# Multi-ASIP Platform Synthesis for Event-Triggered Applications with Cost/Performance Trade-offs

Deepak Gangadharan, Laura Micconi, Paul Pop, Jan Madsen
Embedded Systems Engineering, DTU Compute, Technical University of Denmark
E-mail:{dega,lmic,paupo,jama}@dtu.dk

***Abstract*—In this paper, we propose a technique to synthesize a cost-efficient distributed platform consisting of multiple Application Specific Instruction Set Processors (multi-ASIPs) running applications with strict timing constraints. Multi-ASIP platform synthesis is a non-trivial task for two reasons. Firstly, we need to know the WCET of tasks in target applications to derive platforms (including synthesized ASIPs) in which the tasks are schedulable. However, the WCET of tasks can be known only after the ASIPs are synthesized. We break this circular dependency by using a probability distribution of the WCET of a task (further referred to as the WCET uncertainty model), which takes into account the underlying microarchitectural configurations for the ASIP implementation. Secondly, the datapath area of the multi-ASIPs synthesized is an important design factor that contributes significantly towards the overall cost of the platform. We propose an area estimation model and a WCET uncertainty model that consider the effect of *task datapath similarity*. Based on these two models, we support the designer in exploring cost/performance trade-offs during the platform synthesis. We propose an Evolutionary Algorithm-based approach to solve this multiobjective optimization problem. The proposed approach has been evaluated using several benchmarks and it provides a number of multi-ASIP platform solutions exploring the trade-offs in the cost/performance design space.**

## I. Introduction

Current distributed embedded platforms use heteregenous processing elements (PEs) to run various applications from the automotive, multimedia and networking domains. These platforms are increasingly employing Application Specific Instruction Set Processors (ASIPs) to provide a high degree of flexibility and good performance. We call the set of tasks for which the ASIP is synthesized and tuned as *task cluster*. During ASIP synthesis, the processor microarchitectural parameters such as number and data widths of registers and memory blocks and number of functional units (considering task-level parallelism) are tuned for the task cluster. These parameters influence the WCET of the tasks clustered on an ASIP as well as the area of the datapath (i.e., the portion that performs computations) of the synthesized ASIP. However, as the design space of the microarchitectural parameters is considerably large, it is infeasible to evaluate all possible task clustering solutions considering the microarchitectural design space in order to synthesize the optimal platform solution that minimizes cost and provides the best possible performance. In this paper, we consider *datapath area* as *cost* and *probability of schedulability* of target real-time applications as *performance* of the synthesized platform solution.

In this paper, multi-ASIP platform synthesis derives platform solutions consisting of the number of ASIPs required along with tasks clustered on each ASIP such that there is simultaneous minimization of platform cost (based on minimizing ASIP datapath area) and maximization of the probability of schedulability of applications. This is not straightforward as the underlying microarchitectural design space is large. Moreover, in order to derive platform solutions that ensure that the tasks in the applications are schedulable under strict timing constraints, the WCET of the tasks must be known. However, the WCET of a task is known only after the ASIPs are synthesized (i.e., after the task cluster on the ASIP is decided) as WCET depends on the microarchitectural configurations and datapath of the ASIP. It is necessary to break this *circular dependency* in order to derive good platform solutions while considering the large microarchitectural design space. To add to the above difficulty, the *task similarities* also need to be considered to derive good task clusters and therefore good platform solutions that minimize cost. Two tasks are said to have similarities if some of the operations constituting the tasks are identical and datapath resources implementing them can be shared. This will be further explained in Section III.

In the context of platform synthesis methods, we group the existing research efforts into the following categories.
**1)** Firstly, there are platform synthesis approaches that do not consider ASIPs. There is a large body of work in this category [7], [10], [15], [16] and the assumption is that the details of each component are known.
**2)** There are platform synthesis approaches which consider multiple ASIPs. Some of these approaches [3], [14] assume that the ASIPs have been synthesized, whereas in [18], a small set of microarchitectural configurations is considered and an ASIP is synthesized for each considered microarchitecture before deriving the platform. Hence, these approaches severely limit the design space, disregarding potentially very good solutions because they do not take into account the ASIP microarchitecture design space during platform synthesis.
**3)** Orthogonal to our problem. there has been significant amount of work done in the domain of custom instruction selection and synthesis. An Integer Linear Programming (ILP) formulation and a heuristic approach is proposed in [12] to derive instruction-set extensions that reduce the WCET of a task.
**4)** To the best of our knowledge, there is no work on platform synthesis with multiple ASIPs, where the ASIPs are not synthesized beforehand. Consequently, there have been no works in this category where multiple objectives of area cost and schedulability of target applications are considered for optimization.

In this paper, we address the circular dependency using a WCET uncertainty model that considers all possible ASIP microarchitectural configurations. Our uncertainty model also accounts for similarities among tasks in a task cluster. We propose a platform synthesis approach that uses the WCET

978-1-4799-0850-9/13/$31.00 ©2013 IEEE

uncertainty model to derive a platform including multiple ASIPs. Our proposed approach uses an Evolutionary Algorithm (EA) to obtain a *Pareto-front* of solutions that includes solutions optimized for both cost and schedulability of target applications.

There are two approaches for handling real-time applications namely *Event-Triggered* (ET) and *Time-Triggered* (TT) approaches [9]. In the ET approach, all activities are initiated whenever a significant event occurs. On the other hand, all activities in the TT approach are initiated at a predetermined point in time. In this work, we consider applications consisting of tasks that are event triggered. ET systems require a dynamic scheduling strategy where an appropriate task is initiated in response to an event. Here, we use *fixed-priority preemptive scheduling* (fpps) to schedule tasks in the applications.

We exploit task similarities to synthesize cost-efficient platform solutions by leveraging resource sharing techniques employed in the synthesis of Custom Instruction Set Extensions (ISEs) [19], [20], where resources for similar operations in two tasks are shared while the datapath is designed. More specifically, as a result of this resource sharing, the ASIP datapath area is minimized by clustering tasks which have higher similarity in datapath operations. The tasks are represented as Data Flow Graphs (DFGs). We assume that graph merging techniques as described in [19], [20] will be employed to merge similar portions of the task DFGs in order to design a cost-efficient ASIP datapath.

The paper is organized as follows. In Section II, we present the System Model and introduce the WCET uncertainty model without including the effects of task similarities. Then, we define the problem and explain how task similarities have been accounted for in our area estimation technique and WCET uncertainty model (Section III). The motivational example, which highlights the advantages of our multi-ASIP platform synthesis method is presented in Section IV. In Section V, we discuss the genetic algorithm based approach, which derives the *Pareto-front* of solutions. The experimental results for a set of benchmark applications are presented in Section VI and our conclusions are drawn in Section VII.

## II. System Model

The system consists of multiple target applications and the underlying hardware that consists of PEs. Each target application is modeled as a task graph $\mathcal{A} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ represents the set of vertices of the task graph or the set of tasks $\{\tau_i\}$ (where $i \leq L$ and $L$ is the total number of tasks in an application) and $\mathcal{E}$ represents the set of edges that represent the communication among the tasks. The PEs are denoted by $PE_j$, where $PE_j$ is the $j$-th PE. The PEs may be composed of ASIPs and other PEs, such as General Purpose Processors (GPPs) and Digital Signal Processors (DSPs). The architectural details and timing behavior of GPPs and DSPs are well defined. Although we consider only ASIPs in the rest of the paper, our method can handle the inclusion of GPPs and DSPs.

