Securing a Bright Future for FPGAs

Stefan Nikolić

CROSSING Research Seminar, TU Darmstadt, 28.11.2024

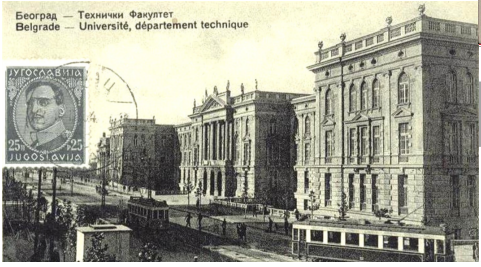
University of Novi Sad, Faculty of Sciences

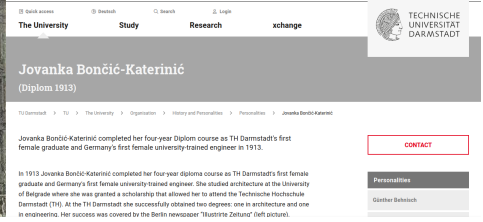
Who am I? A new assistant professor at the University of Novi Sad

A curious connection between Serbia and TU Darmstadt

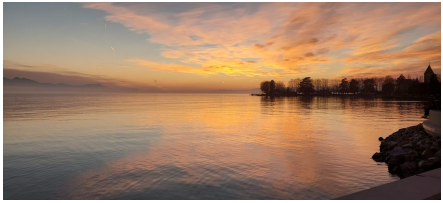
Berliner Illustrierte Zeitung, 1913.

Picture: Archiv TU Darmstadt





The work presented today was done at different places



EPFL, Lausanne, Switzerland

Xilinx, Longmont, USA

University of Novi Sad, Novi Sad, Serbia

Disclaimer: I am no expert on using FPGAs to accelerate computation

Prof. Dr-Ing. Andreas Koch

Technische Universität Darmstadt
Computer Science Department (FB20)
Embedded Systems & Applications Group (ESA)
Hochschulstr. 10
D-64289 Darmstadt

Phone: +49 6151 / 16-22420
Fax: +49 6151 / 16-22422
E-Mail: koch@esa.tu-darmstadt.de

S2|02 Raum E101

Prof. Dr. Zsolt István

Professor

Working area(s)
Distributed and Networked Systems

Contact
zsolt.istvan@cs.tu-...
+49 6151 16-23213
S2|02 A312
Hochschulstraße 10
64289 Darmstadt

But several professors at TU Darmstadt are
(and there may be more, I apologize if I missed them)

I work on

- Developing better ways to design new FPGAs themselves
- Developing better algorithms to map user designs onto these new FPGAs

Why am I here?

Because FPGAs today are not good enough and it is possible that we will not be able to design significantly better **GENERAL-PURPOSE** ones

Hence, we must learn from domain experts about the computational tasks that they are solving, so that we can design families of domain-specific architectures

To sum up, I need your help

But hopefully, you will get something in return

What I hope you will learn from this talk

- How you can benefit from hardware customization in general
- How you can benefit from reconfigurable hardware in particular
- How FPGAs function and how they are designed
- That FPGAs do **NOT** need to contain LUTs
- That you too can have an impact on how future FPGAs and tools used to program them will look like

Today there are open-source tools for generating custom FPGAs

FABulous:

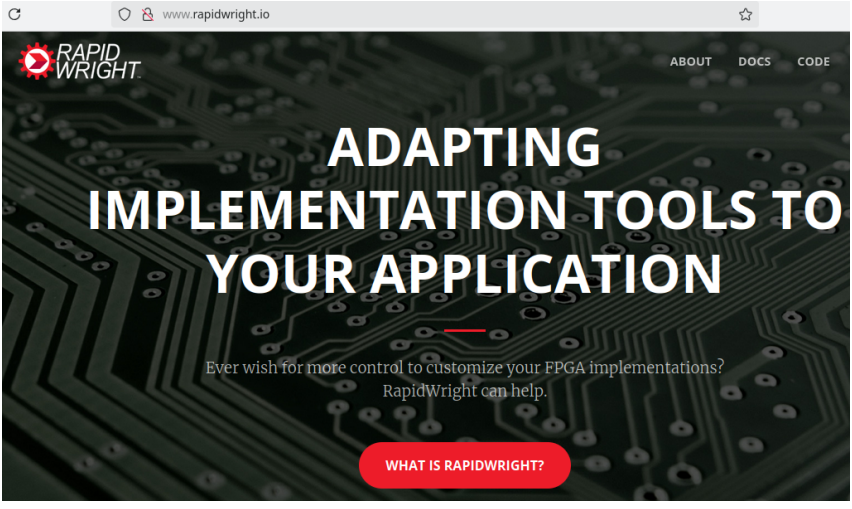
an Open-Everything Framework for (embedded) FPGAs

Open-source tools for programming FPGAs

Yosys+nextpnr: an Open Source Framework from Verilog to Bitstream for Commercial FPGAs

David Shah[*†], Eddie Hung[‡*], Clifford Wolf[*], Serge Bazanski[*], Dan Gisselquist[*] and Miodrag Milanović[*]
[*]Symbiotic EDA; Vienna, Austria; {david,clifford}@symbioticeda.com
[†]Dept. of Electrical and Electronic Engineering; Imperial College London, UK
[‡]Dept. of Electrical and Computer Engineering; University of British Columbia, Canada; eddieh@ece.ubc.ca

And even open access to commercial FPGA tool flows



Before we begin

I would like to thank all my collaborators

Prof. Dr. Paolo Ienne (EPFL)

Dr. Mirjana Stojilović (EPFL)

