VMODEX: A Novel Visualization Tool for Rapid Analysis of Heuristic-Based Multi-Objective Design Space Exploration of Heterogeneous MPSoC Architectures

Toktam Taghavi[a], Andy D. Pimentel[a], Mojtaba Sabeghi[b]

[a]*Computer Systems Architecture Group, Informatics Institute,University of Amsterdam, Amsterdam, the Netherlands*
[b]*Computer Engineering Lab, Delft University of Technology, Delft, the Netherlands*

Abstract

System-level computer architecture simulations create large volumes of simulation data to explore alternative architectural solutions. Interpreting and drawing conclusions from this amount of simulation results can be extremely cumbersome. In other domains that also struggle with interpreting large volumes of data, such as scientific computing, data visualization is an invaluable tool. Such visualization is often domain specific and has not become widely studied and utilized for evaluating the results of computer architecture simulations.

In this paper, we introduce our novel interactive visualization tool, called VMODEX, which is developed to support system-level design space exploration of MPSoC architectures. In our tool, the design space is modeled as a tree in which both the design parameters and criteria are shown in a single view. VMODEX is able to handle large design spaces and allows designers to look at the data from different perspectives and at multiple levels of abstraction.

Due to the exponential design space in real problems and multiple criteria to be considered, heuristic searching algorithms 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. In VMODEX, besides the techniques provided for visualizing the DSE results, additional capabilities are developed to understand the dynamic search behavior of heuristic searching algorithms.

Keywords: multi-objective design space exploration, visualization, multi-objective evolutionary algorithms, MPSoC architectures

1. Introduction

Designers of modern embedded systems face several emerging challenges. Since embedded systems often target mass production and battery-based devices, they should be cheap and power efficient. In addition, they must, increasingly, support multiple applications and standards, for which they need to provide real-time performance. For example, mobile devices must support different standards for communication and coding of digital contents. Furthermore, modern embedded systems should also be flexible to enable easily extending them to support future applications and standards. Such flexible support for multiple applications calls for a high degree of programmability.

However, performance requirements as well as cost and power-consumption constraints require implementing substantial parts of these systems in dedicated hardware blocks. As a result, modern embedded systems often have a heterogeneous system architecture. They consist of components that range from fully programmable processor cores to fully dedicated hardware components for time-critical application tasks. Increasingly, such heterogeneous systems reside together on a single chip, yielding heterogeneous Multi-Processor System-on-Chip (MPSoC) architectures that exploit task-level parallelism in applications.

The heterogeneity of these highly programmable embedded systems and the varying demands of their target applications greatly complicate system design. 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. Therefore, optimal decisions need to be taken in the presence of trade-offs between design criteria. A set of optimal solutions is denoted as the Pareto optimal set. 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.

In order to find the Pareto optimal design points with respect to multiple design criteria, the designer should ideally evaluate and compare every single point in this space. However, such exhaustive search of all possible design points is usually prohibitive due to the sheer size of the design space. When the design space is too large to be explored in an exhaustive manner, heuristic search techniques such as evolutionary algorithms can be used to search the design space for optimum solutions using only finite number of design point evolutions.

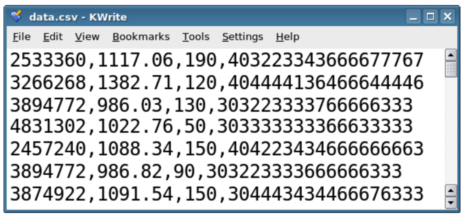
The DSE problem generally deals with two distinct issues:

1. The evaluation of a single design point regarding all objectives.
2. A search strategy for covering the design space during the exploration process.

Methods for evaluating design points range from solely analytical approaches to detailed cycle-accurate simulations. The results of simulation-based methods are more accurate but the evaluation process usually takes longer. A review of different approaches for evaluating a single design point is provided in [1]. In this paper, the evaluation of design points is performed by the Sesame system-level simulator [2, 3]. In Section 2.1 we provide an outline of the Sesame environment.

The searching strategies iteratively walk through the design space and try to properly cover the design space during their explorations. In general, these methods can be divided into two types, based on their progress from iteration to iteration: guided search and unguided search. The guided search methods, such as hill climbing, evolutionary algorithm, ant colony optimization and simulated annealing, use information learned so far to guide the search process. The unguided search methods such as random walk aim to provide an unbiased view of the design space. Each design point is chosen randomly and entirely by chance. In [1] a survey of different searching strategies used for design space exploration of system-on-chip architectures is given. In this paper, we use Multi-Objective Evolutionary Algorithms (MOEA) as the searching strategy. In Section 2.3 we explain the basic concepts of MOEAs.

System-level simulation frameworks that aim for early design space exploration create large volumes of simulation data in exploring alternative architectural solutions and investigating different mappings of application tasks onto architectural components. Interpreting and drawing conclusions from these copious simulation results can be extremely cumbersome. To illustrate the need for good analysis tools, Fig. 1 shows a sample of raw data generated by an MOEA. Here, each row represents an evaluated design point in which the values of objectives (processing time, energy consumption and cost) and the individual encoding 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 individual. 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
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 a MOEA. Each row represents an evaluated design point in which the values of objectives (processing time, energy consumption and cost) and the individual encoding are comma separated.

In other domains that also struggle with interpreting large volumes of data, such as scientific computing, data visualization has become an invaluable tool to facilitate the data analysis. Such visualization is often domain specific and has not become widely studied and utilized for evaluating the results of computer architecture simulations. Here, results are usually still presented in a table or displayed in a 2D or 3D graphs and very little research has been undertaken in the use of visualization to support and guide DSE.

Therefore, in this paper, we have developed a novel interactive visualization tool, VMODEX [1] , which is designed for understanding the multi-objective DSE process of embedded systems that are based on heterogeneous Multi-Processor System-on-Chip architectures. It is especially developed to help designers to get insight into the search process of heuristic methods that are typically used for rapid and efficient exploration of the large design spaces. This paper extends our previous works in [4, 5, 6] as follows:

- A more detailed and comprehensive description of the structure of the DSE tree is given.
- It provides extra visualization techniques for showing additional aspects of the characteristics of each design point. These aspects are referred as secondary objectives in the paper.
- More interactive exploratory techniques are added.
- A more extensive case study is presented to show the benefits of using our tool in the DSE process.
- This paper presents an overview of the entire visualization tool for the first time, while previous papers only focus on some specific aspects.

We have two main motivations for visualizing the design space exploration process. The first one is to gain insight into the landscape of the design space. That is, understanding the characteristics of the optimum design points, the correlations among multiple design criteria such as performance, cost, energy consumption and so on, and the relationships between the design parameters and their effects on the criteria. These relations are often not linear and a small change may lead to a completely different result. However, our visualization tool allows designers to easily understand these relations. Our second motivation is to understand how the design space is searched by heuristic searching algorithms. Our tool enables designers to understand the dynamic search behaviors of such algorithms. For instance, it clearly shows which parts of the design space are not searched at all (no design point is evaluated there), which parts of the design space are searched more often by the MOEA (more design points are evaluated there), which parts of the design space are explored in the later generations, and it illustrates the progress of the searching algorithm in the design space during successive iterations.

The rest of the paper is organized as follows. In section 2, we briefly explain the Sesame environment and some preliminary definitions of multi-objective optimization and multi-objective evolutionary algorithms. A short introduction on visualization and the principles of effective visualizations are also described. Section 3 describes related work. Section 4 introduces techniques we have provided for visualizing multi-objective design space exploration. In Section 5 we describe the interactive capabilities that are provided in our visualization tool and allow designers to directly manipulate the representations and to investigate the results from different perspectives . Section 6 presents a case study with a Motion-JPEG encoder application to illustrate the benefits of using visualization in the design space exploration process. The data from this case study are used in Section 7, which introduces several metrics and their visualization approaches for comparing different subspaces of the explored design space. Finally, section 8 concludes the paper.

2. Preliminaries

2.1. Sesame Environment

In this paper, the evaluation of design points (the first issue of DSE) is performed by the Sesame system-level simulator [2, 3].Models in Sesame are defined at a high level of abstraction and capture only the most important characteristics of the components in the system. As these high-level models minimize the

[1]Visualization of Multi-Objective Design spacE eXploration

modeling effort and are optimized for execution speed, they can be applied for DSE at the very early design stages. Sesame uses independent application and architecture models. The application model describes the functional behavior of the system in an architecture independent manner. That means the application model is free from architectural issues, such as timing characteristics, resource utilization or bandwidth constraints. Applications in Sesame are modeled using the Kahn process network (KPN) [7] model of computation in which parallel processes implemented in a high-level language communicate with each other via unbounded FIFO channels. Reading on a FIFO channel is blocking and writing is non-blocking. The architecture model represents the hardware components in the system. An architecture model is constructed from generic building blocks provided by a library, which contains template performance models for processors, co-processors, memories, buffers, busses, and so on.

Because of the separation of these two concerns (i.e. the application model and architecture model), an explicit mapping step is needed to relate these models for co-simulation. In this step, different mappings of processes and communication channels to various architectural components are evaluated by simulation to find the optimum mapping solutions under the design criteria. Each mapping decision taken in this step corresponds to a single point in the design space. Each mapping decision consists of two main parts: allocation and binding. The allocation determines which architectural components are actually used. The binding indicates each application task is executed by which allocated processor and each communication channel is mapped on which allocated processor/memory. Note that 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.

2.2. Multi-Objective Optimization

Real world design problems often are multi-objective optimization problems in which several incommensurable and often conflicting objectives should be optimized simultaneously. An improvement in one objective often causes deteriorations in other objectives. For example, consider the design of an embedded system such as a mobile phone, digital television, etc. Often the cost of such systems should be minimized, while maximum performance is desirable. Here, high performance and low cost are generally conflicting. High-performance architectures can be quite expensive, while cheap architectures usually provide lower performance. 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.

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

$$\begin{array}{ll} Minimize & y = f(x) = (f_1(x), \cdots, f_m(x)) \\ Where & x = (x_1, \cdots, x_n) \in X \\ & y = (y_1, \cdots, y_m) \in Y \end{array} \tag{1}$$

Without loss of generality, we assume a minimization problem. The duality principle [8], in the context of optimization, can be used to convert a maximization problem into a minimization problem by multiplying the objective functions by -1. By using duality principle, handling mixed maximization/minimization problems is also easy. When an objective is required to be maximized, the duality principle can be used to transform the original objective for maximization into an objective for minimization. The objective function f(x) maps a decision vector (solution) x in decision space (X) to an objective vector y in objective space (Y).

Definition 2: (Dominance Relation) 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: $\forall i \in \{1, \cdots, m\} : f_i(x_1) \leq f_i(x_2)$
2. The solution x_1 is strictly better than x_2 in at least one objective;
   formally: $\exists j \in \{1, \cdots, m\} : f_j(x_1) < f_j(x_2)$

It is intuitive that if x_1 dominates the solution x_2, the solution x_1 is superior to 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". A more rigorous definition of *strict dominance* requires x_1 to be strictly better in all objectives compared to x_2 and is written as $x_1 \ll x_2$, while the less strong definition of *weak dominance* only needs the trueness of the first condition and is denoted by $x_1 \leq x_2$. Two solutions x_1 and x_2 are said to be non-dominated with respect to each other ($x_1 \sim x_2$) if in some objectives x_1 is better than x_2 and in some other objectives x_2 is better than x_1, on the other hand none of them is dominated by another one; formally:

$$\exists\, i \neq j \in \{1, \cdots, m\} : \textit{such that} f_i(x_1) < f_i(x_2) \wedge f_j(x_2) < f_j(x_1)$$
$$\Rightarrow x_1 \nless x_2 \wedge x_2 \nless x_1 \Rightarrow x_1 \sim x_2$$

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

- The solution x_1 is said to be non-dominated regarding a set $X' \subset X$ 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.

Definition 4: (Pareto Optimal Set and Front) The entirety of all Pareto optimal solutions is called the Pareto optimal set. Solutions in this set 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 the "Pareto optimal front" or "Pareto front".

To illustrate the dominance relation and Pareto optimality, an example is given in Fig. 2. The two-dimensional objective space is defined by execution time and cost of evaluated design points, both to be minimized. The solution C dominates solution E ($C < E$) since it is better in both objectives: it has a lower execution time and a lower cost. However, when comparing B and C, neither can be said to be superior, since $B \nless C$ and $D \nless C$. Although solution C is cheaper, it has a higher execution time than solution B. In Fig. 2, the black points construct the Pareto optimal front that contains four solutions: A, B, C and D.

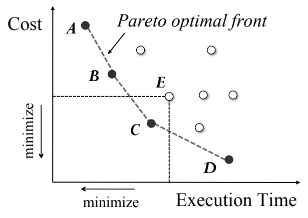


Figure 2: Illustration of the Pareto optimality and dominance relations between solutions in a two-dimension objective space. Both objectives need to be minimized.

Definition 5: (Euclidian Distance) The Euclidian distance between two solutions a and b (of dimension n) is defined as:

$$\|a - b\| = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

If the objective functions have different scales of measurement, they should be normalized before calculating the distance measure. This prevents objectives with initially large ranges to overcome objectives with initially small ranges. In this paper, we use min-max normalization to obtain a "common scale" for all objectives.

Definition 6: (Min-Max Normalization) The min-max normalization performs a linear transformation on the original data values so that it does not change the initial distribution type. It transforms the data values into a desired range, which is usually [0,1]. Suppose that f_i^{min} and f_i^{max} are the minimum and maximum value of the ith objective (among the evaluated design points). We would like to map interval $\left[f_i^{min}, f_i^{max}\right]$ into a target interval [0,1]. Consequently, for each solution $x \in X$ the value of ith objective $(f_i(x))$ from the original interval will be mapped into its normalized value (written as $\bar{f}_i(x)$) using the following formula:

$$\bar{f}_i(x) = \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}}$$

After applying min-max normalization, each objective value will fit in the range [0,1]. However, the underlying distribution of the corresponding objective within the new range will remain the same. Min-max normalization preserves exactly all the relationships among the original objective values and it does not introduce any bias.

2.3. Multi-Objective Evolutionary Algorithms

In this paper, we use Multi-Objective Evolutionary Algorithms (MOEAs) as a strategy for searching the design space (the second issue of DSE). In the last few years, MOEAs have been recognized to be well suited to solve multi-objective optimization problems and have become very popular. They are widely used in many different fields such as computer science, engineering, economics, finance, industry, chemistry, ecology and etc. Over the years, many MOEAs have been proposed such as SPEA2 [9], NSGA-II [10], PAES [11], etc. The main motivation for using MOEAs to solve multi-objective optimization problems is because MOEAs operate on a set of design points (solutions) rather than only one solution at each time. Thus, in a single run of the algorithm, several solutions are evaluated and in this way the search is performed in a parallel manner. Furthermore, MOEAs are less sensitive to the shape or continuity of the Pareto front. For example, they can easily deal with discontinuous, multi-modality and concave Pareto fronts. Additionally, MOEAs are able to search large and complex decision spaces. Moreover, it is not necessary to choose an accurate starting point. The results of MOEAs do not rely on the initial population and they aim at finding the global Pareto optimal solutions, rather than to get stuck in local optimums. These features play a critical role in the efficiency of MOEAs in solving multi-objective optimization problems. These issues are, on the other hand, major concerns when using traditional mathematical programming techniques for finding optima, such as linear programming, nonlinear programming, stochastic programming, robust optimization, etc.

