values to be computed, the total time complexity is  $O(rm^2N)$ .

*Proof of correctness:* Let us assume that there exists a state sequence  $\Pi$  executing up to task  $T'_{i+1}$  ending in state  $S_j$  (refer to the proof above) for which the energy dissipation is less than one of the N-optimal solution computed for execution up to task  $T'_{i+1}$  ending in state  $S_j$ . Let  $S_k$  be the previous state in the state sequence  $\Pi$  during the execution of task  $T'_i$  and  $\Pi - S_j$  is the sequence of states for execution up to task  $T'_i$ . If this sequence of states (solution) is part of the N-optimal solutions  $E_{ik}$ , then there is a contradiction as we must have evaluated state sequence  $\Pi$  while searching for a solution ending in  $S_j$  executing up to task  $T'_{i+1}$ . Conversely, if the sequence  $\Pi - S_j$  is not part of the N-optimal solution, then energy dissipation for  $\Pi$  cannot be less than  $E_{ikl} + q_{kj} + t_{i+1j} \forall l : 1 \leq l \leq N$  as  $E_{ik}$  is the N-optimal solu-

tion for task execution up to  $T'_i$  ending in state  $S_k$ . Hence, the optimal solutions selected are the N-optimal solutions.

MILAN already supports modeling of a linear array of tasks and a reconfigurable device with a set of configurations and reconfiguration costs. Therefore, an implementation of the above heuristic can be integrated into MILAN.

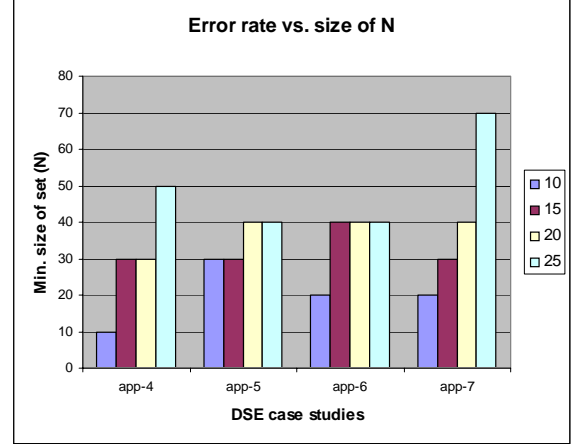


Figure 4: Effect of error rate on N

While using the N-optimization heuristic, the designer specifies the value for N. As discussed earlier, by selecting a set of designs using the optimization heuristics, we overcome the effect of error due to high-level modeling. Figure 4 summarizes an experiment we conducted to study the effect of error-rate on the value of N. We considered the set of applications used in Section 2.3. In this experiment we identified the lowest value of N (in multiples of 10) for which the real-optimal design was included in the set. The x-axis indicates different applications (number of tasks 4, 5, 6, and 7) and y-axis indicates value of N. Each of the bar graph shows the minimum value of N. The value was determined by averaging over 3 experiments per application. The four bars per application indicate the value of N for four different error-rates. As expected, the size of N increases as the rate of error increases. In addition, the value of N affects the number of designs that need to be evaluated by high-level estimation. Therefore, the designer should also take estimation and simulation time into account while deciding the value of N.

On the other hand, DESERT does not support specification of N while performing DSE. The number of designs identified depends on the constraints specified while modeling. However, the designer can tighten or loosen the constraints to vary the number of designs selected by DESERT [7].

### 3.4 Design Browser

The MILAN design browser is a graphical front-end to HiPerE. The input to the browser is the set of designs identified in step one. Figure 5 shows a snapshot of the design browser. Among the features supported are display of mapping information of the designs identified by the optimization heuristics, invocation of HiPerE on one or more designs, duty-cycle parameter specification, and visual comparison of the designs based on the estimates of latency and energy dissipation.