Tasks are grouped into clusters, e.g., a task $\tau_i$ is grouped into a cluster $S_j$, where $S_j$ is the task cluster on the processing element $PE_j$. Each task $\tau_i$ has a period $T_i$. When there are multiple target applications, we aggregate the tasks in all the applications into one global task set. Consequently, for all the target applications, we try to meet a global deadline (which is explained later in Section III-C. The area required for implementation of each task $\tau_i$ on an ASIP is denoted by $a_i$.

### A. Modeling WCET uncertainty

As mentioned in Section I, there is a circular dependency between the schedulability of a platform solution and the WCET of a task $C_i$. This arises due to the existence of a large number of microarchitectural configurations, which cannot be practically evaluated during platform synthesis. However, in order to obtain good solutions, it is necessary to consider a large number of microarchitectural configurations during platform synthesis. Our approach to break the circular dependency is to model the WCET $C_i$ as a probability distribution.

The variability of WCET $C_i$ of a task $\tau_i$ is due to the large number of possibilities in ASIP implementation on which task $\tau_i$ will run, and does not reflect the variation in execution time, which is due to variation in the input data and modern architectural features such as branch prediction. The final implementation of the ASIP running $\tau_i$ will only be available after the time-consuming ASIP microarchitecture synthesis. We use the probability distribution of $C_i$ during design space exploration (DSE) in order to avoid synthesizing every ASIP microarchitecture corresponding to changes in task clustering.

The designer can capture the probability distribution function of the WCET $C_i$ of a task $\tau_i$ by capturing the two bounds: the smallest WCET value $C_i^l$ (WCET lower bound) and the largest WCET value $C_i^u$ (WCET upper bound). These bounds can be estimated by the designer on the basis of his/her knowledge of the tasks' characteristics and the possible range of ASIP microarchitectures. For example, the upper bound can correspond to a sequential execution of $\tau_i$ on the slowest ASIP considered, whereas the lower bound could correspond to an ASIP specifically tailored for $\tau_i$ and which fully exploits

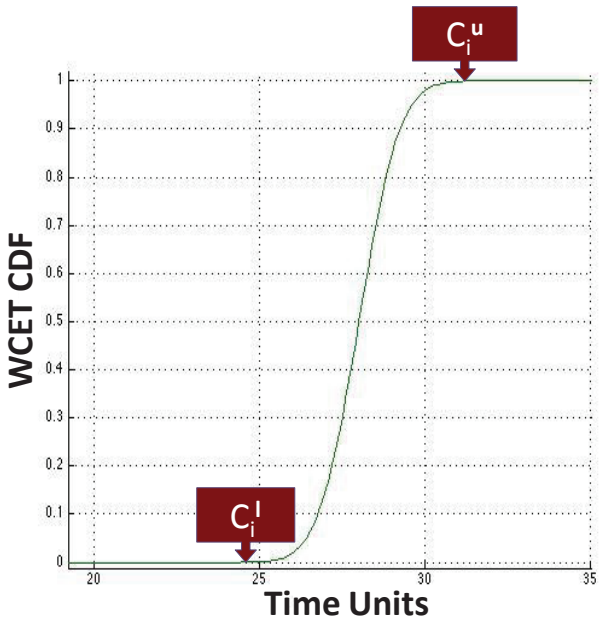


Fig. 1: A Sample WCET CDF

TABLE I: Micro-architecture features explored

| *Task* | *Issue width* | *num. ALU* | *num. MUL* | *RF size* | *Data Cache (KB)* | *Data Cache Line (bytes)* | *Load slot* | *Store slot* |
|---|---|---|---|---|---|---|---|---|
| *mp3 decoder* | 1,2,3, 4,5,6, 7,8 | 4,5,6, 7,8 | 2,3,4, 5,6,7, 8 | 32, 64 | 4, 8, 16 | 16, 32, 64, 128 | 4 | 2 |
| *jpeg decoder* | 2,3,4, 5,6,7,8 | 4, 5,6,7, 8 | 1,2,3, 4,5,6, 7,8 | 16, 32, 64 | 4, 8, 16 | 16, 32, 64 | 4 | 2 |

the parallelism within $\tau_i$ (using As Soon As Possible (ASAP) scheduling of the task DFG). Within these two values, we fit a Normal distribution for $C_i$ that models the WCETs of the task executing on an undefined ASIP that has not been synthesized yet.

The cumulative distribution function (CDF) $P(C_i \leq x)$ is the probability that WCET $C_i$ has a value lesser than or equal to $x$. Alternatively, the task WCET CDF is an indication of the percentage of ASIP configurations that would allow $C_i$ to be lesser than or equal to $x$. The distribution is built by deriving the mean and the standard deviation values. Firstly, the mean is calculated by taking the average of the bounds $C_i^l$ and $C_i^u$ ($\frac{C_i^l+C_i^u}{2}$). Then, the standard deviation is calculated based on the well known equations of the Normal distribution CDF. The CDF is completely defined once the mean and standard deviation are known. A sample WCET CDF is shown in Fig. 1. For GPPs, DSPs and legacy PEs, there is no uncertainty in WCET. Therefore, if the WCET of a task $\tau_i$ on any of these PEs is $y$, then the CDF of $\tau_i$ converges to a step function or a single value, which can be expressed as $P(C_i = y) = 1$. Finally, we define the task information tuple for every task $\tau_i$ as $\Gamma_i = \{C_i^u, C_i^l, a_i, T_i\}$.

We now present simulation results that support our claim that normal distribution is a good approximation to model the WCET of a task. We discuss the results for two tasks of different size and complexity: a mp3 decoder (part of the MAD library [1]) and a jpeg decoder [2]. The variation of the WCET of these tasks is measured for several micro-architectural configurations using the VEX [6] simulator developed at HP laboratories. The simulator models a VLIW architecture that could in principle be used in ASIPs. Table I presents the microarchitecture design space used for the experiments. We varied the number of arithmetic and logic units (ALU), multipliers (MUL), registers in the register file (RF), the issue, load and store slots, the data cache size and the data cache line size. Within the parameters in Table I, we considered a large

TABLE II: Microarchitectures associated to the WCET upper and lower bounds

| *Task* | *WCET* | *Issue width* | *num. ALU* | *num. MUL* | *RF size* | *Data Cache (KB)* | *Data Cache Line (bytes)* | *Load slot* | *Store slot* |
|---|---|---|---|---|---|---|---|---|---|
| mp3 decoder | $C_l$ | 8 | 8 | 8 | 64 | 16 | 128 | 4 | 2 |
| | $C_u$ | 1 | 4 | 2 | 32 | 4 | 16 | 4 | 2 |
| jpeg decoder | $C_l$ | 8 | 8 | 8 | 64 | 16 | 64 | 4 | 2 |
| | $C_u$ | 2 | 4 | 1 | 32 | 4 | 16 | 4 | 2 |

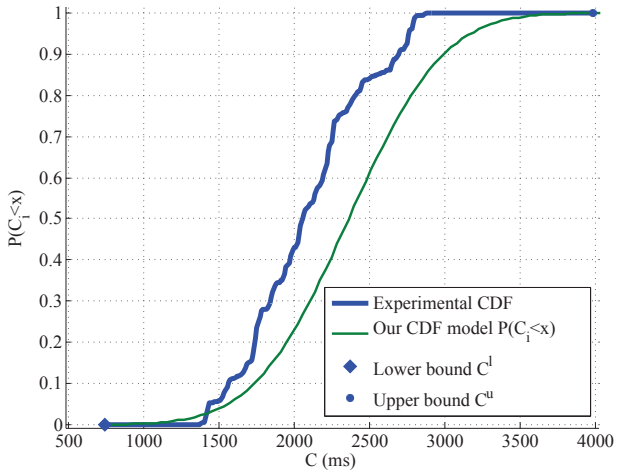


Fig. 2: Comparison of our proposed CDF model ($P(C_i \leq x)$) with the simulation results for mp3 decoder task

