Foreword

This volume commemorates the twentieth anniversary of the International Symposium on Field-programmable Gate Arrays by highlighting the twenty-five most significant papers that have appeared in the conference as viewed from a 2011 perspective. These twenty-five papers represent about five percent of the total papers that have appeared in the conference. These papers have impacted industry, described key building blocks in wide use throughout industry and academia, opened areas of research, resolved serious problems, illuminated difficult issues, and illustrated innovative ways to use FPGAs. They span all areas of concern by the conference including architecture, CAD, circuits and technology, and applications and reconfigurable computing. These papers capture a significant slice of the history of FPGA research and practice, and continue to deliver insights today. They represent ``must read'' background for research in the field.

This volume includes a one-page endorsement written by an expert in the field that captures the historical context in which the paper was written and offers a retrospective view on its significance. Each endorsement contains a URL link to the original paper.

To identify the papers included in this volume, we ran a rigorous nomination and review process over the course of a full year. Over a six month period, we collected nominations through the web. Anyone could submit nominations with self-nominations prohibited. This resulted in nominations for fifty papers out of the nearly five hundred papers that have appeared in the symposium. We assembled a panel of fifteen experts (see p. iv) to make the final selection of twenty-five papers from the fifty nominations. All of the experts had previously served as chairs of the symposium. As part of the expert review, at least five area experts reread all the papers nominated in an area and shared their view of the relative significance of the papers in the area with the entire panel. This was followed by a blind ranking process in which all experts offered their own rank ordering for the papers, independent of area. These rankings were combined using a variety of preferential ranking schemes to best capture the composite expert consensus.

We would like to thank ACM SIGDA for their sponsorship of this volume, and we would also like to thank all the sponsors of the FPGA Symposium throughout its history, including ACM/SIGDA, Actel, Agere, Aldec, Altera, BEECube, Cypress, Eve, ITRC Ontario Canada, Lucent, Mentor Graphics, Microsemi, Synplicity, Trimberger Family Foundation, Vantis, and Xilinx. We also express many thanks to the numerous volunteers and authors who make the FPGA Symposium such a success.

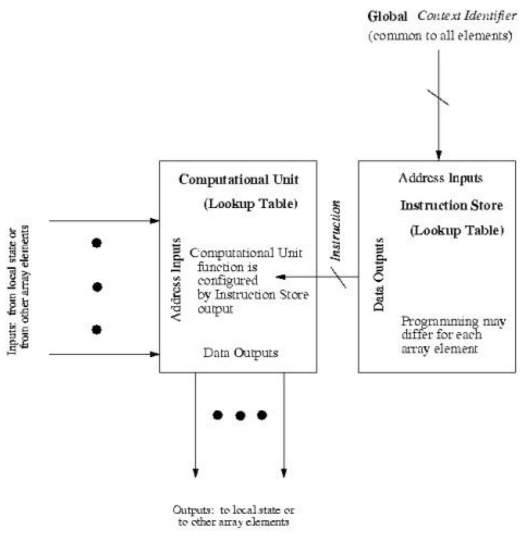
FPGA design, mapping, and usage have come a long way since 1992. Their importance and realm of application has grown. As capacities grow, mask costs increase, and deep submicron effects proliferate, their importance should continue to grow into the future. At the same time, continued integration of diverse functions, programmability, and energy issues present challenges that could limit or accelerate widespread adoption. This volume will help you understand how we got to the FPGAs we have today and illuminate the important areas of research for tomorrow. We anticipate the FPGA Symposium will continue to offer glimpses into the future of this important technology.

André DeHon
Steve Trimberger

Unifying FPGAs and SIMD Arrays

Michael Bolotski, André DeHon,
Thomas F. Knight, Jr.

Year of publication: *1994*
Area: *Architecture*


Global *Context Identifier*
(common to all elements)
Computational Unit
(Lookup Table)
Address Inputs
Computational Unit function is configured by Instruction Store output
Data Outputs
Instruction
Address Inputs
Instruction Store
(Lookup Table)
Data Outputs
Programming may differ for each array element
Inputs: from local state or from other array elements
Outputs: to local state or to other array elements

Configurable Instruction-Store View of DPGA AE

This is a unique paper that takes a philosophical view of the FPGA computing medium and relates it to the single-instruction multiple data (SIMD) approach to computation. The paper was among the first to show how the two methods of computation can be view on a continuum, thus *unifying* them in a sense.

The paper then proposes a hybrid architecture that has elements of both approaches, called a Dynamically Programmable Gate Array (DPGA). Here the configuration bitstream of the logic and routing changes rapidly, taken from a local memory made for that purpose. This is made somewhat SIMD-like by the notion that a central *context identifier*, which is broadcast throughout the array, determines which configuration is loaded from the local memory. If those local memories are all the same, then the same 'instruction' is executed; if the memories are different, then each logic processor can be doing something different. In this way, the array can both operate on data in a spatially-parallel way, and in a data-parallel way.

The authors also present an interesting analysis of the costs and benefits of organization computation in this new way.

This is the first of a series of influential papers on the DPGA, and one of the first that considers multiple *contexts* for the programming of an FPGA. While this method of programmability has yet to catch on, there have been numerous interesting attempts to make it work, and this paper was one of the first to work with it.

Jonathan Rose

DOI: http://www.seas.upenn.edu/~andre/abstracts/dpga_fpga94.html

Pathfinder: A Negotiation-Based Performance-Driven Router for FPGAs

Larry McMurchie, Carl Ebeling

Year of publication: *1995*
Area: *CAD*

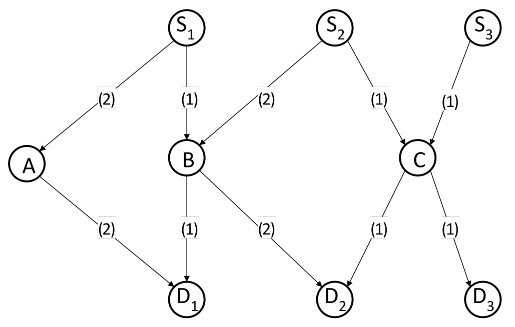


Figure 2. Second order congestion

I personally consider this to be the single most important paper for FPGAs at any technical conference (certainly at the FPGA Symposiums) in the past twenty years. This assertion is based on the accumulated impact of this paper on the FPGA industry and the academia alike. This paper changed the FPGA routing from a major headache with wildly fluctuating results to a reasonably well controlled optimization problem. Today, all FPGA vendors have routers in production that are based on Negotiated Congestion or based on some generalization of the idea. It is also the cornerstone of VPR, the most commonly used tool for Academic Research.

Some papers inspire the audience immediately. Others, like this one, go underappreciated for a while before their significance is fully understood. Most CAD papers propose some new idea, try it on some benchmarks and report 5% to 10% better results than what's available at the time. There are a very large number of such CAD papers. Most of them are good, but they make an incremental impact, which is frequently temporary in nature, until some other CAD paper reports slightly better results once again. Back in 1995, most FPGA researchers expected this to be another 10% step forward. (I certainly did.) Very few understood that this was not an incremental step: this was a game changing fundamental idea that will withstand the challenges of decades and will not be surpassed by any other router, except by its own extensions and generalizations. In the years followed, slowly but surely, the academia and the industry both understood the magnitude of the milestone achieved by the concepts advanced in this paper.

The paper starts by explaining the basic negotiation idea and how the first order congestion can be dealt with. Then the second order congestion is analyzed (see Figure 2 above) and the need for the "history cost" is introduced. Then the concept is generalized to account for routing delays. The paper ends with a pseudo-code of the algorithm and some experimental results (11% better than what was known from a commercial tool.) This is a very good paper, but it is not a model of clarity. This is the kind of paper that one needs to read many times, each reading revealing some nuance that was not apparent in previous readings.

Today, we are able to use the negotiated congestion widely and very successfully. Despite that, why it works so well eludes us at a theoretical level. (For example we rigorously understand why and how annealing works and converges. We do not understand negotiated congestion at an equivalent level.) We do not, yet, have full theory. As such, this paper will continue stimulating further research on this topic, experimental and especially theoretical.

Sinan Kaptanoglu

DOI: http://doi.acm.org/10.1145/201310.201328

Simultaneous depth and area minimization in LUT-based FPGA mapping

Jason Cong, Yean-Yow Hwang

Year of publication: *1995*
Area: *CAD*

This paper was a major step in the evolution of cut-based technology mapping algorithms.

The approach was started by the *FlowMap* algorithm, published a couple of years earlier, which was the first polynomial-time technology mapper that could claim theoretical optimality (in this case, in terms of logic depth). The key idea was to model the LUT mapping problem as the computation of a *minimum-height*, *K-feasible cut*, which is a *global* treatment, as opposed to the existing mapping algorithms of the time, which were based on *local* clustering.