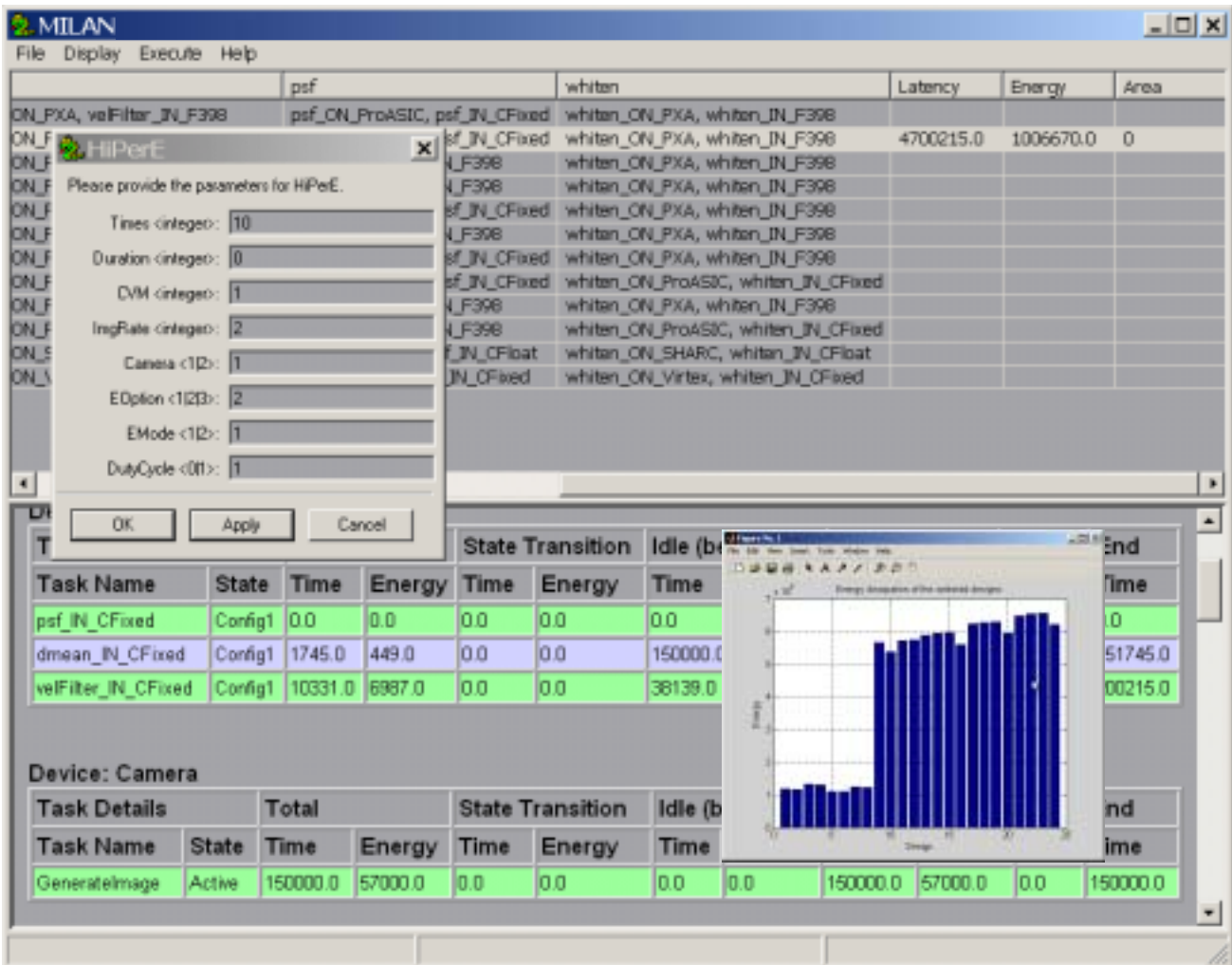


Figure 5: Design browser

Using the design browser, the designer can perform trade-off analysis using the estimation capabilities of HiPerE. Designer can also evaluate the performance impact of allowing the processing components to idle or shutting the components down when not used. HiPerE also produces an activity report for the entire duration of simulation for a duty-cycle based scenario which can be viewed and analyzed through the design browser.

