

# A Framework for Energy Efficient Design of Multi-Rate Applications using Hybrid Reconfigurable Systems

Sumit Mohanty and Viktor K. Prasanna

University of Southern California, CA, USA  
{smohanty, prasanna}@usc.edu

**Abstract.** Hybrid reconfigurable systems integrate DSPs and general purpose processors with an FPGA fabric. These systems may support features such as efficient start-up and shut-down, dynamic voltage scaling, and reconfiguration, that are exploited for energy-efficient application design. Duty cycle is the proportion of time during which a system is operated. Multi-rate applications consist of tasks that execute at different rates. Designing an energy-efficient hybrid reconfigurable system with duty cycle specification that implements a multi-rate application using devices with multiple operating states presents several challenges such as modeling, rapid performance estimation, and efficient design space exploration. We present a design framework that addresses these challenges. Using our framework, we illustrate the design of two energy efficient systems: automated target detection and adaptive beamforming.

## 1 Introduction

Due to dramatic increase in the number of portable and embedded applications, systems implementing signal processing applications which were formerly limited by the available computation power are currently limited by the available energy [11]. Examples of such systems include mobile base stations for software defined radio [4] and target detection and tracking systems [13]. A system consists of an application and a target hardware that implements the application. Our target applications, implemented by the systems discussed above, are multi-rate signal processing applications that can be modeled as data flow graphs [7]. The input is assumed to be a data stream consisting of several frames of data with a specified frame rate. The target hardware is a hybrid reconfigurable system that integrates instruction set architecture (ISA) based processors [5, 12] with a reconfigurable fabric. The tradeoff considered here is high throughput and relatively higher average power dissipation using the FPGAs vs. comparatively lower throughput and power efficient solution using the general purpose processors. A hybrid reconfigurable system can be a single chip device such as the Xilinx Virtex-II Pro [15] or a multi-chip device such as, for example, an Actel ProASIC [1] and a TI DSP [14].

Given a target application, design of a hybrid reconfigurable system involves (device) selection of suitable processing components (FPGAs and ISA-based

processors) and memory. Following device selection, the 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. The designer also needs to identify appropriate hardware/software partitioning 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 based on the duty cycle specification [10]. Duty cycle is the proportion of time during which a system is operated. Such specification allows modeling of a period of execution as alternate active and inactive phases. Energy dissipation (e.g. due to leakage current), especially for systems with low duty cycle, during the inactive phases can contribute significantly to the overall energy dissipation of the system. Therefore, the tradeoff between the performance cost of shutting down and starting up a device and the performance cost of remaining idle needs to be considered during system design [2]. In addition, an application is defined as multi-rate if the constituent tasks execute at different rates. For example, an adaptive beamformer processing up to 105 mega samples per second may need to update the weight coefficients only once every second [3]. Therefore, tasks performing weight coefficient evaluation and update execute once every second. On the other hand, the tasks that process each data sample are executed  $105 \times 10^6$  times every second (see Section 5.2).

In this paper, we present a framework that supports energy efficient design of multi-rate applications based on duty cycle specifications using a hybrid reconfigurable system. Our primary focus is system design problems with performance requirement of minimizing energy while sustaining a given input rate. Furthermore, the task rates affect the quality of the output [3]. Our framework can also be used to evaluate the target hardware(s) in terms of the task execution rates supported for multi-rate applications and range of input rates sustained based on latency requirements. The framework is based on a hierarchical design technique [10] which involves modeling and a two step design space exploration (DSE).

The paper is organized as follows. Section 2 discusses related work. The modeling approach is discussed in Section 3. Section 4 discusses the design framework. Two examples demonstrating the framework are presented in Section 5. We conclude in Section 6.

## 2 Related Work

There are a number of research and commercial initiatives that address the design of efficient systems.