In its original form, the *FlowMap* algorithm actually computed a *min-cut* on a transformed graph that guarantees minimum-height. While simple and efficient, this had a tendency of generating smaller LUTs, and, because each cut is computed independently, it could yield substantial amount of logic replication. Both effects contributed to higher area cost. The post-processing algorithms originally proposed to alleviate this problem, as well as the subsequent revision that considered relaxation of height requirement along non-critical paths, were both essentially local refinement, thus not an integral part of the global cut-based framework.

The *CutMap* algorithm in this paper made several key contributions. First it used a cost function to control the generation of the cuts. This not only allowed the exploration of all K-feasible cuts (not just the minimum cut), but also enabled the general modeling of secondary optimization objectives, or multiple concurrent objectives, in the cut-base mapping procedure. Second it tried to capture logic sharing globally through the pre-assignment of costs, reducing order dependency, and coupling quality with overall network structure more closely. Finally it identified *provably non-degrading* pruning criteria for its cut enumeration speed-up. While in today's light these may seem natural, at its time these strategies were not widely seen in technology mapping works; and indeed after the publication of this paper, a large number of papers in related fields adopted similar approaches. Last but not the least, *CutMap* performed well in practice; it became the base of comparison, in place of the original *FlowMap*, in many subsequent studies.

Cut-based multiple objective optimization has since become the norm of technology mapping both for and beyond FPGA world. This paper played an important role in making that happen.

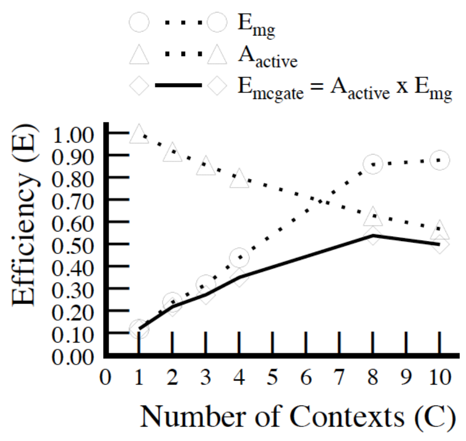
Eugene Ding

DOI: http://doi.acm.org/10.1145/201310.201322

DPGA Utilization and Application

André DeHon

Year of publication: *1996*
Area: *Applications*



As one of the earliest publications on runtime reconfiguration of FPGAs, this paper lays the groundwork for a large body of research to follow.

The authors present an analysis of device architecture and application design patterns directed at improving FPGA efficiency.

Prior to this paper, many researchers had been fascinated by the idea of runtime reconfiguration, and some had even built systems demonstrating applications using dynamic reconfiguration on commercial FPGAs. A notable example is the work by Chris Jones, et. al, on "Issues in Wireless Video Coding using Run-time-reconfigurable FPGAs." However, except for a few ad hoc ideas, there was no clear understanding of the important design patterns necessary to exploit runtime reconfiguration; nor was there a clear understanding of the costs and benefits, or even the metrics to quantify them.

DeHon was an early thinker along these lines and had already introduced the concept of a dynamically reconfigurable device as part of his graduate research at MIT. In this paper, he introduces a set of useful design patterns and shows how they, along with a dynamically reconfigurable device, leads to an area efficiency advantage. He does this in a rigorous manner by introducing metrics and modeling of device and application characteristics. While these contributions are significant in their own right, this paper was inspiring to others in more general ways. It is an early example of how to bring quantitative analysis to understanding the costs and benefits of architectural and system level mechanisms in the context of reconfigurable devices. Furthermore, with DeHon's bold vision of a new type of device—fundamentally different from commercial arrays, with their significantly limited configuration bandwidth—DeHon sets an example of innovative thinking at the micro-architecture level. This was a much-needed example in a research community often content to work within the limitations of commercially available devices, which are optimized for different application objectives.

This paper has had a lasting effect in several ways: it set the standard of quantitative analysis in reconfigurable computing and it inspired a generation of researchers to explore means and applications for runtime reconfiguration. The basic architectural and application partitioning ideas have been implemented by numerous researchers and in start-up companies, including the commercially successful Tabula family of 3D Programmable Logic Devices.

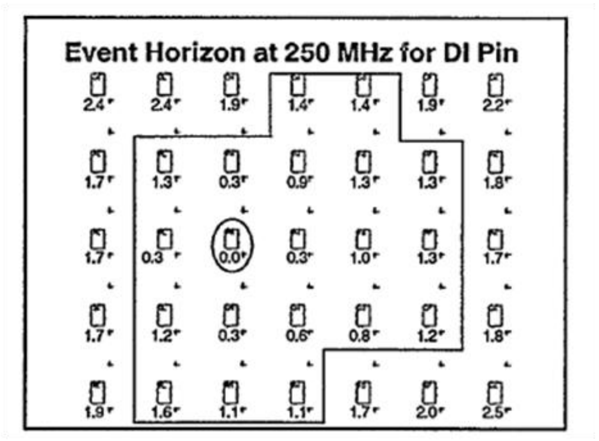
John Wawrzynek

DOI: http://doi.acm.org/10.1145/228370.228387

Signal processing at 250 MHz using high-performance FPGA's

Brian von Herzen

Year of publication: *1997*
Area: *Applications*



This paper was an inspirational tour de force in FPGA design showing the maximum performance achievable with an FPGA, and how one would go about extracting that performance.

To fully appreciate this paper, you have to remember what designs were like in 1997. Intel's Pentium, often today's standard of high-frequency operation, was only running at 75-100MHz in 0.5µm and 0.6µm technologies. You wouldn't even see a 200MHz Pentium until 0.35 micron. Most FPGA designs ran at 25-40MHz. The general impression was that FPGAs were necessarily slow compared to ASICs and processors.

Von Herzen shows that he can extract useful 250MHz operation out of a 0.6µm (today we might say 600nm) Xilinx XC3100A in 1997 – many FPGA users would be happy to see that performance today out of their 45nm parts! It does require heroic effort in matching the design with the FPGA fabric, including careful placement, pipelining to the layout, and careful planning of exactly how far a signal can travel over the network within a clock cycle (the "Event Horizon" shown in the figure above). Significantly, the paper outlines the design approach. This provides a forward look at what CAD and architectures must do to fully utilize these FPGAs. Later work on clocked FPGAs (e.g. GARP, HSRA [see Tsu, et. al. 1999], CHESS, SFRA, Tabula), interconnect retiming, and streaming compute models build out this vision.

In many ways, this is a model of what you want to see from an FPGA application paper. It demonstrates results that seem beyond the capabilities of FPGAs and details how to use the features of the device to do that – providing lessons that are FPGA-specific and applicable beyond the particular design.   Although parts of the paper are dry and read like a lab report, the paper is readable and relevant even today.

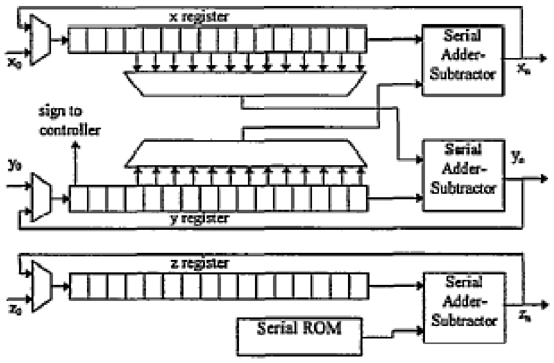
André DeHon and Steve Trimberger

DOI: http://doi.acm.org/10.1145/258305.258313

A survey of CORDIC algorithms for FPGA based computers

Ray Andraka

Year of publication: *1998*
Area: *Applications*



If you ever needed to know how to build efficient circuits for doing transcendental functions in an FPGA you would have likely studied this paper.

In 1998, programmable DSP chips were commonly used for signal processing and the state-of-the-art devices had about 128KB of on-chip SRAM and 100 MIPS of performance. FPGAs were providing the opportunity to build hardware signal processing systems that were not possible using a programmable DSP chip and much easier to build than an ASIC. The FPGAs of 1998 were much smaller than today, having on the order of a thousand four-input LUTs for the Xilinx 4013E mentioned in this paper. Efficient and small implementations of DSP functions and operations were essential.

