

# A Modeling and Exploration Framework for Mapping of Linear Array of Tasks onto Adaptive Computing Systems

Egor Andreev, Sumit Mohanty, Viktor K. Prasanna  
University of Southern California, Los Angeles, CA 90089  
{andreev, smohanty, prasanna}@usc.edu

Project URL: <http://milan.usc.edu>

## Motivation and Background

### Adaptive Computing Systems (ACS)

ACS can dynamically change their operating state to meet performance requirements of the mapped task.

Example ACS components:

- reconfigurable devices such as FPGAs (Xilinx Virtex-II, Actel ProASIC)
- processors supporting dynamic voltage and frequency scaling (Intel PXA 255, PowerPC 405)
- memories with multiple power states and banks that can be individually turned on/off (Micron Mobile SDRAM)

### Linear Array of Tasks

Linear array of tasks models an application as an ordered set of tasks where each task has at most one input and one output. Such a model includes only one source task and one sink task.

Several applications such as automatic target recognition, automated object tracking, MPEG decoder/encoder, software defined radio, etc. can be modeled as a linear data flow graphs.

### Design Issues

A mapping of a linear array of tasks onto an ACS refers to a set of operating states such that each task is associated with an operating state.

Costs associated with such mappings:

- Each operating state is associated with certain amount of latency and energy cost for each task that can be executed in the state
- State transition cost also includes latency and energy dissipation

Example optimization problems:

- Minimize a performance metric (latency or energy)
- Minimize a metric while meeting a pre-specified requirement for another metric