#### 4. ILLUSTRATION OF HIERARCHICAL DESIGN SPACE EXPLORATION

We use two different application design problems to demonstrate the design flow and the advantages of our hierarchical design methodology. The first design problem is to identify an energy-efficient mapping of a beamforming application [15] onto a heterogeneous embedded system while meeting the given latency constraint. The second problem is to identify an energy-efficient hardware for a personnel detection algorithm [18] from a set of devices that consists of traditional processors, FPGAs, and DSPs. Using these examples, we demonstrate the following capabilities of MILAN: (a) support for modeling and rapid design space exploration, (b)

device selection, and (c) energy efficient application mapping onto the target devices.

#### 4.1 Energy-Efficient Mapping for a Beamforming Application

The design problem is to identify an energy efficient mapping of an automated target recognition (ATR) application onto a heterogeneous embedded system while meeting the given latency constraint [15]. The underlying architecture for the PASTA project is already specified. Hence, our approach is used to identify an energy efficient mapping of the ATR application onto the PASTA hardware.

The hardware includes sensor(s), a processor, several microcontrollers (Cygnal 8051), memories, and a radio. Each component can be independently turned on or off. In addition, the processor (Intel PXA 255) supports voltage and frequency scaling [6]. The target application is an automated target recognition algorithm that performs beamforming based on acoustic signals from the sensors [15]. The beamforming application consists of a linear array of 6 tasks (Figure 6). The first three tasks are receive data which is mapped onto the radio, sampling which is mapped onto

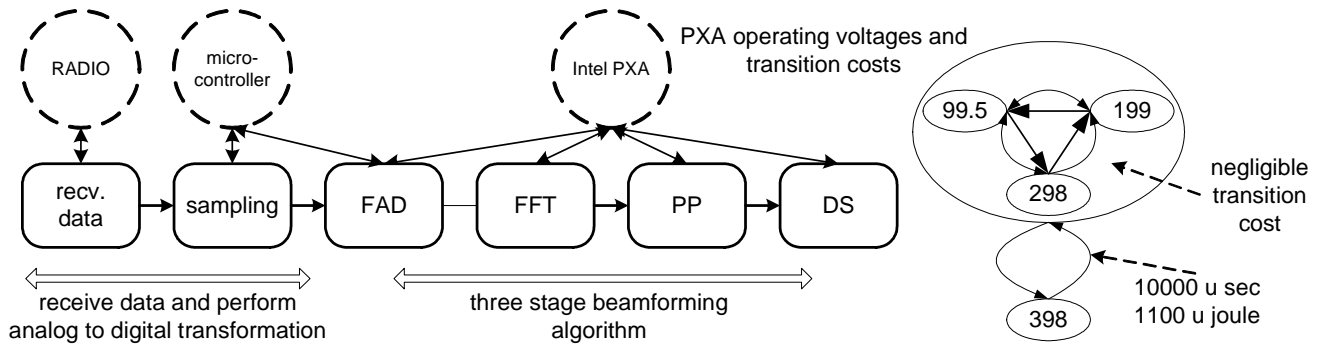


Figure 6: Beamforming application and frequency transition costs for PXA 255

the microcontroller, and false-alarm detection which can be mapped onto either the microcontroller or the processor. The last three tasks that compute beamforming are FFT, peak-pick, and delay sum, all mapped onto the processor.

The design problem for PASTA involves identification of the operating state of each component for each task such that the complete ATR application dissipates the minimum energy while satisfying the latency requirement. All the components in the PASTA stack have at least two operating states; ON and OFF. There is a constant amount of time and energy spent to switch on or off each component. In addition, the processor has 6 different operating frequencies. Tasks can be mapped onto the processor or the microcontroller. When mapped onto a processor, the task can be executed at a certain operating frequency and the performance of the mapping depends on the operating frequency. Transition between any two operating frequencies also involves time and energy costs which depend on the source and destination states. The various operating frequencies supported by the processor are 99.5, 199, 298, and 398 MHz. We modeled all the above in MILAN. The resulting design space was approximately 500,000 designs. However, we noticed that the transition costs between different operating states of the processor are negligible except one transition which involves changing operating frequency of the bus (transition to or from 398 MHz). Ignoring the negligible transition costs reduced the design space to 320 designs. As with the other examples, we used simulators for Intel PXA 255 and the microcontroller to estimate performance of all the mappings. The start up costs and state transition costs are estimated based on the data sheets provided by the vendors [6].

