CLEO-CoDe: Exploiting Constrained Decoding for Cross-Layer Energy Optimization in Heterogeneous Embedded Systems

Siva Satyendra Sahoo, Akash Kumar
Chair of Processor Design, Center for Advancing Electronics Dresden (CfAED)
Technische Universität Dresden
Dresden, Germany
{siva_satyendra.sahoo,akash.kumar}@*tu-dresden.de*

***Abstract*—System-level design for low-power and energy efficiency in embedded systems using Heterogeneous Multi-Processor System-on-Chip (HMPSoC) is a challenging task due to the large design space. The related Design Space Exploration (DSE) suffers from scaling due to various degrees of freedom across multiple layers of the compute stack. Traditional multi-objective meta-heuristic approaches work well for unconstrained system-level design, but do not scale well with the additional system and user constraints. Using a SATisfiablity problem (SAT) solver as a decoder for the meta-heuristics has been explored in related research as a better solution for a discrete constrained multi-objective optimization problem. This approach restricts the problem to the feasible space, hence improving the quality of the results. In this paper, we explore the ways in which constrained decoding such as the SAT decoding approach can be leveraged for cross-layer design space exploration. Low-power methodologies such as Dynamic Voltage and Frequncy Scaling (DVFS) and application-specific implementations are integrated, thus scaling the design space. Additionally, we demonstrate how user constraints on the system synthesis problem can be learned by our proposed approach to prune the meta-heuristic design space and improve the quality of the solutions. As the constraints on the problem increase and the design space scales, the constrained decoding approach outperforms a typical meta-heuristic approach.**

***Index Terms*—Cross-layer System Design, Energy-efficient design, System-level Design, Embedded Systems**

I. Introduction

Heterogenous multi-processor architectures are an increasingly common trend in high-end embedded and cloud **S**ystem-**o**n-**C**hip (SoC). These architectures are ideally exploited for parallel execution of applications.To ensure **Q**uality of **S**ervice (QoS) requirements, the resources of an HMPSoC are shared amongst the applications. The heterogeneity of the cores leads to a diverse set of execution scenarios with varying features such as execution time, energy, thermal profile and reliability. Cross-layer design choices such as DVFS and Activation factor also contribute to the diversity of the execution scenario.

The choices involved in the execution of a set of applications on a HMPSoC represent a design space. The exploration of this design space is known as the classic system synthesis problem. The system synthesis problem involves mapping a set of applications with task level granularity (functional objects) on a HMPSoC comprising of **P**rocessing **E**lements (PEs) and a **N**etwork-on-**C**hip (NoC) (structural objects) [1]. Various design-time and run-time approaches have been explored for the system synthesis problem. Hybrid Mapping, where design-time DSE is used to search for a set of optimized solutions which are then leveraged by the run-time arbiter to lead to a concrete task mapping, has gained significant traction as a design flow for HMPSoCs. [2] [3].

The design time DSE performed during hybrid mapping is a discrete multi-objective optimization problem where the solutions represent a set of trade-offs in the design space [4]. The design space is extremely large and grows exponentially with increase in the number of processors, total tasks in an application and cross-layer design choices. Meta-heuristic based optimization is a common and powerful tool for multi-objective optimization and have been used extensively for task mapping [4]. A realistic meta-heuristic search considers a constrained multi-objective optimization problem, where the constraints are either a result of the on-chip network limitations or user requirements, such as maximum allowed total energy or execution time. In the case of constrained DSE a decoupled decoder based approach such as SAT Decoding [5] are known to give better results than traditional meta-heuristic approaches. This is because a decoupled decoding process is able to restrict the solutions within the feasible space, resulting in a more efficient exploration. As the proportion of the infeasible region increases, normal DSE focuses on removing the infeasible solutions rather than improving solutions in the feasible space. The normal DSE approaches lack a method to restrict the search to the feasible space. Our solution explores the improvements of a constrained **P**suedo **B**oolean (PB) decoder-based or **B**inary **I**nteger **L**inear **P**rogramming (ILP) DSE over a traditional penalty-based DSE for the classic system synthesis problem.