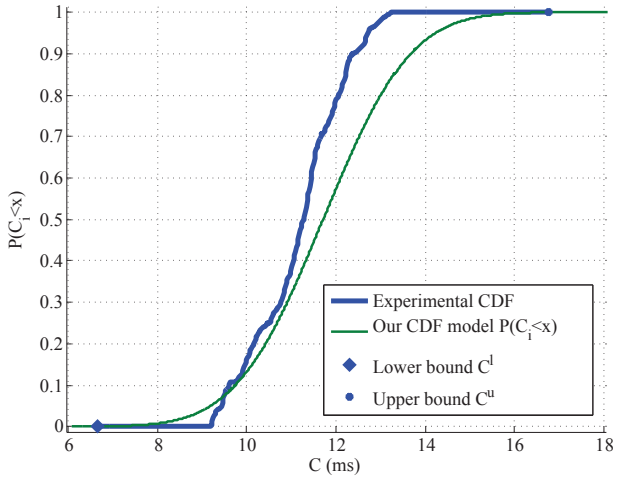


Fig. 3: Comparison of our proposed CDF model ($P(C_i \leq x)$) with the simulation results for jpeg decoder task

number of micro-architectural configurations: 1,632 for the mp3 decoder task and 1,068 for the jpeg decoder task. For each micro-architecture configuration, we compiled and simulated the execution of the tasks thereby obtaining the number of processor cycles required. Assuming a processor frequency of 100 MHz, we compute the corresponding execution times in *ms*. For a particular microarchitecture,the largest value of the execution time was considered as WCET after extensive simulations with multiple input files. We know that such a value does not represent the WCET, which is a theoretical upper bound determined through analysis, but we believe this value is a good approximation for our experiments.

We construct the CDF of WCET for the mp3 decoder and jpeg decoder tasks from the simulation data. These are plotted using a blue line in Fig. 2 and Fig. 3 for mp3 and jpeg decoder respectively. The CDFs obtained using our WCET uncertainty model are plotted using a green line. Our WCET uncertainty model (the green CDF) was obtained as explained earlier, considering a Normal distribution between the lower bound $C_i^l$ and upper bound $C_i^u$ of WCET. As mentioned earlier, we assume that these bounds are provided by the designer by evaluating two extreme microarchitectures from the range considered for the mp3 and jpeg decoder. The

microarchitecture corresponding to the upper and lower bounds of the WCET for the two tasks are summarized in Table II. The results shown in the CDF plots substantiate our claim that using a Normal distribution for WCET is a good approximation to capture the WCET uncertainty. It is important to mention here that the proposed WCET uncertainty model is used only for design space exploration, and not for providing timing guarantees.

## III. Problem Formulation

*Given multiple target applications where each application is modeled as a task graph $\mathcal{A}$, task information tuple $\Gamma_i$ for each task $\tau_i$ in each application, a global deadline $d$ and a set of PEs such as DSPs, GPPs and legacy PEs, the problem is to synthesize a multi-ASIP platform that simultaneously minimizes cost and maximizes the schedulability probability of target applications.*

Synthesis of a multi-ASIP platform means performing DSE to decide on the clustering of tasks that also takes into account the microarchitectural design space and subsequently synthesizing the microarchitecture of each ASIP. Our platform synthesis flow is shown in Fig. 4. Based on a set of target applications and the interconnection details, the input parameters available from the designer are the task information tuple (as discussed in Section II-A) and the message transmission times on the interconnection platform. We consider a bus-based platform in the paper. These input parameters along with the global deadline $d$ and the task similarity parameters (Section III-A and Section III-B) are then used by the various components of the platform synthesis stage. The *Uncertainty Model* has been discussed in Section II-A. The *Area Estimation Model* will be discussed later in Section III-A. The performance and cost quantities derived from these two models are then used for a multiobjective DSE. The DSE includes a Monte Carlo Simulation (MCS) loop (Section III-C). Each MCS loop iteration performs schedulability analysis to verify if the candidate platform solution is schedulable (Section III-C). The output of the Platform Synthesis stage consists of the number of ASIPs required and the cluster of tasks on each ASIP, which can then be synthesized.

There can be a set of legacy components that have to be used in the architecture and it is also possible that some tasks might be clustered on some specific PEs by the designer. Our optimization takes these constraints into account. In the end, we get a cost-efficient platform solution consisting of several ASIPs and possibly also other legacy PEs and the tasks clustered on each PE. We exploit the task similarities during DSE to reduce the cost of the platform. In the following subsections, we explain how ASIP datapath area and WCET uncertainty model are influenced by task similarities.

Let us consider two tasks $\tau_1$ and $\tau_2$ represented by their DFGs $G_1$ and $G_2$ as shown in Fig. 5(a). The DFGs consist of nodes that perform some operation of the task such as addition, multiplication, bit shifting etc. as shown in Fig. 5(a). The clustering of two tasks is represented as the merging of the DFGs $G_1$ and $G_2$ and the merged graph (shown in Fig. 5(b)) represents the datapath of the ASIP with the corresponding task cluster comprising of $\tau_1$ and $\tau_2$. The shaded parts in Fig. 5(a) depict the task similarities between tasks $\tau_1$ and $\tau_2$.

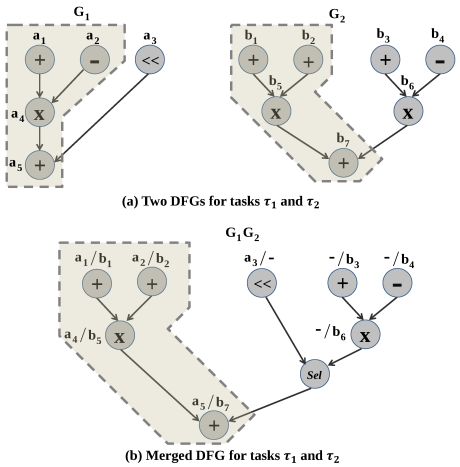


Fig. 5: A Graph Merging (or Task Clustering) Example

*Task similarity arises due to the identical set of nodes (that perform same functionality) and edges connecting these nodes in the task DFGs which can be merged in order to share the resources required to implement the nodes*. In Fig. 5(a), the nodes $\{a_1, a_2, a_4, a_5\}$ in $G_1$ are similar to nodes $\{b_1, b_2, b_5, b_7\}$ in $G_2$. In the clustering process, similar nodes are merged together (as shown in Fig. 5(b)) to reduce the area of the final ASIP datapath.

The existing graph merging techniques for datapaths have tried to optimize for area [13] and latency [19] of the resulting merged datapath. In [13], compatibility graphs are used to detect the similar nodes in the merged DFGs. Datapath merging is done using the compatibility graphs to minimize area. On the other hand, area and latency are traded off during datapath merging in [19]. In this paper, we assume that the designer would adopt a datapath merging technique more optimized for area as cost is one of our optimization objectives. This merging method can be used by the designer before the DSE stage in order to find the area required to implement each task if synthesized alone and the area required for a task pair if clustered together on an ASIP. These areas can be computed by using area values of standard components such as an adder, multiplier, etc. During DSE, we estimate the area required for various task clusters and the effect on WCET and hence schedulability of the solution.

