Towards Purposeful Design Space Exploration of Heterogeneous CGRAs: Clock Frequency Estimation

Dennis Leander Wolf[1], Christoph Spang[2] and Christian Hochberger[1]
[1]Department for Electrical Engineering and Information Technology Computer Systems Group, TU Darmstadt
[2]Department of Computer Science, Embedded Systems and Applications Group, TU Darmstadt
Email: {wolf, hochberger}@rs.tu-darmstadt.de, spang@esa.tu-darmstadt.de

***Abstract*—Coarse Grained Reconfigurable Arrays become increasingly popular. Besides research on scheduling algorithms and microarchitecture concepts, the use of heterogeneous structures can be a key approach to exploit their full potential. Unfortunately, a purposeful design space exploration of CGRAs is not trivial, since one needs to know the clock frequency of the resulting hardware implementation. This paper discusses challenges and a statistical approach to maximum clock frequency estimation of heterogeneous CGRAs with an irregular interconnect on FPGAs. The presented approach allows estimation with a maximum error of 8.8 - 17.4 % and a mean error of only 1.9 - 4.6 %.**

I. Introduction

Varying from simple dataflow pipelines to complex parallel computing processors, all concepts of Coarse Grained Reconfigurable Architectures/Arrays (CGRAs) are based on Processing Elements (PEs) that can process data almost independently. Although the discussion of performance most often revolves around the best possible mapping of an application, the composition of a CGRA plays another major role. Composition means the layout of a CGRA in terms of number of PEs, their interconnect and the individual provision of operators in each PE. The optimal composition is indeed dependent on an application or an application domain, but it is obvious that a heterogeneous composition promises a more efficient increase in performance rather than trivial PE scaling, since more resources can be fitted precisely - resulting in higher utilization.

SOCs incl. FGPAs became a popular choice for testing hardware accelerators, and overlay architectures such as CGRAs can reduce configuration time drastically. Therefore, we extended the CGRA tool presented in [1] to generate arbitrary CGRAs, targeting a Xilinx Zynq XC7Z045. This should be the basis for a purposeful design space exploration (DSE). The framework supports arbitrary numbers of PEs, provides 128 different operators and any kind of direct interconnect, as long as all PEs are reachable. Therefore, the DSE should be fully automated towards a certain property, e.g. run-time of a given set of applications. Initial tests show a clock frequency range between 60 MHz and 150 Mhz, which indicates that the clock frequency must be considered in a purposeful DSE. This can be realized using optimization algorithms like Simulated Annealing (SA)[1]. Such heuristics require a cost-function and if run-time should be optimized, the clock frequency needs to be known for any possible composition. SA makes millions of changes per run, for which synthesis is infeasible and we need to estimate substantially fast and as accurate as possible.

[1]Previous experiments, using less accurate estimators, have shown that SA is the most promising approach in terms of result quality.

Firstly, we will discuss the generation of a database with 12 k reference CGRAs, in order to systematically cover the design space and gain an understanding of the features of these designs (see Section IV). Secondly, an analysis of the database gives a basic understanding of the complexity of the estimation and is explained in Section V. Thirdly, a statistical estimation approach is introduced in Section VI. Based on the most influential parameters concerning the synthesis, ten most similar CGRAs are picked from a reference set. Then a weighted Kernel Density Estimator (KDE) is used to compute the final estimation result. The whole process, from generation of reference compositions to the export of a final estimator, is fully automated and eases portability to any other type of FPGA. To our knowledge all other contributions neither provide similar accuracy even for homogeneous structures nor are capable of estimating heterogeneous and irregular structures nor use a similar estimation technique.

