Visualization of Multi-Objective Design Space Exploration for Embedded Systems

Toktam Taghavi, Andy D. Pimentel
Computer Systems Architecture Group
Informatics Institute, University of Amsterdam
Amsterdam, the Netherlands
{T.TaghaviRazaviZadeh, A.D.Pimentel}@uva.nl

Abstract— **Modern embedded 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. This wide spectrum of design requirements leads to complex heterogeneous system-on-chip (SoC) architectures. The complexity of embedded systems forces designers to model and simulate systems and their components to explore the wide range of design choices. Such design space exploration is especially needed during the early design stages, where the design space is at its largest.**

Due to the exponential design space in real problems and multiple criteria to be considered, multi-objective evolutionary algorithms (MOEAs) are often used to trim down a large design space into a finite set of points and provide the designer a set of tradable solutions with respect to the design criteria.

Interpreting the search results (e.g., where are the Pareto points located), understanding their relations and analyzing how the design space was searched by such searching algorithms is of invaluable importance to the designer. To this end, this paper presents a novel interactive visualization tool, based on tree visualization, to understand the search dynamics of a MOEA and to visualize where the optimum design points are located in the design space and what objective values they have.

Keywords-component; design space exploration; visualization; multi-objective evolutionary algorithms; embedded systems

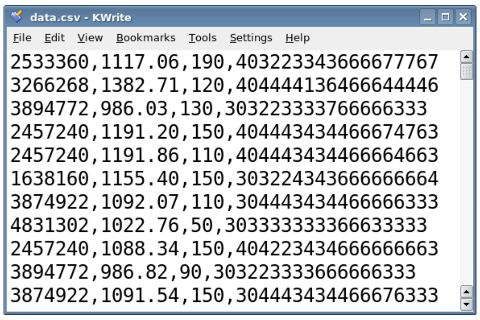
I. Introduction

The complexity of today's embedded systems forces designers to start with modeling and simulating system components and their interactions in the very early design stages. It is therefore crucial to have good tools for exploring a wide range of design choices, especially during the early design stages, where the design space is at its largest. In the Sesame framework [1,2], a modeling and simulation environment is developed for the efficient design space exploration (DSE) of embedded systems that are based on heterogeneous Multi-Processor System-on-Chip (MP-SoC) architectures. Models in Sesame are defined at a high level of abstraction and capture only the most important characteristics of the components in the system.

Sesame maintains independent application and architecture models and relies on the co-simulation of these two models. As a consequence, it needs an explicit *mapping step* which relates each task (i.e., process) and communication channel in the application model to a processor/memory component in the architecture model. Each mapping decision taken in this step corresponds to a single point in the design space (note that the terms point, solution and decision vector are used interchangeably in this paper). In order to achieve an optimal design, the designer should ideally evaluate and compare every single point in this space. However, such exhaustive search quickly becomes infeasible, as the design space grows exponentially with the size of the application(s) and the number of possible architecture components.

In general, to trim down an exponential design space into a finite set of points, which are more interesting (or superior) with respect to some chosen design criteria, design space pruning can be used. In [3], e.g., the mapping decision problem is formulated as a multi-objective optimization problem in which three criteria are considered: the processing time, energy consumption and cost of the architecture. To solve this problem, an Evolutionary Algorithm (EA) has been used to achieve a set of best alternative mapping decisions under the aforementioned multiple criteria. 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.



```
2533360,1117.06,190,403223343666677767
3266268,1382.71,120,404444136466644446
3894772,986.03,130,303223333766666333
2457240,1191.20,150,404443434466674763
2457240,1191.86,110,404443434466664663
1638160,1155.40,150,303224343666666664
3874922,1092.07,110,304443434466666333
4831302,1022.76,50,303333333366633333
2457240,1088.34,150,404223434666666663
3894772,986.82,90,303223333666666333
3874922,1091.54,150,304443434466676333
```

