Efficient System-Level Design Space Exploration for High-Level Synthesis using Pareto-Optimal Subspace Pruning

Yuchao Liao, Tosiron Adegbija, and Roman Lysecky
Electrical and Computer Engineering
University of Arizona
Tucson, Arizona, USA
{yuchaoliao,tosiron,rlysecky}@arizona.edu

ABSTRACT

High-level synthesis (HLS) is a rapidly evolving and popular approach to designing, synthesizing, and optimizing embedded systems. Many HLS methodologies utilize design space exploration (DSE) at the post-synthesis stage to find Pareto-optimal hardware implementations for individual components. However, the design space for the system-level Pareto-optimal configurations is orders of magnitude larger than component-level design space, making existing approaches insufficient for system-level DSE. This paper presents *Pruned Genetic Design Space Exploration (PG-DSE)*—an approach to post-synthesis DSE that involves a pruning method to effectively reduce the system-level design space and an elitist genetic algorithm to accurately find the system-level Pareto-optimal configurations. We evaluate PG-DSE using an autonomous driving application subsystem (ADAS) and three synthetic systems with extremely large design spaces. Experimental results show that PG-DSE can reduce the design space by several orders of magnitude compared to prior work while achieving higher quality results (an average improvement of 58.1x).

KEYWORDS

System-level optimization, design space exploration, high-level synthesis, subspace pruning, embedded system

ACM Reference Format:
Yuchao Liao, Tosiron Adegbija, and Roman Lysecky. 2023. Efficient System-Level Design Space Exploration for High-Level Synthesis using Pareto-Optimal Subspace Pruning. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567841

1 INTRODUCTION

High-level synthesis (HLS) has increased in popularity in recent decades for its ability to significantly improve productivity in the design of complex digital systems. HLS allows designers to specify systems at a high abstraction level, decoupled from low-level circuit details, and use HLS tools to generate an optimized low-level hardware description of the target system. Using existing HLS tools, designers can develop application-specific embedded systems using high-level languages (e.g., C/C++) and map them to hardware register-transfer level (RTL) languages (e.g., Verilog, VHDL), thereby improving design productivity and reducing the design time/cost [8]. One of the most important features of HLS is design automation and diversification, wherein system-level and component-level synthesis directive options (e.g., target frequencies, resource bindings, loop unrolling factors) can be provided at the pre-synthesis (pre-HLS) stage based on the requirements of the application-specific embedded system. HLS can then generate multiple alternative hardware implementations that can be explored to satisfy different design constraints. As the number of synthesis directives increases, the design space of the resulting hardware implementations increases exponentially, raising new challenges for the HLS design process [4].

Much prior work has focused on design space exploration (DSE) in HLS [15]. Given the multi-objective and potentially conflicting nature of the design criteria in complex digital systems (e.g., performance vs. power), DSE must often find Pareto-optimal configurations, making the DSE process more challenging [13]. Many HLS DSE methodologies aim to find Pareto-optimal configurations at the pre-, post-, or both synthesis stages by combining different synthesis directives. For example, DSE methodologies have been proposed using heuristic-based [4] or machine learning-based [5] algorithms to identify the Pareto-optimal fronts. However, a crucial gap still remains in the state-of-the-art of HLS DSE. Current HLS methodologies do not bridge the gap between pre-HLS DSE and post-HLS multi-component system-level DSE. Importantly, none of the current approaches focus on system-level DSE for complex, timing-constrained, application-specific embedded systems. Current HLS DSE methodologies target systems with only one or a few components (e.g., encoder, neural networks) or multicycle computations (MCC) (e.g., matrix multiplication, median filter)[18].

To design complex embedded systems that satisfy their timing constraints using HLS, DSE must follow a holistic system-level approach that considers the individual system components and the combinations of their different implementation alternatives. In this work, we focus on embedded systems that may be comprised of several components, each of which may consist of multiple multicycle computations (MCCs). Each MCC may have multiple design alternatives with different implications for the system's performance, area, and energy. Individual exploration of the different components [3] may be easier but would leave much optimization potential untapped if interactions between the components, MCCs, and MCC design alternatives are not explored. A system-level DSE, on the other hand, results in a massive design space that necessitates new approaches for tractable exploration. For instance, a 10-component

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01.
https://doi.org/10.1145/3566097.3567841