Morten B. Petersen (EPFL)

Shashwat Shrivastava (EPFL)

Dr. Grace Zgheib (EPFL/Intel)

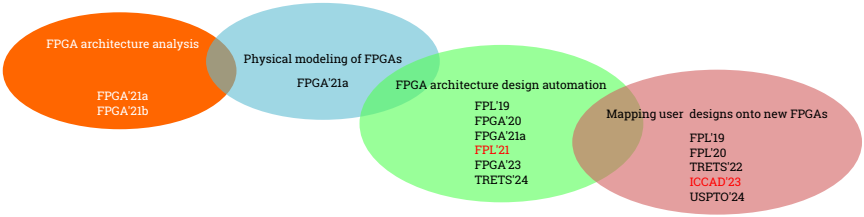
Prof. Dr. Francky Catthoor (Imec)

Dr. Zsolt Tőkei (Imec)

Chirag Ravishankar (Xilinx/AMD)

Dr. Dinesh Gaitonde (Xilinx/AMD)

Work that we have done so far



USPTO'24: Dinesh Gaitonde, Chirag Ravishankar, and Stefan Nikolić. Runtime efficient multi-stage router flow for circuit designs (patent application no. US 20240202423 A1; published, pending approval)
TRETS'24: Stefan Nikolić and Paolo Ienne. Exploring FPGA switch-blocks without explicit pattern listing
ICCAD'23: Shashwat Shrivastava, Stefan Nikolić, Chirag Ravishankar, Dinesh Gaitonde, and Mirjana Stojilović. IIBLAST: Speeding up commercial FPGA routing by decoupling and mitigating the intra-CLB bottleneck
FPGA'23: Stefan Nikolić and Paolo Ienne. Regularity matters: Designing practical FPGA switch-blocks
TRETS'22: Stefan Nikolić, Grace Zgheib, and Paolo Ienne. Detailed placement for dedicated LUT-level FPGA interconnect
FPL'21: Stefan Nikolić and Paolo Ienne. Turning PathFinder upside-down: Exploring FPGA switch-blocks by negotiating switch presence (Best Paper Award)
FPGA'21a: Stefan Nikolić, Francky Catthoor, Zsolt Tőkei, and Paolo Ienne. Global is the new local: FPGA architecture at 5nm and beyond
FPGA'21b: Morten B. Petersen, Stefan Nikolić, and Mirjana Stojilović. NetCracker: A peek into the routing architecture of Xilinx 7-Series FPGAs
FPL'20: Stefan Nikolić, Grace Zgheib, and Paolo Ienne. Timing-driven placement for FPGA architectures with dedicated routing paths (Best Paper Award)
FPGA'20: Stefan Nikolić, Grace Zgheib, and Paolo Ienne. Straight to the point: Intra- and intercluster LUT connections to mitigate the delay of programmable routing
FPL'19: Stefan Nikolić, Grace Zgheib, and Paolo Ienne. Finding a needle in the haystack of hardened interconnect patterns (short paper)

And I would especially like to thank Vukašin for organizing all this

TU Darmstadt > CNS > Team > Vukašin Karadžić

Vukašin Karadžić

CNS - Cryptography and Network Security

Contact

vukasin.karadzic@tu-...

+49 6151 16-20668

S2|20 118
Pankratiusstraße 2
64289 Darmstadt

Why would one want custom hardware?

General-purpose hardware is no longer sufficient

RESEARCH

REVIEW SUMMARY

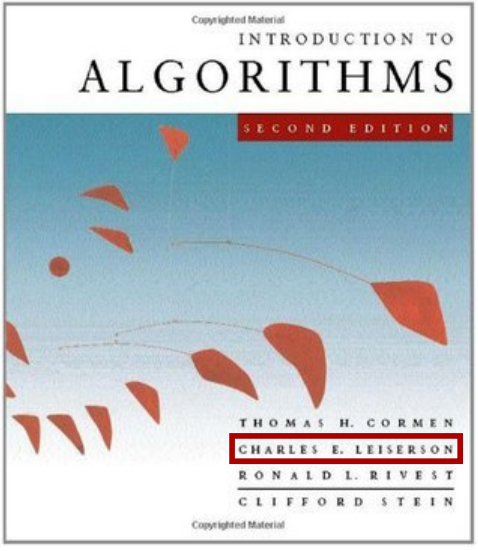
Leiserson *et al.*, *Science* **368**, 1079 (2020) 5 June 2020

COMPUTER SCIENCE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

Charles E. Leiserson, Neil C. Thompson*, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, Tao B. Schardl

For the people coming from a more theoretical background



General-purpose hardware is no longer sufficient

Much of the improvement in computer performance comes from decades of miniaturization of computer components, a trend that was foreseen by the Nobel Prize–winning physicist Richard Feynman in his 1959 address, "There's Plenty of Room at the Bottom," to the American Physical Society. In 1975, Intel founder Gordon Moore predicted the regularity of this miniaturization trend, now called Moore's law, which, until recently, doubled the number of transistors on computer chips every 2 years.

Unfortunately, semiconductor miniaturization is running out of steam as a viable way to grow computer performance—there isn't much more room at the "Bottom." If growth in computing power stalls, practically all industries will face challenges to their productivity. Nevertheless, opportunities for growth in computing performance will still be available, especially at the "Top" of the computing-technology stack: software, algorithms, and hardware architecture.