### A. Area Estimation Model

Let the area required for implementation of each task (represented by DFG $G_i$) on ASIP be $a_i$. In our experiments, we use the number of gates as the unit of area. Let the merged area of two clustered tasks (represented by DFGs $G_i$ and $G_j$) be $a_{i,j}$. Then the area of the clustered tasks can be computed as $a_{G_iG_j} = a_i + a_j - a_{i,j} + a^o_{i,j}$, where $a^o_{i,j}$ is the area overhead due to the introduction of the *Sel* node. The node *Sel* selects one of the inputs and passes it to the output. Depending on the percentage of dissimilar nodes, the number of *Sel* nodes may vary, which also affects the area of clustered tasks. The area overheads for task pairs can also be computed before DSE. Let the number of *Sel* nodes introduced during task merging be $N_{Sel}$ and the area of a standard *Sel* component be $a_{Sel}$. Then the area overhead due to task merging is $N_{Sel} \times a_{Sel}$.

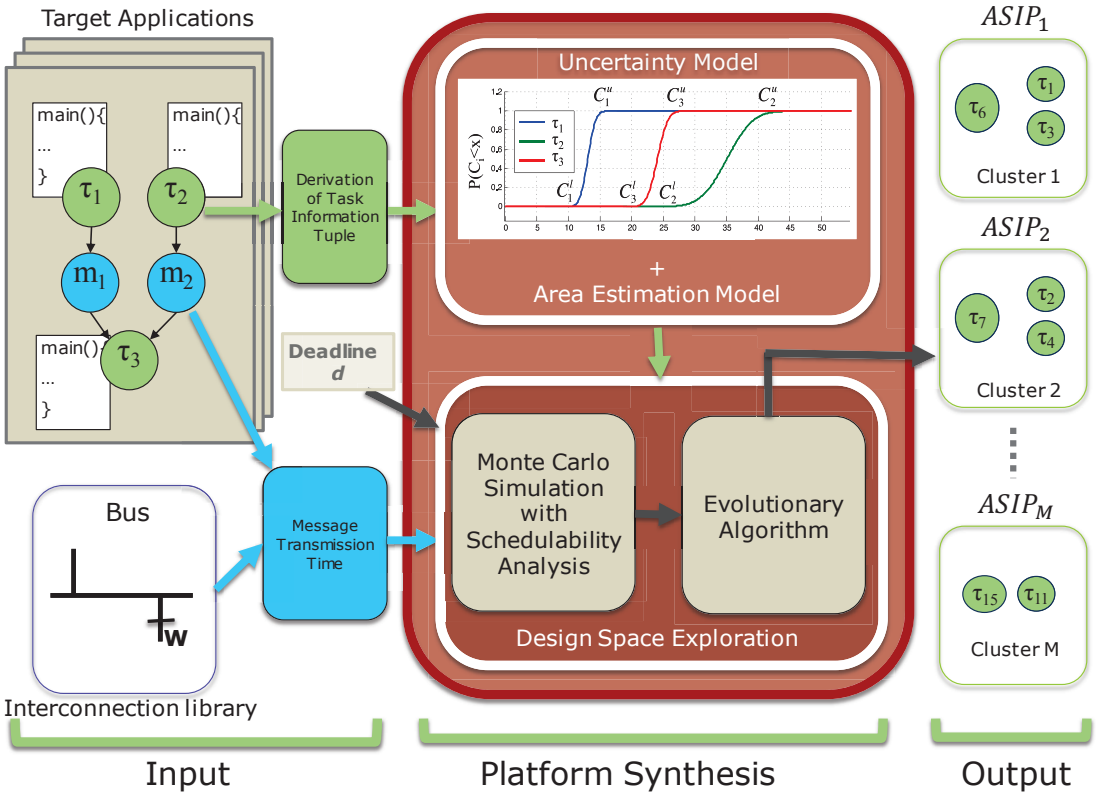


Fig. 4: Our Multi-ASIP Platform Synthesis Flow

In order to compute the area requirements of more than two tasks, after the area of the clustered task pair $a_{G_iG_j}$ is computed as shown earlier, the merged/clustered task pair are considered as a new task with area requirements $a_{G_iG_j}$. If another task represented by DFG $G_k$ is clustered with this new task, then the area of the merged nodes between DFGs $G_iG_j$ and $G_k$ is given by

$$a_{i,j,k} = \min_{x \in \{i,j\}} (Area\ of\ similar\ nodes\ between\ G_x\ and\ G_k) \quad (1)$$

As shown in Eqn. 1, we consider the merged area between $G_iG_j$ and $G_k$ as the minimum of the area requirements of similar nodes between tasks in $G_iG_j$ and $G_k$ in order to reduce the area overhead which may also contribute to WCET (discussed in Section III-B). The area overhead when DFGs $G_iG_j$ and $G_k$ are merged is

$$a^o_{i,j,k} = Area\ overhead\ between\ G_k\ and\ G_{min} \quad (2)$$

where $G_{min}$ is the graph among $G_i$ and $G_j$ that has minimum area of similar nodes with $G_k$. The area after the third task (represented by DFG $G_k$) is merged with $G_iG_j$ is given by $a_{G_iG_jG_k} = a_{G_iG_j} + a_k - a_{i,j,k} + a^o_{i,j,k}$. This clustering and area estimation can be iteratively performed until all the tasks clustered on a PE are included. In this paper, we estimate the area based on a task clustering order that clusters tasks on ASIPs in the order of fixed task priority so that tasks with higher priority incur lesser overhead in critical path of the DFG (explained in Section III-B. If there are $M$ ASIPs in the platform solution and if the area of task cluster on each ASIP, computed as described above, is denoted by $AC_n$, where $1 \leq n \leq M$, then our optimization objective for area cost is to minimize $\sum_{n=1}^{M} AC_n$.

### B. Effect of Clustering on WCET Uncertainty Model

There might be a variation in WCET CDF of a task due to overheads in the critical path of a task, when it is clustered with another task/group of tasks. This additional overhead is incurred when the tasks are merged in order to reduce area as shown in the graph merging example in Fig. 5. For the task with DFG $G_2$, there were three critical paths before merging : $b_1 \rightarrow b_5 \rightarrow b_7$, $b_3 \rightarrow b_6 \rightarrow b_7$ and $b_4 \rightarrow b_6 \rightarrow b_7$ (all paths with the same maximum length of critical operations). Due to merging, there is an additional overhead of a select node (Sel) in the second and third critical paths. This can result in a shift of the estimated upper and lower bounds of the WCET CDF. The above mentioned overhead arises from the group of nodes that are not merged and feed an input to one of the merged nodes. This overhead is significant when the grouped nodes constitute a custom instruction and this custom instruction is implemented on a custom hardware unit.

If the WCET overhead for a task during clustering of two tasks is $\delta$ cycles (or some unit of time), $C^u$ is the upper bound on the WCET and $C^l$ is the lower bound on WCET, then the WCET CDF for the task is formulated by shifting the upper and lower bound as $C^u_{new} = C^u + \delta$ and $C^l_{new} = C^l + \delta$ respectively. The WCET overhead for each task pair can be found before DSE when the area overhead is computed as the number of introduced *Sel* nodes is known. Here, we only consider the sharing of resources in the datapath while accounting for the change in WCET bounds and WCET CDF. The sharing of memory and other resources is not considered in this paper.