Evolutionary algorithms are heuristic search methods that take their inspiration from natural biological evolution. By analogy to natural evolutions, solution candidates are called *individuals* and a set of solutions is called a *population*. Each *individual* represents a possible design point in the decision space. However, an *individual* is not a vector in the decision space. Instead, it is an encoded representation of a decision vector.

MOEAs evaluate a *population* of *individuals* over several iterations, called *generations*. With the help of genetic operators, a MOEA progresses iteratively towards the best possible solutions. Its progression (optimization process) is comparable to natural selection in the evolution of living organisms in that the most proper elements in the population have a higher chance to survive, reproduce and thus transfer their genetic material to the consecutive generations. The algorithm usually starts with a randomly generated *population* and calculates the *fitness* value for each *individual* within the *population*. The *fitness* function is problem specific and determines how good the individual is. Note that in Equation 1, $f(x)$ is equivalent to the *fitness* function and y is equivalent to the *fitness* value. *Individuals* with high *fitness* in the current *population* are selected for reproduction and referred to as *parent individuals*. Such *individuals* are modified (recombined and randomly mutated) to create new *individuals* (offspring). This is motivated by the hope that the new *individuals* will be better than the old ones. The new *individuals* are then considered as the current *population* in the next iteration of the algorithm. Generally, the algorithm terminates when a predefined number of *generations* has been reached. But, also other conditions may be used as the termination criterion. For instance, when there is no improvement in the *fitness* values of the *individuals*, or some *individuals* with sufficient quality have been found. At the end, the best *individuals* found during the search process are considered as the outcome of the MOEA.

By utilizing an MOEA for multi-objective design space exploration of embedded systems, on one hand, we should define the specification of the DSE problem (i.e. application model, architecture model and design criteria) and on the other hand we should set the MOEA parameters such as selection method, mutation rate, crossover rate, etc. Setting the MOEA parameters properly, according to the DSE specification, can minimize the exploration time while achieving better results. Then, it is needed to define a way of encoding the DSE problem (e.g. mapping process networks onto heterogeneous multiprocessor architecture) as a multi-objective optimization problem, known as *individual encoding*. Therefore, each *individual* in MOEA (which can be a string of numbers) represents a design point in the design space (e.g. possible mapping).

2.4. Visualization

Collecting information is no longer a problem, but interpreting and extracting insight from the collected information has become progressively more difficult. Visualization is the process of transforming data, information, and knowledge into visual form making use of humans' natural visual capabilities [12]. Visualization is useful to rapidly extract insights from large amounts of information. It allows users to quickly understand the general structure and patterns in the data and the relationships between different variables.

Information Visualization (InfoVis) deals with abstract data sets, that is, data without a "natural" physical or geometric representation, such as hierarchical or textual information. Therefore, the user has no predetermined mental model to which the data can be automatically mapped. Thus, the main challenge in information visualization is developing new visual metaphors for representing the abstract data, such that they enable users to clearly and effectively understand the underlying information.

In [13], Tufte discussed the properties of an effective graphical display. He concluded with the following *Principles of Graphical Excellence*:

- Graphical excellence consists of complex ideas communicated with clarity, precision and efficiency.
- Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
- Graphical excellence is nearly always multivariate.
- Graphical excellence requires telling the truth.

3. Related Work on Visualizing Exploration Results

Most of the work that has been performed on visualization of computer architecture simulations either focuses on educational purposes (e.g., [14, 15, 16]), tightly couples visualization to one particular architecture simulation environment (e.g., [17, 18, 19, 20]), visualizes only one specific aspect of embedded applications such as memory access patterns [21], cache behavior [22, 23, 24] and data dependencies [25], or only provides some basic support for the visualization of simulation results in the form of 2D (and sometimes 3D) graphs (e.g., [26]). However, in the area of system-level design space exploration, little research has been undertaken on visualization of simulation results and the exploration process.

The work presented in [27, 28] provides advanced and generic visualization support, and tries to do so for a wide range of computer system related information. However, these visualizations are not necessarily applicable to computer architecture simulations and in particular to DSE, with its own domain-specific requirements. Vista [29] aims at visualization support for computer architecture simulations, but it does not target system-level simulations, which may have a serious impact on the scalability requirements of the visualization, nor does it address the needs for visualization from the perspective of DSE.

In [30], 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 designers to understand the information through different perspectives. But this tool does not provide any insight in the searching process as performed by a heuristic method (e.g. MOEA). For example, there is no way to find out which parts of the design space are not searched at all. This tool is only useful in the case of small design spaces, which can be exhaustively searched.

There are only a few research efforts addressing the visualization of EAs. The most common method for analyzing how EA evolves is monitoring individual fitness values, and creating fitness-versus-generations graphs. Although such straightforward graphs show the quality of the solutions considered during search process, they can only provide a limited amount of information. Due to the large number of individuals in a population, it is not possible to display all of their fitness values. Therefore, displays are usually restricted to the best individual fitness and average population fitness for each generation. Genetic information regarding the position of the discovered solutions in the search space cannot be obtained from these graphs.

More complex techniques have focused on how to display the progress of the MOEA in parameter space or objective space [31, 32]. 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 parameters and objectives spaces. Furthermore, due to the large number of dimensions in practical problems, techniques such as Sammon Mapping [33] should be used to transform higher dimensional spaces into smaller ones.

Hart and Ross [34] proposed the GAVEL visualization tool that provides a means to examine how the genetic operations (crossover and mutation) assemble the optimal solutions, and a way to trace the ancestry of individuals. GAVEL only shows the information that is relevant to the formation of the best solutions. All individuals that do not play any part in propagating individual genes to the final optimal solutions are disregarded. So, it does not provide any information on how the problem space is explored and how an entire population behaves or changes across generations.

In [35], a visualization method is proposed, which shows the individual structure, the fitness function and the objective function in a matrix view. For each generation, an $n \times m$ matrix is used where n is the number of individuals in the generation and m is the number of bits in an individual. Each cell of the matrix is related to a bit of the individual. If the value of a bit is one the corresponding cell is represented in red. The blue cell indicates that the value is zero. The brightness of each row (individual) represents the fitness value. The hue of individuals is changed by their objective values. This visualization approach has some drawbacks. It can be used for showing only binary-coded evolutionary algorithms. Furthermore, the structure of solutions in the parameter space is represented in their encoding format. So, understanding the correlation between the problem space and individual space is difficult. It is not easy to find out how EA explores the parameter space during its generations.

There are also some commercial visualization tools such as modeFRONTIER [36], OPTIY [37], and GRAPHEUR [38] that can be used for data analysis. However, these tools are generic visualization tools and they could not meet all our domain specific requirements. Therefore, we have developed our specific tool. For instance, the visualization approach that we propose for mapping decision (Section 4.2.2) clearly and effectively shows how application tasks and communication channels are mapped to the architectural components. However, using commercial visualization tools, it is not possible to visualize this kind of domain specific information. They mainly integrate various standard graphical representations and there is no freedom to create your own methods that may be more suitable for your specific domain.

Furthermore, in the commercial tools, for showing multi-dimensional data, different standard techniques are provided to project a multi-dimensional space to 2D or 3D space, such as scatter plot matrix, parallel coordinates, etc. The main weakness of this kind of projection is that the plots are sequences of pairwise comparisons between dimensions. As such, it becomes difficult to comprehensively understand the relations between all dimensions at once in the same area of the plot. However, in this paper, we propose a new visualization method (based on tree visualization) in which multiple dimensions are shown just in a single view without requirement of projection to the smaller dimensions.

Moreover, additional functionalities are provided in our tool that help users to gain insight into the searching process as performed by a heuristic method. In our tool we have developed several techniques that enable users to understand how a searching algorithm walks through and covers the design space during its successive iterations. However, this kind of analysis is not provided in commercial tools.

Table 1: Table of symbols for the tree model of the design space

Symbol	Definition	Symbol	Definition
T_{DS}	Tree model of the design space	DP	The set of all evaluated design points
Seg_{par}	Parameters segment	GP	The set of global Pareto optimal points
$Seg_{arch-dep-obj}$	architecture-dependent objectives segment	LP	The set of local Pareto optimal points
Seg_{dp}	Design points segment	NP	The set of non-Pareto optimal points
$\|Seg\|$	Number of levels in a segment	Sub	Abbreviation for subspace
LVL	Abbreviation for level	k	Total number of subspaces in the T_{DS}
par	Abbreviation for parameter	$map\,(p)$	The mapping of the application onto the architecture components for the design point p
n	Total number of parameters		

4. Multi-Objective DSE Visualization

In all visualizations, graphical elements are used as a visual syntax to represent semantic meaning. This mapping of information to visual elements is called visual encoding. A simple example is the mapping of temperature to a color scale from blue to red. Cool and warm colors may be used to represent low and high temperatures, respectively. The visual variables are considered as the fundamental building blocks of any visualization. In our proposed visualizations, several visual variables are used, such as size, line width, line style, color, shape, Saturation, blinking, etc.

Information visualization systems have two main components [39]: representation and interaction. The representation component involves the way that data is mapped to the visual form. Subsequently, the interaction component allows a user to directly manipulate the representation for more effective data exploration and help users to deal with large volumes of information. In this section, we focus on the first aspect of visualization systems and explain how we visualize the design space exploration process. In the next section, we describe several interactions provided in VMODEX, such as filtering, zooming, hiding, details on demand, etc., that enable designers to explore the data from various perspectives and discover additional insights.

4.1. Modeling the Design Space as a Tree

As it is conceptually shown in Fig. 3(a), we model the design space as a tree. In this section, we explain how a design space can be modeled as a tree.Table 1 presents the set of mathematical symbols that are used for constructing the tree model of the design space. The tree has three segments: the parameters, architecture-dependent objectives and design points. Equation 2 formally describes tree segments.

$$T_{DS} = Seg_{par} + Seg_{arch-dep-obj} + Seg_{dp} \tag{2}$$

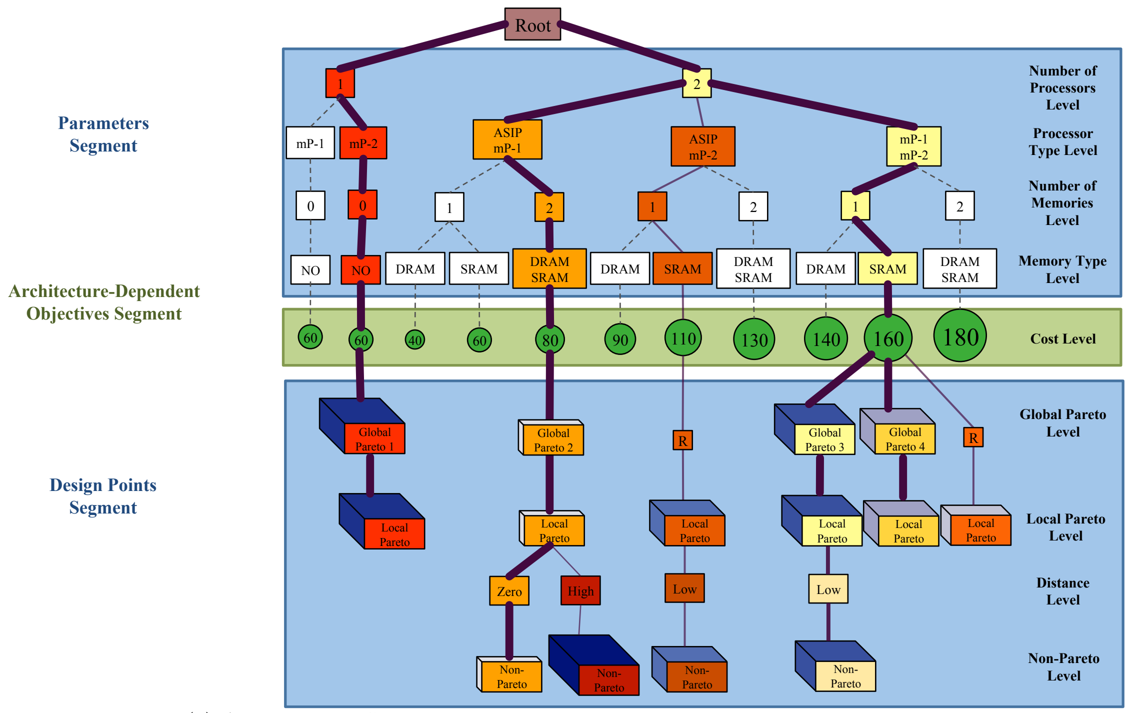
The parameters and design points segments are explained in this section while the architecture-dependent objectives segment will be described in the next section.

4.1.1. Parameters Segment

In this segment, 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 segment is equal to the total number of parameters in the design space; formally:

$$\begin{aligned} &Seg_{par} = LVL_{par_1} + LVL_{par_2} + ... + LVL_{par_n} \\ &|Seg_{par}| = n \end{aligned} \tag{3}$$

For better understanding the process of modeling the design space as a tree, an example is shown in Fig. 3(a). In the tree illustrated in this figure, the design space has four parameters thus there are four levels in the parameters segment, which are:



(a) An example DSE tree. The example shows 11 architecture instances and 10 unique evaluated design points, 4 of which are globally Pareto optimal. Note that all the global Pareto optimal points are local Pareto points as well. So, they will be shown in both the global and local Pareto levels.

Min
Processing Time
Max
Min
Energy Consumption
Max

(b) Color legends

Figure 3: Modeling the design space as a tree. The tree has three segments: the parameters, architecture-dependent objectives and design points.

1. *Number of processors level:* this level shows the number of processors (independent of their type), which are allocated for executing the application tasks. In this particular example, the desired maximum number of processors is two and therefore there are two nodes at this level indicating one or two used processors.
2. *Processors type level:* nodes at this level represent different combinations of processor types. Each node becomes a child of the node at the previous level that shows the number of processors used in its combination. In this example, the platform architecture consists of one Application Specific Instruction Processor (ASIP) and two microprocessors (mPs). Note that only those combinations are shown, which lead to the platform instances that are capable of executing the application (feasible combinations). For instance, in this example, using only one ASIP is not sufficient for executing the application and thus there is no node in the tree that indicates one ASIP.
3. *Number of memories level:* this level shows the number of memories used in a platform instance.
4. *Memory types level:* at this level, the type of memories is shown. In this example, there are two types of memory: one Static RAM (SRAM) and one Dynamic RAM (DRAM).

By modeling the design space as a tree, there is no limitation on the number of design variables as each parameter is located at one level of the tree. Therefore, we can 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. On the other hand, by putting more important parameters down in the tree, the design points with the same parameter are distributed in several sub trees.