This well-known paper by Andraka has long been a common starting point for FPGA designers building signal processing algorithms in hardware. DSP requires transcendental functions like *sine* and *cosine*, and computing them in hardware is not as straightforward as calling the appropriate, vendor-provided library functions when writing a software implementation. With signal processing developers familiar with using programmable DSP technology beginning to explore the use of FPGA technology, this paper served to educate this new community that different architectural solutions were required to implement hardware DSP. More specifically, the paper provided an overview of the theory of CORDIC algorithms, which are based on shift and add operations and therefore, highly suited to implementation on FPGAs. Some example circuits for FPGAs were presented for several devices of the day.

Unlike the typical application paper, where the usual subject is the description of a specific application implemented on an FPGA, this paper targeted a community of application developers. Educating the community is what a good survey paper should do. On the other hand, there are many equations that make it a challenge to read all the details.

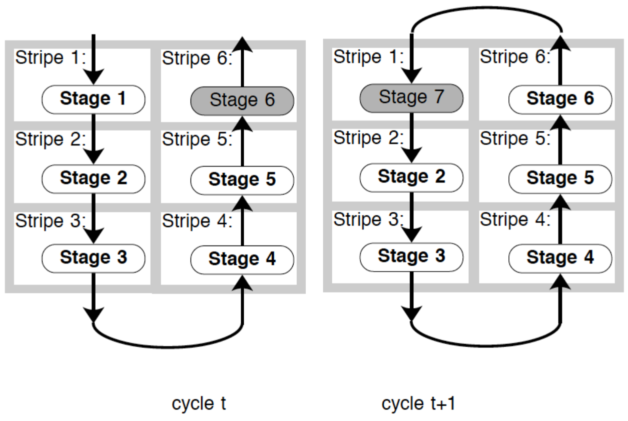
Paul Chow

DOI: http://doi.acm.org/10.1145/275107.275139

Managing Pipeline-Reconfigurable FPGAs

Srihari Cadambi, Jeffrey Weener, Seth Copen Goldstein, Herman Schmit, Donald E. Thomas

Year of publication: *1998*
Area: *Reconfigurable Computing*



How can an FPGA design scale with Moore's Law? This work introduced both a computational model abstracted from the particular size of a specific FPGA and relatively complete details on how you would go about implementing an FPGA to support the model.

In 1998, the largest FPGAs had only 50,000 LUTs; it was not uncommon for users to struggle with designs that would not fit on a single FPGA. Furthermore, even if your design did fit at a particular FPGA generation, when the next generation of FPGAs was introduced, it was necessary to redesign, or at least recompile, the design to exploit the greater capacity available in the next generation parts. While today's million LUT FPGAs make the first problem less of a concern, the lack of compatibility and scalability across generations remains.

PipeRench provided the developer with a logical rather than physical model for run-time reconfiguration. It views the computation as a large, static, feed-forward graph and uses run-time reconfiguration behind-the-scenes to swap different *pieces* of a circuit in and out when that circuit would otherwise be too large to fit in the available hardware. Uniquely, PipeRench showed how it was possible to intimately overlap (pipeline) reconfiguration with computation, allowing rapid configuration switching with modest bandwidth to configuration memory external to the array. The PipeRench model allowed a bitstream compiled for one instance of PipeRench to be loaded into a larger (or smaller) instance of PipeRench, with a corresponding increase (or decrease) in performance

The PipeRench effort was a pioneering large-scale system project covering a wide range of issues in reconfigurable computing. This paper explains the management of configuration data and presents some of the lower-level design details for the PipeRench platform. It discusses the finer points of the configuration process and mechanisms to ensure data arrives at the appropriate physical location where a virtual pipeline stage is mapped. Anyone beginning a reconfigurable architecture project should read this and other PipeRench papers to get a feel for the wide variety of design problems they may encounter and some innovative ways to solve them.

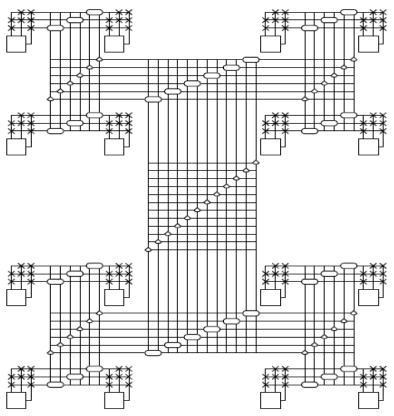
Katherine Compton and André DeHon

DOI: http://doi.acm.org/10.1145/275107.275120

Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization)

André DeHon

Year of publication: *1999*
Area: *Architecture*



This is a model paper for the FPGA conference – an excellent mix of theory, empirical analysis and insightful commentary. The results break commonly-held assumptions about FPGA architecture optimization and in particular demonstrate that the relative breakdown of logic area vs. routing-mux area leads to an optimal point at which logic utilization is not 100%.

This was the first paper to explain the wiring needs of different pieces of a design and the relationship between worst-case wiring requirements and its effect on the overall cost of the device. The paper influenced many future research efforts on FPGA architecture and continues to be cited more than a decade after publication. The definition of a routing approach that would later evolve to the HSRA [see Tsu, et. al. 1999] also revived interest in hierarchical FPGA architectures and their analysis in both academia and industry.

One other contribution is a clear description of the correspondence between the style of architecture and the style of the CAD algorithms – in this case hierarchical and recursive decomposition. DeHon further contrasts the relationship between interconnect growth rate and device size and the empirical results on the efficiency of the architecture at packing designs.

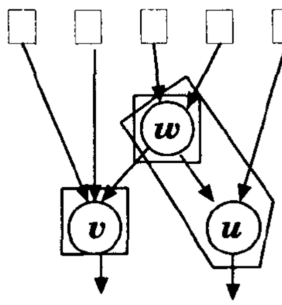
Perhaps the most notable aspect of this work, however, is the overall methodology. Using an architectural model based on tree-of-meshes allowed for a more scalable theoretical model of interconnect growth tied into traditional Rent-based wireability and additionally challenged the traditional concept of a tiled array of logic clusters. At the same time, a concrete area model is used to provide practical credibility to the data that comes from the empirical analysis. Where results differ from the conventional wisdom, DeHon provides clear intuition for the underlying behavior. The overall impact is strengthened by the comprehensiveness of this approach.

Mike Hutton

DOI: http://doi.acm.org/10.1145/296399.296431

Cut Ranking and Pruning: Enabling a General and Efficient FPGA Mapping Solution

Jason Cong, Chang Wu,
Yuzheng Ding



Year of publication: *1999*
Area: *CAD*

The cut generation, ranking, and pruning techniques introduced in this paper reduced the complexity and runtime of cut-based LUT mapping thereby contributing to the scalability of technology mapping for LUTs.

Since the earliest commercial FPGAs, the lookup-table has been the workhorse for implementing logic in most devices. The task of mapping logic to lookup-tables, called technology-mapping, is both central and unique to an FPGA CAD flow. It is not surprising then, that over the years, significant effort has been expended finding effective and efficient techniques to perform technology-mapping.

This paper, published in 1999, was neither the first nor the last paper on the subject. Previous algorithms such as FlowMap, Chortle, and DFmap had already been published, and it was well-known that efficient mapping of logic to lookup-tables was possible. Rather than just presenting a new algorithm, this paper presented techniques that were general enough to improve the efficiency of some of the core operations that are at the heart of almost any technology-mapping algorithm. In particular, this paper addressed *cut generation*, in which a collection of cuts are formed, *cut ranking*, in which the cuts are evaluated and ranked, and *cut pruning*, in which cuts that are not essential are discarded. For large circuits, these tasks are all computationally expensive, so efficient algorithms are essential to scale technology-mapping to large circuits. The authors also showed how these techniques can be applied to improve previously published algorithms.

Taken together, these techniques have since become an essential tool in any FPGA CAD algorithm designer's arsenal and have been built upon numerous times. Today, technology-mapping is still an active research area (driven in part by fracturable LUTs and related innovations), and it is likely that the techniques from this 1999 paper will continue to influence the design and implementation of new technology-mapping algorithms.

