High Level Quantitative Interconnect Estimation for Early Design Space Exploration

Roel Meeuws, Kamana Sigdel, Yana Yankova, Koen Bertels
Computer Engineering, EEMCS
Delft University of Technology
{r.j.meeuws,k.sigdel,y.d.yankova,k.l.m.bertels}@tudelft.nl

Abstract

In this paper, we present an approach for prediction of interconnect resources at the early stages of design. This approach was developed as an extension to the Quipu Multi-Dimensional Quantitative Prediction Model for Early Design Space Exploration. Quipu is a part of the Delft Workbench project, a semi-automatic tool platform supporting integrated hardware–software co–design for heterogeneous computing systems. Because of the highly iterative nature of design in such tool platforms, fast and early estimates of hardware properties are required. One aspect of particular importance is the utilization of interconnect resources, which has increased with designs becoming larger, even to the point where some designs are no longer routable. We establish a method of estimating interconnect from a C-level description using Partial Least Squares Regression (PLSR) and Software Complexity Metrics (SCM) for use in the Delft Workbench tool platform. We show that our approach can make predictions with an expected error of 31.6%.

1 Introduction

As Reconfigurable Computing (RC) is gradually moving from research to industry, the need for tool support assisting designers in developing applications for reconfigurable architectures increases. The unique difficulty in creating such tools lies in the combination of hardware and software development paradigms. Indeed, engineers that are proficient in both paradigms are scarce and as such tools that bridge the gap between the two worlds are essential for the acceptance of RC by a broader community.

Delft Workbench, a tool platform in support of integrated hardware–software co–design targeted at RC, aims to fulfill this need starting from profiling and partitioning to synthesis and compilation. At the early design stages one of the main obstacles for such a tool platform is the lack of details about hardware performance and costs.

As a part of the Delft Workbench project we have developed a quantitative model called Quipu with the purpose of providing hardware cost and performance estimates in the very early stages of design [7]. Because Quipu is able to make estimations from High Level Languages (HLL) like C, it can drive early hardware–software partitioning, but it also helps designers in re-factoring code, estimating project feasibility, or driving optimization.

In this paper we present an extension to our Quipu model that introduces interconnect estimation. With device and design sizes increasing rapidly, interconnect is becoming an important factor. For example, routers can encounter severe problems when routing a design that consists of large amounts of nets, even to the point where the design is unroutable. Early interconnect estimates indicate whether a kernel could pose such problems. Furthermore, interconnect is an important factor in determining the latency and clock period of a design. The main contributions of this paper are:

- we present a high level interconnect estimation scheme targeting High Level Languages like C.
- we use GLM modeling in order to identify the SCMs related to interconnect.

The organization of this paper is as follows. First, we review related research and projects in § 2. Then, we discuss the theory and background of the Quipu model in § 3. The experimental setup and methodology are described in § 4. Subsequently, we present the results of our model in § 5. Finally, we conclude this paper in § 6.

2 Related Research

Interconnect estimation is not a new concept, particularly in relation to placement and routing. One early approach is the use of Rent's Rule[6], which relates the number of pins with the number of logic blocks. But also several other schemes based on e.g. PathFinder[4], reconvergences [1], or local neighborhood analysis[3].

Nevertheless Scheffer and Nequist argued in [8] that in-

terconnect estimation at this level is not viable. The main reasons they mention are the inability of statistical modeling to capture properties based on a small number of nets and the unpredictable nature of the routing process. However, they do express the need for estimation schemes working from a HDL or HLL description, like VHDL or C.

Although, some high level estimation schemes for other properties than interconnect have been presented targeting area estimation[5] or execution time[2], there are to the best of our knowledge no other hardware estimation schemes at such high levels. Particularly, there is a lack of high level interconnect estimation schemes.

3 Quantitative Model: Quipu

As mentioned before, early estimation of hardware costs and performance are important in designing applications for heterogeneous and reconfigurable computing systems. Quipu , our Multi–Dimensional Quantitative Prediction Model for Hardware–Software Partitioning, aims to provide this functionality. Making such hardware estimates from a software–level description requires that Quipu quantifies the relation between hardware and software. We have demonstrated that such a quantified relation can be established in [7]. In this section we outline the basic concepts and theory behind the Quipu model.

