Efficient High Level Synthesis Exploration Methodology Combining Exhaustive and Gradient-Based Pruned Searching

Sotirios Xydis, Christos Skouroumounis, Kiamal Pekmestzi, Dimitrios Soudris, George Economakos
Electrical and Computer Engineering Department, National Technical University of Athens, Greece.
Email: {sxydis, cskourou, pekmes, dsoudris, geconom}@microlab.ntua.gr.

Abstract—This paper presents a methodology for fast and efficient Design Space Exploration during High Level Synthesis. An augmented instance of the design space is studied taking under consideration the effects of both compiler- and architectural-level transformations onto the final datapath. A new gradient-based pruning technique has been developed, which evaluates large portions of the augmented solution space in a quick manner. At a second level, the proposed pruning technique is combined with exhaustive exploration in order to guarantee the quality of design solutions. We show that the proposed methodology delivers (i) higher quality designs than exploration methods which do not account the introduced extended design space, (ii) with considerable reductions of the exploration’s runtime and (iii) efficient convergence to global optima.

I. INTRODUCTION

Since High Level Synthesis (HLS) is now in its mature phase [1], a continuously growing community of IC designers have adopted HLS techniques to tackle design complexity and meet tight time-to-market requirements. HLS offers an automated and seamless path from high level behavioral specifications down to circuit level implementations. However, the rise of design abstraction exposed a large number of inter-dependent design parameters that have to be explored in order to discover the optimal solutions. Different trade-offs are associated with each solution and designers have faced the problem of reasoning on differing trade-offs among the set of design parameters. Thus, efficient exploration methodologies accompanied with automated tools are of great importance for a quick and concrete evaluation of the design space [2].

Design Space Exploration (DSE) is the procedure that evaluates the solution space in order to return a set of Pareto-optimal [3] design points according to some design criterions (execution delay, circuit area, dissipated power). Pareto optimality [3] specifies that one design solution dominates the other when it is at least as good in all of the criteria and also strictly better in at least one of them. Thus, designers are interested mostly on Pareto optimal configurations. The enumeration of all possible solutions in order to explore the Pareto frontier is an extremely time-consuming task, since the size of the design space is usually huge. Thus, DSE methodologies are focused on the development of strategies for efficient traversal of the available design space.

This paper targets the fundamental problem of exploring the Performance-Area trade-offs during HLS [4]. A Performance-Area exploration curve can be generated by iteratively schedul-

1This research is partially supported by the E.C funded program MOSART IST-215244, Website: http://www.mosart-project.org/.

ing the behavioral description with different area constraints. Although the aforementioned exploration strategy produces the exploration curve for the specific set of parameters, it is far from delivering the most efficient design solutions. That is because neither the impact of code transformations on the behavioral description nor the full set of architectural optimizations during scheduling, are considered as effective design parameters. Usually, code-level transformations (i.e. loop unrolling) and architectural level optimizations (i.e. operation chaining) are treated as user-guided pre-defined parameters during exploration. In a common case, such exploration strategies deliver suboptimal design points since the interaction between the structure of behavioral code and the architectural optimizations are silently assumed to be independent.

In this work, we show that using existing DSE methodologies a large portion of Pareto-optimal design points are excluded from the final exploration curve. In order to address this inefficiency, we propose an exploration methodology which accounts for both the loop unrolling code transformation and operation chaining architectural optimization as parameters of the design space. The proposed methodology explores the extended design space in a quick and efficient manner and delivers exploration curves which dominate the curves generated by the existing exploration strategies.

The main contributions of this paper are: 1. We propose the management of an extended design space during HLS exploration, through the incorporation of loop unrolling and operation chaining design decisions as parameters of a uniform solution space, in order to expose higher quality solutions. 2. We developed an efficient heuristic considering a gradient-based pruning technique which quickly explores the extended solution space. 3. We developed a meta-heuristic strategy which integrates in a balanced manner exhaustive together the proposed heuristic exploration. The proposed strategy is fully automated (supported through a CAD tool-flow) and it trades accuracy/quality of the derived curve for overall exploration’s runtime and vice versa.

