Timing-Driven Titan: Enabling Large Benchmarks and Exploring the Gap between Academic and Commercial CAD

KEVIN E. MURRAY, SCOTT WHITTY, SUYA LIU, JASON LUU, and VAUGHN BETZ,
University of Toronto

Benchmarks play a key role in Field-Programmable Gate Array (FPGA) architecture and CAD research, enabling the quantitative comparison of tools and architectures. It is important that these benchmarks reflect modern large-scale systems that make use of heterogeneous resources; however, most current FPGA benchmarks are both small and simple. In this artile, we present Titan, a hybrid CAD flow that addresses these issues. The flow uses Altera's Quartus II FPGA CAD software to perform HDL synthesis and a conversion tool to translate the result into the academic Berkeley Logic Interchange Format (BLIF). Using this flow, we created the Titan23 benchmark set, which consists of 23 large (90K–1.8M block) benchmark circuits covering a wide range of application domains. Using the Titan23 benchmarks and an enhanced model of Altera's Stratix IV architecture, including a detailed timing model, we compare the performance and quality of VPR and Quartus II targeting the same architecture. We found that VPR is at least 2.8× slower, uses 6.2× more memory, 2.2× more wire, and produces critical paths 1.5× slower compared to Quartus II. Finally, we identified that VPR's focus on achieving a dense packing and an inability to take apart clusters is responsible for a large portion of the wire length and critical path delay gap.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays*; B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer-Aided Engineering**]: Computer-Aided Design (CAD)

General Terms: Performance, Measurement

Additional Key Words and Phrases: CAD, Benchmarks, FPGA

ACM Reference Format:
Kevin E. Murray, Scott Whitty, Suya Liu, Jason Luu, and Vaughn Betz. 2015. Timing driven titan: Enabling large benchmarks and exploring the gap between academic and commercial CAD. ACM Trans. Reconfig. Technol. Syst. 8, 2, Article 10 (March 2015), 18 pages.
DOI: http://dx.doi.org/10.1145/2629579

1. INTRODUCTION

Open-source CAD flows, such as the VTR project [Rose et al. 2012], are crucial to Field-Programmable Gate Array (FPGA) research because open-source tools allow the FPGA architecture and CAD algorithms to be easily modified. To obtain accurate CAD or architecture results, however, we need more than an open-source CAD flow. It is essential that the benchmark designs used to exercise a new algorithm or architecture represent

This work was supported by NSERC, Altera, Texas Instruments, the SRC and by a QEII-GSST scholarship. Computations were performed on the GPC supercomputer at the SciNet HPC Consortium [Loken et al. 2010].
Authors' addresses: K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz, Department of Electrical and Computer Engineering, University of Toronto, Ontario, Canada, M5S 3G4; emails: {kmurray, jluu, vaughn}@eecg.utoronto.ca, {scott.whitty, suya.liu}@mail.utoronto.ca.

the current—and, ideally, the future—usage of FPGAs. Unfortunately, the most commonly used FPGA benchmark suites are currently composed of designs that are much smaller and simpler than current industrial designs. The MCNC20 benchmark suite [Yang 1991], for example, has an average size of only 2,960 primitives, whereas current commercial FPGAs [Altera Corporation 2012b; Xilinx Incorporated 2012] contain up to *2 million* logic primitives alone. Furthermore, half of the MCNC benchmarks are purely combinational, and none of the designs contain hard primitives such as memories or multipliers. The more modern VTR benchmark suite [Rose et al. 2012] is an improvement, but it still consists of designs with an average size of only 23,400 primitives, which would fill only 1% of the largest FPGAs. Only 10 of the 19 VTR designs contain any memory blocks, and at most 10 memories are used in any design. In comparison, Stratix V and Virtex 7 devices contain up to 2,660 and 3,760 memory blocks, respectively. Without larger benchmarks, key issues such as CAD tool scalability for very large designs cannot be investigated, and without more up-to-date benchmarks, the validity of architecture studies is questionable.

There are many barriers to the use of state-of-the-art benchmark circuits with open-source tool flows. First, obtaining large benchmarks can be difficult because many are proprietary. Second, purely open-source flows have limited Hardware Description Language (HDL) coverage. The VTR flow, for example, uses the ODIN-II Verilog parser, which can process only a subset of the Verilog HDL; any design containing System Verilog, VHDL, or a range of unsupported Verilog constructs cannot be used without a substantial rewrite. As well, if part of a design was created with a higher level synthesis tool, the output HDL is not only likely to contain constructs unsupported by ODIN-II, but is also likely to be very hard to read and rewrite using only supported constructs. Third, modern designs make extensive use of IP cores, ranging from low-level functions such as floating-point multiply and accumulate units to higher level functions like FFT cores and off-chip memory controllers. Since current open-source flows lack IP, all these functions must be removed or rewritten; this is not only a large effort, but it also raises the question of whether the modified benchmark still accurately represents the original design because IP cores are often a large portion of the design.

To avoid many of these pitfalls, we have created Titan, a hybrid flow that utilizes a commercial tool, Altera's Quartus II design software, for HDL elaboration and synthesis, followed by a format conversion tool to translate the results into a form that open-source tools can process. The Titan flow has excellent language coverage and can use any unencrypted IP that works in Altera's commercial CAD flow, making it much easier to handle large and complex benchmarks. We output the design early in the Quartus II flow, which means we can change the target FPGA architecture and use open-source synthesis, placement, and routing engines to complete the design implementation. Consequently we believe that we have achieved a good balance between enabling realistic designs while still permitting a high degree of CAD and architecture experimentation.

An earlier version of this work was published as Murray et al. [2013b]. We have significantly enhanced and extended it by improving the quality of the Stratix IV architecture capture by including support for carry chains and direct links between adjacent blocks, improving DSP packing, and adding a detailed timing model. This enables timing-driven CAD and architecture research and a detailed comparison of commercial and academic CAD tools. Our contributions include:

—Titan, a hybrid CAD flow that enables the use of larger and more complex benchmarks with academic CAD tools.
—The Titan23 benchmark suite. This suite of 23 designs has an average size of 421,000 primitives. Most designs are highly heterogeneous with thousands of RAM and/or multiplier primitives.

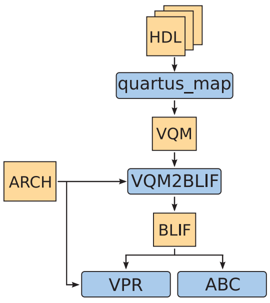

Fig. 1. The Titan flow.

—A timing-driven comparison of the quality and runtime of the academic VPR and the commercial Quartus II packing, placement, and routing engines. This comparison helps identify how academic tool quality compares to commercial tools and highlights several areas for potential improvement in VPR.

2. THE TITAN FLOW

The basic steps of the Titan flow are shown in Figure 1. Quartus II performs elaboration and synthesis (`quartus_map`), which generates a Verilog Quartus Map (VQM) file. The VQM file is a technology-mapped netlist consisting of the basic primitives in the target architecture. The VQM file is then converted to the standard Berkeley Logic Interchange Format (BLIF) using our VQM2BLIF tool, which can then be passed on to conventional open-source tools such as ABC [Mishchenko 2013] and VPR [Betz and Rose 1997]. The Titan flow is described in more detail in Murray et al. [2013b] and Murray et al. [2013a].