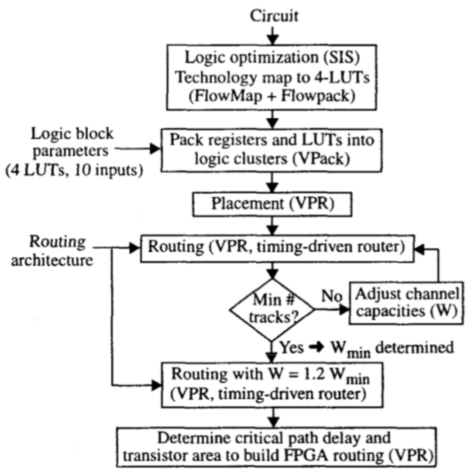
Steve Wilton

DOI: http://doi.acm.org/10.1145/296399.296425

FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density

Vaughn Betz, Jonathan Rose

Year of publication: *1999*
Area: *CAD*


Circuit
Logic optimization (SIS) Technology map to 4-LUTs (FlowMap + Flowpack)
Logic block parameters (4 LUTs, 10 inputs)
Pack registers and LUTs into logic clusters (VPack)
Placement (VPR)
Routing architecture
Routing (VPR, timing-driven router)
Min # tracks?
No
Adjust channel capacities (W)
Yes → W_{min} determined
Routing with $W = 1.2\,W_{min}$ (VPR, timing-driven router)
Determine critical path delay and transistor area to build FPGA routing (VPR)

This work added timing driven routing with accurate delay estimations to VPR, which allowed it to be used to explore FPGA interconnection network architectures. Roughly 90% of the area of FPGAs goes to the configurable routing, and as much as 80% of the critical path delay of typical circuits goes to routing delay. Getting the interconnection network right is therefore crucial to producing an FPGA with good cost and performance. This is especially true as FPGAs scale up since, according to Rent's Rule, the amount of wire must grow faster than the amount of logic.

Unfortunately, architects are often forced to make decisions based on intuition and past experience instead of hard data based on benchmarking and analysis. CAD tools are typically tuned to a single architecture, and changing that architecture can break the tools or at least reduce their effectiveness for evaluating alternative architectures. Moreover, measuring the effect of the interconnect on performance requires timing-driven versions of synthesis, placement and routing.

This paper expands VPR by adding a timing-driven routing algorithm that included an Elmore delay model for accurate delay estimation and describes a methodology for evaluating FPGA routing architectures using VPR. VPR is University of Toronto's architecture-independent set of FPGA CAD tools that included synthesis, placement and routing. These tools allowed the architect to define a new architecture (within limits) via an architecture description language, and the tools automatically adapted to this architecture.

This study started by assuming a traditional island-style FPGA architecture and then tried to determine the best way to segment the wires in the architecture and the best way to connect those segments using either pass transistors and tri-state buffers. Using VPR, the authors were able to automatically explore a large portion of the parameter space using a set of standard benchmarks to find the best values for combinations of these routing parameters.

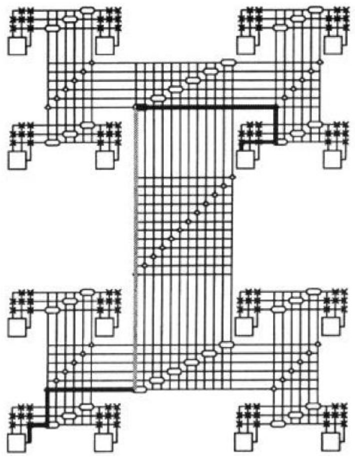
In the end, however, it was the methodology of the paper and the tools that were used that had the most impact. Altera's Stratix architecture, introduced just a few years later, was designed using a similar methodology and a toolset based on VPR, as described in the 2003 Stratix paper [see Lewis et. al. 2003]. It is interesting to note that that paper concluded that it was best to use direct-drive mux-buffers, an alternative not included in this study. This goes to show that innovation requires both thinking outside the box as well as the tools to evaluate new ideas.

Carl Ebeling

DOI: http://doi.acm.org/10.1145/296399.296428

HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array

William Tsu, Kip Macy, Atul Joshi, Randy Huang, Tony Tung, Omid Rowhani, Varghese George, John Wawrzynek, André DeHon



Year of publication: *1999*
Area: *Architecture*

This paper from the Berkeley BRASS group was all about performance: Is it possible to design an FPGA architecture that can compete with processors and ASICs in terms of clock frequency? FPGAs were (and still are) running at 5x – 10x slower clock frequency, largely due to the effect of configurability on both logic and interconnect delay. Von Herzen's [1997] paper provided a sense of what was possible by demonstrating a real application running on a Xilinx 3100 part at 250MHz, at least 5x better than most carefully designed circuits. HSRA was an attempt to realize this potential via a combination of architectural and CAD tool innovations.

The overall methodology was to design an architecture to a specific clock frequency, something that went against the whole philosophy of FPGAs. However, by doing this the authors were able to precisely define the number of logic levels and the interconnect distance traveled in a clock cycle. This led to a highly pipelined architecture, with the configurable interconnect being pipelined as well.

Perhaps the most novel aspect of the HSRA architecture was its hierarchical interconnect structure based on the fat tree. This allowed the wires to be point-to-point and thus registered at clock cycle boundaries. It also allowed the "hop" distance between any two points to be known exactly, something required to make the solution of the place-and-route problem feasible from a timing standpoint.

Of course not all applications can be pipelined to death as required by the HSRA architecture. To address this issue, the paper proposes using C-slowing to introduce extra parallelism in the circuit to cover the latency induced by circuits with large feedback loops. (This is similar to the fine-grained multi-threading approach for hiding memory latency in the Tera MTA processor.) For a follow-up on ideas for achieving deep pipelining in FPGAs, see the 2003 Berkeley Ph.D. thesis by Nicholas Weaver.

In summary, the HSRA paper broke new ground in FPGA architectures by aggressively pushing the envelope in one dimension, timing. While many of the ideas in this paper have had impact on FPGA design, the HSRA architecture itself was too radically different from traditional FPGAs to have had much of an effect on commercial FPGAs.

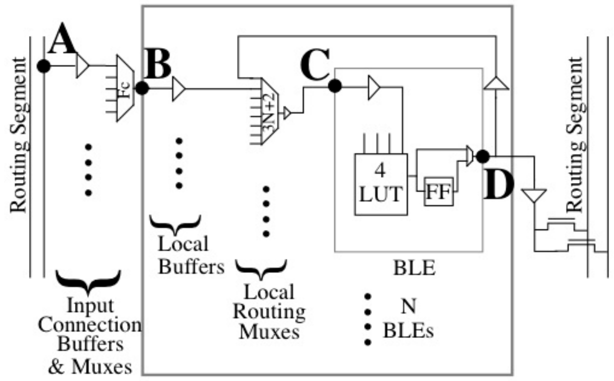
Carl Ebeling

DOI: http://doi.acm.org/10.1145/296399.296442

Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density

Alexander (Sandy) Marquardt, Vaughn Betz, Jonathan Rose

Year of publication: *1999*
Area: *CAD*



This paper was the first to quantify the speed advantage of cluster-based logic blocks.

In 1999, most commercial FPGAs, like the Altera Flex and Xilinx Virtex FPGAs already had cluster-based logic blocks. However, the modeling and evaluation of these sorts of architectures was still in its infancy. In the previous year, Betz had shown that cluster-based logic blocks led to improved density. The real advantage of clustered-based logic blocks, though, was speed, as this paper demonstrates. In doing so, this paper opened up an entirely new research area, setting the framework for numerous packing algorithms that have become a fundamental part of any FPGA CAD flow.

In this paper, the authors first described a CAD flow and evaluation metric suitable for evaluating these sorts of architectures. They then carefully presented a parameterized cluster architecture, inherited from previous work by Betz, which provided a way of speaking and reasoning about variants of cluster-based logic blocks. The authors then presented the T-VPACK algorithm which was the first published timing-driven clustering algorithm for this architecture. Using a careful experimental methodology, they concluded that cluster-based logic blocks can significantly improve the speed of an FPGA, and showed that the "sweet spot" for the cluster size was approximately seven to ten logic elements; such cluster sizes became the standard in academic and commercial architectures for years.

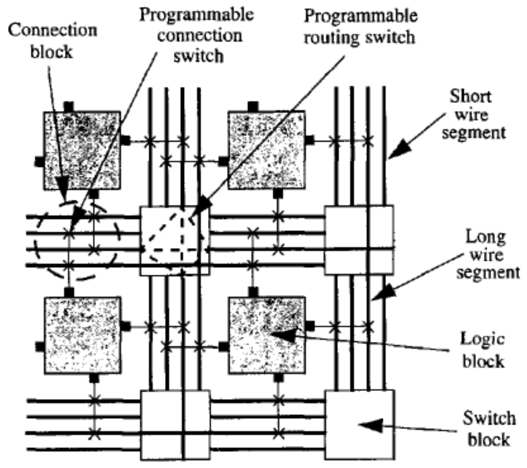
The real value of this paper was that it set the framework for future researchers to develop new clustering algorithms and new cluster architectures. Later clustering algorithms, such as RPACK, iRAC, DVPack, and P-T-VPACK all built upon this framework. Going forward, clustered logic blocks are becoming even more important; very recent work by the same research group has significantly expanded the design space of clustered logic blocks, but have maintained many of the underlying assumptions, evaluation methodology and architectural framework presented in this paper.

