VPR 5.0: FPGA CAD and Architecture Exploration Tools with Single-Driver Routing, Heterogeneity and Process Scaling

Jason Luu, Ian Kuon, Peter Jamieson, Ted Campbell, Andy Ye, Wei Mark Fang, and Jonathan Rose
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto, Toronto, ON, Canada
vpr@eecg.utoronto.ca

ABSTRACT

The VPR toolset [6, 7] has been widely used to perform FPGA architecture and CAD research, but has not evolved over the past decade to include many architectural features now present in modern FPGAs. This paper describes a new version of the toolset that includes four significant features: first, it now supports a broad range of *single-driver* routing architectures [29, 4, 16]. Single-driver routing has significantly different architectural and electrical properties from the multi-driver approach previously modelled, and is now employed in the majority of FPGAs sold. Second, the new release can now model a heterogeneous selection of hard logic blocks, which could include the hard memory and multipliers that are now ubiquitous in FPGAs. Third, we provide optimized *electrical* models of a wide range of architectures in different process technologies, including a range of area-delay tradeoffs for each single architecture. Prior releases of VPR did not publish even one architecture file with accurate resistance and capacitance parameters. Finally, to maintain robustness and to support future development the release includes a set of regression tests to check functionality and quality of result of the output of the tools.

To illustrate the use of the new features, we present a new look at the FPGA area vs. logic block LUT size question that shows that small LUT sizes, with the use of carefully optimized electrical design and single-driver architectures, have better area (relative to 4-LUTs) than previously thought. Another experiment shows that several of the previous architectural results are invariant in moving from multi-driver to single-driver routing architecture and across a range of process technologies.

Categories and Subject Descriptors: B.7 [Integrated Circuits]: Types and Design Styles

General Terms: Algorithms, Experimentation

Keywords: FPGA, Architecture, CAD

FPGA'09, February 22–24, 2008, Monterey, California, USA.

1. INTRODUCTION

Field-programmable gate arrays (FPGAs) have evolved dramatically over the past ten years, as they have taken advantage of new process technologies and architectural innovations. The VPR toolset [6, 7] has been widely used as the core CAD tool in FPGA architectural exploration and CAD algorithm research over that time. However, centralized development of the tool largely stopped in the year 2000 with the release of Version 4.30. Multiple research efforts have made changes to VPR to investigate various aspects of architecture, including alternative routing architectures [16], hard-wired corners [25], power issues [23] and modifications to logic clusters [17] among many others. VPR also formed the basis for a commercial architecture exploration environment [18]. These efforts all produced useful research results but the modified versions of VPR used to produce the results either are not publicly available or lacked the architectural flexibility or robustness required in a broadly useful research tool (with the notable exception of [23] which was released as a simple-to-apply patch to the VPR 4.30 release). In this paper, we describe four significant new features of a new release of VPACK/VPR that are major steps towards creating a modern research environment for FPGA CAD and architecture.

One particular issue has arisen due to the lack of a robust architectural exploration tool that can model heterogeneous hard blocks such as memories and multipliers. Since almost all large-scale modern designs contain significant quantities of hard block memories, the modern academic architect is limited to experimenting with very small (mid-1990's size) benchmark circuits that are increasingly unrepresentative of the current use of FPGAs. This limits the ability of academic research to explore the architectural issues facing the next generations of FPGAs.

To address these limitations, we have a developed a new version of VPR, Version 5.0, which is publicly available at `http://www.eecg.utoronto.ca/vpr`. This release has four key features:

1. *Single Driver* Routing Architectures. The most common method for programmably interconnecting FPGA routing tracks is to employ a single driver for every distinct track [29, 4], as opposed to the multiple tri-state or pass transistor drivers used in previous generations of FPGAs. The driver is itself driven by a multiplexer. This type of routing architecture requires a rather different routing architecture generator, which

has to optimize for different kinds of constraints. VPR 5.0 supports this method of routing while continuing to provide support for the legacy methods.

2. *Heterogeneous Logic Blocks.* Modern FPGAs all contain blocks such as multipliers [29, 4, 3, 28] and memories [29, 4, 3, 28] that complement the conventional lookup table (LUT)-based logic blocks, as discussed above. VPR 5.0 can now support the placement and routing of an arbitrary number of different types of blocks combined on the FPGA.

3. *Optimized Circuit Design in released Architecture Files.* To obtain meaningful results from VPR, accurate area and delay measurements of the target FPGA are required. When using prior versions of VPR, a user was required to design their own specific transistor-level circuits to generate their own area, delay, resistance and capacitance parameters relating to an architecture. This can be a difficult and time-consuming process. VPR 5.0 is released with a large suite of architecture files with many different soft logic architectures and optimized transistor-level design targeting a range of area and delay trade-offs from each. These files *also* span across a wide range of process technologies, currently ranging from 180 nm CMOS down to 22 nm CMOS based on the Predictive Technology Models from [31].

4. *Robustness.* VPR became an effective tool in part because of its broad applicability across many architectures, and because it was high-quality software with an easy-to-understand software architecture. It compiles without difficulty on many platforms (and was even used as part of the SPEC CPU2000 benchmark). The aim in this new version is to maintain this level of high quality design, both through careful software engineering and the inclusion of a suite of regression tests that permit short-turnaround software checking, and longer-turnaround software coverage and quality of result checking.

The remainder of this paper is organized as follows: Section 2 provides background on FPGA architecture and VPR itself. The new features implemented in VPR and the issues encountered are described in Section 3. Then, in Section 4, we illustrate the capabilities of the new VPR by exploring two architectural issues enabled by the new features: LUT size with fully optimized circuit design and single-driver routing, as well as the effect of process scaling on segment track length. Finally, conclusions and future work are summarized in Section 5.

2. BACKGROUND AND TERMINOLOGY

The VPR toolset was designed to enable research in CAD and architecture for FPGAs and one of its key features is that it can perform timing-driven packing, placement, and routing (and timing analysis) for a wide range of different FPGA architectures that are described in a human-readable architecture file. In this section, we review the architectural parameters from the previous version of VPR that are needed to understand the new features we present in this paper. We also review the basic VPR-based CAD flow.