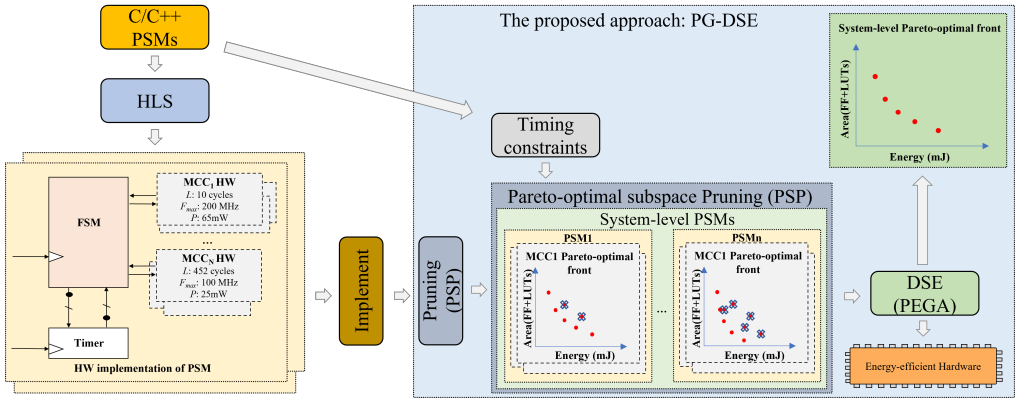


Figure 1: The proposed PG-DSE methodology integrates a Pareto-optimal subspace pruning (PSP) algorithm and design space exploration (DSE) using a Pareto-optimal elite genetic algorithm (PEGA) to find system-level Pareto-optimal configurations

autonomous driving subsystem—a sample target application of this work—may have a design space with 7.12E+66 possible solutions.

This paper presents a new HLS DSE approach, called *pruned genetic design space exploration (PG-DSE)*, that targets complex timing-constrained multi-component, multi-MCC embedded systems. PG-DSE combines a pruning algorithm called *Pareto-optimal subspace pruning (PSP)*—to substantially reduce the system-level design space—and a genetic search algorithm, called *Pareto-optimal elite genetic algorithm (PEGA)* to accurately find the system-level Pareto-optimal configurations. PEGA includes novel features, like *variable Pareto-optimal elitism* and a genetic encoding approach that ensures that high quality configurations are rapidly found during the exploration process. We evaluate PG-DSE using a complex autonomous driving application subsystem and three synthetic systems with different design spaces ranging in size from 2.61E+11 to 1.49E+104. On average, the PSP algorithm substantially reduced the design space by 1.32E+44x without eliminating the best configurations from the design space. The proposed PG-DSE approach (PEGA+PSP) rapidly, effectively, and accurately found the system-level Pareto-optimal configurations, improving the quality of the results by an average of 58.1x compared to a recent prior work that used a genetic algorithm for DSE.

2 BACKGROUND AND RELATED WORK

HLS tools usually enable system-level and component-level design metrics in the behavioral descriptions to generate various hardware implementations. To fully exploit the optimization benefits of application-specific hardware, the design space of these synthesis directives must be explored to determine the solutions that best satisfy the target design constraints.

Existing HLS DSE approaches typically use either heuristics (e.g., genetic algorithms [4]) or machine learning-based approaches in the pre-HLS [5] or post-HLS stage [10] to find the optimal or Pareto-optimal configurations in single- or multi-component systems. However, none of these current approaches explore the complex design space of multi-component systems in which the combinations of the components and their design alternatives can result in extremely large design spaces. Furthermore, none of these works consider a system with precise timing constraints that must be accounted for in the DSE process. Some prior HLS approaches that consider precise system-level timing constraints (e.g., [9, 10]) do not include DSE for complex systems with large design spaces.

Our work addresses these gaps via a post-HLS DSE approach that tightly incorporates the system's timing information into the DSE process. Our approach also incorporates the complexity of multi-component systems whose individual components may have different implementation alternatives with different impacts on the resulting system-level designs.

To make the DSE process more tractable, pruning methods have been used to reduce the design space. Some pruning methods reduce the design space before DSE [10, 19], while others apply pruning to the resulting configurations after DSE [17]. Our work incorporates an a priori pruning algorithm into the proposed DSE approach to enable the rapid exploration of extremely large design spaces. In addition, to prevent the violation of timing constraints, our pruning algorithm explicitly incorporates precise timing information into the pruning process to ensure that only suboptimal configurations are eliminated from the design space.