4.1.2. Design Points Segment

this segment 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. This means that, for each solution, its parents at the previous levels show its design parameters. For instance, the solution labeled by "Global Pareto 2" has the following architectural components: two processors, of which one is an ASIP and the other one is mP, and two memories, of which one type is DRAM and another type is SRAM. Design points are distributed in three levels: global Pareto, local Pareto and non-Pareto. There is also another level in this segment, called distance level, which is used for categorizing the non-Pareto points based on their distance from parents. Thus, the design points segment consists of four levels, as follows:

$$\begin{aligned} &Seg_{dp} = LVL_{gp} + LVL_{lp} + LVL_{dis} + LVL_{np} \\ &|Seg_{dp}| = 4 \end{aligned} \tag{4}$$

Global Pareto level. This 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; formally:

$$\begin{aligned} &LVL_{gp} = \{p \mid p \in GP\} \\ &GP = \{p \in DP \mid \nexists p' \in DP : p' < p\} \end{aligned} \tag{5}$$

The solutions in the global Pareto optimal set has the following two properties:

1. They are non-dominated with respect to each other (no one is absolutely better than another one); formally: $\forall\, p, p' \in GP : p \sim p'$
2. Each point that is not part of the global Pareto set, is dominated by at least one global Pareto point; formally: $\forall\, p \in (DP - GP)\, \exists\, p' \in GP : p' < p$

In this paper, we simply refer to the global Pareto optimal solutions as Pareto optimal solutions.

Local Pareto level. By modeling the design space as a tree, it is divided into exclusive subspaces. Each subspace represents a unique combination of design parameters (in our case, a unique instance of the architecture platform). On the other hand, solutions inside a subspace have exactly the same architecture components (have the same parents at the parameter levels) but the way that the application is mapped onto those components is different. Fig. 4 illustrates the concept of subspaces. It shows the same tree as Fig. 3(a). However, in this figure, the four subspaces of the design space that contain some evaluated design points are indicated by contour lines and are labeled from Sub_1 to Sub_4. For instance, all design points in the subspace Sub_4 have two microprocessors and one SRAM memory in their underlying architecture. However, the binding of application tasks and channels to these resources is different. In total, there are 11 subspaces in the example shown in Fig. 4. Actually, The number of nodes at the last level of the parameters segment (memory type level in Fig. 4) indicates the number of subspaces. Thus, we can consider the tree model of the design space as a union of several subspaces:

$$T_{DS} = \bigcup_{i=1}^{k} Sub_i \tag{6}$$

For each $p, p' \in Sub_i$ following conditions are true:

1. $\forall\, j \in \{1, 2, .., n\} \;\; par_j(p) = par_j(p')$
2. $map(p) \neq map(p')$

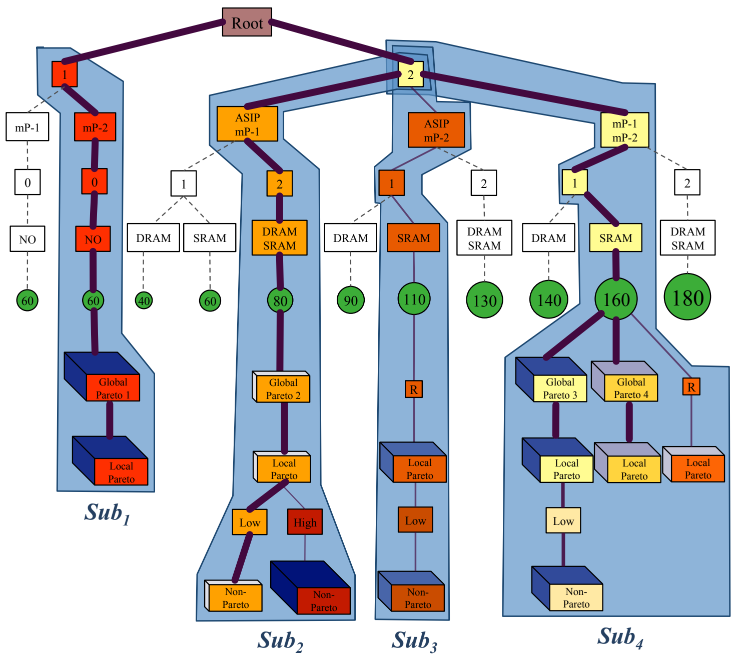


Figure 4: Illustration of the subspaces of the design space. The four subspaces that contain some evaluated design points are indicated by contour lines and are labeled from Sub_1 to Sub_4.

At the local Pareto level, the local Pareto points are shown. In each subspace, the Pareto optimal solutions are called local Pareto. Thus, a design point is called a local Pareto point if within the design points with the same architecture components (same allocation) but with different mappings (different binding), there is no point dominating that one. So, it is an optimal solution with respect to a specific architectural instance. However, in the entire design space, a design point might exist which dominates the local Pareto point. Equation 7 gives a formal definition of the local Pareto concepts:

$$\begin{aligned} LVL_{lp} &= \{p \mid p \in LP\} \\ LP &= \bigcup_{i=1}^{k} LP_i \\ LP_i &= \{p \in Sub_i \mid \nexists\, p' \in Sub_i : p' < p\} \end{aligned} \tag{7}$$

It is clear that all the global Pareto points are local Pareto points as well. However, not all the local Pareto points are global Pareto points and therefore we use a relation node at the global Pareto level to make a connection between them and the previous level. These nodes are labeled with "R" in Fig. 3(a). Equation 8 describes this relation between the global Pareto and local Pareto sets:

$$p \in GP \Rightarrow p \in LP \quad , \quad p \in LP \nRightarrow p \in GP \tag{8}$$

Non-Pareto level. All the other design points are placed at the non-Pareto level. Each non-Pareto point is dominated at least by one point in the local Pareto set of its corresponding subspace (LP_i); formally:

$$\begin{aligned} LVL_{lp} &= \{p \mid p \in NP\} \\ NP &= \{p \in DP \mid \exists\, p' \in LP_i : p' < p\} \end{aligned} \tag{9}$$

Each non-Pareto point 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 (see Section 2.2), 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.

Distance level. For easier interpretation and better analysis of the design points, the children of a local Pareto point are categorized into two groups according to their Euclidian distance from their parent. If the distance between a non-Pareto point 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. Thus, there are two types of nodes at the distance level; formally:

$$LVL_{dis} = \{d \mid d \in \{Low, High\}\} \tag{10}$$

4.2. Showing Objectives

Various features can be considered as objectives for design space exploration of embedded systems, such as processing time to complete a particular task or application, power consumption, energy consumption, architecture cost, latency, utilization, temperature, physical size, weight and so on. We classify objectives, which are considered in the DSE process, in two groups: primary objectives and secondary objectives. The primary objectives are directly used in optimization process and depending on their values the optimization algorithm walks through the design spaces to find optimum design points with respect to the primary objectives. The secondary objectives are not considered as optimization goals. However, they provide supportive information and help designer to do further analysis on the characteristics of design points.

The primary objectives are directly shown in the DSE tree and therefore just by looking at the DSE tree the designer can see their values. However, the secondary objectives only are shown when the designer selects to see them. If the designer is interested to know about the secondary objectives for some specific solutions, it is possible to click on those solutions to see their secondary objective values.

4.2.1. Showing Primary Objectives in the DSE Tree

For representing the primary objectives in the DSE tree, we have divided these objectives into two groups:

1. Objectives that are only dependent on the architectural components, denoted as $obj_{arch-dep}$
2. Objectives that are dependent on the mapping, denoted as $obj_{map-dep}$

Those primary objectives that are only dependent on architectural components (not mapping) are shown in the architecture-dependent objectives segment. An example is the architecture cost as shown in Fig. 3(a). These objectives can be computed after the parameters segment, since all components are known. Furthermore, all design points with the same architecture have the same value in terms of these objectives. Therefore, we add an extra segment between the parameters and design points segments, which shows the values of architecture-dependent objectives for different architectures. The definition of the architecture-dependent objectives segment is as follows:

$$\begin{aligned} &Seg_{arch-dep-obj} = LVL_{obj_{arch-dep_1}} + LVL_{obj_{arch-dep_2}} + ... + LVL_{obj_{arch-dep_{m_1}}} \\ &|Seg_{arch-dep-obj}| = m_1 \end{aligned} \tag{11}$$

Where m_1 is the number of architecture-dependent objectives. In Fig. 3(a), the cost is shown with a different shape; a circle, since it is an objective and not a design parameter. For a better view, the size of the circle becomes bigger as the cost increases. In the case that more primary objectives exist, which are only dependent on architectural components, such as weight, physical size, etc., extra levels can be added in architecture-dependent objectives segment in which each level shows one objective. For better distinguishing different objectives, each one can be shown with a different color and shape, like rectangle, trapezoid, pentagon, etc. In each subspace, the values of architecture-dependent objectives for all design points are the same. For instance, in Fig. 4, the cost of all design points in Sub_4 is 160. Formally, For each $p, p' \in Sub_i$ the following condition is true:

$$\forall j \in \{1, 2, .., m_1\} \quad obj_{arch-dep_j}(p) = obj_{arch-dep_j}(p')$$

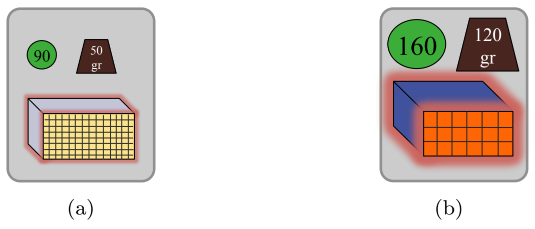


Figure 5: An example of representing solutions with 6 objectives. The following visual variables are used for showing the objectives: node color (processing time), third dimension color and size (energy consumption), size of the circle (architecture cost), texture density (average utilization), size of the glow (temperature) and size of the trapezoid (weight).

Those objectives that are dependent on the mapping (binding application tasks and channels to the hardware components) are shown in a design point node. In Fig. 3(a), there are two primary objectives that are dependent on the mapping: the processing time (i.e. the time needed to execute the given application) and energy consumption of the whole system. 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 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 legends for processing time and energy consumption are shown in Fig. 3(b).

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(a), the minimum processing time is chosen for coloring parameter nodes.

It should be mentioned that although we only show three objectives (architecture cost, processing time and energy consumption) in this paper, VMODEX is able to easily visualize more than three objectives. For those objectives that are only dependent on the architectural components, extra levels can be added in the architecture-dependent objectives segment. Showing more objectives that are dependent on the mapping is also easy. Each node has some attributes like shape, orientation, size, color, transparency, texture, border, glow, etc. Each attribute can be assigned to one objective. For the purpose of illustration, Fig. 5 shows an example of representing a solution with six objectives. In this figure, texture density indicates average utilization (denser texture means higher utilization), the size of the glow shows the temperature (bigger glow means higher temperature) and the size of the trapezoid indicates the weight (bigger trapezoid means heavier weight). The other three objectives are the same as Fig.3(a). Note that the examples shown in Fig. 5 would become the tree nodes in our DSE tree (For the case that we have 6 objectives). Since in the DSE tree the architecture-dependent objectives (e.g. cost and weight) are shown as separate nodes in the segment $Seg_{arch-dep-obj}$, the corresponding nodes are drawn above the 3D rectangle. Fig. 5(a) represents a solution that is superior to the solution shown in Fig. 5(b) with respect to the all six objectives. Thus, the solution in Fig. 5(a) has a lower processing time and energy consumption, and its average utilization is higher. It also produces less heating and it is cheaper and lighter.

As we described above, our visualization tool is extendable to show more than three objectives. However, multi-objective design space exploration problems are usually based on only two or three objectives, typically performance, cost and power. The maximum number of objectives that we found in related work is six [40].

4.2.2. Showing the Mapping Decision

In the Sesame simulator as well as in many other system-level simulation frame-works, 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

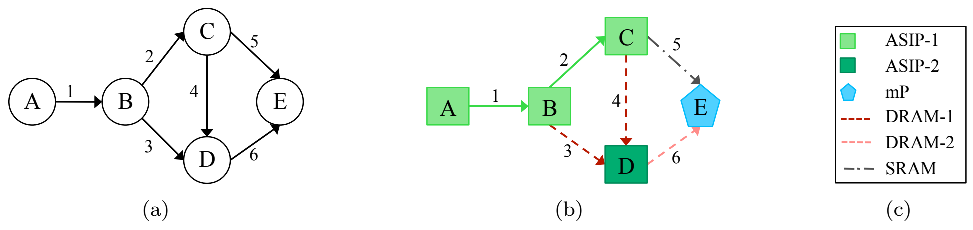


Figure 6: (a) An example of process network with five processes (A-E) and six channels (1-6), (b) Mapping decision visualization. The shape and the color of each node represent the type of the processor executing the corresponding process. The color and the style of each line represents the type of the communication, (c) Mapping legend.

(FIFO) communication channel between two processes. Fig. 6(a) represents an example process network graph, which has five processes (A-E) and six communication channels (1-6).

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. The color of the line and style of the dash represents the memory type. Memories with the same type are shown by the same dash style but with different variants of the same color.

Fig. 6(b) shows how our visualization model shows the process network graph from Fig. 6(a). As can be seen in this figure, processes A, B, C and channels 1 and 2 are mapped to the same processor (ASIP-1). Process D is executed on the same processor type but on a different processor as process A (ASIP-2). The type of the processor executing process E is different from the others since it is shown with a blue pentagon (mP). 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 dash style and different color (SRAM).

Note that the colors assigned to each processor type or memory type in the visualization of the mapping decision are the same as colors used in the visualization of utilization, latency and processes execution time (these visualizations will be explained later). For instance, in all of them the ASIP processor is shown by a green color. Thus, these visualizations are consistent with each other and the designer can easily understand the effect of different mapping decisions on the utilization of architecture components as well as processes execution times and read/write latencies.

4.2.3. Showing Secondary Objectives

Since the secondary objectives are not used in the optimization process and do not have any effect on the exploring procedure, we do not show them directly in the DSE tree. However, if the designer is interested to know about the secondary objectives for some particular solutions, it is possible to click on those solutions to see their secondary objective values. The secondary objectives provide additional information and help the designer to do more in-depth analysis on the characteristics of design points. VMODEX is designed to show various properties of design points, which are considered as secondary objectives. Thus, the designer is able to look at the evaluated data from different perspectives and thereby gets a comprehensive view on the problem under study. Furthermore, for better understanding and easier analysis, we propose several

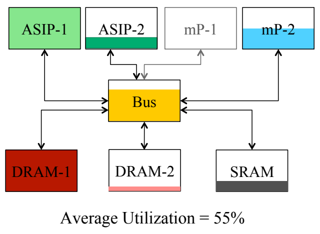


Figure 7: Utilization visualization. For each component, the size of the colored part represents the percentage of the time that it was busy.

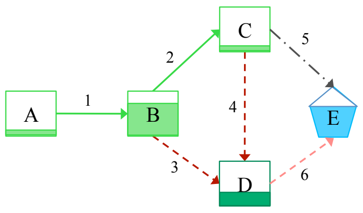


Figure 8: Visualization of processes execution time. For each process node, the size of the colored part represents the percentage of time that its assigned processor was executing it.