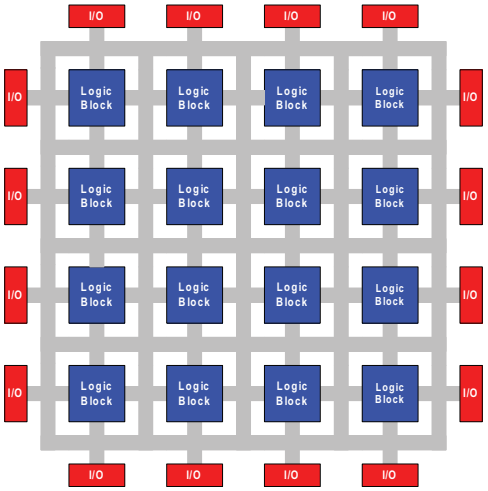

Figure 1: Island Style FPGA

2.1 FPGA Architecture Parameters

VPR [6, 7] was designed to target island-style FPGAs such as that illustrated in Figure 1. These FPGAs contain I/O blocks and logic blocks surrounded by programmable routing. As the logic blocks are all assumed to be identical, a single logic block and its adjacent routing can be combined to form a *tile* that can be replicated to create the full FPGA. The basic logic block and routing architecture parameters are reviewed in the following sections.

2.1.1 Logic Block Architecture

VPR 4.30 modelled a homogeneous array of logic blocks in a two-level hierarchy; the first level consisted of a Basic Logic Element (BLE) that could implement a combinational logic function and a flip-flop. The combinational logic function has commonly been a LUT, but the tool can model any function. The BLE description in VPR is parameterizable to have different numbers of inputs, **K**. The second level of the hierarchy is formed by groups of **N** BLEs, known as a logic *cluster*, as shown in Figure 2. Although there are a total of $K \cdot N$ inputs and N outputs inside the cluster, the cluster only presents I inputs and N outputs to the external-to-the-cluster routing, where $I \leq K \cdot N$. VPR 4.30 assumes that all I inputs to the cluster and N outputs can be routed internally to all of the $K \cdot N$ inputs of the BLEs in what is known as a *fully-connected* cluster. Although commercial FPGAs and [17] use clusters that are less than fully connected, both VPR 4.3 and 5.0 continue to make the fully-connected assumption. This internal-to-the-logic-cluster connectivity is known as *intra-cluster* routing, to distinguish it from the connections between the clusters, called *inter-cluster* routing, which is described below.

2.1.2 Routing Architecture

The inter-cluster routing network employed in VPR consists of routing *channels* between the logic blocks as illustrated in the thick lines in Figure 1. Each channel is composed of individual routing *tracks* which consist of wires and programmable switches. The channel *width*, **W**, is the number of tracks in each channel. Programmable connections between routing tracks are made within *switch blocks* that can be found at the intersection of routing channels. The

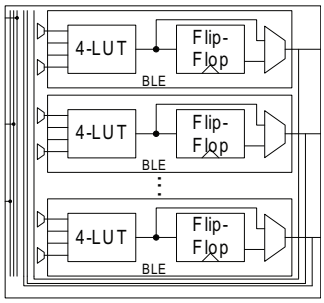

Figure 2: Logic Cluster

number of programmable connections incident upon a track in a switch block is typically defined as the flexibility of the switch block, and is referred to as the parameter $\mathbf{F_s}$. Connections between the clusters and routing channels are formed in connection blocks; the fraction of tracks that an input is connected to is referred to as $\mathbf{F_{c_{in}}}$, and for the outputs as $\mathbf{F_{c_{out}}}$.

A routing track typically consists of *segments* which travel a distance $\mathbf{L}$ in logic clusters before being interrupted by a programmable switch.

The routing tracks can also be programmably connected to logic blocks or I/O blocks and the topology of these connections and the connections between tracks has a significant effect on the FPGA's area and performance.

The previous VPR routing architecture generator assumes that each routing track can be driven by multiple possible sources, which we will refer to as *multi-driver routing*. These multiple drivers are programmably connected to the track either through pass transistors or tristate buffers, only one of which is active during a given FPGA configuration.

2.2 Architecture File Description

In the previous version of VPR, a textual file description of the target FPGA is provided in a human-readable architecture file. That file provides all of the parameters described above, and several more. It also describes the delays, either using absolute delays or resistance and capacitance estimates, of the various components of the FPGA. These delays must be determined through knowledge of the IC process employed by the target FPGA, and through extensive circuit design to choose appropriate transistor sizes. While our research group had access to proprietary IC processes provided by the Canadian Microelectronics Corporation, we were unable to publish realistic architecture files with delays, resistances and capacitances without violating non-disclosure requirements. This meant that there have been no publicly-available realistic architecture/electrical descriptions of VPR-based FPGAs. Furthermore, the amount of work required to design electrically realistic FPGA designs is extensive. The new release of VPR includes architecture files that have circumvented these issues.

2.3 VPR-based CAD Flow

The experimental process used in typical VPR-based CAD and architecture research is illustrated in Figure 3. This flow takes a benchmark design and an FPGA architecture with electrical design specifications and "implements" the design on the specified FPGA. The timing analyzer provides a measure of the critical path delay of the implemented circuit, and

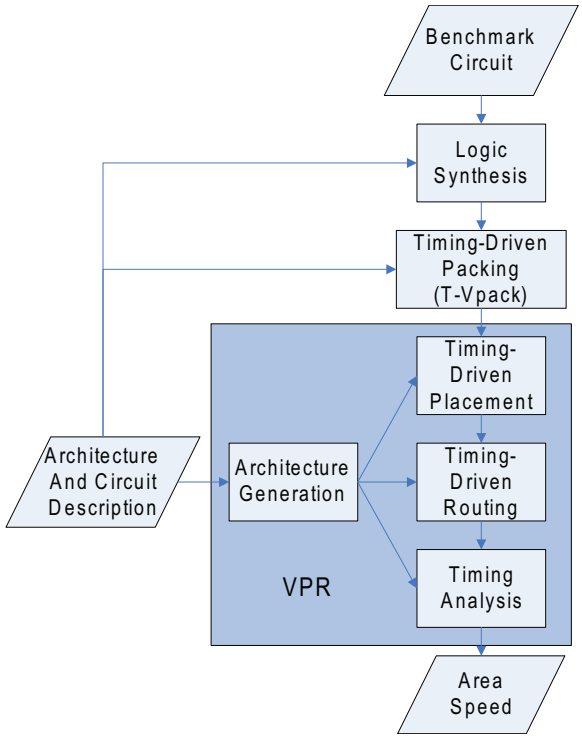

Figure 3: VPR CAD Flow

area models associated with VPR provide an approximation of the area taken up by that circuit in that FPGA.