Extensive experimentation has been conducted based on real-life computationally intensive benchmarks in order to evaluate the efficiency of our approach. In each case, a shift of the exploration’s curve towards more optimized solutions (Pareto-points) is reported in comparison to the existing design exploration methodologies. Additionally, the proposed exploration strategy delivers maximum average speedup of 37.9× in exploration’s runtime and maximum average accuracy of 92.9% on the quality of the final curve compared with the
exhaustive exploration of the design space.

The rest of the paper is organized as follows. Section II discusses the current literature, while Section III provides the basic observations that motivated this work. In Section IV the proposed exploration methodology is presented and analyzed. Section V evaluates the proposed methodology through a rich set of experimental data. Section VI concludes the paper.

II. RELATED WORK

Existing approaches in DSE during HLS concentrate on solving iteratively the resource allocation and operation scheduling problems [5], [6], [7]. Exploration is performed exhaustively traversing the entire defined solution space. Although Pareto-optimal design points are delivered, the exploration and evaluation of each design point imposes unaffordable execution runtime even for the case that heuristic scheduling/allocation is performed [6], [7]. Furthermore, neither the structure of the input code (i.e. loop unrolling) nor the architectural level optimizations (i.e. operation chaining) are considered as exploration parameters, resulting in a large set of unexplored solutions which usually dominates the derived exploration curve. The duality between timing- and resource-constrained scheduling problems has been taken into account during exploration in [8], [9]. The exploitation of the duality between the scheduling problems reports higher quality results than the previous mentioned approaches. However, the impact of loop unrolling is ignored [8], [9] and operation chaining is only partially supported [9] during exploration.

Loop unrolling during HLS has great influence on the datapath implementation [10], [11]. In [12], an HLS exploration methodology is presented considering code transformations. However, the exploration loop is restricted only to the code level, without encountering neither the impact of code transformations on scheduling efficiency under various resource allocations nor the impact of operation chaining.

Operation chaining in HLS [13] schedules data-dependent operations in a single control step by removing the intermediate storage logic (registers). It heavily depends on circuit’s operating frequency and critical path of datapath’s components. Conventional operation chaining targeted data-flow behavioral descriptions [13], [14] for performance improvement. However, the relationship among operation chaining depth (OCD), loop unrolling factor (LUF) and resource allocation has only partially evaluated in existing exploration flows [8].

The aforementioned research works target the problem of DSE either excluding from the automated process the effects of loop unrolling and operation chaining during scheduling [5], [6], [7], [8], [9] or studying code transformations at a pre-synthesis level without considering their effects on operation scheduling under various resource allocation scenarios [11], [12]. Fig. 1 illustrates the different concepts regarding the DSE problem during HLS. The main differentiator of this work is that we jointly encountered into the exploration loop, decisions concerning the operation scheduling under various resource allocation scenarios together with decisions concerning (i) the LUFs and (ii) the OCDs (Fig. 1c). By this way the design space is explored in a more global manner exhibiting new Pareto configurations [15]. Furthermore, the proposed methodology is performed at a meta-level. This makes it orthogonal with a large set of existing work on scheduling algorithms i.e. [5], [6], [7], [8], [9]. since every core scheduling algorithm can be used inside the exploration framework.

III. MOTIVATIONAL OBSERVATIONS

In this paper, we concentrate on the fundamental design problem of discovering the best trade-offs between the circuit’s performance and its area cost. Regarding these trade-offs, each design solution can be included in 2D design space, where the x-axis accounts for the area-cost while the y-axis accounts for the performance. Through a high level area model based on the number of allocated hardware resources [8], [9], each solution point in the design space can be represented as a vector of (Area, #Cycles) = ((#ALUs, #Muls), #Cycles).

The quality of the exploration curve is highly affected by the parameters of the design space that are explored. Considering a larger set of design parameters, the design space is explored more globally and solutions of higher quality are revealed. In the remainder of this section, we show through an illustrative example (Fig. 2) that the consideration of loop unrolling and operation chaining.
unrolling and operation chaining as design parameters during exploration, delivers solution points of higher quality. Even though loop unrolling [10] and operation chaining [13] are well known techniques for HLS tools, their incorporation in DSE methodologies is only partially supported and in many cases guided by the designer. However, pre-configuration of the LUFs and OCDs can lead to loosing Pareto solutions.