+ Design-Technology Co-Optimization (DTCO)

Custom hardware can remove the unnecessary overhead

ON OUR WEBSITE

Read the full article at https://dx.doi.org/10.1126/science.aam9744

Hardware architectures can be streamlined—for instance, through processor simplification, where a complex processing core is replaced with a simpler core that requires fewer transistors. The freed-up transistor budget can then be redeployed in other ways—for example, by increasing the number of processor cores running in parallel, which can lead to large efficiency gains for problems that can exploit parallelism. Another form of streamlining is domain specialization, where hardware is customized for a particular application domain. This type of specialization jettisons processor functionality that is not needed for the domain. It can also allow more customization to the specific characteristics of the domain, for instance, by decreasing floating-point precision for machine-learning applications.

And there is plenty of it

ISSCC 2014 / SESSION 1 / PLENARY / 1.1

1.1 Computing's Energy Problem (and what we can do about it)

Mark Horowitz

Departments of Electrical Engineering and Computer Science,
Stanford University, Stanford, CA

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

Really useful

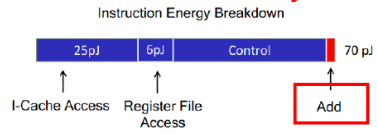


Figure 1.1.9: Rough energy costs for various operations in 45nm 0.9V.

Custom hardware also allows better use of silicon by enabling data width tailoring

ON OUR WEBSITE

Read the full article at https://dx.doi.org/10.1126/science.aam9744

Hardware architectures can be streamlined—for instance, through processor simplification, where a complex processing core is replaced with a simpler core that requires fewer transistors. The freed-up transistor budget can then be redeployed in other ways—for example, by increasing the number of processor cores running in parallel, which can lead to large efficiency gains for problems that can exploit parallelism. Another form of streamlining is domain specialization, where hardware is customized for a particular application domain. This type of specialization jettisons processor functionality that is not needed for the domain. It can also allow more customization to the specific characteristics of the domain, for instance, by decreasing floating-point precision for machine-learning applications.

For instance, area of an integer multiplier is typically $\mathcal{O}(n^2)$

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

$$\frac{0.2}{8^2} \times 32^2 = 3.2$$

Smaller area of a compute unit = more compute units working in parallel

ON OUR WEBSITE

Read the full article at https://dx.doi.org/10.1126/science.aam9744

Hardware architectures can be streamlined—for instance, through processor simplification, where a complex processing core is replaced with a simpler core that requires fewer transistors. The freed-up transistor budget can then be redeployed in other ways—for example, by increasing the number of processor cores running in parallel, which can lead to large efficiency gains for problems that can exploit parallelism. Another form of streamlining is domain specialization, where hardware is customized for a particular application domain. This type of specialization jettisons processor functionality that is not needed for the domain. It can also allow more customization to the specific characteristics of the domain, for instance, by decreasing floating-point precision for machine-learning applications.

This is particularly appealing for machine learning

Binarized Neural Networks

Itay Hubara[1*] itayh@technion.ac.il
Matthieu Courbariaux[2*] matthieu.courbariaux@gmail.com
Daniel Soudry[3] daniel.soudry@gmail.com
Ran El-Yaniv[1] rani@cs.technion.ac.il
Yoshua Bengio[2,4] yoshua.umontreal@gmail.com

(1) Technion, Israel Institute of Technology.
(2) Université de Montréal.
(3) Columbia University.
(4) CIFAR Senior Fellow.
(*) Indicates equal contribution.

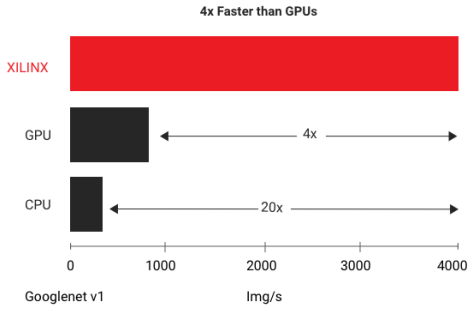
This is particularly appealing for machine learning

Vitis AI in the Data Center

AMD delivers the highest throughput at the lowest latency. In standard benchmark tests on GoogleNet V1, the AMD Alveo U250 platform delivers more than 4x the throughput of the fastest existing GPU for real-time inference. Learn more in the whitepaper: Accelerating DNNs with AMD Alveo Accelerator Cards

AI in the Data Center eBook

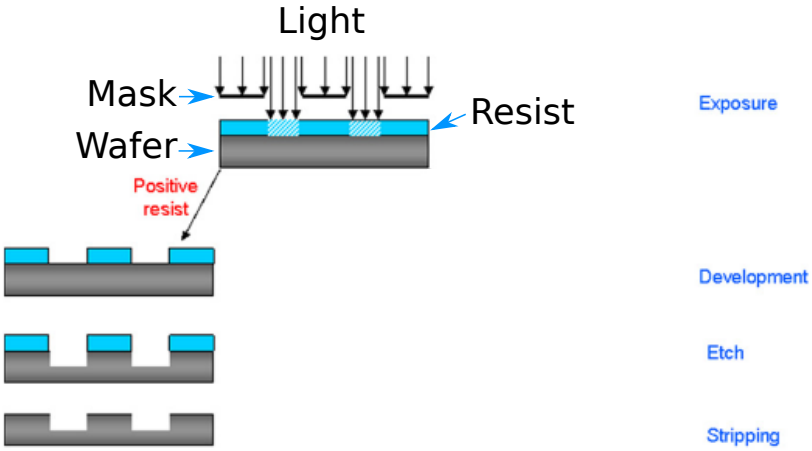
Download eBook



How do we make custom hardware?