The SPADE (System Level Performance Analysis and Design Space Exploration) methodology evaluates embedded system architectures at multiple abstraction levels to perform DSE [8]. DSE involves quick evaluation of a large number of designs using coarse-level architecture models followed by the use of more accurate and detailed models 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 (or simulation) based design space exploration. Additionally, our augmented data flow (DF) model for applications enables performance analysis of multi-rate applications based on operating state transition costs when the target devices support multiple operating states.

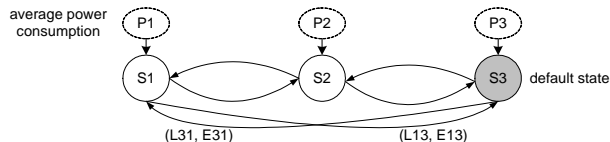
Among the commercial tools, Xilinx System Generator for Simulink provides a high-level interface for application design using pre-compiled libraries of signal processing kernels [15]. However, this tools starts with a single conceptual design. Design space exploration is performed as part of implementation or through local optimizations to address performance bottlenecks identified during synthesis. Additionally, these tools [15] are specific to a set of target devices and cannot be used for device selection problem.

Benini et al. discuss various techniques for dynamic power management such as turning off system components or moving system components to a low performance operating state [2]. However, to the best of our knowledge, there are no tools available that take into consideration both multi-rate applications and duty cycle specifications during system design to reduce energy dissipation.

### 3 System Modeling and Design Problems

A set of models are used in the design framework to capture the system specification (application, target hardware, mapping, performance, etc.).

#### 3.1 Modeling Multiple Operating States



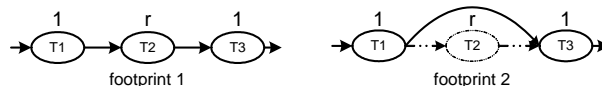
**Fig. 1.** Modeling multiple operating states

We model a candidate devices based on the operating states supported by the device. An operating state can be a configuration for an FPGA or a voltage/frequency setting for devices supporting voltage/frequency scaling. In addition, given two operating states  $A$  and  $B$ , we assume that the transitions from  $A$  to  $B$  and  $B$  to  $A$  are associated with transition costs (latency and energy).

Our model is based on an augmented finite state machine (FSM). Figure 1 shows a sample model for a device with 3 operating states. Each node in an FSM represents one operating state. Each pair of nodes is connected with a pair of directed edges. Each edge correspond to a state transition from the state represented as the source node to the state represented as the destination node. Each edge is also associated with a 2-tuple  $(TL, TE)$  where  $TL$  is the latency cost and  $TE$  is the energy dissipation during the transition. Each operating state is associated with an estimate of average power consumed while idling ( $P1, P2, P3$  in Figure 1). The model also indicates a default state (shown in gray).

### 3.2 Multi-rate Application Modeling and Mapping Specification

To model the application, initially, we create a (data flow) DF model of the application assuming that all the tasks will execute at the same rate. The rate is assumed to be 1 which refers one execution of a task per input frame. Rate of 1 is the maximum rate possible for any task. Afterwards, we associate each task with their respective rates. A rate of  $r$  refers to one execution per  $r$  input frames. Therefore, given a multi-rate application, for some input frames certain tasks will not be executed. In such cases, we assume that the tasks preceding and succeeding the task not being executed are connected (Figure 2) and thus the dependencies between the tasks are maintained.



**Fig. 2.** Multi-rate application modeling

Based on the serial order (*1st, 2nd, ..., nth*) of the frame being processed, the data flow graph will change as it needs to model only the active tasks. We define each distinct DF model for the application as a *footprint*. Figure 2 shows two footprints of the same application with three tasks. Each footprint is analyzed by our framework to identify the most energy efficient design per footprint while performing a duty cycle based design space exploration. A *master footprint* is defined as the footprint of the application when all tasks are active.

In addition, each application task is associated with a set of alternatives. Alternatives refer to the mappings of application tasks on different target devices operating in different operating states. For example, there will be two alternatives if a task can be mapped on a general purpose processor and a DSP. If each device has two operating voltages, then there will be four alternatives instead.