The contributions of our work are as follows,

1) We performed DSE using a constrained decoding based approach to solve the design-time system synthesis problem. Our DSE approach is able to scale appropriately to provide quality solutions.
2) We proposed novel feasible-space learning method for efficiently encoded problem definition by exploiting the inherent monotonocities of the problem. The learning in this methodology were based on realistic constraints on the system synthesis such as user-defined constraints on total energy consumption and total execution time.

The rest of the paper is organized as follows. In Section II, we provide the relevant background and brief overview of previous works using a SAT/Binary ILP based decoder. The system model used for evaluation of the proposed methods is presented in Section III. We also introduce the system synthesis methodology in this section. The proposed DSE including the formulation of the decoder is elucidated in Section IV. In Section V, we discuss the results from the experimental evaluation of the proposed methods and conclude the article in Section VI with a discussion on the scope for related future research.

II. Background and Related Works

A. Cross-layer Low-power Design

Cross-Layer, low-power methodologies such as DVFS have also been explored in literature. [6] surveys the existing techniques for task scheduling and task mapping in DVFS and **D**ynamic **P**ower **M**anagement (DPM) single and multi-processor systems. Energy efficient designs while considering cross-layer design are also prevalent. [7] and [8] provide varied solutions to the task mapping problem including mixed-integer linear programming and heuristics based approaches. [9] and [10] show the benefits of using Genetic Algorithms for the DSE. Meta-heuristics based DSE methods are able to find a large set of operating points representing a wide range of possible

978-1-6654-2614-5/21/$31.00 ©2021 IEEE

solutions [4].

B. Constrained Decoder - Meta-Heuristic approach to System Synthesis

The work on a decoder based approach for solving constrained optimization problems using multi-objective heuristics was first put forth in [11]. [11] highlights the need for an homomorphous mapping between the complete design space and the feasible space in [11]. In the case of constrained multi-objective optimization, using meta-heuristics, the feasible space is generally only a fraction of the complete design space. In certain cases when finding even a single feasible solution of the multi-objective problem is NP Hard, the traditional strategies for restricting the solution to the feasible space, such as repair strategies or penalty based strategies, perform poorly. This poor performance is a result of the shift of focus from optimization to finding feasible solutions. This can be attributed to the sparseness of feasible solutions in the design space combined with the lack of active avoidance of the infeasible region.

A novel approach of using a SAT problem to restrict the search space to the feasible region has been put forth in [5]. The approach involves representing the encoded genotype as a decision strategy for a SAT problem. The SAT problem represents the constraints to find a single feasible solution which can be translated into a feasible phenotype using a SAT solver based on the particular decision strategy. This decoding is done using the **D**avis-**P**utnam-**L**ogemann-**L**oveland (DPLL) algorithm which is capable of finding a feasible solution based on a decision strategy and can guarantee complete coverage of the feasible space. Each decision strategy corresponds to a point in the feasible design space. The genetic operators act on the decision strategy to generate new solutions. This approach has been demonstrated to perform well on hard-constrained discreet optimization problems, as shown in [5].

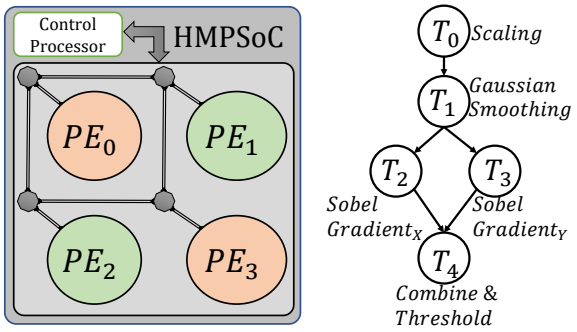
The SAT decoding approach put forth in [5] has been leveraged for system-level DSE. The work in [12] focuses on symmetry eliminating DSE for HMPSoC. A unique intermediate operating point, a constraint graph representation, is used to eliminate architecture symmetries. The tasks are considered to be preemptive and scheduled in system slots. The paper also introduces a learning mechanism based on a feasibility check of communication and binding constraints. The symmetry elimination leads to a costly feasibility check. The focus of this work is majorly centred around ensuring feasible message passing on the NoC.