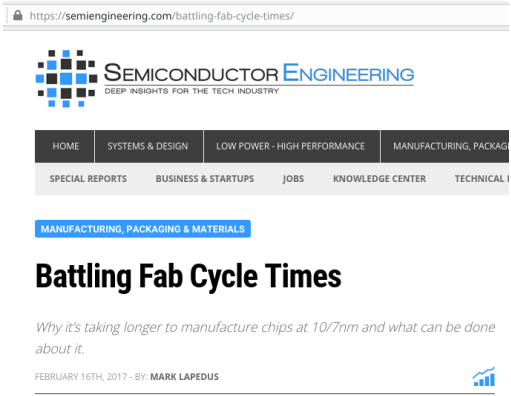
Traditional way: Application-Specific Integrated Circuits (ASICs)

Traditional way: Application-Specific Integrated Circuits (ASICs)



Courtesy of Bénédicte Mortini and Elsevier Masson SAS

Traditional way: Application-Specific Integrated Circuits (ASICs)



https://semiengineering.com/battling-fab-cycle-times/

SEMICONDUCTOR ENGINEERING
DEEP INSIGHTS FOR THE TECH INDUSTRY

HOME | SYSTEMS & DESIGN | LOW POWER - HIGH PERFORMANCE | MANUFACTURING, PACKAG

SPECIAL REPORTS | BUSINESS & STARTUPS | JOBS | KNOWLEDGE CENTER | TECHNICAL

MANUFACTURING, PACKAGING & MATERIALS

Battling Fab Cycle Times

Why it's taking longer to manufacture chips at 10/7nm and what can be done about it.

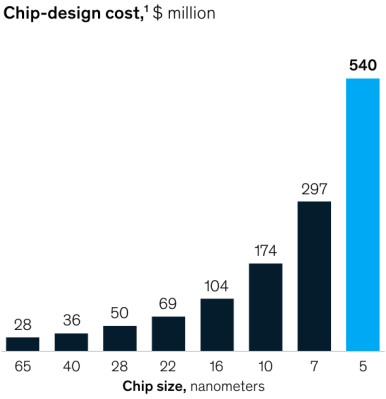
FEBRUARY 16TH, 2017 - BY: **MARK LAPEDUS**

The shift from planar devices to finFETs enables chipmakers to scale their processes and devices from 16nm/14nm and beyond, but the industry faces several challenges at each node.

Generally, the most common metric for cycle time in the fab is "days per mask layer." On average, a fab takes 1 to 1.5 days to process a layer. The best fabs are down to 0.8 days, Leachman said.

A 28nm device has 40 to 50 mask layers. In comparison, a 14nm/10nm device has 60 layers, with 7nm expected to jump to 80 to 85. 5nm could have 100 layers. So, using today's lithographic techniques, the cycle times are increasing from roughly 40 days at 28nm, to 60 days at 14nm/10nm, to 80 to 85 days at 7nm. 5nm may extend to 100 days using today's techniques, without extreme ultraviolet (EUV) lithography.

Problem: accessing new technologies is increasingly costly



McKinsey & Company, 2020

Only a few companies with large production volumes (or for whom computation is a highly-profitable service) can afford custom silicon at cutting-edge nodes

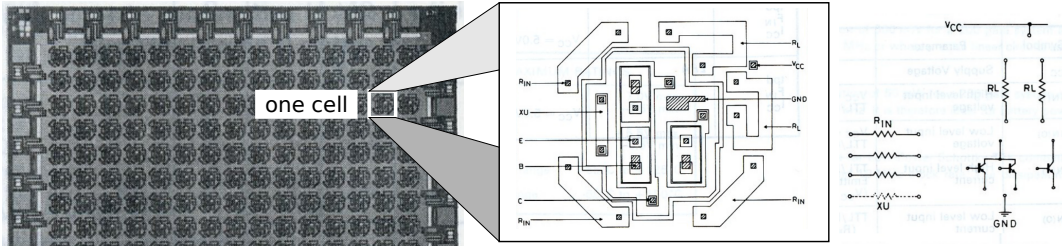
But, this is not a new problem

Centuries old principle: If you are rich, you get to buy handcrafted products

Centuries old principle: If you are poor, you get to buy the mass-produced alternatives

But, the beauty of the house is less important if one is happy with the neighbors

Removing the masks: Step 1—Mask Programmed Gate-Array (MPGA)

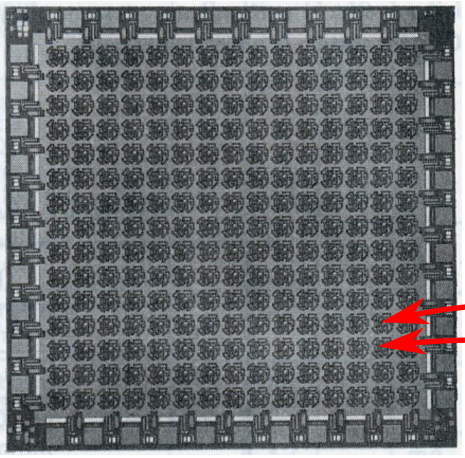


Ferranti ULA 2000, 1977

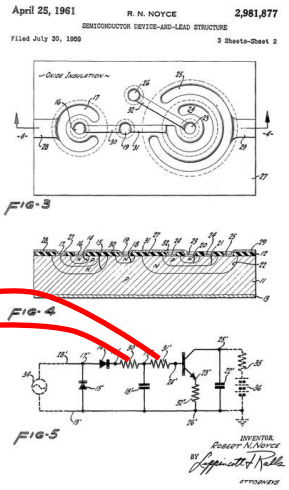
Array of identical cells

- Layout of the sole cell can be highly optimized
- Easy capacity variation

Implementing a circuit on an MPGA: Placement



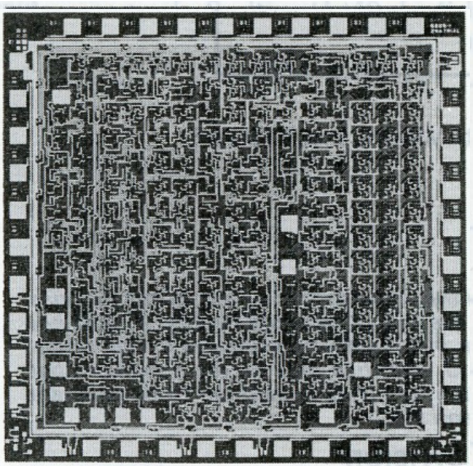
Ferranti ULA 2000, 1977



Placement:

Assign components of the circuit to the prefabricated ones, such that those that have to be connected are neighbors

Implementing a circuit on an MPGA: Routing

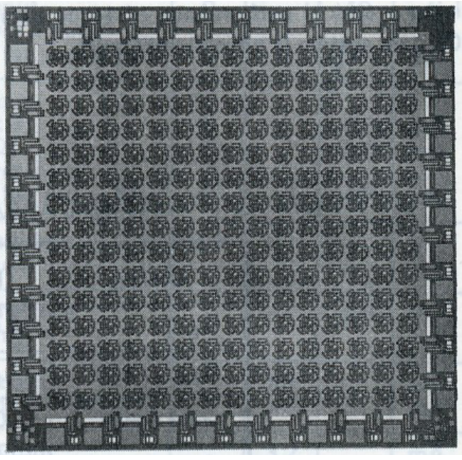


programmed gate array

Routing:

Connect placed components together by tracing metal wires

Single-mask specialization



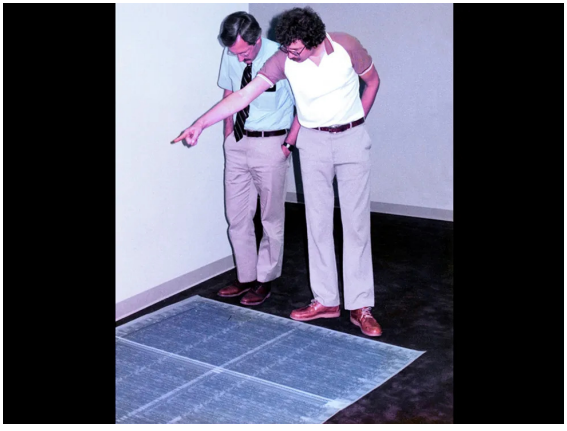
nonprogrammed gate array

1977

Ferranti ULA 2000

- only one additional metal mask
- NRE of only a couple $k
- produced in a couple of weeks

Ross Freeman inspecting the layout of the first FPGA



Oral History of Bill Carter

Computer History Museum

152K subscribers

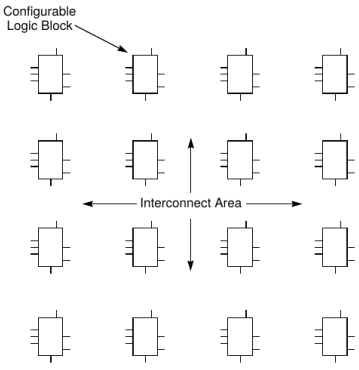
Can we do zero-mask specialization?
(i.e., without reentering the fab)

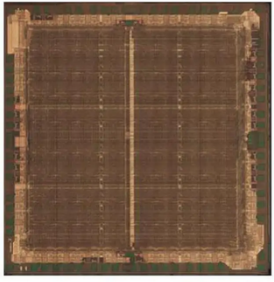
Xilinx XC2064 Logic Cell Array (Field-Programmable Gate Array)