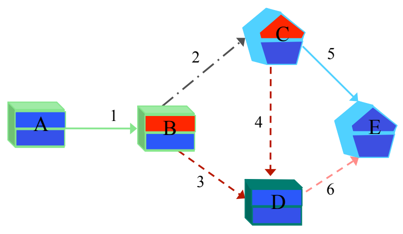
visualization techniques for showing the secondary objectives. In the following subsections the provided secondary objectives and their visualization methods are described.

Showing Utilization. For better understanding the utilization of each separate hardware component and also for easier comparison between different design points, we have visualized the utilization property. In this paper, utilization is defined as the percentage of the time that a hardware component was busy. For visualizing the utilization, the platform architecture is shown as a directed graph in which each node represents an architectural component and the edges show connectivity between components. Each node (component) is filled with its corresponding color (see subsection 4.2.2) in a way that the size of the colored part represents the percentage of the time that the corresponding component was busy. The components in the platform architecture that are not allocated are shown by gray borders. The average utilization is written at the bottom of the visualization.

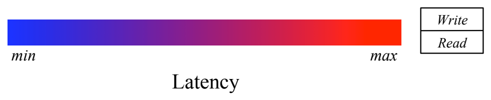
Fig. 7 shows an artificial example of utilization visualization for a platform architecture consisting of two Application Specific Instruction Processors (ASIPs), two microprocessors (mPs), one Static RAM (SRAM) and two Dynamic RAMs (DRAMs). In this example, the mP-1 is not allocated since it is shown by gray border. The utilization of ASIP-1 and DRAM is 100% while for the Bus it is almost 75% and for the other components it is less than 50%. By visualizing the utilization, besides understanding the utilization of each architecture component individually, the designer is able to see how these components are connected together, operate with each other and which resources are shared.

Showing Processes Execution Time. VMODEX also allows to modify the mapping visualization in such a way that it shows the processes execution time as well. Thus, it also shows the amount of time that each process was executed by the processor onto which it has been mapped. In the mapping visualization, the shape and color of each process in the process network graph represent the type of the processor executing the corresponding process. To add information about how long a processor was busy with executing a process, instead of fully coloring the process nodes in the graph, the size of the colored part represents the percentage of time that its assigned processor was busy executing it. Fig.8 shows the same mapping decision as shown in Fig.6(b), but the processes execution time information is added. From this figure we can see that processor ASIC-1 was executing process B in most of its busy time, while executing process A and C took relatively a small amount of time. The processes execution time visualization enables designers to easily explore the effect of different mappings on the execution time of processes.

Showing read/ write latency. Similarly, VMODEX is also capable to show the amount of time that each process is waiting for read and write communications. Since in process network graph the communication channels between processes are shown, we visualize it in a way that shows the read and write latencies as well. The color-coding, from blue to red, is utilized to represent the waiting time. In Fig.9(b) the color legend for latency is shown. In the process network graph, each process node is divided into two halves. The color of the top part shows the write latency and the color of the bottom part represents the read latency. By clicking on each part, the exact value of latency is shown. The mapping information is also added to the latency visualization as well. Therefore, the designer can easily investigate the effect of different mappings



(a)



(b)

Figure 9: Latency Visualization. The color of the top part shows the write latency and the color of the bottom part shows the read latency. The shape and color of the third dimension of each process node represent the type of the processor executing it.

on the latency. The shape and color of the third dimension of each process in the graph represent the type of the processor executing the corresponding process (similar to the mapping visualization). Fig. 9(a) shows an example of latency visualization. In this figure, processes B and C have high write latencies since the color of their top parts are red. By including mapping information in the latency visualization we can understand that the reason of these high latencies is because of mapping channels 3 and 4 on the same memory (both of them are shown by dashed red lines). Thus, processes B and C are competing with each other to write to the shared memory. For the other processes both the read and write latencies are relatively low.

Note that our proposed latency visualization can be used to show both relative and absolute latencies. Actually, from the visualization side, it gets some values and maps them to the color, based on the color-coding scheme. So, it is dependent on the simulation model that which kind of latency (absolute or relative) is evaluated. In our case, the absolute latency is used.

Showing Processing Time of Processor Operations. Basically, processors perform three operations: read, write and execution [2]. There is an option in VMODEX to see the percentage of time that a processor was doing each operation separately. To show this, for each processor in the architecture platform, a stacked bar chart is drawn. Each operation is shown as a stack in the bar with a different color. The stack height represents the percentage of time that the processor was performing the corresponding operation. Fig. 10 represents an example of visualizing the processing time of different processor operations for a platform architecture consisting of two ASIPs and one mP.

Showing Generation Numbers. In some cases, the designer wants to know what interesting design points are evaluated in which search generations. Therefore, we have developed a method in VMODEX that allows the designer to easily find out this kind of information. During the process of design space exploration using an MOEA, some design points that are near to the optimal solutions may be regenerated in different generations. There is an option in the VMODEX user interface to show the generation numbers. By selecting

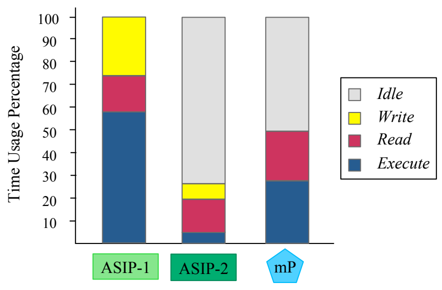


Figure 10: Visualization of processing time of processor operations. The stack height represents the percentage of time that the processor was performing the corresponding operation.

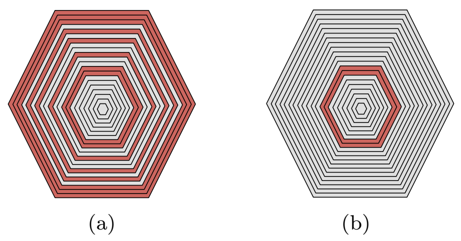


Figure 11: Visualization of the generation numbers. For each generation, a hexagon is drawn. The red hexagons indicate those generations in which the desired design point was evaluated.

that option, for each design point node in the DSE tree, a hexagon will be drawn at the upper left corner of the node. The number of the last/first (chosen by the designer) generation in which the corresponding design point was generated is written inside the hexagon.

Moreover, the designer is also able to select a specific design point and see all the generation numbers in which the design point was evaluated. Fig. 11 shows the visualization approach for showing the generation numbers. For each generation, a hexagon is drawn. The size of the hexagons increases from the first to the last generations. To save space, these pentagons are nested together. If within a generation the selected design point is found, then the color of the pentagon representing that generation is red. Otherwise, a gray pentagon is drawn. Fig. 11(a) shows the situation that the corresponding design point is close to the optimum. Since it is regenerated in many search generations during the entire search. But in Fig. 11(b) the corresponding design point is generated in only two generations. This indicates that the design point is far from the optimal solutions and after a few generations it is not regenerated any more.

Showing Exact/Normalized objective values. Instead of showing objectives with visual variables (color and size), VMODEX can also show 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 values in the range [0, 1] for their objective values.

4.3. Edge Visualization

Edge visualization helps designers to navigate through the DSE tree and easily find more important parts. One feature of the design points is chosen as an importance factor and then the tree edges are visualized according to that factor, as follows:

- A minimum and maximum edge width is defined, and this range is linearly mapped against the range of importance factor values.
- A specific color with various saturations is chosen. Similar to the line width, a linear mapping is done between the maximum and minimum saturation and the importance factor values range. Darker edges represent more important parts and lighter edges show less important subtrees.

In VMODEX, two importance factors are defined: minimum Euclidian distance and last generation number. In the following we explain these factors.

Minimum Euclidian Distance. For each design point, the Euclidian distance (in the objective space) between a solution and the nearest global Pareto optimal point is calculated. A smaller distance indicates that the solution is closer to the optimal solutions and therefore is better. Thus, in the DSE tree, the edges in the path from the root to the global Pareto optimal points are the thickest and darkest since the distance is zero (see 3(a)). As the distance increases the edges become thinner and lighter. In this manner, just by looking at the DSE tree, the designer can easily determines which parts of the design space contain optimal and near optimal solutions and which parts contain solutions that are far away from the optimal solutions.

Last Generation Number. The number of the last generation in which a design point is evaluated can be considered as an importance factor. As the MOEA gradually converges to a set of Pareto optimal points, we expect better design points in the later generations. The edge visualization can show the progress of the searching algorithm in exploring and covering the design space during its generations. The edges with a higher generation number in their subtrees (i.e. the design points in their subtrees are evaluated in the later generations) are thicker and darker. As a result, the paths from the root to the last generated data nodes are the darkest and thickest paths.

As the importance factors are not applicable for edges that have no data nodes in their sub tree, these edges are shown by gray dashed lines.

4.4. Visualization of the Design Space Coverage

In VMODEX, we provide a technique to show how a heuristic searching algorithm walks through the design space and accesses new parts of the design space during its exploration process. To do this, a color-coding scheme is used to color the parameter nodes based on the generation number in which a design parameter is explored for the first time. Thus, the color of each parameter node represents the first generation that a design point, containing that parameter, is evaluated. On the other hands, for each parameter node, the number of the generation in which the first design point is added to its subtree, is used for coloring the node. For visualizing the design space coverage, the green color with variations in lightness and saturation is used, such that the color of parameter nodes, which are searched in the earlier generations, is light green. The color of nodes becomes darker as they are investigated at the later generations. Therefore, the designer can easily see how a searching algorithm covers different parts of the design space during its successive iterations. In Section 6, an example of this visualization is shown for our case study results (Fig. 15(b)). In Fig. 15(a), the color legend for visualizing the design space coverage is shown.

5. Interactive Exploratory Techniques

User interaction plays an essential role in the effective visualization of data and information. It allows users to explore and work with the data actively to investigate the data from different perspectives. Interactive visualizations are very useful in analyzing data and can provide new insights that could not be obtained using static graphics or statistical methods. Interactivity means that users cannot only observe the visualization but moreover play with it. The visualization changes in response to the users actions and enable users to create customized views that are more interpretable and informative to be able to find hidden patterns in the data and complex relationships among variables. Colin Ware [41] notes,

> *”The best visualizations are not static images to be printed in books, but fluid, dynamic artifacts that respond to the need for a different view or more detailed information”* (p. 385).

A user cannot be expected to deal with a single image consisting of thousands of information elements. Therefore, we have provided a set of interaction techniques in VMODEX to help designers to handle large design spaces and to analyze the data and explore the search results from different perspectives and at multiple levels of abstraction in order to find out interesting and important features that may not be found just by looking at the DSE tree visualization.

5.1. Zoom

Zooming and scrolling are traditional tools in visualization; they are quite indispensable when large trees are explored. In our visualization tool, besides normal zooming and scrolling, we provide two extra zooming features for augmenting the exploration process: *bird view* and *satellite view*.

Bird view provides overview (context) and detail (focus) on the same screen by allowing users to view items at different scales. The aim of bird view is to enlarge items closer to the users point of interest while compress the objects that are farther away from the interested region. Bird view is a window moving with the mouse-pointer and works like a magnifier.

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 in the tree without losing track of the position in the entire scene.

5.2. Hiding nodes

In VMODEX, two options are provided to reduce the number of visible nodes in the DSE tree in order to make its size smaller for better and more effective exploration. It should be mentioned that by hiding some nodes, we recalculate the location of visible nodes in the tree to optimize their fit on the screen. This means that, the empty spaces from the hidden nodes are occupied by the visible nodes. Note that the order of the visible nodes in the tree will not be changed. So, the structure of the tree remains the same.

Hiding Sub Trees without Exploration Data. Since we use heuristic search techniques, some areas of the design space may not be visited by the searching algorithm (e.g., they are not interesting enough). So, we do not have any evaluated design points for those parts. In VMODEX, it is possible to hide the sub trees of the nodes that have no evaluated data. This way, the designer can focus on the explored sub trees, which are more important. However, if the designer is interested to see the entire design space (both evaluated and not evaluated parts), there is another option in the tool that shows all parts of the design space. In this case, the color of parameter nodes that are not visited by the searching algorithm is white. So, the designer can easily recognize which parts of the design space are not searched during the exploration process.

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 (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 is, the more nodes in the sub tree exist. By double clicking again, the sub tree expands (becomes visible) and the blue triangle is removed. If the designer is interested to know the exact number of nodes in a hidden sub tree, by clicking its corresponding triangle the amount of nodes is shown.

5.3. Filtering

The filtering option allows designers to easily filter out parts of the solutions, which are not required for further investigation, and view only preferred design points. Therefore, the designer can focus on the more interesting design points and find the similarities/dissimilarities between them. Various kinds of filtering are available based on different designer preferences. The combinations of different filtering approaches are also possible. The filtering results can be shown in three ways: view all design points that fall within the filtering conditions or only see local Pareto optimal points or only global Pareto optimal points.

Filtering Based on the Objective Values. In some cases, the designer wants to consider only design points with some specific objective values (e.g. design points with the best processing time). 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 design points with preferred objective values and find out the relationships between them.

Filtering Based on the Design Parameters. VMODEX allows designers to hide the parts of the design space that are not being considered for further analysis to pay more attention to the more interesting parts. For instance, the designer may not be interested in design points with more than four processors. Then it is possible to hide those subtrees in the DSE tree that contain more than four processors and focus only on those parts of the design space that contain design points with the preferred parameters.

Filtering Based on the Distance from the Global Pareto Optimal Set. In the multi-objective context, the goodness of a design point can be evaluated by its distance from the Pareto optimal set. The smaller the distance the better the solution. This distance measure is the Euclidean distance (in the objective space) between the design point and the nearest member of Pareto optimal set. In VMODEX, we provide a filtering option based on this distance measure. The designer is able to define a distance threshold and then filters the design points to see solutions that have lower/higher distance values than the threshold. Furthermore, as in VMODEX both the design parameters and objectives are shown in a single view, the designer can easily understand which parts of the design space contains solutions that are close to/far away from the Pareto optimal set.

Filtering Based on the mapping decision. For better analyzing the effect of different mappings on the objective values, we provide the following filtering option. The designer can specify some constraints on the mapping and then filters the design points to see only those solutions that not violate the specified mapping constraints. After that, it is possible to investigate the values of different objectives for the design points, which satisfy the mapping restrictions and find out the influences of mapping decisions on the objective values. There are three ways for defining a mapping constraint:

1. Some specific tasks (determined by the designer) are mapped on the same processor, without specifying the type of the processor. In this case, we can study the effectiveness of the internal communications between them as well as comparing the results of using different processor types for executing those tasks.
2. Some specific tasks are not mapped on the same processor. So, we can investigate the effect of using (shared) memories for their communications.
3. For one or more tasks a specific processor type is determined to be mapped to. In this case, we can investigate the effect of other tasks mappings on the objective values.

5.4. Step by Step Animation

Since exploring the design space is an iterative process, it is important to know how a searching algorithm walks through the design space and trace its progression in finding new design points and covering the design space during successive iterations. In VMODEX, we provide an interactive user interface that allows designers to follow the evolutionary exploration process during numerous generations. This interface is called step-by-step animation. The designer chooses an arbitrary generation number as a start point. At this step, the DSE tree shows only those design points that are found in the specified generation number or in earlier generations. Then, the designer can move forward or backward in generations to see the dynamic exploration. When moving through the generations (i.e., replaying parts of or even the entire search process), it is important to know in each generation which new design points are found (added to the DSE tree). To show this, the design points generated in the current generation are blinking. If a parameter node, which had no data node in the previous step, receives its first data node in the current generation, it starts blinking as well. Therefore, the designer can easily notice that some new parts of the design space are visited for the first time in the current generation. For the hidden sub trees, in case of any data node addition, the blue triangle starts blinking and if the user is interested in viewing that sub tree, it can be expanded as well.