### 3.3 Modeling Duty Cycle

Duty cycle is the proportion of time a system is active. Therefore, based on the duty cycle specification, system execution can be modeled as alternate active and inactive phases. Duration of each phase depends on the input rate and the latency required to process a single input frame. Input rate can be variable. Our design framework needs the input rates to be statically defined. Variable input rate is specified as an ordered set of 2-tuples  $(Ir, Nf)$  where  $Ir$  refers to the input rate in Hz and  $Nf$  refers to number of frames processed at the above rate. Given two consecutive 2-tuples,  $(Ir_i, Nf_i)$  and  $(Ir_{i+1}, Nf_{i+1})$ , application execution is modeled as “process  $Nf_i$  frames at the rate of  $Ir_i$  Hz followed by  $Nf_{i+1}$  frames at the rate of  $Ir_{i+1}$  Hz”. Our framework assumes that the given set of 2-tuples can repeat indefinitely to model the processing of large number of input frames. Similarly, maximum allowed latency to process an input frame can also be specified. If not specified, the framework derives the constraint based on the given maximum input rate.

### 3.4 Constraints

The constraints are divided into two types; *performance constraints* and *design constraints*. Performance constraints are of the form “metric < constant” where

the metric can be latency or energy. The design constraints specify valid combinations of task mappings through pairwise relations between two mappings (e.g. task  $A1$  implemented on device  $B1$  operating in state  $C1$  is not compatible with task  $A2$  implemented on device  $B2$  operating in state  $C2$ , or task  $X$  can be mapped only to device  $R$ ). Similarly, design constraints also specify valid combinations of devices. As our focus is designing a hybrid reconfigurable system, given one FPGA, two DSPs, and two general purpose processors, one can specify a constraint that a valid target hardware is a combination of one FPGA, and one DSP or one general purpose processor only.

### 3.5 System Design Problems

The models discussed above define a design space. Due to large number of design choices such as target devices, operating states, mappings, device shut down or leave on, etc., the size of the design space can be very large (see Section 5.1). Traversal of such a large design space is not practical using a simulators or even high-level estimators (see Section 5). Additionally, to estimate the performance of a candidate design, during estimation, it is required to support duty cycle, multi-rate applications, and multiple operating states for the devices. Our framework addresses the above issues using a hierarchical approach towards DSE and an enhanced high-level performance estimator. The framework supports the two classes of system design problems (a) **device selection**: design a system by selecting a combination of devices from a given set of candidates while satisfying the performance and design constraints and (b) **hardware evaluation**: evaluate a target hardware in terms of the input rates or the range of task execution rates supported.

## 4 System Design Framework

The models discussed above form the basis for the system design framework. The framework supports user friendly visual modeling to specify multi-rate applications, target devices, duty cycle specifications, mappings, and constraints. The system design framework is implemented using MILAN, a Model-based Integrated SimuLatioN environment [9].

### 4.1 Design of the Framework

MILAN is a model based integrated simulation environment for embedded system design and optimization through the integration of various simulators and tools into a unified environment [9]. Using the MILAN environment, the designer formally models the target application, target devices, and constraints through a graphical interface. The models are used to drive various tools and simulators integrated in MILAN. Additional details about the MILAN environment can be found in [9]. We extended the MILAN modeling paradigm to support the models discussed in Section 3. Specific extensions were support for multi-rate application modeling, devices with multiple operating states based on the augmented finite state machine model, and duty cycle specification. MILAN integrates a heuristic based design space exploration tool, DESERT [6] and HiPerE, a high-level performance estimator [10].