The approach in [5] has encoding that is highly inefficient. Furthermore they used genetic operators similar to the ones stated in [13]. The inefficiencies in the encoding strategy are addressed in [14]. They put forth an efficient encoding strategy for the classic system synthesis problem. The genetic operators are clearly defined to maintain order and maintain the efficiency of encoding. The improvement over previous approaches is clearly highlighted.

Neither of these approaches focus on any cross-layer parameters, nor do they utilize the ability of the SAT decoding to eliminate infeasible portions of the search space during the DSE. [12] uses SMT like learning but the approach was focused on communication feasibility.

III. System Model

This section contains the details of the basic terms and models used in our DSE. The system model and synthesis is common to the one used in the state-of-the-art research on task mapping in low-power heterogeneous systems. [15] [16]



(a) Architecture (b) Application

Fig. 1: System Model

A. Architecture Model

For the purpose our problem formulation we consider a HMPSoC connected in a 2D mesh NoC topology. The NoC comprises of an network interface for each **P**rocessing **E**lement (PE) and the interconnecting links. X-Y routing is assumed for message passing [17]. The hop distance between the PE is determined by the routing mechanism. The hop distance is defined as the sum of the absolute values of the coordinates of the two PEs on the NoC. The hop distance ($hops$) determines the latency between two PEs. Fig. 1(a) shows an abstracted 2x2 NoC architecture model.

The messaging latency is modeled similar to that proposed in [18]. The bandwidth of the each message is fixed and the net bandwidth of each link in the NOC is considered to be large enough to avoid bandwidth conflicts between temporally overlapping messages on the link. We will also consider all links in the NoC to be similar. The latency will be a function of the size of the message (Msg_{size}), the bandwidth allocated to the message (B_m), the hop distance between the PEs and network specific variables such as the link width and the router delay at each node [15]. The calculation for the latency during routing is shown in Eq. (1).

$$Latency = (hops) * (Msg_{size})/(B_m * Link\_width) + (hops - 1) * (router\_delay) \quad (1)$$

The energy of routing will be calculated according to [18]. We can extrapolate the energy expenditure of the system based on the number of bits in each message and the hop distance from [18] considering circuit switched routing. We consider a wire length of $2mm$ between every core [15].The resulting energy consumption for communication in terms of the message size and the number of hops is as shown in Eq. (2). The constant values in the equation are derived using VHDL models of the NoC.

$$Energy_{message} = (Msg_{size}) * (100 * (hops) - 63)pJ \quad (2)$$

B. Application Model

The application model comprises of an application scenario which is essentially a set of independent application task graphs. A single application task graph is represented by a **D**irected **A**cyclic **G**raph (DAG) where the nodes represent the tasks and the edges represent the messaging data and the precedence constraints of the tasks. Fig. 1(b) shows the task graph for *Sobel Edge Detection*. The tasks correspond to execution of various stages of the application such as *Scaling*, *Gaussian Smoothing*, *Gradient Computation* and *Thresholding*. We assume the tasks are scheduled in a non-preemptive manner. The algorithm and the encoding used for the implementation of each task can result in varying power dissipation and execution time for the task. For instance, Gaussian Smoothing of an image can be implemented

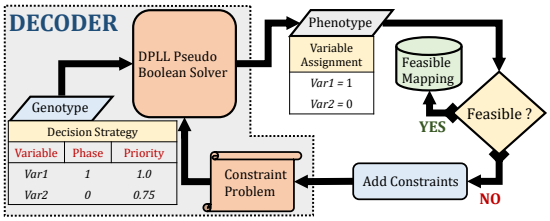


Fig. 2: Decoder

with varying window sizes for the convolution operation and each such design decision may result in varying performance for the task. We abstract the execution time and the power dissipation of each implementation of a task by the number of execution cycles and the activation factor resulting due to executing the implementation on a PE.

C. System Synthesis

As the system is considered to be a Soft-Real Time System, with no task deadlines or preemptions. The DVFS levels are discrete set of frequency-voltage pairs. The voltage is considered proportional to the frequency squared. A list of possible frequency-voltage pairs are associated with each type of PE. A task executing on a particular PE can choose between the available DVFS levels. The switching time between DVFS levels is ignored and considered to be zero. A feasible task mapping can be defined as a concrete strategy for binding and scheduling the applications onto a HMPSoC. In case of a power aware system, consideration of other cross-layer parameters is also a part of the task mapping process. Furthermore in our DSE we consider additional constraints on the total energy and execution time which lead to a smaller feasible design space. Each solution of our meta-heuristic search represents a particular mapping of the application scenario. The solution, called an operating point can be evaluated to ascertain feasibility. The feasibility check in the case of constrained optimization would be on the evaluated values of objectives. The feasible mappings would encode binding and scheduling while being aware of cross-layer parameters corresponding to each scheduled task.