Table 2: Performance Estimates of the Tasks at Different Operating Frequencies

Tasks	Freq.	99.5	199	298	398
FFT	Latency	431958	215870	143937	107962
	Energy	107989	90665	64772	68556
peak-pick	Latency	7933	3964	2643	1982
	Energy	1983	1665	1189	1259
delay-sum	Latency	231956	115919	77292	57974
	Energy	57989	46868	34782	36813
FAD	Latency	43200	21600	14400	10800
	Energy	8212	9261	8302	10556

Figure 6 shows the details of the application design prob-

lem. Table 2 provides the latency ( $\mu$  sec.) and energy dissipation ( $\mu$  Joule) for each task for different frequencies.

Using MILAN, we modeled the hardware and the application. The latency constraint was specified as  $\leq 200$  milliseconds for data processing after receiving data using the radio (radio takes on average 655 milliseconds to receive data). We apply hierarchical design space exploration to identify an energy-efficient design that meets the input latency constraint. As the application can be modeled using linear array of tasks we chose to apply the dynamic programming based N-optimization heuristic as the first step ( $N = 8$ ). Overall size of the design space is approximately 320. Even though the design space is not very large, we use this experiment to show that we only need to evaluate 8 designs (in the second step of our methodology) to identify the most energy-efficient design that meets the given latency constraint (results shown in Table 3 includes cost for receiving data). The performance of the most energy efficient solution (Des. 7) that meets the given latency constraint is highlighted. On the other hand, an optimization heuristic that identifies the most energy optimal solution fails to meet the latency as the energy optimal solution (all task executed at frequency 298 MHz) has a latency of 238 milliseconds.

## 4.2 Energy-Efficient Architecture Selection for a Personnel Detection Application

We consider a personnel detection application for our second example [18]. The personnel detection algorithm is required to processes input in real-time and hence there is a hard latency requirement. In addition, as the system needs to be deployed in a power-constrained environment, energy dissipation is also an important metric. The application consists of 5 tasks as shown in Figure 7. The input comes from a sensor. The application design problem is to select the most energy-efficient hardware and the corresponding mapping of the tasks onto the hardware components. The hardware needs to be selected from a set that consists of Xilinx Virtex-II Pro, Actel ProASIC<sup>PLUS</sup>, Intel PXA 255, PowerPC 405, and TI C6711 DSP [1, 6, 16, 20]. Micron Mobile SDRAM was chosen as the memory. In order to identify the most energy-efficient solution, it is necessary to evaluate the designs based on a duty-cycle which was provided as an input. For our target mapping problem, the various duty-cycle parameters are input rate of the camera, rate of computation of the co-variance matrix (CVM) and inverse, and shut down or leave on the components when idle.

Before applying our hierarchical design methodology, we

Table 3: N-optimal Results Using Our Methodology

	Des. 1	Des. 2	Des. 3	Des. 4	Des. 5	Des. 6	Des. 7	Des. 8
Energy ( $\mu J$ )	222422	224692	226653	226723	227306	227376	229407	231537
Latency ( $\mu s$ )	893632	912971	894314	893653	867657	866996	847678	868339

