# Analyzing the Divide between FPGA Academic and Commercial Results

Elias Vansteenkiste
Department of Electronics and Information Systems
Computer Systems Lab, Ghent University
Ghent, Belgium
Elias.Vansteenkiste@ugent.be

Alireza Kaviani and Henri Fraisse
Xilinx Inc., 2100 logic drive
San Jose, California, USA
{Alireza.Kaviani, Henri.Fraisse}@xilinx.com

***Abstract*— The pinnacle of success for academic work is often achieved by having impact on commercial products. In order to have a successful transfer bridge, academic evaluation flows need to provide representative results of similar quality to commercial flows. A majority of publications in FPGA research use the same set of known academic CAD tools and benchmarks to evaluate new architecture and tool ideas. However, it is not clear whether the claims in academic publications based on these tools and benchmarks translate to real benefits in commercial products. In this work we compare the latest Xilinx commercial tools and products with these well-known academic tools to identify the gap in the major figures of merit. Our results show that there is a significant 2.2X gap in speed-performance for similar process technology. We have also identified the area-efficiency and runtime divide between commercial and academic tools to be 5% and 2.2X, respectively. We show that it is possible to improve portions of the academic flow such as ABC logic optimization to match the quality of commercial tools at the expense of additional runtime. Our results also show that depth reduction, which is often used as the main figure of merit for logic optimization papers does not translate to post-routing timing improvements. We finally discuss the differences between academic and commercial benchmark designs. We explain the main differences and trends that may influence the topic choice and conclusions of academic research. This work emphasizes how difficult it is to identify the relevant FPGA academic work that can provide meaningful benefits for commercial products.**

***Keywords—FPGA; Benchmark Designs; CAD Tool Flows; Academic vs Commercial; Vivado; Ultrascale; Verilog-To-Routing; ABC;***

## I. INTRODUCTION

Commercial Field-Programmable Gate Arrays (FPGAs) have been rapidly growing in both capacity and performance, opening the door to a large number of applications. Advances in process technology along with FPGA CAD tools and architecture have enabled this growth. Further advances in both tools and architecture are required to sustain this growth. Potentially, academic research efforts in these areas could contribute to this success by identifying promising tool or architecture ideas. This is especially important as FPGAs serve a wider range of applications compared to ASIC or ASSP counterparts in semiconductor business.

FPGA architecture and tool ideas that are seeded from FPGA academic community have decreased significantly over the last decade. The few that are proposed do not offer significant benefits when incorporated and evaluated in a commercial framework. If this trend continues, the academic work in this area might become irrelevant. This will adversely impact both FPGA industry and academic community, as the products can no longer leverage the broader academic ecosystem.

In this work, we claim one of the main reasons of this trend to be the significant performance gap between the academic and commercial framework. We try to examine this claim by comparing the most prevalent academic architecture tools with Xilinx Vivado used for UltraScale devices [1], [2]. When academic tools lag behind the state of the art by a large amount, it is easy to show improvement, but those improvements do not translate to any benefits for commercial tools and devices. After identifying the gap, we try to provide guidance on how to reduce it and also provide a few rules of thumb for assessing the merits of academic work.
The next section summarizes the related work. In section III, we introduce both academic and commercial flows and measure the gap for the main figures of merit. In Section IV a hybrid commercial and academic tool flow is presented. It shows that academic tools can perform on par with commercial tools at the expense of extra runtime. Section V examines another important factor in any FPGA assessment: benchmark designs. We highlight the trends in how benchmarks are changing and its impact on academic work. We provide a final discussion and concluding remarks in section VI.

## II. BACKGROUND AND RELATED WORK

The most popular academic open source tools used for FPGAs are Versatile Place and Route (VPR) [3] and ABC logic optimization and technology mapping [4]. There is also a front-end synthesis tool, called ODIN II [5], which takes a Verilog design and performs RTL elaboration. A recent academic framework, called Verilog-To-Routing (VTR), combines ODIN

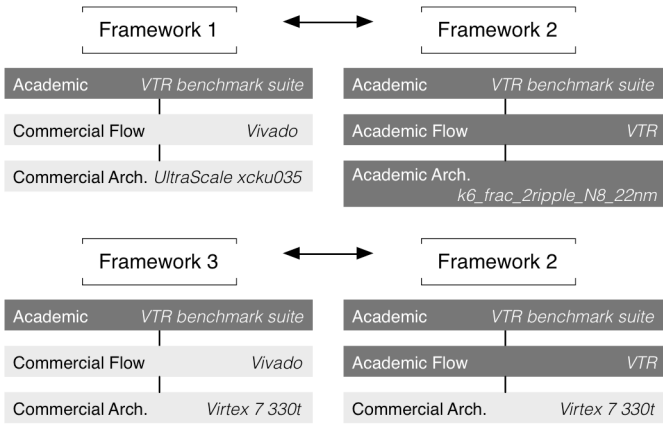



Fig. 1. The four frameworks used to compare commercial and academic flows

II, ABC and VPR to offer a complete unified flow for FPGA compilation [6]. We chose this well-known academic framework as our academic reference because it is the most flexible framework available. It gives researchers control over every part of the framework from architecture to tools and benchmarks designs. The front-end synthesis step produces a Berkeley Logic Interchange Format (BLIF) file, which is read by ABC to perform logic optimizations and technology mapping to LUTs. VPR then packs the LUTs and FFs into CLBs, places the CLBs and routes the whole design. There are three main components in an evaluation framework: the target FPGA architecture, the CAD tools and the benchmark designs.