The VQM2BLIF tool, detailed documentation, and scripts to run the Titan flow, along with the complete benchmark set and enhanced architecture capture, are available from: http://www.eecg.toronto.edu/∼vaughn/software.html.

3. FLOW COMPARISON

Using a commercial tool like Quartus II as a "front-end" brings several advantages that are hard to replicate in open-source flows. It supports several HDLs including Verilog, VHDL, and SystemVerilog, and it also supports higher level synthesis tools like Altera's QSYS, SOPC Builder, DSP Builder, and OpenCL compiler. It also brings support for Altera's IP catalogue, with the exception of some encrypted IP blocks.

These factors significantly ease the process of creating large benchmark circuits for open-source CAD tools. For example, converting an LU factorization benchmark [Zhang et al. 2012] for use in the VTR flow [Rose et al. 2012] involved roughly one month of work removing vendor IP and recoding the floating point units to account for limited Verilog language support. Using the Titan flow, this task was completed within a day because it only required the removal of one encrypted IP block from the original HDL, which accounted for less than 1% of the design. In addition, since more than 68% of the design logic was in the floating point units, the Titan flow better preserves the original design characteristics.

A concern in using a commercial tool to perform elaboration and synthesis is that the results may be too device- or vendor-specific to allow architecture experimentation. However, this is not necessarily the case. The Titan flow still allows a wide range of experiments to be conducted, as shown in Table I. The ability to use tools like ABC to resynthesize the netlist ensures experiments with different LUT sizes, and even totally different logic structures such as AICs [Parandeh-Afshar et al. 2012] can still

Table I. Comparison of Architecture Experiments Supported by the VTR and Titan Flows

Experiment Modification	VTR	Titan	Titan Flow Method
Device Floorplan	Yes	Yes	Architecture file
Inter-cluster Routing	Yes	Yes	Architecture file
Clustered Block Size/Configuration	Yes	Yes	Architecture file
Intra-cluster Routing	Yes	Yes	Architecture file
Logic Element Structure	Yes	Yes	Architecture file
LUT size/Combinational Logic	Yes	Yes	ABC re-synthesis
New RAM Block	Yes	Yes	Architecture file (up to 16K depth)
New DSP Block	Yes	Yes	Architecture file (up to 36 bit width)
New Primitive Type	Yes	No	No method to pass black box through Quartus II

occur. RAM is represented as device-independent "RAM slices," which are typically 1 bit wide, and up to 14 address bits deep. These RAM slices are packed into larger physical RAM blocks by VPR, and hence arbitrary RAM architectures can be investigated. Similarly, multiplier primitives (up to 36×36 bits) are packed into DSP blocks by VPR, allowing a variety of experiments. A simple remapping tool could also resize the multiplier primitives if desired. The structure of a logic element (connectivity, number of Flip-Flops, etc.) can also be modified without having to resynthesize the design, and interblock routing architecture and electrical design can both be arbitrarily modified. Compared to VTR, the largest limitation is the inability to add support for new primitive types.

Another use of Titan is to test and evaluate CAD tool quality. Both physical CAD (e.g., packing, placement, routing) and logic resynthesis tools can be plugged into the flow. Titan provides a front-end interface between commercial and academic CAD flows that is complementary to the back-end VPR-to-bitstream interface presented in Hung et al. [2013]. Overall, the Titan flow enables a wide range of FPGA architecture experiments, can be used to evaluate new CAD algorithms on realistic architectures with realistic benchmark circuits, and allows for more extensive scalability testing with larger benchmarks.

4. BENCHMARK SUITE

We selected the 23 largest benchmarks that we could obtain from a diverse set of application domains to create the Titan23 benchmark suite. The benchmarks often required minor alteration to make them compatible with the Titan flow. The conversion methodology is described in Murray et al. [2013b].

4.1. Titan23 Benchmark Suite

The Titan23 benchmark suite consists of 23 designs ranging in size from 90K to 1.8M primitives, with the smallest utilizing 40% of a Stratix IV EP4SGX180 device, and the largest designs unable to fit on the largest Stratix IV device. The designs represent a wide range of real-world applications and are listed in Table II. All benchmarks make use of some or all of the different heterogeneous blocks available on modern FPGAs, such as DSP and RAM blocks.

Although these benchmarks (as released) will synthesize with Altera's Quartus II, it should also be possible to use them in other tool flows such as Torc [Steiner et al. 2011] and RapidSmith [Lavin et al. 2011] by replacing the Altera IP cores with equivalents from the appropriate vendor.

4.2. Comparison to Other Benchmark Suites

The characteristics just outlined make the Titan23 benchmark suite quite different from the popular MCNC20 benchmarks [Yang 1991], which consist of primarily

Table II. Titan23 Benchmark Suite

Name	Total Blocks	Clocks	ALUTs	REGs	DSP 18×18s	RAM Slices	RAM Bits	Application
gaussianblur	1,859,485	1	805,063	1,054,068	16	334	1,702	Image Processing
bitcoin_miner	1,061,829	2	455,263	546,597	0	59,968	297,664	SHA Hashing
directrf	934,490	2	471,202	447,032	960	40,029	20,307,968	Communications/DSP
sparcT1_chip2	824.152	2	377,734	430,976	24	14,355	1,585,435	Multi-core μP
LU_Network	630,103	2	194,511	399,562	896	41,623	9,388,992	Matrix Decomposition
LU230	567,992	2	208,996	293,177	924	64,664	10,112,704	Matrix Decomposition
mes_noc	549,045	9	274,321	248,988	0	25,728	3,99,872	On Chip Network
gsm_switch	491,846	4	159,388	296,681	0	35,776	6,254,592	Communication Switch
denoise	342,899	1	322,021	8,811	192	11,827	1,135,775	Image Processing
sparcT2_core	288,005	2	169,498	109,624	0	8,883	371,917	μP Core
cholesky_bdti	256,072	1	76,792	173,385	1,043	4,920	4,280,448	Matrix Decomposition
minres	252,454	2	107,971	126,105	614	17,608	8,933,267	Control Systems
stap_qrd	237,197	1	72,263	161,822	579	9,474	2,548,957	Radar Processing
openCV	212,615	1	108,093	86,460	740	16,993	9,412,305	Computer Vision
dart	202,368	1	103,798	87,386	0	11,184	955,072	On Chip Network Simulator
bitonic_mesh	191,664	1	109,633	49,570	676	31,616	1,078,272	Sorting
segmentation	167,917	1	155,568	6,561	104	5,658	3,166,997	Computer Vision
SLAM_spheric	125,194	1	112,758	8,999	296	3,067	9,365	Control Systems
des90	109,811	1	62,871	30,244	352	16,256	560,640	Multi μP system
cholesky_mc	108,236	1	29,261	74,051	452	5,123	4,444,096	Matrix Decomposition
stereo_vision	92,662	3	38,829	49,049	152	4,287	203,777	Image Processing
sparcT1_core	91,268	2	41,968	45,013	8	4,277	337,451	μP Core
neuron	90,778	1	24,759	61,477	565	3,799	638,825	Neural Network

combinational circuits and make no use of heterogeneous blocks. Furthermore, the MCNC designs are extremely small. The largest (`clma`) uses less than 4% of a Stratix IV EP4SGX180 device, making it one to two orders of magnitude smaller than modern FPGAs.