Fig. 2 illustrates exploration curves (derived through exhaustive DSE) for the case of 1D DCT kernel (part of 2D DCT kernel found in JPEG application [16]). Each exploration curve considers a different set for the LUF and the ODC. In case that neither loop unrolling nor operation chaining is considered, the exploration curve derived by evaluating only a small portion of the design space. Taking into account the impact of loop unrolling and operation chaining, a new unexplored design space is revealed which delivers new Pareto optimal design solutions (thus better performance-area trade-offs). However, arbitrarily increasing the LUF and/or the OCD parameters does not guarantee convergence to the "true" Pareto frontier (Fig. 2). This unexplored portion of the design space together with the lack of efficient automated strategies to traverse through it, form the basic observations that motivated this paper.

IV. DESIGN SPACE EXPLORATION METHODOLOGY

In this section we describe an exploration methodology during HLS which searches the augmented design space in a fast and efficient manner. We follow a top-down approach in our presentation. At first, we present the overall exploration methodology which is based on the incorporation of both exhaustive and heuristic techniques. Heuristic exploration is performed through a newly introduced gradient-based design space pruning (Subsection IV-A).

The overall exploration flow is depicted in Fig. 3. It is based on the iterative partitioning of the overall design space into smaller and disjoint subspaces (considering the area cost). Both exhaustive and gradient-based pruned heuristic exploration (Subsection IV-A) are used in a combined manner during the exploration procedure. The adopted principle is that, each design (sub)space explored through the gradient-based heuristic is decomposed to a set of smaller subspaces than the initial one. "Large" design (sub)spaces are explored further heuristically while "small" design (sub)spaces are further explored in an exhaustive manner. Each design (sub)space is characterized as "large" or "small" according to its cardinality and a designer specified threshold. Cardinality is the number of different configuration/solutions existed in each (sub)space.

The user provides (i) the behavioral description, (ii) the set of exploration parameters and (iii) their ranges. At the first iteration, no partition of the design space has been performed. Thus, without loss of generality, we handle the design space as a single partition. In case that the cardinality of the design space is smaller than the threshold specified by the designer, exhaustive evaluation is performed and the exploration completes. In the common case in which the cardinality is larger than the specified threshold, gradient-based heuristic exploration is performed to the whole design space. Gradient-based exploration returns a first set of pseudo-Pareto points. We call these points pseudo-Pareto since at the specific iteration they form true Pareto solutions of the explored (sub)space which can be replaced in a next iteration if a design solution that dominates them is found.

The initial design space is now partitioned in several subspaces. Design subspaces are defined dynamically during exploration, considering the neighboring pseudo-Pareto points as boundaries. The cardinality of each design subspace is further evaluated in order to be explored exhaustively or heuristically with gradient-based pruning. Every time a gradient-based pruned exploration is performed, new design subspaces are generated. The exploration procedure for each design space terminates when one of the following conditions are evaluated to true: (i) the subspace has been explored exhaustively or (ii) two subsequent gradient-based explorations returns the same subspace boundaries.

A. Description of Gradient-Based Pruning Technique

The number of resource allocation scenarios that have to be evaluated during HLS exploration is a critical factor of the overall exploration’s runtime. The more the examined allocation scenarios, the larger the runtime of the exploration procedure. Gradient-based pruning technique defines a fast heuristic way to decide whether a resource allocation scenario has to be examined or to be excluded during exploration. Gradient-based pruning is based on the following lemma:

Lemma 1: Given the operation chaining degree and the loop unrolling factors, any resource allocation scenario $C_i = \{\#ALU_i, \#Mul_i\}$ with larger area cost than the cost found in the allocation scenario that matches exactly the maximum operation level parallelism of the application, $C_{MOLP} = \{\#ALU_{MOLP}, \#Mul_{MOLP}\}$, delivers either the same or worst latency results.