3 PRUNED GENETIC DESIGN SPACE EXPLORATION (PG-DSE)

Fig. 1 depicts a high-level overview of the proposed PG-DSE approach. PG-DSE is distinguished from existing methods by focusing on timing-constrained complex multi-component embedded systems, wherein the different components may have multiple multicycle computations (MCCs). To represent such systems and their timing constraints, we use *Periodic State Machines (PSMs)* [7, 9] as our modeling formalism. PSMs are similar to finite state machines (FSM), which are common for modeling the behavior of computer systems. However, PSMs modify FSMs to enable the specification of time-triggered execution defined by a fixed period, global time, clock constraints, and time-driven events. The fixed period is a constraint that allows all the MCCs within a PSM to execute in one clock cycle. As such, PSMs enable the high-level specification of complex embedded systems with precise timing constraints, while maintaining the state-based features of FSMs that allow for RTL translation of system specifications.

The PSM specification can be written in a high-level language (e.g., C, C++) and HLS can be used to generate the RTL implementation of the PSMs (in the form of FSMs coupled with custom

Algorithm 1: Pareto-optimal subspace pruning (PSP)

```
   Input: c, tc, w, f_max, and A for each MCC, M alternative, am, in each PSM P, F_size, T
   Output: pruned PSM for each frequency combination f_comb
1  E, fs, s, ← calculateParameter(c, f_max, tc, w)
2  f_comb = {(f_1, ..., f_F_size), ..., (f_1, ..., f_F_size)} ← calculate a set of frequency
     combinations
3  for i ← 1 to size(f_comb) do
4      for each am in each M in each P do
5          f_am ← closest frequency in f_combi
6          if no valid am in M then
7              i ← i + 1, continue
8          else
9              if all am is valid then
10                 E_am ← f_am × s
11                 sort all am in each M based on Pareto-optimal E_am and A
12             end
13         end
14     end
15     if all am in each m in each P sorted then
16         output new PSM
17     end
18     i ← i + 1
19 end
```

datapaths). Within each PSM, the MCCs are extracted to generate their design alternatives through HLS. MCCs are computations (e.g., medium filter, matrix multiplication) that dominate the system's execution time, and whose designs have significant impacts on the system-level design efficiency. The MCC's hardware implementation alternatives are generated using state-of-the-art tools (e.g., Xilinx Vitis HLS) based on different features, like the critical path, maximum frequency, loop unrolling factor, etc. The combination of the different PSMs and their MCC alternatives forms the design space that must be explored.

Given the design space, PG-DSE is then applied to identify Pareto-optimal solutions that satisfy different design objectives. PG-DSE involves a two-step process: first, the design space is significantly reduced using the proposed *Pareto-optimal subspace pruning (PSP)* algorithm. Thereafter, the pruned design space is explored using the *Pareto-optimal elite genetic algorithm (PEGA)*. The following subsections describe these algorithms in detail.

3.1 Pareto-optimal subspace pruning (PSP)

The goal of the pruning process is to preemptively eliminate sub-optimal solutions in the subspace of MCC design alternatives to make DSE more tractable and less time consuming. By pruning the design space, the DSE algorithm can focus on quickly identifying system-level solutions that improve the target objective functions (e.g., area/energy). For timing-constrained embedded systems, the proposed PSP algorithm aims to prune solutions that do not satisfy the timing constraints.

Algorithm 1 presents the pseudocode for the PSP algorithm. The algorithm takes as input the execution cycles c, maximum frequency f_{max}, critical path tc, power w, and area A for each MCC alternative am. In addition, two critical input constraints for PSP are the range of allowed clock frequencies F_{size} for the embedded system, constrained by the target hardware—we focus on FPGAs—and the assigned period T for each PSM (based on the timing constraint). PSP outputs the pruned design space for the input system. To prune the design space, PSP first calculates the energy E, a scaled frequency fs, and a scaling factor s. The scaled frequency is the minimum frequency at which an MCC alternative can run under T, and the scaling factor is calculated as: $\frac{w}{f_{max} \times T}$ which is used to estimate the new energy when applying a new frequency to an MCC alternative. Next, with F_{size}—the global minimum and maximum frequency range for PSMs—PSP explores a set of possible frequency combinations f_{comb} (lines 1 - 2) by iterating through them. In each iteration i, PSP goes through every MCC alternative, finds the closest frequency in the f_{comb_i} to each MCC alternative's minimum frequency (lines 3 - 5), and assigns the new frequency f_{am} to the MCC. If any MCC alternative cannot find a valid f_{am} in f_{comb_i}, PSP jumps to the next f_{comb_i} (lines 6 - 7). Otherwise, if f_{am} is valid for every MCC alternative, PSP calculates the current energy E_{am} for each MCC alternative by multiplying f_{am} and s. Next, using E_{am} and A, PSP prunes MCC alternatives to find Pareto-optimal alternatives in each MCC (lines 8 - 13). Finally, PSP outputs a set of pruned PSMs corresponding to each valid frequency combination (lines 15 - 17). After the PSP process, the Pareto-optimal elite genetic algorithm is then applied to identify the system-level Pareto-optimal solutions.