XILINX

XC2000 Logic Cell Array Families

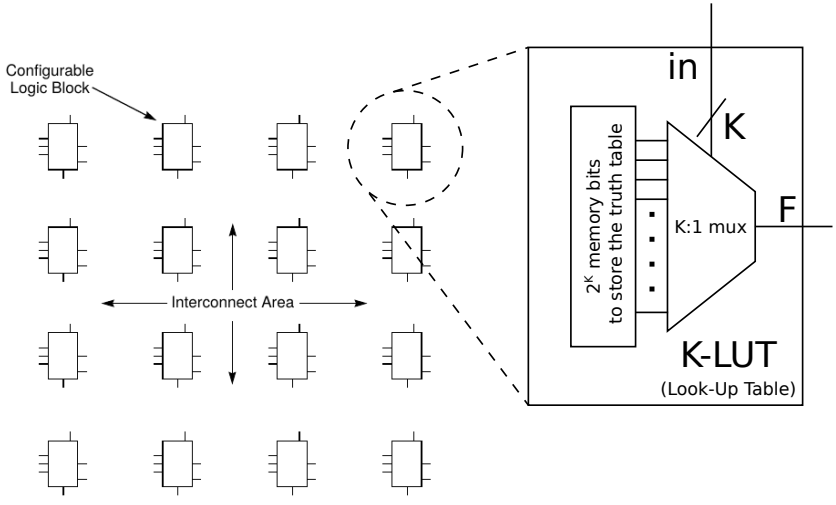
1985



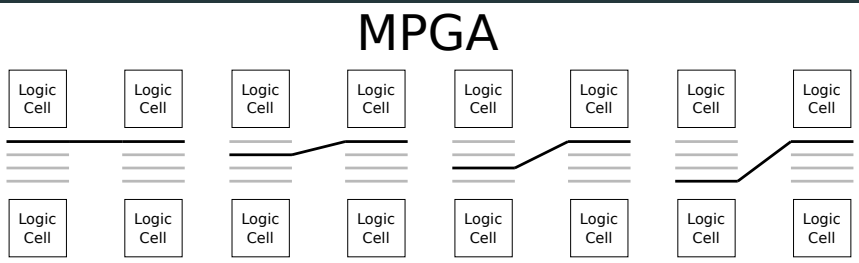


XC2064
8×8 tiles

FPGA: Identical generic cells

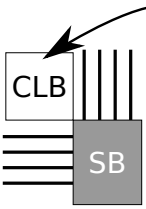


FPGA: routing channels

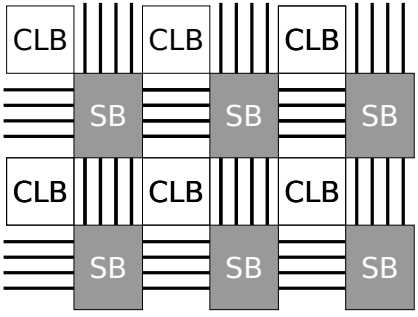


Island-Style FPGA

single tile

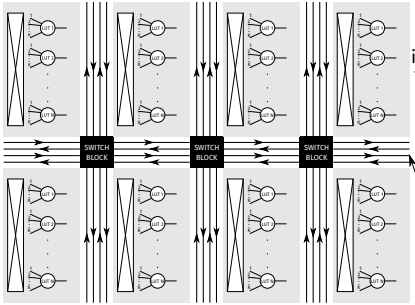


(a collection of LUTs and flip-flops sharing common local interconnect)



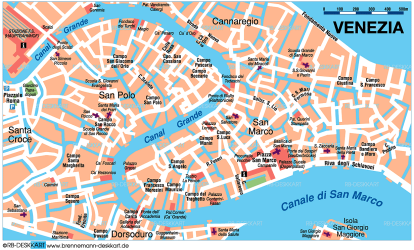
- switch-block (SB): collection of multiplexers feeding all channel wires in one tile
- connection-block (CB): collection of multiplexers feeding all CLB inputs in one tile

Island-Style FPGA



island

channel



Wasn’t MPGA good enough?

1977

Implementing a design on

Ferranti ULA 2000

- only one additional metal mask
- NRE of only a couple $k
- produced in a couple of weeks

1985

Implementing a design on

Xilinx XC2064

- zero-mask customization
- (close to) zero NRE
- (close to) instant "production"

Wasn’t MPGA good enough?

1977

Implementing a design on

Ferranti ULA 2000

- only one additional metal mask
- NRE of only a couple $k
- produced in a couple of weeks

1985

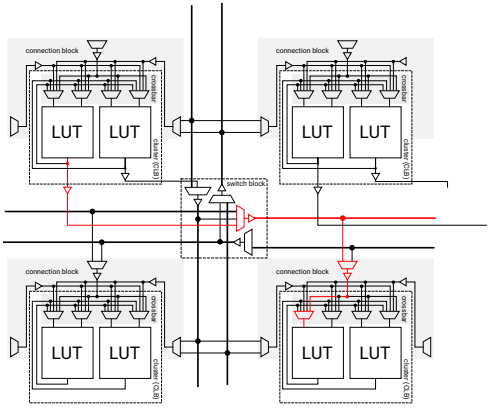
Implementing a design on

Xilinx XC2064

- zero-mask customization
- (close to) zero NRE
- (close to) instant "production"

Not for prototyping

Reconfigurability comes at a price



All these multiplexers are slow $\implies$ FPGA implementation

$\sim 2\times$ slower than MPGA and $\sim 10\times$ than standard-cell ASIC

So is FPGA good only for prototyping?!

Let’s turn to cryptography for an answer

Henk C.A. van Tilborg · Sushil Jajodia (Eds.)

Encyclopedia of Cryptography and Security

250× better performance despite 24× lower clock frequency

FPGA Field Programmable Gate Array

►Trusted Computing

FPGAs in Cryptography

Tim E. Güneysu
Department of Electrical Engineering and Information Technology, Ruhr-University Bochum, Bochum, Germany

An application perfectly suited for FPGAs (i.e., fulfilling all the mentioned criteria above) is the standardized Data Encryption Standard (DES) block cipher. DES was specifically designed for optimal hardware efficiency and thus uses only straightforward low-level bit operations and only a negligible amount of memory. A pipelined implementation of DES based on four separate processors on a low-cost Xilinx Spartan-3 FPGA (currently approx. US$20–30) can provide 500 million encryptions per second at 125 MHz. On the contrary, a single-threaded software implementation of DES on an Intel Pentium 4 running at 3 GHz (Prescott) computes roughly two million encryptions per second. This demonstrates the impressive cost–performance advantage that FPGAs can gain with respect to software solutions for specific designs.

Accelerating an entire workload is more difficult, but still possible

2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)

Xuantie-910: A Commercial Multi-Core 12-Stage Pipeline Out-of-Order 64-bit High Performance RISC-V Processor with Vector Extension

Industrial Product

Chen Chen, Xiaoyan Xiang, Chang Liu, Yunhai Shang, Ren Guo, Dongqi Liu, Yimin Lu, Ziyi Hao, Jiahui Luo, Zhijian Chen, Chunqiang Li, Yu Pu, Jianyi Meng, Xiaolang Yan, Yuan Xie and Xiaoning Qi*