### C. Schedulability Analysis

Stochastic Schedulability Analysis was proposed in [8] where each job/task had different execution times and the execution time was modeled as a probability distribution. However, in our case, this analysis is not possible because each job/task has the same WCET for a specific microarchitectural configuration and the WCET varies with different microarchitectural configuration. Thus, as explained earlier, WCET is a stochastic variable. Our approach is to use MCS to sample randomly, in each iteration, a new value for $C_i$ based on its CDF. As MCS is known for being time consuming, a reduced number of samples has been used to speed up the execution. We randomly sample 5000 values from the WCET CDF of each task. In each MCS iteration, the sampled $C_i$ value is used to compute the number of tasks serviced within the global deadline $d$. The schedulability probability (which is our performance objective) is then computed using the number of tasks serviced, which is explained in detail next.

In this paper, we consider *fpps* policy to schedule tasks. However, our platform synthesis technique can be used in the context of other scheduling policies also. Firstly, we define service as the number of jobs of a task processed by each ASIP within a deadline $d$. The global deadline that we consider here is the hyperperiod[1]. The schedulability analysis is performed by computing the worst-case time remaining for each task and therefore *worst-case service* provided to each task. We use the worst-case service as the parameter for schedulability analysis because exact service offered to each task is not known until the ASIP architectures are defined. *Moreover, instead of using conventional response time analysis (RTA) for schedulability, we use the worst-case service based approach because a large number of applications (such as multimedia applications) that will be run on the ASIP would require a certain number of jobs of a task to be serviced in a particular time interval instead of having individual job deadlines*. Towards this, we firstly find the worst-case remaining time $\Delta_i$ for each task $\tau_i$ in accordance to the *fpps* policy, which is given by:

$$\Delta_i = \max\left\{\left(\Delta_h - \sum_{\tau_j \in hp(\tau_i)} \left(\frac{\Delta_h}{T_j}\right) C_j\right), 0\right\} \tag{3}$$

where $hp(\tau_i)$ is the set of tasks which have higher priority than task $\tau_i$ and are clustered together with $\tau_i$ on the same ASIP. Here $T_j$ and $C_j$ are the period and WCET (it is one of the sampled points from the WCET CDF), respectively, of the higher priority task $\tau_j$. The hyperperiod is denoted by $\Delta_h$. We use Eqn. 3 to compute the worst-case remaining time for a task $\tau_i$ in an interval equal to the hyperperiod. For this, we first compute the worst-case number of jobs (computed as $\frac{\Delta_h}{T_j}$) of all higher priority tasks and the worst-case times required to process them (computed as $\left(\frac{\Delta_h}{T_j}\right) C_j$). The worst-case processing times of tasks $\tau_j$ are then summed up to get the worst-case processing time of the higher priority tasks in $hp(\tau_i)$ and this is finally subtracted from the hyperperiod to get the worst-case remaining time for task $\tau_i$. If there is no remaining time after the higher priority tasks are processed, then we set $\Delta_i = 0$. Tasks having the longer critical path are assigned a higher priority [11].

[1]least common multiple (LCM) of the periods of the tasks

In order to analyze the schedulability of tasks, we compute the worst-case number of processed jobs of each task over the time interval $\Delta_h$. Once the worst-case remaining time of each task $\Delta_i$ is obtained, we can compute the worst-case number of processed jobs of each task as shown below

$$J_i = \begin{cases} \left\lfloor \frac{\Delta_i}{C_i} \right\rfloor & \text{if } t_i > \Delta_i \\ \frac{\Delta_h}{T_i} & \text{else} \end{cases} \tag{4}$$

where $t_i = \left(\frac{\Delta_h}{T_i}\right) C_i$ is the worst-case processing time for maximum possible number of jobs of task $\tau_i$ in an interval $\Delta_h$.

The worst-case number of processed jobs of each task can be used to evaluate the overall schedulability of the task clusters on $M$ ASIPs using the schedulability metric as shown below:

$$S_M = \begin{cases} S_M + 1 & \text{if } \sum_i J_i = \sum_i \frac{\Delta_h}{T_i} \\ S_M & \text{else} \end{cases} \tag{5}$$

If the task clustering on the ASIPs is schedulable (i.e., all jobs of the tasks mapped to $M$ ASIPs are serviced in time $\Delta_h$ or $\sum_i J_i = \sum_i \frac{\Delta_h}{T_i}$), then schedulability metric value $S_M$ is incremented. If the task clustering is not schedulable, then $S_M$ remains at the same value. For each iteration of MCS, $S_M$ is computed. To incorporate message communication between tasks on the bus, we check for schedulability of messages on the bus by ensuring that the bus utilization is below 100%. Although this does not guarantee the worst-case end-to-end schedulability, we provide the task clusters that have a high probability for schedulability of tasks and messages. This can be used as the starting point for ASIP synthesis. Once the ASIPs are synthesized, a detailed end-to-end schedulability analysis can be performed using techniques proposed in [17]. Let us assume that there are $P$ iterations of MCS corresponding to $P$ sample points of WCET probability density function of each task. Then the probability of schedulability of the platform solution is computed as $pS_M = \frac{S_M}{P}$ if the bus utilization is less than 100%, otherwise $pS_M$ is set to zero. Our optimization objective for performance is to maximize $pS_M$.

## IV. Motivational Example

The motivating example consists of a task set with 10 tasks. The area values $a_i$ for each task if synthesized alone on an ASIP are given in Table III. For 10 tasks, there are 100 values for $\delta_{i,j}$, $a_{i,j}$ and $a^o_{i,j}$ corresponding to each task in a task pair. Therefore, we do not show it here.

In the conventional case, without the uncertainty model and the area estimation model considering task similarities, the designer would characterize the WCET of each task $\tau_i$ with an average value $C_i^{avg}$. In this case, there cannot be any

TABLE III: WCET values (in *ms*) and Datapath Area (in *KGates*) for the Motivating Example

| | $\tau_1$ | $\tau_2$ | $\tau_3$ | $\tau_4$ | $\tau_5$ | $\tau_6$ | $\tau_7$ | $\tau_8$ | $\tau_9$ | $\tau_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $C_i^u$ | 16.5 | 13.2 | 12.1 | 6.6 | 7.7 | 8.8 | 13.2 | 8.8 | 7.7 | 14.3 |
| $C_i^l$ | 9 | 7.2 | 6.6 | 3.6 | 4.2 | 4.8 | 7.2 | 4.8 | 4.2 | 7.8 |
| $C_i^{avg}$ | 15 | 12 | 11 | 6 | 7 | 8 | 12 | 8 | 7 | 13 |
| $a_i$ | 150.6 | 180.5 | 300.7 | 109.7 | 99.2 | 106.5 | 250.3 | 130.4 | 115.6 | 260.6 |

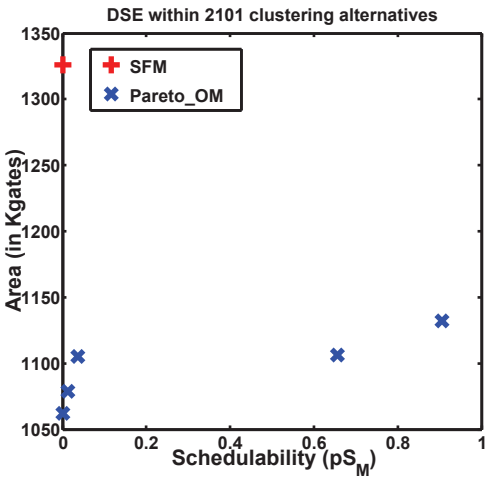


Fig. 6: Clustering solutions using *SFM* and *Pareto_OM* approaches

optimization of the datapath area as task similarities are not considered and therefore clustering will not minimize area. The $C_i^{avg}$ are presented in Table III. The optimization objective in this case is schedulability probability considering the $C_i^{avg}$ values of tasks. We denote this straightforward approach as *SFM*. The $C_i^l$ and $C_i^u$ values for each task $\tau_i$ are presented in Table III. The task WCET CDFs are constructed using these WCET bounds and $\delta_{i,j}$ values. In our proposed approach, $\delta_{i,j}$, $a_{i,j}$, $a_{i,j}^o$, $C_i^l$ and $C_i^u$ are used to optimize both area and schedulability. We denote our approach as *Pareto_OM*, which stands for Pareto Optimal Clustering method.