IV. Proposed Methodology

A. Methodology

In case of a complex feasible space we can leverage the efficiency of modern PB/SAT solvers as a decoder and ensure that each solution obtained by the meta-heuristic lies within the feasible space. This would mean, that instead of eliminating infeasible solutions from the design space after evaluating them, we can manipulate the design space itself to eliminate infeasible solutions during the decoding process. Alternately in cases where the problem constraints cannot be learned apriori, the infeasible space needs to be learnt during the meta-heuristic by the introduction of additional constraints to the PB/SAT problem definition.We use the later, as user-defined constraints on the objectives cannot be be learned before evaluations to determine feasibility. Our approach focuses on using the decoder methodology introduced in [5], with efficient encoding similar to [14] for a robust DSE. We define the system synthesis problem as a PB constraint problem, where each feasible solution of the constraint problem, corresponding to an assignment of the involved variables, represents a task mapping. In the subsequent sections, the components of the meta-heuristic are explained in detail.

1) Decoding

The decoding process includes formulating the constraint problem for system synthesis and using a decision strategy to select a particular solution. Fig. 2 shows the components of the decoding methodology. A genotype, i.e. a decision strategy, is a list of *phase* and *priority* *value* associated with each variable in the PB problem. The $DecStrat$ (Decision Strategy) for each variable can be described as shown in Eq. (3). $Phase$ refers to a binary variable and $Priority \in [0, 1]$. The $Phase$ represents the expected value of the variable in the final assignment of the variables. The $Priority$ decides priority of initial assignments of the variables.

$$\forall V \in Vars : DecStrat(V) = (Phase, Priority) \tag{3}$$

The system synthesis problem can be described by a set of constraints. For task binding we can use a set of binary variables of the type $map(task, PE)$ where the value 1 would correspond to binding and 0 would correspond to not bound. For each task t in the application scenario, there will be a list of processing elements it can be mapped to called the PE_list_t. Then, the constraint for task binding can be represented as shown in Eq. (4). This constraint ensures that a task is mapped to one and only one PE.

$$\forall task \in Scenario \sum_{PE \in PE\_list_t} map(task, PE) = 1 \tag{4}$$

A discrete set of DVFS model are associated with each PE. Let $mode_{max}$ be the maximum number of modes of all the PEs in the HMPSoC. For each PE the number of DVFS modes can be represented by a integer $num\_modes(PE)$. The chosen DVFS mode is encoded as an integer value starting from 1 to $num\_modes(PE)$. A binary variable $DVFS_{Mode}(task, mode)$ corresponds to the mode of DVFS of the task. $mode$ corresponds to each task and is independently chosen regardless of the PE assignment. $mode$ can vary from 1 to $mode_{max}$. The constraint represented by Eq. (5) makes sure a task picks one and only DVFS mode to execute on. While Eq. (5) ensures a unique mode is assigned to a task but does not take in considerations of the PE it is mapped to. This can be included in our constraint problem by ensuring that the DVFS mode assigned can actually be reached on the mapped PE. This involves making the assignment of the DVFS mode aware of the mapping of each task. For ensuring that the mode assigned is a valid mode, the constraints shown in Eq. (6) are included in the problem formulation.

$$\forall task \in Scenario \sum_{mode <= mode_{max}}^{mode=1} DVFS_{Mode}(task, mode) = 1 \tag{5}$$

$$\forall task \in Scenario \sum_{mode <= mode_{max}}^{mode=1} mode * DVFS_{Mode}(task, mode) - \sum_{PE \in PE\_list_t} num\_modes(PE) * map(task, PE) <= 0 \tag{6}$$