In the first step of the flow, the benchmark circuit is synthesized and technology mapped, typically using a combination of tools such as SIS [24], FlowMap [8] or ABC [21], all of which use BLIF as the input and output format. The output after technology mapping must then be packed into the logic clusters available on the FPGA. This is done using T-VPack [19, 7], which performs a timing-driven packing. Once packing is complete, VPR is then used to perform placement and routing for the circuit. Finally, timing analysis is completed to determine the performance of the circuit on the FPGA.

A key step in the figure is called *architecture generation*, which takes the human-readable architecture file parameters, and generates the specific structures and connections for the entire FPGA. This is a difficult process that has to meet many simultaneous constraints as described in [7].

The most difficult part of architecture generation is the creation of the *routing resource graph* which contains the precise details of every wire and switch in the architecture. The timing-driven placement algorithm uses the graph to extract timing information needed during placement, and the router directly uses the graph during routing both to determine the available connections and the delay of connections.

VPR was designed so that new ideas in routing architectures (those not presented as possibilities by the current architecture file description) can be explored by making changes only to the architecture generator without having to change the placement tool, the router or the timing analyzer. However, the scope of the changes required for the features of this new version necessitated significant changes to the entire tool that we will describe in the following section.

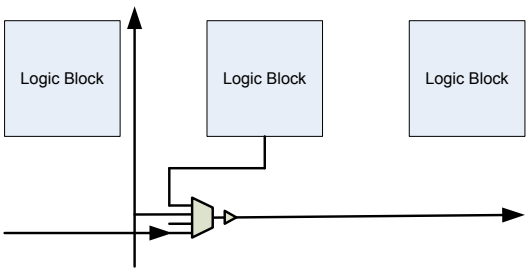

Figure 4: Single Driver Routing

3. NEW FEATURES

In this section we describe the four key new features of VPR: single-driver routing, heterogeneity modelling, a wide set of electrically optimized architecture files, and a regression test suite.

3.1 Single-driver Routing Architecture

In *single driver routing* each routing track has only one physical driver and programmability is achieved by using a multiplexer on the input of the driver as illustrated in Figure 4.

The work in [18, 16] has shown that the single driver approach first employed by [30] *dominates* the multi-driver approach, in the architectural sense, because it provides superior performance with less area. All of the recent devices from the two largest FPGA vendors use single-driver routing architectures exclusively [4, 29]. VPR 5.0 now supports a wide range of single-driver routing architectures (many more than described in [16] in addition to the multi-driver routing provided by VPR 4.30.

This restriction to a single driver means that logic block outputs can only connect to the routing tracks whose drivers are adjacent to the logic block, to avoid overly-long wires to neighbouring logic clusters. Typical single-driver routing architectures place the driver at one end of a wire rather than, say, the middle, which would likely add unusable loading to the wire. This implies that each wire can only send signals in a specific direction (right, left, up or down), and so there must be two kinds of tracks in each channel: left (or up)-driving and right (or down)-driving, a concept also known as *uni-directional* routing. Despite this restriction, prior work [18, 16] has shown that the total number of tracks required per channel is only slightly more than needed in bi-directional routing.

The routing architecture generator produces tracks in pairs, one in each direction. The new single-driver routing architecture generator is faced with the following constrained graph generation problem: it must create a detailed routing architecture with W tracks per channel, where a fraction of the channel F_i should consist of tracks of length L_i (where length is measured in number of clusters traversed without switching), each track can connect to F_s other tracks, each logic cluster input pin can connect to $F_{c_{in}}$ tracks and each cluster output can connect to $F_{c_{out}}$ tracks. These programmable connections must all be implemented by feeding the pins and tracks to an appropriate multiplexer connected to the single-drivers. An additional issue that must be faced is that each of the multiplexers should be approximately the same size, a constraint we call the *mux balancing* criterion.

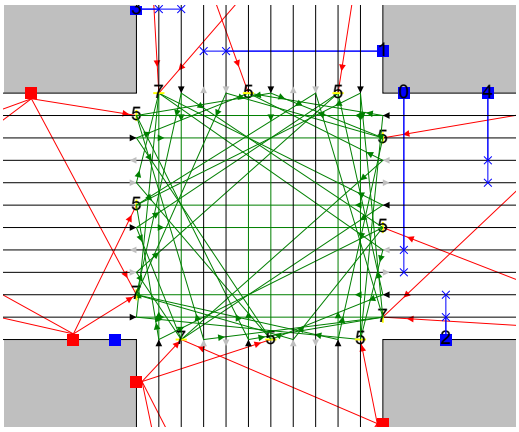

Figure 5: Example Single-Driver Switch Block

We constrain the multiplexer sizes to differ by no more than 2 inputs. In addition, the start point of each wire in a track is staggered in the usual way [7]. The generation problem is somewhat more constrained than the multi-driver case because there are fewer drivers for tracks and output pins to connect to at each (X,Y) grid location. All of this, combined with the goal of correctly generating an architecture for a wide range of routing architecture parameters makes this generation problem difficult.

The generation algorithm roughly proceeds in the following way: connections from the logic block outputs to the drivers are made first, followed by the track-to-track connections. The logic block output pins connections are created in pairs (one for each direction in the uni-directional system) connecting to a starting wire of each of the two adjacent switch blocks, which drive in opposite directions. This proceeds in a round-robin until either the $F_{c_{out}}$ specification is met or all the available starting wires are used by that pin. Each subsequent output pin continues the round-robin from the tracks used by the previous output pin. Next, the track-to-track connections are made to the single-drivers, using the Wilton switch block pattern [26] for any starting and ending wires. All other wires are connected to the multiplexers in a round-robin fashion.

Figure 5 illustrates a single-driver routing switch block produced by VPR 5.0, with $F_s = 3$, $F_{c_{in}} = 0.25$, and $F_{c_{out}}$ = full (where full means connect to all possible local drivers; this is typically less than all of the adjacent tracks). If the figure is viewed in colour, the red lines are output pin connections into the driver, the blue are input connection block connections, and the green lines are track-to-track connections.