Steve Wilton

DOI: http://doi.acm.org/10.1145/296399.296426

Automatic Generation of FPGA Routing Architectures from High-Level Descriptions

Vaughn Betz, Jonathan Rose



Year of publication: *2000*
Area: *CAD*

FPGA architecture research is a very complex undertaking to answer even the simplest questions. An architect might be convinced that their new idea for purely unidirectional wires, or larger logic blocks, or a new type of carry chain would significantly improve the area, power, performance, or reliability of future devices, and is clearly "the right way to do it". But, standing between the aspiring researchers and their justified fame is the need for applications and tools to test their ideas, as well as a host of architectural details that frankly aren't part of their brilliant idea. For tools, the development of Pathfinder [see McMurchie and Ebeling 1995]) and VPR [see Betz and Rose, 1999] provide an efficient back-end for most mapping tasks.

However, for the architecture of the FPGA's interconnect, one can't simply ignore these issues. We must pick the wirelength, connectivity, logic block structure, and innumerable other details that aren't what we want to focus on, but must still be determined. While unidirectional wires might be a great idea, if we fully populate all the switchboxes and connection blocks, the resulting area and capacitance increases will swamp the benefit of most improvements. Alternatively, we could only 50% populate these units, but what if we connect the outputs of all our logic blocks to the odd wires, and the inputs to the even ones? Now we have a completely disconnected architecture that can't compute anything at all.

The key to solving this problem was VPR's architecture description language and architecture generators described in this paper. Simply put, this paper focuses on the unglamorous task of handling all the details of the routing architectures we don't care about. How are the logic blocks connected, and how do we make sure weird interactions between odd and even wires don't make our system unusable? How are the switchboxes organized? When we have long wire segments traversing a fraction of the chip, how do we make sure the breaks are staggered in a reasonable way? The research described in this paper, and embodied in the VPR toolsuite, answered each of these problems for numerous FPGA architecture research tasks. As a result they helped create a huge amount of architectural innovation by solving the portions that weren't the focus of the research, since they didn't need to be – they made the researcher more productive, and as a result helped move the entire field forward.

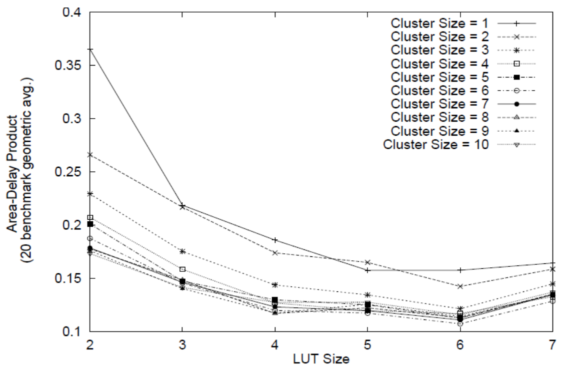
Scott Hauck

DOI: http://doi.acm.org/10.1145/329166.329203

The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density

Elias Ahmed, Jonathan Rose

Year of publication: *2000*
Area: *Architecture*



This seminal study considered both the performance and area impacts of organizing logic into hierarchies in FPGA architecture, namely LUT size (K) and cluster size (N). At the time this paper was written, academic FPGAs were just starting to consider clustered organization. Commercially, both Altera and Xilinx FPGAs used hierarchy but with different cluster sizes and interfaces. This paper was the first work to fully consider both the area and delay tradeoffs in context with the number of inputs to the cluster, and to do so in a clear scientific manner.

The key results of this paper were to relate the ideal number of input muxes (I) given the overall cluster-size (as a function of N,K) and to illustrate the optimal ranges of N and K for specific design goals combining area and delay. Remarkably, the conclusion that LUT-sizes of 5-6 were even better for area-delay than previously seen foreshadowed the change to 6-input LUT structures that we see in modern FPGAs.

The paper is notable not just for its conclusions, but for the clarity and persistence of the end-to-end exploration methodology. Ahmed and Rose point out that the optimal values for these architectural results are dependent on current process parameters, that process parameters had changed the optimal point since previous studies, and outline a clear and reproducible framework for updating the work as the underlying technology continues to evolve.

Beyond the obvious influences of this paper, by now well-known to every FPGA architect, this work also set a standard for a complete (synthesis to routing) architectural evaluation as subsequent architectural studies adopted the ideal ranges set forth here for their follow-on studies in both architecture and CAD algorithms.

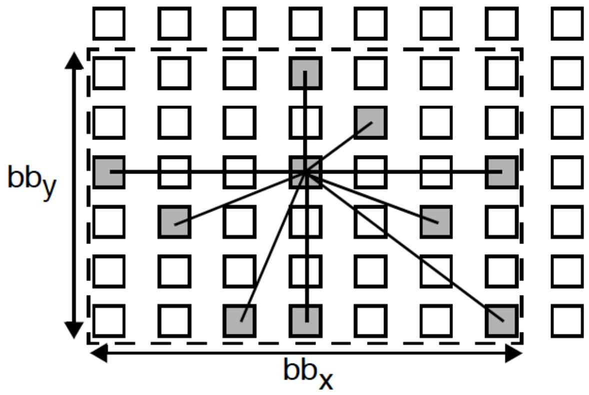
Mike Hutton

DOI: http://doi.acm.org/10.1145/329166.329171

Timing-driven placement for FPGAs

Alexander (Sandy) Marquardt, Vaughn Betz, Jonathan Rose

Year of publication: *2000*
Area: *CAD*



This paper presents the timing-driven placement algorithm used in the most popular open-source FPGA placement and routing system from academia, VPR. Almost every project for new FPGA architecture evaluation has used VPR since its introduction. In addition to its popularity and widespread usage, the T-VPlace algorithm introduced here has three important algorithmic contributions to FPGA timing-driven placement.

- Timing optimization in T-VPlace is carried out by minimizing the weighted sum of the wirelength cost and timing cost in a simulated annealing based optimization engine. The timing cost is measured by the sum of the weighted wirelengths of all nets, where the weight of each net is a polynomial function of the net timing criticality. It was later shown that the criticality-based weighting function used in T-VPlace satisfies the property of asymptotic slack control, which leads to good timing convergence. Moreover, both the wirelength cost and the timing cost in T-VPlace are self-normalized based on the results from the previous iteration, providing great stability to the algorithm.
- This paper shows that the timing slack of each net need not be updated after each cell movement during the iterative placement framework. Accurate path-based timing analysis is applied only periodically, at the end of each temperature iteration in the simulated annealing framework. Using the “stale” slack information did not hurt the timing optimization result but greatly improves the efficiency of the T-VPlace algorithm, although later work shows that this stale slack information does hurt the performance of highly pipelined designs.
- Given the segmented programmable interconnect architecture, the delay between a source-sink pair cannot be estimated simply using its Manhattan distance. However, invoking a detailed router to compute the delay between every source-sink pair during placement is way too costly. Exploiting the symmetry in FPGA architectures, T-VPlace uses a pre-computed delay lookup table, indexed by the distances in the horizontal and vertical directions, to allow fast delay lookup.

These three techniques combined enable T-VPlace to produce high-quality timing optimization results with efficiency. In fact, the first two techniques can also be applied to timing-driven placement for standard cell designs. T-VPlace was a cornerstone in the FPGA place and route system produced by Right Track CAD Corporation, which was acquired by Altera in 2000. The optimization techniques used in T-VPlace have been incorporated into the Altera’s FPGA design system Quartus and used by tens of thousands of FPGA designers worldwide.

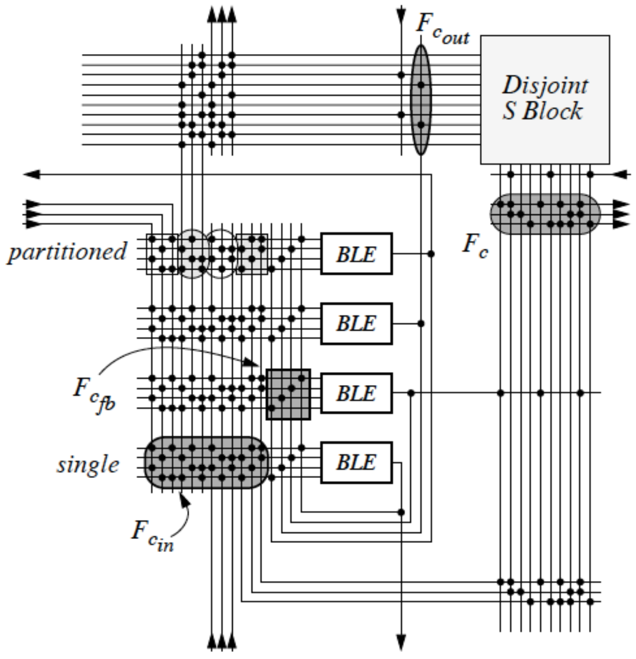
Jason Cong

DOI: http://doi.acm.org/10.1145/329166.329208

Using Sparse Crossbars within LUT Clusters

Guy Lemieux, David Lewis

Year of publication: *2001*
Area: *Architecture*



I consider this paper to be one of the most significant pieces of academic research on routing architectures for FPGAs. This paper investigates the previously unexplored realm of building intra-cluster routing without assuming full connectivity. The FPGA architectures based on clusters have been explored previously in the literature (and in particular at the FPGA Conference) by many research groups. All of this previous work however, went on to analyze cluster and LUT sizes and connectivity matrices while assuming that the intra-cluster routing was done by a full crossbar. What we learned from this paper was the very important fact that some of the conclusions of this analysis about the optimality of clustered FPGA architectures can be significantly altered by heavily depopulating the crossbar, while maintaining good routability and good performance.