II. Related Work

Only few publications try to provide clock frequency estimations in the context of CGRA overlays on FPGA. None of them consider the impact of a composition on the clock frequency.
CGRA-ME [2] uses a static timing analysis allowing adjustments of the frequency determined by paths that are actually used by a given application. This means not the resulting maximum clock frequency of a design but an application specific path delay of an already existing and therefore well-known implementation is estimated. An average error of 9.6% is achieved without mentioning the maximum error. Yan et. al. [3] present a similar approach for homogeneous CGRAs yielding an average error of 8% and a maximum error of up to 30%.
Suh et. al. [4] discuss a tool that is capable of a DSE on PE and interconnect level on a heterogeneous CGRA. They explicitly avoid clock frequency estimation and use a priori architectural knowledge instead, confirming our perception of this being a complex task. A neural network to estimate the max. clock frequency for FIR Filters on an FPGA is evaluated in [5]. As in our case the estimation is only based on the parameters of the design, but the complexity of a homogeneous FIR filter is much smaller compared to a heterogeneous CGRA. The mean relative error is 22.22 %, the maximum error is not mentioned. Chen et. al. [6] focus on power estimation but also estimate path delay. For 25 data points they reach a mean error of about

21 % and max error of 50 % for a single benchmark. The runtime usually lies below one minute. Earlier approaches as [7] for High-Level estimation suffer from error peaks higher than 100 %. A more complex and certainly more time-consuming approach is using a VHDL Simulator as in [8]. The runtime is not mentioned, but since it is based on a detailed analysis, it would not meet our requisites. However, the estimation is very accurate with a mean error of 0.89 % and a peak error of 3.39 %.

III. CGRA Framework and Heterogeneity

In this section gives a brief introduction to the architecture. A more detailed explanation can be found in [1].

A. Micro-Architecture

The overview of the CGRA and its PE structure is illustrated in Fig. 1. Each PE holds a Registerfile (RF) that is read combinatorially and an ALU, which can process or bypass data from neighboring PEs and its own RF. The ALU contains multiple operator modules, each realizing one operation. There are operations like an INT-addition that are combinatorial modules and operations like FLOAT-Division that are realized in multiple cycles. $live_{in}$ is used to load parameters for invocations. Local and global result buffers are used to return results to the host processor. The port *out* is used to forward data to other PEs and the result buffers. The interconnect is realized as an undelayed[2] and direct wiring from the output of one PE to the inputs of other PEs, as it is shown in Fig. 3. All configurations, e.g. addresses to the RF or the op-code for the ALU are loaded every cycle from an individual Context Memory for each PE. All Context Memories are driven by the Context Counter (*ccnt*) that is generated in the Context Control Unit (CCU), as shown in Fig. 1.

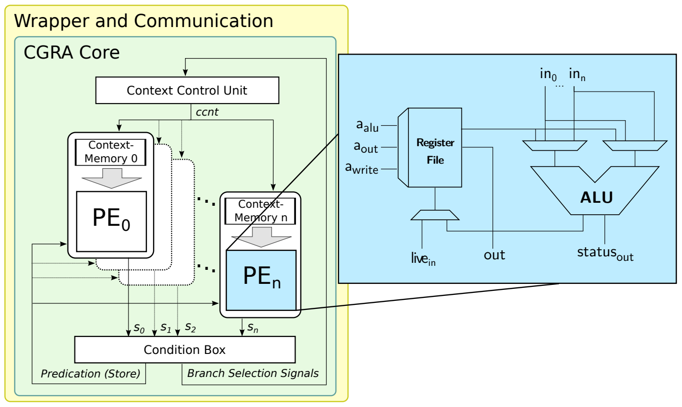


Fig. 1: Overview of the CGRA Core and the Wrapper that includes system specific communication interfaces.

The architecture can handle any kind of control flow by using speculation and predication. Therefore, PEs can process comparison operations (if the related operator is included) and feed a status signal $(S_0, S_1, S_2...S_n)$ to a Condition Box (CBox). The CBox combines these signals equivalent to the logical operation in the code. Then, predication signals for stores in RFs are used, allowing only valid branches to write back their results. Switching the *ccnt* (jump) is done in the CCU based on the Branch Selection Signal (BSS).

[2]General optimization of the microarchitecture, e.g. the insertion of pipeline registers on the interconnect, shall be discussed in later publications.