Proof: Given the OCD and LUFs the minimum latency of a behavioral description is delivered through As-Soon-As-Possible (ASAP) scheduling [4] which exploits the whole operation-level parallelism. In case that a resource allocation configuration, $C_i$, with $\{\#ALU_i > \#ALU_{MOLP}\}$ and $\{\#Mul_i > \#Mul_{MOLP}\}$ is considered, then the remaining resources over the allocation of $C_{MOLP}$ are idle since there is no available operation parallelism to be exploited.
Lemma 1 provides a key insight for the exploration of resource allocation scenarios. It says that for a given OCD and LUF, the performance-area exploration curve presents zero-gradient for resource allocation scenarios which allocate a greater number of resources than the resources that match the maximum operation level parallelism. In other words, when a zero-gradient at the maximum operation level parallelism is occurred for a given set of resources, the increment of resources has no beneficial impact on the latency of the datapath, thus no Pareto point can be found and there is no need to evaluate these allocation scenarios.

Gradient-based pruning takes into consideration the above observations and proposes the following: Each time during the design space exploration, a zero-gradient, \( \frac{\Delta \text{Latency}}{\Delta \text{Area}} = 0 \), is occurred in the performance-area exploration curve, the solution with the lowest area cost is: (i) a possible Pareto-solution and (ii) a “good” design point to alter the examined configuration mode (i.e. by increasing either the number of allocated Muls or the OCD or the LUF). By this way, large portions of resource allocation scenarios are excluded from the exploration procedure improving the overall exploration runtime without degrading the finding of Pareto solutions.

However, zero-gradient segments can occurred also at resource allocations with lower operation level parallelism than the maximum, when the specific allocations exhibit no performance improvement. These zero-gradient segments are local and have to be de-characterized as “good” design points for altering configuration mode. Thus, examining zero-gradient only between two successive design solutions may lead to the loss of Pareto solutions. In order to avoid such situations, the zero-gradient segments are evaluated for a \( Depth > 2 \) of successive design solutions. Thus, for \( Depth > k \) the zero-gradient segment condition is formed as: \( \frac{\Delta \text{Latency}}{\Delta \text{Area}} = 0 \) for \( i = k+1 \) to \( i = Depth \).

Fig. 4 illustrates the application of gradient-based pruning during the exploration of various resource allocation scenarios for the 1D DCT kernel of the JPEG application. We consider a configuration setting of \( OCD = 3 \) with no loop unrolling, \((\text{Min.} \#\text{ALUs}, \text{Min.} \#\text{Muls}) = (4, 2), (\text{Max.} \#\text{ALUs}, \text{Max.} \#\text{Muls}) = (32, 9)\) and \( Depth = 5 \).

The explorative procedure evaluates resource allocation scenarios by increasing the ALUs until a zero-gradient segment of \( Depth = 5 \) is occurred. Whenever such segments are found, the configuration mode alters by incrementing the Muls until the \( \text{Max.} \#\text{Muls} \). Gradient-based pruning derives that resource allocation scenarios for \( \#\text{ALUs} > 17 \) have to be excluded from the exploration. Thus, in a set of \((\text{Max.} \#\text{ALUs} - \text{Min.} \#\text{ALUs}) \times (\text{Max.} \#\text{Muls} - \text{Min.} \#\text{Muls}) = 196\) available resource allocation scenarios only 60 design points have been evaluated delivering a speedup of \( 3 \times \) in exploration’s runtime (under the coarse assumption that the evaluation time of each design point is the same).

Fig. 4. Exploration of 1D JPEG DCT resource allocation scenarios based on gradient-based pruning.

Fig. 5. Pseudocode for the gradient-based pruning technique. The various kernel parameter exploration loops depicted in lines 6-14. Thus, the outer loops encounter for the loop unrolling factors per loop indexes (lines 6-8). After the loop unrolling factors are known the CDF of the behavioral description is formed (line 10). CDFG depends only from the LUFs’ values. The next exploration loop concerns the operation chaining degrees, OCDs (line 12), while the two inner loops (lines 13-14) concentrate on the exploration of resource allocation scenarios by incrementing the \#Muls and \#ALUs, respectively.