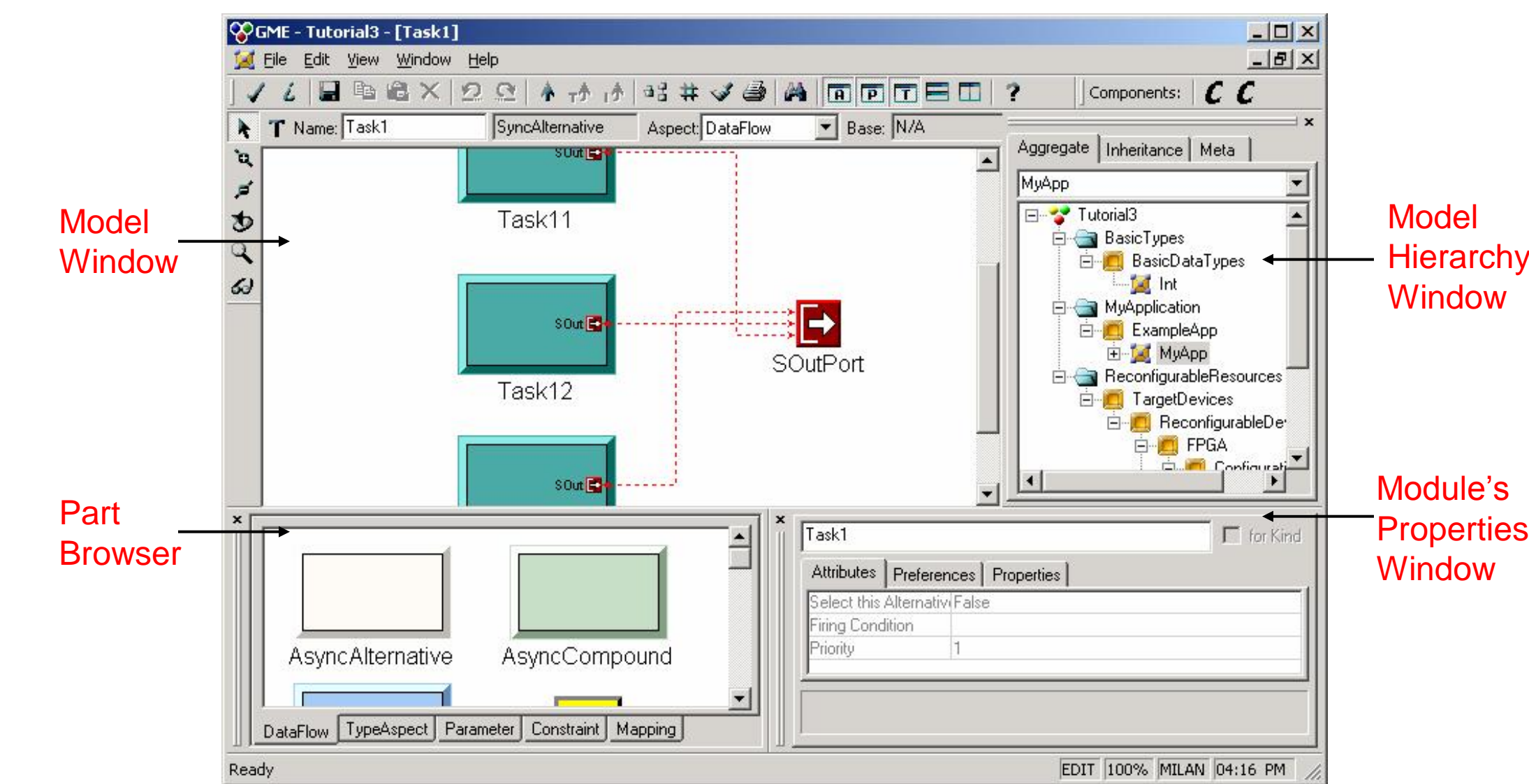
## Modeling and Exploration Framework

### MILAN Framework

MILAN is a model-based, extensible simulation framework that facilitates rapid evaluation of different performance metrics, such as power, latency, and throughput, at multiple levels of granularity of a large set of embedded systems by seamlessly integrating different widely-used simulators into a unified environment

### GME 2000\* for Modeling

GME 2000 is a Graphic Modeling Environment that – when configured using MILAN – can be used to model this class of applications. It allows the user to visually model his or her application.



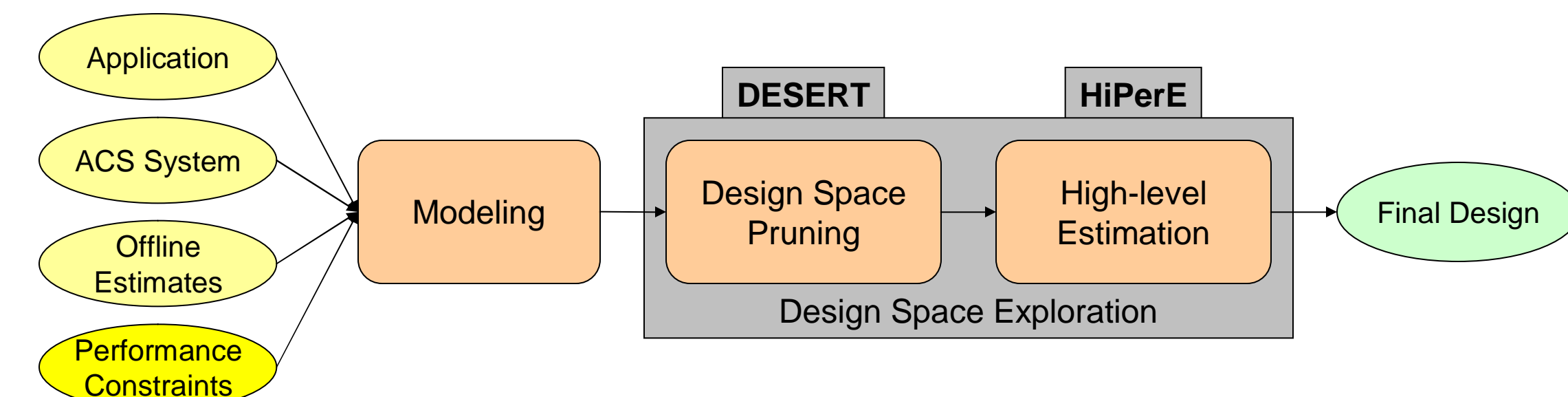
- User friendly interface for modeling (dragging & dropping visual objects)
- Interpreters are used to convert information from a model into input for external tools
- Interpreters are also used to feed the output from tools back into the models

In our case, GME configured for MILAN enables modeling of a linear array of tasks in a format suitable for the DESERT tool and HiPerE.

