Partial Encryption of Behavioral IPs to Selectively Control the Design Space in High-Level Synthesis

Zi Wang[1] and Benjamin Carrion Schafer[2]
Department of Electrical and Computer Engineering, The University of Texas at Dallas
zi.wang5@utdallas.edu[1], schaferb@utdallas.edu[2]

***Abstract*—Commercial High-Level Synthesis (HLS) tool vendors have started to enable ways to protect Behavioral IP (BIPs) from being unlawful used. The main approach is to provide tools to encrypt these BIPs which can be decrypted by the HLS tool only. The main problem with this approach is that encrypting the IP does not allow BIP users to insert synthesis directives into the source code in the form of pragmas (comments), and hence cancels out one of the most important advantages of C-based VLSI design: The ability to automatically generate micro-architectures with unique design metrics, *e.g.* area, power and performance. This work studies the impact to the search space when synthesis directives are not able to be inserted in to the encrypted IP source code while other options are still available to the BIP users (*e.g.* setting global synthesis options and limiting the number and type of functional units) and proposes a method that selectively controls the search space by encrypting different portions of the BIP. To achieve this goal we propose a fast heuristic based on divide and conquer method. Experimental results show that our proposed method works well compared to an exhaustive search that leads to the optimal solution.**

I. Introduction

To address the increase in VLSI design complexity, companies have started to build their heterogeneous MPSoCs using third party IPs (3PIPs). In addition, they have started to rely on High-Level Synthesis (HLS). HLS takes input as a behavioral untimed description (*e.g* ANSI-C or C++) and converts it to efficient RTL (Verilog or VHDL). One main advantage of HLS compared to RTL is that it enables the automatic generation of micro-architectures with unique design metrics without the need to modify the behavioral description. In most commercial HLS tools this is typically done by setting three complementary, but also partially overlapping, exploration *knobs*. The first, and also most powerful *knob* is through synthesis directives in the form of pragmas (comments) inserted directly into the source code. The second exploration *knob* is global synthesis options. This *knob* is complementary to the first one as it allows to *e.g.* control the target synthesis frequency and FSM encoding scheme, but also overlapping as it allows to specify how to synthesize the arrays, loops and functions too, but at a much coarser granularity (*e.g.* all loops should be unrolled or all functions inlined). This *knob* forces all the explorable operations to be synthesized in the same way as it applies to the entire behavioral description. The last *knob* enables different levels of resource sharing by allowing the designers to control the maximum number and types of functional units (FUs) allowed.

This new architecture paradigm combined with the new design methodology has opened a new market in the area of third party behavioral IPs (3PBIPs), with new challenges and opportunities. One of the main challenges for BIP providers that is hampering the expansion of BIPs, which is how to protect the IP from being reused illegally. Also, BIPs normally have different price policies depending on the amount of disclosure of the IP. Full disclosure gives the BIP user full control and full visibility of the code and hence is priced differently than provided as a synthesized netlist. An alternative service model involves the encryption of the BIP with a set of predefined constraints (*e.g.* IO bitwidths, synthesis pragmas), which the BIP consumer cannot modify and hence is less expensive.

This last service model allows BIP users to benefit from some of the advantages of HLS, including the generation of fast cycle-accurate models, but severely limits the ability to perform HLS design space exploration as it does not allow the insertion of synthesis directives. Encrypting the BIP nevertheless still allows to generate different micro-architectures by using the other two exploration *knobs* (global synthesis options and functional units constraints). Unfortunately, the synthesis directive knob is the most powerful one among the three knobs. This work addresses this issue, by first investigating how the design space is affected due to the encryption of the BIP and then proposing a method to partially disclose portions of the BIP to selectively control the search space visible by the BIP user. In summary, this work makes the following contributions:
(1) Investigates how fully encrypting BIPs for HLS affects the search space in HLS DSE.
(2) Present an efficient method to selectively disclose different explorable operations to control the visibility of the BIP vs. search space.