Another benchmark suite of interest is the collection of 19 benchmarks included with the VTR design flow. These benchmarks are larger than the MCNC benchmarks, with the largest (`mcml`) reported to use 99.7K 6-LUTs [Rose et al. 2012]. Interestingly, when this circuit was run through the Titan flow, it uses only 11.7K Stratix IV ALUTs (6-LUTs) after synthesis, indicating the differences between ODINII+ABC and Quartus II's integrated synthesis. Additionally, only 10 of the VTR circuits make use of heterogeneous resources. The Titan23 benchmark suite provides substantially larger benchmark circuits that make more extensive use of heterogeneous resources.

Several non-FPGA-specific benchmark suites also exist. The various ISPD benchmarks [Viswanathan et al. 2011] are commonly used to evaluate ASIC tools, but are only available in gate-level netlist formats. This makes them unsuitable for use as FPGA benchmarks since they are not mapped to the appropriate FPGA primitives. The IWLS 2005 benchmarks [IWLS 2005] are available in HDL format, and the Titan flow enables them to be used with FPGA CAD tools. However, the largest design consists of only 36K blocks after running through the Titan flow—too small to be included in the Titan23.

5. ARCHITECTURE MODEL ENHANCEMENTS AND MODIFICATIONS

Several enhancements have been made to the Stratix IV architecture model used in Murray et al. [2013b], with the dual aims of enabling a reasonably accurate comparison of the timing optimization capabilities of VPR and Quartus II and providing a realistic architecture on which enhanced CAD algorithms can be tested.

5.1. Carry Chains

Most modern FPGAs such as Stratix IV have embedded carry chains, which are used to speed up arithmetic computations. These structures are important from a timing perspective because they help to keep the otherwise slow carry propagation from dominating a circuit's critical path. VPR 7 supports chain-like structures, which are identified during packing and kept together as hard macros during placement. Using this feature, we were able to model the carry chain structure in Stratix IV, which runs downward through each LAB and continues in the LAB below.

One of VPR's limitations when modeling carry chains is that a carry chain cannot exit a LAB early if the LAB runs out of inputs. In Stratix IV, the full adder and LUT are treated as a single primitive, where the adder is fed by the associated LUT. This allows additional logic (such as a mux or the XOR for an adder/subtractor) to be placed in the LUT. However, for a full LAB carry chain (20-bits), this additional logic may require more inputs than the LAB can provide. This issue is avoided in Stratix IV by allowing the carry chain to exit early, at the midpoint of the LAB, and continue in the LAB below [Lewis et al. 2005]. Since this behaviour is not supported in VPR, we had to increase the number of inputs to the LAB to 80 to ensure that VPR would be able to pack carry chains successfully. This is notably higher than the 52 inputs that exist in Stratix IV and may allow VPR to pack more logic inside each LAB as a result.

5.2. Direct-Link Interconnect and Three-Sided LABs

Stratix IV devices also have a "Direct-Link" interconnect between horizontally adjacent blocks [Altera Corporation 2012a]. This allows adjacent blocks to communicate directly by driving one another's local (intrablock) routing without having to use global routing wires. These connections act as fast paths between adjacent blocks and also help to reduce demand for global routing resources.

Within VPR, these connections were modeled as additional edges (switches) in the routing resource graph connecting the output and input pins of adjacent LABs. As modeled, each LAB can drive and receive 20 signals to/from each of its horizontally adjacent LABs. To ensure that this capability was fully exploited, VPR's placement delay model was enhanced to account for these fast connections.

Additionally, Stratix IV LABs can only drive global routing segments on three sides (left, right, and top). This was modeled by distributing all block pins along those sides.

5.3. Improved DSP Packing and Spacing

One of the differences identified in previous work was that VPR used significantly ($\sim$2.3$\times$) more DSP blocks than Quartus II [Murray et al. 2013b]. It was also observed that VPR's packer spent a large amount of time packing DSP blocks.

In an attempt to improve these results, we provided hints ("pack patterns") to VPR's packer indicating that certain sets of netlist primitives should be kept together. Doing this for two DSP operating modes (which account for 80% of all DSP modes in the Titan23 benchmarks) significantly decreased both the number of DSP blocks required and the time required to pack DSP heavy circuits.

We also found that, when run in VPR, many DSP heavy circuits required substantially larger devices than when run in Quartus II. This was caused by the relatively low DSP density of the EP4SE820 device upon which the architecture model's floorplan was based. To resolve this issue, we reduced the spacing between DSP columns from 92 to 40 columns, resulting in a DSP density more comparable to the smaller and more DSP-focused Stratix IV devices.

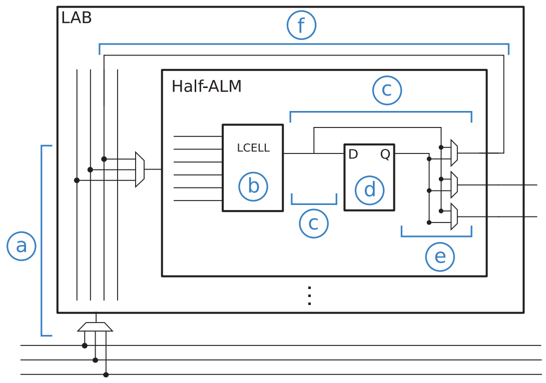

Fig. 2. Simplified LAB diagram illustrating modeled delays.

Table III. Modeled LAB Delay Values

Location	Delay (ps)	Description
a	171	LAB Input
b	261	LUT Comb. Delay
	11	C_{in} to C_{out} (Normal)
	65	C_{in} to C_{out} (Mid-LAB)
	124	C_{in} to C_{out} (Inter-LAB)
c	25	LUT to FF/ALM Out
d	66	FF T_{su}
	124	FF T_{cq}
e	45	FF to ALM Out
f	75	LAB Feedback

5.4. Constant Nets

Although Quartus II will recognize that netlist primitive ports connected to vcc or gnd can be tied off within the primitive, VPR does not and will attempt to route these (potentially high fanout) constant nets. To avoid this behavior, the VQM2BLIF netlist converter now removes such constant nets from the generated BLIF netlist.

6. TIMING MODEL

One of the primary limitations of the previous work to compare VPR and Quartus II was that both tools were run only in wire length (WL)-driven mode [Murray et al. 2013b]. Since real-world industrial CAD tools would be almost exclusively run with timing optimization enabled, it is important to compare both VPR and Quartus II in this mode. However, this comparison requires that VPR have a reasonably accurate timing model. This ensures that both tools will face similar optimization problems and that the final critical path delays can be fairly compared.

Although it is practically impossible to create an identical timing model between VPR and Quartus II, we have captured the major timing characteristics of Stratix IV devices. To do so, we used micro-benchmarks to evaluate specific components of the Stratix IV architecture. Timing delays were extracted from post-place-and-route circuits using Quartus II's TimeQuest Static Timing Analyzer for the Slow 900mV 85C timing corner. Delay values were averaged across multiple locations on the device to account for location-based delay variation.

Some device primitives in Stratix IV contain optional input and/or output registers. To capture the timing impact of these optional registers, VQM2BLIF was enhanced to identify blocks using such registers and generate a different netlist primitive, thus allowing a different timing model to be used.

6.1. LAB Timing

The LAB timing model captures many of the important timing characteristics of the block, as shown in Figure 2 and Table III. The carry chain delay varies depending on where in the LAB it is located. As noted in Table III, the delay is normally 11ps, but can be larger when crossing the midpoint of the LAB (due to crossing the extra control logic in that area) and when crossing between LABs.