Figure 1. Example of raw data generated by an EA

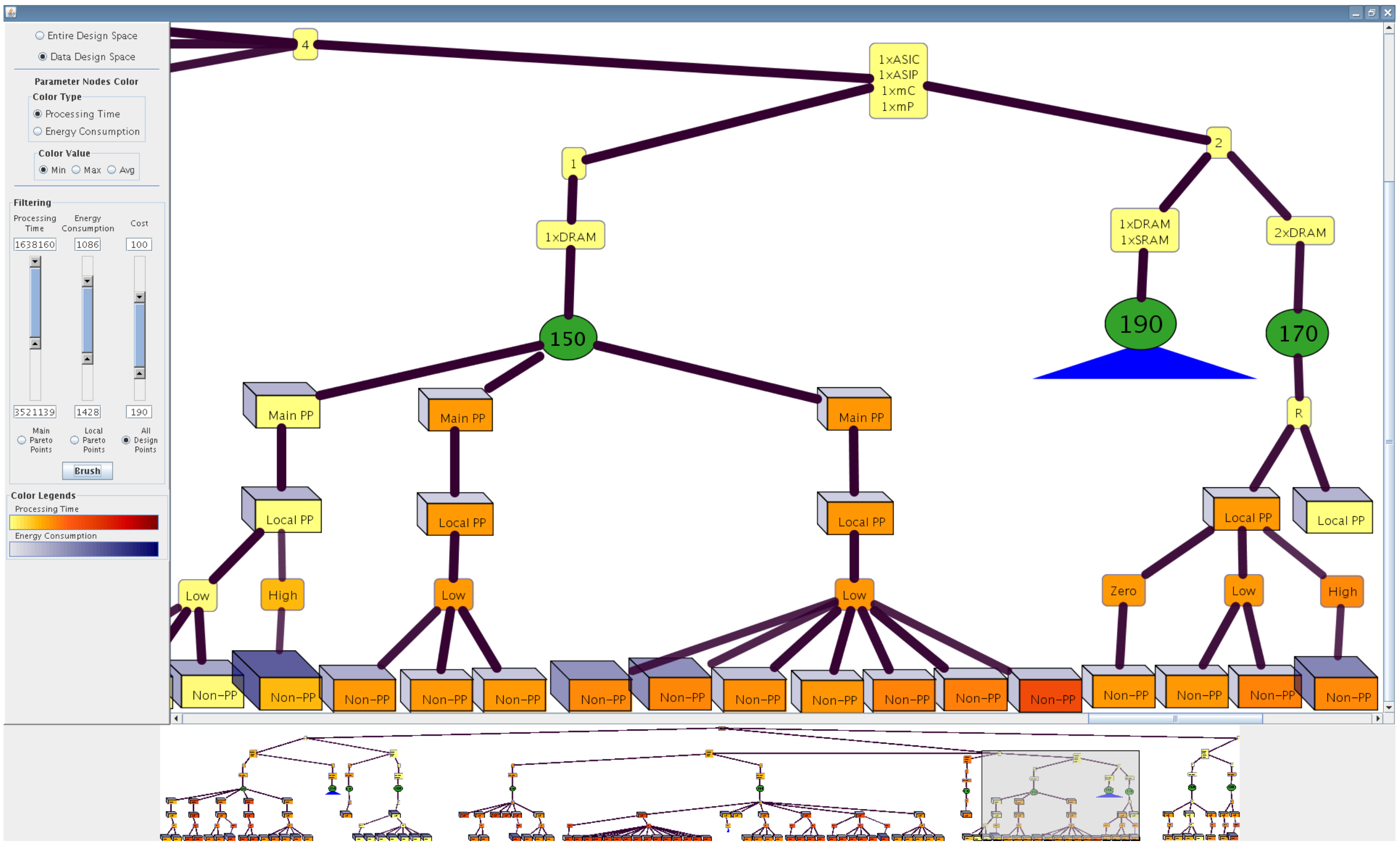


Figure 2. Screenshot of the multi-objective visualization

Therefore, we have developed a novel interactive visualization tool, VMODEX [1], to understand how an evolutionary algorithm, such as presented in [3], 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. To give a rough feeling of how such visualization looks like, Fig. 2 shows a screenshot of our visualization application.

The rest of the paper is organized as follows. Section II describes related work. In section III, we briefly explain some preliminary definitions of multi-objective optimization. Section IV introduces techniques we have provided for visualizing multi-objective design space exploration. Section V presents a case study with a Motion-JPEG encoder application to illustrate the benefits of using visualization in the design space exploration process. Finally, section VI 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., [4,5]), or only provides some basic support for the visualization of simulation results in the form of 2D (and sometimes 3D) graphs.

The work presented in [6,7] 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 [8], an interactive visual tool is presented to visualize the results from system-level design space exploration experiments. The simulation results are visualized using a coordinated, multiple-view approach which enables users to understand the information through different perspectives. It is possible to compare different design points with respect to various characteristics and gain more insight in the performance landscape of the design space. 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.

There are only a few research efforts addressing the visualization of MOEAs. Most visualization approaches simply use standard visual representations such as bar charts, line graphs, scatter plots, etc. [9]. Although such diagrams show the quality of the solutions considered during search process, they do not show anything about the properties of the solutions

[1] Visualization of Multi-Objective Design spacE eXploration

being searched, or the regions of the search space being explored.

More complex techniques have focused on how to display the progress of the MOEA in variables (parameters) space or objectives space [10,11]. Usually they use 2D or 3D plots in which either variables or objectives are shown. Therefore, two separate views are needed to show the distribution of the solutions in both variables and objectives spaces. Furthermore, due to the large number of dimensions in practical problems, techniques such as Sammon Mapping [12] should be used to transform higher dimensional search spaces into smaller ones. Multi-objective visualization, as presented in this paper, enables us to easily visualize more than three dimensions as well as to show variables and objectives in one view.

This paper proposes an extension to our previous work [13], in which the visualization techniques have been extended to show multiple objectives and provide insight into the "landscape" of the multi-objective optimization process. Furthermore, a detailed study of the design space exploration for a particular design is presented to show how the proposed visualization can help the designer to explore the design space.

III. MULTI-OBJECTIVE OPTIMIZATION

Real world design problems often are multi-objective optimization problems in which two or more conflicting objectives should be optimized simultaneously. An improvement in one objective often causes deteriorations in other objectives. Therefore, optimal decisions need to be taken in the presence of trade-offs between objectives. On the other hand, in multiple conflicting objectives problems, there cannot be a single optimum solution which simultaneously optimizes all objectives; instead, a set of optimal solutions, denoted as the Pareto optimal set, with a varying degree of objective values has to be found.

A. Domination and Pareto optimality

Most multi-objective optimization algorithms use the concept of domination. In these algorithms, two solutions are compared on the basis of whether one solution dominates the other solution or not. In this subsection we will briefly describe the concept of domination and Pareto optimality.

Definition 1: A general multi-objective optimization problem with *m* decision variables (parameters) and *n* objective functions is defined as:

$$\begin{aligned} &\text{Minimize} && y = f(x) = (f_1(x),\ldots, f_n(x)) \\ &\text{Where} && x = (x_1,\ldots, x_m) \in X \\ & && y = (y_1,\ldots,y_n) \in Y \end{aligned}$$

The objective function f(x) maps a decision vector (solution) x in decision space (X) to an objective vector y in objective space (Y). Two different solutions are related to each other in two possible ways: either one dominates the other or none of them is dominated.