3.2 Pareto-optimal elite genetic algorithm (PEGA)

Although the PSP algorithm substantially reduces the MCC subspace, the system-level design space remains very large. Using the PSM formalism ensures that the pruned solutions have already been determined to satisfy the timing constraints. But we still have a multi-objective optimization problem, given the potentially conflicting objective functions of energy and area. To address this challenge, we employ a genetic algorithm in the PG-DSE approach.

Genetic algorithms, which are inspired by the process of natural selection, have been commonly used for multi-objective HLS DSE problems [4, 12]. In general, a genetic algorithm (GA) involves a population of candidate solutions that are iteratively evolved toward better solutions by evaluating the fitness of individual solutions, stochastically selecting more fit solutions, and randomly mutating and combining solutions to form new generations. Genetic algorithms are appropriate for our work given the multimodal search space and the need to explore and exploit the tradeoffs between solutions in the search space for conflicting objectives. Furthermore, GAs, given their population-based search approach, can offer more design flexibility by producing multiple designs that can satisfy different user-specified design constraints.

Our proposed Pareto-optimal elite genetic algorithm (PEGA) incorporates three important novel features to improve efficiency. First, we use *variable Pareto-optimal elitism*, whereby Pareto-optimal solutions in one generation are carried over to the next generation. This eliminates the constraint of a static elite population size and ensures that the quality of solutions does not decrease with subsequent generations. Second, we use a genetic representation in which the input structure contains important information that is uniquely suited for the target systems' complexity. We define each MCC alternative with its assigned frequency from frequency combinations (Section 3.1) as a gene; the MCC alternative forms the first part of a chromosome C_r, and the frequency combination forms the second part of C_r. Third, we leverage the inherent parallelism of GAs by separating the pruned design space according to the valid frequency combinations for the target FPGA. When PSP prunes the design space and generates new PSMs based on each frequency

Algorithm 2: Pareto-optimal elite genetic algorithm (PEGA)

Input: f_{comb}, s, p_s, p_c, p_m, k, and E, A from pruned PSMs.
Output: Pareto-optimal system-level configurations

```
1  for each valid f_comb and corresponding pruned PSM do
2      P_k ← initial a population of k randomly-generated individuals
3      while !terminate condition do
4          new P_k ← ParetoElite(P_k, eliteP_k)
5          S_k ← fitness(P_k)
6          PS_k ← select(P_k, S_k, p_s)
7          PC_k ← crossover(PS_k, p_c)
8          PM_k ← mutate(PC_k, p_m)
9      end
10     savePareto(eliteP_k)
11 end
12 function ParetoElite (P_k, eliteP_k):
13     P_k ← Insert old eliteP_k into P_k
14     new eliteP_k ← find Pareto-optimal chromosome Cr in E and A in P_k
15     return P_k
```

combination, each new PSM can be assigned to a different thread during the DSE process, thereby reducing the execution time for exploring large design spaces.