There are two works that have also raised the issue of the gap between commercial and academic results. The first one provides the ability to compile designs for commercial devices using a VTR-to-Bitstream (VTB) flow presented in [7]. The VTB flow translates the mapped and placed circuits to an XDL description, a format provided by Xilinx ISE tools. These XDL text descriptions are translated to binary files and subsequently the design is routed with Xilinx ISE's PAR. VTR supports FPGA routing but is unable to model complex routing structures that exist in Xilinx FPGAs. New advances in the VTB project [8] enables routing designs on Xilinx' older architectures, such as the Virtex6. The routed designs are analysed by Xilinx ISE's TRCE static timing analyser to get reliable timing information. Our work is different in several aspects: we use the most recent commercial and academic tools (in contrast with a decade old commercial tools) and show that the divide is actually much wider now. In [7], the authors attribute the gap to the lack of support for carry logic, but we used a newer version of VTR that contains carry logic support and showed that it is not a key factor. The focus of VTB is to realize a design on a commercial device and they achieved that goal. However, XDL and the relevant flow are no longer supported by Xilinx and the proposed flow will unfortunately not work with the latest products, such as UltraScale.

A second work [9] focuses on addressing the mismatch in benchmark designs by providing larger designs for the open source community. They contributed 23 large benchmark designs and 20 mid-size designs. They identify the critical path delay gap as 50%, but they use a commercial tool for synthesis providing a hybrid evaluation flow and this gap is only measured on the benchmarks that did not fail. Their hybrid flow was only able to place and route 13 of the 23 large designs. They are also comparing to older 40nm products from Altera. This framework can easily be used to test advancements in place and route tools

In contrast to previous work, we focus on a new comparison to identify the gap for the most recent products and tools and show that it is much wider than stated in the literature. If the quality of academic tools is inadequate such that the required figures of merit are not met, there is little value in implementing those designs on commercial products using the VTB flow. We also address the area-efficiency and runtime scaling gaps. We then take a deeper dive in one of the academic tools, ABC logic optimization, to show that it is possible to achieve quality results on par with commercial tools with some effort.

## III. Commercial and Academic Tool Comparison

In order to make a fair comparison, we should use the same benchmark designs and the same target architecture. Unfortunately, VTR is not designed to compile for commercial architectures. First, we compare Vivado and VTR for the architectures available for the latest comparable process technology nodes. Later, we will use the VTB flow to target a commercial architecture and assess if the gap diminishes.

### *A. Evaluation frameworks*

We selected the smallest 20nm Ultrascale Kintex device (*xcku035*) with the largest package (*ffva1156*) and the fastest speed grade to match that of academic architectures. Together with the Vivado 2014.3 tool flow, we call this the commercial implementation.

The academic target device is the most advanced architecture closest to 20nm available in VTR. We choose *k6_frac_2ripple_N8_22nm*, because it performed best in terms of speed-performance of all the architectures available in VTR. We will call this architecture VTR-22nm from here on. The original architecture was sized for a 22nm high performance process and we needed to resize the transistor-level circuit for this architecture so that both commercial and academic devices are optimized for the same nominal operating voltage (0.95V). We used an automatic transistor sizing tool [9] and 22nm predictive technology models optimized for high performance [10]. It is worth noting that the process technology for xcku035 is a low power process technology, and hence our speed-performance results for the academic flow will be somewhat optimistic.

The VTR-22nm architecture contains carry chains, fracturable 36x36 multiplier blocks, and fracturable 32Kb memory blocks. Each CLB contains 8 fracturable LUTs similar to that of the xcku035, but contains only one flip-flop per LUT and no distributed memory capabilities. The routing architecture was kept simple with only length-four wires. We refer to the VTR-22nm architecture together with VTR tool flow revision 4591 as the academic implementation and use it as a reference in this section. As benchmark suite we used the 19 designs available in the VTR framework. The results for the most important figures of merit (area, maximum clock frequency