Definition 2: A solution $x_1 \in X$ is said to dominate the other solution $x_2 \in X$ (also written as $x_1 < x_2$) if and only if both of the following conditions are true:

1. The solution x_1 is not worse than x_2 in all objectives; formally: $\forall i \in \{1,\ldots,n\}: f_i(x_1) \le f_i(x_2)$
2. The solution x_1 is strictly better than x_2 in at least one objective; formally: $\exists j \in \{1,\ldots,n\}: f_j(x_1) < f_j(x_2)$

It is intuitive that if x_1 dominates the solution x_2, the solution x_1 is better than x_2 in the context of multi-objective optimization. It is also common to say "x_2 is dominated by x_1" instead of saying "x_1 dominates x_2".

Definition 3: Let $x_1 \in X$ be an arbitrary solution (decision vector)

- The solution x_1 is said to be non-dominated regarding a set $X' \subseteq X$ if and only if there is no solution in X' which dominates x_1; formally: $\nexists\, x_2 \in X' : x_2 < x_1$
- The solution x_1 is called Pareto optimal if and only if x_1 is non-dominated regarding the whole decision space X.

Pareto-optimal solutions cannot be improved further in terms of a certain objective without causing a simultaneous degradation in at least one other objective. They represent in that sense globally optimum solutions. Any solution which does not belong to the Pareto optimal set is dominated by at least one Pareto optimal solution. The set of objective vectors corresponding to a set of Pareto optimal solutions is called "Pareto optimal front" or "Pareto front".

IV. MULTI-OBJECTIVE VISUALIZATION

A. Modeling the Design Space as a Tree

As it is conceptually shown in Fig. 3, we model the design space as a tree. The tree has three sections: the Parameters section, Cost section and Design Points section.

In the Parameters section, each level shows one parameter of the design space, such as the number of processors in the MPSoC 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. 3, 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 microprocessors (mPs), 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 true 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

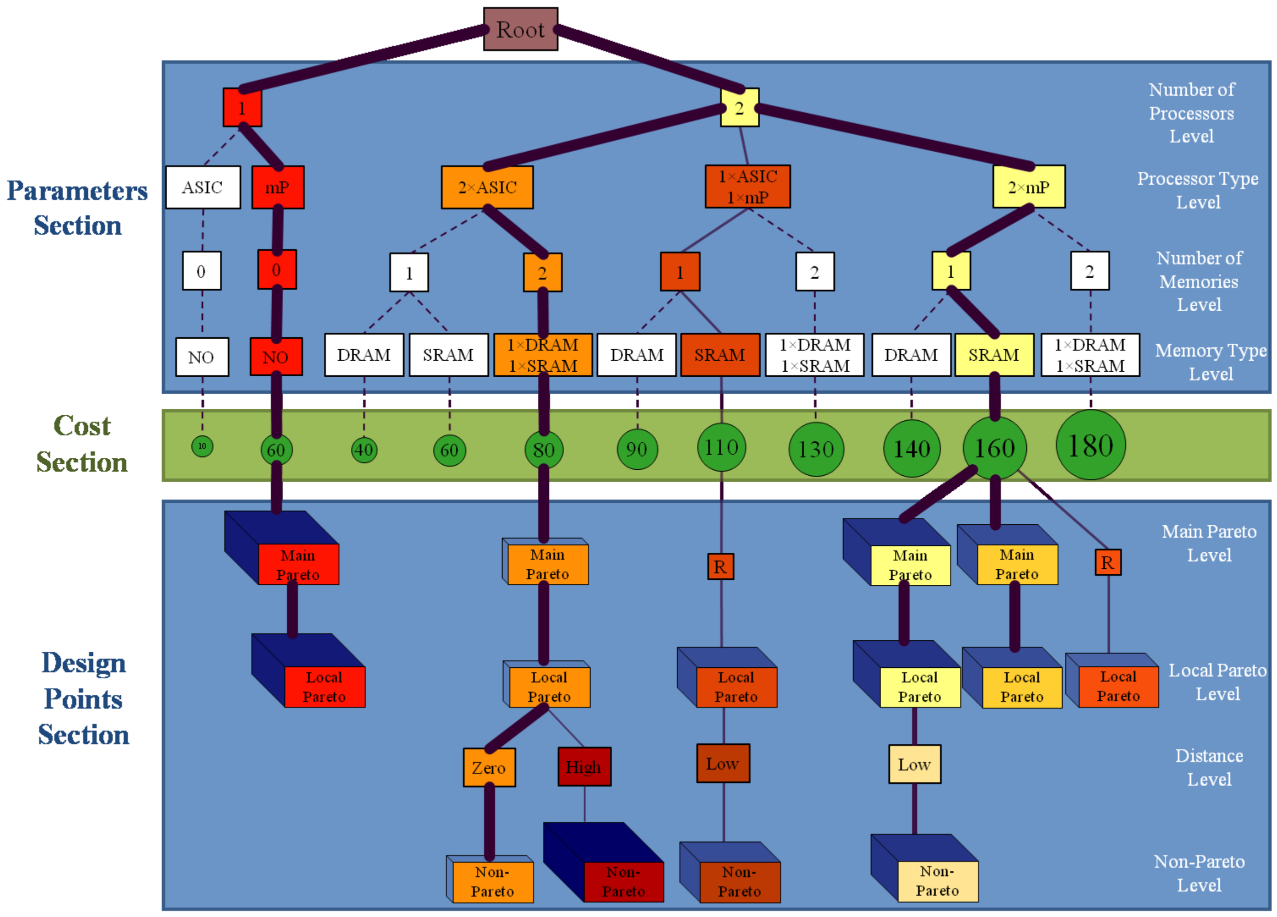


Figure 3. Modeling the design space as a tree

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. 3.