Algorithm 2 depicts the pseudocode of the Pareto-optimal elite genetic algorithm (PEGA). PEGA takes as input the frequency combinations f_{comb}, select rate p_s, crossover rate p_c, mutation rate p_m, population size k, and energy E, area A from the pruned MCC alternatives *am* in each PSM P. The algorithm outputs the Pareto-optimal system-level configurations. PEGA starts by iterating through each valid f_{comb} and its corresponding P (line 1). In each P, PEGA randomly generates an initial population P_k with size k (line 2). Next, PEGA iterates until a terminate condition is met. The terminate condition can be a predefined number of generations, which we use in this work, or a threshold of the quality of results (e.g., the distance from a reference set). In each generation, for the $ParetoElite$ function, PEGA goes through the current population P_k and the previous generation's elite population $eliteP_k$ (initially an empty set) to find and save the current Pareto-optimal C_r to the new $eliteP_k$ set. Unlike typical elite functions with a fixed elite member size [11], PEGA stores every Pareto-optimal C_r. $ParetoElite$ outputs the new population P_k. If the new P_k exceeds size k, any C_r with the least fit score is discarded until the population size constraint is satisfied (lines 12 - 15). We use a cost function comprising the weighted sum of energy and area to calculate the fitness of the population (line 5). The rest of PEGA contains selection, crossover, and mutation (lines 6 - 8). PEGA uses the roulette-wheel method for selection [16]; for crossover, PEGA generates children C_r until the population size equals k; and mutation randomly selects and changes the selected MCC alternative. Finally, after a terminate condition is met, $savePareto$ function saves the last $eliteP_k$ from each P (line 10). P_s, p_c, p_m, and k are hyperparameters that can be tuned to improve the search process for the target design space. We used $p_s = 0.5$, $p_c = 0.7$, $P_m = 0.5$ in our experiments.

4 EXPERIMENTS

To evaluate PG-DSE, we used a complex autonomous driving application subsystem (ADAS) [6] and three synthetic systems with different design space sizes. Fig. 2 shows ADAS' system-level block diagram, which consists of five main parts: *sensors*, *localization and perception*, *method fusion*, *planning*, and *vehicle control*. Apart from the sensors, the rest of the subsystem is implemented using PSMs.

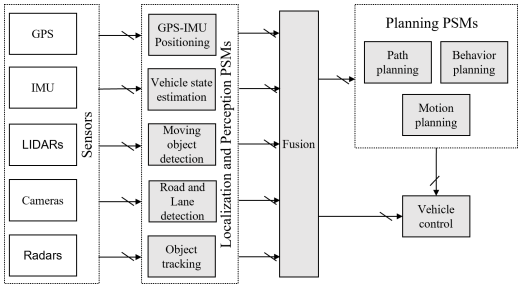


Figure 2: Block diagram of the autonomous driving application subsystem (ADAS). Shaded blocks are the components implemented as PSMs in our experiments.

In total, the ADAS comprises ten PSMs, 52 MCCs, and 244 MCC alternatives.

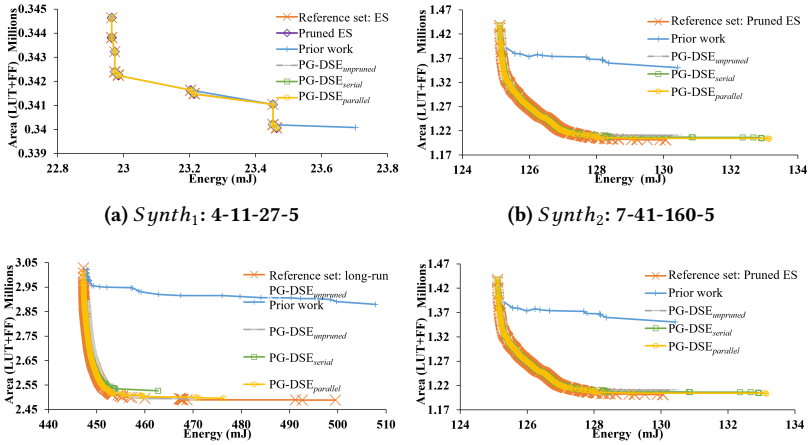
Similarly, we generated three synthetic systems with different characteristics to represent varying levels of complexity. We built a *PSM generator*, a custom tool to randomly generate a combination of implementation results and timing constraints for synthetic systems. The tool contains a database of essential implementation data generated using HLS and implemented with Xilinx Vivado targeting an Artix-7 FPGA. These data include latency (ns), energy (mJ), scaled frequency (MHz), etc. and are based on real complex multi-component embedded systems like ECG biometric authentication system [2], asthma monitoring system [14], wearable pregnancy monitoring system [1], etc. Table 1 shows the system information for the synthetic systems ($Synth_{1-3}$) and ADAS, depicted as: <number of PSMs>-<total number of MCCs>-<total number of MCC alternatives>-<number of frequency candidates>.

