# User-Centric Design Space Exploration for Heterogeneous Network-on-Chip Platforms

Chen-Ling Chou and Radu Marculescu
*Department of Electrical and Computer Engineering*
*Carnegie Mellon University, USA*
*{chenlinc,radum}@andrew.cmu.edu*

***Abstract* - In this paper, we present a design methodology for automatic platform generation of future heterogeneous systems where communication happens via the Network-on-Chip (NoC) approach. As a novel contribution, we consider explicitly the information about the user experience into a design flow which aims at minimizing the workload variance; this allows the system to better adapt to different types of user needs and workload variations. More specifically, we first collect various user traces from various applications and generate specific clusters using machine learning techniques. For each cluster of such user traces, depending on the architectural parameters extracted from high-level specifications, we propose an optimization method to generate the NoC system architecture. Finally, we validate the user-centric design space exploration using realistic traces and compare it to the traditional NoC design methodology.**

## I. Introduction

Future multiprocessor systems-on-chip (MPSoCs) will likely consist of heterogeneous computational cores, memories, peripherals, *etc.*, all integrated in a single die and connected via complex networks-on-chip (NoCs) [1]. In addition, the shortening time-to-market forces designers to heavily reuse pre-designed modules in the form of intellectual properties (IP) in order to cope with the increasingly complex design space.

With industry shifting to platform-based design, much progress in the traditional design space exploration (DSE) techniques has been made via task-level [3][21], resource-level [4], and even system-level analysis [5]. Overall, these techniques target system optimization with the goal of improving the system performance. However, nowadays and more so for the future, it is clear that performance, while still important, is not anymore the dominant metric for the platform-based design. In other words, "just-enough performance" is often perfectly acceptable and more attention needs to be paid to user experience and energy efficiency issues (*i.e.*, "user experience per unit energy") which are directly related to the system workload variation [7]. Therefore, as opposed to the traditional design flow considering the task-, resource-, or system-level optimization, our proposed methodology targets one level above, namely, the application-/user-level design. By analyzing the user interaction with the system, we are able to provide more robust platforms for applications characterized by high workload variation.

The traditional flow for application-specific MPSoC design follows the *Y-chart* in Fig. 1(a). Given the architecture template

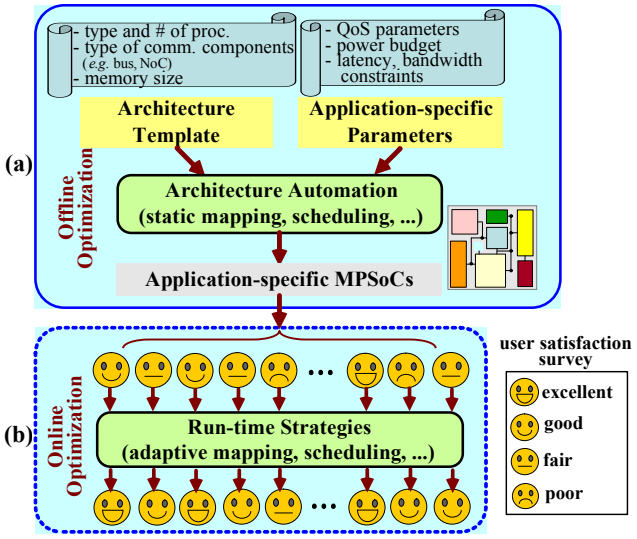


Fig. 1. (a) Traditional design flow for application-specific MPSoC platforms. (b) Online optimization to determine users satisfaction.

(*i.e.*, computation and communication components, network topology, *etc.*) and application-specific parameters (*i.e.*, power constraints, maximum latency, *etc.*), the customized architecture (or system platform) is automatically generated using *static* techniques like task mapping and scheduling; this architecture automation step takes place offline. Afterwards, the system is manufactured and deployed for use by different users as shown in Fig. 1(b). However, due to differences in users behavior, the platform will likely not satisfy all the users equally well. In other words, some users may find the system difficult or inefficient to use even though it may be highly recommended by other users. Such issues are typically the cause for significant losses in product sales and revenues. Therefore, good resource management techniques, such as those based on adaptive mapping and scheduling, are needed to better satisfy a wider range of customer needs [9][10].

Not surprisingly, complex MPSoCs running multiple applications concurrently should rely on a variety of system configurations which are challenging to design. Although prior work for evaluating and covering the design space exists [8], the traditional design flow still can generate only *one* or *a few* platform configurations, the so-called application scenarios [21], belonging to the *same* Pareto curve trading off multiple objectives [13]. Therefore, even assuming perfect techniques for run-time optimization, such

978-3-9810801-5-5/DATE09 © 2009 EDAA

a platform can hardly meet all user needs (Fig. 1(b)). This motivates us to redefine the DSE methodology for future systems by considering an extra degree of freedom, namely, the *user experience*; this encompasses all aspects of end-user interaction with the platform and the associated power/performance costs. Based on this new vision, we aim at designing systems from *user* perspective with the goal of satisfying different types of user needs and design constraints.