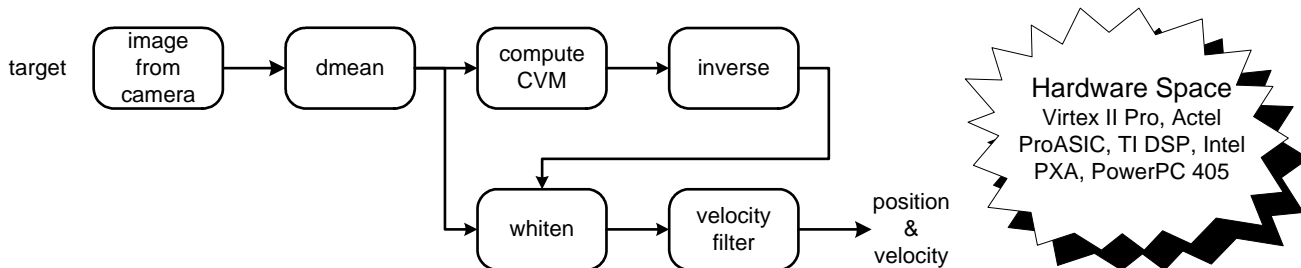


Figure 7: Personnel detection application and the hardware design space

modeled the application design problem using MILAN. Modeling involved specification of the application as a synchronous data flow graph. The modeling of the target devices involved modeling of individual processing components and memory. Once the application and the hardware devices are modeled, possible mapping choices are indicated in the model for mapping. For example, inverse and whiten needs to be computed using floating point arithmetic for which Actel ProASIC *PLUS* is not a suitable choice (due to large area requirement). Such constraints are indicated by not allowing inverse and whiten to be mapped onto ProASIC<sup>PLUS</sup> while modeling. Additionally, while we have specified 5 distinct devices in the design space, the 4 valid combinations are, a stand alone Virtex-II Pro and ProASIC<sup>PLUS</sup> combined with either the DSP or with one of the two processors. Such constraints are specified in the model using the object constraint language supported by MILAN [7].

The duty-cycle was specified as follows: rate of CVM computation is 2, input rate of the camera is 0.5 Hz, and 10 instances of continuous execution, with devices to be shut down when idle. We used DESERT and HiPerE to perform hierarchical design space exploration. As DESERT cannot be used to evaluate designs based on duty-cycle requirements, we used DESERT to prune the design space using performance of each design for a single end-to-end execution. Therefore, we selected a latency constraint of  $\leq 1$  second to identify the designs that can support an input rate of 1 Hz while staying idle for at least 1 second (note that we want to select a set of designs). Through trial and error, energy constraint of  $\leq 860$  millijoules was chosen to have DESERT select 16 designs as output of the first step. The size of the initial design space was approximately 73,000. Once 16 designs were identified by DESERT, we used HiPerE to perform duty-cycle based design space exploration to identify the best design that meets the duty-cycle requirements and dissipates the minimum energy. As Intel PXA 255 and PowerPC 405 do not support floating point arithmetic, due to high latency for floating point emulation, designs that included these two processors were eliminated by DESERT. The 16 designs identified by DESERT included designs that use only Virtex-II Pro or a combination of the DSP and ProASIC<sup>PLUS</sup>. Designs that use the same device (or device combination) differed in mapping. For Virtex-II Pro,

availability of different configurations for each task resulted in different designs. Among the designs we chose 3 designs; the most energy efficient design using only Virtex-II Pro, only the TI DSP, and both TI DSP and ProASIC<sup>PLUS</sup> for analysis. Results of our design space exploration is provided in Table 4.

In Scenario 1, the application is executed only once (no start up or shut down cost included) and in Scenario 2, the application was executed based on the duty-cycle requirement specified to us (Table 4). Note that though the Virtex-II Pro based design is the most energy-efficient for Scenario 1, it is the least energy-efficient for Scenario 2. This fact demonstrates the advantage of using a hierarchical design space exploration and also the usefulness of higher parameter coverage of HiPerE when compared with DESERT. On the other hand, the use of an optimization heuristic in the first step allows us to evaluate a design space of size 73,000 in less than a minute. Using HiPerE, it takes approximately 10 hours to estimate performance of all the designs and a tedious manual comparison of all the estimates to identify the most suitable design.

## 5. RELATED WORK

There are a number of research and commercial initiatives that address the design of energy and latency efficient embedded system design.