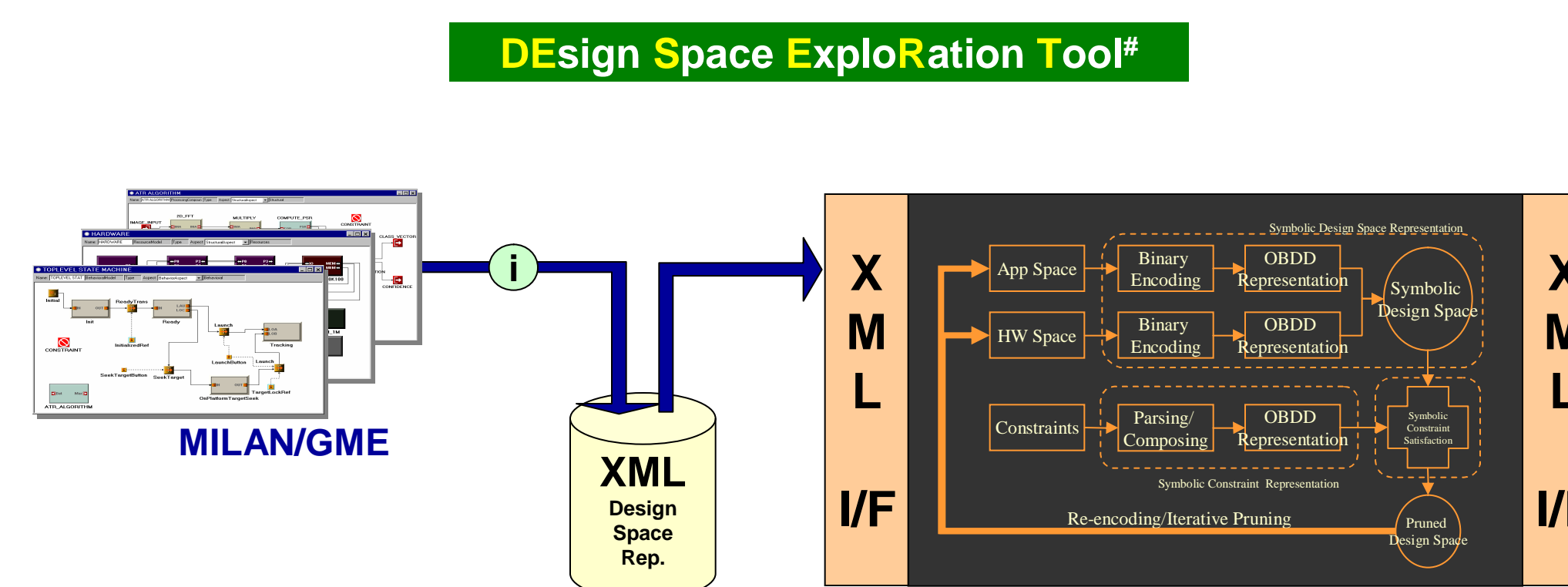
\* GME 2000 has been developed at ISIS, Vanderbilt

## Our Approach

### Design Flow



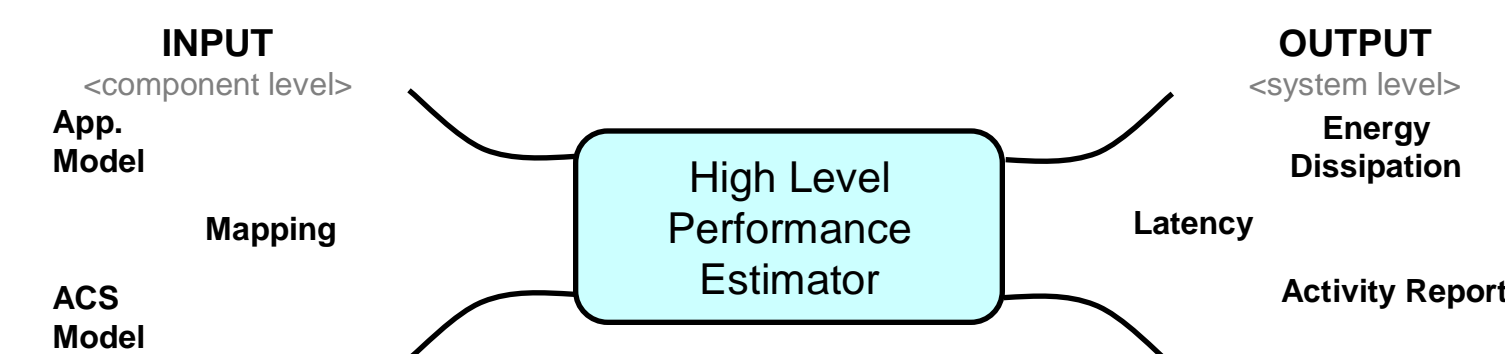
### Constraint Satisfaction using DESERT



- Generic, domain-independent, constraint-based design space exploration tool
- Uses Ordered Binary Decision Diagrams (OBDD) to represent the application and the constraints in the form of Boolean variables
- DESERT outputs list of designs that satisfy the constraints