The optimization results obtained using the two methods *SFM* and *Pareto_OM* are shown in Fig. 6. It is quite evident from the plot that the *SFM* approach gives a clustering solution which has zero schedulability probability in the presence of task similarities. On the other hand the *Pareto_OM* approach provides more than one clustering solution, which have good schedulability probabilities. More precisely, it provides a solution with a 90.66% chance for the tasks to be schedulable. Moreover, it is also interesting to note that the datapath area obtained using the *Pareto_OM* approach (approx. 1132 *Kgates* for the best performance solution) is lower than that obtained using the *SFM* approach (approx. 1326 *Kgates*).

## V. Cost and Performance Optimization

The design space of the possible clustering solutions is huge and cannot be exhaustively explored. Therefore, we propose a Genetic Algorithm (GA)-based optimization approach. In particular, among the algorithms described in literature, we use a controlled elitist GA (a variant of the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [4]) for multiobjective optimization.

GA is a metaheuristic optimization strategy that follows the principles of natural evolution. It defines an initial set of randomly generated candidate solutions called "population". Each candidate solution is identified by a string called "chromosome". A set of solutions from the initial population undergo "recombination" and "mutation" to combine and vary the existing chromosomes (parent solutions) and therefore replace the existing population with one that has a better "fitness value". This new set of solutions form the next generation over which the earlier steps are repeated.

The controlled elitist GA that we use allows mantaining some diversity in the population by also retaining solutions with a lower fitness value across different generations. This is important for the convergence to an optimal Pareto front. The GA causes the population to evolve towards a better one until a termination condition is reached. In our case the algorithm stops when there is no improvement in the fitness of the population after a certain number of generations $g_{max}$.

In a candidate solution, a chromosome is encoded as a string in which each element called "gene" represents a task $\tau_i$ and the value assigned indicates the $j^{th}$ index of $PE_j$ on which the task is clustered. The initial population is composed of $n$ solutions. From this initial population, a set of solutions is selected as the parent population, which undergoes the next few steps. Firstly, recombination (or crossover) is performed according to a probability $p_c$. We used a standard single point crossover: given two parents, they are partitioned at a random point and the resulting parts of the two parents solutions are combined to generate a child solution if a randomly generated number $\leq p_c$. Mutation is then applied on the children generated from crossover. A probability $p_m$ is used to determine if each single gene of a child solution is randomly changed or not (i.e. modify the PE associated to each task). After mutation $n$ offspring solutions are generated. Finally, out of the $2n$ solutions ($n$ parent solutions + $n$ offspring solutions), the best $n$ solutions are selected according to the fitness function. The steps discussed earlier are repeated until a certain number of generations are traversed. The parameters $g_{max}$, $n$, $p_c$ and $p_m$ have been tuned according to the results obtained running multiple executions of the algorithm with different synthetic applications.

In this work, the fitness function consists of two optimization objectives discussed in Sections III-A and III-C. The computation of datapath area/cost considering the task similarities is performed as presented in Section III-A whereas schedulability probability ($pS_M$) is computed as presented in Section III-C. These computations are performed for every candidate platform solution in each generation of GA.

## VI. Experimental Results

To evaluate the effectiveness of our *Pareto_OM* approach in deriving a multi-ASIP platform, we used 3 real-life benchmarks from the Embedded System Synthesis Benchmark Suite (E3S), version 0.9 [5] and 4 synthetic benchmarks. The real-life benchmarks used from E3S are *consumer-cords*, *telecom-cords* and *networking-cords*. The details of the benchmark and the obtained results are presented in Table IV. We also synthesized 6 benchmarks and their details and obtained results are presented in Table V. In each of the tables, the number of tasks constituting each benchmark is given in column 2 and the number of ASIPs used for task clustering is given in column 3. We provide the area cost (in Kgates) and performance results (in terms of schedulability probability $pS_M$) obtained with the *SFM* approach in columns 4 and 5 respectively. The same results obtained with *Pareto_OM* approach are shown in columns 6 and 7 respectively.

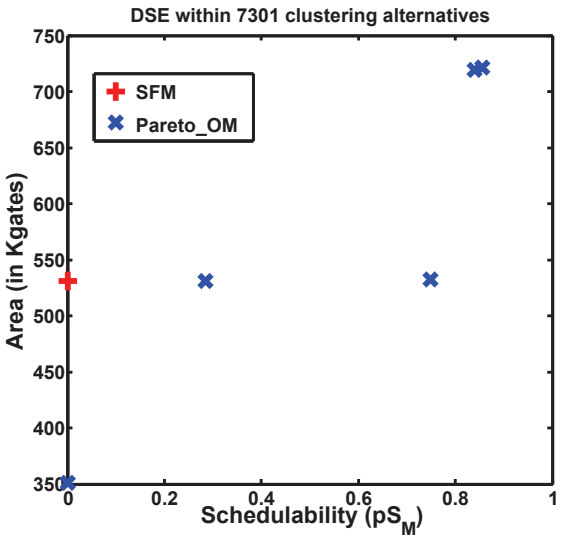


Fig. 7: Comparison of results using *SFM* and *Pareto_OM* for *consumer-cords*

The values for WCET upper and lower bounds of each task (both for real-life and synthetic benchmarks) are not shown here due to lack of space. The WCET value given in the real-life benchmark was considered as the average WCET value $C_i^{avg}$ of task $\tau_i$ and the WCET upper ($C_i^u$) and lower ($C_i^l$) bounds of each task were obtained by scaling $C_i^{avg}$ with some multiplication factors. For the synthetic benchmarks, the value of $C_i^{avg}$ was generated and the values of $C_i^u$ and $C_i^l$ were obtained by scaling the corresponding $C_i^{avg}$. For real-life benchmarks, the area of each task $\tau_i$ (denoted as $a_i$) was obtained from the code size of tasks in the E3S benchmark and area of real implementation of a well known task (from the benchmark) on a processor. For instance, if the area of the real implementation of a well known task $\tau_j$ (such as FFT) obtained from literature is $a_j^{real}$ and the code sizes of tasks $\tau_i$ and $\tau_j$ are $cs_i$ and $cs_j$ respectively, the area of the task $\tau_i$ was computed as $a_j^{real} \times \frac{cs_i}{cs_j}$. In the case of synthetic benchmarks, these values were generated.