Our user-centric design methodology relies on collecting user traces from existing systems or prototypes. To get useful traces for building the next generation systems, we monitor the user behavior independently of the actual platform; that is, the information collected for each user (*i.e.*, user trace) shows what applications are running, at what times, in the system. The novel contributions of our proposed DSE methodology are as follows:

- First, we apply machine learning techniques to cluster the traces from various users such that the differences in user behavior for each class are minimized.
- Then, for each cluster, we propose an offline algorithm for automated architecture generation of heterogeneous NoC platforms that deal explicitly with computation and communication components and satisfy various design constraints, while facing significant workload variations.

We note that by taking the user experience into consideration into the DSE methodology, the generated system platforms exhibit less variation among the users behavior; this implies that each system is highly suitable for a particular user cluster and therefore the overhead of later applying various online optimization techniques can be reduced as well [22][23]. In this paper, however, we restrict ourselves to the offline optimization part of platform generation, while follow up work will consider the run-time optimization aspects.

The remaining of this paper is organized as follows. In Section II, we review some relevant work. Section III discusses the proposed DSE methodology and provides detailed algorithms for the entire design flow. Experimental results are presented in Section IV. Finally, we summarize our contribution in Section V.

## II. Related Work

In an early attempt, Dick and Jha propose a multiobjective genetic search algorithm for co-synthesis of hardware/software embedded systems which trades off price and power consumption [13]. Some design methodologies for automatic generation of architecture for heterogeneous embedded MPSoCs were later studied in [11][12]. Different from the heuristics used to handle a large design space, Ozisikyilmaz *et al.* propose a predictive modeling technique to estimate the system performance by looking at information from past systems [14]. More recently, Shojaei *et al.* propose a BDD-based approach to efficiently obtain Pareto points which help multi-dimensional optimization [15]. Instead of using the bus-based communication, Chatha *et al.* address the automated synthesis of an application-specific NoC architecture with optimized topology [16]. However, their approach targets single application characteristics (*i.e.*, the communication trace graph is fixed) which is not realistic to use for different users. Murali *et al.* consider multiple use-cases during the NoC design process [24]. However, they optimize the NoC using only worst case constraints. In reality, the distribution of use-cases for various users are very different.

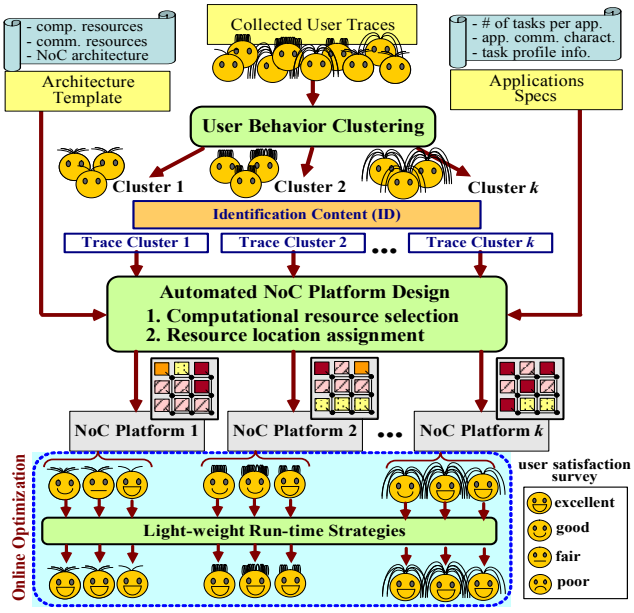


Fig. 2. User-centric design flow for heterogeneous NoCs.

The differences in users behavior have been also considered. For instance, Kang *et al.* in [6] observe the differences between younger and middle-aged adults in the use of complicated electronic devices. Rabaey *et al.* in [7] discuss the wide range of workloads of the future and advocate for new metrics to guide the exploration and optimization of future systems, such as the user functionality, reliability, composability.

## III. NoC Design Space Exploration Methodology Considering User Experience

### A. User-centric Design Flow

The proposed user-centric DSE methodology is shown in Fig. 2. In order to take the user behavior into consideration, the inputs of our design flow are:

- *Architecture template*, which consists of computation resources (*e.g.* FPGA, DSP, ASIC), communication resources (*e.g.* router, FIFO, segmented bus), and the communication protocol (*e.g.* routing/switching scheme). Of note, we focus only on 2-D mesh topology and *XY* routing, but the communication architecture may be more general.
- *Applications specification* which captures the task graph characteristics (*e.g.* number of tasks and communication rate between them), inter-application synchronization, computation profile (*e.g.* power consumption, application deadlines).
- User traces which record the relevant user behavior over time.