The T-Head Division, Alibaba Cloud
Email: jianyi.mjy@alibaba-inc.com

scale is limited to several hundreds. The implementation on the Xilinx VU9P FPGA runs at 200 MHz frequency in Linux OS. Taking blockchain transaction acceleration as an example, in terms of per-core performance, the FPGA edition is still 20% higher than the x86_64 Intel Xeon Platinum 8163 CPU that runs at 2.5 GHz in ubuntu16.04 OS. A cost-down ASIC edition has been taped out and the chip is expected in July, 2020. It is projected to run at a frequency of 2.0 - 2.5 GHz, resulting in 12-15X higher performance than the x86_64 Intel Xeon Platinum 8163 CPU counterpart. In addition to internal use,

FPGAs let us implement circuits that we cannot buy

FPGAs let us implement circuits that we cannot buy

FPGAs let us implement circuits that we cannot buy

FPGAs let us implement circuits that we cannot buy

Highly-repeatable structures allow them to be at the technological forefront

Proceedings of the IEEE | Vol. 103, No. 3, March 2015

Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology

This paper reflects on how Moore's Law has driven the design of FPGAs through three epochs: the age of invention, the age of expansion, and the age of accumulation.

By Stephen M. (Steve) Trimberger, *Fellow IEEE*

Pioneering the fabless business model, FPGA startup companies typically could not obtain leading-edge silicon technology in the early 1990s. As a result, FPGAs began the Age of Expansion lagging the process introduction curve. In the 1990s, they became process leaders as the foundries realized the value of using the FPGA as a process-driver application. Foundries were able to build SRAM FPGAs as soon as they were able to yield transistors and wires in a new technology. FPGA vendors sold their huge devices while foundries refined their processes. Each new generation of silicon

Hence an FPGA implementation could actually be faster than an ASIC implementation in an older technology that a user could afford

But reconfigurability goes beyond poor man's ASICs

Automated Masking of FPGA-Mapped Designs

Nicolai Müller*, Sergej Meschkov†, Dennis R. E. Gnad†, Mehdi B. Tahoori†, Amir Moradi*
* *Horst Görtz Institute for IT Security, Ruhr University Bochum, Bochum, Germany*
{nicolai.mueller, amir.moradi}@rub.de
† *Institute of Computer Engineering, Karlsruhe Institute of Technology, Karlsruhe, Germany*
{sergej.meschkov, dennis.gnad, mehdi.tahoori}@kit.edu

I. Introduction

Field Programmable Gate Arrays (FPGAs) play an important role in security-critical systems, as their reconfigurability enables security updates directly in the hardware and throughout the system's lifetime [1]. Hence, the overall system's security and flexibility are greatly increased by FPGAs. More concretely, if an FPGA implements a security-related operation, e.g. a cryptographic primitive operating on sensitive data, the circuit is updatable in case of a security flaw. However,

The work described in this paper has been supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and through the project 435264177 (SAUBER).

It also enables customizing hardware to a problem INSTANCE

Direct Spatial Implementation of Sparse Matrix Multipliers for Reservoir Computing

Matthew Denton and Herman Schmit
Google Brain
{myuzaki, schmit}@google.com

***Abstract*—Reservoir computing systems rely on the recurrent multiplication of a very large, sparse, fixed matrix. We argue that direct spatial implementation of these fixed matrices minimizes the work performed in the computation, and allows for significant reduction in latency and power through constant propagation and logic minimization. Bit-serial arithmetic enables massive static matrices to be implemented. We present the structure of our bit-serial matrix multiplier, and evaluate using canonical signed digit representation to further reduce logic utilization. We have implemented these matrices on a large FPGA and provide a cost model that is simple and extensible. These FPGA implementations, on average, reduce latency by 50x up to 86x versus GPU libraries. Comparing against a recent sparse DNN accelerator, we measure a 4.1x to 47x reduction in latency depending on matrix dimension and sparsity. Throughput of the FPGA solution is also competitive for a wide range of matrix dimensions and batch sizes. Finally, we discuss ways these**

to enable time-multiplexing of the multiplier unit to support different matrix dimensions. Having two operands can be costly from area and power. The primary power-saving technique for these multipliers is batching, where the matrix values are kept constant while performing multiple vector times matrix computations.

Reservoir computing, specifically Echo State Networks as discussed in this paper, uses very large, sparse matrices. These matrices are generated randomly and are never modified by training, and therefore the matrix is fixed for the lifetime of the computation. In order to handle these matrices, conventional ML accelerators perform indexing and tiling of the sparse matrix, which effectively transform the large sparse operation into multiple small dense operations. However, this transfor-

FPGA $\neq$ LUT array

Field-Programmable: enabled by programmable interconnect (it removed the last mask from MPGAs)

Gate Array: does not specify which gate (can be anything)

FPGAs do not need to be general-purpose

FPGA ≠ LUT array

Multi-input Serial Adders for FPGA-like Computational Fabric

Herman Schmit
Google Research
schmit@google.com

Matthew Denton
Google Research
myuzaki@google.com

ABSTRACT

In this paper, we present a new functional unit to replace the LUT in an FPGA-like computational fabric designed specifically for use to accelerate instance-specific sparse integer matrix multiplication. We use a suite of matrices, the VPR place-and-route tool, and modern architecture representations of the interconnect to examine this architectural idea. The new cell, called the K–ADD, increases density by 2.5× to 4×, and increases performance by 8% to 30% by simultaneously increasing the clock rate and reducing the number of cycles to compute the product. This benefit magnifies the two-orders-of-magnitude advantage of using instance-specific matrix multipliers demonstrated in prior work. We investigate the cluster

key is used to reduce the logic size and depth [13], and Boolean Satisfiability solvers that are specialized to a set of equations [23].