Now we discuss how the values that account for task similarity were obtained. Once the area values required for the implementation of each task were obtained, for real-life benchmarks, we obtained the merged area of two clustered tasks ($\tau_i$ and $\tau_j$) $a_{i,j}$ and the area overhead introduced $a_{i,j}^o$ by looking at the task similarity as explained in Section III-A. In the case of synthetic benchmarks, these values were generated such that they did not exceed the area of each task. The WCET overhead $\delta$ (see Section III-B) introduced due to task clustering was obtained from the merged nodes in the case of real-life benchmarks. In the case of synthetic benchmarks,

TABLE IV: Real-Life Benchmarks

| Benchmark | Number of | | *SFM* | | *Pareto_OM* | |
|---|---|---|---|---|---|---|
| | tasks | ASIPs | Area | $pS_M$ | Area | $pS_M$ |
| consumer-cords | 12 | 2 | 531.011 | 0% | 532.5 | 74.88% |
| network-cords | 13 | 2 | 182.845 | 0% | 182.845 | 95.96% |
| telecom-cords | 30 | 3 | 1163.929 | 0% | 1148.896 | 98.16% |

TABLE V: Synthetic Benchmarks

| Benchmark | Number of | | *SFM* | | *Pareto_OM* | |
|---|---|---|---|---|---|---|
| | tasks | ASIPs | Area | $pS_M$ | Area | $pS_M$ |
| synth_1 | 24 | 4 | 408.092 | 0% | 394.665 | 94.04% |
| synth_2 | 30 | 4 | 521.364 | 0% | 377.344 | 75.72% |
| synth_3 | 34 | 4 | 615.237 | 0% | 407.307 | 55.02% |
| synth_4 | 46 | 6 | 723.982 | 0.16% | 493.012 | 94.22% |

these values were manually generated such that the WCET overhead did not exceed $C_i^{avg}$ value of each task in a clustered task pair.

The parameters of GA were tuned so that results obtained converge towards the optimal result. The initial population size was set to $n = 100$. The crossover and mutation probability were set to $p_c = 0.4$ and $p_m = 0.2$ respectively. The GA terminated when there was no improvement in the fitness function for 6 generations. The maximum number of generations were set to $g_{max} = 100$. Both the optimization approaches *SFM* and *Pareto_OM* were implemented in Matlab 2012 and run on Intel Core i7 CPU (2.8 GHz). The runtime of the benchmarks ranged between 10 *minutes* to 1 *hour*.

In our experiments, we present the advantage of our proposed platform synthesis approach *Pareto_OM* in comparison to the *SFM* approach for real-life benchmarks and synthetic benchmarks. The experimental setup details and the results obtained using both the approaches are presented in Table IV and Table V. The *Pareto_OM* approach generates a pareto front of solutions. We only report the solution that gives maximum schedulability probability, but still has lesser area than what is obtained using *SFM* in Table IV and Table V. It is clear from the results that the solution obtained using *Pareto_OM* approach outperforms the solution obtained using *SFM* approach for all the real-life and synthetic benchmarks. In order to compare the performance of the two approaches, the clustering solution obtained by *SFM* was then evaluated under the inclusion of WCET uncertainty and task similarity effects.

We also present the pareto plots for the real-life and synthetic benchmarks. For *consumer-cords* (Fig. 7), using *Pareto_OM* approach, there are two solutions which have a high schedulability probability, but require higher area in comparison to the *SFM* approach. Therefore, a better choice would be to select the solution given in Table IV. The comparison between the *SFM* approach and the *Pareto_OM* approach for *network-cords* is shown in Fig. 8. Although the *Pareto_OM* approach returns a couple of solutions with almost comparable area cost to the solution given by *SFM* approach, these solutions have a higher probability of schedulability once they are synthesized into ASIPs. This result highlights the fact that for *network-cords*, the solution that optimizes for performance is not able to exploit the task similarities well to reduce area. The comparison between the *SFM* approach and the *Pareto_OM* approach for *telecom-cords* is shown in Fig. 9. In this case, there a few solutions proposed by *Pareto_OM* that have higher probability of schedulability in comparison to the solution proposed by *SFM* with significant area savings. This result is because the task similarities in *telecom-cords* are well exploited by the *Pareto_OM* approach. However, there are fewer points on the pareto front because the message

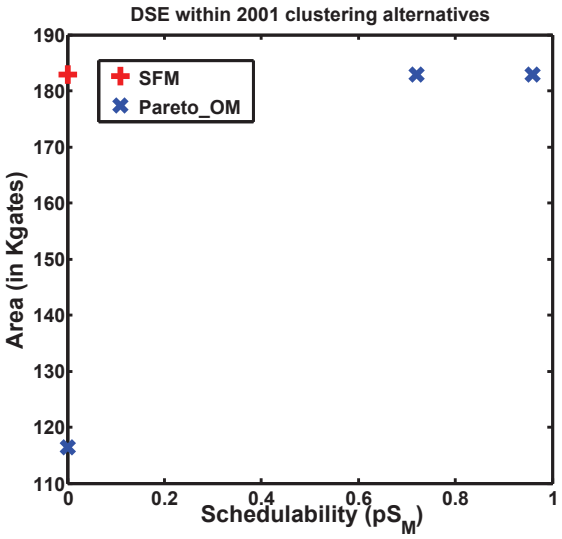


Fig. 8: Comparison of results using *SFM* and *Pareto_OM* for *network-cords*

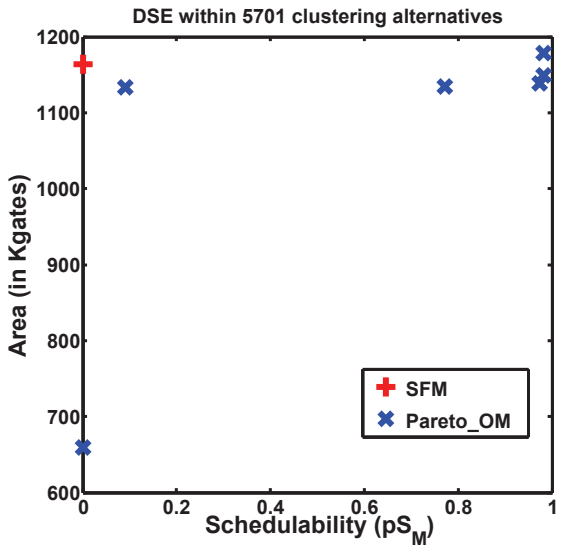


Fig. 9: Comparison of results using *SFM* and *Pareto_OM* for *telecom-cords*

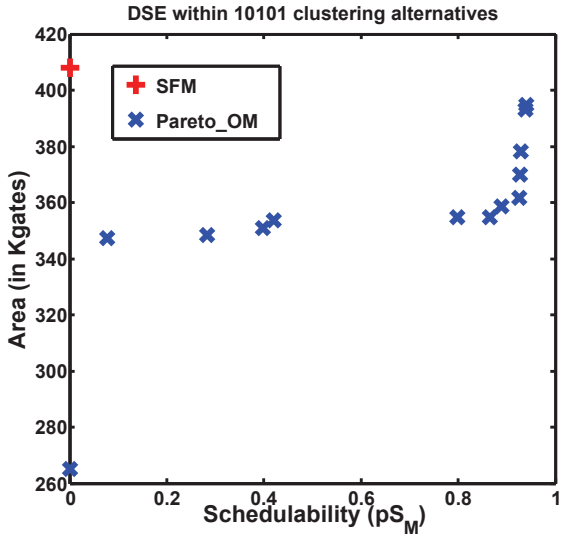


Fig. 10: Comparison of results using *SFM* and *Pareto_OM* for *synth_1*

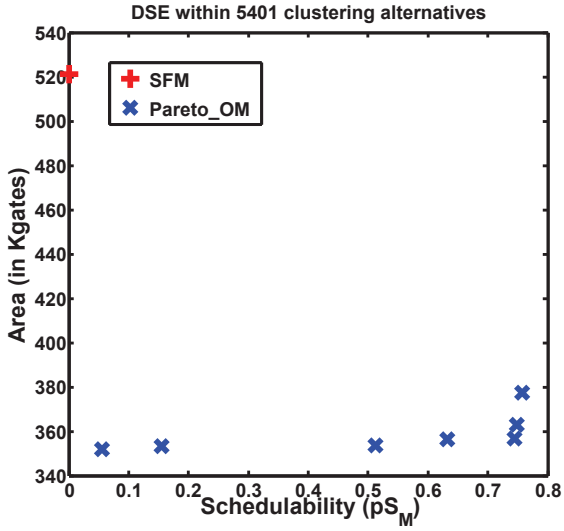


Fig. 11: Comparison of results using *SFM* and *Pareto_OM* for *synth_2*

schedulability condition (Section III-C) on the bus was not satisfied.