The entire user-centric design flow involves a number of critical steps with the goal of generating systems that meet the user needs. Towards this end, we first explore the problem of *clustering* the user traces such that all users belonging to the same cluster have a similar behavior while interacting with the target system (more details are given in Section III.*B*). At the same time, we store the user behavior characteristics, namely the identification content (ID), for testing purposes. During the second step, an NoC platform is automatically generated for the user traces in each cluster (as discussed in Section III.*C*) such that multiple design constraints are satisfied. Last but not least, to show the

potential of the user-centric design flow, a validation process is presented in Section III.*D*. To formulate these problems, some terminology is needed:

- $r_i$ : a resource of type *i*. Assume there exist *n* different types of resources, $r_1$, $r_2$,..., $r_n \in R$, $N(r_i)$ represents the number of resource $r_i$ in the platform, while $W(r_i)$ represents the price of resource $r_i$.
- $q_i$ : an application with a set of tasks. Assume there exist *m* different applications which can run on the platform, *i.e.*, $q_1$, $q_2$,..., $q_m \in$ Q. $G^{q_i} = (T^{q_i}, Y^{q_i})$ represents the task graph of the application $q_i$. $t_j^{q_i} \in T^{q_i}$ represents a task $t_j$ in application $q_i$, while $e_{jk}^{q_i} \in Y^{q_i}$ represents the communication between $t_j^{q_i}$ and $t_k^{q_i}$, and $M(e_{jk}^{q_i})$ represents the communication rate between $t_j^{q_i}$ and $t_k^{q_i}$.
- $\Lambda = (A, \Omega(A))$ characterizes the NoC platform, where $A$ represents a set of resources, and $\Omega(A)$ represents the precise location of each resource $r_i \in A$ (*i.e.* resource mapping). More precisely, $A = (r_1,r_1,..., r_1, r_2,r_2,..., r_2,..., r_n,r_n,..., r_n | N(r_1), N(r_2),..., N(r_n))$ captures the number and type of resources of the NoC platform.
- $E_{comp}(\alpha, \{\beta\})$ represents the computation energy consumption while running $\alpha$ onto $\{\beta\}$, where $\alpha$ can be a task $t_i$, an application $q_i$, or a trace of a set of applications, and $\{\beta\}$ stands for a resource set with one or multiple number of available resources able to run $\alpha$ on an NoC platform. Similarly, $E_{comm}(\alpha, \Omega(\{\beta\}))$ represents the communication energy consumption (based on bit energy model in [17]) of running $\alpha$ onto an NoC platform with the resource set $\{\beta\}$, where the *XY* routing mechanism is applied for data communication in such a platform.
- User trace $\Re_i = \langle \Re_i^t \rangle$ : the collected sequences from user *i*, while logging the system. Each element $\Re_i^t = \{q_1, q_2, ...\}$ represents a set of applications running on the system at discrete time *t*.

### *B. User Behavior Clustering Problem*

In order to generate different platforms that satisfy the user needs, building a model for users behavior is critical. Here, we define some terms specifically for the clustering problem; the steps of user behavior clustering process are shown in Fig. 3.

- *Inter-application similarity*: Two applications requiring similar resources have a higher inter-application similarity.
- *Application-usage similarity*: Two user traces reflecting a similar frequency of application appearance have a higher application-usage similarity.
- *Application resource demand* (*L*): The degree of resource demands for an application. The application $q_i$ which demands a larger number of resource of type $r_n$ has a higher $L_{r_n}^{q_i}$ value.
- *Application sets appearance probability* ($p_v^{\Re_i}$): The probability of observing a set of applications, *v*, in the user trace $\Re_i$.
- *Subset function* (*S*): If *A* is a subset of (or is included in) *B*, then $S(A, B) = 1$; otherwise it is 0.
- *Cluster mapping* (*C*): $C(i) = j$ indicates that *i* has been clustered into the $j^{th}$ group, where $i{:}C(i) = j$ represents all the elements in the $j^{th}$ group.
- *k-MEAN* clustering: An algorithm [18] groups the objects (or data points) based on attributes/features into *k* different groups, where *k* is positive integer. The grouping here is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

**Input**: task graph characteristics of each application $q_i$ $G^{q_i} = (T^{q_i}, E^{q_i})$, and the task-level computing cost, $E_{comp}(t_j^{q_i}, r_k))$