TABLE I. OVERVIEW OF THE POST-ROUTING RESULTS FOR THE VTR BENCHMARKS

| | **Academic (VTR - *k6_frac_2ripple_N8_22nm)*** | | | | | | **Commercial (Vivado – UltraScale)** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***Area*** | | | | ***Fmax*** | ***Runtime*** | ***Area*** | | | | | ***Fmax*** | | ***Runtime*** | |
| ***Benchmarks*** | ***CLB*** | ***Mult*** | ***Mem*** | ***Norm**** | ***(Mhz)*** | ***(min)*** | ***CLB*** | ***DSP*** | ***BRAM*** | ***Norm**** | ***rel*** | ***(Mhz)*** | ***rel*** | ***(min)*** | ***rel*** |
| bgm | 4259 | 11 | 0 | 4424 | 52 | 10.4 | 2150 | 22 | 0 | 2260 | 0,51 | 183 | 3,52 | 6.3 | 0.61 |
| blob_merge | 717 | 0 | 0 | 717 | 96 | 4.7 | 1437 | 0 | 0 | 1199 | 1,67 | 364 | 3.79 | 2.8 | 0.60 |
| boundtop | 280 | 0 | 1 | 300 | 146 | 4.5 | 813 | 0 | 1 | 836 | 2,79 | 367 | 2,52 | 2.8 | 0.63 |
| diffeq1 | 33 | 5 | 0 | 108 | 64 | 4.3 | 78 | 9 | 0 | 123 | 1,14 | 135 | 2,11 | 1.6 | 0.36 |
| diffeq2 | 21 | 5 | 0 | 96 | 81 | 4.3 | 34 | 9 | 0 | 79 | 0,82 | 149 | 1,84 | 1.5 | 0.35 |
| LU8PEEng | 2645 | 8 | 45 | 3665 | 16 | 8.5 | 2534 | 16 | 23 | 3143 | 0,86 | 24 | 1,51 | 5.2 | 0.61 |
| LU32PEEng | 8794 | 32 | 168 | 12634 | 16 | 25.1 | 8867 | 64 | 136 | 12315 | 0,97 | 23 | 1,46 | 9.2 | 0.37 |
| LU64PEEng | 17028 | 64 | 340 | 24788 | 16 | 59.4 | 15574 | 128 | 188 | 20538 | 0,83 | 26 | 1,63 | 18.5 | 0.31 |
| mcml | 8137 | 27 | 159 | 11722 | 27 | 32.1 | 6988 | 104 | 154 | 11050 | 0,94 | 55 | 2,01 | 13.2 | 0.41 |
| mkDelayW~ | 755 | 0 | 43 | 1615 | 117 | 5.2 | 140 | 0 | 27 | 761 | 0,47 | 645 | 5,51 | 1.9 | 0.37 |
| mkSMAda~ | 210 | 0 | 5 | 310 | 158 | 4.6 | 193 | 0 | 3 | 262 | 0,85 | 491 | 3,11 | 2.3 | 0.50 |
| or1200 | 308 | 1 | 2 | 363 | 102 | 4.7 | 365 | 4 | 1 | 408 | 1,12 | 176 | 1,73 | 1.9 | 0.40 |
| raygentop | 266 | 7 | 1 | 391 | 148 | 4.6 | 390 | 9 | 0,5 | 446,5 | 1,14 | 469 | 3,18 | 1.9 | 0.41 |
| sha | 244 | 0 | 0 | 244 | 179 | 4.6 | 212 | 0 | 0 | 212 | 0,87 | 299 | 1,68 | 2.2 | 0.47 |
| stereovision0 | 1195 | 0 | 0 | 1195 | 245 | 4.9 | 1013 | 0 | 0 | 1013 | 0,85 | 635 | 2,59 | 2.6 | 0.52 |
| stereovision1 | 1916 | 46 | 0 | 2606 | 149 | 5.6 | 2511 | 0 | 0 | 2511 | 0,96 | 337 | 2,27 | 4.1 | 0.73 |
| stereovision2 | 3290 | 201 | 0 | 6305 | 100 | 7.9 | 2213 | 270 | 0 | 3563 | 0,57 | 136 | 1,36 | 4.6 | 0.58 |
| stereovision3 | 22 | 0 | 0 | 22 | 270 | 4.4 | 30 | 0 | 0 | 30 | 1,36 | 474 | 1,76 | 1.3 | 0.29 |
| **Geometric mean** | | | | | | | | | | | **0.95** | | **2.24** | | **0.46** |
| Geometric Standard Deviation | | | | | | | | | | | 1.52 | | 1.45 | | 1.30 |
| **Total** | | | | | | **199.8** | | | | | | | | **83.9** | |

* Normalized Area: the total area expressed in terms of CLB tiles, it includes the DSP/Mult and Mem/BRAM usage, see equation (1) and (2)

and compilation runtime) are listed in TABLE I and will be discussed in the following subsections.

### B. Speed-performance

Vivado is designed to compile a design for a set of known constraints and not to find the highest possible operating frequency for a given design. To find the maximum clock frequency we started with constraining the designs with a clock period that could be easily met. Subsequently the data path delay of the most critical path in the clock domain was used as a new constraint for the clock period. We repeated this process until Vivado just failed the constraint with a violation of less than 1ns. Another approach could be to constrain the design with an unrealistic clock period like in VTR, for example 1ns, but Vivado would recognize that it could never meet the constraint and it would exit early. Therefore the latter approach is not an option for the commercial flow.

As noted in TABLE I, the maximum clock frequency for all benchmark designs is higher for the commercial implementations compared to the academic implementations. The geomean of the maximum clock frequencies of the commercial implementations is 2.24 times higher than that of the academic implementations. We believe this 2.24X divide in quality of results is an important conclusion from this work. It indicates why many academic FPGA architecture and tool improvements cannot translate to realistic benefits for FPGA industry. This wide gap includes architecture and tool differences, but excludes differences caused by benchmark designs and process technology.

Referring to previous work [9], we may also estimate that 50% of this divide is due to synthesis and the rest is from the place and route portion of the flow and the architecture difference. We will discuss this further in the following sections.

### C. Area-efficiency

Comparing the area-efficiency between commercial and academic implementations is more difficult because of the different hard blocks in the target architectures. The VTR-22nm architecture contains fracturable 36x36 multiplier blocks and fracturable 32 Kb memory blocks. The Ultrascale fabric has versatile DSP48E2 blocks that can implement 27x18 multiplications, 48-bit addition/subtraction, XOR, and some additional functionality. It also contains fracturable 36Kb block RAMs.

Comparing the multiplier logic consumption for the academic and commercial implementations, we find the commercial DSP usage to be about twice the amount of academic multiplier block usage. This corresponds with the size of the respective multipliers. The only exception is *stereovision1*. The default behaviour of Vivado is to implement the divisions in *stereovision1* without DSP blocks in contrast with VTR. This leads, however, to an increased LUT count for Vivado, but a faster circuit. Vivado chooses the most delay-optimal implementation if there are enough resources available.