The SPADE (System Level Performance Analysis and Design Space Exploration) methodology evaluates embedded system architectures at multiple abstraction levels to perform DSE [9]. The application is modeled based on Kahn Process Network and hardware is modeled using a large library of architecture models at different granularity. Architecture evaluation at multiple granularity level exploits time versus accuracy trade-off. Hence, DSE involves quick evaluation of a large number of designs using coarse-level architecture models followed by the use of more accurate and detailed modeling as the number of designs reduces. In contrast, we use a hybrid approach by integrating optimization heuristics and high-level performance estimation. Because of optimization heuristics, we can rapidly evaluate a much larger design space prior to estimation (and simulation) based design space exploration. Additionally, synchronous data flow model for applications enables analysis of

**Table 4: Results for Personnel Detection Application**

Designs	Scenario 1		Scenario 2	
	Latency (ms)	Energy (mJ)	Latency (ms)	Energy (mJ)
Virtex-II Pro only	247.33	114.06	17895	13938.1
TI DSP only	614.57	670.11	18286	9692.7
Actel ProASIC + TI DSP	496.50	538.18	17166	8652.9

operating state transition costs such as reconfiguration and dynamic voltage scaling between the execution of successive tasks.

Stammermann et al.[19] presents ORINOCO, a software tool for power dissipation analysis and optimization at the algorithmic abstraction level from C/C++ and VHDL description. ORINOCO allows a designer to evaluate choice of algorithms, scheduling, and implementation for register-transfer level design. Co-design based on hierarchical partitioning of algorithms into basic operations is presented in [14]. This methodology gradually decomposes an algorithm into processes and functions (till basic operations such as addition, multiplication are reached) and evaluates all possible partitioning to verify if performance constraint can be met. However, both these efforts do not exploit the available capabilities such as frequency scaling, reconfiguration, shut down or leave idle, etc. that can be manipulated to achieve optimal energy performance. In addition, ORINOCO focuses only on hardware synthesis. On the other hand, our methodology exploits the above capabilities for energy-efficient application design using heterogeneous embedded systems that integrates general purpose processors and reconfigurable devices onto a single device.

Several C/C++ language based approaches such as SystemC, SpecC are primarily aimed at making the C language usable for hardware and software design and allow efficient and early simulation, synthesis, and/or verification [2]. However, these approaches do not facilitate modeling of applications at a level of abstraction that enables algorithmic analysis. Additionally, generating source code and going through the complete synthesis process to evaluate each algorithm decision is time consuming. However, such high-level languages are complimentary to our methodology. MILAN can be augmented to support system synthesis using these languages to implement design identified through design space exploration [10].

Among the commercial tools, Xilinx System Generator for Simulink provides a high-level interface for application design using pre-compiled libraries of signal processing kernels [20]. Similarly, Cadence Signal Processing Workbench (part of the Incisive Verification Platform) [3] supports development of signal processing algorithms using communications and multi-media libraries. Other tools such as Xilinx EDK and Mentor Graphics FPGA Advantage provide integrated design environments that accept high-level specification as VHDL or Verilog scripts, schematics, finite state machines, etc. and provide simulation, testing, and debugging support for the designer to implement a design using FPGAs [17, 20]. However, these tools start with a single conceptual design. Design space exploration is performed as part of implementation or through local optimizations to address performance bottlenecks identified during synthesis. Additionally, these tools [17, 20] are designed for a specific target device and do not allow integration of additional sim-

ulators or design tools. In contrast, our methodology and the environment based on MILAN allows evaluation of a wide variety of hardware devices and performs device selection followed by design of application through evaluation of a large design space.

In summary, we use a hybrid model-based approach by integrating optimization heuristics and high-level performance estimation. Model-based approach enables rapid performance estimation and performance optimization using algorithms and heuristics. Additionally, because of optimization heuristics, we can rapidly evaluate a much larger design space prior to estimation (and simulation) based design space exploration. As we make no assumption about the underlying hardware, using our approach, it is also possible to evaluate a set of candidate hardware choices for a given application.

## 6. CONCLUSION

We proposed a hierarchical design space exploration methodology that integrates optimization heuristics, high-level estimation tools, and simulators to perform efficient design space exploration during application design using heterogeneous embedded systems. We also discussed the implementation using the MILAN framework which provides an user friendly environment for designers to apply our methodology. The key advantages of MILAN are a unified modeling environment and the ability to seamlessly integrate tools and simulators. Additionally, as MILAN can potentially integrate any available tool for simulation, compilation, and design space exploration, it is possible to modify MILAN based on the target devices and create a customized design environment. Therefore, MILAN has an advantage when there is a need of a specialized high-performance and low-energy implementation using a variety of COTS components. In addition, as alternatives to DESERT, genetic algorithm or simulated annealing based design space exploration are also suitable for our hierarchical design space exploration. We are considering these techniques to integrate them into MILAN as alternate optimization heuristics in our methodology.