One limitation of VPR compared to Quartus II is that it does not rebalance LUT inputs so that critical signals use the fastest inputs. As a result, we model all LUT inputs as having a constant combinational delay equal to the average delay of the six Stratix IV LUT inputs.

6.2. RAM Timing

In Stratix IV, inputs to RAM blocks are always registered, but the outputs can be either combinational or registered. Since VPR does not support multicycle primitives, we model each RAM block as a single sequential element with a short or long clock-to-q delay depending on whether the output is registered or combinational. Although this neglects the internal clock cycle from a functional perspective, it remains accurate from a delay perspective provided the clock frequency does not exceed the maximum supported by the blocks (540 and 600MHz for the M144K and M9K, respectively) [Altera Corporation 2012a].

6.3. DSP Timing

Each Stratix IV DSP block consists of two types of device primitives: multipliers (`mac_mults`) and adder/accumulators (`mac_outs`) [Altera Corporation 2009]. For the `mac_mult` primitive, inputs can be optionally registered, whereas the output is always combinational. For the case with no input registers, the primitive is modeled as a purely combinational element. For the case with input registers, it is modeled as a single sequential element with the combinational output delay included in the clock-to-q delay.

The `mac_out` can have optional input and/or output registers and is modeled similarly, as either a purely combinational element or as a single sequential element with the setup time/clock-to-q delay modified to account for the presence or absence of input/output registers. From a delay perspective, these approximations remain valid provided the clock driving the DSP does not exceed the block's maximum frequency of 600MHz [Altera Corporation 2012a]. The different delay values associated with different `mac_out` operating modes (accumulate, pass-through, two-level adder, etc.) are also modeled.

6.4. Wire Timing

In Murray et al. [2013b], the global routing network was modeled as a combination of length 4 (L4) and length 16 wires (L16). Stratix IV uses length 4 wires, with additional length 12 wires in the vertical and length 20 wires in the horizontal directions.

For the modeled wires, resistance, capacitance, and driver switching delay values were chosen based on ITRS 45nm data and adjusted to match the average delays observed in Quartus II. The modeled L4 wire parameters were chosen to match Stratix IV's length 4 wire delays, and the modeled L16 wire parameters were chosen to match the averaged behavior of Stratix IV's length 12 and 20 wires.

6.5. Other Timing

A basic timing model was included for simple I/O blocks, and a zero delay model was used for other more complex I/O blocks (such as DDR) and is included only so that circuits including such blocks will run through VPR correctly. As a result, I/O timing should be considered approximate and is not reported.

6.6. VPR Limitations

Although VPR supports multiclock circuits, it does not support multiclock netlist primitives (e.g., RAMs with different read and write clocks). To work around this issue, VQM2BLIF was enhanced to (optionally) remove extra clocks from device primitives to allow such circuits to run through VPR.

Table IV. Stratix IV Timing Model Correlation Results

Benchmark	VPR Path Delay (ps)	Quartus II Path Delay (ps)	VPR:Q2 Delay Ratio	Note
`L4 Wire`	131	132	0.99	
`L16 Wire`	293	289	1.01	
`32-bit Adder`	1,674	1,718	0.97	
`8:1 Mux`	932	1,498	0.62	Extra inter-block wire
`8-bit LFSR`	3,400	3,346	1.02	
`18-bit Comb. Mult`	9,494	8,760	1.08	
`32-bit Reg. Mult`	7,751	7,015	1.10	
`M9K Comb. Output`	4,757	4,813	0.99	
`M9K Reg. Output`	3,733	3,788	0.99	
`diffeq1`	9,935	11,289	0.88	Small Benchmark
`sha`	6,103	5,416	1.13	Small Benchmark

VPR also treats clock nets specially, requiring that clock nets not connect to non-clock ports and vice versa. This occurs occasionally in Quartus II's VQM output and is fixed by VQM2BLIF, which disconnects clock connections to nonclock ports and replaces nonclock connections to clock ports with valid clocks.

Although both of these work-arounds do modify the input netlist, they typically only affect a small portion of a design's logic. However, despite these modifications, some circuits were unable to run to completion due to bugs in VPR.

6.7. Timing Model Verification

To verify the validity of our timing model, we ran micro-benchmarks through both VPR and Quartus II and compared the resulting timing paths. Using small micro-benchmarks helps to minimize the optimization differences between each tool. The correlation results for a subset of these benchmarks are shown in Table IV.

The correlation is reasonably accurate, with VPR's delay falling within 10% of the delay measured in Quartus II, except for the `8:1 Mux`, `diffeq1`, and `sha` benchmarks. For the `8:1 Mux`, Quartus II uses an additional interblock routing wire that VPR does not, thus accounting for the delay difference. The `diffeq1` and `sha` benchmarks, while still small, are large enough so that each tool produces a different optimization result.

7. BENCHMARK RESULTS

In this section, we use the Titan23 benchmark suite described in Section 4, in conjunction with the enhanced Stratix IV architecture capture and timing model described in Sections 5 and 6. This allows us to compare the popular academic VPR tool with Altera's commercial Quartus II software. Using the Stratix IV architecture capture, VPR was able to target an architecture similar to the one targeted by Quartus II, thus allowing a coarse comparison of CAD tool quality.

7.1. Benchmarking Configuration

In all experiments, version 12.0 (no service packs) of Quartus II was used, whereas a recent revision of VPR 7.0 (r4292) was used. During all experiments, a hard limit of 48 hours runtime was imposed; any designs exceeding this time were considered to have failed to fit. Most benchmarks were run on systems using Xeon E5540 (45nm, 2.56GHz) processors with either 16GB or 32GB of memory. For some benchmarks, systems using Xeon E7330 (65nm, 2.40GHz) and 128GB of memory or Xeon E5-2650 (32nm, 2.00GHz) and 64GB of memory were used. Where required, runtime data are scaled to remain comparable across different systems.

To ensure both tools were operating at comparable effort levels, VPR packing and placement were run with the default options, whereas Quartus II was run in

Table V. Timing-Driven and Enhanced Architecture Tool Performance Impact

Performance Metric	Baseline	No Timing	No Chains	No Direct	No DSP Hints
Pack	1.00	1.55	1.45	1.01	2.42
Place	1.00	0.45	0.94	1.03	1.11
Route	1.00	0.15	0.62	1.18	0.96
Total	1.00	0.28	0.68	1.15	1.21
Peak Memory	1.00	1.02	1.02	1.00	1.08

`STANDARD_FIT` mode. Due to long routing convergence times, VPR was allowed to use up to 400 routing iterations instead of the default of 50. Quartus II supports multi-threading, but was restricted to use a single thread to remain comparable with VPR.

Quartus II targets actual FPGA devices that are available only in discrete sizes. In contrast, VPR allows the size of the FPGA to vary based on the design size. Although it is possible to fix VPR's die size, we allowed it to vary so that differences in block usage after packing would not prevent a circuit from fitting. To remain comparable to Stratix IV, VPR's routing channel width was set to 300.