**Output**: user behavior cluster *S*

- *Step 1*: Derive the Pareto curve trading off the resources and computation power consumption for each application $q_i$ (similar to the solution proposed in [13]). Each Pareto point $E_{comp}(q_i, N(r_1), N(r_2), \ldots, N(r_n))$ gives the minimum power consumption for application $q_i$.
- *Step 2*: Given all Pareto points, calculate $L^{q_i} = \left(L_{r_1}^{q_i}, L_{r_2}^{q_i}, \ldots, L_{r_n}^{q_i}\right)$, *i.e.* the resource demand $L_{r_j}^{q_i}$ for application $q_i$ to each resource type $r_j$ for $j = 1,..., n$, where $L_{r_j}^{q_i} = \sum_{x=1}^{\max(N(r_j))} \{avg[E_{comp}(q_i, N(r_1),..., N(r_j) = x-1,..., N(r_n)) - avg[E_{comp}(q_i, N(r_1),..., N(r_j) = x,..., N(r_n))]\}$
- *Step 3*: Normalize $L^{q_i}$ for each application $q_i$. $\bar{L}^{q_i} = \left(\bar{L}_{r_1}^{q_i}, \bar{L}_{r_2}^{q_i}, \ldots, \bar{L}_{r_n}^{q_i}\right) = \left(L_{r_1}^{q_i} - avg(L^{q_i}), \ldots, L_{r_n}^{q_i} - avg(L^{q_i})\right)$ where $avg(L^{q_i}) = \left(L_{r_1}^{q_i} + L_{r_2}^{q_i} + \ldots + L_{r_n}^{q_i}\right)/n$
- *Step 4*: Set each application $\bar{L}^{q_i}$ as a data point $d_i$ and apply *k-MEAN* clustering to group all data points $d_i$ into *z* clusters. Assign the center of each cluster, $\mu_r$, where $r=1$, ..., *z*, to the identification content ($ID_r$) which will be utilized in the testing stage and define an *z*-dimensional application vector $V = (v_1,v_2,...,v_z) = (d_i{:}C(d_i)=1,\ d_i{:}C(d_i)=2,...,\ d_i{:}C(d_i) = z)$ capturing the applications within the corresponding cluster.
- *Step 5*: Calculate $p_v^{\Re_i} = \left(p_{v_1}^{\Re_i}, p_{v_2}^{\Re_i}, \ldots, p_{v_z}^{\Re_i}\right)$ for each user trace $\Re_i$, *i.e.* the application sets appearance probability with corresponding application set $v_i$ for $i = 1,..., z$, where $p_{v_j}^{\Re_i} = \frac{\sum_{\forall \langle \Re_i^t \rangle \in \Re_i} \sum_{\forall \{q_k, q_l\} \in \langle \Re_i^t \rangle} S(\{q_k, q_l\}, v_j)}{\left| p_v^{\Re_i} \right|}$
- *Step 6*: Set $p_v^{\Re_i}$ from each user trace as a data point $d_i$ and apply *k-MEAN* clustering to group data points $d_i$ into *k* clusters. Assign the center of each cluster, $\mu_{r'}$, where $r'=1$, ..., *k*, to the identification content ($ID_{r'}$) and generate a *k*-dimensional trace cluster vector $S = (S_1,\ S_2,...,S_k) = (d_i{:}C(d_i)=1,\ d_i{:}C(d_i)=2,...,\ d_i{:}C(d_i) = k)$, where $S_i$ is a group of user traces with high correlation application-usage similarity.
- *Step 7*: Assign $\mu_{r'}$ to the identification content ($ID_{r'}$). Then, generate a *k*-dimensional trace cluster vector $S = (S_1, S_2,...,S_k) = (d_i{:}C(d_i)=1,\ d_i{:}C(d_i)=2,...,\ d_i{:}C(d_i) = k)$ capturing the user traces within the corresponding cluster.

Fig. 3. Main steps of user behavior clustering.

As shown in Fig. 3, the clustering is achieved by first grouping together similar applications (*i.e.* applications in the same cluster $v_i$ have a high inter-application similarity, see *Steps 1-4*), and then clustering the traces that use these application groups in a similar way (*i.e.* traces in the same cluster $S_i$ have a high application-usage similarity, see *Steps 5-7*).

### *C. NoC Platform Automation Problem*

From the algorithm in Section III.*B*, we obtain a set of user traces which have a similar interaction with the system. Here, for each cluster of traces, our goal is to generate a suitable NoC platform, while minimizing the given design constraints. In this paper, an NoC platform describes a number of resources connected using a mesh-like on-chip network. Therefore, the design automation process of our NoC platform involves two critical

steps: *i*) Computational resource selection, which decides the number and the type of resources needed to build the platform, and *ii*) Resource location assignment, which provides the tile location for each resource in the 2-D tile-based NoC. The problem formulations and solutions of these steps are described in Section III.*C.1* and Section III.*C.2*, respectively.

Of note, while running a user trace on platform *A*, it can be observed that applications enter and leave the system dynamically. Here, we apply the greedy approach for task mapping problem; that is, assign tasks $t_i$ to the currently available resources $r_j$, while optimizing the design metrics of interest (*e.g.*, minimal computation energy consumption). The task mapping function is denoted as *map*( ), *i.e.*, $map(t_i) = r_j$.

### *C.1. Computational resource selection*

The resource selection problem is formulated as follows:

**Given** all user traces in a cluster *S*, *i.e.*, $\forall \Re_i = \langle \Re_i^t \rangle \in S$ and the price constraint $\Phi$.

**Find** a resource set *A* which

$$\min \left\{ \sum_{\forall \Re_i \in S} E_{comp}(\Re_i, A) \right\}, \textbf{ such that}: \sum_{\forall r_i \in A} W(r_i) \le \Phi$$

The steps for the computational resource selection problem are summarized in Fig. 4.

*Step 1*: initialize platform *A* with unlimited resources, find $A^{(0)}$ which $\min \left\{ \sum_{\forall \Re_i \in S} E_{comp}(\Re_i, A^{(0)}) \right\}$ and $j \leftarrow 0$.

