Estimating Circuit Delays in FPGAs after Technology Mapping

Berg Severens, Elias Vansteenkiste, Karel Heyse, Dirk Stroobandt

Hardware and Embedded Systems Team, Computer Systems Lab

Department of Electronics and Information Systems

Ghent University, Belgium

{Berg.Severens, Elias.Vansteenkiste, Karel.Heyse, Dirk.Stroobandt}@ugent.be

Abstract—An FPGA implementation requires a significant effort of the hardware designer, who optimizes FPGA designs by going through many time-consuming CAD flow iterations. These iterations provide two types of feedback: (1) the FPGA performance and (2) the identification of the parts having the highest impact on the FPGA performance. Both depend on the wirelength behavior.

Studies have been dedicated to the estimation of local [5] and global [4] wirelengths, but to our knowledge both performance estimations and identification of the critical zone are not present in literature. Therefore this paper, firstly, presents a comparison of three performance estimation techniques: logic depth, Monte Carlo simulation and fast placement (ordered from low to high accuracy and runtime). Secondly, four methods identifying the critical zone are compared. Results show that Monte Carlo simulations provide a good identification of the parts having the highest impact on the performance. We conclude that Monte Carlo simulations provide useful feedback within a short runtime (about 30 times faster than placement), reducing the time-to-market of FPGA implementations.

I. INTRODUCTION

FPGAs are digital chips that can be configured by the customer. Thanks to mass production and configurability, FPGAs are ideally suited for medium-volume electronic applications. The FPGA CAD tool flow is used during the design process, typically comprising five steps [8], see Figure 1. First, HDL code is converted to an and-inverter graph during logic synthesis. Then the graph is mapped onto a network of Lookup Tables (LUTs). Subsequently the packer groups the LUTs into clusters, which are assigned to the FPGA grid in the placement step. Finally routing determines the signal paths between these blocks. This paper presents early performance feedback after technology mapping. As such, the hardware designer is able to finetune the HDL code without executing the slow packing, placement and routing steps.

After technology mapping, early performance feedback is given. As such, the hardware designer is able to finetune the HDL code without executing the slow packing, placement and routing steps.

In this paper we explore the possibility of a shorter feedback loop via accurate timing estimates after technology mapping in order to avoid the time-consuming full iterations (12 hours or more is not uncommon [7]). This optimization cycle can be shortened significantly: logic synthesis and technology mapping are responsible for 23% of the total CAD flow runtime. This has been measured by processing the

Fig. 1. The conventional way to optimize the FPGA's performance is by iterating over the whole CAD tool flow. This paper presents a shorter feedback loop by providing information after technology mapping.

VTR [8] benchmarks through Vivado [1]. Furthermore, this tool provides performance estimates after mapping, yielding a correlation of 0.989 (estimation errors of tens of percents are not uncommon) with the post-routing delays for the same benchmarks. The logic depth, which is a very simple metric, already provides a correlation of 0.979 (in the remaining of this paper, a collection of MCNC and VTR benchmarks will be used). The modest Vivado accuracy indicates that significant improvements to the state-of-the-art estimations might be possible and necessary for a reliable design process.

To our knowledge this is the first paper focussing on estimating total circuit delay after technology mapping. However, similar literature can be found: extensive research considered local wirelength estimations [5] and total wirelength estimations [4]. In [3] fast placement was used as an early timing feedback model for improved technology mapping. A possible reason for the poor exploration of our topic is [9]: this paper discourages interconnect prediction stating that the critical path is determined by the exceptional long wires, which are difficult to estimate. However, Monte Carlo simulations include these exceptions in a reasonably accurate way (see Section III).

The outline of this paper is as follows: in the next section we will discuss the influence of the non-deterministic placement on the post-mapping estimation accuracy. Section III contains two different ways to estimate performance after technology mapping: fast placement and Monte Carlo simulations. Subsequently Section IV contains a discussion about

estimating the impact of nodes on the circuit delay. These impact models help the hardware designer to determine which HDL code should be optimized first. Finally a conclusion will be given.

II. PLACEMENT VARIATION

There are two types of placement variation: delay and structural variation. Delay variation is an upper bound for the accuracy of delay estimations: the decisions made in the placement algorithm affect the final circuit delay. Since we use simulated annealing, placement is not deterministic. Based on experiments with VTR and MCNC benchmarks in VPR, the performance may differ up to 15% for different placements.

Next to the delay variation, there is even more structural variation: based on similar experiments, we concluded that the delays of output nodes (both critical and non-critical) may differ more than 15% compared to the average value in 20% of the cases (see Figure 2). This phenomenon affects the accuracy of routing slacks being used for the impact estimation on the critical path, since the critical path changes according to the placement.