II. Motivational Example

The main motivation for the proposed work is the observation that fully encrypted BIPs do not allow the insertions of synthesis directives and thus, limit the quality and quantity of Pareto-optimal micro-architectures that can be obtained from that particular BIP. Fig. 1 shows an overview that summarizes the current BIP protection flow and its limitations. The input is a behavioral description that can either be encrypted or not. To encrypt it, commercial HLS tools provide an encryption

978-3-9819263-2-3/DATE19/©2019 EDAA

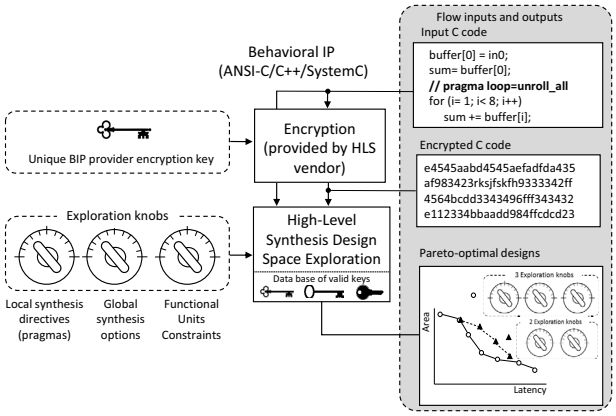

Fig. 1: Encryption flow for BIPs and HLS DSE result obtained when setting 2 vs. 3 different exploration knobs.

tool for the BIP vendors and a unique encryption key which identifies that vendor. The encryption key is in turn used by the IP provider to encrypt the BIP. The HLS tools can then internally decrypt this behavioral description as it contains a database with all the legitimately issued keys. Although similar across HLS vendors, this process is still tool and vendor specific and thus, the BIP vendor would have to encrypt the BIP individually for each HLS tool targeted.

As shown in the figure, the input C code contains a synthesis directive to fully unroll the loop. A particular micro-architecture is generated based on this pragma setting. Changing this pragma to *not unroll* leads to a completely different micro-architecture. Thus, setting different combinations of these pragmas leads to a trade-off curve of unique micro-architectures shown graphically in the Fig. 1. The main problem when the C code is encrypted is that it limits the search space severely as it does not allow the insertion of these synthesis directives. Out of the three tuning *knobs*, this has been shown to be the most important as it fixed the main micro-architecture [1]. This in turn implies that the exploration result obtained by using the remaining two tuning *knobs* is sub-optimal.

Fig.2 shows an example of the exploration result of a multi-stage graphical filter synthesized using a commercial HLS tool (CyberWorkBench [2]). As shown, the trade-off curve obtained when the BIP is encrypted misses most of the dominant designs found by the trade-off curve obtained when the BIP is not encrypted and thus, all three *knobs* are used. It can be observed that in particular the encrypted version finds designs in the extremes of the trade-off curve, but misses most of the designs in the middle. This is mainly due to the fact that through the global synthesis options knob, the explorer can still control how to synthesize arrays, loops and functions, but the knob settings affects all of the constructs at the same time. *E.g.* if the behavioral description has N arrays, through the global options we can set them all to be synthesized as RAM or registers, but we can not do a fine-grained selection setting some arrays to be synthesized as RAMs and others as registers.

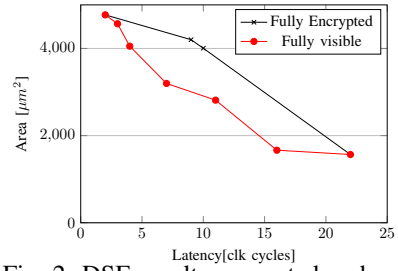

Fig. 2: DSE results encrypted and not.

III. Proposed Framework

The proposed method is based on a Divide and Conquer strategy composed of two phases. The method starts by performing a full HLS DSE on the un-encrypted BIP using the three exploration knobs. The results is the reference trade-off curve (PO_{ref}) which contains all the Pareto-optimal designs for this particular BIP. When the number of synthesis directives is small, an exhaustive exploration is performed, while for larger number of synthesis directives we use a genetic algorithm, which has shown to produce good results for this type of multi-objective optimization problems [3]. Due to space limitations we will not go into details about how the GA-based explorer works, as this is also not a contribution of this work.