Academic implementations typically use more memory blocks than the commercial ones. The Ultrascale fabric has slightly larger memory blocks, but that is not the main reason. The VTR benchmark designs contain a lot of shallow memories and Vivado implements these shallow memories with distributed memory. To overcome the issue of comparing resource usage for different types of hard blocks, we define a normalized area measure. The measure is expressed in terms of CLB tiles and encompasses the CLB count and all hard block occurrences:

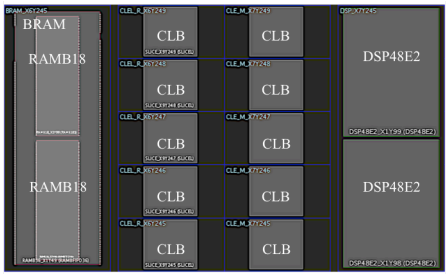



Fig. 3. RAMB and DSP height vs CLB height, image taken from Vivado Design Editor.

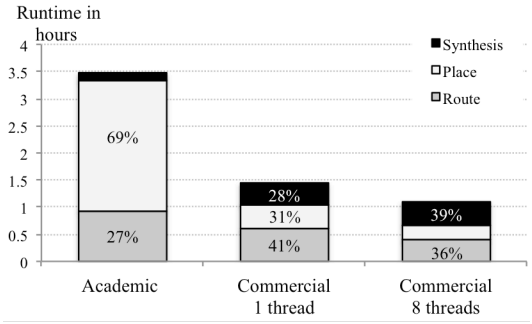



Fig. 2. Total Runtime for compiling the VTR benchmark designs for a single run

$$Area_{norm,\ academic} = n_{CLB} + k_{mult}.n_{mult} + k_{mem}.n_{mem} \quad (1)$$

$$Area_{norm,\ commercial} = n_{CLB} + k_{DSP}.n_{DSP} + k_{BRAM}.n_{BRAM} \quad (2)$$

For the academic area constants $k_{mult}$ and $k_{mem}$, we use the minimum transistor width count as reported in the architecture description for each type of hard block. We compare it to the area used for one CLB tile in the newly sized academic architecture [9]. Taking the interconnect area into consideration, we set the constants to $k_{mult} = 15$ and $k_{mem} = 20$. For the commercial area constants $k_{DSP}$ and $k_{BRAM}$ a similar approach to the academic calculations is used, but we scale the block areas based on multiplier bits and memory bits. Each DSP and memory block pair has a height of 5 CLBs, as seen in Fig. 2 from Vivado design editor. This results in the following area constants, $k_{DSP} = 5$ and $k_{BRAM} = 23$. *The* hard block occurrences and normalized Area is reported for each design in Table I. The commercial implementations use on average 96% of the normalized area used by academic implementations. This gap is significant, but not a showstopper in contrast with the other figures of merit we investigated.

### D. Runtime

The benchmark designs were compiled on a workstation with a 3.4 GHz quad-core Intel Core i7-3770 processor and 32 GB memory. In TABLE I, the runtime for each benchmark design is reported for the commercial and academic flow. VTR can only operate in a single-threaded mode, so to be fair we compare it to Vivado restricted to only run a single thread. Even in single-threaded mode Vivado is on average 2.2X faster than VTR. This runtime gap is consistent with a geometric standard deviation of 1.3. All benchmark designs compile faster when using Vivado with runtimes ranging from 73% to 35% of the runtime of VTR.

Fig. 3 shows the total runtime to compile all VTR benchmark designs for both the commercial and academic flow and a breakdown for each major step in the compilation. As a reference we also included a run in which we let Vivado run unrestricted. On our test machine this mode used 8-threads, which decreased the total runtime from 1.45h to 1.1h. This is not a huge decrease mainly because we only deal with smaller benchmarks for which the runtime is so small that Vivado cannot fully take advantage of multithreading. Overall we see the same picture for the total runtime as for the separate benchmark designs. The total runtime for Vivado is a little under one hour and half and a little over 3 hours for VTR. So Vivado compiles the benchmarks in less than half the time VTR does. In Vivado the runtime is more equally divided between the three major steps, synthesis, placement, and routing than in VTR. In VTR, synthesis takes only 4% of the total runtime. The placer in VTR is responsible for biggest chunk of runtime (69%) followed by the routing step (27%).

We also considered the runtime scalability with respect to the size of the benchmark designs. We choose the LUT count as area figure and we only selected the designs with more than 20K LUTs to minimize the impact of the nonrecurring runtime cost. The runtime for smaller designs is often dominated by fixed portions, such as reading or writing the files. The fixed portions don't have a significant impact on scalability. The data points for each flow are plotted in Fig. 4 and fitted using a power regression model. The runtime scaling gap widens proportional to the equation $0.14 \cdot x^{0.24}$, where x represents the number of LUTs, so for each 60K LUT increase the runtime gap doubles. We predict that the runtime gap will increase in the same fashion beyond the 160K mark. This makes the compile time of academic flows impractical for today's FPGA application sizes, because they easily surpass the 160K LUT mark. The small number of designs shown in Fig. 4 might not be sufficient for calculating a statistically valid scaling factor. However, the trend of growing runtime gap, which is the main message of this section, will hold as we add larger and more designs to the graph.

Out of this comparison, placement is clearly a main cause of the gap in runtime scalability. At its core, the VTR placer still uses simulated annealing which is more runtime intensive and does not scale as well as the analytical placement techniques used in the Vivado placer. The academic framework could benefit greatly from an open-source analytical placer, which is not available to our knowledge at the moment of writing.