3.1 Software Metrics

Quipu tries to capture the relation between hardware and software using SCMs. These metrics represent intrinsic aspects of computer programs and kernels as program size, control intensity, or data usage. As we have shown in [7] SCMs also relate to hardware area costs. Currently, our model utilizes a set of 48 different SCMs like the number of paths or the cyclomatic complexity. For a detailed description of these metrics please refer to [7].

The main problem of using SCMs is their inherent multicollinearity. This phenomenon arises when different metrics measure the same quality. For example, LOC and the number of statements both measure the length of the program in some way. One approach to solve this is to transform the data into a set of independent components using Principal Component Analysis [7] (PCA). Otherwise one can manually select a set of independent metrics. In the latter case the contribution of each metric can be better explained. However, PCA techniques often yield better results.

Because not every metric necessarily relates to hardware costs in a linear fashion, it is also important to linearize the metrics by applying the appropriate transformations. As an example, note that the number of paths (n_p) in a kernel grows exponentially with the number of subsequent loops. However, consecutive loops do not imply an exponential growth in hardware. In fact, linear regression improves significantly if the n_p data is transformed with a double logarithmic transformation ($n'_p = \log(\log(n_p + 1) + 1)$)

Hardware Property	$R^{2\,a}$	Expected Error	Regression technique
Flip-Flops	0.99	28%	PLSR
Slices	0.98	71%	GLM, manual[b]
Multipliers	0.98	30%	GLM, manual, transform[c]
States	0.96	67%	GLM, manual

[a] R^2 expresses the amount of variance explained by the model
[b] manually selected metrics
[c] all data was processed with a log-transformation

Table 1: Summary of the different sub-models in the Quipu model characterizing hardware area.

3.2 Regression Techniques

In order to formulate a quantitative model, we use linear regression on a dataset of SCMs and hardware cost measurements from a library of software kernels. In previous work [7] we considered classical linear regression, however, the error margins of 67%–129% were not encouraging. In our current model we employ more advanced techniques, like Generalized Linear Modeling (GLM) and Partial Least Squares Regression (PLSR). For our interconnect extension we also apply these more advanced techniques. The first technique relaxes the need for normally distributed data, accepting any distribution from the exponential family. The other method tries to find linear components of the independent variables that maximize the covariance between the dependent and independent variables, solving the multicollinearity problem.

3.3 Criteria

The Quipu model is tightly coupled with the design methodology it targets, i.e. it is built using data generated with the targeted tool chain. This allows for a higher accuracy in the context of the targeted tool chain. However, it also implies the model is not directly applicable to other tool chains without rebuilding the dataset. It also implies some target platforms might require different criteria to be predicted than others. Currently, Quipu targets Xilinx FPGAs in the Virtex family.

Quipu consists of several sub–models for each hardware property. Outside our interconnect extension, several area properties, like slices, flip-flops, multipliers, and states are available. A summary of previous results can be viewed in Table 1. Although one could argue that the error is quite large, one should consider Quipu targets the very early stages of design..

3.3.1 Interconnect

The Quipu model, as part of the Delft Workbench, targets estimation for FPGAs. Within this context, there are several properties we can try to estimate, like Total Wire Length, Average Wire Length, or Reachability. However, in our pa-

Domain	Kernels	Bit-Based	Streaming	Account-Keeping[a]	Control-Intensive
Compression	2	x		x	x
Cryptography	56	x	x		x[b]
DSP	5	x	x		x[b]
ECC	6	x	x		x
Mathematics	19				
Multimedia	32	x[b]	x		x
General	15			x[b]	x
Total	135				

[a] Non-constant space complexity.
[b] Only some instances in that domain express this characteristic.

Table 2: Number of functions in each domain with the main algorithmic characteristics present in each application domain.