All the other design points are placed at the non-Pareto level. Each one becomes a child of a local Pareto point which 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 (Figure 3) 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. The color legends for processing time and energy consumption can be seen in left side of Fig. 2.

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. 3, 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:

- 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.
- 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. It should be mentioned that, in principle, the designer has total freedom of ordering the parameters in the levels of the tree. However, putting more important parameters higher up in the tree facilitates the information organization in such a way that it produces sub trees which are more likely to show a better view of the design space characteristics. Because the more important design points (according to the design parameters) are clustered in only one sub tree, the designer can easily select that sub tree to investigate and compare these design points. But by putting more important parameters down in the tree, the design points with the same parameter are distributed in several sub trees.
- 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, shown at the bottom of Fig. 2, 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 in the tree without losing track of the position in the entire scene.

2) *Hiding Sub Trees without Exploration Data*

Since some areas of the design space are not 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 becomes invisible 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, the more nodes in the sub tree. By double clicking again, the sub tree becomes visible and the blue triangle is removed.

It should be mentioned that by hiding a node, the entire tree will be redrawn, meaning that the empty space from that node will be used by the other nodes. We recalculate the location of visible nodes to optimize their fit to the screen.

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: exact objective values and mapping decision.

1) *Exact Objective Values*

Instead of showing objectives with visual variables (color and size), this option shows the exact values of processing time and energy consumption. It also represents the normalized value of the objectives. We normalize objective values to make them scale independent. At the end of normalization, all design points get a value in the range [0, 1] for their objective values.

Before normalization, it is not possible to compare e.g. processing time and energy consumption with each other since they have different magnitudes. However, after normalization, comparing them is possible.

2) *Showing Mapping Decision*

In the Sesame simulator, 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. Fig. 4 represents an example process network graph which has five processes and six communication channels.

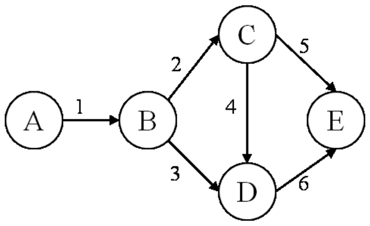


Figure 4. An example of process network graph

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. For example, 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 these internal communications are represented by a solid line 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. Fig. 5 shows how our visualization model shows the process network graph from Fig. 4.

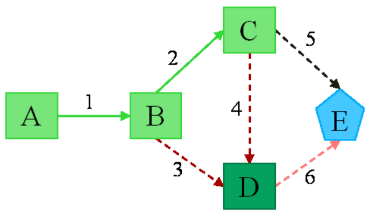


Figure 5. Mapping decision

As can be seen in this figure, processes A, B, C and channels 1 and 2 are mapped to the same processor. Process D is executed on the same processor type but on a different processor as process A. The type of the processor executing process E is different from the others since it is shown with a blue pentagon. 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. Channel 6 is mapped to another memory but with the same type and Channel 5 is mapped onto a different memory type because it has a different color.

To be able to find out what objectives are achieved for a particular mapping, the objective values are also shown together with the mapping. They are represented in the same way as multi-objective visualization (shown in Fig. 6).

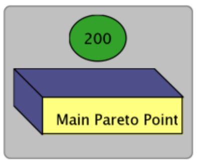


Figure 6. Representing Objective values

V. A CASE STUDY

In this section, we present a real application case study to illustrate the benefits of using visualization in the design space exploration process. In this case study, we map a Motion-JPEG (M-JPEG) encoder to an MP-SoC platform architecture consisting of a general-purpose microprocessor (mP), a microcontroller (mC), an application specific instruction processor (ASIP), two Application Specific Integrated Circuits (ASICs), one SRAM and two DRAMs. The M-JPEG encoder process network is shown in Fig. 7.

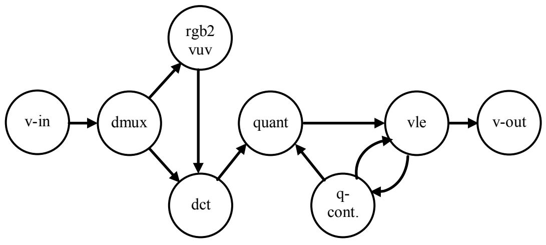


Figure 7. M-JPEG process network

In our case study, an mC or an mP processor can execute all the different processes in the M-JPEG application while an ASIP can execute only three processes, namely: “dct”, “quant” and “rgb2yuv”. We also assume one ASIC is designed for executing the “dct” process and another one is designed for the “v-in” and “v-out” processes.

Using a multi-objective evolutionary optimizer [3], we intend to find a set of optimal design points (in terms of alternative architectural solutions and mappings) under three criteria: processing time, energy consumption and architecture cost.

For this study, we run the EA for 60 generations with 50 individuals per population. Therefore, 3000 design points are searched by the EA. Fig. 8 shows a snapshot of the visualization of the M-JPEG case study. It should be mentioned that the purpose of Fig. 8 only is to show an overview of the entire design space and it is not meant to be readable. Just by looking at the depicted tree, the designer can immediately understand some general information about the design space searched by the EA. For example, it is obvious that there is no design point evaluated for single processor architecture platforms (node color is white). Moreover, in two processors platforms, there are five different combinations of processor components which are capable of executing the application but only three of those combinations are searched. However, in four processors platforms, all the possible combinations are explored.

From this picture, we can also find out that most of the evaluated design points have one or two memories, and architectures with three memories are rarely searched. Only architectures consisting of all different processor types have some design points with three memories.