### E. Using VTR for a Commercial Target Device

VTR is not designed to map to commercial devices but we made an attempt to use the VTB flow introduced in the previous work. In [7], the authors present a VTR-to-Bitstream (VTB) flow that enables users to map to Virtex 6 devices. Vivado does not support the older Virtex 6 devices. We extended VTB to target the Virtex 7 vx330t device with the help of the authors of [7]. We compared the frameworks 3 and 4 as shown in Fig. 1. VTB and Vivado target the same commercial device. We choose to target the fastest speed grade

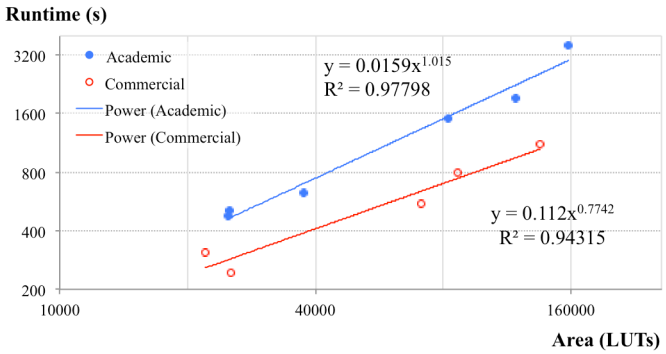



Fig. 5. Runtime scalability of the Academica and Commercial CAD tools with respect to the size of the design.

and the largest footprint. This resulted in Vivado implementations that consume 25% less area and are able to operate at 2.1X higher operating frequencies on average than VTB implementations. This is more or less in line with the gap reported for the commercial versus academic comparison. The only major difference was that the VTB was remarkably slower than Vivado by a factor of 5.5X.

There is only a very slight reduction in speed-performance gap if we compare Vivado versus VTB targeting a commercial device (2.1X) and the commercial versus fully academic comparison (2X). We attribute this reduction in the gap to a better architecture, but conclusions are difficult here because VTB is not designed to fully exploit this commercial architecture, so the actual architecture gap could be much wider. Our initial intention for using VTB was to identify which part of the gap can be attributed to the architecture and which part to the tools. We believe the large quality gap in the tools may be misleading and hence we defer making solid conclusions on the architecture gap to future work, after more detailed investigations.

## IV. Hybrid Commercial and Academic Evaluation Flow

We described the gap between academic and commercial tools for FPGA design implementation in section III. However, the main advantage of open-source academic tools is that they are easier to change and augment toward a research goal. The tools are often data-driven and skip unnecessary detail, helping the researcher conclude faster. The question we are trying to answer in this section is how to combine the credibility of commercial tools with the flexibility of academic tools to reach pragmatic architectural or tool conclusions. In contrast to previous section, we use commercial tools as our baseline for assessing a new tool flow.

We created a hybrid evaluation flow using Vivado and ABC [1], which is a well-known academic tool for logic optimization and technology mapping. The advantages of such a hybrid flow are two-fold: 1) we can accurately quantify the quality of logic optimization; 2) we can quickly evaluate architecture ideas or opportunities in commercial tool optimization. Even if such

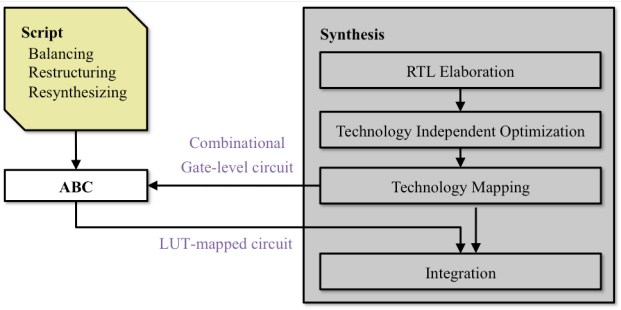



Fig. 4. Hybrid flow for logic optimization

evaluation flow helps us detect failures for certain ideas, it will prevent researchers from investing unnecessary additional time.

Fig. 5 summarizes the hybrid flow that we created. The key to creating such a flow is identifying the best interception points to exit and re-enter the commercial Vivado flow. Vivado synthesis tool processes the design in three steps: elaboration, architecture-independent optimizations, and technology mapping. During RTL elaboration, common data path operations such as additions and storage elements such as memory blocks are identified and inferred. Architecture-independent optimizations include constant propagation, operation sharing, strength reduction; expression optimization, finite state machine encoding/minimization, generic restructuring and don't-care optimizations. During technology mapping the optimized design is mapped onto target architecture structures, such as DSP blocks, adders with dedicated carry-chains, BRAMs, LUTs and FFs.

In the new hybrid flow, the combinational portion of the logic gate network is cut out and written to a BLIF file. ABC reads in the BLIF file and performs logic optimizations and mapping as stated in the script given by the user. The script may contain commands to restructure and balance the logic network and to perform different mapping algorithms. After ABC optimizations, the circuit mapped to LUTs is stitched back into the design in Vivado. This new flow replaces the commercial technology mapping and optimization with that of the academic flow.

Initially, this new hybrid flow with ABC performed worse than the baseline Vivado flow in all figures of merits: performance, area and runtime. However, the differences were all within 2% except for runtime. This indicates that the significant performance gap we noted earlier is not due to logic optimization portion of the flow. After a number of iterations and modifications to both ABC and the script, we managed to show some improvements compared to baseline Vivado. According to our results 2.5% increase in maximum clock frequency along with 1.8% decrease in area was achieved on average for more than 80 commercial benchmark designs, as summarized in TABLE II. These modest average improvements compared to Vivado were achieved at the expense of additional runtime in ABC.