The PG-DSE approach is implemented in C++. We first evaluate the design space reduction achieved by PSP. Next, we evaluate the quality of results (QoR) for the solutions resulting from using PG-DSE in comparison to the state-of-the-art, represented by a genetic algorithm recently presented in [4]. For robust evaluation, we created three variants of PG-DSE: 1) PG-DSE without PSP (PG-DSE$_{unpruned}$), to evaluate any performance loss resulting from a priori subspace pruning; 2) PG-DSE with serial PEGA (PG-DSE$_{serial}$); and 3) PG-DSE with parallelized PEGA (PG-DSE$_{parallel}$).

To serve as a base for our experiments, we used different reference sets of Pareto-optimal configurations because exhaustive search (ES) was prohibitive for some of the design spaces due to their sizes. For instance, ES for $Synth_1$ parallelized over 44 cores took ~12 hours, whereas ES for $Synth_2$ or $Synth_3$ would have taken 8.17E+34 and 2.56E+89 years, respectively. Instead, we used ES for $Synth1$; for $Synth_2$, we applied PSP and used ES on the resulting design space; for $Synth_3$ and ADAS, we ran PG-DSE$_{unpruned}$ for longer than normal—two hours per run—and averaged the results over five runs.

Table 1: Design space sizes before and after applying PSP algorithm. PSP reduced the size by an average of 1.32E+44x.

System Info	Design space		
	Before PSP	*After PSP*	*Improvement*
$Synth_1$: 4-11-27-5	2.61E+11	96	2.72E+09x
$Synth_2$: 7-41-160-5	1.71E+49	3.85E+11	4.43E+37x
$Synth_3$: 10-85-347-8	1.49E+104	9.88E+23	1.51E+80x
ADAS: 10-52-244-10	7.12E+66	4.24E+17	1.68E+49x
Geo. mean	8.29E+57	6.27E+13	1.32E+44x



(a) $Synth_1$**: 4-11-27-5** **(b)** $Synth_2$**: 7-41-160-5**

(c) $Synth_3$**: 10-85-347-8** **(d)** *ADAS*: **10-52-244-10**

Figure 3: Energy and area of the Pareto-optimal system-level configurations for three synthetic systems and autonomous driving application subsystem using the reference set (exhaustive search or an approximation for intractable design spaces), prior work's genetic algorithm [4], PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$.

5 RESULTS

5.1 Subspace pruning using PSP

Table 1 shows the design space for the three synthetic systems and ADAS before and after we applied the PSP algorithm. The design space before PSP is the exhaustive set of solutions in the search space. One of the first things we observe is that as the complexity of the design space increases, going from $Synth_1$ to $Synth_3$, PSP's pruning improves. PSP is able to prune a larger portion of the subspace for larger input design spaces. For instance, although $Synth_3$ and ADAS both have 10 PSMs, $Synth_3$ has more MCCs and MCC alternatives in each PSM, resulting in a larger design space. As a result, PSP is able to prune more subspaces in each of $Synth_3$'s MCCs compared to the ADAS. Overall, PSP significantly reduced the design space by an average of 1.32E+44 times. However, the true quality of a pruning algorithm lies in its ability to reduce the design space without eliminating potential high quality solutions from the search space. Thus, in the following, we evaluate the quality of results achieved using the PSP algorithm with respect to the system-level Pareto-optimal configurations.

5.2 Design space exploration using PG-DSE

With the help of the PSP algorithm, the design space becomes more tractable to find the Pareto-optimal system-level configurations using PEGA. To evaluate PG-DSE and determine whether there is any loss from the PSP method, we explored the three variants of the proposed algorithm (Section 4) for our experimental systems in comparison to prior work [4]. Fig. 3 shows the Pareto-optimal system-level configurations resulting from prior work, PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$. Each system is compared to its reference set as described in Section 4.

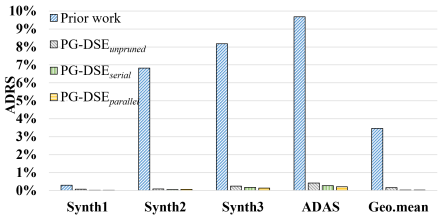


Figure 4: Average distance to reference set (ADRS) score for five runs of prior work, PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$.

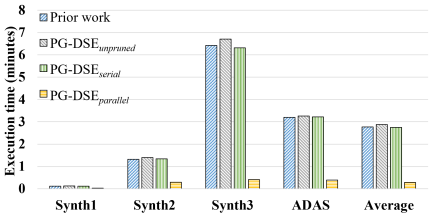


Figure 5: Average execution time for five runs of prior work, PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$.