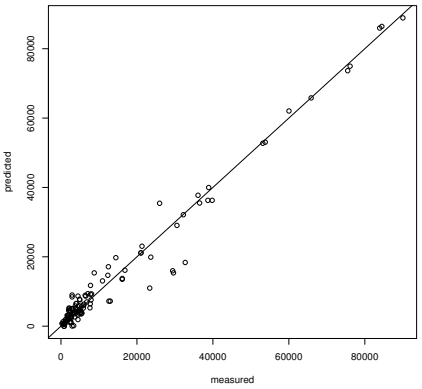

Figure 1: Actual number of nets vs. predicted number of nets for the GLM model.

per we assume the size and location of the nets in a design are evenly distributed, as well as that the behaviour of the router is not predictable from the software-level. Therefore, we consider the number of nets as a quantification of the number of used interconnect resources.

4 Methodology and Setup

In order to define our model we have gone through the following steps. Previously, a library of 135 C-functions from a wide range of application domains was created for Quipu, as summarized in Table 4. By using source code from existing applications from many different areas in computing, we aim to build models that are generally applicable. In order to determine the SCMs from this set of functions we instrumented the Elsa/Elkhound GLR parser and compiler front-end from UC Berkeley. We used the instrumented parser to generate the dataset of SCMs from the library of functions. On a 2.4GHz AMD Athlon64 this process took 7.5 seconds to complete.

We then used the DWARV [9] C-to-VHDL compiler to translate all 135 kernels to VHDL. The VHDL code was then synthesized, mapped, placed, and routed for the VirtexII Pro FPGA (XC2VP30-7-ff896) using the Xilinx ISE $9i$ workflow optimizing for speed. Six kernels did not compile, two other kernels were not synthesizable, and 9 more kernels were not routable in a reasonable amount of time. Thus we collected 118 observations for regression analysis.

5 Experimental Results

In this section we present the results of the regression analyses. We present both the results with GLM and PLSR in order to choose the best model. Furthermore, we compare our approach with other interconnect estimation schemes.

The results for the GLM model can be seen in Fig. 1 and Table 3. Fig. 1 depicts the prediction quality of the model and it shows the linear relation between the metrics and the number of nets close to the ideal $x = y$ line. The GLM model had an expected error of 46.7%, which is comparable with other Quipu estimation models as can be observed in Table 1.

We observe from the p–values[1] that in particular the number of definition–use pairs in the code is a significant metric. This can be explained when we consider that exactly the transfer of values (*definitions*) via variables to their location of *use* requires interconnect resources. Another significant set of metrics are related to program size. It stands to reason that the larger a kernel becomes, the more interconnect it will require. Furthermore, the AICC metric, which represents the entropy or variation in the operators in the code, relates to interconnect, which might be explained by hardware reuse.

We have also generated a PLSR model as depicted in Fig. 2. In order to choose the number of components the PLSR model we evaluated the Root Mean Square Error for Prediction (RMSEP) for each number of components. The minimum RMSEP was attained when 9 components are used. In Fig. 2, we can see the performance of the PLSR model with 9 components using Leave-One-Out (LOO) prediction. LOO, is a technique where each data point is predicted using regression on the remaining data points. This way the actual prediction error instead of the fitting error can be obtained. The prediction error in the PLSR model was 31.6%.

When we compare our model with previous interconnect estimation approaches, as in [4, 1, 3], the first thing that stands out is the comparably large error. In contrast with the other approaches that target pre– and post–placement estimation, the Quipu model targets the very early stages of design. Thus, the added advantage of early estimation

[1] The chance that the relation described by the estimation model was obtained by pure chance.

Parameter	Coefficient	σ^2	p-value
(Intercept)	-4355.347	1859.184	0.020981
Avg. Information Content (AICC)	2492.660	596.432	$5.96e-05$
Avg. number of expressions per basic block	-16.125	2.360	$5.12e-10$
Max. number of expressions per basic block	30.294	4.308	$1.94e-10$
Number of basic blocks	399.182	51.358	$4.79e-12$
Total number of expressions	-8.002	2.120	0.000262
Number of multiplications	-623.877	158.546	0.000147
Total bit-width of multiplications	31.007	3.901	$1.97e-12$
Number of paths (NPATH [a]	-2927.119	864.924	0.000996
Oviedo Definition-Use pairs	16.668	1.368	$< 2e-16$

[a] transformed with a double log-transformation

Table 3: Summary of the independent variables of the Interconnect model based on GLM.