# developed at ISIS, Vanderbilt

### Rapid Estimation using HiPerE

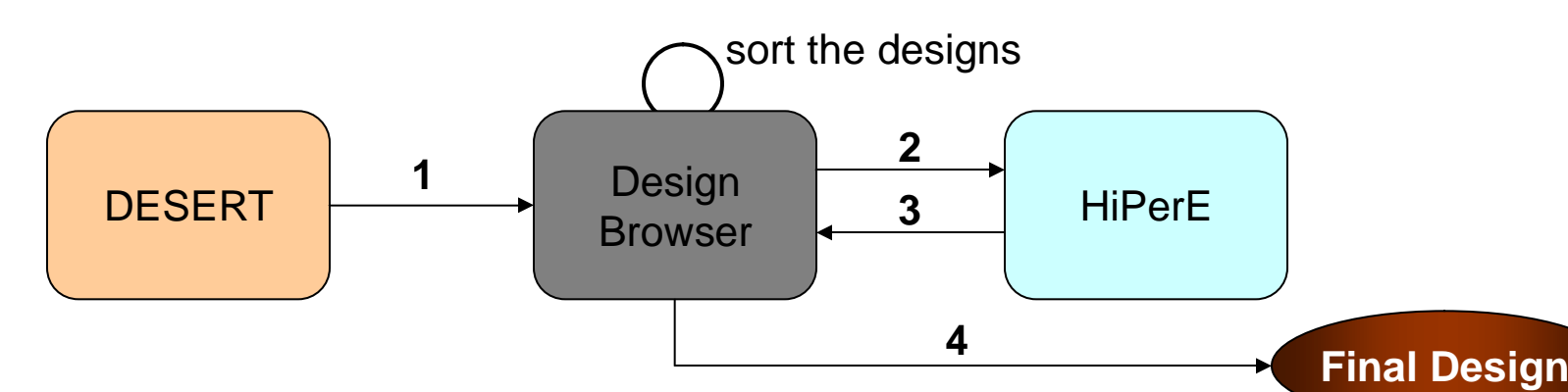


- Given an application model, a resource model, and a mapping, provides rapid system-level performance estimates (latency and energy)
- Supports applications modeled as data flow graphs
- Interpretive simulation allows rapid hardware/software co-estimation
- Activity report provides execution details of a given mapping

### Identify Final Design Through Comparison

Final design is selected by evaluating each design identified by DESERT using the Design Browser (DB) for HiPerE

- DESERT output fed to DB
- HiPerE to evaluate each design
- HiPerE results fed back to DB
- DB identifies the final design



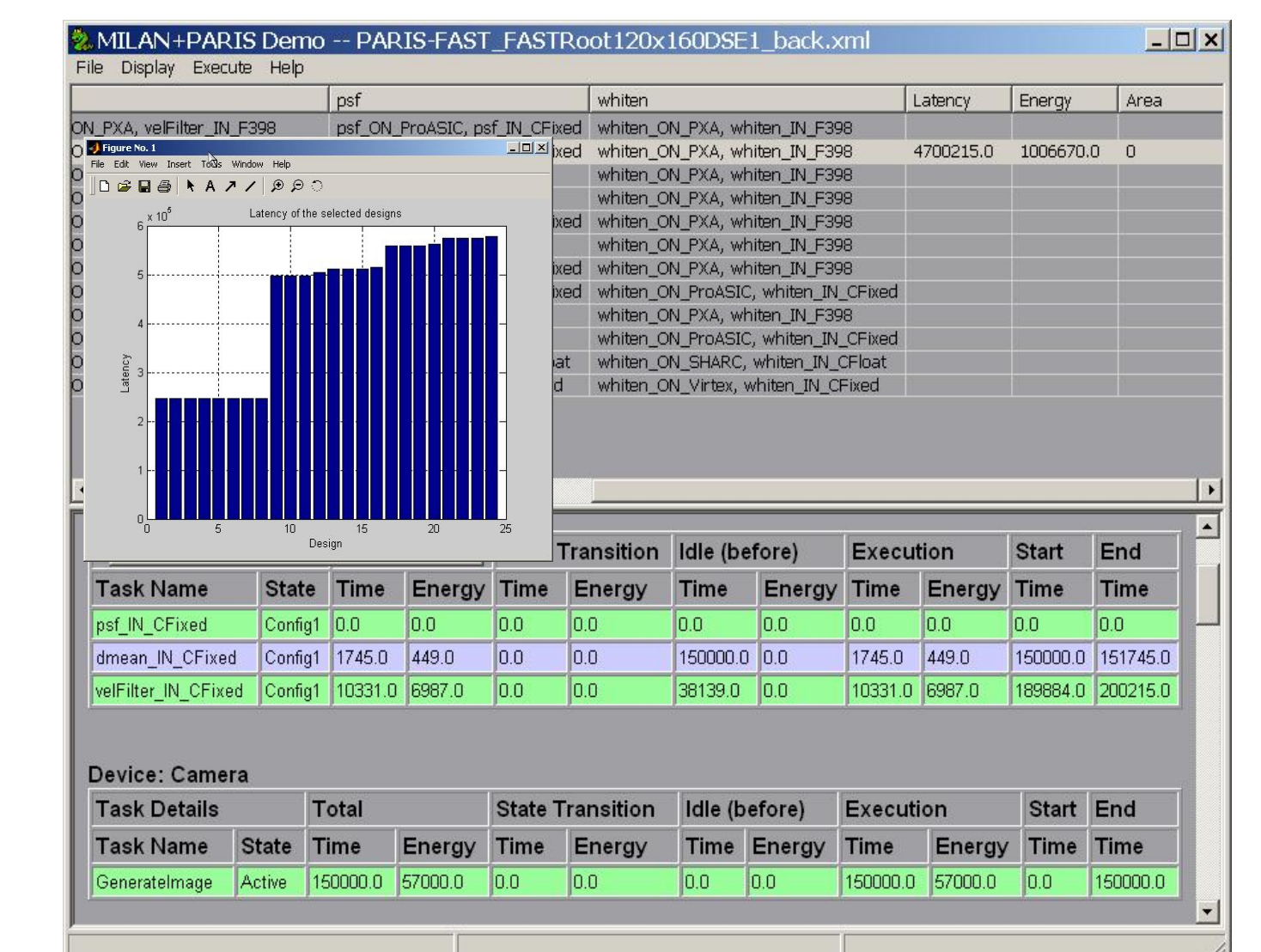
## Design Space Exploration using DESERT and HiPerE