DESERT supports DSE when the design space is defined by an application modeled as a data flow graph with alternatives [10]. Alternatives are associated

with a task and refer to different mappings of the task onto devices. Given a set of performance and design constraints, DSE using DESERT refers to identification of a set of designs that satisfy the given constraints. DESERT does not support devices with multiple operating states while performing DSE. We developed a technique that uses pseudo tasks and additional design constraints to enable DESERT perform DSE using devices with multiple operating states. This technique introduces a pseudo task between each pair of connected (via an edge in the model) tasks to model operating state transitions. The alternatives for this pseudo task are all possible operating state transitions among the devices. We specify a set of design constraints to ensure that appropriate state transition is chosen based on the mappings chosen for the pair of tasks. Thus, when DESERT performs DSE and selects the designs that meet the design constraints, correct operating state transitions are also automatically chosen. In addition, operating state transition costs also get added to the overall performance of each design while evaluating the designs against the performance constraints.

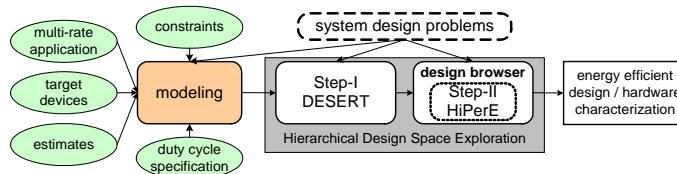
We enhanced HiPerE to support duty cycle specification, multi-rate applications, and devices with multiple operating states. Given a system design and the number of input frames to process, HiPerE estimates the overall latency and energy dissipation of the system. HiPerE uses the models discussed in Section 3 to extract the required performance estimates for task mapping, operating state transitions, etc. By default, HiPerE assumes that the devices are idle during inactive phases. However, HiPerE can be configured to shut down the devices during the inactive phase in which case HiPerE takes into account the start up cost as well for the next active phase.

#### 4.2 Design Flow

The design flow begins with modeling of the application and the target devices. Target devices are modeled using the augmented finite state machine (FSM) model. The performance estimates for various components of the augmented FSM are derived from vendor provided data sheets. Application modeling involves application specification as a data-flow graph with alternatives. Rate of execution is also specified for each application task. The functional specification of the target system specifies the structure of the data-flow graph and the choice of implementations specifies the alternatives. Each alternative is mapped onto a target device operating in a specific state. Each mapping is associated with performance estimates in terms of latency and energy dissipation. Constraint specification follows modeling.

The design framework uses a hierarchical approach for design space exploration (DSE). The basic idea of hierarchical approach is a two step DSE [10]. 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. In our framework, Step-I is performed using DESERT and Step-II is performed using HiPerE (Figure 3). The key to fast DSE using the hierarchical approach is a rapid DSE in Step-I using an approximated model followed by exhaustive search based on a more detailed model [10].

In Step-I, we do not consider the duty-cycle specification and the multi-rate specification of the application while performing DSE. We only consider the multiple operating states of the target devices, the design constraints, and the latency constraint based on the minimum input rate to be sustained. The designs rejected by Step-I are the ones which do not satisfy the given design constraints and the latency constraint based on minimum input rate to be sustained. Therefore, our approach guarantees that none of the rejected designs would have satisfied the given performance requirement even when duty cycle and multi-rate are considered. DSE in Step-I using DESERT is fast. Our experience with DESERT shows that we can prune a design space with  $10^4 \sim 10^5$  designs in order of minutes on a typical uniprocessor system [10]. If the number of designs is too large, we can use energy constraint to further evaluate the designs. The energy constraint can be relaxed (tightened) if too few (many) designs are selected.



**Fig. 3.** System design approach

In Step-II, the selected designs are evaluated using HiPerE based on the duty cycle specification and the multi-rate aspect of the application to identify design(s) based on the specific system design problem being solved. Using HiPerE, a designer specifies the duty cycle and the rates of different tasks and evaluates all the designs. The most energy efficient design is selected manually. In addition, HiPerE provides automated support to evaluate designs for range of input rates and ranges of rates supported for each task.

## 5 Illustrative Examples

We use two problems to demonstrate the flow of our design framework. The first 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 [13]. The second problem also solves the device selection problem for an LMS (Least Mean Square)-based MVDR (Minimum Variance Distortionless Response) adaptive beamforming algorithm [3] and evaluates the selected target hardware(s) in terms of the rate of weight coefficient update. LMS-based MVDR adaptive beamforming algorithm is used in the base station for software defined radios to design smart antennas [3].