*Step 2*: **While** $\left( \sum_{\forall r_i \in A^{(t)}} W(r_i) = \Theta^{(j)} > \Phi \right)$

*find* $A^{(j+1)} \leftarrow (..., N(r_x)\text{-}1, ..., N(r_y)\text{+}1, ...)$ (*i.e.* replace $r_x$ with $r_y$, where $W(r_y) < W(r_x)$), which:

$$min \left\{ \frac{\sum_{\forall \Re_i \in S} E_{comp}(\Re_i, A^{(j+1)}) - \sum_{\forall \Re_i \in S} E_{comp}(\Re_i, A^{(j)})}{\sum_{\forall r_i \in A^{(j)}} W(r_i) - \sum_{\forall r_i \in A^{(j+1)}} W(r_i)} \right\}$$

and $j \leftarrow j+1$

**End While;** *return* $A^{(j)}$.

Fig. 4. Computational resource selection.

We start out with a resource set, while minimizing our objective without considering the price constraint (*Step 1*). The price constraints can later be met by replacing the more expensive resources with the cheaper ones. Since there are at most $n(n\text{-}1)$ pairs of possible replacements for a platform with $n$ types of resources, $n(n\text{-}1)$ evaluations are performed. Then, the replacement that results in largest price reduction and smallest computation energy consumption overhead is updated (*Step 2*). We continue this step until the price of the updated resource set satisfies the price constraint.

### *C.2. Resource location assignment*

After obtaining the number and type of computational resources from Section III.*C.1*, our task here is to allocate each resource to the tile-based NoC platform with the goal of minimizing the *communication* energy consumption when *all* user traces belonging to a certain cluster are running in the system. The resource location assignment problem is formulated as follows:

**Given** all user traces in a cluster *S*, *i.e.*, $\forall \Re_i = \langle \Re_i^t \rangle \in S$ and a $W \times H$ 2-D tile-based NoC with a resource set *A* that satisfies $\forall r_i \in A$, $\sum_{i=1}^{n} N(r_i) \le (W \times H)$.

**Find** a one-to-one resource location assignment $\Omega(\ )$ from any resource $r_i$ in *A* to a specific tile location, $\Omega(r_i)=(x_i, y_i)$, which

$$\min \left\{ \sum_{\forall \Re_i \in S} E_{comm}(\Re_i, \Omega(A)) \right\}$$

**such that**: $1 \le x_i \le W$, $1 \le y_i \le H$.

To solve this problem, we need the following notation:

- $B(x_i, y_i)$: The neighbors of tile $(x_i, y_i)$, *i.e.*, $(x_i\text{+}1, y_i)$, $(x_i, y_i\text{+}1)$, $(x_i\text{-}1, y_i)$, $(x_i, y_i\text{-}1)$, where $1 \le x_i\text{+}1, x_i\text{-}1 \le W$ and $1 \le y_i\text{+}1, y_i\text{-}1 \le H$.
- Empty/Filled tile: The tile $(x_i, y_i)$ without/with a computational resource $r_i$ already assigned to it.
- Transmission matrix $\psi$: Each entry $\psi_{uv}$ stores the aggregate communication rate from resource $r_u$ to $r_v$.

The steps for the resource location assignment problem are summarized in Fig. 5.

*Step 1*: generate an $n \times n$ transmission matrix $\psi$ with each entry

$$\psi(u, v) = \psi_{uv} = \sum_{\forall \Re_i \in S} \sum_{\forall \langle \Re_i^t \rangle \in \Re_i} \sum_{\forall q_i \in \langle \Re_i^t \rangle} \sum_{\forall e_{ik}^{q_i} \in Y^{q_i}} W(e_{jk}^{q_i})$$

where $map(t_j) = r_u$ and $map(t_k) = r_v$

*Step 2*: normalize the transition matrix $\psi$, *i.e.*,

$$i.e., \psi_{uv} \leftarrow \psi_{uv} \Big/ \left( \sum_{v=1}^{n} \psi_{uv} \right)$$

*Step 3*: get $u$ with largest $\psi_{uv}$ or $\psi_{vu}$ value, then set the location of $r_u$, *i.e.* $\Omega(r_u) = (x_u, y_u)$, to the center of the platform.

*Step 4*: decide $B(x_u, y_u)$ such that $r_i$ with greater $\psi_{ui}$ has a higher possibility to be assigned to $B(x_u, y_u)$.

$$i.e., \psi_{u1} : \psi_{u2} : ... : \psi_{un} \approx N(r_1) : N(r_2) : ... : N(r_n),$$

where $r_1, r_2, ..., r_n \in B(x_u, y_u)$

*Step 5*: get a filled tile $(x_u, y_u)$ with the greatest empty neighboring tiles.

*Step 6*: repeat *Steps 4* and *5*, until all resources get assigned to the corresponding tile locations in the NoC platform.

Fig. 5. Resource location assignment.

We start out calculating and normalizing the transmission matrix $\psi$ (*Steps 1-2*). Then, by allocating two resources, $r_u$ and $r_v$, with highest $\psi_{uv}$ values as close as possible, we are able to minimize the communication energy consumption while running applications onto the system (*Steps 3-5*). More precisely, the neighboring resources of $r_u$ are assigned based on the ratio $\psi_{ui}$, for $i = 1, ..., n$, as shown in *Step 4*.