5.5. Details on Demand

Because of limited screen space as well as high data complexity, showing every detail for all design points is impractical. Therefore, generally, in proposing a visualization method, only the most important data features are shown directly in the visualization and detailed information for interesting parts is shown just after the user requests them. In VMODEX, a variety of detailed information are provided, enabling designers to interpret the evaluated design points from different aspects and gaining additional insight into the underlying information. This detailed information has been described earlier in the previous section and are called the secondary objectives. By right clicking on an interesting design point, a pop-up menu appears that shows the available detail options. The user can simply select the desired detailed information from the list and see it.

5.6. Adjusting DSE tree Appearance

In VMODEX, there are some options for modifying the way that the DSE tree looks, based on alternatives for showing/hiding sub tress, coloring the parameter nodes and tree edges.

Showing/Hiding Sub trees. As we explained in Section 5.2, the designer can choose to see the entire design space or only those parts of the design space that are visited by the searching algorithm. Furthermore, it is possible to hide the sub tree of a (uninteresting) node by double clicking on it.

Coloring Parameter Nodes. As we described in Section 4.2.1, parameter nodes do not represent single design points and therefore do not have the direct notion of processing time or energy consumption. For this reason, we provide 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.

Another option for coloring the parameter nodes is based on the generation number in which the searching algorithm could get access to that part of the design space for the first time. (See Section 4.4)

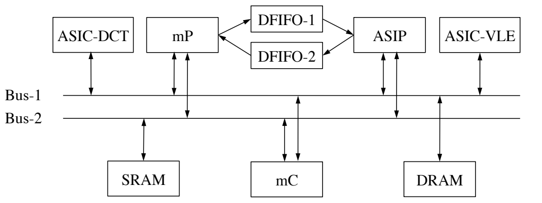


Figure 12: Heterogeneous multi-processor system-on-chip architecture model with 5 processors, 2 shared memories and 2 dedicated point-to-point FIFOs.

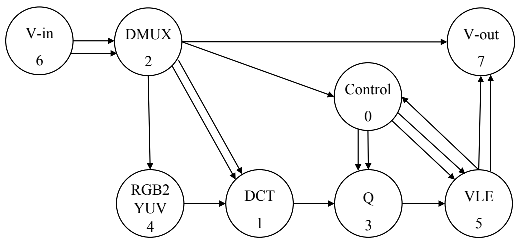


Figure 13: M-JPEG encoder process network graph with 8 processes and 17 FIFO channels.

Coloring and Thickness Regulation of Tree Edges. As we explained in Section 4.3, there are two options for visualizing the tree edges based on the minimum Euclidian distance or the last generation number.

6. 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 a heterogeneous multi-processor system-on-chip platform architecture consisting of a general-purpose microProcessor (mP), a microController (mC), an Application Specific Instruction Processor (ASIP) and two Application Specific Integrated Circuits (ASIC-DCT and ASIC-VLE). For communication, the platform architecture contains two dedicated point-to-point FIFOs (between mP and ASIP) and two shared memories; one Static RAM (SRAM) and one Dynamic RAM (DRAM), each one is accessible through a common bus. Note that the evaluated design instances only use a subset of the platform resources, based on the mapping of application tasks and communication channels onto the platform resources. Fig. 12 represents our MP-SoC architecture model and Fig. 13 shows the M-JPEG encoder application.

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, Q and RGB2YUV. The ASIC-DCT is designed for executing the DCT process and ASIC-VLE is dedicated for executing the VLE process.

The design space that we consider in this study has four parameters: number of processors, processor type, number of memories and memory type. Each architecture platform instance is indicated by a unique combination of these parameters. Those combinations that lead to the platform instances that are capable of executing the application are denoted as *Possible Combinations*. We formulated the design space exploration problem as a multi-objective optimization problem [42]. Using the NSGA-II multi-objective evolutionary optimizer [10], 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 MOEA for 100 generations with 100 individuals per population. Therefore, 10000 design points are searched by the MOEA. In this experiment, we have used the uniform crossover with probability 0.9. The mixing ratio between two parents is 0.5 (i.e. the offspring has approximately half of the bits from first parent and the other half from second parent). The mutation probability [2] is 0.23 and the bit mutation probability [3] is 0.01. For evaluating design points in terms of aforementioned criteria, we used the Sesame system-level simulator. In our case study, the processing time varied by factor of 6.4, energy consumption by factor of 4.45 and architecture cost by factor of 10.25. These large variations indicate the need for automated exploration and optimization.

It should be mentioned that the goal of this case study is to illustrate how our proposed visualization methods can help designers to gain additional insights. Actually, for evaluating the effectiveness of our proposed visualization tool and examining how successful it is in generating insights, we performed a real

[2]i.e., the likelihood of mutating a particular individual.
[3]i.e., the likelihood of mutating each bit of an individual in mutation.

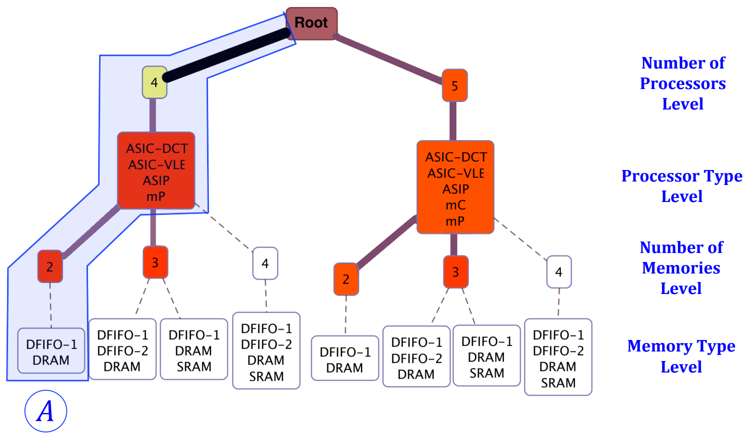


Figure 14: Platform instances that are not visited by the MOEA. The white nodes represent those parts of the deign pace that have no evaluated design point.

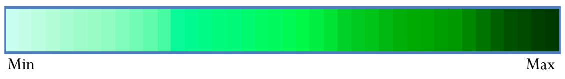
case study and used our tool to explore the results and see what new and unexpected insights we can gain from the tool that is very hard (if not impossible) to find out those insights from the raw data.

VMODEX allows designers to look at the evaluated design points and explored design space from different perspectives and analyze the results at multiple levels of abstraction. In the following subsections, we analyze the M-JPEG case study with respect to the following issues: 1) design space coverage, 2) The characteristics of the global Pareto optimal points, 3) investigating the absence of ASIC-VLE in the Pareto optimal set, and 4) studying the effects of executing the DCT process by different processor types on design criteria.

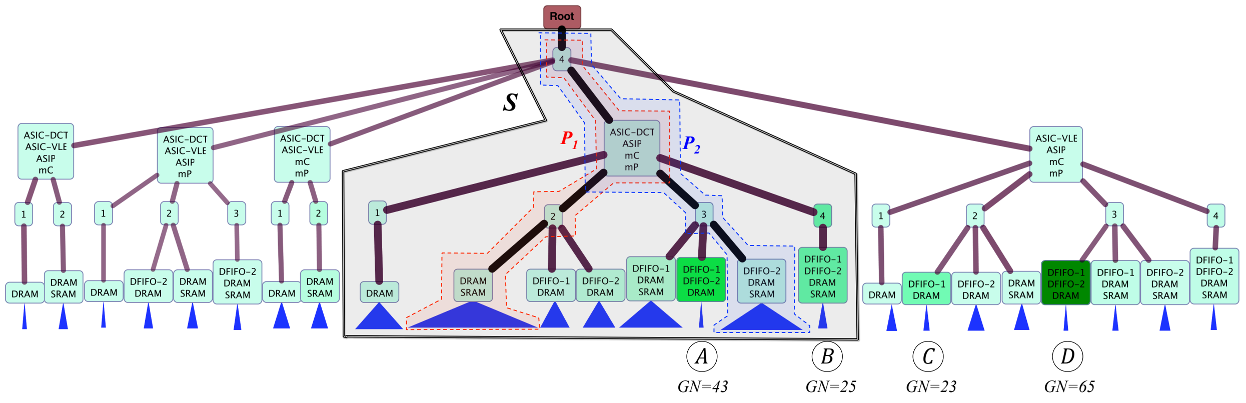
6.1. Design Space Coverage

Since we use an evolutionary algorithm (instead of exhaustive search) for exploring the design space, the searching algorithm may not visit some parts of the design space. Therefore, there is no evaluated data for those parts. However, it is essential that a searching algorithm achieves a broad coverage of the design space. This means that the algorithm is able to find solutions in as many as possible different regions of the decision space, even if those solutions are not Pareto-optimal. In this case, the exposed solutions represent the variety of possible designs in the decision space and enable the designer to do a more comprehensive study on the relationship between design parameters and their effects on the design criteria. Furthermore, a higher diversity and coverage in the searching algorithm also means that there is a higher chance of finding the global optimal solutions rather than the local optimum ones (the algorithm does not get trapped in local optimum points in the design space). The spread of design points in the design space can help the algorithm to escape from such local optima.

Using VMODEX, the designers can easily see the parts of the design space that are covered/not covered by the MOEA. This tool clearly shows which parts of the design space contain evaluated solutions and which parts are not searched at all (no design point is evaluated there). As we have described before, parameter nodes with a white color and dashed line have no data. In our case study there are 99 possible combinations of design parameters. By visualizing the exploration results, we could find out that more than 90% of possible platform instances are searched (91 out of 99). For platforms with two and three processors, all the possible combinations are searched. Further, only a few combinations of platforms with four and five processors have not been visited by the MOEA. Those parts of the design space that are not investigated during the exploration process are shown in Fig. 14. In this figure, minimum processing time is used for coloring the parameter nodes. The white nodes represent the combinations that are not searched by the MOEA. For instance, there is no evaluated design point that consists of four processors of the types ASIC-DCT, ASIC-VLE, ASIP and mP and two memories of which one is DFIFO-1 and another one is DRAM (this platform instance is indicated by subtree A in Fig. 14). However, the color of the node labeled with



(a) Color legend



(b) Four-processor architecture instances

Figure 15: Visualization of design space coverage for our case study. For each parameter node, the color of the node represents the number of the generation that the searching algorithm visited that part of the design space for the first time (i.e. generation when a solution appeared).

"2" in this subtree (at the number of memories level) indicates that the other combinations of two memories such as one DRAM and one SRAM are searched.

Next, we are going to study how our MOEA walked through the design space and found new architecture platform instances during its search iterations. We used the visualization of design space coverage option in VMODEX to understand the searching behavior of the MOEA in the design space. By selecting this option, parameter nodes are colored based on the generation number in which the searching algorithm gained access to that part of the design space for the first time. Therefore, the color of the parameter nodes, which are searched in the earlier generations, is lighter and those parameters that are visited in the later generations are darker. In our case study, we found that most of the architecture platform instances (possible combinations of design parameters) are searched in the 15 first generations. Actually, in our initial population, which is generated randomly, 58 unique instances are visited and then until the 15th generation totally 86 (out of 99) possible combinations are searched. In the next generations, the algorithm could found 4 more instances and after the 65th generation it could not find any new combination of design parameters. Fig. 15(b) shows the visualization of design space coverage for our case study. Since the size of the DSE tree representing the entire design space is large and it is not possible to include it here, this figure shows only four processors architecture instances. Furthermore, the four last combinations that are found by the algorithm are in this part of the design space. So, we can easily see which architecture instances are searched in the later generations.

As can be seen in this figure the color of most parameter nodes is light green, which indicates that those parts of the design space are searched in the beginning generations. Only four nodes (denoted by A-D) are colored with darker green, which points out that those architecture instances are visited in the later generations. The number written at the bottom of these nodes shows the number of the first generation during which the algorithm has investigated them.

As we described before, the size of the blue triangle at the bottom of each node represents the number of design points in its subtree. From Fig. 15(b) we can see that the platform instances containing four processors of the types ASIC-DCT, ASIP, mC and mP (subtree indicated by S) are searched more often by our MOEA since nodes in this subtree have the biggest triangles.

In Fig. 15(b), the minimum Euclidian distance is used as an importance factor for edge visualization.

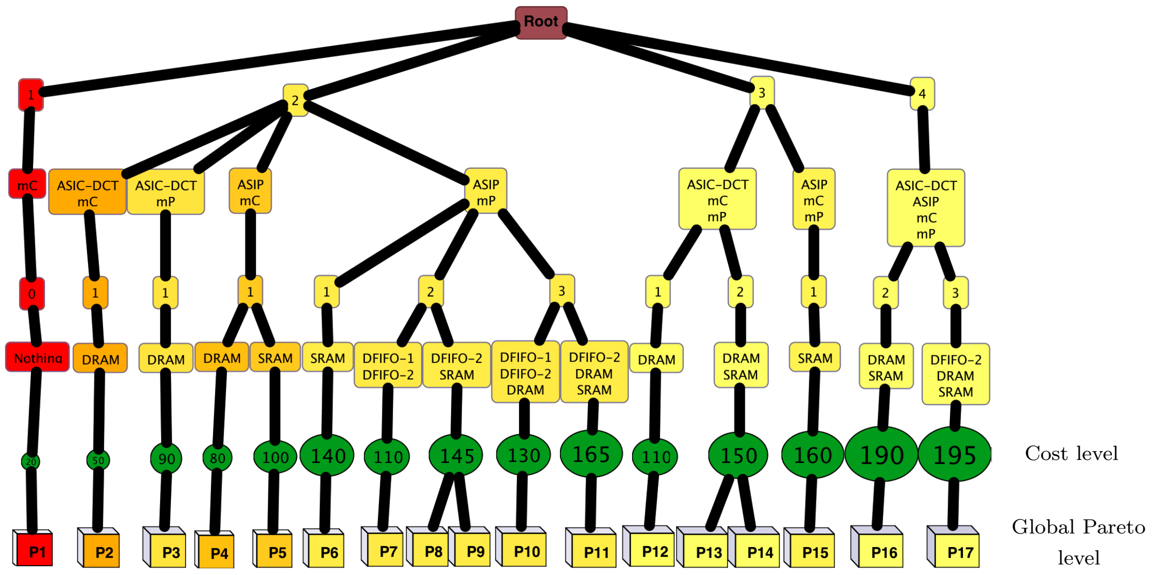


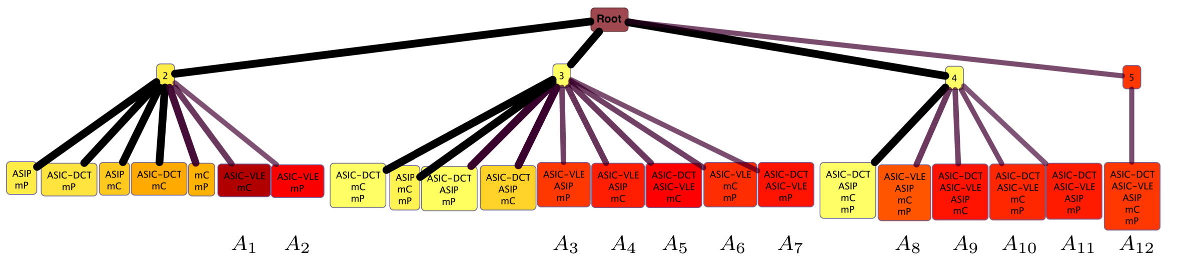
Figure 16: Global Pareto optimal points found by the MOEA (denoted by P_1-P_{17}). 15 unique combinations of architectural components lead to the 17 Pareto optimal points.