### 5.1 Personnel Detection Application

The application consists of 5 tasks (Figure 4) [13]. Tasks shown in gray have rates higher than 1. 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. Application modeling involved specification of the application as an augmented data flow graph (Figure 4). Individual target devices were modeled using the augmented FSM model. Possible mapping choices for each task were also indicated. For example, inverse and whiten needs to be computed using floating point arithmetic for which Actel ProASIC<sup>PLUS</sup> is not a suitable choice (due to a smaller number of gates). Such requirements are indicated by not allowing inverse and whiten to be mapped onto ProASIC<sup>PLUS</sup> while modeling. Additionally, the 4 valid device combinations are, 1) Virtex-II Pro only, 2) ProASIC + DSP, 3) ProASIC + PXA 255, and 4) ProASIC + PPC 405. The rate of CVM computation and inverse was specified as 2 and input rate was specified as 0.5 Hz. The latency constraint was specified as  $\leq 2$  seconds. We also consider only DSP as the fifth choice for performance comparison between DSP and hybrid reconfigurable system. Modeling effort required for this experiment was approximately 4 hours. Modeling effort does not include simulation or estimation necessary to estimate performance of mapping and state transitions. However, as our framework supports model reuse, simulation performed off-line can be reused during modeling.



**Fig. 4.** Application model for Personnel Detection

Following modeling, we performed design space exploration using DESERT and HiPerE. Through several iterations using DESERT, energy constraint of  $\leq 860$  mJ was chosen to have DESERT select 16 designs as output of Step-I. The size of the initial design space was approximately 73,000. Once 16 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. From the 16 designs, we chose 3 designs with different target hardware for analysis (Table 1).

In Scenario 1, the system processes only one frame (no start up or shut down cost included) and in Scenario 2, the system processed 10 input frames (Table 1). 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 is due to high energy dissipation during inactive phases. In both the scenarios, the hybrid reconfigurable system out-performs the DSP based system. Based on the above analysis, we selected TI DSP and ProASIC<sup>PLUS</sup> as the target hybrid reconfigurable system.

DSE using our framework was performed using a PC with a 848 MHz Pentium III. The use of an optimization heuristic in Step-I allows us to evaluate a design space of size 73,000 in less than a minute. Step-II, evaluating 16 designs, also took about 2 minutes. On the other hand, if we use HiPerE (which runs significantly faster than low-level simulators), it takes approximately 10 hours to estimate the performance of all the designs and a tedious manual comparison of all the estimates to identify the most energy efficient design.

**Table 1.** Results for device selection

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

## 5.2 Adaptive Beamforming for Software Defined Radio

The LMS-based MVDR algorithm consists of three steps. The first step is the calculation of filter output. The second step is the calculation of input signal power, correlations between the signals, normalization, and the calculation of error signals. The third step is the update of the weight coefficients of the adaptive filter. The incoming data rate is approximately  $105 \times 10^6$  samples per second [3]. This provides us with the latency constraint of  $\leq 9.5$  ns.

Typically, the rate of weight coefficient update is once every second [3]. However, the rate of update depends on the stability of the environment and the application requirement. For example, an environment with high interference and path-loss might require a higher rate of update. Therefore, in addition to device selection, we also use the framework to identify the maximum rate of update that can be supported without affecting latency. The set of devices considered includes two Xilinx Virtex-II Pro devices (xc2vp2 and xc2vp20), one Xilinx Virtex-II (xc2v1500), Intel PXA 255, PowerPC 405, and TI C6711 DSP. Most of these devices are also used by the system design problem discussed in Section 5.1. Thus, we were able to reuse the models of the devices defined in MILAN. The size of the design space was 243.