In Fig. 8(b), parameter nodes that have no data are omitted. In this figure, minimum processing time is used for coloring the parameter nodes. From this figure, it is clear that platform instances with two processors are searched less than others since their blue triangles are small. We can also see that most of the design points being searched by the EA contain two memories; one DRAM and one SRAM because all the biggest blue triangles have these two memories. Moreover, as can be seen in this figure, all design points with the minimum processing time include at least one ASIP, one mC and one mP (node color is yellow). Another conclusion is that, by adding an ASIP to the architectures without an ASIP, we can get a significant improvement of processing time. This is illustrated by the designs indicated by “A” and “B” in Fig. 8(b). Where architecture A consist of one mC and one mP, in architecture B an ASIP is added.

Fig. 9 shows the main Pareto points found by the EA. By looking at the picture, the designer can immediately recognize the characteristics of the main Pareto points, which are the best design points with respect to the design criteria. For example, in our case study, there is no main Pareto point with five processors. That means that with less processors (which is cheaper) the designer can get the same or better processing time and energy consumption. Therefore, using five processors is not appropriate for this application. Another interesting feature is that all the main Pareto points have a microcontroller in their underlying architecture. The slow but cheap and low-power microcontroller apparently provides a good tradeoff for executing the less demanding tasks in the application. It can also be seen that all the main Pareto points have at least one DRAM memory. A few of them have one SRAM besides the DRAM. Thus, using three memories or two DRAMs is not an appropriate solution in this case study. So, by using VMODEX, the designer can easily find out which combinations of architectural components yield optimum design points.

In the following, we carry out a more detailed analysis of the design space using our visualization tool. For instance, we want to investigate the effect of adding one SRAM memory to the architecture. As can be seen in Fig. 9, for every main Pareto point with one SRAM and one DRAM memory, there exists a main Pareto point with only one DRAM memory which basically offers the same processing time and energy consumption (such as the design points indicated with A and B in Fig. 9). By looking at their exact objective values (blue

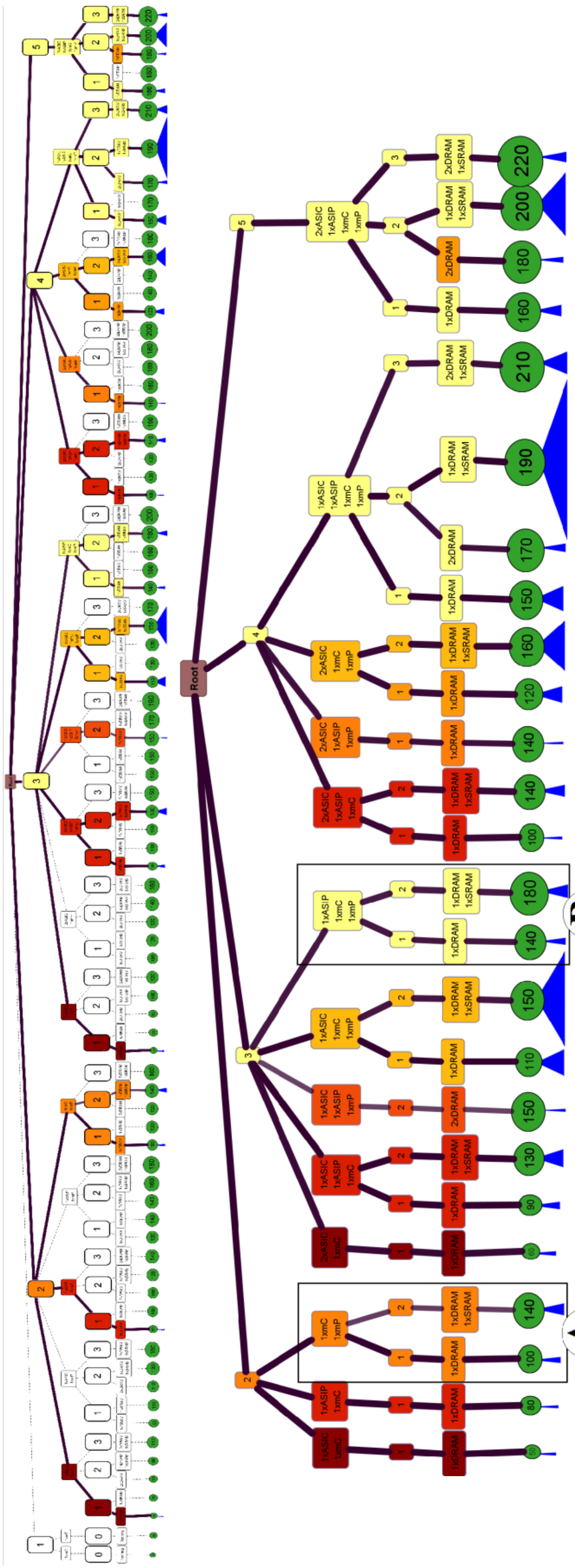


Figure 8. Screenshot of the M-JPEG case study (a) Entire design space, (b) Parts of the design space having DSE data