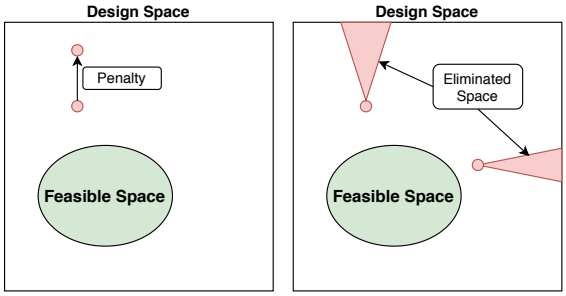
In order to include the exploration across different implementations of a task with varying activation factor values, we represent the each of the implementation of the task by an integer. Let the total number of implementations corresponding to a particular task be α. Each task implementation is represented by a number from 1 to α. We define a binary variable $active(task, \beta)$ to represent the choosing of the β implementation of the $task$. The constraint to ensure that that only one implementation is chosen for each task is shown in Eq. (7).

$$\forall task \in Scenario \sum_{\beta <= \alpha}^{\beta=1} active(task, \beta) = 1 \tag{7}$$

The constraint problem along with the decision strategy constitute the input for the PB solver. The PB solver works on a popular backtracking algorithm, DPLL. The basic skeletal structure of the DPLL algorithm is shown in Algorithm 1. This algorithm ensures that the variables with the higher priority in the decision strategy have the greatest chance of maintaining their phase values.

Algorithm 1 Back-Tracking based DPLL Algorithm

```
for V in DecStrat do
    V ← Phase(V)
    is_Feasible = checkFeasibility(V, constraints)
    if is_Feasible == True then
        Continue
    else
        Backtrack
        Phase(V) ← ¬Phase(V)
    end if
end for
return Assigned Values
```



(a) Penalty Function (b) Proposed Approach

Fig. 3: Differences in learning methodologies

2) Evaluation

The assignment of variables by the PB solver are processed during the evaluation. The evaluation includes converting the raw assignments to a viable task mapping using priority based scheduling. The priority of the processes is ascertained by precedence constraints. The obtained mapping is evaluated for its energy and execution time. Evaluation for other objectives can also be performed. This evaluation determines the feasibility of the solution. If the mapping is feasible, we store the mapping information. In case of infeasibility, as shown in Fig. 2, we add constraints to the design space, eliminating a portion of the infeasible space from the solution space.

3) Learning the feasible space

The infeasible space is learnt by the meta-heuristic during run-time. This learning has significant impact on the design space. This learning ability of our approach contributes to the improvement over popular techniques such as a penalty-function based approach. The constraint problem learns the infeasible space during run-time by the addition of constraints. These constraints not only remove the infeasible solution from the design space, but also leverage monotonocities to increase the amount of infeasible points excluded from the problem definition. In case a solution is evaluated with energy or execution-time values higher than the feasible values, we remove the solution from the design space. In case of infeasibility of a particular mapping, we eliminate the collective set of the activated assignments (Variables whose value is assigned as 1), $activeVars$, from the feasible space. Let $num\_activeVars$ be the number of activated variables, then the constraint added will be similar to that shown in Eq. (8). The monotonocites of the problem can also be exploited in a similar manner.

$$\sum_{V \in activeVars} V < num\_activeVars \tag{8}$$

Fig. 3 highlights the fundamental difference between a penalty based approach and our learning methodology in the case where the feasible space is a subset of the design space. When a penalty-based meta-heuristic finds a point outside the feasible region, the penalty on the objectives affect the quality of the solutions adversely.The penalised individual changes position in the objective space as shown in Fig. 3(a). The design space remains intact for all further individuals. On the other hand, a constrained decoder eliminates a portion of the infeasible space from the design space. Over multiple iterations, this restricts the solutions to only the feasible region. Fig. 3(b) demonstrates visually how the infeasible space is removed from the design space. Additionally, penalty-based approaches can still be used along with the constrained decoder approach.