B. CGRA Framework for Arbitrary Heterogeneous Structures

In order to search the design space, arbitrary CGRA compositions can be modeled. Table I lists the main parameters and their range, by which a composition is defined. Depending on the parameterization, the framework can realize any kind of feasible CGRA. *Global* states a parameter's equality for all components, e.g. the context-memory size. A *local* parameter can be chosen for each component individually. The column *This Contribution* in Table I denotes the parameter's range for the presented work. Concerning the BSS of the CBox, a combination using a multiplexer allows up to 5 points in the CBox, where the BSS can be selected from. Zero denotes that there is no branch selection signal. Two of them are in front of an internal memory, three are behind the memory.

TABLE I: List of main parameters of a composition

Parameter	Global/Local	Supported Range	This Contribution
#PEs	-	2 - ∞	2 - 49
Interconnect	-	any feasible	any feasible
Set Operator	local	1 - 128	1 - 128
Context size	global	2 - ∞	64 - 4096
RF size	local	1 - ∞	1 - 256
Local buffer size	local	0 - ∞	0 - 256
Global buffer size	global	0 - ∞	32 - 256
CBox BSS	-	0 - 5	0 - 5

IV. Reference Database

The critical path of heterogeneous and irregular CGRAs can only be exactly determined with the lengthy process of first generating Verilog, then synthesizing and implementing. Since this takes too long for an iterative DSE, we explore clock frequency estimation based on the CGRA-Model.

A. Design Space and Challenges

For this contribution we have limited the number of PEs to 49, since finding CGRAs with over 40 PEs fitting on the targeted FPGA becomes difficult. The dimension of the resulting design space with a valid interconnect can be calculated with equation 1. The interconnect can be interpreted as the number of unlabeled strongly connected digraphs with n nodes, while the operators can be modelled as a simple distribution problem.

$$\begin{aligned} \mathbf{size_{designspace}} &= 2^{nrPEs\times(nrPEs-1)} \times 2^{nrPEs\times nrOps} \\ &= (2^{49\times(49-1)})2^{49\times128} = \mathbf{1.21 \times 10^{2596}} \end{aligned} \tag{1}$$

The given amount of combinations clearly demonstrates the massive size of the design space. Moreover, there is another factor which makes the estimation even more challenging: due to heterogeneous PEs and interconnect, we have found 43 different types of critical paths (more details in Section V-A). A critical path can lie within a PE's multi-cycle operation or almost randomly be spread over the interconnect. This leads to an increase of complexity, particularly in comparison to other approaches which exclusively estimate well-known path types like [2].

B. *Creating the Reference Database*

As a first step, we create a large reference database to allow a fundamental analysis of the design space. A fully automated and parallelized tool randomly generates legal CGRA compositions. Then, the related Verilog code is generated and run through Xilinx Vivado. In order to avoid synthesizing CGRAs that will surely not fit onto the FPGA, we have implemented a LUT estimator, which linearly adds up reference values (LUTs) for each operator in the composition. Finally, the timing report is read in and all relevant information is stored in a reference data entry on disk, see Table II.

TABLE II: Main information in a reference entry

Entry	Contained Information
Path 1-20	20 most critical paths incl. routes, delay and slack
Platform	FPGA device, here: xc7z045ffg676-2
LUTs	Consumed and total # LUTs
BRAMs	Consumed and total # BRAMs
Registers	Consumed and total # Registers
DSPs	Consumed and total # DSPs

For each CGRA the timing constraint is iteratively adapted until the highest frequency is found. It took four months to generate the reference set on two not-exclusively used server class machines (2x Intel Xeon E5-2690 and AMD EPYC 7501) with 32 Vivado instances, leading to 12.131 synthesized compositions. Random generation avoids patterns in generation, but may lead to an unequal distribution. Hence, the tool uses a coarse multidimensional grid to enhance equal distribution and ensure good coverage of the design space. Grid dimensions are: # PEs, # OPs per PE and their standard distribution, interconnect density, standard dist. of # Inputs per PE and # LUTs.

C. *Timing Report Interpretation*

Path information in Vivado Timing Reports often contains naming inconsistencies. We call them artifacts. For instance, they occur when two elements of different modules are packed into the same CLB. Then both elements names are extended with the same hierarchy module. This happens due to hierarchy flattening. As another example, artifacts may appear for wires that drive more than one element, since routing elements are named after the sink. Fig. 2 shows the original path as found in the timing report and below the legalized path. It can be seen that there is a suspicious RF from PE2 (red), duplicate ALU entries for PE2 (green) and a missing CCU (yellow).