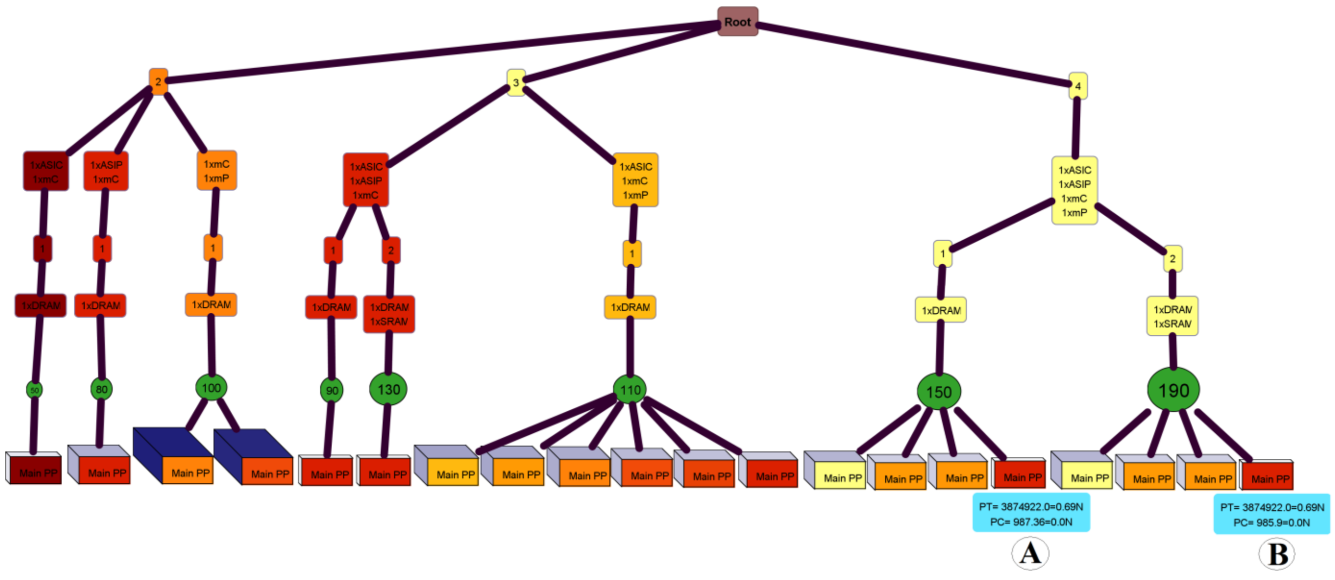


Figure 9. Main Pareto Points

rectangle), we find out that adding one SRAM memory to the architecture does not yield any improvement in processing time and improves energy consumption only a little bit while the architecture cost is increased a lot. Therefore, in our case study, using an additional SRAM next to a DRAM memory is not beneficial.

Now, let us zoom in on the part of the design space which contains two or three processors and two memories. Here, we investigate the local Pareto points which are shown in Fig. 10.

As we mentioned before, for each specific architecture, the best design points with respect to the processing time and energy consumption are located at the local Pareto level. Needless to say, local Pareto points with the same parent have the same cost since they use the same architectural components. Considering the local Pareto set indicated by “A” in Fig. 10, we have a good tradeoff between processing time and energy consumption. There are some design points with good processing time but poor energy consumption and also some design points with poor processing time but good energy consumption. So, in the case that there is a system where applications can dynamically be mapped, this architecture could be suitable. In the case the system needs to be optimized for speed, the mapping with the lowest processing time can be selected. But when the system needs to safe energy consumption (e.g. in case the battery is running low), then the mapping with the lowest energy consumption can be selected.

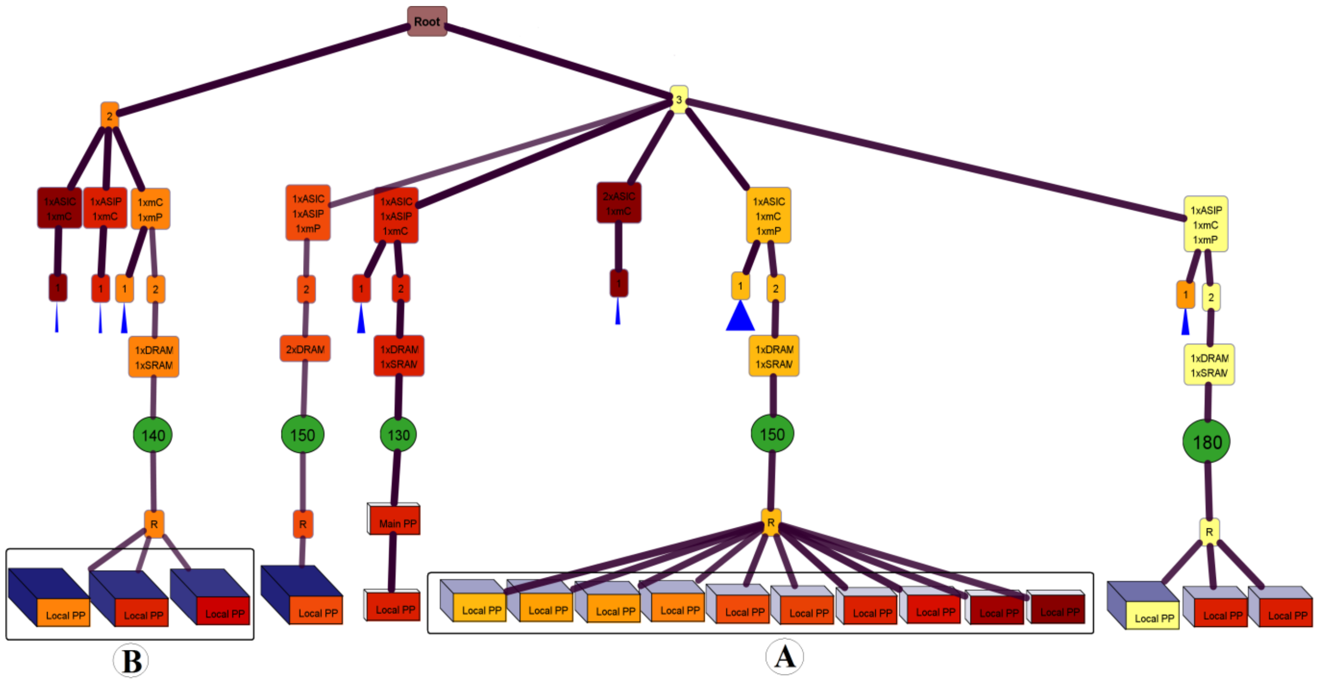


Figure 10. Local Pareto Points

Considering two local Pareto sets indicated by "A" and "B" in Fig. 10, the architectural design points denoted by A have an extra ASIC and, as a result, obtain much better energy consumption results while the cost only increases a little bit.