To enable a fair comparison of timing optimization results, we constrained both tools with equivalent timing constraints. All paths crossing netlist clock domains were cut, thus ensuring that the tools can focus on optimizing each clock independently. The benchmark I/Os were constrained to a virtual I/O clock with loose input/output delay constraints. Paths between netlist clock domains and the I/O domain were analyzed to ensure that the tools cannot (unrealistically) ignore I/O timing [Altera Corporation 2007]. All clocks were set to target an aggressive clock period of 1ns. Since VPR does not model clock uncertainty, clock uncertainty was forced to zero in Quartus II. Similarly, VPR does not model clock skew across the device; this cannot be disabled in Quartus II, but its timing impact is small (typically less than 100ps).

7.2. Quality of Results Metrics

Several key metrics were measured and used to evaluate the different tools. They fall into two broad categories.

The first category focuses on tool computational needs, which we quantify by looking at wall clock execution time for each major stage of the design flow (Packing, Placement, Routing), as well as the total runtime and peak memory consumption.

The second category of metrics focus on the Quality of Results (QoR). We measure the number of physical blocks generated by VPR's packer and the total number of physical blocks used by Quartus II. Another key QoR metric is Wire Length (WL). Unlike VPR, Quartus II reports only the routed WL and does not provide an estimate of WL after placement. If a circuit fails to route in VPR, we estimate its required routed WL by scaling VPR's placement WL estimate by the average gap between placement estimated and final routed WL ($\sim$31%). Finally, with a Stratix IV like timing model included in the architecture capture, we also compare circuit critical path delay. This was done using the timing constraints described in Section 7.1. For multiclock circuits, we report the geometric mean of critical path delays across all clocks, excluding the virtual I/O clock.

7.3. Timing Driven Compilation and Enhanced Architecture Impact

It is useful to quantify the impact of running VPR in timing-driven mode and the impact of the architectural changes outlined in Section 5. This was evaluated by either disabling timing-driven compilation or specific architecture features. The results shown in Tables V and VI are averaged across the benchmarks that ran to completion and normalized to the fully featured architecture run in timing-driven mode.

Disabling timing-driven compilation in VPR resulted in significant runtime improvements. In particular, placement and routing took 0.45$\times$ and 0.15$\times$ as long, respectively,

Table VI. Timing-Driven and Enhanced Architecture QoR Impact

QoR Metric	Baseline	No Timing	No Chains	No Direct	No DSP Hints
LABs	1.00	0.99	1.01	1.00	1.00
DSP	1.00	1.12	1.09	1.00	2.22
M9K	1.00	1.00	1.00	1.00	1.01
M144K	1.00	1.00	1.00	1.00	0.97
WL	1.00	0.79	1.04	1.01	1.10
Crit. Path Delay	1.00	—	2.16	1.03	1.12

whereas packing took 1.55× longer. VPR's runtime is usually dominated by routing (Section 7.4), and, as a result, VPR ran 3.6× faster in non-timing-driven mode. Although the speed-up during placement seems reasonable, since no timing analysis is being performed, the large speed-up in the router makes it clear that VPR's timing-driven router suffers from convergence issues on this architecture. As expected when run in non-timing-driven mode, the routed WL decreases to 0.79× compared to timing-driven mode.

Disabling carry chains (Section 5.1) increases packer runtime by 1.45× but reduces routing runtime to 0.62×. The slow-down in the packer indicates that carry chains provide useful guidance to the packer. The speed-up in the router can be attributed to the reduction in routing congestion caused by the dispersal of input and output signals used by the carry chains. From a timing perspective, disabling carry chains has a significant impact, increasing critical path delay by 2.16×.

Disabling the direct links between adjacent LABs (Section 5.2) increases router runtime to 1.18× and results in a small (3%) increase in critical path delay. This indicates that the direct-link connections make the architecture easier to route.

Disabling the packing hints for DSP blocks (Section 5.3) increased the packer runtime by 2.42× while also increasing the required number of DSP blocks by 2.22×. This increase in DSP blocks had an appreciable impact on WL and critical path delay, which increased by 10% and 12%, respectively.

7.4. Performance Comparison with Quartus II

Table VII shows both the absolute runtime and peak memory of VPR and the relative values compared to Quartus II on the Titan23 benchmark suite using the enhanced architecture. Quartus II's absolute runtime and peak memory across the same benchmarks while targeting Stratix IV are shown in Table VIII. Both tools were run in timing-driven mode.

VPR spends most of its time on routing, which takes on average 80% of the total runtime on benchmarks that completed. In contrast, Quartus II has a more even runtime distribution, with placement taking the largest amount of time (38%) and with a significant amount of time (28% and 25%) spent on routing and miscellaneous actions, respectively. For both tools, runtime can be quite substantial on larger benchmarks, taking in excess of 48 hours.[1] Looking at the relative runtime of the two tools in Table VII, we can gain additional insights into each step of the CAD flow.

Packing is slower (2.2×) in VPR than in Quartus II, which can be partly attributed to VPR's more flexible packer, which allows it to target a wide range of FPGA architectures.

On average, both VPR and Quartus II spend a comparable amount of time during placement, with VPR using 19% less execution time. However this is somewhat pessimistic for VPR since it also spends time generating the delay map used for placement.

[1]In contrast, the largest MCNC20 circuit took 60 seconds in VPR and 65 seconds in Quartus II, highlighting the importance of using large benchmarks to evaluate CAD tools.

Table VII. VPR 7 Runtime in Minutes and Memory in GB

Name		TotalBlocks	Pack		Place		Route		Total		Mem.		Outcome
gaussianblur	*	1,859,485	745.8										ERR
bitcoin_miner	*	1,061,829	248.1	(2.38×)	427.7	(0.35×)							UNR
directrf	*	934,490											ERR
sparcT1_chip2	†	824,152	76.8	(1.01×)	117.1	(0.47×)	568.7		762.6		46.0		
LU_Network	†	630,103	48.2	(1.45×)	113.1	(0.84×)							OOT
LU230	*	567,992	148.3	(1.82×)									OOM
mes_noc	†	549,045	53.2	(2.84×)	117.2	(1.21×)	433.0	(7.90×)	603.4	(2.72×)	39.0	(5.42×)	
gsm_switch	*	491,846	85.3	(1.94×)	204.1	(1.07×)							OOT
denoise		342,899	39.8	(3.01×)	111.8	(1.21×)	1,335.7	(27.86×)	1,487.4	(8.14×)	25.0	(4.60×)	
sparcT2_core		288,005	37.0	(3.33×)	50.1	(0.71×)	348.3	(9.16×)	435.4	(3.06×)	18.0	(4.58×)	
cholesky_bdti		256,072	16.6	(1.51×)	32.0	(0.77×)	188.2	(12.17×)	236.8	(2.67×)	25.0	(6.78×)	
minres	†	252,454	13.8	(1.76×)	20.9	(0.65×)	135.4	(9.28×)	170.1	(2.38×)	42.0	(9.96×)	
stap_qrd		237,197	15.3	(1.04×)	47.1	(1.31×)	86.7	(7.05×)	149.0	(1.83×)	23.0	(6.65×)	
openCV	†	212,615	14.2	(2.63×)	20.9	(0.84×)							OOT
dart		202,368	17.7	(2.34×)	20.6	(0.73×)							OOT
bitonic_mesh	†	191,664	19.2	(3.87×)	28.2	(0.91×)	1,914.9	(20.02×)	1,962.3	(12.86×)	55.0	(11.63×)	
segmentation		167,917	17.1	(3.07×)	37.4	(0.99×)	546.1	(22.30×)	600.5	(7.30×)	17.0	(5.61×)	
SLAM_spheric		125,194	12.0	(2.90×)	22.2	(0.98×)							OOT
des90	†	109,811	9.3	(4.22×)	12.4	(0.80×)	228.6	(5.61×)	250.3	(3.63×)	28.0	(9.29×)	
cholesky_mc		108,236	6.1	(1.94×)	10.2	(0.85×)	30.4	(4.74×)	46.6	(1.34×)	16.0	(6.90×)	
stereo_vision		92,662	3.3	(1.27×)	8.0	(0.69×)	11.1	(3.31×)	22.4	(0.96×)	9.2	(5.30×)	
sparcT1_core		91,268	9.8	(3.77×)	8.7	(0.85×)	46.0	(3.61×)	64.5	(1.94×)	7.1	(3.89×)	
neuron		90,778	4.6	(1.90×)	7.4	(0.71×)	19.6	(3.46×)	31.5	(1.08×)	10.0	(4.63×)	
Geomean			26.4	(2.20×)	36.3	(0.81×)	171.0	(8.23×)	229.4	(2.82×)	21.8	(6.21×)	