The routing algorithm used in VPR 5.0 to route a circuit on the single-driver detailed routing architecture is the same as that used in VPR 4.30, and is a timing-driven advancement on the PathFinder algorithm [20].

As a demonstration of the capabilities of the single-driver and multi-driver routing architecture generation capabilities, Figure 6 gives the single-driver and multi-driver minimum routable track counts (W) for a set of standard benchmark circuits implemented on an FPGA with clusters of 10 4-input LUTs, with $F_{c_{in}} = 0.25$, and $F_{c_{out}}$ = full, and segments of length L=1. The average increase in track count

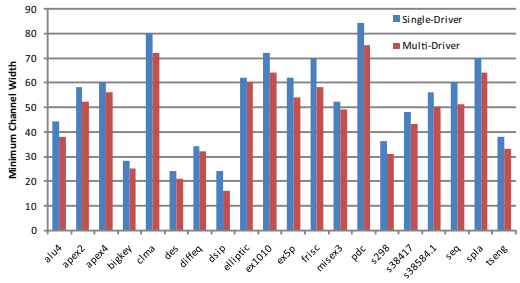

Figure 6: Single- vs. Multiple-Driver Track Count

of the single driver architecture over the multiple driver is 14%, consistent with the results published in [18].

3.2 Heterogeneity

It is now common for there to be a heterogeneous set of logic blocks on an FPGA. In addition to the basic soft logic cluster, the additional blocks are often "hard" circuit structures that are designed to perform specific operations and, for this reason, these heterogeneous blocks are commonly called *hard blocks*. The differentiated blocks could also be a different kind of soft logic cluster that gives better performance or saves area [10, 9].

Commercial FPGAs commonly include hard memories and multipliers, and there are many other possibilities that could be considered for inclusion as hard blocks, such as crossbars [13] and floating-point multipliers [5]. The specific selection of which blocks to make hard and include in an FPGA is one of the central questions in FPGA architecture. These blocks can offer significant benefits when used but, if unused, they are wasted. VPR 5.0 now supports the use of heterogeneous logic blocks.

The basic X-Y coordinate system in VPR 5.0 is dictated by the soft logic cluster tile: one grid unit in the X and Y directions is designated as the space taken up by the basic soft logic cluster. All other blocks must be multiples of this size. Hard blocks are restricted to be in one grid width column, and that column can be composed of only one type of block. Although this restriction prevents a more general cross-column approach, it appears sufficient for all but the extremely large hard blocks. Each hard block may be broken in a different number of sub-blocks, not unlike the logic elements in a cluster. Each type of block may have different timing characteristics, routing connectivity, and height. The height of a block must be an integral number of grid units. In the event that a block's height is indivisible with the height of the core, some grid locations are left empty.

The routing architecture is *transparent*; this means that if a hard block spans multiple rows, the horizontal routing tracks pass through at every grid location, but there are no input or output pins from the block where the routing passes through.

VPR 5.0 models logic blocks, heterogeneous blocks, and I/Os using the same data structure. This differs from the previous version which modelled logic blocks and I/Os with separate data structures. This new way permits an arbitrary number of different types of blocks to be modelled, and actually simplified the code.

To make it easier to identify different blocks, colour was added to all types of blocks in the core of the FPGA for

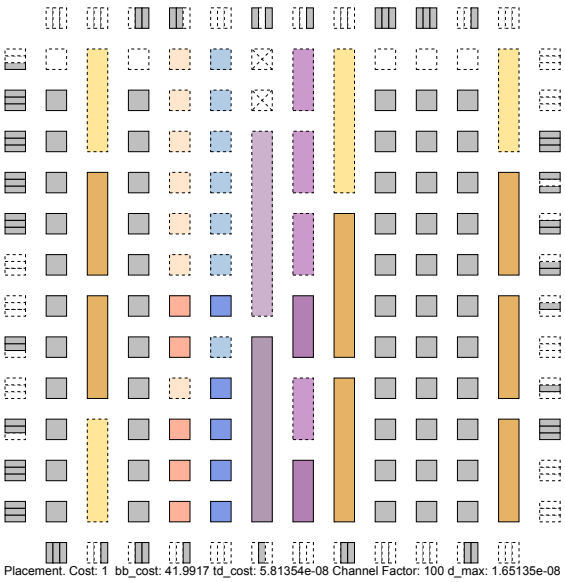

Figure 7: A Heterogeneous FPGA in VPR 5.0

up to 6 different colours. If there are more than 6 types of blocks in the core, subsequent types use the same colour as the 6th type. An unused block location will have a lighter colour than a used block location. Figure 7 illustrates a heterogeneous FPGA generated by VPR 5.0 with 8 different kinds of blocks.

The timing parameters of all blocks are now specified using a *timing matrix*. The timing in a subblock is modelled as a complete set of all possible delays from each input to each output of the subblock. This is an improvement on the previous version which allowed different delays for different subblocks but each subblock can only have one single delay. Heterogeneous subblocks can have purely combinational or registered output.

VPR 5.0 uses an XML-based architecture file format so as to leverage the convenient modelling hierarchy in XML [27].

These additions give VPR the capability to implement circuits with hard blocks. Although the support for hard blocks in the flow upstream to VPR is not very sophisticated, we have succeeded in synthesizing circuits with hard multipliers from Verilog (through the ODIN tool [12]) through synthesis, packing, placement and routing on an FPGA with 18 x 18 multipliers and clustered logic blocks (CLBs) of 10 4-input LUTs, with $F_{c_{in}} = 0.25$, and $F_{c_{out}} =$ full, $F_s = 3$, and segments of length L=1. This was done for a number of circuits obtained from OpenCores (`http://www.opencores.org`) and internal University of Toronto projects (available at `http://www.eecg.utoronto.ca/~jayar/benchmarks/bench.html`. Table 1 gives the minimum routable track counts (W) for these circuits, as a demonstration of the basic heterogeneous capability.