Fig. 11 shows three-processor architectures of the design space in which the children of only two combinations are visible. The processing time of all design points with one ASIC, one ASIP and one mC (the left sub tree) is extremely poor. Even with different mappings, we cannot get a good processing time. However, the energy consumption is relatively good for these design points. Therefore, if the designer is interested in lower processing times, this architecture is not a suitable solution but if he prefers low energy, this architecture could be a good choice.

In the right sub tree the situation is reversed. Depending on the mapping, you may get a very good (e.g. the design point indicated by A) or a very poor (e.g. the design point indicated by B) processing time. However, the energy consumption of these design points is quite high. Therefore, this architecture is not appropriate for obtaining low energy but if the designer is interested in performance, he should take care about the mapping because a wrong mapping decision can make the difference between the best or the worst processing time.

To investigate why the design points A and B significantly differ in their processing times, the mapping decisions of these design points are shown in Fig. 12. As can be seen in this figure, in design point A, tasks "v-in" and "dmux" are mapped on the mC component (brown triangle) and tasks "vle" and "v-out" are mapped on the mP (violet circle) while in design point B this is the other way around. For all the other tasks, the processors executing them are the same. In design point B, the mC component forms a bottleneck for the application's throughput. Such visualization of mapping decisions clearly enables the designer to study the effect of different mappings on the design criteria.

To show the benefits of the filtering option described in section IV.D, we apply a filtering scenario to our case study. We are interested in those design points which are good