The core of gradient-based pruning is placed at the innermost loop since it targets the resource allocation level. At first, the CDFG is scheduled given the resource constraints and the OCD value (line 16) and a solution point \( x_i \) is constructed to handle the examined configuration vector (line 17). After scheduling the CDFG, the information about the number of control steps (line 18) and the area cost (line 19) of the examined solution, is extracted. We adopted the linear area estimation model used also in \( [9] \): \( \text{Area Cost} = (\#\text{ALUs} \times \text{Area}_{\text{ALU}}) + (\#\text{Muls} \times \text{Area}_{\text{MUL}}) \). At next, the gradient of the exploration curve, \( \text{Grad} \), is evaluated according to a user specified depth (line 20). In case that there is a negative gradient the examined solution forms a possible
Pareto point and it is inserted in the Examined Curve (line 22). In case that $Grad$ is either positive or zero, the current ALU based exploration loop is broken and the last found possible Pareto point concerning the number of ALUs is retrieved $(x[i]−depth+1).ALU_{no}$ to form the initial point for the exploration loop of the next configuration (line 23). The points inserted in the Examined Curve are characterized as possible Pareto points, since there is a large possibility the real Pareto point to be generated in an exploration of a later configuration. Thus, at the end of the exploration procedure the points stored in the Examined Curve are re-evaluated and the final Pareto points are extracted (line 30).

V. EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of our approach, we have developed a fully automated HLS exploration framework implementing the proposed methodology by properly extending SPARK-HLS tool [17]. We used a representative set of computationally intensive DSP benchmark applications to evaluate the efficiency of the proposed DSE approach. The benchmark suite consists of 7 real-life DSP kernels in C including: (i) a 16th-order FIR filter (FIR16), (ii) a 1D Discrete Cosine Transformation (1D DCT), (iii) a YUB to RGBA filter (YUB2RGBA) [18], (iv) a Discrete Haar Wavelet Transform (DWT) [18], (v) a MESA Matrix Multiplication kernel (MatMul) [16], (vi) the 2D DCT kernel found into the JPEG application (Jpeg DCT) [16], (vii) the 2D IDCT kernel from the MPEG2 (Mpeg IDCT).

The HLS exploration tool was installed and run on a Linux Xeon server at 2.33 GHz with 4 GB RAM. For each kernel, a range of $(\text{Min} \# \text{ALUs}, \text{Min} \# \text{Muls}) = (4, 2), (\text{Max} \# \text{ALUs}, \text{Max} \# \text{Muls}) = (32, 16)$ resource allocation scenarios were explored. The area cost was of each component extracted through post-synthesis characterization [19] for a 0.13 um standard cell library [20]: $\text{Area}_{\text{ALU}} = 4000\text{um}^2, \text{Area}_{\text{MUL}} = 12000\text{um}^2$. The operation chaining degree ranged between $(\text{Min} \text{OCD} = 0), (\text{Max} \text{OCD} = 3)$. The minimum and maximum per loop index unrolling factors (LUFs) depend on the behavioral description of each kernel. In each case, the whole range of each LUF has been explored. Empirically, we considered $\text{Max} \text{Cardinality} = 100$ for the designer specified cardinality threshold which guides the invocation of exhaustive or gradient-based pruned exploration.

Due to space limitations, Fig. 6 depicts some of the exploration curves derived from the proposed exploration methodology. The depth parameter for the gradient-based pruning was set to $Depth = 10$, thus 10 solutions are examined to decide whether a zero-gradient segment is global or local. In the exploration diagrams of Fig. 6, the exploration curve of the non-augmented design space (exploration unawareness of loop-unrolling and operation chaining parameters considering only the various resource allocation scenarios) has been overlapped. The Pareto curve shifting towards higher quality performance-area trade-offs is depicted, confirming our motivational observations. Fig. 6 depicts also the non-discovered Pareto points from the proposed methodology (solution points which did not lay onto the exploration curve). The "lost" Pareto points were generated after performing exhaustive exploration on the whole design space, evaluating each possible solution. The proposed methodology discovered the 100% of Pareto solutions in three out of four of the depicted benchmarks with an average speedup of 10x in exploration’s runtime. The loss
of some Pareto points in MatMul kernel is a side-effect of the heuristic nature of gradient-based pruning.