Fig. 2. Our benchmarks were placed 10 times, providing 10 different delays for each output node. Each sample was compared to the average node delay. In 20% of the cases the delay differs more than 15% compared to the average delay. We call this phenomenon ‘structural variation’.

Instead of estimating a single sample of the output delay, we estimated a percentile of the distribution of possible outcomes per benchmark. This gives the designer a clear view on the feasibility of his design. The circuit delays for the 50th percentile were estimated by considering the routing result of the placement with cost closest to the average cost of 10 (trade-off between runtime and accuracy of the distribution of the outcome) placements. All suggested procedures included in this paper can be executed for any percentile. Using the 50th percentile of 10 placements this can be called a semi-deterministic problem, since the variation on the variable is significantly reduced.

Nowadays analytical placement [2] techniques are used in commercial placement tools. Analytical placement is a deterministic algorithm, but if the mapped input circuit changes slightly, the resulting placement can be changed significantly compared to the old placement. Consequently, the problem of structural variation is still relevant.

III. DELAY ESTIMATION METHODS

A. Experimental Setup

Three performance estimation methods are described in this section: logic depth, Monte Carlo simulations and fast placement (ordered from low to high accuracy and runtime). The experimental setup can be described as follows:

- The VTR tool flow is used with a homogeneous architecture (k6_frac_N10_40nm of the VPR architecture collection). We consider a collection of MCNC and VTR benchmarks [8].
- We place these benchmarks 10 different times (as was discussed in Section II). The placement with cost closest to the average placement cost is chosen and the circuit is routed for the chosen placement. The variable to be estimated is the resulting routing estimate of VPR (the 50-percentile of the distribution).
- We use the correlation for comparing the accuracy of different methods. Although this is not an intuitive measure, it is a very general way to reveal the relation between two vectors: it can handle non-linear relationships and has a reasonable sensitivity to outliers.
- The runtimes of the fast placements are given by VPR, which is written in C. On the other hand, our Monte Carlo simulator is written in java. Since java is generally slower than C it is difficult to compare both results. However, the most important conclusion will be that Monte Carlo is at least 30 times faster than the packing and fast placement. Even with the disadvantage of the slower java code this statement holds.

B. Monte Carlo

After technology mapping, the LUT delay is known, which is not the case for the interconnection delay as it depends on packing, placement and routing. We will model the interconnection delays with Monte Carlo simulations. These are widely known for performing probabilistic analysis of various systems. Monte Carlo simulations for our purpose comprise the following steps:

- 1) A general distribution of the connection delays is constructed, i.e. the probability in function of the delay per connection. Our results show that the probability of the occurrence of a connection crossing N multiplexers decreases exponentially with N .
- 2) For each edge in the circuit a stochastic sample is taken from the distribution.
- 3) The circuit delay is calculated as follows: for each LUT, the input signal with the highest delay is taken. The output signal delay of the LUT is given by this delay, increased with the delay of the LUT itself. This procedure is executed from the inputs towards the outputs.
- 4) Repeat steps (2) and (3) N times. The result is a histogram of N possible circuit delays. Thus, Monte Carlo shows multiple possible outcomes per node. The mean of these values will be used as performance estimate.

Note that the results depend on the number of iterations N , providing a trade-off between runtime and accuracy. The Monte Carlo performance estimations of our benchmark suite are depicted in Figure 3 for $N = 100$.

Fig. 3. The post-mapping Monte Carlo delay estimations perform reasonably well. For most estimation techniques, such as the Vivado estimation, the relative errors are high for low-delay circuits. Monte Carlo is able to reduce the impact of this phenomenon significantly, because it implicitly takes into account the internal structure of the netlist. Errors of low-delay circuits have low weights in the calculation of the correlation, but this is an important achievement since industrial benchmarks often have low delays.

C. Comparison

The goal of this paper is to show that Monte Carlo simulations provide useful feedback for the hardware designer within a low runtime. In this section we discuss the estimation delay feedback, considering both runtime and accuracy. The Monte Carlo methodology is compared against logic depth and fast placement. Therefore consider Figure 4 which includes a Pareto graph exploring the different solutions for both accuracy and runtime.