3.3 Electrical Optimized Architecture Files

In order to provide accurate area and delay results for an architecture, VPR requires that the delay, resistance, capacitance, and area of various circuit elements of that architecture be specified in an architecture file. When an industrial FPGA is designed, circuit designers will spend months care-

Table 1: Track Counts for Circuits with Multipliers

Circuit	CLBs	18x18 Multipliers	W
diffeq0	208	4	34
diffeq1	79	5	24
fir_scu_rtl	70	17	28
rs_decoder_1	144	13	28
rs_decoder_2	265	9	52
cf_fir_3_8_8	22	4	22
cf_fir_24_16_16	366	25	42
oc54_cpu	301	1	42
iir1	70	5	26
iir	46	5	30
Stereo Vision	1543	152	68
Ray Tracer	341	18	42

fully tuning and trading off aspects of the circuit design. It is clearly not possible to expend such manual effort for every potential architecture file. Furthermore, for any given *logical* architecture as specified by the list of parameters given above, there could be many different electrical designs with different area and speed trade-offs. A major feature of this new release of VPR is the inclusion of a large suite of logical architectures files, with optimized circuit design targeting a range of area and delay trade-offs for each logical architecture. The electrical design of these logical architecture files was produced by the automated transistor sizing tool and methodology described in [15]. In addition to this, we have leveraged the Predictive Technology Models described in [31] to produce architecture files describing optimized FPGAs in a wide range of IC process technologies, from 180 nm CMOS down to 22 nm CMOS. Architecture files were not released in the past since they would reveal information obtained under non-disclosure agreements but, with these predicted models, such issues are avoided.

There are three primary inputs to the automated electrical design tool: the logical architecture describing the high-level structure of the FPGA including parameters such as cluster size, LUT size, and routing flexibility. The tool can only handle FPGAs built using cluster-based logic blocks (delays and areas for hard blocks must be estimated separately by the user). The second input must describe the process technology to determine the characteristics of the transistors that will be used to implement the architecture. Finally, an optimization objective must be supplied to determine to what extent the transistors will be sized for area or delay.

The sizing tool uses these three inputs to produce a transistor sizing specific for the given architecture and IC process, optimized with respect to the optimization objective. From the final electrical design determined by the tool, the area and delay measurements needed for the architecture file can be generated. In the past, optimizing the FPGA transistor sizes for each architecture was not feasible and, instead, fixed designs were used for a broad range of parameters. With the more thorough optimization that is possible with this new optimizer, more realistic area and delay measurements can be made which improves the quality of the final area and delay measurements from VPR. The architecture files are available for download from the VPR website, and can be found directly at: `http://www.eecg.utoronto.ca/vpr/architectures`. Table 2 gives a listing of the range

Table 2: Range of Parameters Included in Architecture Listing

Parameter	Range/Values Considered
Cluster Sizes	2 – 12
LUT Sizes	2 – 7
Track Length	1 – 8
Channel Width	Up to 3 Values
$F_{c_{in}}$	Up to 3 Values
$F_{c_{out}}$	Up to 3 Values
Cluster Inputs	Up to 3 Values
IC Processes	22 nm – 180 nm
Optimization Objectives	$Area^{10}Delay$, AreaDelay, Delay

Table 3: Regression test suites

Test	Test Description
checkin reg tests	Fast tests and high code coverage
arch sweep	Randomly generated architectures
mult exp	Designs with multipliers
N K sweep	Comprehensive LUT and cluster sweep
options sweep	All VPR options
QoR	Quality of results

of logical architectures, IC processes, and optimization objectives that can be found in that website.

3.4 Robustness and Regression Tests

A key goal of the new release is to maintain two key properties of robustness from the original VPR: to be able to work reliably across a wide-range of logical architectures, and to be high quality software. We created a system and suite of regression tests that gives high architecture coverage and quality coverage. We note that while many researchers have modified VPR for point experiments, it is far more difficult to keep these two properties intact. The regression tests will also aid future developers modifying VPR 5.0 by enabling them to test their implementations against known results. One major challenge with developing a regression test system is that the the exact results of the algorithms used in VPR are often sensitive to small perturbations in the inputs or order of computation. A naive approach of comparing numerical metrics for equality results in many failed tests for valid results. To overcome this problem, the regression tests specify a range, that the user may adjust, for each quantity and for each test in order to judge if a difference is due to an error or experimental variation. A second major challenge with the regression tests is the large test space to cover. The goal was to test critical points in this test space in as little time as possible. To achieve this goal, different test suites were created to allow the user to choose the coverage and runtime trade-offs to test. Table 3 gives an overview of the test suites available in the regression test system.

4. EXPERIMENTS USING NEW FEATURES

This new version of VPR will enable the exploration of a number of architectural issues, two of which will be examined in this section. The first will be to revisit the effect of the logic block functionality on the performance and area of an FPGA, with the use of single-driver routing and electrically optimized circuits. The second will be to explore the

effect of process technology scaling on FPGAs with different routing segment lengths (L) and LUT size, again with single-driver routing.

4.1 Experimental Methodology

We will employ the standard empirical methodology in which benchmark circuits are implemented on each target FPGA (using the CAD flow described in Section 2.3) and the area and performance of each benchmark is measured.

A number of architectural parameters will be varied in the following experiments. In all cases, the number of inputs to the logic cluster (I) was set based on the LUT size (K) and cluster size (N) using the formula $I = K/2 * (N + 1)$ from [2] to ensure that 98 % of the BLEs can be used. The routing channel width (W) will be set to be 20 % larger than the minimum channel width required to route each of the benchmark circuits. Since we are focusing only on clustered logic block architectures (and not employing the new heterogeneity feature) we will use the 20 largest MCNC benchmark circuits. Each of these circuits was technology mapped by SIS with FlowMap [8], clustered with T-VPack [19] and then placed and routed with the new version of VPR. Placement and routing is repeated with ten different placement seeds and only the results from the fastest design will be used.

We have used the new architecture files which include timing and area information required to produce accurate area and performance results. For the experiments described here, we will restrict our attention to FPGAs electrically optimized for minimum area-delay product. These are the architecture files labelled, under the "objective" column in the archive (described in Section 3.3). as $area \cdot delay$.

The area metric reported here is the usual minimum-width transistor area-based model described in [14], which is based on that in [7]. The critical path delay of each circuit is measured by VPR's timing analysis engine.