Relative speed to Quartus II (VPR/Q2) is shown in parentheses.
ERR: Error in VPR. UNR: Unroute. OOT: Out of Time (>48 hours). OOM: Out of Memory (>128GB).
*Run on 128GB machine. †Run on 64GB machine.

Table VIII. Quartus II Runtime in Minutes and Memory in GB

Name		Total Primitives	Pack	Place	Route	Misc.	Total	Mem.	Note
gaussianblur	*	1,859,485							DEV
bitcoin_miner	*	1,061,829	104.1	1226.8	2387.6	337.5	4379.9	10.5	
directrf	*	934,490							DEV
sparcT1_chip2	*	824,152	76.3	251.3					OOT
LU_Network	*	630,103	33.2	134.7	85.4	57.3	300.2	8.4	
LU230	*	567,992	81.6	290.1	211.3	122.7	823.5	9.5	
mes_noc	*	549,045	18.7	96.6	54.8	63.4	222.2	7.2	
gsm_switch	*	491,846	44.0	190.7	266.0	40.1	579.2	7.0	
denoise		342,899	13.2	92.4	48.0	29.1	182.6	5.4	
sparcT2_core		288,005	11.1	70.1	38.0	23.1	142.4	3.9	
cholesky_bdti		256,072	11.0	41.5	15.5	20.9	88.8	3.7	
minres	*	252,454	7.9	32.1	14.6	20.6	71.4	4.2	
stap_qrd		237,197	14.7	35.9	12.3	18.7	81.6	3.5	
openCV	*	212,615	5.4	24.8	11.6	15.9	54.8	3.7	
dart		202,368	7.6	28.0	23.9	741.9	801.3	3.2	
bitonic_mesh	*	191,664	5.0	31.0	95.7	25.6	152.6	4.7	
segmentation		167,917	5.6	37.8	24.5	14.4	82.2	3.0	
SLAM_spheric		125,194	4.2	22.7	16.2	13.0	56.1	2.6	
des90	*	109,811	2.2	15.5	40.7	12.8	69.0	3.0	
cholesky_mc		108,236	3.1	11.9	6.4	13.3	34.8	2.3	
stereo_vision		92,662	2.6	11.6	3.4	5.9	23.4	1.7	
sparcT1_core		91,268	2.6	10.3	12.8	7.6	33.3	1.8	
neuron		90,778	2.4	10.4	5.7	10.9	29.3	2.2	
Geomean			10.3	48.9	32.8	28.8	133.4	4.0	

DEV: Exceeded size of largest Stratix IV device. OOT: Out of Time (>48 hours).
*Runtime scaled to 64GB or 128GB machine.

Quartus II, in contrast, uses a precomputed device delay model. This is an example of where VPR has additional overhead because of its architecture independence. Additionally, VPR typically uses fewer LABs than does Quartus II (see Section 7.5), which decreases the size of VPR's placement problem. Quartus II also enforces stricter placement legality constraints and uses more intelligent directed moves than VPR, which also affect its runtime [Ludwin and Betz 2011].

VPR's timing-driven router is also substantially slower (8.2×) than Quartus II's. Furthermore, the router's runtime is volatile, ranging from 3.3× slower in the best case to nearly 28× slower in the worst case. This can be partly attributed to VPR's default congestion resolution schedule, which increases the cost of overused resources slowly with the aim of achieving low critical path delay.

As to overall runtime, for benchmarks it successfully fits, VPR takes 2.8× longer than Quartus II. However, it should be noted that this result is skewed in VPR's favor since it does not account for benchmarks that did not complete. Peak memory consumption is also much higher (6.2×) in VPR. This is quite significant and will often limit the design sizes VPR can handle. It is interesting to note that the largest benchmark that Quartus II will fit (`bitcoin_miner`), uses approximately the same memory in Quartus II as the smallest Titan23 benchmark (`neuron`) uses in VPR.

It is also useful to compare the scalability of VPR and Quartus II with design size since scalable CAD tools are required to continue exploiting Moore's Law. As shown in Table VII, VPR is unable to complete at least six of the benchmarks due to either excessive memory or runtime. Quartus II, in contrast, completes all but one of the benchmarks that fit on Stratix IV devices (Table VIII). Furthermore, when considering total runtime, VPR is closest (1.0×–1.9×) to Quartus II on the four smallest

Table IX. VPR 7/Quartus II Quality of Result Ratios

Name	Total Primitives	LAB	DSP	M9K	M144K	WL	Crit. Path	Note
gaussianblur	1,859,485							
bitcoin_miner	1,061,829	0.89		0.91	3.45	3.85		*
directrf	934,490							
sparcT1_chip2	824,152							
LU_Network	630,103	1.38	1.00	1.00		2.86		*
LU230	567,992	0.53	1.00	3.57	21.38			
mes_noc	549,045	0.84		1		1.97	1.37	
gsm_switch	491,846	0.65		1.48		2.38		*
denoise	342,899	0.73	1.50	2.66		1.77	1.02	
sparcT2_core	288,005	0.92		1.00		1.43	1.51	
cholesky_bdti	256,072	1.03	1.02	1		2.58	1.87	
minres	252,454	0.61	1.49	1.00		2.69	1.59	
stap_qrd	237,197	1.75	0.99	0.76		2.81	2.52	
openCV	212,615	0.78	1.31	1.15	1.00	3.30		*
dart	202,368	0.72		0.93		2.26		*
bitonic_mesh	191,664	0.65	0.77	0.96	1.94	1.77	1.77	
segmentation	167,917	0.70	1.17	1.32	2.50	1.76	1.10	
SLAM_spheric	125,194	0.66	1.09			1.52		*
des90	109,811	0.67	0.56	0.95		1.70	1.33	
cholesky_mc	108,236	0.87	0.98	1.10	1.00	2.43	2.44	
stereo_vision	92,662	0.71	4.00	1.11		2.24	1.21	
sparcT1_core	91,268	0.89	1.00	1.1		1.31	1.16	
neuron	90,778	0.70	0.82	1.65		2.61	1.84	
Geomean		0.80	1.12	1.20	2.67	2.19	1.53	

* VPR WL scaled from placement estimate.

benchmarks but generally falls behind as design size increases. From these results, it appears that Quartus II scales better with increasing design size than VPR.