4) Metaheuristics-based Optimization

The previous sections provide plug-and-play components to meta-heuristic methods such as **G**enetic **A**lgorithms (GA). The decoding methodology and the learning methodology can be integrated in classic genetic algorithm approaches. The Decision Strategy acts as an individual. Special genetic operators change the decision strategy while maintaining the efficiency of the encoding. The Genetic operators ensure fair and complete coverage of the design space. The efficacy of the meta-heuristic is heavily dependent on the ability of genetic operators to avoid converging to a locally optimal solution and to ensure continuing improvement of the Pareto-front. The variables in decision strategy are grouped together both during the initial assignment of the variables and the while being operated on by the genetic operators. The $map(task, PE)$ variables are grouped together for each $task$ in the $Scenario$. Similarly the $DVFS_{Mode}(task, mode)$ and the $active(task, \beta)$ are grouped together for each $task$ in the $Scenario$. Hence there are three groups associated with each task. The following conditions apply to each group:

- Only the phase of one variable in each group can be 1, each other variable has a phase equal to 0
- The relative priorities in a group are maintained throughout the meta-heuristic

The mutate operation is multi point and operates on a group as a whole. The operation mutates the variables maintaining the above conditions. The operation is similar to the one explained in [13]. The crossover operation is done simply by exchanging the decision strategy of a complete group between the two individuals. NSGA-2 [19] is used as the selection algorithm to increase the number of points in the pareto front.

V. Experiments and Results

A. Experiment Setup

The DSE is implemented as an extension of the DEAP framework [20]. The SAT4J [21] Solver, a SAT/PB solver implemented in Java was used as the decoder. The system data pertaining to the cores and the tasks run on the cores is obtained from **E**mbedded **S**ystems **S**ynthesis Benchmark **S**uite (E3S) [22].

1) Architecture

For each message in the task graph we allocate 100 Mhz of Bandwidth. We consider a link width of 8 bits. The PE type considered for this experimental setup are from the E3S benchmark. We consider 3 AMD cores with 133Mhz,400Mhz and 550Mhz base frequencies, 2 IBM cores with 266Mhz and 500 Mhz base frequencies and a NEC VR core with a base frequency of 167 Mhz. Each PE in the HMPSoC is of either of the six types. The bandwidth of each message is limited to 100Mhz on each link in the PE. A constant delay is added between each hops. The tests are run for 2x2,4x4 and 8x8 HMPSoC. The DVFS levels are introduced by scaling the frequency of the core. The values of power usage and the execution time of the tasks are scaled accordingly considering only dynamic power and a cubic relation between the frequency and the power consumption of the processor.

2) Application

The tests are run on the application scenarios from the E3S benchmark. The tests are also run on randomly generated task graphs using TGFF [23] to show scalability w.r.t. application size. The benchmark contains 40 different types of basic tasks. To account for the difference in power usage of different implementations, the power and the execution time of the tasks are scaled accordingly corresponding to its activation factor.

3) Metaheuristics

We use a multi-objective GA for our experiments. There are two implementations of the GA, one based on a penalty-based infeasibility check (Normal DSE) and another based on our proposed constraint decoder method (PB strat DSE). Each application scenario is explored with a population size of 100 individuals for a total of 250 generations. The objectives of the DSE are total execution time and total energy consumption. Each application has a unique energy and time profile. To emulate realistic user-defined constraints on the maximum energy and execution time, we use single objective optimizations to find the lowest possible value of each objective. These obtained lowest values are then scaled to define the feasible region of the design space. This enables us to have a application specific limit on the user constraints.

4) Performance metrics

To measure the performance of the GA we compare the Pareto-fronts of both, Normal and the PB strat based DSE. The Pareto-front is saved during each generation. The final Pareto-fronts obtained by both solutions are compared. The quality of the solutions obtained by the DSE is measured by computing the hypervolume of the Pareto-front. The quality of the divergence of the DSE to find a wide range of operating points is compared by comparing the number of points in the Pareto-front. The hypervolume noted in each of the tables is the ratio of the hypervolume of the PB-strat DSE divided by the hypervolume obtained by the Normal DSE. This relative hypervolume is a indicator of the relative quality of the approaches. A hypervolume less than 1 means that the Normal approach performed better and vice-versa.

B. Results

1) Unconstrained Optimization

For unconstrained optimization, both the approaches should perform equally well. A comparison between the two approaches can demonstrate the efficiency of encoding of the PB strat. As we can see in TABLE I the similarities in the values obtained during DSE, for a 4x4 HMPSoC without any constraints on the objectives, illustrates the efficiency of encoding of the PB strat.