The paper starts by proposing a highly generic type of clustered FPGA architecture in order to demonstrate the main point. To be able to calculate and compare relevant parameters (such as area and performance), the authors first introduce area and delay models. Then they introduce some enhancements to the CAD tools (VPR) in order to deal with the consequences of heavy switch depopulation. Finally they present experimental results on area and performance by exploring architectural parameters; they even consider additional LUT inputs and CAD tool mapping time.

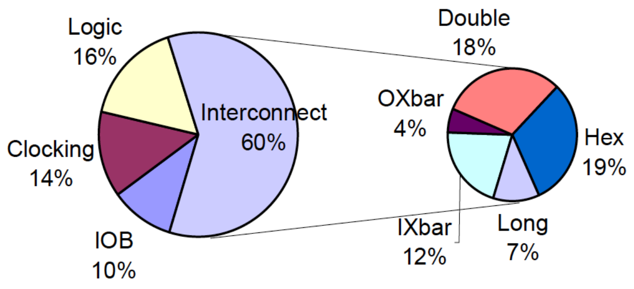
This paper has been inspirational for further work in this area, especially in the FPGA industry. Today, most of the modern commercial routing architectures are based on clusters. In all known cases to me, these clusters contain partially populated crossbars. They are all optimized for slightly different objectives using different switch depopulation schemes. The impact of this paper is not purely industrial however. This paper also had a significant academic impact. In the following years, the experimental work of Lemieux and Lewis has been generalized and put on strong theoretical foundations by other researchers. I am certain that this paper will continue stimulating research in this fertile area of sparsely connected intra-cluster routing.

Sinan Kaptanoglu

DOI: http://doi.acm.org/10.1145/360276.360299

Dynamic Power Consumption in Virtex™-II FPGA Family

Li Shang, Alireza Kaviani, Kusuma Bathala



Year of publication: *2002*
Area: *Architecture*

This paper provided an important first step in improving the understanding of FPGA power consumption and identifying possible areas of optimization. Before this paper, very little work on FPGA power consumption had been published.

For some time it was assumed that interconnect was the main consumer of dynamic power, but this paper provided the experiments to show that this assumption is indeed the case. The dynamic power analysis of the additional FPGA components provided the motivation for a collection of subsequent papers on power-aware CAD. The power distribution results were clearly believable given the inside knowledge of the Xilinx Virtex II available to the authors and the validation which included both simulation and physical measurements.

Ten years ago FPGA power reduction was just starting to move to the forefront to take its place next to area reduction and delay minimization as an important optimization goal. Not only did this paper provide valuable end results on FPGA dynamic power consumption, it also provided details of the power analysis methodology that would be used by numerous FPGA researchers over the coming decade.

In many ways the paper provides an ideal model for a paper that involves both academic and industry researchers. In this case Xilinx provided access to models and benchmarks needed for the exploration, and advanced academic power analysis research techniques were used to examine the results. The familiarity of the Virtex II architecture by the research community precluded the need for the disclosure of confidential information. Overall, the paper provided an important kick-start for substantial FPGA dynamic power reduction in both CAD techniques and in architecture.

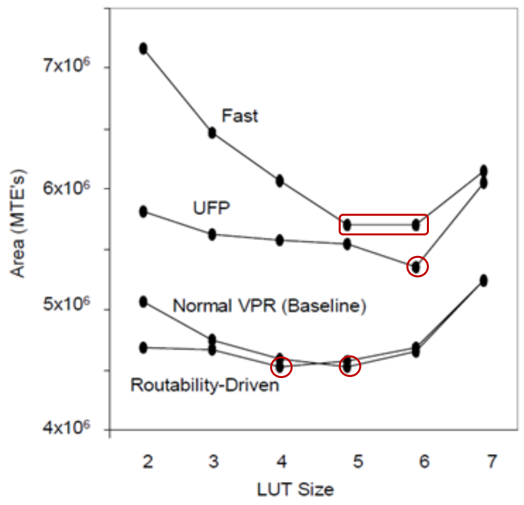
Russ Tessier

DOI: http://doi.acm.org/10.1145/503048.503072

On the Sensitivity of FPGA Architectural Conclusions to Experimental Assumptions, Tools, and Techniques

Andy Yan, Rebecca Cheng,
Steven J.E. Wilton

Year of publication: *2002*
Area: *Architecture*



This paper is essential reading for any beginning researcher in architecture and CAD.

Architectural exploration research is a staple of the FPGA conference. Experienced researchers in this area know that the outcomes of this (or any other) type of research can be heavily influenced by the chosen experimental methodology. The set of examined benchmark circuits, the CAD tool that maps the circuits to the architecture, and other architectural parameters can all affect the answer to the question, "*what is the best value for* ***this particular*** *FPGA architectural parameter?*"

Knowing that these factors *could* affect the outcome is one thing; seeing that they *do* affect the outcome is another. This paper presents actual data as proof that conclusions drawn from experiments designed to find the "best" FPGA architectural parameters (such as LUT size or connection block flexibility) are not necessarily conclusive—the results may actually be dependent on the experimental setup. As an example, the figured included above demonstrates that the LUT size that minimizes overall FPGA area for a set of circuits ranges from four to six inputs depending on the CAD tools used in the experiment**.**

More significantly, the paper encourages researchers to more carefully design their experiments and frame their conclusions. It is therefore important reading for both experienced researchers and new ones. In fact, it is essential reading for any beginning researcher because they are the most prone to take results at face value and make dramatic, sweeping conclusions based on the data they have gathered.

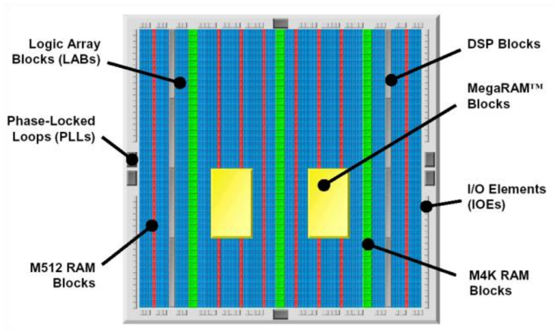
In short, this paper demonstrates the importance of carefully constructing experiments and thoughtfully analyzing the results—questioning their validity and applicability. The core message of the paper, the dependent nature of experimental methodologies and results, is timeless and transcends the FPGA research field.

Katherine Compton

DOI: http://doi.acm.org/10.1145/503048.503071

The Stratix™ Routing and Logic Architecture

David Lewis, Vaughn Betz, David Jefferson, Andy Lee, Chris Lane, Paul Leventis, Sandy Marquardt, Cameron McClintock, Bruce Pedersen, Giles Powell, Srinivas Reddy, Chris Wysocki, Richard Cliff, Jonathan Rose



Year of publication: *2003*
Area: *Architecture*

The FPGA Symposium was conceived to be an interchange between academia and industry. Most of that interchange occurs informally: during question and answer sessions, around posters, or during lunch or dinner. This paper is a rare illustration of that interchange between academia and industry in the form of an actual paper.

The University of Toronto team lead by Jonathan Rose had, over a period of years, built the VPR tool suite to explore simplified classes of FPGA architectures. VPR included packing, placement and routing algorithms controlled by a single architectural representation. This enabled many architectural questions to be explored quantitatively, and these results established Toronto as one of the most important academic centers for FPGA research.