Therefore, the edges in the path from the root to the global Pareto optimal points are the thickest and darkest (black color) since the distance is zero. From Fig. 15(b) we can see that among four processors platform instances only one of them leads to the global optimal design points, which is indicated by subtree S. So, this explains why the searching algorithm evaluated more design points in subtree S, as, in this part of the design space it has found design points that are closer to the optimum solutions. Within subtree S, two combinations of memory types yield optimum solutions. The two architecture instances that contain optimum design points in their subtrees are denoted by P_1 and P_2.

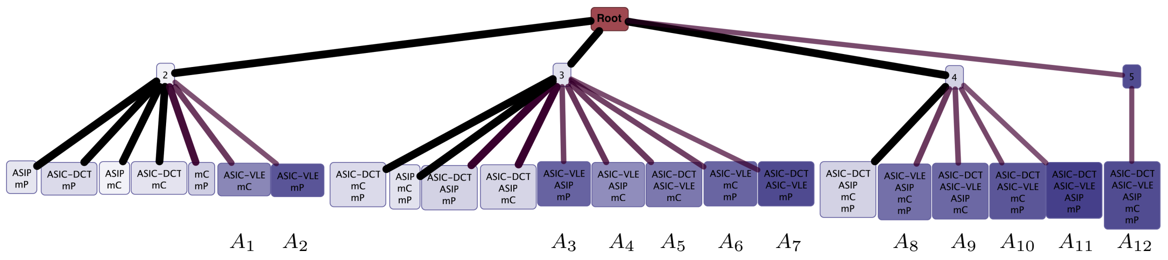
6.2. The Characteristics of the Global Pareto Optimal Points

Fig. 16 shows the global Pareto optimal points found by our MOEA. By looking at the picture, the designer can immediately recognize the characteristics of the Pareto optimal points, which are the best design points with respect to the design criteria. As in the DSE tree both design parameters and objective values are shown in a single view, the designer can easily find out which combinations of architectural components yield optimum design points and what the trade off is between objective values. As can be seen in Fig. 16, in our case study, fifteen unique combinations of architectural components lead to the seventeen Pareto optimal points. The variety of architecture cost in the Pareto optimal set is quite large. The cost is varying from the cheapest one to almost the most expensive one that have been encountered during the whole search (approximately $10x$ (195/20)). For the processing time, except the design point with a single processor architecture (labeled by "P1" in Fig. 16) all the other points have relatively good processing time. The normalized value of the processing time (excluding P1) is in the interval [0, 0.21]. The energy consumption of all the found Pareto optimal points is relatively low. Its normalized value is in the range [0,0.16].

Now we are going to analyze the discovered Pareto optimal points in terms of design parameters. From Fig. 16 we can see that, in our case study, there is no Pareto optimal point with five processors. This means that with less processors (which is cheaper) we 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 most of the obtained Pareto optimal points (10 out of 17) contain two processors in their underlying architecture. Thus, maybe two processors architectures are more suitable for executing



(a) Minimum processing time is used for coloring parameter nodes. For all platform instances containing ASIC-VLE (indicated by A_1-A_{12}), the processing time is quite high (the color of node is red).



(b) Minimum energy consumption is used for coloring parameter nodes. For all platform instances containing ASIC-VLE (indicated by A_1-A_{12}), the energy consumption is quite high (the color of node is dark violet).

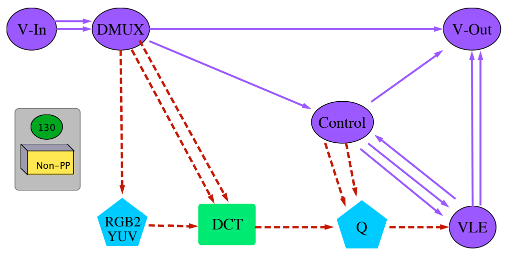
Figure 17: Possible combinations of processor types

our application. It can also be seen that there is no Pareto optimal point with four memories. So, the communications between processors can be done efficiently with less memories. Furthermore, we can see that the processor ASIC-VLE, which is dedicated for executing the VLE process, is never used in the Pareto optimal set. In the next subsection we are going to find out the reason of this remarkable property.

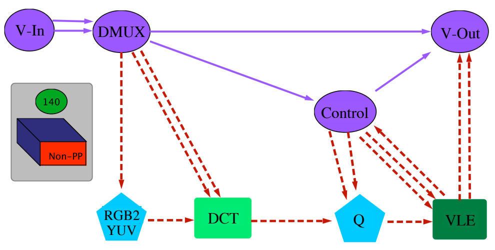
6.3. Investigating the Absence of the ASIC-VLE in the Pareto Optimal Set

As we can see in Fig. 16, there is no Pareto optimal point that contains ASIC-VLE in its underlying architecture. By looking at the DSE tree representing the entire design space, we can find that there are several evaluated design points that use ASIC-VLE. Thus, our MOEA could access the parts of the design space that contain ASIC-VLE. However, the quality of design points in these parts with respect to the design criteria is not optimal and they are dominated by the solutions in the other parts. Fig. 17 shows all the possible combinations of processor types. To make the picture fit here, different alternatives with respect to the number and types of memories are not shown. Thus, only the first two levels of the parameter segment are shown (i.e. the number of processors and processor types). In Fig. 17(a) the minimum processing time and in Fig. 17(b) the minimum energy consumption is used for coloring the parameter nodes. As can be seen in these two figures, in all of the platform instances containing ASIC-VLE, both the processing time and energy consumption is quite high.

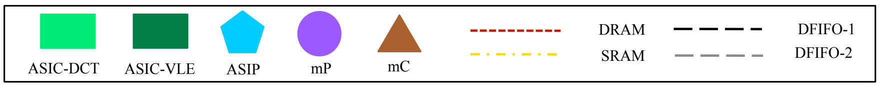
We use filtering options provided in VMODEX to realize why both the processing time and energy consumption are significantly increased when using an ASIC-VLE. First, we filter design points based on their primary objective values. We were interested to see all design points which are relatively good in both processing time and energy consumption (their normalized value is in the range [0, 0.2] for both objectives). There was no restriction in the architecture cost objective. We found out that in all of them three processes V-Out, VLE and Control are mapped to the same processor. We did similar filtering to see those design points that are extremely poor in both (their normalized value is in the range [0.8, 1.0] for both objectives). Then, we could recognize that in all of them, the three aforementioned processes are mapped to different processors. Thus, we can conclude that the communications between these processes are



(a) VLE is mapped to the same processor as V-Out and Control. Both processing time and energy consumption are relatively good.



(b) VLE is mapped to the dedicated ASIC-VLE processor. Both processing time and energy consumption are quite high.



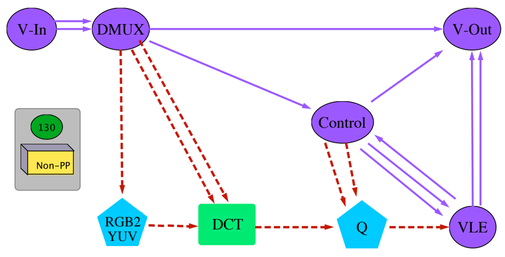
(c) Legend for shape and color of processor and memory types

Figure 18: Mapping decision (with and without ASIC-VLE). Objective values are shown at the left side of mapping visualization.

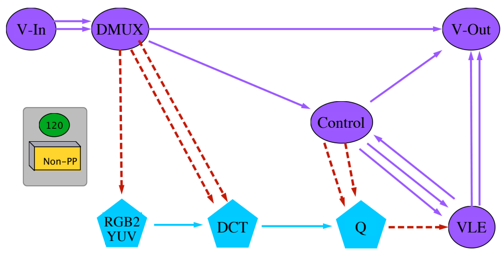
much more intensive than the communications between other processes.The reason for this is that over these channels tables for Huffman encoding are being transmitted (see [43]). However, for the other processes, the communications are all the same as they communicate pixel blocks, which are small units of data. Therefore, using shared memories for these communications causes large overheads that can actually make the system slower. However, if these communications are done internally (mapping in the same processor) the mapping is much more efficient with respect to both processing time and energy consumption. As ASIC-VLE is dedicated for executing the VLE process, by using it, the processes V-Out and Control have to be mapped on the other processors and therefore their communications will be done externally using shared memories, which is not efficient. Thus, even though using an ASIC implementation for the VLE process is computationally efficient, because of its intensive communication, it is not an effective solution. In Fig. 18 we illustrate this conclusion with an example. This figure represents the mapping decisions of two design points in which they have exactly the same mapping for all the processes except for VLE. In Fig. 18(a) the VLE process is mapped to the mP (the same processor as V-Out and Control) and in Fig. 18(b) it is mapped to the ASIC-VLE. From the visualization of their objective values we can see that they are significantly different in both processing time and energy consumption. The legend for shape and color of processor and memory types are shown in Fig. 18(c).

6.4. Studying the Effects of Executing the DCT Process by Different Processor Types on Design Criteria

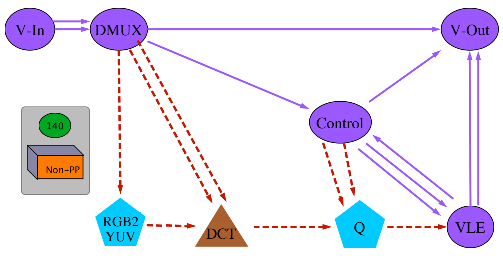
In the M-JPEG application, the DCT is the most computationally intensive process and therefore it is essential to be mapped on the processor that is optimal for executing it. We are going to investigate how the values of processing time and energy consumption change by running the DCT on different types of processors. To do that, we use mapping filtering. From the previous discussion we understand that if three processes V-Out, VLE and Control are not mapped on the same processor, regardless of the mapping of the other processes, the processing time and energy consumption are quite high. So, in this study, we assume that these three processes are mapped on the same processor and then we examine the influences of different mappings for DCT on the objective values. Using mapping filtering, we could find out that if DCT is mapped to ASIC-DCT or ASIP both the processing time and energy consumption is relatively good. But, by executing the DCT on the mC or mP, we cannot get any satisfactory solution. To illustrate this conclusion, an example is shown in Fig. 19. This figure represents the mapping decisions of four design points in which the mappings of all processes excluding DCT are the same. The variation on mapping the DCT are shown in Fig. 19(a) to Fig. 19(d). From the visualization of the objective values we can see the effect of each mapping on processing time and energy consumption.



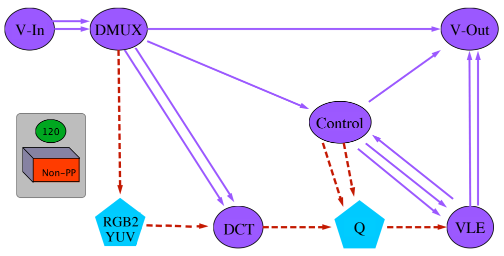
(a) DCT is mapped to the dedicated ASIC-DCT



(b) DCT is mapped to the ASIP



(c) DCT is mapped to the mC



(d) DCT is mapped to the mP

Figure 19: Illustration the effects of executing DCT process by different processor types on design criteria.

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.

7. Comparing Subspaces

As we described in Section 4.1, in VMODEX, the design space is modeled as a tree and this kind of modeling causes the design space to be divided in several subspaces. Each subspace represents a unique instance of the architecture platform. On the other hand, solutions inside a subspace have exactly the same architecture components (have the same parents at the parameter levels) but the way that the application is mapped onto those components is different. We have provided additional functionality in VMODEX to allow designers to evaluate and compare the properties of interesting subspaces from various perspectives.

Fig. 20 represents the design space of our case study, which is visualized by VMODEX. This figure shows only those parts of the design space that are compared with each other in this section. To make the picture smaller to be able to put it here, only global and local Pareto points are shown and non-Pareto points are not displayed. In the DSE tree shown in Fig. 20 there are five subspaces, which are indicated by dashed contour lines and are labeled from S_1 to S_5. For instance, the subspace S_1 consists of one ASIP, one mC, one DRAM and one SRAM. Thus, all design points in this subspace utilize these architecture components (same resource allocation) but their task and communication bindings are different. In each subspace, the Pareto optimal solutions found by a MOEA are called local Pareto optimal solutions, which are optimal with respect to a specific architectural instance. However, in the entire design space, a design point might exist which dominates the local Pareto point. In the following subsections, we explain the techniques VMODEX provides for analyzing and comparing the properties of discovered design points in different subspaces. We use these techniques for comparing the properties of the five local Pareto optimal sets, which are labeled by LPS_1 to LPS_5 in Fig. 20.

The concept of local Pareto solutions and proposing some methodologies for evaluating and comparing different local Pareto optimal sets (like we explain here) is a new point of view in the multi-objective DSE process and has not been considered before. Such comparison is, however, essential to the designer as it

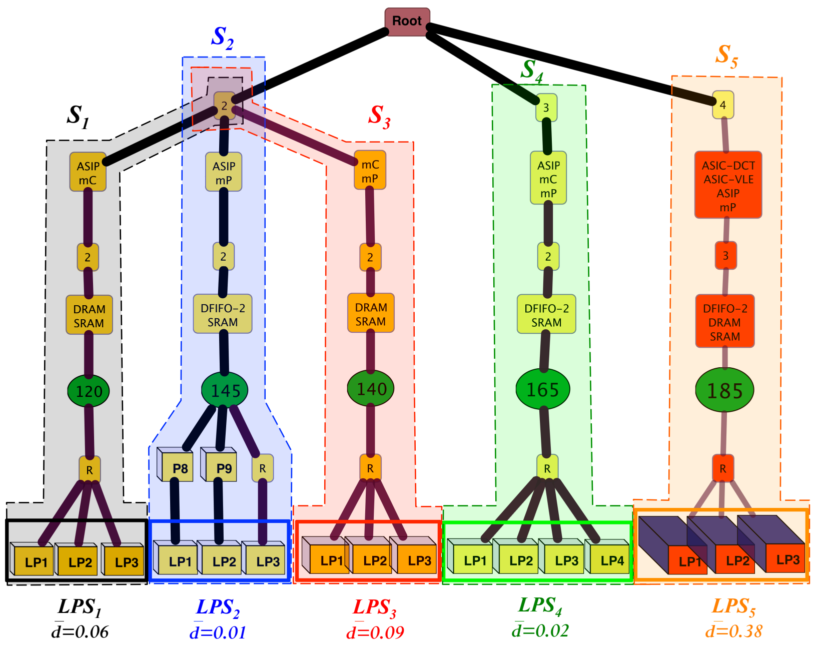


Figure 20: Subspaces of the design space. Five subspaces are indicated by dashed contour lines and are labeled from S_1 to S_5.

provides insight into the landscape of the design space and can help him to comprehensively understand the properties of the discovered design points in different subspaces of the explored design space. Using VMODEX the designer is able to select the interesting subspaces and compare them with respect to the different aspects, which are explained in the following subsections.