**Acknowledgment:** We acknowledge Akos Ledeczki, James R. Davis, and Sandeep Neema of Institute for Software Integrated Systems, Vanderbilt University for helpful discussions regarding GME and DESERT in implementing the MILAN project. We thank Julius F. Bogdanowicz and Edward W. Wanek of Raytheon and Michael Bajura of USC Information Sciences Institute, East for their inputs regarding the illustrative examples. We also thank Jingzhao Ou for his input regarding the design browser.

## 7. REFERENCES

- [1] Actel ProASIC<sup>PLUS</sup> - The Nonvolatile Reprogrammable Gate Array.

- <http://www.actel.com/products/proasic/>
- [2] L. Cai, M. Olivarez, P. Kritzinger, and D. Gajski, "C/C++ Based System Design Flow Using SpecC, VCC, and SystemC," Tech. Report 02-30, UC, Irvine, June 2002.
- [3] Cadence Incisive Verification Platform and Signal Processing Workbench.  
<http://www.cadence.com/products/incisive.html>
- [4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms," MIT Press and McGraw Hill, 2001.
- [5] Generic Modeling Environment.  
<http://www.isis.vanderbilt.edu/Projects/gme/>
- [6] Intel PXA 255 Processors.  
<http://www.intel.com/design/intelxscale/>
- [7] A. Ledeczi, J. Davis, S. Neema, and A. Agrawal, "Modeling Methodology for Integrated Simulation of Embedded Systems," ACM Transactions on Modeling and Computer Simulation (accepted).
- [8] E. A. Lee and D. G. Messerschmitt, "Synchronous Data Flow," Proceedings of IEEE, Vol. 75, September 1987.
- [9] P. Lieverse, P. van der Wolf, K. Vissers, and E. Deprettere, "A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems Design Methodology," Journal of VLSI Signal Processing for Signal, Image and Video Technology, vol. 29, no. 3, pp. 197-207, November 2001.
- [10] Model-based Integrated Simulation.  
<http://milan.usc.edu/>
- [11] S. Mohanty and V. Prasanna, "Rapid System-Level Performance Evaluation and Optimization for Application Mapping onto SoC Architectures," IEEE Intl. ASIC/SOC Conference, 2002.
- [12] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis, "Rapid Design Space Exploration of Heterogeneous Embedded Systems using Symbolic Search and Multi-Granular Simulation," Language Compilers and Tools for Embedded System, 2002.
- [13] T. Mudge, "Power: A First Class Design Constraint," Computer, April 2001.
- [14] H. Oudghiri, B. Kaminska, J. Rajski, "A Hardware/Software Partitioning Technique with Hierarchical Design Space Exploration," Custom Integrated Circuits Conf., 1997.
- [15] Power Aware Sensing Tracking and Analysis.  
<http://pasta.east.isi.edu/>
- [16] PowerPC 405 Embedded Cores.  
[http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC\\_405\\_Embedded\\_Cores](http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405_Embedded_Cores)
- [17] Seamless Hardware/Software Co-Verification and FPGA Advantage, Mentor Graphics.  
<http://www.mentor.com/>
- [18] P. Singer, "The Optimal Detector," SPIE Conference: Signal and Data Processing for Small Targets, 2002.
- [19] A. Stammermann, L. Kruse, W. Nebel, A. Pratsch, E. Schmidt, M. Schulte, and A. Schulz, "System Level Optimization and Design Space Exploration for Low Power," Intl. Symposium on System Synthesis, pp. 142-146, 2001.
- [20] Xilinx Virtex-II Pro and Xilinx System Generator for Simulink (Matlab). <http://www.xilinx.com/>