### *D. Design Flow Validation*

Here, we validate the potential and robustness of our user-centric design flow (see Fig. 6). Generally speaking, we are given the training and the testing datasets. The training dataset is used to generate platforms under the user-centric design flow, while the testing dataset is used to determine whether or not this design flow produces robust platforms for different types of users. Theoretically, the user traces (see Fig. 2) observed from the former platform, $D_{before}$, can be set as the training dataset to generate the new generation platform. Then, we should take the user traces running on the new platform, $D_{after}$, as the testing dataset to validate the design flow. However, in practice, we cannot have the

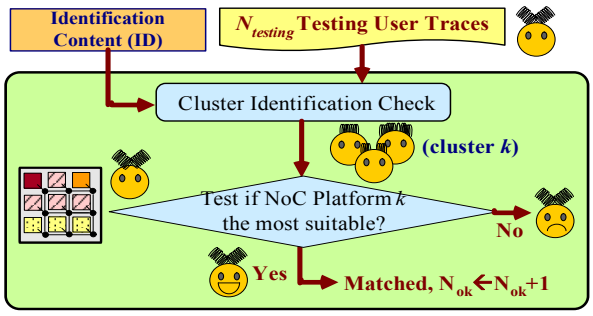


Fig. 6. Design flow of the validation process.

latter traces, $D_{after}$, in advance. Therefore, if we have a reasonable amount of dataset $D_{before}$, then this is usually split into two parts, namely the training and testing datasets, that are used to build and evaluate the design flow. If we have too little data, then the bootstrap method [18] is used for generating enough data.

As seen in Fig. 6, we are given the user traces in the testing dataset with size $N_{testing}$. For each user trace $\Re_i$, we do the cluster identification check. More precisely, with the information of the identification content (ID) obtained from the training process (see Fig. 2 and *Steps 4* and *7*), we report which cluster the user trace $\Re_i$ belongs to; that is, $\Re_i$ has higher inter-application and application-usage similarities with other traces belonging to the same cluster (say cluster $k$). Ideally, the user trace which is identified to be in the $k^{th}$ cluster during the testing stage should report the best performance while executed on NoC platform $k$ generated from the training stage. Therefore, to validate the accuracy of our user-centric design flow, we evaluate whether or not the NoC platform $n = (A, \Omega(A))$ is the most suitable platform for user $i$, *i.e.*, the total energy consumption of running the user trace $\Re_i$ on it, $\sum_{\forall \Re_i \in S} [E_{comp}(\Re_i, A) + E_{comm}(\Re_i, \Omega(A))]$, is smaller than *all* other generated platforms. If yes, we denote it as a match. Finally, the accuracy rate for our user-centric design flow, *i.e.*, $(N_{ok}/N_{testing})\times 100\%$, is reported.

## IV. Experimental Results

To evaluate the user behavior model and the associated design flow, we apply our proposed methodology to real applications with realistic user traces. Our environment and design inputs are as follows:

- Five different types of computational resources $r_i$ are available in the architecture template; the corresponding processor model and its price (in U.S. dollars), $M(r_i)$ are listed in Table I.
- Seven applications are executed on the system platform, including two synthetic applications generated by TGFF package [19], *Automotive/Industrial*, *Consumer*, *Networking*, *Office automation*, and *Telecom* from the embedded system benchmark suite (E3S) [20]. Some pre-processing (such as task biding, scheduling) is done for these seven applications where task graphs with task size being 7, 7, 8, 6, 5, 4, and 6, respectively. Each task is going to execute on one resource later. In addition, the task profile, the power consumption of running task $t_i$ on each processor type, are analyzed beforehand under specified performance constraints.
- Hundreds of user traces (*i.e.*, both training and testing datasets) are used to validate the accuracy of the design flow. We use realistic traces collecting the behavior of the Windows XP environment from twenty users; the bootstrap method [18] is later applied to generate even more traces. The length of each user trace is set to 500.

TABLE I: Architecture Template for the NoC Platform.

| *Resource Type*, $r_i$ | *Part Number* | *Price*, $M(r_i)$ |
|---|---|---|
| $r_1$: DSP 300MHz | TI TMS320C6203 | 112 |
| $r_2$: RISC 266MHz | IBM PowerPC 405GP | 65 |
| $r_3$: DSP 60MHz | Analog Devices 21065L | 10 |
| $r_4$: x86 $\mu$processor 400MHz | AMD K6-2E | 77 |
| $r_5$: $\mu$controller 133MHz | AMD ElanSC520 | 33 |

Assume that, due to their incompatibility, at most four applications can execute on the platform at the same time. In addition, from market survey or previous design experience, our goal is to generate three different platforms (*i.e.*, parameter $k$ is set to 3) in order to satisfy different types of users. And the price constraint for each platform is set to 1500 (*i.e.*, $\Phi = 1500$).