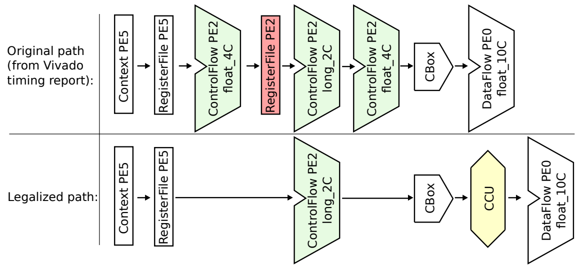


Fig. 2: Cleaning of naming artifacts in Xilinx Vivado reports.

Artifact correction is realized by a large set of sequential functions gradually legalizing the path. These are basically filters forcing a path to stick to the architecture of the CGRA. I.e. multiple duplicate and suspicious control flow ALU operations prior to the CBox are resolved. Impossible RFs are removed and a missing CCU, which combinatorially generates the eventual abort signal for a multi-cycle ALU operation, is inserted after the CBox in example 2. This correction approach has limits, such as a low probability of artifacts transforming a path into a wrong but legal path.

V. Analysis of the Reference Set

One needs to understand the complexity and characteristics of a problem, before proposing an estimation approach. Hence, we analyze the reference set in order to gain first insights.

A. *Types of Critical Paths*

In the reference data set, we found a total of 43 path types. Following are some examples. Fig. 3 can be used to track the described path possibilities. Most paths start at a PEs Context Memory and continue with the PEs RF. Then the path may continue (RF is read combinatorial) to the local or to a neighboring ALU, and may end at a multicycle operator (Path A) or continue to the RF (Path B). Alternatively, the path may continue to the CBox, if the operator is a comparison operator. There, the path might end (Path C) or continue (Paths $D_{1...n}$) depending on whether at least one BSS selection option is in front of the CBox Condition Memory. If it continues, the next possible ending may be the CCU (Path D_0). Otherwise, the path continues and ends at an arbitrary ALU-operator (as an abort signal for a multi-cycle operator - not shown in Fig. 3) or at a Context Memory of the OCM- (D_2), LogBuffer, PE (D_1) or Actuator (D_3).

B. *Correlating Model Parameters and Critical Path*

There is another type of critical path, which is located in a single multi cycle operator in the ALU. An intuitive guess would be that the maximum frequency should be in a fairly narrow range for different CGRAs if their path is in the exact same operator. As an example, we manually filtered the reference database and identified all CGRAs with the most critical path in a DOUBLE-ADD operation. Unfortunately, the path delay was neither in the same range nor could it be clearly interpreted by trying to find correlations to the CGRA's parameters. The same result can be observed for critical path types in multiple other operators. Some of the CGRAs have a smaller value in all parameters but result in a longer critical path. This finding disallows the critical path estimation strategy of linearly adding up static or individually pre-computed delays even for short and well-known paths as it is presented in [2]. We expect the search for correlation to be even more difficult for more complex paths.

C. *Different Path Types in the same Composition*

Another observation is the presence of more than one path type within the 20 most critical paths for a single composition. During synthesis, placement obviously searches for a solution with a lowest maximum delay, meaning most critical paths get prioritized. Therefore, their final delay comes closer to the delay of less critical paths. This complicates the isolation of path types and the correlation to composition parameters.