The proposed method then continues in the first phase by pre-characterizing the impact of each *explorable* operation (OP) in the BIP on the search space, while the second phase continues by merging the results to obtain the best solutions. One important aspect to consider is that the solution to the proposed problem is not a single optimal solution but a set of different Pareto-optimal solutions with different quality vs. visibility results of the exploration. Intuitively, the more of the source code we make visible, the more exploration operations we can control using local pragmas and thus, expand the search space (which is good), but at the same time we are exposing more of the BIP (which is bad). Thus, this problem needs to be formulated as a multi-objective optimization problem where the cost function is defined as $C = \alpha \times DSE_{quality} + \beta \times BIP_{Visibility}$. Setting different values for α and β makes the proposed method either optimize for quality of the DSE result or reduction of the visibility, with α=1 and β=0, leading to a fully un-encrypted BIP where each explorable operation can be controlled through a synthesis directive, while α=0 and β=1 leads to a fully encrypted BIP. Thus, the next question that we need to address is how to measure the DSE quality result and the visibility.

DSE Quality Metric: To measure the quality of the exploration result, when larger or smaller portions of the BIP are made visible, we use Average Distance to Reference Set (ADRS), which is a widely used metric to compare multi-objective optimization problems like this one. ADRS indicates the normalized average distance between the reference Pareto-front (PO_{ref}) and the approximate Pareto-set, *i.e.*, tells how close a Pareto front (Ω) is to the reference front, where the reference front (Γ) is obtained by performing a HLS DSE for

the fully un-encrypted BIP using the three exploration knobs. ADRS is computed as shown in (1). Function f calculates the distance between points γ from Patreto front(Γ) and ω from reference front(Ω).

$$ADRS(\Omega,\Gamma) = \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\min_{\gamma\in\Gamma}\frac{f(\omega,\gamma)}{f(\omega,O)} \quad (1)$$

BIP Visibility: The easiest way to account how much of the BIP the IP provider needs to disclose to the user would be by enumerating the explorable operations that are left un-encrypted such that the BIP user can the insert the synthesis directives at these operations. Because different tools vendors use different ways to encrypt BIPs, another way to account for this would be to count the lines of code encrypted. In this work we use the first method as the commercial HLS tool used in this work allow to selectively encrypt different portions of the description.

Based on this, the cost function can be re-written as $C = \alpha \times ADRS + \beta \times Visibility$. The next subsections describe our proposed two-phase method in detail.

A. Phase 1: Individual Explorable Operation Characterization:

This first phase pre-characterizes the effect on the design space exploration of every single explorable operation. In this work we consider arrays, loops and functions. Thus, for every operation, our proposed method, exhaustively synthesizes (HLS) the BIP with all the combinations of all the synthesis directives that can be assigned to that particular operation. That means, besides the function unit constraint optimization, we set all possible pragmas to the specific explorable operation while others detected by different global options. The result is a trade-off curve of Pareto-optimal designs for that particular operation, $PO_{op1} = \{D_1, D_2, \ldots, D_n\}$, with $D_i = \{attr1\}$. The results of all the different explorations are added a Pareto-optimal trade-off list ($POList$).

B. Phase 2: Merging Stage:

This second phase can be further subdivided in two steps. The first step estimates the effect of combining different explorable operations on the DSE result, while the second step re-explores the best set of combinations to get accurate results. *Step 1: Combined DSE Result Prediction:* This step makes use of the exploration results obtained in phase 1 and *predicts* the quality of DSE result when more than one explorable operation is explored in a compositional way. Compositional design techniques have been show to produce good results in these type of applications [4]. This enables to quickly identify systems with unique trade-offs without having to perform any synthesis. For this, we assume that the exploration results are additive and hence combine the trade-off curves obtained by the individual characterization and remove all non-optimal designs. Although this is not fully accurate, it serves as an upper boundary. The loss in accuracy happens because two attributes, when set together, can lead to a much better result, which an offline compositional technique cannot capture. *E.g.* unrolling a loop with the same unroll factor as the number of read/write ports of an array synthesized as registers. This analysis is done for all the attributes' combinations, which although exponential in nature is very fast because no synthesis is required.

The method then analysis the theoretical trade-off curves obtained when different explorable operations are explored together and prunes away those combinations that do not lead to at least one Pareto-optimal design.

