# VMODEX: A Visualization Tool for Multi-Objective Design Space Exploration

Toktam Taghavi, Andy D. Pimentel
*Computer Systems Architecture Group, Informatics Institute*
*University of Amsterdam, Amsterdam, The Netherlands*
(*Demonstration Paper*)
{T.TaghaviRazaviZadeh, A.D.Pimentel}@uva.nl

***Abstract*—VMODEX is an interactive visualization tool to support system-level Design Space Exploration (DSE). It provides insight into the search process of Multi-Objective Evolutionary Algorithms (MOEAs) that are typically used in the DSE process, and therefore it facilitates the analysis of the DSE results. In our tool, we provide several capabilities to be able to handle large design spaces and filter design points according to their objective values to see only preferred solutions.**

## I. INTRODUCTION

Modern embedded and reconfigurable systems come with contradictory design constraints. On one hand, these systems often target mass production and battery-based devices, and therefore should be cheap and power efficient. On the other hand, they need to achieve high (real-time) performance and flexibility. The complexity of these systems forces designers to simulate systems and their components early during the design process to explore the wide range of design choices. Such design space exploration (DSE), during which multiple criteria should be considered simultaneously, is called multi-objective DSE. Since objectives are often in conflict, there cannot be a single optimum solution, which simultaneously optimizes all objectives. Instead, a set of optimal solutions denoted as the Pareto optimal set or non-dominated set has to be found. This is the set of those solutions for which one objective cannot be improved further without causing a simultaneous degradation in at least one other objective. These optimal decisions provide the designer trade-offs between the design objectives.

System-level simulation frameworks that are deployed for DSE, usually use independent application and architecture models. The application model describes the functional behavior of the system expressed as processes (computations) and channels (communications). The architecture model represents the hardware components in the system, such as processors, reconfigurable modules, memories, etc. Then, different mappings of processes and communication channels to various architectural components are evaluated by simulation to find the optimum mapping solutions. Each mapping decision taken in this step corresponds to a single point in the design space.

In order to find a Pareto optimal set with respect to the design criteria, the designer should ideally evaluate and compare every single point in the design space. However, such an exhaustive search is infeasible, as in real-scale problems the design space is too large to be explored in an exhaustive manner. Therefore, heuristic search techniques, such as multi-objective Evolutionary Algorithms (MOEA), are often used to search the design space for optimum design points using only a finite number of design-point evaluations. As the searched design space still is vast, interpreting all evaluation data and understanding how the EA searches through or prunes the design space is cumbersome. Such analysis is, however, essential to the designer as it provides insight into the "landscape" of the design space (e.g., indicating which design parameters are more important than others).

To illustrate the need for good analysis tools, Fig. 1 shows a sample of raw data generated by an EA. Here, each row represents an evaluated design point in which the values of objectives (processing time, energy consumption and cost) and the chromosome string are comma separated. The way that application tasks and their communications are mapped onto the architecture components is encoded in a string of digits, which is called the chromosome. It is evident that interpreting and analyzing the evaluated data in this format is not possible.

```
113,2703395,1297.71,120,403434146366644444
114,6295312,1343.13,140,334433444673776336
115,2545190,1389.61,210,423243417765266676
116,3906602,1169.77,130,323223136762267333
117,2545190,1116.96,200,403223417766667776
118,3421510,1136.81,160,404333136766637663
119,4719232,1047.58,140,303233116366677336
120,2281912,1243.07,160,303434136366646676
121,3906602,1170.56,130,323223136662267333
122,2525440,1138.57,200,404423317766664776
```

Fig. 1. Example of raw data generated by an EA

Therefore, we have developed a novel interactive visualization tool, VMODEX (Visualization of Multi-Objective Design spacE eXploration), to understand how an evolutionary algorithm searches the design space, where the optimum design points are located, how design parameters influence each objective, and provides insight into the relationship between the different objectives. The main challenge that needs to be addressed by such a visualization environment is how the raw data (as illustrated in Fig. 1) can be represented in a visual form such that it is possible to analyze the data – in a single view – from different perspectives and for various aspects. To this end, this paper proposes a visualization approach in which we visualize the design space as a tree in which both design parameters and objectives are shown.

The rest of the paper is organized as follows. Section II describes related work. Section III introduces techniques we have provided for visualizing multi-objective DSE. Section IV illustrates the benefits of using visualization in the DSE process. Finally, section V concludes the paper.

## II. Related Work

In the field of computer architecture simulation, and especially in the area of system-level design space exploration, little research has been undertaken on visualization of simulation results in exploring alternative architectural solutions. Most of the visualization work in this area focuses on educational purposes (e.g., [1], [2]), or only provides some basic support for the visualization of simulation results in the form of 2D / 3D graphs.