The final area and delay measurements reported for each FPGA design will be the ensemble results across all 20 benchmarks. The area is reported as the total area required for the twenty circuits and delay is reported as the geometric mean critical path delay for the twenty benchmark circuits.

4.2 Effect of LUT Size with Single-Driver Routing and Fully Optimized Circuits

In this section, we revisit (for the third time) the effect of FPGA logic block LUT size on area and performance, as was done in [2], but now in the context of single-driver routing and electrically optimized circuits. Although [2] optimized routing buffer sizes (to account for track length differences with different logic blocks) our new architecture files optimize *every* transistor, including logic and intra-cluster routing transistors, which the previous work did not. We look at FPGA architectures with LUT size ranging from 2 to 7 and clusters size ranging 2 to 12. For this work, single-driver length four (L=4) routing segments were used and the target process technology was 90 nm CMOS. The methodology described above was used with the benchmark circuits and the resulting area measurements for each architecture are plotted in Figure 8. In this figure, each cluster size is plotted as a different curve while the LUT size is varied along the X-axis. The results for the different cluster sizes are all similar which mirrors past observations [2] for multi-driver routing.

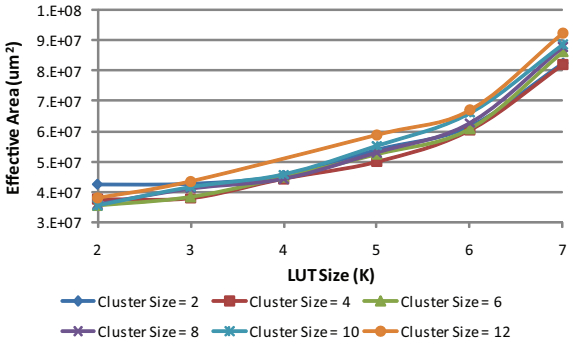

Figure 8: Area vs. Cluster and LUT Size

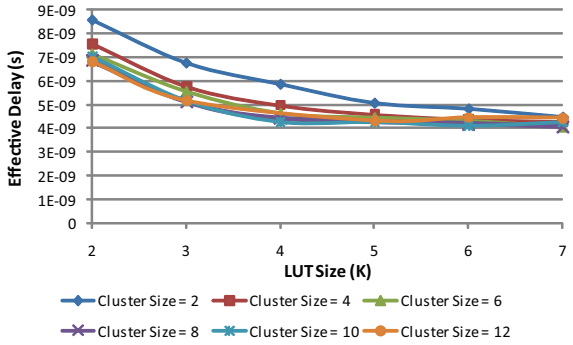

Figure 9: Delay vs. Cluster and LUT Size

As well, the same general trend seen in [2] of increasing area required for LUTs larger than size four can be seen in the figure. However, unlike in [2], the FPGA area *decreases* as the LUT size is reduced below four, whereas [2] has it increasing. Since area is measured as the number of clusters required multiplied by the area of these clusters (including the routing), this means that the reduced area for clusters with smaller LUT sizes more than offsets the increase in the number of clusters required as the LUT size decreases. In the prior work, the opposite behaviour was observed as the increased number of clusters required dominated the area results.

There are a number of potential reasons for these differences and the use of single-driver routing may be one contributing factor. The overall area was observed to increase for the smallest LUT sizes in past studies due to the increased area required for inter-cluster routing [2]. We observe this increase as well but the increase is not as large. In [2], as the LUT size increased from 2 to 7 the total routing area decreased by over 40 % but, for the same change in LUT size, we observe a decrease in routing area of only 20 to 25 %. This smaller change in the routing area means the penalty of the smaller LUT sizes is not as significant and the area savings within the cluster of the smaller LUT sizes is able to compensate. Since single-driver routing reduces the area required for routing tracks, this may be one reason for the difference. However, it is also possible, that the fully optimized (for area-delay product) electrical designs, which may have reduced transistor sizes for the smaller LUT sizes, also contributed to this difference.

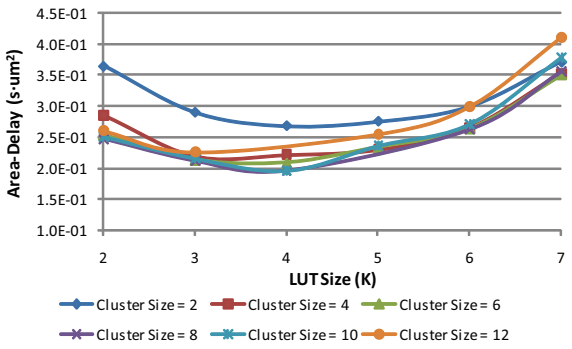

Figure 10: Area-Delay vs Cluster and LUT Size

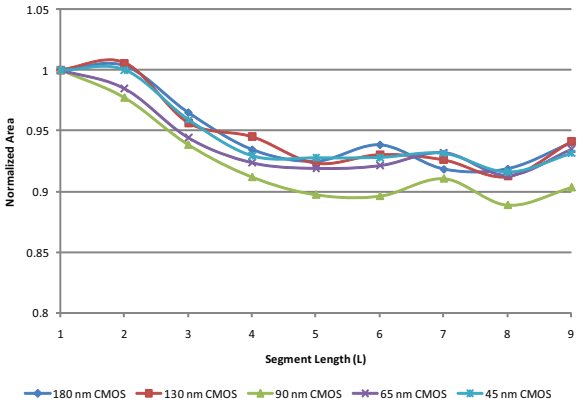

Figure 11: Area vs. Segment Length/Technology

The average delay results are plotted for these architectures in Figure 9. It can be seen that increasing the LUT size improves the performance of the FPGA but the improvements begin to diminish beyond LUTs of size four. It appears that increasing the cluster size typically offers better performance. Both these trends match the general trends seen in multi-driver routing [2].

The delay results can be combined with the area results and the resulting area-delay product is plotted in Figure 10. In this figure, one can see the 4-LUT architectures yield the lowest area-delay designs but, in general, the results are very similar for 3-, 4-, and 5-LUTs.

These results indicate that changes to the logic block architecture are not necessary to take full advantage of single-driver routing architecture since many of the conclusions reached in past works continue to apply. In the next section, we examine whether process technology scaling should lead to any architectural changes.