7.1. Distance from the Global Pareto Optimal Solutions

A subspace containing local Pareto optimal solutions, which are closer to the global Pareto optimal set, is more preferable. The distance can be measured in two ways: 1) the number of solutions in a local Pareto optimal set which are also in the global Pareto optimal set, and 2) the average of Euclidian distances (in the objective space) between the solutions in a local Pareto optimal set and the nearest member of the global Pareto optimal set.

In the DSE tree, both distance measures can simply be evaluated. Just by looking at the tree, one can easily recognize which solutions of a local Pareto optimal set are in the global Pareto optimal set as well. For those solutions, their parents (in the tree) are Pareto optimal nodes; otherwise they become children of a relation node. For example, in Fig. 20, two solutions (out of three) in the set LPS_2 are globally Pareto optimal. However, in the other sets, none of the solutions are in the global Pareto optimal set (their parents are relation nodes).

The second distance measure can be seen directly in the DSE tree as well. The color and thickness of edges show the distance from the nearest global Pareto optimal solutions. The edges in the path from the root to the main Pareto optimal solutions are the thickest and darkest since the distance is zero. As the distance increases the edges become thinner and lighter. The value of the average distance between solutions in a local Pareto optimal set and global Pareto optimal set is shown at the bottom of the local Pareto set (denoted by $\bar{d}$). If the designer is interested to know the exact value of the distance measure for a particular solution, then the distance value is shown by clicking the corresponding edge. For example, in Fig. 20, the average distance for LPS_2 is 0.01 which means that solutions in this set are quite close to the Pareto optimal set. However, the solutions in set LPS_5 are relatively far from the global Pareto optimal set since their edges are thin and light and the average distance is 0.38.

7.2. Coverage of Local Pareto Sets

Ziztler and Thiele [44] introduced the Coverage (C) metric, which directly compares two Pareto optimal sets with each other. The metric $C(P_1, P_2)$ calculates the proportion of solutions in Pareto optimal set P_2 which are weakly dominated at least by one solution in P_1:

$$C\left(P_1, P_2\right) = \frac{|\{p' \in P_2 | \exists\, p \in P_1 : p \leq p'\}|}{|P_2|} \tag{12}$$

Where $\leq$ is the weakly dominance relationship. For two solutions p and p' it is said that p weakly dominates p' , if p is not worse than p' in all objectives. In fact, the function C maps the ordered pair (P_1, P_2) to the interval [0, 1]. The value $C\left(P_1, P_2\right) = 1$ means that all members of P_2 are weakly dominated by P_1 and $C\left(P_1, P_2\right) = 0$ represents the situation where none of the solutions in P_2 are weakly dominated by P_1. Note that for fully understanding the dominance relations between two Pareto optimal sets, both directions $C\left(P_1, P_2\right)$ and $C\left(P_2, P_1\right)$ have to be considered, since $C\left(P_1, P_2\right)$ is not necessarily equal to $1 - C\left(P_2, P_1\right)$. The C metric compares only two sets with each other. Thus, for comparing more than two sets, we propose a new metric called Total Coverage (TC), as follows:

$$TC\left(P_i\right) = \sum_{j=1, j\neq i}^{n} C\left(P_i, P_j\right) - C\left(P_j, P_i\right) \tag{13}$$

Where n is the number of comparing sets. $TC > 0$ means that the dominating rate is higher than the dominated rate and the $TC < 0$ implies that the solutions are more dominated by the other sets than they dominate solutions in the other sets. Therefore, a set with a bigger TC value is better. In VMODEX, the designer is able to select the interesting local Pareto sets and compare them using the TC metric. For better understanding the dominance relations between sets, we visualize this metric. A directed weighted graph is used for visualizing the TC metric. Each comparing set is shown as a node in the TC graph. For each two sets P_1 and P_2, if $C\left(P_1, P_2\right) \neq 0$ then an edge is drawn from P_1 to P_2, of which the weight is equal to the C value. In the case of $C\left(P_1, P_2\right) = 0$, there is no edge from P_1 to P_2. Due to the summation property, we can break up a summation across a sum or difference. Therefore, we can rewrite the TC formula as follows:

$$TC\left(P_i\right) = \sum_{j=1, j\neq i}^{n} C\left(P_i, P_j\right) - \sum_{j=1, j\neq i}^{n} C\left(P_j, P_i\right) \tag{14}$$

According to the TC graph, the first sum in Equation 14 is equal to the sum of the weights of outgoing edges and the second sum is equivalent to the sum of the weights of incoming edges. Thus, for each node in the TC graph, the value of TC metric is calculated by the sum of the weights of outgoing edges minus the sum of the weights of incoming edges:

$$TC\left(P_i\right) = \sum (Outgoing\,Edges) - \sum (Incoming\,Edges) \tag{15}$$

The size of the nodes in the graph indicates the TC value. Therefore, nodes with higher TC values are bigger. Because the TC value greater than zero (means more dominating) or less than zero (means more dominated) have completely opposite meaning, we demonstrate this property in the nodes color. Nodes with $TC > 0$ are shown in blue while nodes with $TC < 0$ are shown in red. Fig. 21 shows the visualization of the TC metric for sets LPS_1 to LPS_5 shown in Fig. 20. As can be seen in this figure, all solutions in LPS_5 are dominated by all the other comparing sets (there are four incoming edges from LPS_1 to LPS_4 with the weight $4/4 = 1$), while there is no solution in LPS_5 that dominates a solution in the other sets (LPS_5 does not have any outgoing edge). Thus the TC value for LPS_5 is less than zero ($TC = -4$) and this node is shown by a red color. Furthermore, we can understand that solutions in LPS_1 dominate all the solutions in set LPS_3, while no solutions in LPS_1 are dominated by any other sets, since there is no incoming edge. Moreover, we can see that half of the solutions is LPS_4 are dominated by solutions in LPS_2. As a result, LPS_1 has the highest TC value (biggest node in the graph) and thus is the best local Pareto optimal set among the comparing sets with respect to the TC metric.

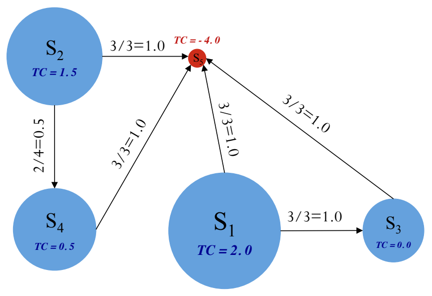


Figure 21: Visualization of the TC metric. The size of the nodes indicates the TC value. Nodes with $TC > 0$ are blue and with $TC < 0$ are red.

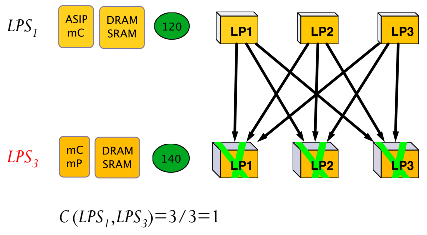


Figure 22: Visualization of the C metric. Cross symbol indicates the dominated solutions.

If the designer is interested to know more about the dominance relation between each two sets, such as which solutions in one set dominate which solutions in the other set, it is possible to select those sets to see more details. To this end, we visualize the dominance relation between two sets as follows. Solutions in both sets P_1 and P_2 are shown in two different rows. If a solution in P_1 dominates a solution in P_2, an arrow is drawn between them coming out from the solution in P_1 to the solution in P_2. Furthermore, a cross is displayed at the dominated solution in P_2 to show that this solution is dominated by another one. Fig. 22 shows the visualization of the dominance relation between two local Pareto sets LPS_1 and LPS_3. From this figure, we can understand that all solutions in set LPS_1 dominate all the solutions in LPS_3 and therefore LPS_1 is absolutely better than LPS_3. The architectural components of each local Pareto set are shown on the left side of each set. As can be seen in Fig. 22, both local Pareto sets have the same components except that in LPS_3 the ASIP is replaced by mP. Using an mP instead of an ASIP is not beneficial since it increases the cost but does not yield a better processing time nor a better energy consumption.

7.3. Size of the Dominated Region

In [44], the *Hypervolume* (HV) metric is proposed, which measures how much of the objective space is dominated by a given Pareto optimal set. Fig. 23 illustrates the Hypervolume metric. In this figure, the back points are the Pareto optimal solutions and the gray region represents the Hypervolume metric for two objectives (f_1 and f_2) where these objectives are to be minimized. Each point in the gray region is dominated at least by one member of Pareto optimal set. The reference point (W) can simply be found by constructing a vector of worst objective values. A set with a larger Hypervolume is desirable. We use this metric to compare the local Pareto optimal sets in different subspaces of the design space. Moreover, we visualize this metric to clearly show which area of the objective space is dominated by solutions in a local Pareto set. Fig. 24 represents the visual form of the Hypervolume metric for two local Pareto sets LPS_1 and LPS_5 shown in Fig. 20. Each side of the cube shows one objective. The colored region indicates the dominated area in the objective space. For better vision, each side of the cube is colored with the corresponding color scheme in the DSE tree. Black lines in the colored region denote the solutions in a local Pareto optimal set. Since we have used the normalized value of objectives, the reference point is $W = (1, 1, 1)$. The normalized values of processing time (PT) and energy consumption (EC) for each solution, is written at the bottom of the cube. As can be seen in Fig. 24, in set LPS_1 almost half of the objective space is dominated while only a small portion of the objective space is covered by the solutions in the LPS_5. Using the Hypervolume visualization, it is possible to see the dominated parts of each objective separately. For instance, in Fig. 24 the energy consumption and processing time surfaces are almost completely covered for LPS_1 while only half of the cost surface is dominated.

7.4. Sensitivity of Subspaces to Different Mappings on Design Criteria

In studying the properties of a certain subspace (i.e. specific platform architecture instance), it is a worthwhile issue to investigate how the objective values are being changed with modifying the mapping decisions. In this paper, we propose a visualization method for exploring the sensitivity of each subspace

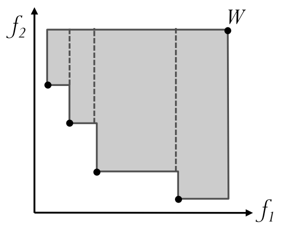


Figure 23: Illustration of the HV metric. The gray region indicates the area in the objective space that is dominated by the Pareto optimal solutions.

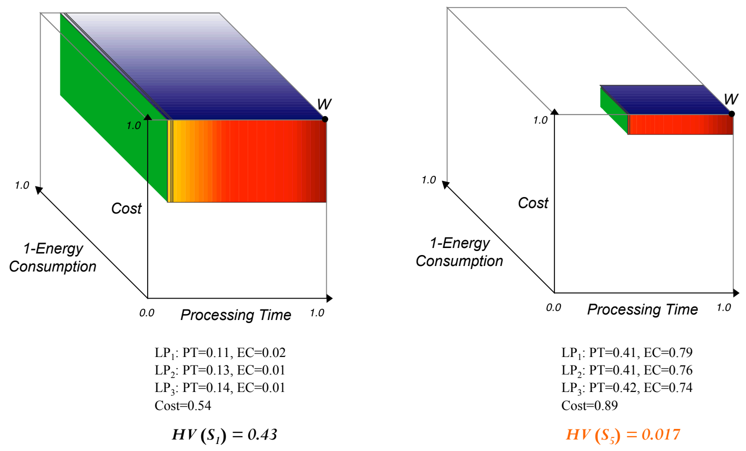
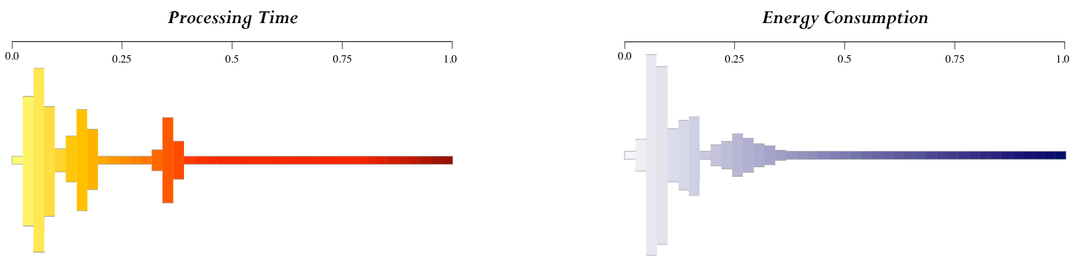


Figure 24: Visualization of the HV metric. Each side of the cube shows one objective. The colored region indicates the dominated area in the objective space.

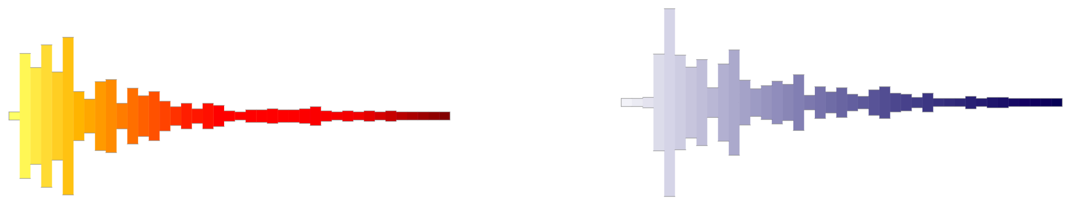
to different mappings on the objective values. To do this, the frequency distribution of objective values in each subspace is visualized to see the range as well as the frequency of objective values that can be achieved with a specific architecture instance by using different mappings. Since all solutions in a subspace have the same architecture cost, the frequency distribution is not applicable for this objective and should only be considered for the other two objectives: processing time and energy consumption.

For each objective, a horizontal axis from 0 to 1 is drawn and colored like the color-coding technique used for showing that objective in the DSE tree. For example, in our case, colors from yellow to red are used for representing the processing time and therefore, this color scheme is used for coloring the corresponding axis in the frequency distribution visualization. The height of each color bar in the (color-coded) objective axis indicates the number of design points with the objective value inside that range. Fig. 25 shows the visualization of the frequency distribution for three subspaces. Fig. 25(a) shows the frequency distribution of solutions in a subspace that contains an ASIP, mP, DFIFO-2 and SRAM. This subspace is indicated as S_2 in Fig. 20. As can be seen in this figure, for all design points, both the processing time and energy consumption is relatively good (less than 0.5). Therefore, for this particular architecture, with different mappings we can get approximately good design points. So if the designer is looking for a system that can flexibly deal with different mappings, this architecture is a good solution. The subspace in Fig. 25(b) consists of an ASIP, mC, mP, DFIFO-2 and SRAM, which is denoted as S_4 in Fig. 20. In this subspace, both the processing time and energy consumption are varying from the best to almost the worst. However, the processing time and energy consumption of most design points are approximately good. Therefore, if the designer is interested in this platform architecture instance, he should take care about the mapping because a wrong mapping decision can make the difference between the best or the worst design point. Fig. 25(c) represents the frequency distribution of subspace S_5 in Fig. 20. This subspace is made up of following components: ASIC-DCT, ASIC-VLE, ASIP, mP, DFIFO-2, DRAM and SRAM. As can be seen in this figure, for all design points, both the processing time and the energy consumption are extremely poor. Even with different mappings, we cannot get an acceptable design point. Thus, this architecture is not a suitable solution for our case study.