In 1998, Professor Rose founded a startup company, Right Track CAD, in an effort to commercialize this work. Simultaneously, Altera was attempting to improve their FPGA architecture in order to compete with Xilinx's successful Virtex family. In 2000, Altera acquired RightTrack, and the new team developed the Altera FPGA Modeling Toolkit, or FMT, to optimize the first version of the Stratix architecture. This paper describes the Stratix architecture details, but more importantly describes the process used to make the architectural decisions that resulted in Stratix. In that way, it demonstrated that the quantitative approach taken by VPR could be applied using actual user metrics like physical area and critical path delay. The process, with improved versions of FMT, has been used for five generations of Stratix architecture.

The primary technical contribution of this paper is the demonstration that direct-drive routing, composed of pass-transistor multiplexors followed by buffers reduced both area and delay compared to other routing switch topologies. This paper's primary significance is due to the process and the academic and technical collaboration demonstrated by this work.

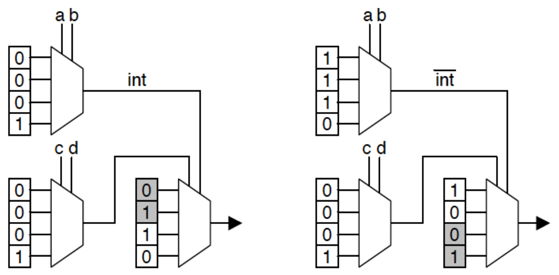
Herman Schmit

DOI: http://doi.acm.org/10.1145/611817.611821

Active Leakage Power Optimization for FPGAs

Jason H. Anderson, Farid N. Najm, Tim Tuan

Year of publication: *2004*
Area: *CAD*



This paper introduced a simple and free optimization that could provide significant leakage power reduction on FPGAs.

As FPGA feature sizes dropped below 100nm and interest in FPGA for mobile applications increased, leakage power emerged as a potential concern and optimization target. Prior to 100nm, leakage power was considered negligible and was not a concern for FPGA mapping tools. Additionally, target FPGA devices were primarily used for prototyping, glue logic, and compute acceleration in environments with abundant power availability. In today's sub-100nm realm, leakage is a substantial fraction (>30%) of total power consumption. FPGA leakage power consumption continues to be a concern since no commercial FPGA allows for the complete power shutdown of device regions. The migration of FPGAs into energy-sensitive applications involving mobile communications and data centers has also increased interest in leakage power reduction in recent years.

Generally, in optimization for a specific constraint, tradeoffs are required. To achieve a better result for one metric, one or more alternative metrics generally become worse. The power-saving technique described in this paper pleasantly proves this axiom wrong. The outlined approach provides up to 25% FPGA interconnect static power savings simply by performing some modest FPGA design preprocessing. No hardware area or performance penalty is incurred by the final design to achieve this power optimization.

The paper takes advantage of a simple, but powerful concept. A CMOS transistor consumes reduced static power if its drain and source are set to logic 1 values when its gate is also set to a logic 1. Thus, if signals driven through the routing interconnect are primarily set to a logic 1, interconnect static power can be significantly reduced. Fortunately, the configurability of FPGAs allows for the selection of logic function signal polarity in most cases. The authors provide a software algorithm to take advantage of this configurability. One of the best aspects of the paper is the complimentary nature of the developed idea. This technique can be used with all other FPGA power reduction techniques without negatively affecting them. The software necessary to implement the technique is simple and easily coded. Undoubtedly, this approach is a triumph of a simple idea applied in an intelligent fashion to take advantage of FPGA configurability.

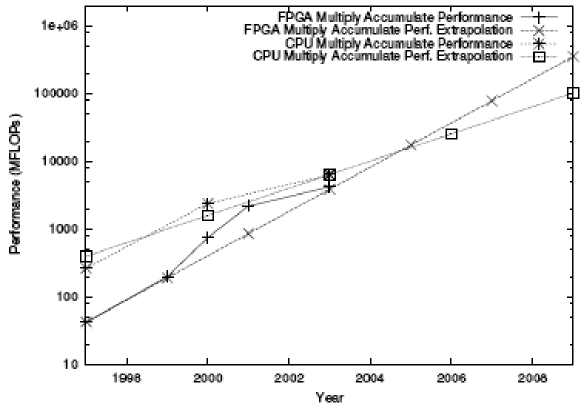
Russ Tessier

DOI: http://doi.acm.org/10.1145/968280.968287

FPGAs vs. CPUs: Trends in Peak Floating-Point Performance

Keith Underwood

Year of publication: *2004*
Area: *Reconfigurable Computing*



The results presented in this paper showed that there was promise for the future use of FPGAs in supercomputers running applications requiring intensive high-precision, floating-point operations.

Ever since the beginning of FPGA history, there was the vision that FPGAs could be used for computing, which was not the primary intent or market for these devices. At the time this paper was published, computing using fixed-point operations on FPGAs was clearly beneficial and best demonstrated in the area of digital signal processing where FPGAs often showed a significant advantage over programmable DSP processors. However, most scientific computations required, and still require, the use of IEEE floating-point, often in double-precision mode, to achieve computational stability. Until this paper, it was generally believed that FPGAs could not implement enough floating-point functionality to compete with the CPUs of the day.

By measuring the size and maximum clock rate of floating-point adders, multipliers, dividers and multiply accumulate operators over several generations of FPGAs and then dividing the sizes into the capacity of the devices, Underwood generated trend lines for the peak performance of the FPGAs for those operators. For CPUs, trend lines were derived for each operation using the corollary to Moore's Law that predicted performance would double every 18 months. The results showed that the peak capabilities of FPGAs had already crossed, or were about to cross ,the trend lines for CPUs and the lines would continue to diverge.

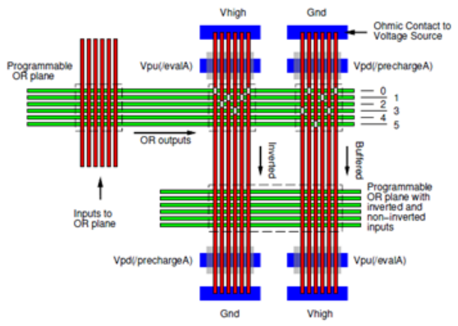
This is the first paper that tried to quantifiably show that FPGAs could be practical as computing devices in the floating-point domain. By developing a rather coarse model, it was possible to show that the trend lines were favourable towards FPGAs being competitive with CPUs at some point. Reconfigurable computing research still had a future! However, the conclusion of the paper is very much dependent on the meaning of peak performance and how that relates to achievable performance – a problem that is yet to be solved even in the much more mature CPU world.

Paul Chow

DOI: http://doi.acm.org/10.1145/968280.968305

Nanowire-Based Sublithographic Programmable Logic Arrays

André DeHon, Michael J. Wilson



Year of publication: *2004*
Area: *Architecture*

The year of 2003 is marked with two breakthroughs in nanowire-based circuit development. One breakthrough is a general and efficient solution-based method for controlling organization and hierarchy of nanowire structures over large areas demonstrated by Harvard University. Another is the fabrication and testing of nanoscale molecular-electronic circuit components that comprise molecular switches sandwiched between metal nanowires, published by HP Labs. Nonetheless, the majority of researchers then focused on nanowire devices and simple nanowire logic/memory components.

DeHon and Wilson were inspired by such breakthroughs and captured the opportunity to timely bridge nanodevices and nanocomponents fabrication to nanosystems building, where the true impact of nanotechnology lies. This paper uses nanowire crossbars to build large-scale PLA-based programmable circuit through modeling and presents detailed architecture design issues addressing various unique challenges. These challenges include how to address the nanowires through a stochastic scheme, how to build the PLA logic and restore the logic through nanowire restoring buffers and inverters, and how to perform precharge clocking. The paper also provides detailed analysis on area, yield, and timing of the nanoarchitecture. Finally, it maps benchmarks to this new architecture to evaluate its logic density. This provides readers a thorough understanding of the design/fabrication challenges of the architecture and its potential advantages over the traditional CMOS FPGA architecture. Such a study is essential for the industry to understand where nanotechnology can elevate the next-generation FPGA devices to. Note that the paper doesn't specifically address fabrication defects of the nanowire structures. The authors addressed such issues in their follow-up publications.

This work shows that it is possible to use emerging, bottom up, fabrication technologies to build high-density large-scale programmable logic without using lithography in the future. This provides an alternative solution that can scale beyond the limitation of lithography which has been the fundamental technology for printing CMOS circuits.