These results are notably different from those previously reported for WL-driven optimization in Murray et al. [2013b]. The most significant difference is that VPR's runtime is now spent primarily during routing, rather than during packing. This is attributable to two main factors. First, VPR's packing performance has been significantly improved due to recent algorithmic enhancements and the addition of packing hints (Section 5.3). Second, VPR's timing-driven router is significantly slower (Section 7.3) than the WL-driven router, often requiring significantly more routing iterations to resolve congestion. We observed that VPR spends a large number of later routing iterations attempting to resolve congestion on only a handful of overused routing resources, which were always logic block output pins. Additionally, we found that small tweaks to the router cost parameters or architecture can cause large variations in the timing-driven router's run time.

7.5. Quality of Results Comparison with Quartus II

The relative QoR results for the Titan23 benchmark suite are shown in Table IX. These results show several trends. First, VPR uses fewer LABs (0.8×) than Quartus II. Whereas this reduced LAB usage may initially seem a benefit (since a smaller FPGA could be used), this comes at the cost of WL, as will be discussed in Section 7.6.

Looking at the other block types, VPR uses 1.1× as many DSP blocks and 1.2× as many M9K blocks as Quartus II, showing that Quartus II is somewhat better at utilizing these hard block resources. Since only six circuits use M144K blocks in both tools, it is difficult to draw meaningful conclusions.

Routed WL is one of the key metrics for comparing the overall quality of VPR and Quartus II. Somewhat surprisingly, the WL gap is quite large, with VPR using 2.2×

Table X. QoR Ratios for Different Quartus II Packing Density and Placement Finalization Settings

Q2 Settings	Q2:Q2 Def. LAB	Q2:Q2 Def. WL	Q2:Q2 Def. Crit. Path	VPR:Q2 LAB	VPR:Q2 WL	VPR:Q2 Crit. Path
Default	1.00	1.00	1.00	0.85	2.07	1.52
No Finalization	1.03	1.09	1.10	0.82	1.90	1.39
Dense	0.85	1.22	1.02	1.01	1.71	1.50
Dense & No Finalization	0.76	1.57	1.19	1.11	1.32	1.28

Note: The default VPR:Q2 values are different from Table IX since some benchmarks would not fit for some Quartus II settings combinations.

more wire than Quartus II.[2] Without access to Quartus II's internal packing, placement, and routing statistics, it is difficult to identify which step(s) of the design flow are responsible for this difference. However, as will be shown in Section 7.6, VPR's packing quality has a significant impact. In addition, it is likely that Quartus II achieves a higher placement quality than VPR, as shown in Ludwin and Betz [2011]. A lower quality placement would increase VPR's routing time and routed WL.

The other key metric to consider is critical path delay. VPR produces a critical path that is 1.5× slower than Quartus II on average. This difference exceeds the range of variation expected between the VPR and Quartus II timing models and indicates that VPR does not match Quartus II at optimizing critical path delay. There are several potential reasons for this. One reason is the connectivity in the interblock routing network. In our Stratix IV model, both long and short wires are accessible from block pins, which limits the number of connections that can easily reach the small number of long wires. In actual Stratix IV devices, long wires are only accessible from short wires [Lewis et al. 2003]. This connectivity may improve delay by allowing the short wires to act as a feeder network for the long wires, thus making them easier to access. Additionally, the use of the Wilton switch block in our architecture model makes it unlikely that long wires will connect to other long wires, potentially limiting their benefit. VPR also tends to pack more densely than Quartus II and is unable to take apart clusters after packing to correct poor packing decisions, both of which may increase VPR's critical path delay. Finally, Quartus II has additional algorithmic optimizations (not included in VPR) that help it to achieve lower critical path delay, such as timing budgeting during routing [Fung et al. 2008].

Compared to the previously reported WL-driven results, the relative QoR between the two tools is similar, with VPR still using fewer LABs and using additional wire compared to Quartus II. The most significant change, the decrease in the relative amount of DSP blocks, can be attributed to the hints given to VPR's packer (Section 5.3).

7.6. Modified Quartus II Comparison

To investigate the impact of packing density and taking apart clusters, we re-ran the benchmarks through Quartus II using several different combinations of packing and placement settings. The impact of these settings on the relative QoR between VPR and Quartus II is shown in Table X.

We investigated the effect of telling Quartus II to always pack densely and the effect of disabling "placement finalization." In its default mode, Quartus II varies packing density based on the expected utilization of the targeted FPGA, spreading out the design if there is sufficient space. Also by default, Quartus II performs placement finalization, in which it breaks apart clusters by moving individual LUTs and flip-flops.

[2]The WL gap is quite different (0.7×) on the largest MCNC20 circuit, emphasizing how modern benchmarks can impact CAD tool QoR.

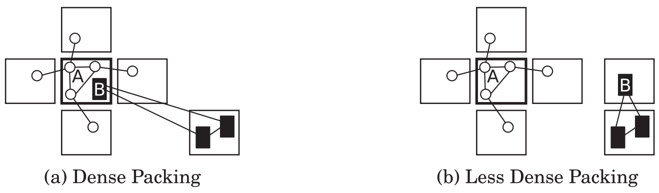

Fig. 3. Packing density and wire length example.

Disabling placement finalization resulted in a moderate increase in Quartus II's WL and critical path delay. Forcing Quartus II to pack densely significantly reduced the number of LABs used but caused a large increase in Quartus II's WL, thus narrowing the WL gap between VPR and Quartus II while having minimal impact on critical path delay. Simultaneously disabling finalization and forcing dense packing further reduced the number of LABs used, further increased Quartus II's WL, and significantly increased Quartus II's critical path delay. With these settings, the WL gap between VPR and Quartus II reduced to 1.3× from the original 2.1×, while the critical path delay gap reduced from 1.5× to 1.3×.

This indicates that significant portions of VPR's higher WL and critical path delay are due to packing effects. The focus on achieving high packing density hurts WL, whereas the inability to correct poor packing decisions (no placement finalization) hurts critical path delay. Together, these settings have an even larger impact. We suspect that VPR's packer is sometimes packing largely unrelated logic together to minimize the number of clusters. This appears to be counterproductive from a WL and delay perspective.

For example, consider a LAB (Figure 3(a)) that is mostly filled with related logic A, but which can accommodate an extra unrelated register B. During placement, the cost of moving this LAB will be dominated by the connectivity to the related logic A. This could result in a final position that is good for A but may be very poor for the extra register B (i.e., far from its related logic). If this is a common occurrence, it could lead to increased WL and critical path delay.

A better solution (Figure 3(b)) would have been to utilize additional clusters (pack less densely) to avoid packing unrelated logic together. Alternately, if the placement engine was able to recognize the competing connectivity requirements inside a cluster, it could break it apart, much like Quartus II's placement finalization. These results agree with those presented in Tom and Lemieux [2005], which showed that the routing demand (as measured by the minimum channel width required to route a design) could be significantly decreased by packing logic blocks less densely.

7.7. Comparison of VPR to Other Commercial Tools

In Hung et al. [2013], VPR packing and placement were compared to Xilinx's ISE tool on four VTR benchmarks. Similar to our results, the authors found that VPR produced a denser packing than ISE, had slower critical paths, used more routing resources, took more execution time, and required more memory. Despite differences in methodology and tools, the general conclusion is the same: VPR does not optimize as well and requires more computational resources than commercial CAD tools.