The work presented in [3] provides advanced and generic visualization support, but tries to do so for a wide range of computer system related information which may not necessarily be applicable to computer architecture simulations and in particular to design space exploration, with its own domain-specific requirements.

In [4], an interactive visual tool is presented to visualize the results from system-level DSE experiments. The simulation results are visualized using a coordinated, multiple-view approach, which enables users to understand the information through different perspectives. But this tool does not provide any insight in the searching process as performed by e.g. a MOEA. For example, there is no way to find out which parts of the design space are not searched at all.

## III. Demonstration Overview

### A. Modeling the Design Space as a Tree

As it is conceptually shown in Fig. 2, we model the design space as a tree. The tree has three sections: the Parameters section, Cost section and Design Points section.

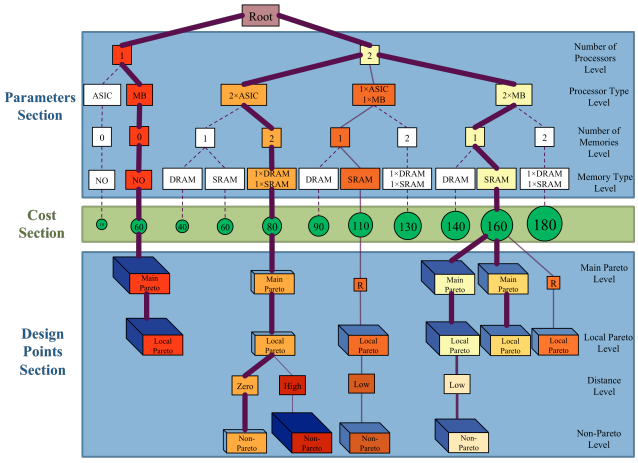


Fig. 2. Modeling the design space as a tree

In the Parameters section, each level shows one parameter of the design space, such as the number of processors in the architecture platform. So, the number of levels in this section is equal to the total number of parameters in the design space. For example, in the tree illustrated in Fig. 2, the design space has four parameters: number of processors, processor type, number of memories and memory type. In this example, the platform architecture consists of two Application Specific Integrated Circuits (ASICs), two MicroBlazes (MBs), one Static RAM (SRAM) and one Dynamic RAM (DRAM).

The design points' section includes the design points searched by the MOEA. Here, a design point is defined as a specific instance of the architecture platform as well as a task and communication mapping. Each point is shown as a node, which is a child of its corresponding architecture. Design points are distributed in three levels: main Pareto, local Pareto and non-Pareto.

The main Pareto level shows the global Pareto points found by the MOEA. The solutions at this level are better than all other solutions in the entire design space but they are non-dominated by each other. On the other hand, each point, which is not part of the main Pareto set, is dominated by at least one main Pareto point. At the local Pareto level, the local Pareto points are shown. A design point is called a local Pareto point if within the design points with the same architecture (but with different mappings), there is no point dominating that one. However, in the entire design space, a design point might exist which dominates the local Pareto point. It is clear that all the main Pareto points are local Pareto points as well. However, not all the local Pareto points are main Pareto points and therefore we use a relation node at the main Pareto level to make a connection between them and the previous level. These nodes are labeled with "R" in Fig. 2.

All the other design points are placed at the non-Pareto level. Each one becomes a child of a local Pareto point that dominates it. If a design point is dominated by more than one local Pareto point, we calculate the Euclidean distance (in the objective space) between the dominated point and each dominating local Pareto point and the design point becomes the child of the local Pareto point with the smallest distance. A smaller distance means that the points are more similar according to the objectives.

For easier interpretation and better analysis of the design points, the children of a local Pareto point are categorized into three groups according to their Euclidian distance from their parent. The solutions, which are equivalent to the local Pareto point with respect to all objectives, are put under the "Zero" distance node. If the distance between a solution and its corresponding local Pareto point is more than a certain threshold (determined by the designer), it becomes a child of a "High" distance node, otherwise it becomes a child of a "Low" distance node.

The color and thickness of edges show the Euclidean distance (in the objective space) from the nearest main Pareto point. The edges in the path from the root to the main Pareto points are the thickest and darkest since the distance is zero. As the distance increases the edges become thinner and lighter.

### B. Showing objectives in the tree

In this paper, we consider three objectives: processing time, energy consumption (i.e., power consumption times processing time) and architecture cost. The cost of each design point is dependent on the architectural components forming it. So, all solutions with the same architecture have the same cost. After the parameters section, the architecture cost can be computed since all components are known. Therefore, we add an extra section (Fig. 2) between the parameters section and design points section, which is called the cost section and shows the costs of the different architectures. Since the cost is an objective and not a design parameter, we represent it with a

different shape; a circle. For a better view, the size of the circle becomes bigger as the cost increases.