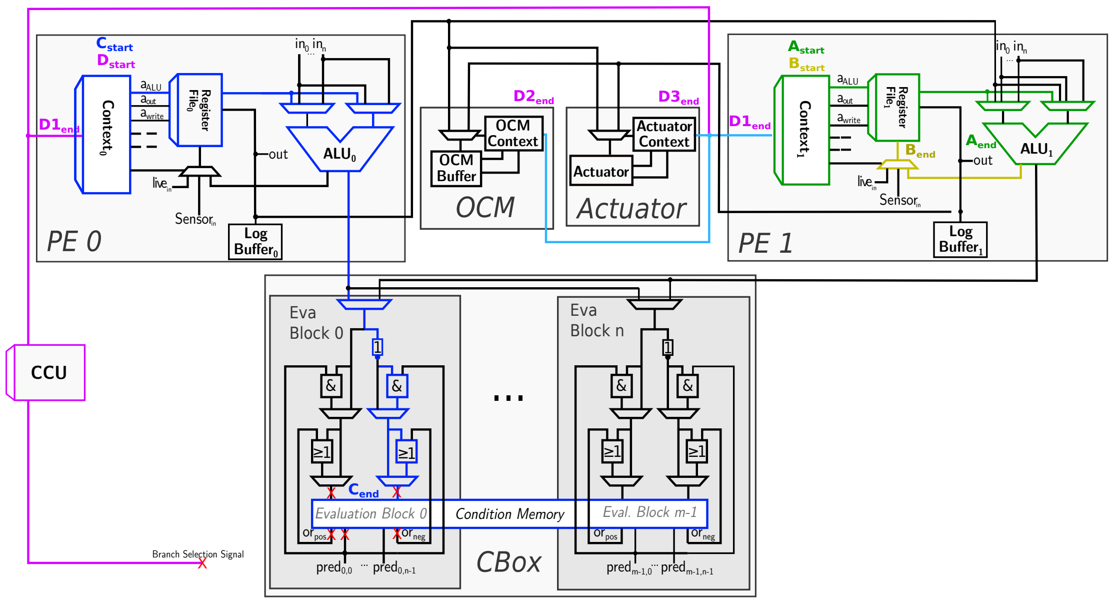


Fig. 3: Examples A - D of the possible critical paths. Only the context memories are read sequentially.

Since the underlying FPGA is bit accurate whereas the overlay CGRA has a word-width of 32 bit, the same path may occur multiple times within Vivado's timing reports. When analyzing the first 20 critical paths, we noticed that 5.402 % of the compositions contain the same path multiple times (different bits), while the amount of different paths per model is 4.034 on average. 13.985 % of the reference models contain only one type of critical path and the average amount of different path types per model is 2.965.

D. Multidimensional Parameters and Visualization

To answer the question which parameterization results in what type of critical path, we have implemented a tool that takes a reference data (sub-) set and outputs table documents for manual analysis. To break down the multi-dimensionality of parameters, the search space is divided into user-defined segments which can be bound to #PEs and/ or #LUTs. Each of these segments can then be easily analyzed concerning the percentage of paths per type. Fig. 4 shows the occurrence of the path type of the single most critical path in the reference set. The tool allows even deeper analysis not discussed here.

E. Evidences

To sum up, the identification of all correlations between a composition, the resulting critical path type and its delay is cumbersome and unpromising. Hence, an analytical approach seems pointless, even though this would be our favored method. The use of a neural network might be an alternative, but preliminary tests have shown that a neural network needs more than 12k compositions for training.

VI. A Statistical Estimator

The proposed statistical estimator relies on the insight that CGRAs with certain similar parameters likely result in quite similar clock frequency. As shown in Figure 5, we pick the most similar compositions and calculate a weighted KDE.

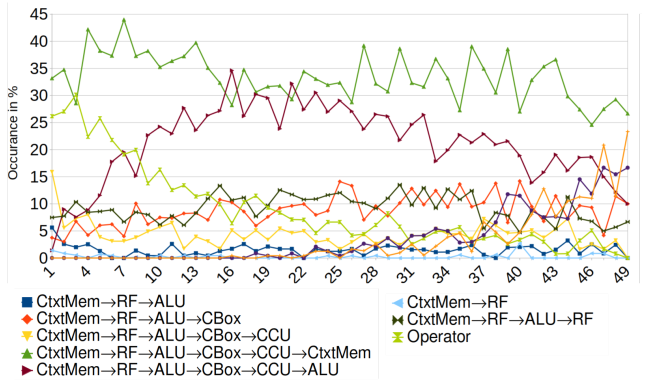
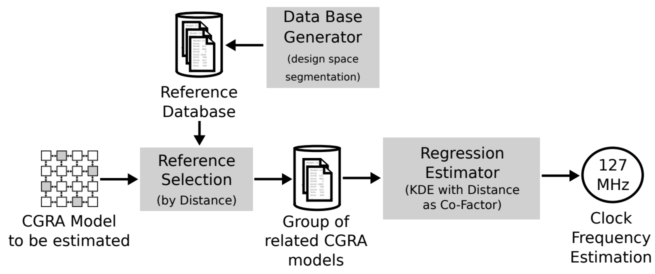


Fig. 4: Probabilities of critical path types from 2-49 PEs.