- The logic depth can be found in the upper left corner: very short runtime, but limited accuracy.
- There are several Monte Carlo solutions included for increasing N (number of iterations) from left to right. The runtime is linearly related with N , while the accuracy converges starting from 100 iterations.
- VPR [8] provides a post-placement estimation of the performance. Also in placement (simulated annealing) there is a trade-off between runtime and quality-of-results. However, in this case we search for the best performance estimation instead of best quality-of-results. It turns out that the runtime trade-off for the placement estimation is negligible compared to the remaining coincidence in the placement tool. Therefore we recommend to use fast placement. This is shown in Figure 4 in the right corner below. The runtime is the sum of packing and placement (about the same order of magnitude).

We conclude that all three methods provide a solution that is part of the Pareto-front, whereof Monte Carlo simulations with 100 iterations provide a reasonable accuracy, while the runtime is about 30 times faster than the placement method.

IV. IDENTIFICATION OF THE CRITICAL ZONE

Next to delay estimation, we also want to identify the critical zone of the circuit for short time-to-market development.

Fig. 4. Pareto graph (note the inverted accuracy axis) of three estimation methods: logic depth, Monte Carlo simulations and fast placement (increasing order for both runtime and accuracy)

The critical zone is the collection of nodes having the highest impact on the circuit delay. Our definition of delay impact per node (or collection of nodes) is: the average reduction in circuit delay that is accomplished by removing the node (collection of nodes) out of the network. The accuracy of the delay impact estimate is of crucial importance for the efficiency of the hardware designer: he should know which nodes (corresponding with an HDL code) should be optimized first. Structural placement variation (see Section II) makes it difficult to identify the impact of nodes in the circuit.

Four different metrics are proposed to estimate the delay impact per individual node. Subsequently these metrics are compared quantitatively according to our definition of delay impact per collection of nodes.

A. Four Delay Impact Estimation Techniques

The conventional optimization technique of hardware designers is to use the post-routing slacks of a mapped circuit to rank the impact of nodes. A second technique is completely similar, but with fast placement in order to reduce the feedback time. Note that these two techniques neglect the problem of structural placement variation: one technology mapped circuit can have many different critical zones for different placements (due to the structural differences, see Section II).

The following two techniques are based on Monte Carlo simulations. A widely used Monte Carlo approach is to consider the criticality (for instance in project management [6]): the probability of a node being part of the critical path. Our fourth approach is to identify for each node the likelihood of adding a significant extra delay to the critical path. Therefore we suggest a new metric to evaluate the impact of a node on the circuit delay using Monte Carlo simulations: the "Expected Added Delay" (EAD). If we want to calculate the EAD of LUT L , first we run a normal Monte Carlo iteration resulting in a circuit delay of t_0 . Second, we set the delay of L and all its neighbouring connections equal to zero. The circuit delay is recalculated and is called t_1 . The EAD is equal to the average (over many iterations) difference $t_d = t_0 - t_1$. There are three possible outcomes: (1) L was not on the critical path. Consequently t_0 equals t_1 , thus $t_d = 0$. (2) L was on the critical path and is still on the critical path when its delay is zero. Then t_d equals the delay of L including the delay of

the critical neighbouring connections. (3) L was on the critical path, but when its delay is zero, the critical path changes. In this case t_d is a strictly positive value, but lower than the previous case.

B. Comparison Delay Impact Estimation

This section contains a comparison of the four proposed delay impact estimation techniques. A valid experimental setup to compare different techniques on estimating the delay impact of a node should include the structural variation of the placement (see Section II). Therefore consider the following experimental setup, based on our definition of delay impact per collection of nodes:

- We run the MCNC benchmark suite with the homogeneous VPR architecture *k6_frac_N10_mem32K_40nm* for 10 different placements. The average circuit delays can be found in the column 'Original' in Table 1.
- For each of the four presented techniques, we do the following: the collection of 10 output nodes (or inputs of latches) with highest estimated impacts are cut out per benchmark, as if they were perfectly optimized by the designer. Removed nodes are called "left-out-nodes" in the following. The new mapped circuits are packed, placed and routed all over again for 10 different placements. Table 1 depicts the average delay results per technique.

The results of Table 1 show that the removal of nodes was on average about the same for normal placement and for fast placement. Monte Carlo simulations did slightly better for both the criticality and the EAD. However, this experimental setup has limitations. Probably the most important source of uncertainty is the following: the removal of an output node also removes the inputs of this node. The number of additionally removed nodes can affect the delays of the remaining output nodes. On the other hand this effect should not be overestimated as the correlation between the number of LUTs and the circuit delay is very low (0.37), indicating that the number of nodes does not affect the delay drastically. Due to the remaining uncertainty we do not claim that Monte Carlo is able to predict the impact more accurately than post-routing slacks. But we can state that Monte Carlo might be able to yield a reasonably accurate impact estimate, which is a surprising result for a low-runtime technique.