Step 2: Re-Exploration: This second step takes as inputs the configurations that the previous step identified as leading to at least one Pareto-optimal design and fully explores them using either the exhaustive search or GA-based method used to obtain the reference set (line 13). It then compares the resultant trade-off curve with the reference curve (PO_{ref}) by computing the ADRS and also annotating the amount of disclosure (visibility) (line 14).

The final steps prunes away all configurations that do not lead to any Pareto-optimal configuration and returns the configurations composed of explorable attributes and visibility that lead to unique ADRS vs. visibility trade-offs (lines 16 and 17).

IV. Experimental Results

Six computationally intensive applications taken from the open source Synthesizable SystemC Benchmark suite (S2CBench) [5] are used to measure the effectiveness of our proposed method. The HLS tool used is CyberWorkBench v.6.1 [2] from NEC, the target technology is Nangate open source 45nm and the target synthesis frequency set in all cases to 100Mhz (clock period of 10ns).

Two sets of experiments are performed. The first measures the importance of the synthesis directives in the form of pragmas compared to the two other exploration knobs (global synthesis options and FUs count) by performing a HLS DSE on the 6 benchmarks using the three knobs combined followed by a DSE of using only two knobs (global options and FU count) when the benchmarks are fully encrypted.

Table I summarizes the HLS DSE results qualitatively and quantitatively using the ADRS and runtime using an exhaustive search explorer for all six benchmarks when the BIP is fully encrypted vs. the the GA-based explorer for the not encrypted case. For this latter case, due to the randomness in the GA process, the exploration is executed 5 times for each benchmark and the best results plotted. The main reasons for using an exhaustive search for the encrypted case and GA for the non-encrypted case is that encrypting the BIP considerably reduces the search space and makes an exhaustive search using the global options and FU count feasible.

From the results it can be concluded that the ADRS worsens by an average of 18.13% when the BIP is encrypted due to not being able to use the synthesis directives. In all 6 benchmarks the un-encrypted version leads to an ADRS of 0%, which means that it finds all of the dominating results. One positive side-effect of encrypting the BIP is that the exploration running time is substantially reduced leading to a comparatively small number of unique combinations. From Table I it can be shown

TABLE I: DSE results encrypted vs. not encrypted BIP

Bench	ADRS[%] Encrypted	ADRS[%]Not Encrypted	Run[s] Encrypted	Run[s]Not Encrypted
ave8	9.21	0.00	50	392
adpcm	2.27	0.00	203	880
fir	8.76	0.00	93	4,354
interp	16.52	0.00	116	8,057
fft	22.05	0.00	102	11,394
quicksort	49.98	0.00	197	7,135
Avg.	**18.13**	**0.00**		
Geom.			**114**	**3,154**

that the DSE of encrypted BIP, took was on average 28 × faster.

Fig.3 shows the results of the second sets of experimental results that show how the quality of the DSE is affected by increasing the amount of visibility into the BIP. In this case the visibility is given by the number of explorable operations that need to be left un-encrypted. Our proposed method is compared against an exhaustive search that performs a full exploration for each combination of visible explorable operations. In both cases the GA explorer is used when performing the HLS DSE and it is again executed 5 times. The best results for each method are reported. The results show, as expected, that disclosing, larger portions of the BIP has an important effect on the DSE results. It can also be observed that our proposed method leads to similar results compared to the exhaustive search with an average ADRS degradation of 0.63%. Fig.3 also shows that in most cases our proposed method finds the optimal solution, leading to the same trade-off curve as the exhaustive search method. In terms of runtime, our proposed method is on average 2 × faster than the exhaustive search.

These results highlight the importance of synthesis directives in the form of pragmas on HLS design space exploration and thus, implies that HLS vendors should specifically target the ability to control these without revealing the BIP description.

Previous Work

Most previous work in the field of hardware IP obfuscation deal with hard IPs, while this work is mainly focuses on the efficient encryption of behavioral IPs to maximize HLS DSE, which comes under the category of soft IPs.

Previous work in this area proposed by [6] and [7] represents RTL code as a data flow [6] or state transition matrix [7] graph. The graph is then modified with additional states(which are also called as key states) in the finite state machine representation of the code which should be passed through with the aid of a key sequence [6] or a code word [7]. The IP will start functioning only after the application of correct keys;otherwise the IP will be stuck in a futile, obfuscated state.