Across the three variants, PG-DSE was able to generate identical or close Pareto-optimal solutions to the reference sets for different design space sizes. For a small design space like $Synth_1$ (Fig. 3a), the pruned exhaustive search (pruned ES), PG-DSE$_{serial}$, and PG-DSE$_{parallel}$ generated the same Pareto-optimal front as the reference set. The fact that pruned ES and PG-DSE achieved the same Pareto-optimal front as exhaustive search shows that there was no performance loss resulting from pruning using the PSP algorithm. Prior work and PG-DSE$_{unpruned}$ yielded a few Pareto-optimal configurations with higher energy or larger area than the reference set. For the larger design spaces (Fig. 3b, 3c, and 3d), the reference set clearly yielded a lower Pareto-optimal curve (i.e., with lower energy and area) than the DSE algorithms. However, in general, PG-DSE consistently yielded better curves than prior work.

To further evaluate PG-DSE, we quantified each algorithm's QoR using the *average distance to reference set (ADRS)*. A similar metric has been used in prior work [15] as it represents the quality of the results compared to the reference set. A lower ADRS score means that the estimated Pareto-optimal configurations are closer to the reference set. Fig. 4 shows the average ADRS score for five runs of prior work, PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$ for the

three synthetic systems and ADAS. For $Synth_1$, PG-DSE$_{serial}$ and PG-DSE$_{parallel}$ achieved an ADRS score of zero, meaning that the algorithms achieved the same Pareto-optimal front as the reference set, as observed in Fig. 3. Prior work and PG-DSE$_{unpruned}$ achieved ADRS scores of 0.29% and 0.081%, respectively. For the larger design spaces, PG-DSE significantly outperformed prior work. PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$, on average, improved over prior work by 40.2x, 58.1x, and 66.2x, respectively. Compared to PG-DSE$_{unpruned}$'s score, PG-DSE$_{serial}$ and PG-DSE$_{parallel}$ were an average of 1.45x and 1.65x better, respectively. PG-DSE$_{serial}$ and PG-DSE$_{parallel}$ outperformed PG-DSE$_{unpruned}$ because the PSP algorithm successfully eliminated sub-optimal configurations, enabling PEGA to rapidly explore more accurate Pareto-optimal system-level configurations.

5.3 Runtime overhead

We evaluate the overhead of PG-DSE by quantifying the execution time to find the system-level Pareto-optimal front compared to prior work [4]. Fig. 5 shows the average execution time for five runs of prior work, PG-DSE$_{unpruned}$, PG-DSE$_{serial}$, and PG-DSE$_{parallel}$ for the different systems. For all four systems, prior work slightly outperformed the PG-DSE$_{unpruned}$'s and PG-DSE$_{serial}$'s execution time by an average of 3.97% and 0.15%, respectively. On the other hand, PG-DSE$_{parallel}$ *reduced* the execution time by an average of 82.6% compared to prior work. The longer execution time of PG-DSE$_{unpruned}$ and PG-DSE$_{serial}$ in the larger systems was a tradeoff for the significantly improved performance in accurately finding the Pareto-optimal front.

6 CONCLUSION AND FUTURE WORK

High-level synthesis DSE can be challenging in complex multi-component embedded systems with extremely large design spaces. This paper proposes a post-HLS pruned genetic design space exploration (PG-DSE) approach for timing-constrained complex embedded systems. PG-DSE integrates a Pareto-optimal pruning (PSP) algorithm and a Pareto-optimal genetic algorithm (PEGA) to accurately and rapidly find system-level Pareto-optimal configurations for complex embedded systems. To quantify the benefits of PG-DSE in comparison to exhaustive search and prior work, we used three synthetic systems and an autonomous driving application subsystem (ADAS) with very large design spaces, ranging in size from 2.61E+11 to 1.49E+104 possible solutions. Experimental results reveal that PG-DSE successfully found Pareto-optimal configurations that were close to the reference set. Compared to the the state-of-the-art in multi-objective HLS DSE, PG-DSE improved the quality of the results by an average of 58.1x, demonstrating PG-DSE's effectiveness.