### A. Evaluation of User Behavior Clustering

The clustering of user behavior is the most critical step in this design flow. Indeed, if the user traces in the same cluster have a high variance in terms of the resource requirements, the corresponding platform may not fit well most users in this cluster.

Fig. 7 shows the clustering results. All feasible Pareto points are derived trading off the price of the platform and the computation energy consumption. We randomly select two users in each trace cluster and plot the corresponding Pareto curves. As shown in Fig. 7, the variation of users within the same cluster is quite small. We also come out with three resource sets ($A_1$, $A_2$, and $A_3$) for these three trace clusters, while meeting the price constraint ($\Phi = 1500$). For example, for cluster 1, the resource set $A_1$ consists of 3 resources $r_1$, 6 $r_2$, 6 $r_3$, 6 $r_4$, and 7 $r_5$, with the total price being equal to 1479. As seen, these three resource sets ($A_1$, $A_2$, and $A_3$) are quite different although their prices are close to 1500.

Table II shows the normalized computation energy consumption of applying these three resource sets to each user trace cluster. For example, for the second entry in second column, the value 1.22 gives the computation energy consumption ratio of running all traces in cluster 1 onto $A_1$ and $A_2$; that is,

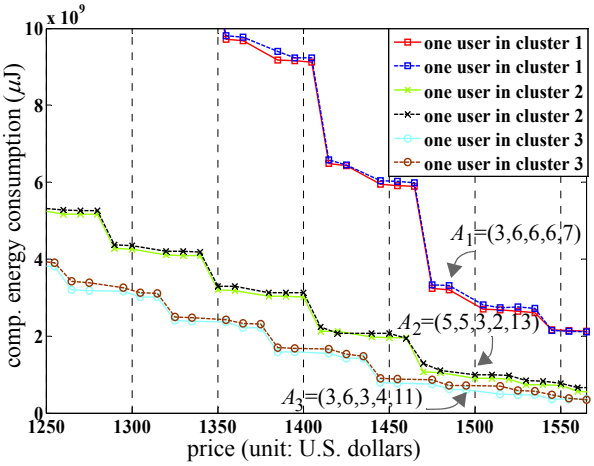


Fig. 7. Pareto points showing the tradeoffs between price and computation energy consumption. For each cluster, two users are randomly selected and their Pareto curves are plotted.

$$\frac{\sum_{\forall \Re_i \in S_1} E_{comp}(\Re_i, A_1)}{\sum_{\forall \Re_i \in S_1} E_{comp}(\Re_i, A_1)} : \frac{\sum_{\forall \Re_i \in S_1} E_{comp}(\Re_i, A_2)}{\sum_{\forall \Re_i \in S_1} E_{comp}(\Re_i, A_1)} = 1 : 1.22$$

Of note, from Fig. 7 and Table II, we can conclude that even if the workload variation from different types of users is high, our user behavior clustering process can separate them quite effectively.

TABLE II: COMPUTATION ENERGY CONSUMPTION COMPARISON FOR THREE TRACE CLUSTERS AND DIFFERENT RESOURCE SETS DERIVED BY THE PROPOSED AND TRADITIONAL DESIGN FLOW.

| Resources \ Traces | Cluster 1 | Cluster 2 | Cluster 3 |
|---|---|---|---|
| Set $A_1$ | 1 | 1.47 | 1.35 |
| Set $A_2$ | 1.22 | 1 | 1.33 |
| Set $A_3$ | 1.33 | 1.28 | 1 |
| Set $A'$ | 1.50 | 1.18 | 1.31 |

Finally, we compare our proposed methodology against the traditional design flow which generates only one platform, $A'$ (see the last row of Table II), while optimizing the computation energy consumption for the entire set of user traces, given the price constraint $\Phi = 1500$. As observed, we achieve about 30% computation energy consumption savings, on average, compared to the unique platform, $A'$, derived from the traditional design flow.

### B. Evaluation of NoC platform

We first evaluate the solution quality of the computational resource selection algorithm in Section III.*C.1* for user traces in the training dataset ($D_{before}$), against the best solution which can be derived from the Pareto curve in Fig. 7. The experiments are performed on an AMD Athlon™ 64 Processor 3000+ running at 2.04GHz. Compared to the optimal solution, our method consumes 5% more energy in computation, on average, for all these three clusters. However, it requires more than 10 hours to get the optimal resource sets for one cluster while our algorithm takes only about 10 minutes.

Next, we evaluate the solution quality of the resource location assignment algorithm (Section III.*C.2*) against the optimal solution, given a fixed set of resources running the user traces. We observe that our method consumes 7% more energy in communication, on average, compared to the optimal resource assignment.

### C. Evaluation of Entire Design Methodology

Finally, we apply the validation process in Fig. 6 to show the potential of the user-centric design methodology. The size of training dataset ranges from 100 to 700, while the size of the testing dataset is fixed to 500. We observe that the accuracy rate, $(N_{ok}/N_{testing})\times 100\%$, increases as the size of the training data increases (for training dataset size of 100, 300 and 500, the accuracy rate is 73%, 84%, and 87%, respectively). By applying 700 training data for building these three platforms, we can have more information for user behavior clustering and therefore, come up with a higher accuracy rate (around 90%) at the end.