4.3 Process Technology Scaling

The increased integration enabled by shrinking process technologies enabled FPGAs to leverage the dramatic scaling of Moore's law. However, the shift to smaller technologies is also accompanied by new challenges, including those related to interconnect scaling. In smaller process technologies, the delay of a constant-length wire increases. This increase is partially offset by the ability to use shorter wires, since logic also shrinks due to smaller transistors and, therefore, a shorter wire can still reach the same number of devices. However, looking at the predictions from the International Technology Roadmap for Semiconductors (ITRS), the delay of local and intermediate-length wires is expected to increase [1]. (This has long been recognized as a problem for global wires as their dimensions often do not shrink with each process node [11].) This has potential implications for FPGAs as the routing, including the wiring, is a significant fraction of the delay and area of an FPGA. The delay of the routing may increase as a result; alternatively, area needed for routing may increase as buffers are added and sized to ameliorate the delay issues.

Given the potential for wiring delay to alter the delay and area of the routing, we sought to investigate the potential impact of process scaling on a number of architectural parameters. We explored the effect of new process technologies on routing segment length and LUT size. Routing segment length is interesting to consider because increased wire delays should have the greatest effect on these parameters. LUT size is worth investigating as it has the strongest influence on the cluster areas and shifts in sizing may alter the best LUT size for a logic block.

The effect on segment length was first explored by considering segment lengths of between 1 and 9 clusters for technologies ranging from 180 nm down to 45 nm CMOS. In all cases, 100 % single-driver routing was assumed and 4-LUT logic clusters of size 10 were used for the logic blocks. When measuring delays, the wires in the inter-cluster routing and intra-cluster routing are modelled as having resistance and capacitances that are based on their physical length (which is set by the wire's logical length and the area required for the logic cluster and routing) and process technology. These wires are assumed to be implemented in an intermediate layer of metal and the properties of these wires were set based on the predictions from the ITRS [1]. In the delay simulations, the transistors are modelled using the Predictive Technology Models from [31], as represented in the newly-released architecture files.

The area and delay for each architecture was then determined using the standard experimental process described above. Figure 11 gives the area (normalized to the architecture with segment length 1) versus segment length with each curve showing a different process technology. One can observe that the same general trends hold in all technologies. Segments of length 1 and 2 have significant area overhead; but for longer segment lengths, the area decreases until approximately length 9 where area begins to increase again. These results are consistent with past observations made for multi-driver architectures [7].

Figure 12 gives the delay (normalized in each technology to the segment length 1 architecture) versus segment length with each curve showing a different process technology. This experiment offers finer resolution on segment length than that in [7] and shows that there is a wide region in which segment length does not affect FPGA speed when segments are greater than length three. Most interestingly, the results do not appear to be significantly affected by the different process technologies.

The impact of process scaling on LUT size was also examined. For this investigation, single-driver length 4 routing segments were assumed and logic clusters of size 10 were used. LUTs ranging in size from 2 to 7 were examined in

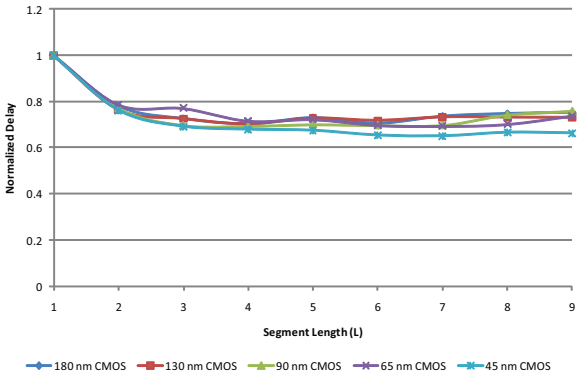

Figure 12: Delay vs. Segment Length/Technology

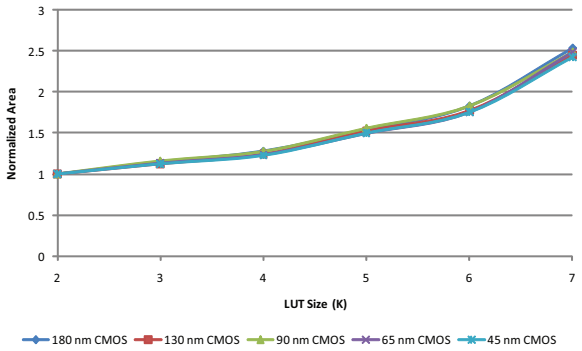

Figure 13: Area vs. LUT Size/Technology

technologies ranging from 180 nm to 45 nm CMOS using the Predictive Technology models [31]. The area and delay results for these architectures are plotted in Figures 13 and 14 respectively. In both figures, the different processes are plotted in different colours as indicated in the legend. For each technology, the results are normalized to the area or delay of the 2-LUT architecture.

The area results are all consistent between the technologies with increasing LUT size leading to increased area. Similarly, for the delay, the trends are similar for all technologies and the delay improves with increased LUT size until 6-LUTs beyond which there is only a minimal change to the FPGA's performance. Clearly, it appears that despite the challenges of new process technologies, their effect on architectural parameters is minimal. It is likely, however, that if other issues such as power consumption are considered then architectural adjustments may be needed.

5. CONCLUSIONS AND FUTURE WORK

This paper has described four new features of the VPR tool suite for FPGA CAD and architecture research, including single-driver routing architectures, heterogeneity, electrically optimized circuit and architecture files and software regression tests. We have illustrated the new tool's use by performing a number of experiments showing each of these features.

VPR 5.0 and the associated architecture files can be downloaded at `http://www.eecg.utoronto.ca/vpr`.