HiPerE was configured to power down components when idle if it reduces the overall energy dissipation. Table 2 shows the designs selected by our framework. Scenario 1 refers to coefficient update rate of once every  $105 \times 10^6$  samples. In Scenario 2, the coefficient update rate is the maximum that can be sustained by a design. We evaluated all the designs based on their performances when the system executes for a period of 1 second. For each design, Table 2 shows the energy dissipation per scenario. The last column shows the maximum update rate that can be supported.

An additional advantage of using our framework is the savings in design time due to reduction in the simulation time. Cycle-accurate simulation of the processors and RT-level simulation of the FPGAs are time consuming. For example, simulation of the task filter using the DSP simulator takes approximately 15 minutes and using ModelSim and XPower takes about 30 minutes. Therefore, simulating all the 243 designs would take 100+ hours. Using our approach, we could identify the results in minutes. There are some overheads associated with computing the performance estimations of different mappings while modeling. However, in the worst case, the overhead will be 5 simulations (one simulation of the three stage application per device) as opposed to 243 simulations.

## 6 Conclusion

Our framework implements a hierarchical approach for hybrid reconfigurable system design based on duty cycle specification and multi-rate applications using

**Table 2.** Candidate designs and max update rates supported

No.	designs	total energy dissipated (mJ)		max rate supported
		Scenario 1	Scenario 2	
1	two xc2vp2	921.6	1113.1	8
2	xc2vp2 + PowerPC	816.6	755.6	$12 \times 10^3$
3	xc2vp2 + PXA 255	682.6	839.6	$11 \times 10^3$
4	xc2vp2 + TI DSP	490.6	1960	$1.5 \times 10^3$
5	xc2v1500	792.3	852.2	8
6	xc2vp20	968.6	1019.6	8

devices with multiple operating states. DSE using our framework is faster than DSE using only low-level simulators or high-level estimators. In addition, we provide a user friendly interface for modeling and DSE. We depend on the vendor provided data sheets and third-party simulators to estimate various model parameters (e.g. task mapping costs, operating state transition costs, etc.). The quality of estimates affects the quality of the output of our framework. In addition, our framework does not perform synthesis for which we rely on the design tools provided by respective vendors [1, 15].

## References

1. Actel ProASIC<sup>PLUS</sup> <http://www.actel.com/>
2. L. Benini, A. Bogliolo, and G. De Micheli "A Survey of Design Techniques for System-level Dynamic Power Management," IEEE Tran. on VLSI Systems, Volume: 8 Issue: 3, June 2000.
3. M. Devlin, "Product Focus DSP: How to Make Smart Antenna Arrays," Xcell Journal Q1 2003.
4. C. Dick, "FPGA: Enabling the Software/Reconfigurable Radio," Advanced Radio Technologies, 2003.
5. Intel PXA 255 Processors. <http://www.intel.com/design/intelxscale/>
6. 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.
7. E. A. Lee and D. G. Messerschmitt, "Synchronous Data Flow," Proc. of IEEE, Vol. 75, Sep. 1987.
8. P. Lieverse, P. Van Der Wolf, K. Vissers, and E. Deprettere "A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems," JVLISI Signal Processing for Signal, Image and Video Technology, Nov. 2001.
9. Model-based Integrated Simulation. <http://milan.usc.edu/>
10. S. Mohanty and V. K. Prasanna, "A Hierarchical Approach for Energy Efficient Application Design Using Heterogeneous Embedded Systems," Compilers, Arch., and Synthesis for Embedded Sys., 2003.
11. J. Ou, S. Choi, and V. K. Prasanna, "Energy Efficient Hardware/Software Co-Synthesis on Platform FPGAs," Journal on Embedded Systems, June 2004.
12. PowerPC 405 Embedded Cores. <http://www.ibm.com/>
13. P. Singer, "The Optimal Detector," SPIE Conf.: Signal and Data Processing for Small Targets, 2002.
14. TI TMS320 Series DSPs. <http://dspvillage.ti.com/>
15. Xilinx Virtex-II Pro and Xilinx System Generator for Simulink (Matlab). <http://www.xilinx.com/>