The other two objectives are dependent on the mapping and are therefore shown in a design point node. The size and color of the third dimension of a design point node shows the energy consumption. As the energy consumption increases, the size of the third dimension becomes bigger and its color becomes darker. The color of the node itself represents the processing time. Colors are varied from yellow to red with all color grades in between. Nodes with the lowest processing time are yellow and nodes with the highest processing time are red.

Parameter nodes, however, do not represent single design points and therefore do not have the direct notion of processing time or energy consumption. For this reason, there are some options to color the parameter nodes: based on the average, minimum, or maximum of either processing time or energy consumption of the design points in their sub trees. The color of parameter nodes that have no data node (i.e., do not have any DSE data) is white. In Fig. 2, the minimum processing time is chosen for coloring parameter nodes.

### C. Benefits of Tree Visualization

Modeling the design space based on a tree structure, as presented in this paper, has the following benefits:

Firstly, both the design space parameters and the objective values can be seen in one view. Therefore, it is easy to understand where the optimum design points are located and what objectives they have. Secondly, there is no limitation on the number of design variables since each parameter is located at one level of the tree. Therefore, modeling the design space as a tree enables us to easily visualize multivariate data. Lastly, it can easily be extended to show more than three objectives. Each node has some attributes like shape, orientation, size, color, transparency, texture, border, etc. Each attribute can be assigned to one objective. In this paper, only color and size are used to show objectives.

### D. Handling Large Trees

In reality, DSE trees can become extremely large. Therefore, we provide the following techniques to handle large trees.

*1) Satellite View:* Satellite view, gives an overall, smaller scale view of the entire scene, which allows the user to navigate quickly across the view. It also enables the user to zoom in on certain parts of the scene to focus on certain nodes without losing track of the position in the entire scene.

*2) Hiding Sub Trees without Exploration Data:* Since some areas of the design space may not have been visited by the searching algorithm (e.g., they are not interesting enough so we do not have any evaluated design points for those parts), it is possible to hide the sub trees of the nodes that have no data. This way, the designer can focus on the sub trees which are more important and can easily see which parts of the tree are searched by the EA.

*3) Hiding Uninteresting Sub Trees:* If the designer is not interested in some parts of the tree, then he is able to hide them in order to make the tree smaller and pay more attention to other nodes. By double clicking on a node, its sub tree collapses and a blue triangle appears at the bottom of the node specifying that the children of the node are hidden. The size of the triangle represents the size of the sub tree. The bigger the triangle is, the more nodes in the sub tree exist. By double clicking again, the sub tree becomes visible and the blue triangle is removed.

*4) Filtering:* In some cases, the designer wants to consider only design points with some specific objective values. The value of each objective is controlled by a range slider bar, in which the designer can set upper and lower limits on that objective. Design points with objective values inside the selected ranges are visible and the others become invisible. Therefore, the designer has the ability to easily view only preferred design points. There is an option to view all design points that fall within the filtering conditions or to only show local Pareto points or only main Pareto points.

### E. Detailed information

The DSE tree shows an overall view of the design space. For example, it shows where in the design space more design points have been evaluated or where the optimum design points (with respect to all objectives) are located. However, if the designer wants to know more about a specific design point, it is possible to select the design point to see more details. Two kinds of detailed information are provided for each design point: mapping decision and utilization.

*1) Showing Mapping Decision:* In our case, the application behavior is modeled as a process network. A process network is a computational model of the application and uses a directed graph notation, where each node represents a process and each edge represents a one-way FIFO communication channel between two processes. The Fig. 3a represents an example process network graph, which has five processes and six communication channels.

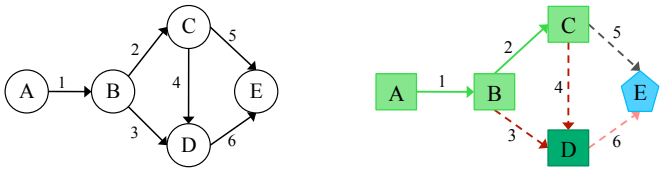


Fig. 3. a) An example of process network b) mapping decision visualization

We visualize the process network graph in a way that shows the mapping decisions as well. That means that it shows how the application is being mapped to the underlying architecture both in terms of processes and communication channels. The shape and the color of each node in the graph represent the type of the processor executing the corresponding process (i.e., a green rectangle for one processor type and a blue pentagon for another type). If there are multiple processors of the same type in the platform architecture, then they are differentiated using different variants of the same color such as light green and dark green.

If two communicating processes are mapped onto the same processor, then their communications are done internally and therefore communication channel(s) between them are mapped onto the processor in question. In the process network graph, a solid line represents these internal communications with the

same color as the corresponding processor. In the case that a channel is mapped onto an external memory, a dashed line is drawn with the color representing the memory type. Similar to the processors, memories with the same type are shown by a different variant of the same color.