Deming Chen

DOI: http://doi.acm.org/10.1145/968280.968299

Measuring the Gap between FPGAs and ASICs

Ian Kuon, Jonathan Rose

Year of Publication: *2006*
Area: *Architecture*

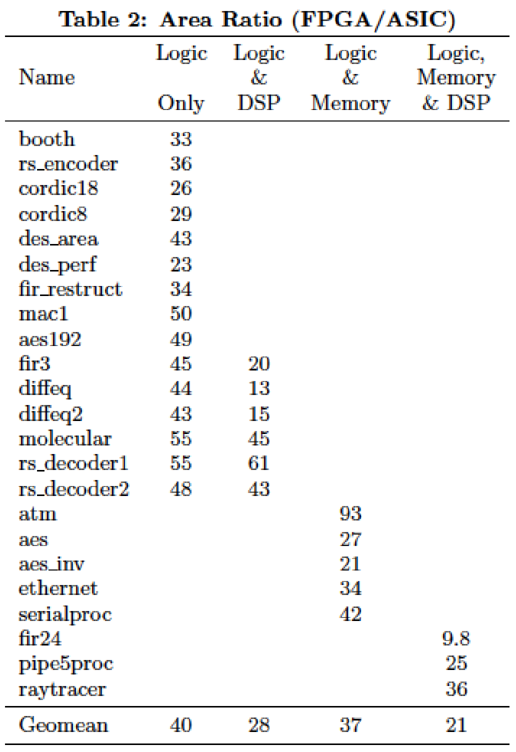


Table 2: Area Ratio (FPGA/ASIC)

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	33			
rs_encoder	36			
cordic18	26			
cordic8	29			
des_area	43			
des_perf	23			
fir_restruct	34			
mac1	50			
aes192	49			
fir3	45	20		
diffeq	44	13		
diffeq2	43	15		
molecular	55	45		
rs_decoder1	55	61		
rs_decoder2	48	43		
atm			93	
aes			27	
aes_inv			21	
ethernet			34	
serialproc			42	
fir24				9.8
pipe5proc				25
raytracer				36
Geomean	40	28	37	21

Everyone understands that programmability has a cost. This paper is one of the most cited papers in this collection because it quantifies the cost of programmability. The abstract claims the core area for an FPGA is, on average, a surprising 40 times higher than a standard cell ASIC and is motivational to all work improving FPGA architectures and structured ASICs.

Prior to this paper, most comparisons were anecdotal characterization of small circuits and tended to only compare FPGAs with mask-programmable gate arrays, suggesting only a 10× area penalty. However, by 2006, ASIC CAD had improved and synthesized standard cell designs were the more common choice for ASIC implementations.

In defense of the FPGA architects of the world, the highly-cited 40× result is exaggerated, because it considers only core area, and is obtained by considering designs with both logic and arithmetic in an FPGA architecture lacking hardened multipliers. This paper breaks down the benchmark suite into four classes based on whether they contain arithmetic or memory in addition to unstructured logic and registers. In the class containing logic and arithmetic, the FPGA architecture that includes hardened multipliers has an area ratio of 28 versus the ASIC. Perhaps the more enduring contribution of this paper is the demonstration of the correlation between benchmark results and FPGA architecture features like memories and DSPs. The contribution of hardened components to optimizing cost and performance cannot be ignored. In modern FPGAs, the decisions about what components to harden and how are as important as the traditional FPGA architecture questions like LUT size and interconnect topology.

Benchmarking papers like this are always controversial because they either make imperfect comparisons between different quantities or they use abstractions that make the comparisons more equivalent but less meaningful. This paper does an exemplary job of making the comparisons, and describing in detail exactly how those comparisons are made to allow the readers to form their own conclusions from the results.

Herman Schmit

DOI: http://doi.acm.org/10.1145/1117201.1117205

High-Quality, Deterministic Parallel Placement for FPGAs on Commodity Hardware

Adrian Ludwin, Vaughn Betz,
Ketan Padalia

Year of publication: *2008*
Area: *CAD*

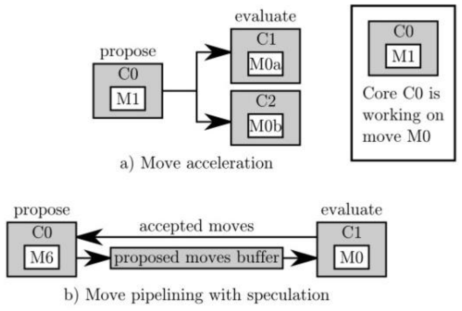


Figure 3: Accelerated vs. pipelined moves.

One of the key issues facing the FPGA industry today is the long compile times for processing designs, brought about by the stagnation in the improvement of processor performance and the increasing size of FPGAs with each new generation of process technology. One of the weapons to combat this trend will be to employ the multiple processor cores that are now ubiquitous on the desktops and servers typically charged with running the compile.

This paper addresses the parallelization of one of the slowest parts of the FPGA CAD flow, placement, and does it in several highly unique and important ways: this is the first and only attempt to parallelize actual industrial placement software, resulting in released, working and successful commercial software. While there have been numerous prior attempts at parallel placement that have been published, they have all been on academic versions of algorithms which don't have to deal with massive device databases, complex timing analysis and the many detailed issues that show up in commercial software.

Second, the paper makes an important statement about the need for *determinism* – that each run of the algorithm gives exactly the same results regardless of how many processors it runs on. Although controversial among academics, determinism is essential in a commercial environment where it is necessary to replicate results both for consistency and debugging. The paper describes the careful work that needs to be done to ensure determinism and shows that the cost in performance loss is small.

Third, the paper performs an insightful analysis of the effect of memory architecture on the performance of the parallel algorithm, and interestingly, shows fairly significant effects that occur from different memory organizations.

The overall performance improvement achieved is impressive and useful: on the placement phase, the speedup is 2.2 times using four cores. On a large design this could save time approaching hours. Since the full compile consists of several more timing-consuming phases, there is more work to be done; this paper is an exemplar for how to do that.

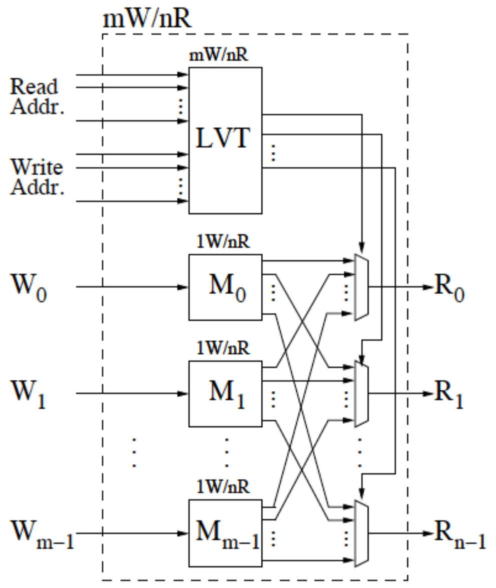
Jonathan Rose

DOI: http://doi.acm.org/10.1145/1344671.1344676

Efficient Multi-Ported Memories for FPGAs

Charles Eric LaForest,
J. Gregory Steffan

Year of publication: *2010*
Area: *Applications*



One of the key features of an FPGA is the ability to provide a large number of simple building blocks, which can be cascaded together to form ever larger and more powerful elements as needed. While the logic block in an FPGA may only support 4 to 6 inputs each, a tree of LUTs can support an arbitrary number of inputs for any desired computation. An individual flipflop can store a single bit of information, but can be ganged together to support 32-bit data, and these elements sequenced into a multi-stage FIFO.

As FPGAs evolved, they started adding fixed components for specific tasks, such as multipliers, memories, and even entire processor cores. These radically improved the implementation of specific common computations, but their relative inflexibility makes them harder to cascade into larger computations. If the hard block does not have the right set of features, we may be forced to abandon those blocks, and go back to cobbling together LUTs and flipflops. A good example of this problem is embedded hard memory blocks. If the memory unit has too little total storage capacity, or too few bits per address, we can easily group multiple blocks to provide the appropriate memory structure. However, the number of access ports to these memories is fixed – if the memory blocks allow two writes a clock cycle, yet our applications needs three, there is little that can be done. Extra read ports are easily supported by duplicating the memory, but for RAMs the number of independent write ports is seemingly a fixed bound.

This paper fundamentally changed this tradeoff. By realizing that the underlying hardware can often operate much faster than the user's actual design, memory ports can be time-multiplexed to increase the number of independent accesses. Also, by clever indirection mechanisms, we can gang together multiple memories to support increased read and write bandwidth, while maintaining control information to find the proper live value. Combined, this gives an effective mechanism for providing increased memory ports for many different applications.

I still recall seeing this work presented for the first time. "Damn, I wish I thought of that!" was my immediate reaction. This is a problem that many of us have faced, but it took LaForest and Steffan to recognize how to overcome this fundamental problem and significantly increase the capabilities of modern FPGAs via a simple memory generator structure.

Scott Hauck

DOI: http://doi.acm.org/10.1145/1723112.1723122