Data Base Generator
(design space segmentation)
Reference Database
CGRA Model to be estimated
Reference Selection
(by Distance)
Group of related CGRA models
Regression Estimator
(KDE with Distance as Co-Factor)
127 MHz
Clock Frequency Estimation

Fig. 5: Estimation toolflow: First reference selection, then regression using KDE.

A. Implementation

The estimation is based on the previously described database containing synthesized and implemented references covering the design space. The parameters selected for analysis are the ones used during reference database creation for distribution levelling (Section IV) plus 15 additional parameters which are listed in Table III. These were picked based on the greatest correlations in the previously mentioned analysis of the reference set. Empiric test runs showed that providing only a subset of these parameters may lead to better results in some parts of the design space while degrading estimation quality for other parts.

TABLE III: Additional parameters for reference selection

- #of interconnects	- max. #of PE inputs
- max # of estimated LUTs for a single PE	- standard dev. est. #LUTs per PE
- max RF size	- standard deviation RF sizes
- sum of RF sizes	- max context width
- standard dev. of context widths	- sum of context widths
- max Log Buffer size	- max OcmBuffer size
- size of Context Memory	- #PEs containing CF OPs
- empiric sum for specific CF path type	

The estimation is described in Algorithm 1. As a first step, we distinguish between compositions with and without a combinatorial BSS, effectively cutting search space in half. Next, we compute a multidimensional euclidean distance value to create a group of 10 closest CGRAs using the parameters above. The parameters are normalized beforehand. For each member of the group, we take the critical path delay and place it on the x-axis of a diagram (see Fig. 6). We then draw a Hann-Curve over the center of each of these x-values and scale it with the inverse of its multidimensional distance normalized to 1, thus decreasing the impact of far-away references. The width of the Hann-Curve stays constant for all entries and equals the difference from longest to shortest reference's critical path delay. Hann-Curve is picked as it yields best results with low computational complexity. As a last step the curves are summed up and the x-value that is coupled to the highest y-value is returned. This is the point with highest kernel density.

Algorithm 1: Statistical estimation algorithm

```
Data: Reference CGRAs, CGRA to estimate
Result: Clock period estimation
1  load CGRA to estimate;
2  if CGRA has combinatorial BSS then
3      load reference CGRAs with combinatorial BSS;
4  else
5      load reference CGRAs with non-comb. BSS;
6  for all loaded CGRAs do
7      for each relevant parameter do
8          compute normalized dimension value;
9  select 10 nearest CGRAs in multidimensional space
     using euclidean distance;
10 width of Hann-Curves = difference from fastest to
     slowest reference;
11 for each CGRA in group fill up a KDE graph do
12     x = critical path delay;
13     y = Hann-Curve scaled height with inverse
         distance to CGRA to est.;
14 sum up all 10 curves;
15 return x-value of highest y;
```

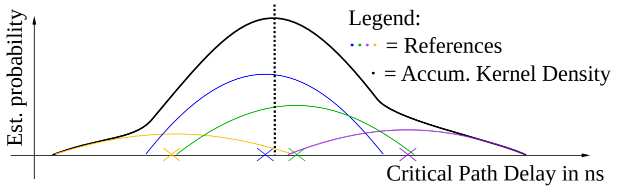


Fig. 6: Regression with KDE using a group of similar CGRAs.

VII. Evaluation

A. Timing Report Interpretation

Artifact filtering increases the percentage of legal timing report paths from 17.96 % to 99.30 % for the reference set. The term "legal" does not directly imply "correctness". There is a risk that an illegal path mutates to be incorrect but legal. However, we could not find wrong cleanings based on a manual verification.