TABLE II. SUMMARY FOR THE HYBRID FLOW VS VIVADO

| | Early depth | Fmax | Area (CLBs) |
|---|---|---|---|
| Whole suite | -16 % | + 2.5 % | - 1.8 % |
| High depth | -24 % | + 5 % | - 3 % |
| Low Depth | -13.5 % | + 0.4 % | - 1 % |
| Arithmetic | -7 % | + 1.1 % | - 1 % |

The percentages indicate relative improvement for the geomean of the ratios

Fig. 6 depicts the maximum clock frequency ratio for each design in the commercial suite, providing a more detailed view. Fmax improved for roughly 70% of the designs and up to 20% in the best cases. The main reason for improving the quality was less emphasis on early depth reduction. Initial scripts aggressively reduced the depth of the deepest paths in the designs, which led to worse post routing results. This is expected because at the time of technology mapping there is too much uncertainty to predict the real critical path after routing. The critical path could be dominated by routing and aggressive depth reduction will adversely affect the final results. The ABC script that produced the best results is available in [17]. It contains multiple LUT mapping iterations interleaved with sum –of-product balancing.

The hybrid flow helped us find the right balance between the area and depth reduction by focusing on average depth reduction and observing post-routing results from the commercial tool. It is worthwhile to note that even the initial results from the hybrid flow (before our optimizations) were within a few percentage of the baseline Vivado flow. This indicates that the synthesis gap observed in previous work is not due to logic optimization portion of the academic flows.

We can make two high level observations using our hybrid flow. First, the fact that we could reach the quality of commercial tools and even improve the results for some designs shows the potential for academic tools if used in a correct framework. The second high level conclusion from this exercise was that depth reduction does not translate to post routing improvement directly. A good rule of thumb to estimate post-routing benefits of the academic work that claim improvements by reporting depth reduction is to divide the gain by an order of magnitude. We further investigated this by focusing on depth-oriented designs and confirmed that ABC indeed improves the results by 5% on average on these designs. This is a significant improvement even for commercial products and we will elaborate on this classification more in the next section.

We also used this evaluation flow to dismiss some of the published architecture ideas and tool optimizations quickly without additional expensive investments in changing the commercial flow. For example, previous work has suggested using cascaded LUTs [18-19] as potential FPGA architecture improvements, because they improved area and the depth of the circuit. Since these ideas are often implemented in ABC framework we used our hybrid flow to evaluate some of them. We found that conclusions that are mainly based on early depth reduction will not hold after routing the designs. Another example is the and-inverter cones [20]. In this case the authors further investigated their claims in a second publication [21]

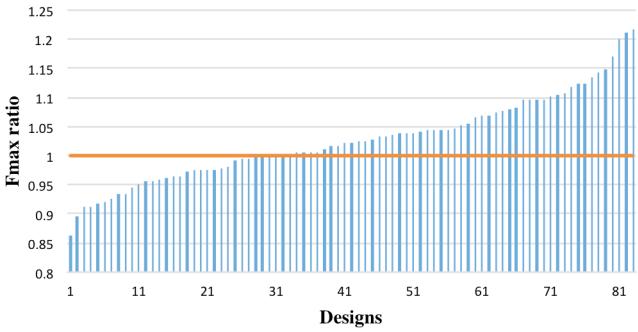



Fig. 6. Maximum clock frequency ratio for the hybrid flow versus the Vivado baseline

and came to the conclusion that the observed benefits after technology mapping did not translate to post-routing improvements.

## V. BENCHMARK DESIGN SUITE

An important aspect of any evaluation framework is the benchmark designs. In this section we focus on highlighting the differences between academic and commercial benchmark suites.

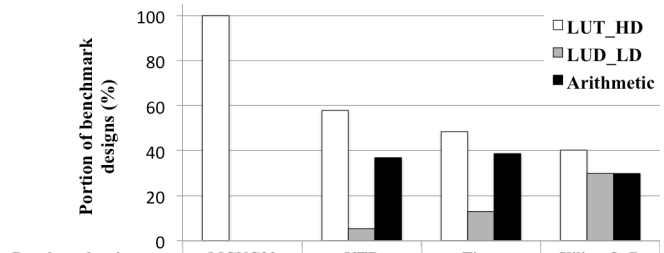
### *A. Academic benchmark designs*

Unfortunately it is hard to separate benchmark designs from the framework they were written for, so we inspect the benchmark designs with their framework in mind. Typically used in academia are the well-known evaluation frameworks such as the VPR framework [3], the VTR framework [6] and the recent Titan framework [9].

The *Versatile Place and Route (VPR) framework* consists of 20 large benchmark designs synthesized by the Microelectronics Centre of North Carolina (MCNC). VPR is used as place and route tool and a homogeneous LUT-only architecture at 48nm technology node as a target architecture. The MCNC benchmarks are still quite popular [12-15] and the VPR framework is still maintained as part of the VTR framework.

The *Verilog-To-Routing framework (VTR)* [2] includes several benchmark designs described in Verilog. There is a range of architectures that can be targeted in VTR and researchers can add or tweak their own architecture. We used the most advanced architecture available, called *k6_frac_2ripple_N8_22nm*. Researchers working on applications or tools will probably not change the default architectures provided in the VTR framework.