A current limitation of PG-DSE is that it targets homogeneous PSM-level timing constraints. Future work involves incorporating variable timing constraints into PG-DSE to support systems in which different components or MCCs may be subject to different timing requirements. In addition, we plan to explore and extend PG-DSE to more complex and varied multi-accelerator systems in which the interactions between complex subsystems must be considered in DSE, while satisfying the system-level timing constraints.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under grant CNS-1563652. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Emmanuel Abidemi Adeniyi, Roseline Oluwaseun Ogundokun, and Joseph Bamidele Awotunde. 2021. IoMT-based wearable body sensors network healthcare monitoring system. In *IoT in healthcare and ambient assisted living*. Springer, 103–121.

[2] Renato Cordeiro, Dhruv Gajaria, Ankur Limaye, Tosiron Adegbija, Nima Karimian, and Fatemeh Tehranipoor. 2020. ECG-based authentication using timing-aware domain-specific architecture. *IEEE transactions on computer-aided design of integrated circuits and systems* 39, 11 (2020), 3373–3384.

[3] Lorenzo Ferretti, Giovanni Ansaloni, and Laura Pozzi. 2018. Lattice-traversing design space exploration for high level synthesis. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 210–217.

[4] Yiheng Gao and Benjamin Carrion Schafer. 2021. Effective High-Level Synthesis Design Space Exploration through a Novel Cost Function Formulation. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[5] Quentin Gautier, Alric Althoff, Christopher L Crutchfield, and Ryan Kastner. 2022. Sherlock: A Multi-Objective Design Space Exploration Framework. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 27, 4 (2022), 1–20.

[6] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, and Myoungho Sunwoo. 2015. Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture. *IEEE Transactions on Industrial Electronics* 62, 8 (2015), 5119–5132.

[7] Hermann Kopetz, Christian El-Salloum, Bernhard Huber, and Roman Obermaisser. 2007. Periodic finite-state machines. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*. IEEE, 10–20.

[8] Sakari Lahti, Panu Sjövall, Jarno Vanne, and Timo D Hämäläinen. 2018. Are we there yet? A study on the state of high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 5 (2018), 898–911.

[9] Yuchao Liao, Tosiron Adegbija, and Roman Lysecky. 2022. A high-level synthesis approach for precisely-timed, energy-efficient embedded systems. *Sustainable Computing: Informatics and Systems* 35 (2022), 100741.

[10] Hung-Yi Liu, Michele Petracca, and Luca P Carloni. 2012. Compositional system-level design exploration with planning of high-level synthesis. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 641–646.

[11] Aditya Paliwal, Felix Gimeno, Vinod Nair, Yujia Li, Miles Lubin, Pushmeet Kohli, and Oriol Vinyals. 2019. Reinforced genetic algorithm learning for optimizing computation graphs. *arXiv preprint arXiv:1905.02494* (2019).

[12] Christian Pilato, Daniele Loiacono, Antonino Tumeo, Fabrizio Ferrandi, Pier Luca Lanzi, and Donatella Sciuto. 2010. Speeding-up expensive evaluations in high-level synthesis using solution modeling and fitness inheritance. In *Computational intelligence in expensive optimization problems*. Springer, 701–723.

[13] Andy D Pimentel. 2016. Exploring exploration: A tutorial introduction to embedded systems design space exploration. *IEEE Design & Test* 34, 1 (2016), 77–90.

[14] A Raji, P Kanchana Devi, P Golda Jeyaseeli, and N Balaganesh. 2016. Respiratory monitoring system for asthma patients based on IoT. In *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*. IEEE, 1–6.

[15] Benjamin Carrion Schafer and Zi Wang. 2020. High-Level Synthesis Design Space Exploration: Past, Present, and Future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2628–2639. https://doi.org/10.1109/TCAD.2019.2943570

[16] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. 2015. Comparative review of selection techniques in genetic algorithm. In *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*. IEEE, 515–519.

[17] Sufian Sudeng and Naruemon Wattanapongsakorn. 2015. Post Pareto-optimal pruning algorithm for multiple objective optimization using specific extended angle dominance. *Engineering Applications of Artificial Intelligence* 38 (2015), 221–236.

[18] Zi Wang and Benjamin Carrion Schafer. 2022. Learning from the Past: Efficient High-level Synthesis Design Space Exploration for FPGAs. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 27, 4 (2022), 1–23.

[19] Sotirios Xydis, Christos Skouroumounis, Kiamal Pekmestzi, Dimitrios Soudris, and George Economakos. 2010. Efficient high level synthesis exploration methodology combining exhaustive and gradient-based pruned searching. In *2010 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 104–109.