B. Estimation Accuracy

The estimation accuracy correlates with the desired region of the design space. Accuracy also depends on the amount, density and distribution of the reference data. For models with 9 PEs we could observe, that the estimation accuracy stays almost constant for reference data amounts between 256 and 1152. When using less than 256 reference entries, the mean accuracy only worsens slightly. Instead, the risk of a CGRA-to-estimate lying outside the relatively small reference database rises, leading to error peaks.

The clock frequencies in the used reference set ranges from 55 MHz to 190 MHz. We have realized two estimators - the first one for CGRAs with a combinatorial BSS, see Figure 7 and the second one for CGRAs with non-combinatorial BSS, see Figure 8. There is a peak of inaccuracy for CGRAs with 6 or less PEs. Fortunately, most applications mapped on CGRAs scale well with a growing amount of PEs, as shown by Rákosy et. al. [9] for instance. Eventually, if an application does not benefit from a larger CGRA, e.g. a modern (embedded) CPU might be a more suitable choice anyhow. Regarding the use of SA in a DSE, the run-time of a given application(-set) is most likely used as the cost-function. In test runs of SA, we could notice that the scheduling length dictates the run-time of an application for PE amounts lower than 9. Therefore, the relevance of the lower end of the design space in terms of PE amount is low and can be arguably neglected. Estimations for more than 6 PEs all lie below 17.37 % with a maximum mean error of 4.63 % (see Figure 8). Results for CGRAs with a combinatorial CBox are even better. In certain regions of the design space a maximum error of 8.83 % and mean errors of around 1.9% are reached (see Figure 7). The critical paths in CGRAs with an non-combinatorial BSS are usually shorter causing the relative error to increase. The same holds for CGRAs with fewer PEs, which explains an increase of accuracy for an increase in PE numbers. All presented values result from completely separated reference and verification sets. Both sets were created by different runs of the database creation tool in order to equally cover the design space. Amounts per region are shown in Table IV.

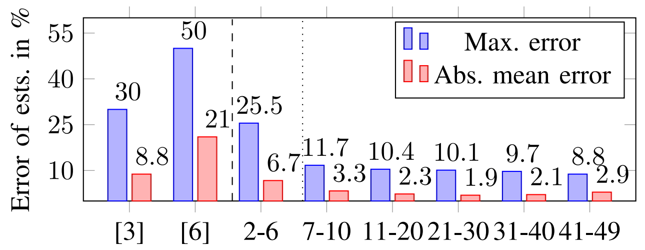


Fig. 7: Estimation Error for CGRAs with combinatorial BSS.

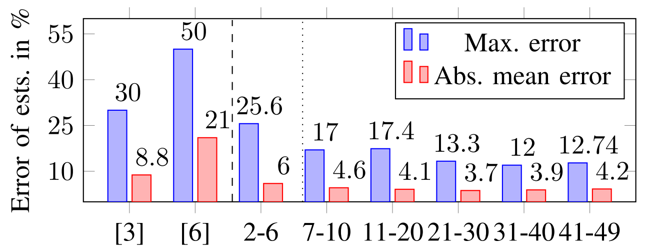


Fig. 8: Estimation Error for CGRAs with non-comb. BSS.

C. *Run-time and Complexity*

We expect the built-in sorting algorithm of the frequently used SortedMaps within the reference selection to be the most complex part of our estimation process with $O(n \times (\log(n)))$. The following barchart (Fig. 9) presents run-time per region of the setup, including loading of training and verification data for CGRAs with combinatorial BSS. The graph shows the total run-time per PE-region for estimations. Table IV contains the corresponding amounts of reference and verification data as well as the mean run-time per estimation and appends run-time information for CGRAs with non-combinatorial BSS. All measurements were taken on an AMD RYZEN 1600@3.2 GHz with activated boost and 32 GB of memory.
Run-time not only depends on the amount of reference data, but also correlates with CGRA complexity, particularly the amount of PEs. This is mainly caused by the not run-time optimized linearly adding-up LUT-estimator which we use per PE per Model. Preparation and estimation run-time may be further improved by increasing parallelism. This work is partially implemented in an estimator-extension which optimizes parameter weights with SA, reaching single-digit ms run-time per estimation.