The *Titan framework* [4] consists of 23 large benchmarks and 20 mid-sized benchmarks. They are synthesized with Quartus II and VPR is used for backend of the flow to map to an architecture closely matching the Altera Stratix-IV architecture [15]. They used identical hard blocks, but the routing architecture was only modelled approximately. Unfortunately VPR does not succeed at routing 13 of the 23 large designs because of memory requirements or routing congestion.




| Benchmark suite | MCNC20 | VTR | Titan | Xilinx QoR |
|---|---|---|---|---|
| CAD tool flow | VPR | VTR | QuartusII & VPR | Vivado |
| Target Architecture | Homogeneous 48nm | VTR-22nm | Stratix IV* | Ultrascale |

*hybrid architecture

Fig. 7. Benchmark Suite Profiles. For each framework, the benchmark designs are classified in three categories depending on the paths and the type of instances in the critical zone of the circuit, LUT dominant & High Depth (LUT_HD), LUT dominant & Low Depth (LUT_LD) and Arithmetic

### *B. Comparing with commercial benchmark designs*

We profiled more than 80 industry benchmark designs in order to understand the differences with academic designs. The academic designs are much smaller compared to the industry benchmark designs we used, which typically have more than 100k LUTs. The other noticeable difference is that the majority of VTR benchmark designs are I/O-bound. They also have fewer memory and DSP components compared to industrial designs. All these differences may contribute to misleading academic conclusions in the academic frameworks. Some of these differences such as size of the benchmark designs are already highlighted in previous work [9].

In this work, we highlight and analyse another subtle, yet important difference that may skew the academic conclusions. Fig. 7 depicts how depth profile differs for various academic and industrial designs compiled within their respective framework. For each framework, the benchmark designs are classified in three categories depending on the paths and the type of instances in the critical zone of the routed circuit. For the commercial framework, each benchmark category contains around the same number of designs. For the VTR and Titan benchmark designs, the category of designs with a LUT dominant and shallow critical zone is under represented. Lastly, the MCNC20 benchmark suite contains only LUT dominant designs with deep critical zones. It is clear from this comparison that academic benchmark suites contain relatively much more designs with a high depth critical zone than industrial benchmark suites.

A large number of academic publications, especially those that use ABC, make conclusions based on depth improvement after technology mapping. Therefore, it is important to understand how depth reduction correlates to the end performance improvement after routing. In the next subsection we dig deeper into this depth profiling to understand the trends.

### *C. Depth classification of designs: discussion and trends*

Our depth classification is based on the profile of the critical zone in the commercial benchmark designs. We define the critical zone as all the paths in the design with 5% worst slack. Taking into account the type of instances in the critical zone, we observe that 68% of the designs' critical zone is dominated by LUT instances. The most occurring instance type is CARRY blocks for 20% of the designs. The DSP blocks dominate the critical zone for the 12% remaining designs. The average logic depth of the DSP dominated designs is typically lower than the carry dominated designs. We group both CARRY and DSP dominated designs in the same class, the arithmetic designs, because they show similar behaviour regarding our analysis.

The remaining designs with critical path dominated by LUTs are further divided into 2 groups. We take into account the average depth of the paths in the critical zone for the LUT dominant circuits. 29% of the designs have an average logic depth smaller than or equal to 2. This class contains heavily pipelined designs with critical zone dominated by routing and net delays. We also refer to this group of applications as low depth. The other group contains benchmark designs with an average logic depth higher than 2.

We now revisit the results of our hybrid flow explained in section IV in the context of this logic depth classification. The results are summarized in TABLE II. The hybrid flow augmented with ABC has an average 5% higher maximum clock frequency and uses on average 3% less CLBs for the high depth, LUT-dominant designs. LUT-dominant, low depth designs show no significant improvement in performance, but they show a 1% area reduction. For the arithmetic dominant circuits the new flow produces solutions with 1.1% higher clock frequency and 1% lower CLB usage.

Our results show that ABC advantages for performance are mostly applicable to a third of the designs which have critical paths with a lot of logic levels. This is in line with the academic literature where most of ABC work is focused on depth reductions. However, the FPGA application trends are in the direction of highly pipelined designs with lower depth. Therefore, these advantages will be less pronounced in the future. The representativeness of academic benchmark suites could be improved by adding low-depth designs. Another important observation is that depth reduction no longer translates to significant post-routing delay improvement in commercial frameworks.

## VI. CONCLUDING REMARKS

We examined the divide between the quality of the FPGA configurations produced by the commercial and academic frameworks to show that it has grown beyond acceptance. For example, the speed-performance quality gap is more than 2.2X. This makes it hard to assess the merits of academic results, because it is much easier to improve something that is so far from optimal. On the other hand, we showed that it is still possible to use academic tools in a credible framework that is a hybrid with a commercial framework. Our results showed that close to 5% improvement is possible on average for designs with high depth paths in the critical zone. This work also highlighted a trend in industrial applications towards low depth, highly pipelined designs. Designs with shallow LUT dominated critical zones are under-represented in the academic frameworks. This further emphasizes the need for updating benchmark designs and suggests that academic tools

need to focus on other optimizations such as retiming instead of early depth reduction.

Academic contributions in the area of FPGA architecture and tools are still possible, but only if the wide divide highlighted in this work is addressed or academic work is done in the context of intercepting commercial framework at the right access points in the flow. Such effort requires joint cooperation and involvement of academic and commercial interested parties. Commercial parties are often questioning the return of investment on such efforts due to significant gap. On the other hand, some academics dismiss the importance of quality gap as the responsibility of the industry. This is leading to a tentative stale-mate and the solution requires contribution from both parties. Industry needs to provide easier interface at appropriate interception points for their tools. Academics need to build hybrid flows that use commercial framework with the exception of the portion under investigation.