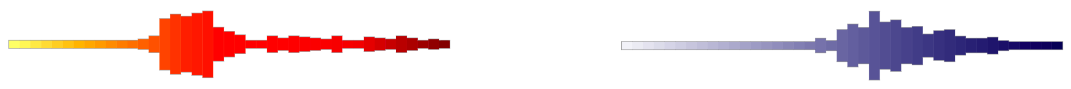
Therefore, by using the frequency distribution visualization, the designer is able to analyze the effect of different mappings on the design criteria for a certain architecture instance. Furthermore, it is easy to compare the sensitivity of different subspaces with respect to the mapping decisions.



(a) Subspace consisting of: ASICP, mP, DFIFO-2, SRAM



(b) Subspace consisting of: ASICP, mC, mP, DFIFO-2, SRAM



(c) Subspace consisting of: ASIC-DCT, ASIC-VLE, ASICP, mP, DFIFO-2, SRAM, DRAM

Figure 25: Visualization of frequency distribution of objective values for three subspaces.

8. Conclusion

VMODEX is an interactive visualization tool, which is developed for visualizing multi-objective design space exploration of embedded systems that are based on heterogeneous multi-processor system-on-chip architectures. It provides insight into the search process of heuristic searching algorithms that are typically used in the DSE process. It helps designers to understand how such algorithms explore the design space during their iterations and converge towards the optimal design points. It should be mentioned that although in this paper we choose the Sesame simulator for evaluating design points and MOEAs for searching the design space, our visualization tool is not restricted to neither evaluating approach nor searching mechanism. It can easily be used for different types of evaluating methods as well as searching strategies.

The tree model that we propose in this paper for visualizing the design space enables us to easily visualize multivariate data. There is no limitation on the number of neither design parameters nor criteria. Furthermore, in our DSE tree model, the concepts of subspaces and local Pareto points are proposed, which are new points of view in the multi-objective DSE process and has not been considered before.

VMODEX also enables very powerful and rapid analysis of DSE results. Designers can look at the data from different perspectives and at multiple levels of abstraction. Several interactive capabilities are provided, which allow designers to play with data and find out some interesting and important features that may not be found just by looking at the static visualization.

To summarize the different visualization approaches that we proposed in this paper, Fig. 26 shows all our proposed visualization methods together with the graphical tricks that we used for visualizing them. Furthermore, in this figure, we clarify how these visualizations are related to each other.

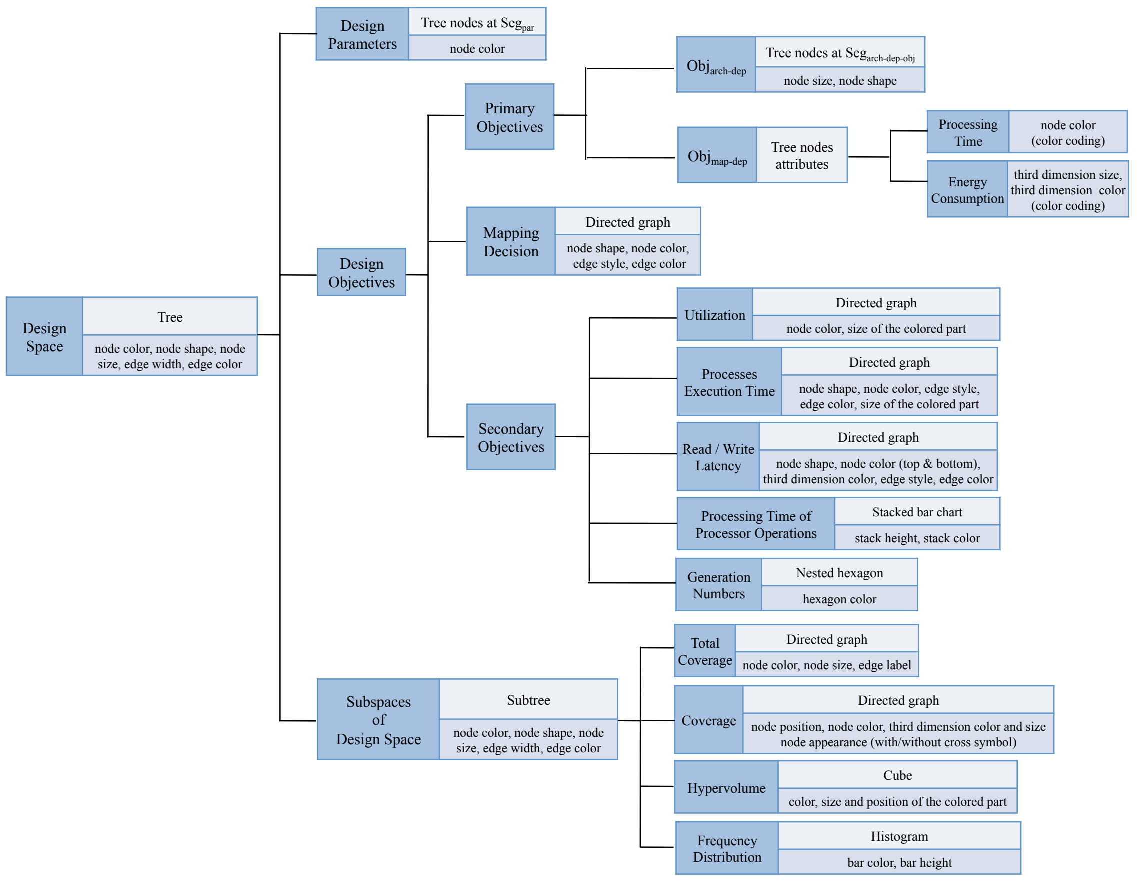

Design Space
Tree
node color, node shape, node size, edge width, edge color
Design Parameters
Tree nodes at Seg_{par}
node color
Design Objectives
Primary Objectives
$Obj_{arch-dep}$
Tree nodes at $Seg_{arch-dep-obj}$
node size, node shape
$Obj_{map-dep}$
Tree nodes attributes
Processing Time
node color (color coding)
Energy Consumption
third dimension size, third dimension color (color coding)
Mapping Decision
Directed graph
node shape, node color, edge style, edge color
Secondary Objectives
Utilization
Directed graph
node color, size of the colored part
Processes Execution Time
Directed graph
node shape, node color, edge style, edge color, size of the colored part
Read / Write Latency
Directed graph
node shape, node color (top & bottom), third dimension color, edge style, edge color
Processing Time of Processor Operations
Stacked bar chart
stack height, stack color
Generation Numbers
Nested hexagon
hexagon color
Subspaces of Design Space
Subtree
node color, node shape, node size, edge width, edge color
Total Coverage
Directed graph
node color, node size, edge label
Coverage
Directed graph
node position, node color, third dimension color and size node appearance (with/without cross symbol)
Hypervolume
Cube
color, size and position of the colored part
Frequency Distribution
Histogram
bar color, bar height

Figure 26: summarization of all our proposed visualization approaches together with their relations with each other

References

[1] M. Gries, Methods for evaluating and covering the design space during early design development, Integr. VLSI J. 38 (2004) 131–183.

[2] C. Erbas, A. D. Pimentel, M. Thompson, S. Polstra, A framework for system-level modeling and simulation of embedded systems architectures, EURASIP J. Embedded Syst. 2007 (2007) 2–2.

[3] A. D. Pimentel, C. Erbas, S. Polstra, A systematic approach to exploring embedded system architectures at multiple abstraction levels, IEEE Trans. Comput. 55 (2006) 99–112.

[4] T. Taghavi, A. D. Pimentel, Visualization of multi-objective design space exploration for embedded systems, in: Digital System Design: Architectures, Methods and Tools, IEEE Computer Society, Lille, France, 2010, pp. 11–20.

[5] T. Taghavi, A. D. Pimentel, An interactive visualization tool for the analysis of multi-objective embedded systems design space exploration, in: Rapid Simulation and Performance Evaluation: Methods and Tools (Rapido), Crete, Greece, 2011.

[6] T. Taghavi, A. D. Pimentel, Techniques and visualization approaches for analyzing local and global pareto optimal sets in multi-objective design space exploration, in: Parallel Programming and Run-Time Management Techniques for Many-core Architectures (PARMA), Lake Como, Italy, 2011.

[7] G. Kahn, The Semantics of a Simple Language for Parallel Programming, in: J. L. Rosenfeld (Ed.), IFIP Congress on Information Processing, New York, NY, USA, pp. 471–475.

[8] K. Deb, Optimization for Engineering Design: Algorithms and Examples, Prentice-Hall,, New Delhi, 1995.

[9] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization, in: K. Giannakoglou, et al. (Eds.), Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, 2002, pp. 95–100.

[10] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, H.-P. Schwefel (Eds.),

Parallel Problem Solving from Nature PPSN VI, volume 1917 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2000, pp. 849–858.

[11] J. D. Knowles, D. W. Corne, Approximating the nondominated front using the pareto archived evolution strategy, Evol. Comput. 8 (2000) 149–172.

[12] N. Gershon, S. G. Eick, S. Card, Information visualization, interactions 5 (1998) 9–15.

[13] E. R. Tufte, The visual display of quantitative information, Graphic Press, 2001.

[14] P. Marwedel, B. Sirocic, Multimedia components for the visualization of dynamic behavior in computer architectures, in: Computer architecture education, ACM, New York, NY, USA, 2003.

[15] C. Yehezkel, W. Yurcik, M. Pearson, D. Armstrong, Three simulator tools for teaching computer architecture: Little man computer, and rtlsim, ACM Journal of Educational Resources in Computing 1 (2001) 60–80.

[16] R. Ibbett, Computer architecture visualisation techniques, Microprocessors and Microsystems 23 (1999) 291–300.

[17] K. Berkbigler, B. Bush, K. Davis, N. Moss, S. Smith, A la carte: A simulation framework for extreme-scale hardware architectures, in: IASTED International Conference on Modeling and Simulation, Palm Springs, CA, 2003.

[18] P. Coe, F. W. Howell, R. Ibbett, L. Williams, A hierarchical computer architecture design and simulation environment, ACM Transactions on Modeling and Computer Simulation 8 (1998) 431–446.

[19] T. A. Diep, J. P. Shen, Vmw: A visualization-based microarchitecture workbench, Computer 28 (1995) 57–64.

[20] C. Stolte, R. Bosch, P. Hanrahan, M. Rosenblum, Visualizing application behavior on superscalar processors, in: IEEE Symposium on Information Visualization, IEEE Computer Society, Washington, DC, USA, 1999, pp. 10–17.

[21] W. Fang, C.-L. Wang, W. Zhu, F. C. M. Lau, Pat: a postmortem object access pattern analysis and visualization tool, in: IEEE International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, USA, 2004, pp. 379–386.

[22] E. van der Deijl, G. Kanbier, O. Temam, E. D. Granston, A cache visualization tool, Computer 30 (1997) 71–78.

[23] Y. Yu, K. Beyls, E. H. D'Hollander, Visualizing the impact of the cache on program execution, in: Fifth International Conference on Information Visualisation, London, England, pp. 336–341.

[24] B. Quaing, J. Tao, W. Karl, Yaco: A user conducted visualization tool for supporting cache optimization, in: High Performance Computing and Communications (HPCC), Sorrento, Italy, pp. 694–703.

[25] Y. Yu, E. H. D'Hollander, Loop parallelization using the 3d iteration space visualizer, Journal of Visual Languages and Computing 12 (2001) 163–181.

[26] H. Hlavacs, D. F. Kvasnicka, C. W. Ueberhuber, Clue-a tool for cluster evaluation, in: Distributed and Parallel Systems, Balatonfuered, Lake Balaton, Hungary,, pp. 61–64.

[27] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, P. Hanrahan, Rivet: a flexible environment for computer systems visualization, SIGGRAPH Comput. Graph. 34 (2000) 68–73.

[28] R. P. Bosch, Using visualization to understand the behavior of computer systems, Ph.D. thesis, Stanford University, Computer Science Dept, Berkeley, CA, 2001.

[29] A. D. Mihalik, Vista: A visualization tool for computer architects, Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2004.

[30] T. Taghavi, M. Thompson, A. D. Pimentel, Visualization of computer architecture simulation data for system-level design space exploration, in: International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 149–160.

[31] H. Pohlheim, Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization, in: Genetic and Evolutionary Computation Conference, Morgan Kaufmann, Los Altos, CA, 1999, pp. 533–540.

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

[33] J. W. Sammon, A nonlinear mapping for data structure analysis, IEEE Transactions on Computers 18 (1969) 401–409.

[34] E. Hart, P. Ross, Gavel - a new tool for genetic algorithm visualization, IEEE Trans. Evolutionary Computation 5 (2001) 335–348.

[35] S. I. Ito, Y. Mitsukura, H. N. Miyamura, T. Saito, M. Fukumi, A visualization of genetic algorithm using the pseudo-color, in: ICONIP (2), Kitakyushu, Japan, pp. 444–452.

[36] modefrontier: the multi-objective optimization and design environment, `http://www.esteco.com/home/mode_frontier/mode_frontier.html`, 2011.

[37] OPTIY : Multidisciplinary Analysis and Optimization, `http://www.optiy.eu/`, 2010.

[38] Grapheur: Business intelligence and interactive visualization tool, `http://grapheur.com/`, 2011.

[39] B. Shneiderman, The eyes have it: A task by data type taxonomy for information visualization, in: IEEE Symposium on Visual Languages, Boulder, CO , USA, pp. 336–343.

[40] B. De Smedt, G. Gielen, Watson: a multi-objective design space exploration tool for analog and rf ic design, in: IEEE Custom Integrated Circuits Conference (CICC), Orlando, USA, 2002, pp. 31–34.

[41] C. Ware, Information Visualization: Perception for Design, Morgan Kaufmann, 1st edition, 2000.

[42] C. Erbas, S. Cerav-erbas, A. D. Pimentel, Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design, IEEE Transactions on Evolutionary Computation 10 (2006) 358–374.

[43] A. D. Pimentel, S. Polstra, F. Terpstra, A. W. v. Halderen, J. E. Coffland, L. O. Hertzberger, Towards efficient design space exploration of heterogeneous embedded media systems, in: Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation - SAMOS, Springer-Verlag, London, UK, UK, 2002, pp. 57–73.

[44] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto evolutionary algorithm, IEEE Transactions on Evolutionary Computation 3 (1999) 257–271.