The Fig. 3b represents how our visualization model shows the process network graph from Fig. 3a. As can be seen in this figure, processes A, B, C and channels 1 and 2 are mapped to the same processor (ASIC_1). Process D is executed on the same processor type but on a different processor as process A (ASIC_2). The type of the processor executing process E is different from the others since it is shown with a blue pentagon (MB_2). Channels 3,4,5 and 6 are mapped to memories (not processors) as they are shown with dashed lines. Channels 3 and 4 are mapped to the same memory (DRAM_1). Channel 6 is mapped to another memory but with the same type (DRAM_2) and Channel 5 is mapped onto a different memory type because it has a different color (SRAM).

*2) Showing Utilization:* For showing utilization, the platform architecture is shown as a directed graph. Each node represents an architectural component and the edges show connectivity between components. Each node (component) is filled with its corresponding colour, which is discussed in the above section, in a way that the size of the coloured part represents the percentage of the time the corresponding component was busy.

Fig. 4 shows an example of utilization visualization for a platform architecture consisting of two Application Specific Integrated Circuits (ASICs), two MicroBlazes (MBs), one Static RAM (SRAM) and two Dynamic RAMs (DRAM).

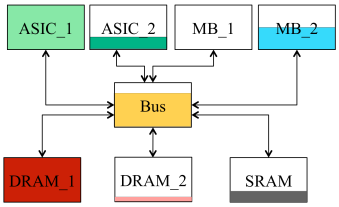


Fig. 4. Utilization visualization

In this example, the utilization of ASIC_1 and DRAM_1 is 100% while for the Bus it is almost 75% and for the other components it is less than 50%.

## IV. Evaluation

VMODEX enables designers to easily and clearly understand the DSE process and analyse the results from different aspects. Due to the lack of the space, a detailed study of the design space exploration data for a particular design is not presented here. However, to illustrate the benefits of using our tool in the DSE process, in the following, we mention some interesting conclusions that a designer can immediately draw just by looking at the visualization and could not be made so easily by using only the raw data or traditional 2D/3D graphs.

First of all, it shows which parts of the design space are not searched at all (no design point is evaluated there). As we mentioned before, nodes with a white colour and dashed line have no data. Furthermore, it illustrates which parts of the design space are searched more often by the EA (more design points are evaluated there). In these areas, the tree provides more nodes so the sub trees of the corresponding nodes are bigger. Moreover, it shows the parts of the design space that contain the main Pareto points. Therefore the designer can immediately recognize which combinations of architectural components yield optimum design points. Our visualization enables the user to see the design variables (architecture components) of the Pareto points and their objective values in one view.

Next, it points out the poor design points. By poor, we mean the distance (in objective space) between them and the nearest main Pareto point is big. The edges between them and their parents are thinner and lighter.

As we mentioned before, for each architecture instance, the best design points with respect to the design criteria are located at the local Pareto level. Therefore, the designer can easily compare the best design points of different architecture instances with each other.

By coloring the parameter nodes, it is possible to do some statistical analysis. The designer can compare different architectures (in terms of number and type of the processors and memories) according to the minimum, maximum or average of each of the design criteria.

By visualizing the mapping decision, it is easy to investigate the influence of the different mappings on each objective. In addition, by visualizing the utilization, the effect of different mappings on the utilization of the architecture components can be easily compared.

By using the filtering option, the designer can select to see only design points with desirable objective values and understand which parts of the design space contain the preferred design points.

## V. Conclusion

In this paper, we presented a visualization tool, VMODEX, which helps designers to understand the search behavior in MOEA based design space exploration as well as to gain insight into the landscape of the design space. That is, understanding the characteristics of the optimum design points with respect to the design criteria, the relationships between design parameters and their effects on the objectives, the effects of mapping decisions on the design criteria and the correlations among multiple objectives.

In our tool, we provide several capabilities to be able to handle large design spaces and filter design points according to their objective values to see only preferred solutions. Besides, we discussed some of the interesting conclusions that can be immediately drawn by looking at our visualizations.

## References

[1] P. Marwedel, B. Sirocic. Multimedia components for the visualization of dynamic behavior in computer architectures, in the Proc. of the Workshop of Computer Architecture Education, 2003.

[2] C. Yehezkel, W. Yurcik, M. Pearson, D. Armstrong. Three simulator tools for teaching computer architecture: Easycpu, little man com- puter, and rtlsim, Journal on Educational Resources in Computing (JERIC), vol. 1, no. 4, pp. 60-80, 2001.

[3] R. Bosch, et al, “Rivet: A flexible environment for computer systems visualization”, SIGGRAPH Computer Graphics, vol. 34, no. 1, 2000.

[4] T. Taghavi, A. D. Pimentel, and M. Thompson. “Visualization of Computer Architecture Simulation Data for System-level Design Space Exploration”, in *Proc. SAMOS '09'*, July 2009.