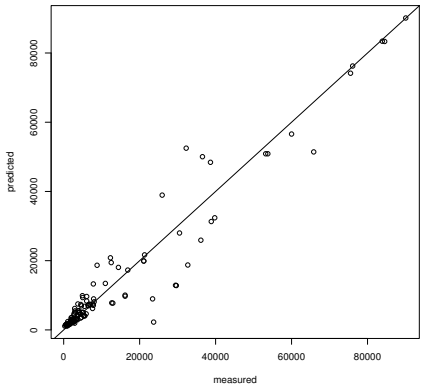

Figure 2: Actual number of nets vs. predicted number of nets for the PLSR model using 9 components and Leave-One-Out prediction.

Approach	Property	Level	Error
Kannan Bhatia [4]	Total Wire Length	post-placement	1.4%
Kannan Bhatia [3]	Total Wire Length	pre-placement	8.2%
Balachandran Bhatia [1]	Total Wire Length	pre-placement	11.6%
Quipu Interconnect (GLM)	number of nets	High Level (software)	46.7%
Quipu Interconnect (PLSR)	number of nets	High Level (software)	31.6%

Table 4: Overview of different interconnect estimation approaches compared to Quipu.

justifies the increased error of 31.6%. Furthermore, we observe that our model currently estimates the number of nets, whereas the other approaches focus on the wire length of those nets. Also, the data suggests the use of a PLSR model as opposed to a GLM model for interconnect estimation.

6 Conclusions

In this paper we have introduced an extension for interconnect estimation to the Quipu Multi-Dimensional Quantitative Prediction Model. The presented model uses statistical modeling techniques based on SCMs in order to predict hardware properties from a HLL like C in contrast with other schemes that operate on pre– or post–placement level. The extension for interconnect has an expected prediction error of 31.6%, which is acceptable in high level design space exploration.

Acknowledgments

Delft Workbench is sponsored by the hArtes (IST-035143), the MORPHEUS (IST-027342), and RCOSY (DES-6392) projects. Also, we would like to thank Michel Vos, Silvian Bensdorp, and Rick van Akkeren.

References

[1] S. Balachandran and D. Bhatia. A priori wirelength and interconnect estimation based on circuit characteristics. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(7):1054–1065, July 2005.

[2] M. Holzer and M. Rupp. Static estimation of execution times for hardware accelerators in system-on-chips. In *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*, pages 62–65, 2005.

[3] P. Kannan and D. Bhatia. Estimating pre-placement fpga interconnection requirements. In *VLSID '04: Proceedings of the 17th International Conference on VLSI Design*, pages 869–874, Washington, DC, USA, 2004. IEEE Computer Society.

[4] P. Kannan and D. Bhatia. Interconnect estimation for fpgas. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(8):1523–1534, Aug. 2006.

[5] D. Kulkarni, W. A. Najjar, R. Rinker, and F. J. Kurdahi. Fast area estimation to support compiler optimizations in fpga-based reconfigurable systems. In *FCCM '02: Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, page 239, Washington, DC, USA, 2002. IEEE Computer Society.

[6] B. Landman and R. Russo. On a pin versus block relationship for partitions of logic graphs. *Computers, IEEE Transactions on*, C–20(12):1469–1479, Dec. 1971.

[7] R. J. Meeuws, Y. D. Yankova, K. Bertels, G. N. Gaydadjiev, and S. Vassiliadis. A quantitative prediction model for hardware/software partitioning. In *Proceedings of 17th International Conference on Field Programmable Logic and Applications (FPL'07)*, page 5, August 2007.

[8] L. Scheffer and E. Nequist. Why interconnect prediction doesn't work. In *SLIP '00: Proceedings of the 2000 international workshop on System-level interconnect prediction*, pages 139–144, New York, NY, USA, 2000. ACM.

[9] Y. D. Yankova, K. Bertels, S. Vassiliadis, R. J. Meeuws, and A. Virginia. Automated hdl generation: Comparative evaluation. In *Proceedings of International Symposium on Circuits and Systems (ISCAS2007)*, May 2007.