Comparison of *SFM* approach and the *Pareto_OM* approach for *synth_1* is shown in Fig. 10. In this case, our *Pareto_OM* approach exhibits better clustering solutions giving better area cost and performance in comparison to *SFM* approach. This is because the similar area among clustered tasks and WCET overhead are significant factors that considerably affect the optimality of the clustering solutions. Fig. 11 shows the comparison of *SFM* approach and the *Pareto_OM* approach for *synth_2*. There are multiple clustering solutions shown for *synth_2*, which save considerable area in comparison to *SFM* approach for a small reduction in $pS_M$, but still having a higher $pS_M$ in comparison to the clustering solution proposed by *SFM* approach in red marker. Better clustering solutions were also observed for *synth_3* (Fig. 12) and *synth_4* (Fig. 13) using the *Pareto_OM* approach. However, for *synth_3*, the schedulability probability is lower than the other synthetic benchmarks because the contention on the bus due to inter-processor task communication does not satisfy the message schedulability condition.

## VII. Conclusion

In this paper, we have proposed a multi-ASIP platform synthesis approach. This is challenging as the schedulability of tasks clustered on an ASIP depends on the WCET of the task and the WCET of the task is known only when the ASIP is synthesized. We break this circular dependency by introducing a WCET uncertainty model. A datapath area estimation model is also proposed. The two models have taken task similarity into account while clustering. We propose a

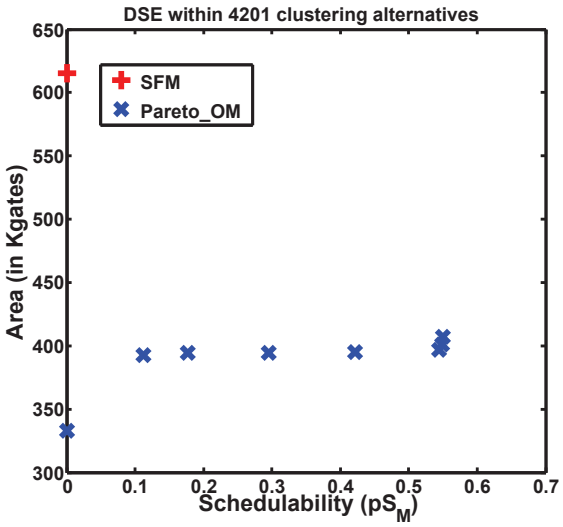


Fig. 12: Comparison of results using *SFM* and *Pareto_OM* for *synth_3*

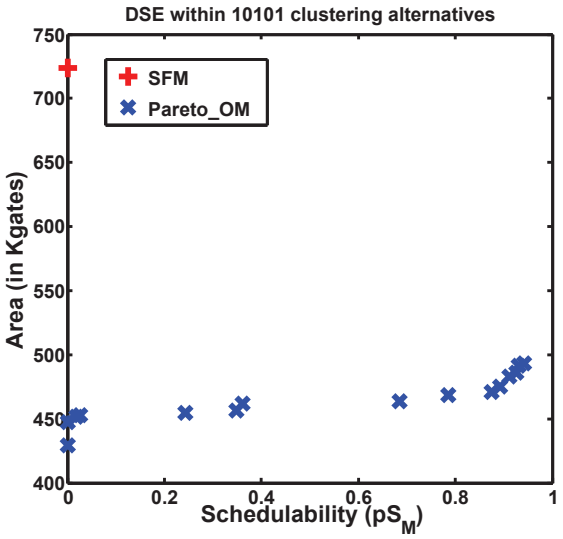


Fig. 13: Comparison of results using *SFM* and *Pareto_OM* for *synth_4*

GA-based multiobjective optimization approach to get a pareto front of solutions that enables cost performance trade-offs for the multi-ASIP platform. The efficacy of our approach has been evaluated using real-life and synthetic benchmarks. From the experimental results, it was widely observed that our proposed approach (*Pareto_OM*) provided a number of platform solutions that exhibited lesser cost and higher schedulability probability in comparison to the *SFM* approach.

## REFERENCES


[1] MAD, MPEG Audio Decoder. http://www.underbit.com/products/mad/.

[2] Mamps project, partitioned jpeg decoder algorithm. http://www.es.ele.tue.nl/mamps/example.php.

[3] C. Brehm, T. Ilnseher, and N. Wehn. A scalable multi-asip architecture for standard compliant trellis decoding. In *International SoC Design Conference (ISOCC)*, pages 349–352, 2011.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, 1917:849–858, 2000.

[5] R. Dick. Embedded system synthesis benchmarks suite, 2002.

[6] J. A. Fisher, P. Faraboschi, and C. Young. VEX, a VLIW Example. http://www.hpl.hp.com/downloads/vex/.

[7] T. Kangas, P. Kukkala, H. Orsila, E. Salminen, M. Hännikäinen, T. D. Hämäläinen, J. Riihimäki, and K. Kuusilinna. Uml-based multiprocessor soc design framework. *ACM TECS*, 5:281–320, 2006.

[8] K. Kim, J. L. Diaz, L. L. Bello, J. M. Lopez, C.-G. Lee, and S. L. Min. An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Transactions on Computers*, 54(11):1460–1466, 2005.

[9] H. Kopetz. *Real-time systems: design principles for distributed embedded applications*. Springer, 2011.

[10] A. Kumar, S. Fernando, Y. Ha, B. Mesman, and H. Corporaal. Multiprocessor systems synthesis for multiple use-cases of multiple applications on fpga. *ACM TODAES*, 13(3):40, 2008.

[11] Y.-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4):406–471, 1999.

[12] T. Mitra and P. Yu. Satisfying real-time constraints with custom instructions. In *CODES+ISSS*, pages 166–171, 2005.

[13] N. Moreano, E. Borin, C. De Souza, and G. Araujo. Efficient datapath merging for partially reconfigurable architectures. *IEEE TCAD*, 24(7):969–980, 2005.

[14] O. Muller, A. Baghdadi, and M. Jézéquel. From parallelism levels to a multi-asip architecture for turbo decoding. *IEEE TVLSI*, 17(1):92–102, 2009.

[15] H. Nikolov, T. Stefanov, and E. Deprettere. Multi-processor system design with espam. In *CODES+ISSS*, pages 211–216, 2006.

[16] H. Nikolov, T. Stefanov, and E. Deprettere. Systematic and automated multiprocessor system design, programming, and implementation. *IEEE TCAD*, 27(3):542–555, 2008.

[17] J. C. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *IEEE Real-Time Systems Symposium*, pages 26–37, 1998.

[18] S. L. Shee and S. Parameswaran. Design methodology for pipelined heterogeneous multiprocessor system. In *44th DAC*, 2007.

[19] M. Zuluaga and N. Topham. Resource sharing in custom instruction set extensions. In *IEEE SASP*, pages 7–13, 2008.

[20] M. Zuluaga and N. Topham. Design-space exploration of resource-sharing solutions for custom instruction set extensions. *IEEE TCAD*, 28(12):1788–1801, 2009.