TABLE I
UNCONSTRAINED DSE

Application Domain	No. of Tasks	Hypervolume(Relative) (PB-strat)/Normal	points in PF (For PB-strat)	points in PF (For Normal)
auto-industry	24	1.014	29	51
consumer	12	1.103	107	96
telecom	30	1.133	136	49
networking	13	1.115	8	13
office-automation	5	1.021	61	89

2) Constrained Optimization

The user constraints lead to a focused DSE. The constraints on the problem definition need to be complete before the PB strategy shows improvement. The results obtained when only a single objective is constrained can be observed in TABLE II and TABLE III. The results do not show any clear improvement.

TABLE II
TIME CONSTRAINED DSE

Application Domain	Hypervolume(Relative) (PB-strat)/Normal	points in PF (For PB-strat)	points in PF (For Normal)
auto-industry	0.97	108	129
consumer	1.038	41	23
telecom	1.023	24	11
networking	1.020	43	27
office-automation	1.023	7	7

TABLE III
ENERGY CONSTRAINED DSE

Application Domain	Hypervolume(Relative) (PB-strat)/Normal	points in PF (For PB-strat)	points in PF (For Normal)
auto-industry	0.990	82	69
consumer	1.001	166	141
telecom	1.016	96	112
networking	0.975	3	3
office-automation	0.993	158	109

3) Cross-Layer Parameters

A fully constrained DSE provides us with a clearer understanding of how the constrained decoding works. Adding Cross-layer design choices, namely DVFS and activation factor lead to a larger design space. A penalty-based GA in unable to explore the complete design space. It is much more likely to get stuck in a local optima and in constrained optimization, getting out of the local optima is harder due to the size of the infeasible space. In fact, as any solution away from the local optima that does not lie in the feasible space is less likely to be in the Pareto-front, due to the penalty function.

TABLE IV
FULLY CONSTRAINED DSE ON 4X4 HMPSOC

Application Domain	Hypervolume(Relative) (PB-strat)/Normal	points in PF (For PB-strat)	points in PF (For Normal)
auto-industry	∞	90	0
consumer	1.36	35	19
telecom	∞	0	18
networking	1.402	24	27
office-automation	∞	28	0

TABLE IV tabulates the results of a fully constrained GA. The maximum number of implementations for a task is set to 10 and the maximum number of dvfs modes is also set to 10. The PB strat outperforms the normal approach by a significant margin. In the cases where the Pareto-front is a non-empty set the number of points in the Pareto-front is much larger in the case of the PB solver based strategy. This can be attributed to a wider covering of the design space. Fig. 4 shows an example of how the Pareto-front of the PB strat covers a wider range of operating points that the Normal approach.

4) Architecture Scaling

The constraint decoding approach scales well with increase in the size of the problem definition. Increasing the resolution of the cross-layer parameters and scaling the HMPSoC, both contribute to the increase of the design space. The results for the run for larger HMPSoC of size 8x8 for a maximum of 10 DVFS modes and 10 implementations can be observed in TABLE V

VI. CONCLUSION

This paper explores a constrained decoding methodology to perform realistic Cross-layer DSE for heterogeneous embedded systems. A set of non-dominated points are found within design space which represent optimal mappings of an application scenario on a physical

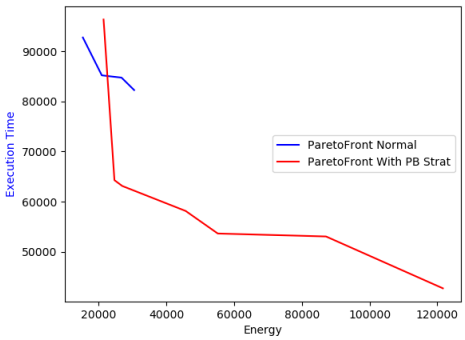


Fig. 4: Pareto-Front for *networking* in TABLE IV

TABLE V
8X8 HMPSOC - CONSTRAINED POWER AWARE DSE

Application Domain	Hypervolume(Relative) (PB-strat)/Normal	points in PF (For PB-strat)	points in PF (For Normal)
auto-industry	1.851	88	21
consumer	1.228	61	46
telecom	∞	22	0
networking	1.200	32	24
office-automation	1.486	25	14

HMPSoC. The DSE is performed with considerations of user constraints on the objectives. The constrained decoder-based approach is better suited for heavily constrained system synthesis.