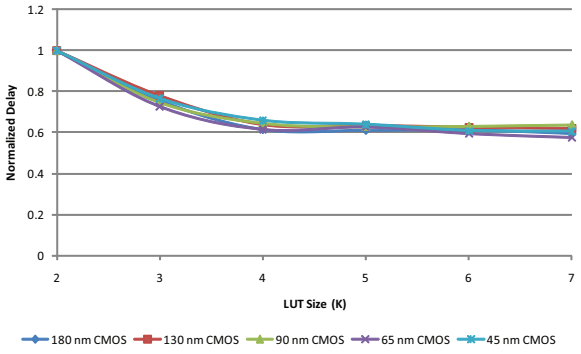

Figure 14: Delay vs. LUT Size/Technology

5.1 Future Work

While this work is an important step in the development of publicly available computer-aided design (CAD) tools that can support modern FPGA architectures, there is much work that remains. A significant limitation currently is the lack of support for heterogeneous blocks in the upstream tools in the CAD flow, from Register Transfer Level (RTL) design to packing. The synthesis, technology mapping and clustering support for hard blocks is limited to treating any hard functionality as black boxes. Future improved upstream tools must be flexible enough to permit a wide range of heterogeneous blocks; we suggest that they should all connect to the same architecture description file for consistency of specification. As well, given the close relationship between clustering and placement, integration of the currently separate tools that perform these tasks (T-VPack and VPR) would simplify the handling of hard blocks in the future and work on this integration is already underway. It will also be important to include power measurement and optimization capabilities, such as those provided by [23] in previous versions. Finally, additional work is needed to add support for features such as arithmetic carry chains and depopulation of the intra-cluster routing that are now common in commercial FPGAs. The challenge is to add this functionality with sufficient flexibility to enable architectural exploration of these features.

Acknowledgements

This work has benefited from the contributions of many others. Daniele Paladino [22], Russ Tessier, Andrew Ling and Mark Jarvin, and Steve Wilton were involved in the development of versions of VPR that supported heterogeneous blocks and their experiences provided guidance for the implementation used in VPR 5.0. The authors are also grateful to Vaughn Betz, the original author of VPR.

6. REFERENCES

[1] International Technology Roadmap for Semiconductors 2007 Edition, December 2007. `http://www.itrs.net/reports.html`.

[2] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(3):288–298, March 2004.

[3] Altera Corporation. Cyclone III device handbook, Sept 2007. ver. CIII5V1-1.2 http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf.

[4] Altera Corporation. Stratix IV device handbook version SIV5V1-1.1, July 2008. http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf.

[5] M. J. Beauchamp et al. Embedded floating-point units in FPGAs. In *FPGA '06: Proceedings of the 14th international ACM/SIGDA symposium on Field programmable gate arrays*, pages 12–20, New York, NY, USA, 2006. ACM.

[6] V. Betz and J. Rose. VPR: A new packing, placement and routing tool for FPGA research. In *Seventh International Workshop on Field-Programmable Logic and Applications*, 1997.

[7] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[8] J. Cong and Y. Ding. FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):1–12, Jan 1994.

[9] J. Cong and S. Xu. Delay-optimal technology mapping for FPGAs with heterogeneous LUTs. In *Design Automation Conference, 1998. Proceedings*, pages 704–707, 1998.

[10] J. He and J. Rose. Advantages of heterogeneous logic block architectures for FPGAs. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 7.4.1 – 7.4.5, May 1993.

[11] R. Ho, K. Mai, and M. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, Apr 2001.

[12] P. Jamieson and J. Rose. A verilog RTL synthesis tool for heterogeneous FPGAs. In *International Conference on Field Programmable Logic and Applications, 2005*, pages 305–310, 2005.

[13] P. Jamieson and J. Rose. Architecting hard crossbars on FPGAs and increasing their area efficiency with shadow clusters. In *International Conference on Field-Programmable Technology, 2007*, pages 57–64, 2007.

[14] I. Kuon and J. Rose. Area and delay trade-offs in the circuit and architecture design of FPGAs. In *FPGA '08: Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, pages 149–158, New York, NY, USA, 2008. ACM.

[15] I. Kuon and J. Rose. Automated transistor sizing for FPGA architecture exploration. In *DAC '08: Proceedings of the 45th annual conference on Design automation*, pages 792–795, New York, NY, USA, 2008. ACM.

[16] G. Lemieux et al. Directional and single-driver wires in FPGA interconnect. In *IEEE International Conference on Field-Programmable Technology*, pages 41–48, December 2004.

[17] G. Lemieux and D. Lewis. Using sparse crossbars within LUT clusters. In *FPGA '01: Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*, pages 59–68, New York, NY, USA, 2001. ACM.

[18] D. Lewis et al. The Stratix™ routing and logic architecture. In *FPGA '03: Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*, pages 12–20. ACM Press, 2003.

[19] A. R. Marquardt. Cluster-based architecture, timing-driven packing and timing-driven placement for FPGAs. Master's thesis, University of Toronto, 1999.

[20] L. McMurchie and C. Ebeling. Pathfinder: A negotiation-based performance-driven router for FPGAs. In *FPGA*, pages 111–117, 1995.

[21] A. Mishchenko, S. Chatterjee, and R. K. Brayton. Improvements to technology mapping for LUT-based FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):240–253, Feb 2007.

[22] D. Paladino. Academic clustering and placement tools for modern field-programmable gate array architectures. Master's thesis, University of Toronto, 2008. Available online at https://tspace.library.utoronto.ca/handle/1807/11159.

[23] K. K. W. Poon, S. J. E. Wilton, and A. Yan. A detailed power model for field-programmable gate arrays. *ACM Trans. Des. Autom. Electron. Syst.*, 10(2):279–302, 2005.

[24] E. M. Sentovich et al. SIS: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, University of California, Berkeley, Electronics Research Lab, Univ. of California, Berkeley, CA, 94720, May 1992.

[25] G. Wang et al. Statistical analysis and design of HARP routing pattern FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2088–2102, October 2006.

[26] S. J. E. Wilton. *Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memories*. PhD thesis, 1997. PhD Thesis.

[27] World Wide Web Consortium. Extensible markup language (xml), Sept 2008. http://www.w3.org/XML/.

[28] Xilinx. Spartan-3A FPGA family: Data sheet, May 2008. Ver. 1.0 http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf.

[29] Xilinx. Virtex-5 user guide, March 2008. UG190 (v4.0) http://www.xilinx.com/support/documentation/user_guides/ug190.pdf.

[30] S. P. Young, T. J. Bauer, K. Chaudhary, and S. Krishnamurthy. FPGA repeatable interconnect structure with bidirectional and unidirectional interconnect lines, Aug 1999. US Patent 5,942,913.

[31] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45 nm early design exploration. *IEEE Transactions on Electron Devices*, 53(11):2816–2823, Nov 2006. Transistor models downloaded from http://www.eas.asu.edu/~ptm/.