In Table I, we study the speedup gains of the proposed versus an exhaustive methodology and the impact of the gradient’s $Depth$ on the exploration’s runtime. We considered three values of the gradient $Depth$ parameter, namely $Depth_1 = 2$, $Depth_2 = 5$, $Depth_3 = 10$. For each $Depth_i$ information about (i) the exploration’s runtime, (ii) the speedup over the exhaustive approach and (iii) the accuracy of the exploration have been recorded for each benchmark. Accuracy of exploration is defined as the ratio of the found Pareto points by the proposed methodology over the Pareto points extracted by the exhaustive exploration. Respectively, exploration’s speedup is defined as the ratio of the solutions explored exhaustively over the solutions explored with the proposed methodology. It is expected that smaller values of gradient $Depth$ deliver higher speedup in the expense of lower accuracy. However, this is true only under the false assumption that the maximum number of exhaustive explorations performed is the same for all the DSP kernels. Actually, there is a trade-off concerning exploration’s runtime between the $Depth$ and the $Max\_Cardinality$ parameters. This type of trade-off is exposed in the 1D DCT case, in which the exploration with $Depth = 5$ (11 exhaustive explorations) reports lower speedup than the exploration with $Depth = 10$ (6 exhaustive explorations).

In average, the proposed methodology delivers speedup gains of 37.9 ×, 13.3 ×, 7.9 × for $Depth_1 = 2$, for $Depth_2 = 5$, for $Depth_3 = 10$, respectively. The average accuracy is 64.6%, 82.9%, 92.9% for the aforementioned gradient $Depth$ values. Thus, the proposed methodology in the worst accuracy case ($Depth = 2$) delivers an exploration’s runtime reduction of 37.9 ×, while at the worst speedup case ($Depth = 10$) it delivers an accuracy ratio of 92.9%.

**VI. CONCLUSION**

This paper addressed the problem of design space exploration during HLS. We identified the need to explore the design space in a more global manner than the existing techniques and we introduced a new methodology combining exhaustive along with heuristic exploration for quick and efficient traversing of the design space. Experimental results have shown significant quality improvements of the explored solutions and reductions in exploration runtime comparing with the exhaustive approach.

**TABLE I**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Explor.</td>
<td>Explor. Speedup</td>
<td>Accuracy (%)</td>
<td>#Explor.</td>
</tr>
<tr>
<td>FIR16</td>
<td>2784</td>
<td>27.75</td>
<td>12.8 × 100%</td>
<td>3092</td>
</tr>
<tr>
<td>1D DCT</td>
<td>13920</td>
<td>1347</td>
<td>10.3 × 80%</td>
<td>2562</td>
</tr>
<tr>
<td>YUB2RGBA</td>
<td>13920</td>
<td>736</td>
<td>18.9 × 33.3%</td>
<td>2672</td>
</tr>
<tr>
<td>MatMul</td>
<td>7840</td>
<td>525</td>
<td>14.9 × 42%</td>
<td>969</td>
</tr>
<tr>
<td>DWT</td>
<td>334080</td>
<td>2061</td>
<td>162 × 50%</td>
<td>7481</td>
</tr>
<tr>
<td>Jpeg 2D DCT</td>
<td>111360</td>
<td>9425</td>
<td>12 × 66.6%</td>
<td>11227</td>
</tr>
<tr>
<td>Mpeg 2D IDCT</td>
<td>111360</td>
<td>3259</td>
<td>34.2 × 80%</td>
<td>7853</td>
</tr>
<tr>
<td>Average</td>
<td>-</td>
<td>-</td>
<td>37.9 × 64.6%</td>
<td>-</td>
</tr>
</tbody>
</table>

**REFERENCES**