7.8. VPR versus Quartus II Quality Implications

It is clear from the previously presented results that Quartus II outperforms VPR in terms of QoR, performance, and scalability. However, it may be argued that this is not surprising. VPR is used primarily as an academic research platform and, as a result, is capable of targeting a wide range of FPGA architectures. Quartus II, in contrast, is used for FPGA design implementation on real devices and targets the narrower set of Altera FPGA architectures. This means additional optimizations can be made in Quartus II, for both QoR and tool performance, which may not be possible (or have not been implemented) in VPR.

It is important, however, that this gap not be too large. Given the empirical nature of most FPGA CAD and architecture research, research conclusions can become dependant on the CAD tools used [Yan et al. 2002]. To be confident in research conclusions, it is important for CAD tools such as VPR to remain at least reasonably comparable to state-of-the-art commercial tools.

8. CONCLUSION

First, we presented Titan, a hybrid CAD flow that enables the creation of large benchmark circuits for use in academic CAD tools and supports a wide variety of HDLs and range of IP blocks. Second, we presented the Titan23 benchmark suite built using the Titan flow. The Titan23 benchmarks significantly improve the state of open-source FPGA benchmarks by providing designs across a wide range of application domains, which are much closer in both size and style to modern FPGA usage. Third, we presented an enhanced architecture capture, including a correlated timing model, of Altera's Stratix IV family. As a modern high-performance FPGA architecture, this forms a useful baseline for the evaluation of CAD or architecture changes. Finally, we used this benchmark suite and architecture capture to compare the popular academic CAD tool VPR with a state-of-the-art commercial CAD tool, Altera's Quartus II. The results show that VPR is at least 2.8× slower, consumes 6.2× more memory, uses 2.2× more wire, and produces critical paths 1.5× slower than Quartus II. Additional investigation identified VPR's focus on achieving high packing density and inability to take apart clusters to be an important factors in the WL and critical path delay differences.

9. FUTURE WORK

The Titan23 benchmark suite represents a first step forward, but it will need to be continually updated to keep pace with increasing FPGA design size and complexity. Therefore, we would welcome additional benchmark contributions to cover larger design sizes and a wider range of applications.

It is possible that, with large designs, CAD tools may benefit from additional guidance such as a system-level floorplan. It should be possible to use the Titan23 benchmarks in a floorplanning-based flow, provided CAD tool support is available.

Finally, given the substantial gap between VPR and commercial FPGA CAD tools, it is clear that there remains significant room for improvement in the runtime, memory usage, and result quality of this academic CAD tool. Specific areas to focus on include packing for wireability instead of density and faster router convergence with timing optimizations.

ACKNOWLEDGMENTS

We are grateful to the following organizations and individuals for providing benchmark circuits: Altera, Braiden Brousseau, Deming Chen, Jason Cong, George Constantinides, Zefu Dai, Joseph Garvey, IWLS2005 [IWLS 2005], Mark Jervis, LegUP [Canis et al. 2011], Simon Moore, OpenCores.org, OpenSparc.net, Kalin Ovtcharov, Alex Rodionov, Russ Tessier, Danyao Wang, Wei Zhang, and Jianwen Zhu. We also thank David Lewis, Jonathan Rose, and Tim Liu for useful discussions and the anonymous reviewers for their insightful comments.

REFERENCES

Altera Corporation 2007. *Guidance for Accurately Benchmarking FPGAs*. Altera Corporation.

Altera Corporation 2009. *Quartus II University Interface Program*. Altera Corporation.

Altera Corporation 2012a. *Stratix IV Device Handbook*. Altera Corporation.

Altera Corporation 2012b. *Stratix V Device Overview*. Altera Corporation.

V. Betz and J. Rose. 1997. VPR: A new packing, placement and routing tool for FPGA research. In *FPL*. 213–222.

A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J. H. Anderson, S. Brown, and T. Czajkowski. 2011. LegUp: High-level synthesis for FPGA-based processor/accelerator systems. In *FPGA*. 33–36.

R. Fung, V. Betz, and W. Chow. 2008. Slack allocation and routing to improve FPGA timing while repairing short-path violations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 27, 4 (2008), 686–697.

E. Hung, F. Eslami, and S. J. E. Wilton. 2013. Escaping the academic sandbox: Realizing VPR circuits on xilinx devices. In *FCCM*.

IWLS 2005. *2005 Benchmarks*. IWLS.

C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings. 2011. RapidSmith: Do-it-yourself CAD tools for xilinx FPGAs. In *FPL*. 349–355.

D. Lewis, E. Ahmed, G. Baeckler, V. Betz, M. Bourgeault, D. Cashman, D. Galloway, M. Hutton, C. Lane, A. Lee, P. Leventis, S. Marquardt, C. McClintock, K. Padalia, B. Pedersen, G. Powell, B. Ratchev, S. Reddy, J. Schleicher, K. Stevens, R. Yuan, R. Cliff, and J. Rose. 2005. The Stratix II logic and routing architecture. In *FPGA*. 14–20.

D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose. 2003. The Stratix routing and logic architecture. In *FPGA*. 12–20.

C. Loken, D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen, L. J. Dursi, J. Chong, S. Northrup, J. Pinto, N. Knecht, and R. Van Zon. 2010. SciNet: Lessons learned from building a power-efficient top-20 system and data centre. *Journal of Physics: Conference Series* 256, 1 (Nov. 2010).

A. Ludwin and V. Betz. 2011. Efficient and deterministic parallel placement for FPGAs. *ACM Transactions on Design Automation of Electronic Systems* 16, 3 (June 2011), 22:1–22:23.

A. Mishchenko. 2013. *ABC: A System for Sequential Synthesis and Verification*. Berkeley Logic Synthesis and Verification Group.

K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz. 2013a. From quartus to VPR: Converting HDL to BLIF with the titan flow. In *FPL*. 1–1.

K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz. 2013b. Titan: Enabling large and complex benchmarks in academic CAD. In *FPL*. 1–8.

H. Parandeh-Afshar, H. Benbihi, D. Novo, and P. Ienne. 2012. Rethinking FPGAs: Elude the flexibility excess of LUTs with and-inverter cones. In *FPGA*. 119–128.

J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson. 2012. The VTR project: Architecture and CAD for FPGAs from Verilog to routing. In *FPGA*. 77–86.

N. Steiner, A. Wood, H. Shojaei, J. Couch, P. Athanas, and M. French. 2011. Torc: Towards an open-source tool flow. In *FPGA*. 41–44.

M. Tom and G. Lemieux. 2005. Logic block clustering of large designs for channel-width constrained FPGAs. In *DAC*. 726–731.

N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy. 2011. The ISPD-2011 routability-driven placement contest and benchmark suite. In *ISPD*. 141–146.

Xilinx Incorporated 2012. *7 Series FPGAs Overview*. Xilinx Incorporated.

A. Yan, R. Cheng, and S. J. E. Wilton. 2002. On the sensitivity of FPGA architectural conclusions to experimental assumptions, tools, and techniques. In *FPGA*. 147–156.

S. Yang. 1991. *Logic Synthesis and Optimization Benchmarks User Guide 3.0*. Technical Report. MCNC.

W. Zhang, V. Betz, and J. Rose. 2012. Portable and scalable FPGA-based acceleration of a direct linear system solver. *ACM TRETS* 5, 1, Article 6 (March 2012), 26 pages.

Received December 2013; revised February 2014; accepted April 2014