The approach explored is limited by the solving capabilities of modern PB solvers, particularly PB solvers using backtracking algorithms. The PB solver puts a upper limit on the number of variables and constraints allowed in the system. The time overhead of a PB solver needs to be considered against its benefits. Novel ways to exploit the problem definition for monotonousness can lead to better quality of solutions. Increasing the amount of run-time feedback to the decoder equates to faster learning of the feasible space. A learning based approach to constrained decoding for system synthesis performs better than or at least as good as state-of-the-art DSE. The approach also scales well with increase in problem size.

Further work on constrained decoding could include a hybrid approach comprising of a well defined constraint problem together with a run-time learning strategy.

REFERENCES

[1] T. Blickle, et al. System-level synthesis using evolutionary algorithms. *Design Automation for Embedded Systems*, 3(1):23–58, 1998.

[2] A. K. Singh, et al. Mapping on multi/many-core systems: survey of current and emerging trends. In *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pages 1:1–1:10. ACM, 2013.

[3] A. K. Singh, et al. A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems. *ACM Comput. Surv.*, 50(2), April 2017.

[4] A. D. Pimentel. Exploring exploration: A tutorial introduction to embedded systems design space exploration. *IEEE Design Test*, 34(1):77–90, 2017.

[5] M. Lukasiewycz, et al. Sat-decoding in evolutionary algorithms for discrete constrained optimization problems. In *2007 IEEE Congress on Evolutionary Computation*, pages 935–942, 2007.

[6] M. Bambagini, et al. Energy-aware scheduling for real-time systems. *ACM Transactions on Embedded Computing Systems*, 15:1–34, 01 2016.

[7] K. Li. Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels. *J. Parallel Distrib. Comput.*, 95(C):15–28, September 2016.

[8] L. Mo, et al. Energy-quality-time optimized task mapping on dvfs-enabled multicores. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2428–2439, 2018.

[9] N. Kumar et al. A ga based energy aware scheduler for dvfs enabled multicore systems. *Computing*, 99(10):955–977, 2017.

[10] H. Ali, et al. Energy efficient heuristic algorithm for task mapping on shared-memory heterogeneous mpsocs. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1099–1104, 2018.

[11] S. Koziel et al. A decoder-based evolutionary algorithm for constrained parameter optimization problems. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, PPSN V, page 231–240, Berlin, Heidelberg, 1998. Springer-Verlag.

[12] T. Schwarzer, et al. Symmetry-eliminating design space exploration for hybrid application mapping on many-core architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(2):297–310, 2018.

[13] M. Lukasiewycz, et al. A feasibility-preserving local search operator for constrained discrete optimization problems. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1968–1975, 2008.

[14] V. Richthammer et al. Efficient search-space encoding for system-level design space exploration of embedded systems. In *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 273–280, 2019.

[15] A. Weichslgartner, et al. Daarm: Design-time application analysis and run-time mapping for predictable execution in many-core systems. In *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, 2014.

[16] A. K. Singh, et al. Communication-aware heuristics for run-time task mapping on noc-based mpsoc platforms. *Journal of Systems Architecture*, 56(7):242–255, 2010.

[17] S. D. Chawade, et al. Review of xy routing algorithm for network-on-chip architecture. *International Journal of Computer Applications*, 43(21):975–8887, 2012.

[18] P. T. Wolkotte, et al. Energy model of networks-on-chip and a bus. In *2005 International Symposium on System-on-Chip*, pages 82–85, 2005.

[19] K. Deb, et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[20] F.-A. Fortin, et al. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[21] D. L. Berre, et al. Sat4j: A satisfiability library for java. 2005.

[22] R. Dick. Embedded systems synthesis benchmark suites (e3s). *http://ziyang.eecs.umich.edu/ dickrp/e3s/*.

[23] R. P. Dick, et al. Tgff: task graphs for free. In *Proceedings of the Sixth International Workshop on Hardware/Software Codesign. (CODES/CASHE'98)*, pages 97–101, 1998.