## V. CONCLUSION

In this paper, we have addressed the user-centric design problem for heterogeneous NoC platforms. Efficient algorithms have been proposed for clustering the users behavior and automatically generate 2-D NoC platforms such that the values of the total computation and communication energy consumption are minimized. We have also proposed a validation process for the proposed user-centric design flow.

Our experimental results using real applications show that by designing a system from *users* perspective, we are able to minimize the workload variance; this allows the system to better adapt to different types of user needs and workload variations. Future work will consider the run-time optimization aspects for these newly generated NoC platforms.

## ACKNOWLEDGEMENTS

The authors acknowledge support from NSF via grant CCF-0702420 and SRC via grant 2008-HJ-1823.

## REFERENCES

[1] L. Benini, G. De Micheli, "Networks on chip: a new paradigm for systems on chip design," Proc. DATE 2002, pp. 418-419.
[2] K. Richter, *et al.*, "Bottom-up performance analysis of HW/SW platforms,". Proc. of the IFIP 17th World Computer Congress - Tc10 Stream on Distributed and Parallel Embedded Systems: Design and Analysis of Distributed Embedded Systems, pp. 173-183, 2002.
[3] R. Alur, D. L. Dill, "A theory of timed automata," Theoretical Computer Science, 1994, vol. 126, pp. 183-235.
[4] L. Sha, R. Rajkumar, and S. S. Sathaye, "Generalized rate-monotonic scheduling theory: a framework for developing real-time systems," Proc. of the IEEE, vol.82, no.1, pp.68-82, Jan. 1994.
[5] P. Pop, P. Eles, Z. Peng, "Bus access optimization for distributed embedded systems based on schedulability analysis," Proc. DATE, pp. 567-575, 2000.
[6] N. E. Kang, W. Yoon, "Age- and experience-related user behavior differences in the use of complicated electronic devices," Int. J. Hum.-Comput. Stud. 66, 6 2008, pp 425-437.
[7] J. M. Rabaey, D. Burke, K. Lutz, J. Wawrzynek, "Workloads of the Future," IEEE Design and Test of Computers, vol. 25, no. 4, pp. 358-365, July-August, 2008.
[8] M. Gries, "Methods for evaluating and covering the design space during early design development," Integr. VLSI Journal, 2004, pp. 131-183.
[9] D.-I. Kang, J. Suh, J. O. McMahon, S. P. Crago, "Preliminary study toward intelligent run-time resource management techniques for large tiled multi-core architectures," Workshop of High Performance Embedded Computing, Sept. 2007.
[10] C.-L. Chou, R. Marculescu, "User-aware dynamic task allocation in networks-on-chip,". Proc. DATE 2008, pp. 1232-1237.
[11] A. Baghdadi, *et al*, "An efficient architecture model for systematic design of application-specific multiprocessor SoC," Proc. DATE 2001, pp. 66-62.
[12] D. Lyonnard, *et al.*, "Automatic generation of application-specific architectures for heterogeneous multiprocessor system-on-chip," Proc. DAC, 2002, pp. 518-523.
[13] R. P. Dick, N. K. Jha, "MOGAC: A multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems," IEEE T-CAD, 1998, pp. 920-935.
[14] B. Ozisikyilmaz, G. Memik, A. Choudhary, "Efficient system design space exploration using machine learning techniques," Proc. DAC 2008, pp. 966-969.
[15] H. Shojaei, *et al.*, "SPaC: A symbolic pareto calculator," Proc. CODES+ISSS 2008.
[16] K.S. Chathak, K. Srinivasan, G. Konjevod, "Automated techniques for synthesis of application-specific network-on-chip architectures" IEEE T-CAD, Aug. 2008, pp. 1425-1438.
[17] T. T. Ye, L. Benini, G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," Proc. DAC 2002, pp. 524-529.
[18] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), 2006.
[19] Task graphs for free (TGFF v3.0) Keith Vallerio, 2003. http://ziyang.ece.northwestern.edu/tgff/.
[20] R. Dick, "Embedded system synthesis benchmarks suites (E3S)," http://www.ece.northwestern.edu/~dickrp/e3s.
[21] S. V. Gheorghita, T. Basten, H. Corporaal, "Application scenarios in streaming-oriented embedded-system design," Design & Test of Computers, IEEE, vol.25, no.6, pp.581-589, Nov.-Dec. 2008.
[22] A. Shye, *et al.*,"Power to the People: Leveraging Human Physiological Traits to Control Microprocessor Frequency," Proc. IEEE/ACM MICRO, Nov. 2008.
[23] A. Shye, *et al.*,"Learning and Leveraging the Relationship between Architecture-Level Measurements and Individual User Satisfaction" Proc. ISCA, June 2008.
[24] S. Murali, *et. al.*,"Mapping and configuration methods for multi-use-case networks on chips" Proc. ASPDAC, pp. 24-27, Jan. 2006.