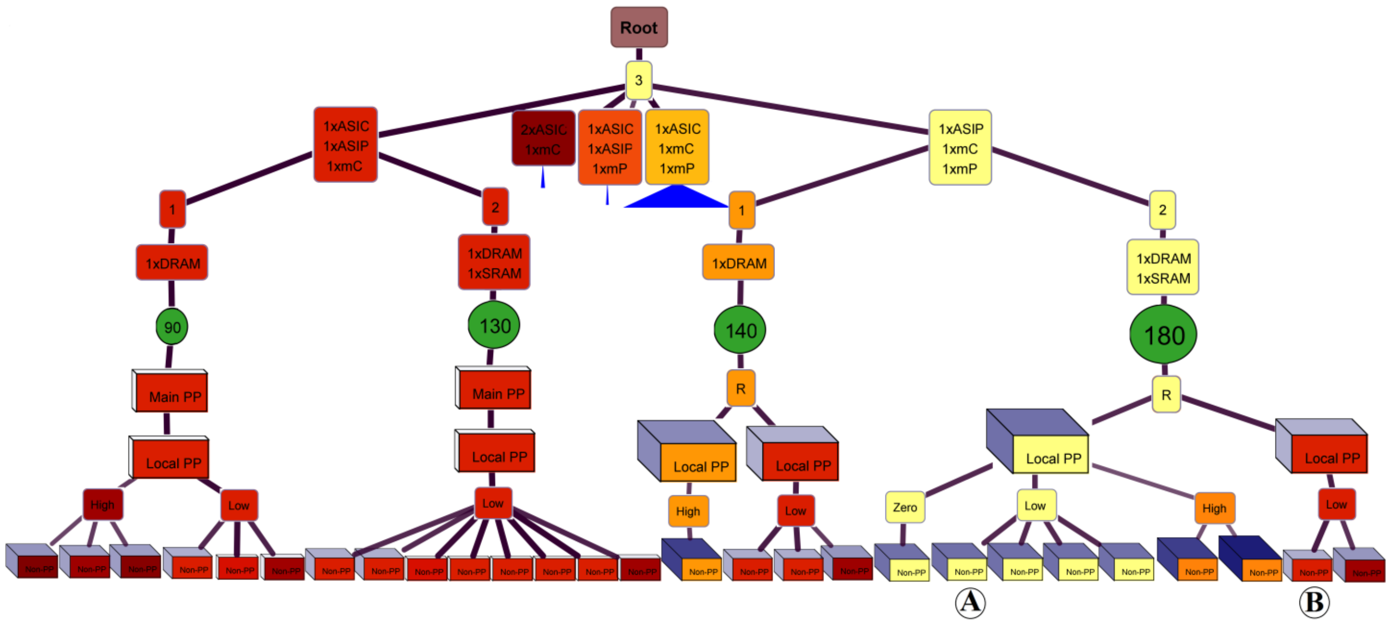


Figure 11. Design Points

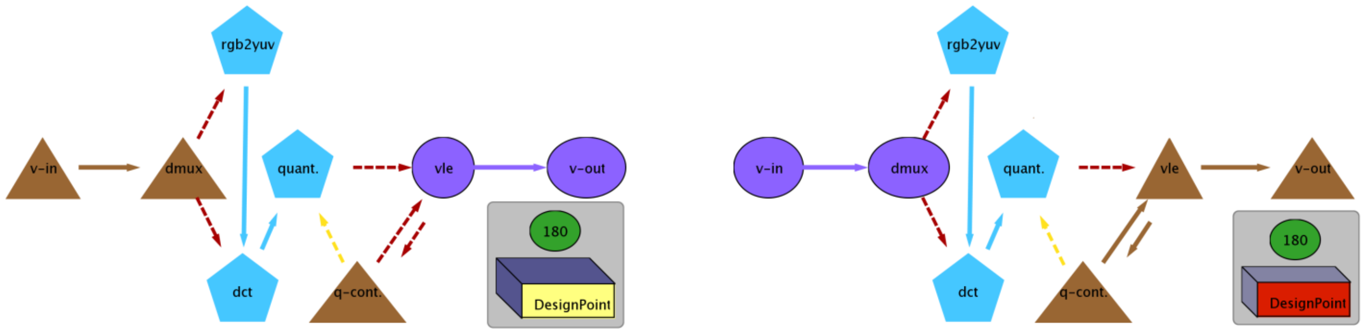


Figure 12. Mapping decision: (a) design point A, (b) design point B

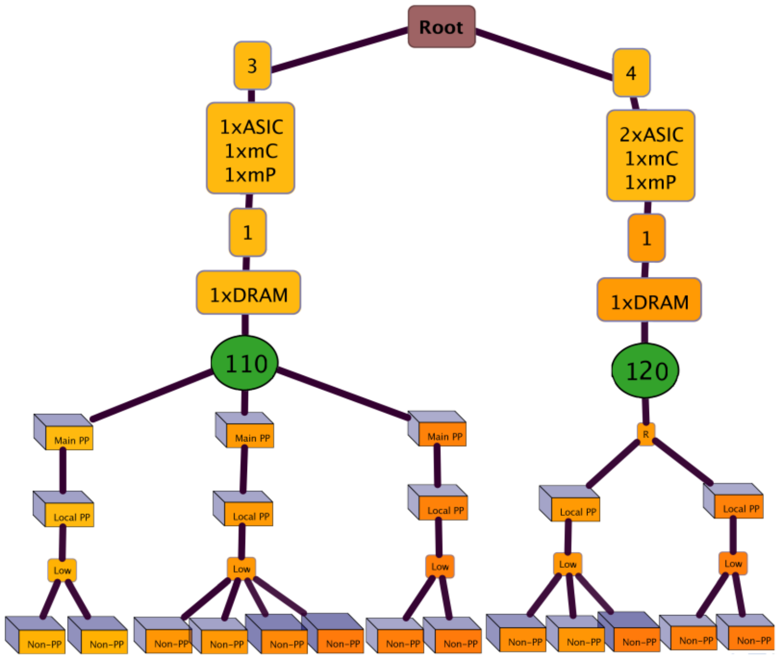


Figure 13. Filtering

enough in all three objectives. By good enough we mean those design points which are better than the median in all the three objectives. Fig. 13 represents the design points that fall within this condition. As it is shown, only two architecture platforms provide design points that are relatively good in all objectives. Fig. 13 is the filtered version of Fig. 8. As it can be seen, a lot of design points have been omitted which can help the designer to focus on more important design points.

The analysis we performed in this section would have been very cumbersome and time consuming to do by only looking at the raw data or by using traditional 2D/3D graphs. Several traditional graphs are needed in order to interpret the data like we did. However, using VMODEX, a single visualization view of the design space enables very powerful and rapid analysis of the DSE data.

VI. 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. We have also illustrated the benefits of such visualization using a Motion-JPEG encoder case study.

REFERENCES

[1] C. Erbas, A. D. Pimentel, M. Thompson, and S. Polstra. A Framework for System-level Modeling and Simulation of Embedded Systems Architectures, EURASIP Journal on Embedded Systems, 2007, DOI 10.1155/2007/82123.

[2] A.D. Pimentel, C. Erbas, and S. Polstra. A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels, IEEE Transactions on Computers, vol. 55, no. 2, pp. 99-112,(2006).

[3] C. Erbas, S. Cerav-Erbas and A.D. Pimentel. Multiobjective Optimization and Evolutionary Algorithms for the Application Mapping Problem in Multiprocessor System-on-Chip Design, IEEE Transactions on Evolutionary Computation, pp. 358-374, Vol. 10 (No. 3), June 2006.

[4] 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.

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

[6] R. Bosch, et al, Rivet: A flexible environment for computer systems visualization, SIGGRAPH Computer Graphics, vol. 34, no. 1, pp. 68-73, 2000.

[7] R.P. Bosch. Using Visualization to Understand the Behavior of Computer Systems, PhD thesis, Stanford University, 2001.

[8] T. Taghavi, A. D. Pimentel, and M. Thompson. Visualization of Computer Architecture Simulation Data for System-level Design Space Exploration, in the Proc. of Int. Symposium on Systems, Architectures, Modeling and Simulation (SAMOS '09), July 2009.

[9] E. Hart and P. Ross. GAVEL - A New Tool for Genetic Algorithm Visualization, IEEE Trans on Evolutionary Computation, Vol. 5, No. 4, pp. 335-348, August 2001.

[10] H. Pohlheim. Visualization of evolutionary algorithms — set of standard techniques and multidimensional visualization, in the Proceedings of the 1999 Genetic and Evolutionary Computation Conference GECCO'99, Morgan Kaufmann, Los Altos, CA, 1999, pp. 533–540.

[11] T.D. Collins. Applying software visualization technology to support the use of evolutionary algorithms, Journal of Visual Language and Computing 14 (2003), 123-150.

[12] Sammon, J. W. jr. A Nonlinear Mapping for Data Structure Analysis. IEEE Transactions on Computers, vol. C-18, no. 5, pp. 401-409, 1969.

[13] T. Taghavi, A.D. Pimentel, M. Thompson, "System-level MP-SoC Design Space Exploration using Tree Visualization", in the Proc. of the IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia '09), Grenoble, France, Oct. 2009.