V. CONCLUSION

This paper explored the trade-off between accuracy and runtime for performance and performance impact feedback information. The accuracy of the estimations is affected by the non-deterministic behavior of the placement algorithm. Firstly, three feedback performance estimations were compared: logic depth, Monte Carlo simulations and fast placement. All three estimations are Pareto-optimal considering both accuracy and runtime, where Monte Carlo simulations provide a good trade-off between the two considered dimensions. Secondly, four impact estimations were compared, indicating that the accuracy of the Monte Carlo techniques is similar to the accuracy of normal post-routing information. We conclude

TABLE I. CIRCUIT DELAYS (NS) WITH 10 LEFT-OUT-NODES FOR DIFFERENT NODE REMOVAL DECISION TECHNIQUES. THE EAD, A LOW-RUNTIME PERFORMANCE ESTIMATION BASED ON MONTE CARLO SIMULATIONS, PROVIDES A BETTER IMPACT ESTIMATION THAN POST-ROUTING SLACKS.

	Original	Normal Placement		Fast Placement		Monte Carlo Criticality		Monte Carlo EAD	
	[ns]	[ns]	[%]	[ns]	[%]	[ns]	[%]	[ns]	[%]
apex4	5.56	5.01	-9.9	4.58	-17.6	4.97	-10.6	4.97	-10.6
bigkey	2.56	2.01	-21.5	2.00	-21.9	2.09	-18.4	2.18	-14.8
clma	8.6	6.57	-23.6	6.83	-20.6	3.69	-57.1	3.69	-57.1
des	3.77	4.11	9.0	3.86	2.4	4.01	6.4	4.00	6.1
diffeq	4.87	5.17	6.2	4.99	2.5	4.85	-0.4	4.84	-0.6
dsip	2.63	2.59	-1.5	2.57	-2.3	2.62	-0.4	2.55	-3.0
elliptic	7.08	7.26	2.5	7.07	-0.1	7.55	6.6	7.19	1.6
ex5p	4.84	4.96	2.5	5.08	5.0	5.06	4.5	4.81	-0.6
frisc	9.17	8.94	-2.5	9.17	0.0	8.76	-4.5	9.19	0.2
misex3	4.74	4.03	-15	4.08	-13.9	4.43	-6.5	4.43	-6.5
pdc	6.91	6.99	1.2	6.96	0.7	6.92	0.1	6.71	-2.9
s38417	5.09	5.83	14.5	5.56	9.2	5.73	12.6	5.35	5.1
s38584.1	4.74	4.53	-4.4	4.53	-4.4	4.92	3.8	4.94	4.2
seq	4.83	4.22	-12.6	4.75	-1.7	4.48	-7.2	4.53	-6.2
spla	6.83	5.6	-18	6.04	-11.6	6.43	-5.9	6.54	-4.2
Geomean	5.16	4.87		4.88		4.79		4.76	
Gain			-5.6%		-5.4%		-7.0%		-7.7%

that Monte Carlo simulations are reasonably accurate for both performance and performance impact estimations within a low runtime. The combination of these properties in a tool can help the hardware designer to reduce the time-to-market of an FPGA implementation.

REFERENCES

- [1] Vivado design suite, 2014.
- [2] M. Gort and J. Anderson. Analytical placement for heterogeneous FPGAs. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 143–150, Aug 2012.
- [3] J. Y. Lin, A. Jagannathan, and J. Cong. Placement-driven technology mapping for LUT-based FPGAs. In *In Proceedings of the ACM Int. Symposium on FPGAs*, pages 121–126, 2003.
- [4] Q. Liu, J. Ma, and Q. Zhang. Neural network based pre-placement wirelength estimation. In *FPT*, pages 16–22. IEEE, 2012.
- [5] V. Manohararajah, G. R. Chiu, P. Singh, and S. D. Brown. Predicting interconnect delay for physical synthesis in a FPGA CAD flow.
- [6] G. Mummolo. Measuring uncertainty and criticality in network planning by pert-path technique. *International Journal of Project Management*, 15(6):377 – 387, 1997.
- [7] K. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz. Titan: Enabling large and complex benchmarks in academic CAD. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–8, Sept 2013.
- [8] J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson. The VTR project: Architecture and CAD for FPGAs from verilog to routing. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA '12*, pages 77–86, New York, NY, USA, 2012. ACM.
- [9] L. Scheffer and E. Nequist. Why interconnect prediction doesn't work. In *Proceedings of the 2000 International Workshop on System-level Interconnect Prediction, SLIP '00*, pages 139–144, New York, NY, USA, 2000. ACM.