Other academic FPGA work in the areas of applications or where commercial tools are evolving such as high-level synthesis is still relevant if quality of results is properly maintained. This may also imply combining them correctly with the relevant commercial framework and collaboration between industry and academic ecosystem. We also encourage academic researchers to use commercial tool flows and architectures as a baseline when possible. The evaluation framework we used is available online at [17]. It includes the VTR benchmarks partly rewritten to enable compilation with Vivado and a collection of scripts to derive the statistics used in this paper.

## ACKNOWLEDGMENT

We want to thank Eddie Hung for the support with using the VTR-to-bitstream extension. The opinions expressed by authors are theirs alone and do not represent the opinions of Xilinx and are not indications of any future policy on FPGA software or hardware held by Xilinx.

## REFERENCES

[1] Xilinx Inc. (2015, April) “UltraScale Architecture and Product Overview” [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview, April 27, 2015

[2] Xilinx Inc. (2015, April) “Vivado Design Suite User Guide, Xilinx Inc” [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_1/ug910-vivado-getting-started.pdf,

[3] J. Luu,, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, K. Kent and J. Rose, “VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol *4 (*4), 2011.

[4] Berkeley Logic Synthesis and Verification Group, *ABC: A System for Sequential Synthesis and Verification*, Berkeley, CA: 2014. http://www.eecs.berkeley.edu/~alanmi/abc/

[5] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, “ODIN II-an open-source verilog HDL synthesis tool for CAD research,” in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on,* 2010, pp. 149-156.

[6] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose and V. Betz "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," ACM TRETS, Vol. 7, No. 2, June 2014, pp. 6:1 - 6:30.

[7] E. Hung, E. Fatemeh, and S. Wilton, "Escaping the academic sandbox: Realizing VPR circuits on Xilinx devices," in *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*., 2013.

[8] E. Hung, "Mind The (Synthesis) Gap: Examining Where Academic FPGA Tools Lag Behind Industry," in *25th International Conference on Field Programmable Logic and Applications (FPL),* 2015.

[9] K. E. Murray, S. Whitty, S. Liu, J. Luu and V. Betz, "Timing Driven Titan: Enabling Large Benchmarks and Exploring the Gap Between Academic and Commercial CAD", *ACM Trans. Reconfig. Technol. Syst., , vol. 8 no. 2, pp.10:1-10:18, 2015*

[10] C. Chiasson and V. Betz. "COFFE: Fully-Automated Transistor Sizing for FPGAs", in *IEEE International Conference on Field-Programmable Technology (FPT),* Kyoto, 2013, pp. 34-41.

[11] W. Zhao and Y. Cao, "New generation of Predictive Technology Model for sub-45nm early design exploration," in *IEEE Transactions on Electron Devices*, vol. 53, no. 11, November 2006, pp. 2816-2823.

[12] A. Petkovska, D. Novo, A. Mishchenko and P. Ienne, “Constrained interpolation for guided logic synthesis,” in *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on, 2014,* pp. 462-469.

[13] DeHon, André, and Nikil Mehta. "Exploiting partially defective LUTs: Why you don't need perfect fabrication," in *IEEE International Conference on Field-Programmable Technology (FPT),* Kyoto, 2013, pp. 12-19.

[14] P. E. Gaillardon, X. Tang, G. Kim and G. De Micheli (2015). “A Novel FPGA Architecture Based on Ultrafine Grain Reconfigurable Logic Cells.” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 2015, to appear, available online (early access).

[15] S. U. Rehman, A. Blanchardon, A. Ben Dhia, M. Benabdenbi, R. Chotin-Avot, L. Naviner, L. Anghel, H. Mehrez, E. Amouri and Z. Marrakchi, "Impact of Cluster Size on Routability, Testability and Robustness of a Cluster in a Mesh FPGA," in *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, Tampa, FL, 2014, pp. 553 – 558.

[16] Altera Corporation. (2008, November) “Stratix IV Device Handbook” [Online]. Available: https://www.altera.com/en_US/pdfs/literature/hb/stratix-iv/stratix4_handbook.pdf

[17] E.Vansteenkiste, An Evaluation Framework for an Acadamic and Commercial comparison. 2015. https://github.com/EliasVansteenkiste/EvaluationFramework

[18] S. Ray, A. Mishchenko, N. Een, R. Brayton, S. Jang, and C. Chen, “Mapping into LUT structures,” in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2012, pp. 1579-1584.

[19] A. Mishchenko, "Enumeration of irredundant circuit structures," in *Proceedings of International Workshop on Logic and Synthesis,* San Francisco, CA, 2014.

[20] H. Parandeh-Afshar, H. Benbihi, D. Novo, and P. Ienne, “Rethinking FPGAs: Elude the Flexibility Excess of LUTs with And-Inverter Cones,” in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, 2012. ACM, New York, NY, USA, pp. 119-128.

[21] G. Zgheib, L. Yang, Z. Huang, D. Novo, H. Parandeh-Afshar, H. Yang and P. Ienne (2014, February). “Revisiting and-inverter cones,” in *Proceedings of the ACM/SIGDA international symposium on Field-programmable gate arrays, 2014. ACM, New York, NY, USA,* pp. 45-54