- DESERT does not inherently support the idea of mapping onto adaptive computing systems
- Therefore, (in this work) we have used pseudo tasks to model reconfiguration
- Output of DESERT is provided as input to HiPerE
- HiPerE uses performance estimations to compare the designs to identify the final design

## Identifying Final Design

### Salient Aspects of Design Browser for HiPerE

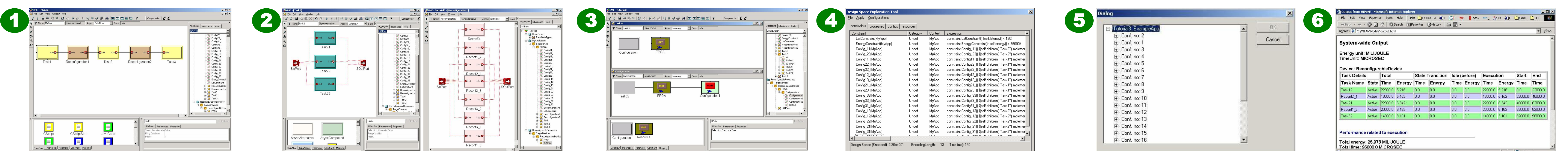
- Graphical interface between DESERT and HiPerE
- Automatically invokes HiPerE for one or more designs and shows activity reports
- Allows sorting of designs based on performance
- Designs can be compared using bar graphs
- Shows mapping and scheduling details for each design



### Constraint Specification

- Latency constraint:** The total latency of the application < CONSTANT
- Energy constraint:** The total energy consumption during execution < CONSTANT
- Transition constraint:** Task A is executed in State 1 and Task B is executed in State 2 implies Reconfiguration in between Tasks A and B is implemented by Option "Transition from State 1 to State 2"
- Mapping constraint:** Task A is executed in State 1 implies Task B is executed in State 2

## A Typical Design Flow for Optimization



- Model:**
- The application as a linear array of tasks
  - State transitions:
    - include a pseudo task between each pair of tasks

- Model:**
- The options for tasks and state transitions
  - Specify transition constraints that ensures that correct transition is selected

- Map each option of a task onto a hardware and an operating state of the hardware

- Specify the constraints:
  - for latency, energy, and mapping
- Invoke DESERT to determine acceptable designs

- Review the resulting designs
- Adjust the constraints:
  - too many designs -> tighten them, and go to step 4
  - too few designs -> loosen them, and go to step 4

- Apply HiPerE through the design browser
- HiPerE also provides an activity report per design which details mappings and state transitions

Tasks are the yellow boxes and state transitions are the brown ones.

The green boxes are the options of a task and the pink boxes are possible state transitions. The dashed lines indicate dataflow and the red boxes are inputs and outputs.

The gray box refers to the task option, the brown box refers to the mapped device, and the green box refers to the operating state.

The constraint selection dialog of DESERT allows the user to choose and apply the constraints. An estimate of the size of the design space is provided as well.

The final output of DESERT is a set of configurations that satisfy the constraints.

Design browser enables performance comparison of the designs. HiPerE outputs the results as an easy-to-read HTML file.