V. Summary and Conclusions

This works illustrates the importance of synthesis directives in the form of pragmas (comments) for HLS design space exploration (DSE). The proposed work can be used to selectively control the search space and thus, lead to a new business model for BIP providers, selling BIPs which lead to larger search spaces at a higher cost than BIPs restricted to a smaller search space.

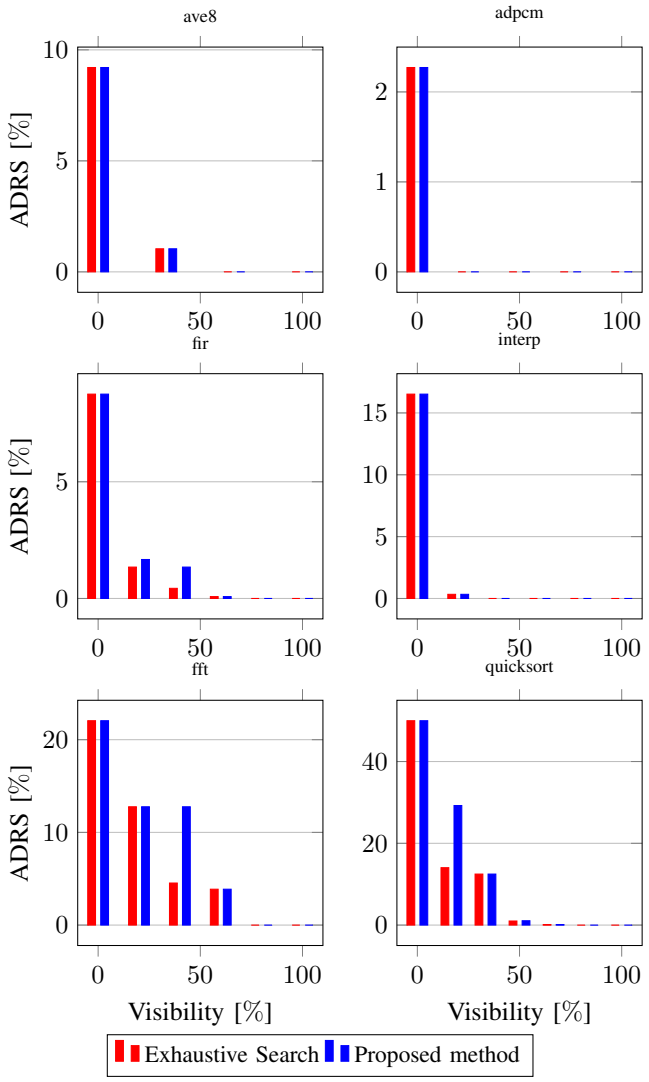

Fig. 3: Proposed method results showing ADRS vs. visibility results compared to an exhaustive search.

References

[1] B. C. Schafer, "Probabilistic multiknob high-level synthesis design space exploration acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[2] NEC CyberWorkBench. (2017). [Online]. Available: www.cyberworkbench.com

[3] A. E. Eiben, P.-E. Raue, and Z. Ruttkay, "Genetic Algorithms with Multi-parent Recombination," in *International Conference on Parallel Problem Solving from Nature*, 1994.

[4] L. Piccolboni, P. Mantovani, G. D. Guglielmo, and L. P. Carloni, "Cosmos: Coordination of high-level synthesis and memory optimization for hardware accelerators," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, pp. 150:1–150:22, Sep. 2017.

[5] B. C. Schafer and A. Mahapatra, "S2cbench: Synthesizable systemc benchmark suite for high-level synthesis," *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 53–56, Sept 2014.

[6] R. S. Chakraborty and S. Bhunia, "Rtl hardware ip protection using key-based control and data flow obfuscation," in *2010 23rd International Conference on VLSI Design*, Jan 2010, pp. 405–410.

[7] A. Desai, M. Hsiao, C. Wang, and c. Nazhandali, "Interlocking obfuscation for anti-tamper hardware," in *In: Proceedings of the eighth annual cyber security and information intelligence research workshop. ACM*, 2013, pp. 1–4.