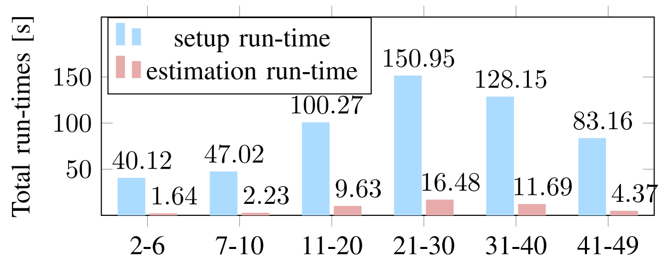


Fig. 9: Total setup and estimation run-times per region for combinatorial BSS dataset.

TABLE IV: Number of references, estimations and mean run-time per estimation per region

# PEs	2-6	7-10	11-20	21-30	31-40	41-49
# References comb.	1561	1198	1510	1537	1193	765
# Estimations comb.	95	75	190	204	137	52
Run-time per est. in ms	**17.26**	**29.73**	**50.68**	**80.78**	**85.33**	**84.04**
# References non-comb.	730	596	496	534	407	316
# Estimations non-comb.	63	53	129	112	93	51
Run-time per est. in ms	**17.81**	**27.11**	**46.31**	**77.04**	**82.95**	**77.31**

VIII. Conclusion

The estimation of the clock frequency of heterogeneous CGRAs is complex due to the massive design space and multiple different paths that can occur. Using a segmentation of the design space, reference data can be generated systematically and used for a statistical approach. By choosing references of similar compositions, any composition above 6 PEs can be estimated with a mean error of 1.9 - 4.6 % and a maximum error below 17.4 %.

This research work has been funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

References

[1] D. Wolf, T. Ruschke, C. Hochberger, A. Engel, and A. Koch, "UltraSynth: Integration of a CGRA into a Control Engineering Environment," in *Applied Reconfigurable Computing*, C. Hochberger, B. Nelson, A. Koch, R. Woods, and P. Diniz, Eds. Cham: Springer International Publishing, 2019, pp. 247–261.

[2] K. Niu and J. H. Anderson, "Compact Area and Performance Modelling for CGRA Architecture Evaluation," in *2018 International Conference on Field-Programmable Technology (FPT)*, Dec 2018, pp. 126–133.

[3] L. Yan, T. Srikanthan, and N. Gang, "Area and Delay Estimation for FPGA Implementation of Coarse-grained Reconfigurable Architectures," *SIGPLAN Not.*, vol. 41, no. 7, pp. 182–188, Jun. 2006. [Online]. Available: http://doi.acm.org/10.1145/1159974.1134677

[4] D. Suh, K. Kwon, S. Kim, S. Ryu, and J. Kim, "Design Space Exploration and Implementation of a High Performance and Low Area Coarse Grained Reconfigurable Processor," in *2012 International Conference on Field-Programmable Technology*, Dec 2012, pp. 67–70.

[5] A. Monostori, H. Holm Frühauf, and G. Kókai, "Quick Estimation of Resources of FPGAs and ASICs Using Neural Networks," in *LWA*, 01 2005, pp. 210–215.

[6] D. Chen, J. Cong, Y. Fan, and Z. Zhang, "High-Level Power Estimation and Low-Power Design Space Exploration for FPGAs," in *2007 Asia and South Pacific Design Automation Conference*, Jan 2007, pp. 529–534.

[7] R. Enzler, T. Jeger, D. Cottet, and G. Tröster, "High-Level Area and Performance Estimation of Hardware Building Blocks on FPGAs," in *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*, R. W. Hartenstein and H. Grünbacher, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 525–534.

[8] M. L. J. Sokolovic and V. B. Litovski, "Using VHDL Simulator to Estimate Logic Path Delays in Combinational and Embedded Sequential Circuits," in *EUROCON 2005 - The International Conference on "Computer as a Tool"*, vol. 2, Nov 2005, pp. 1683–1686.

[9] Z. E. Rákossy, D. Stengele, G. Ascheid, R. Leupers, and A. Chattopadhyay, "Exploiting scalable CGRA mapping of LU for energy efficiency using the Layers architecture," in *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct 2015, pp. 337–342.