An FPGA-based solution was recently presented that uses an instance-specific configuration for a large sparse matrix-vector multiplication (SpMV) [7, 8]. That paper demonstrated one- and two-order of magnitude speedups in the inference speed of a recurrent system compared to GPU based solutions. In this particular application, the large matrix is permanently fixed. The overheads for design and configuration therefore can be amortized. That paper noted that the LUTs of the FPGA are underutilized: only three of the six inputs to each LUT were used, and one of those three inputs is local feedback for the state of serial adder.

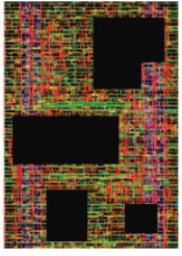
FPGAs open new avenues for obfuscation

978-3-9819263-5-4/DATE21/©2021 EDAA

Hardware Redaction via Designer-Directed Fine-Grained eFPGA Insertion

Prashanth Mohan, Oguz Atli, Joseph Sweeney, Onur Kibar, Larry Pileggi and Ken Mai
Department of Electrical and Computer Engineering, Carnegie Mellon University
Pittsburgh, PA, USA
{*pmohan, aatli, jsweene1, okibar, pileggi, kenmai*}*@andrew.cmu.edu*

***Abstract*—In recent years, IC reverse engineering and IC fabrication supply chain security have grown to become significant economic and security threats for designers, system integrators, and end customers. Many of the existing logic locking and obfuscation techniques have shown to be vulnerable to attack once the attacker has access to the design netlist either through reverse engineering or through an untrusted fabrication facility. We introduce soft embedded FPGA redaction, a hardware obfuscation approach that allows the designer substitute security-critical IP blocks within a design with a synthesizable eFPGA fabric. This method fully conceals the logic and the routing of the critical IP and is compatible with standard ASIC flows for easy integration and process portability. To demonstrate eFPGA redaction, we obfuscate a RISC-V control path and a GPS P-code generator. We also show that the modified netlists are resilient to SAT attacks with moderate VLSI overheads. The secure RISC-V design has**



Connection between FPGAs and security spans many years

Cryptographic cores have been used in fundamental FPGA architecture research

Logic Synthesis and Optimization Benchmarks User Guide
Version 3.0

Saeyang Yang[1]

Circuit Name	Circuit Function	Inputs	Outputs	Latches	Approx. Gates
bigkey	Key Encription	262	197	221	4765
clma	Bus Interface	382	82	33	35000
clmb	Bus Interface	382	0	33	35000
dsip	Encription Circuit	228	197	224	2097
mm30a	Minmax Circuit	33	30	90	1549
mm4a	Minmax Circuit	7	4	12	153
mm9a	Minmax Circuit	12	9	27	492

MCNC benchmark suite from 1991 (still used today)

Circuit Name	Circuit Function	Inputs	Outputs	Approx. Gates
des	Data Encription	256	245	4000
example2	Logic	85	66	277
f51ml	Arithmetic	8	8	43
frg1	Logic	28	3	105
frg2	Logic	143	139	1004

How Much Logic Should Go in an FPGA Logic Block?

VAUGHN BETZ
JONATHAN ROSE
University of Toronto

ALL FIELD-PROGRAMMABLE GATE arrays contain both programmable logic blocks and programmable routing. The nature of the logic block strongly influences an FPGA's input lookup tables. Notice that many connections can be made via the local interconnect within a cluster. Because this local interconnect can be faster than the general-

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

Vaughn Betz and Jonathan Rose
Department of Electrical and Computer Engineering, University of Toronto
Toronto, Ontario, Canada M5S 3G4
{vaughn, jayar}@eecg.toronto.edu

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

Elias Ahmed
Dept. of Electrical & Computer Engineering
University of Toronto
Toronto, Canada
elias@eecg.toronto.edu

Jonathan Rose
Dept. of Electrical & Computer Engineering
University of Toronto
Toronto, Canada
jayar@eecg.toronto.edu

Rethinking FPGAs: Elude the Flexibility Excess of LUTs with And-Inverter Cones

Hadi Parandeh-Afshar
hadi.parandehafshar@epfl.ch

Hind Benbihi
hind.benbihi@epfl.ch

David Novo
david.novobruna@epfl.ch

Paolo Ienne
paolo.ienne@epfl.ch

Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences, 1015 Lausanne, Switzerland

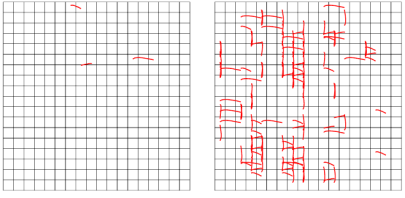
We have used them too

Straight to the Point: Intra- and Intercluster LUT Connections to Mitigate the Delay of Programmable Routing

Stefan Nikolić
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
stefan.nikolic@epfl.ch

Grace Zgheib
Intel Corporation
San Jose, USA
grace.zgheib@intel.com

Paolo Ienne
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
paolo.ienne@epfl.ch



(a) Before permutation. **(b) After permutation.**

Figure 3: Influence of LUT permutation on usability of direct connections. The figure shows the same placement of the *sha* benchmark from the *VTR* set (a) before and (b) after permuting the LUTs inside the clusters. Each cell is a location on the fabric grid. Connections implemented as direct are shown in red.

FPGAs have been used for early evaluation of new cryptographic algorithms

Community Awards

Here we collect an archive of awards that have been bestowed by conferences in our community as well to members of our community.

- FPGA Best Paper Awards
- FCCM Best Paper Awards
- FPL Best Paper Awards
- FPT Best Paper Awards
- ACM TRETS Best Paper Awards

2010, Milan, Italy

MS: **Enhancing FPGA Device Capabilities by the Automatic Logic Mapping to Additive Carry Chains**
Thomas B. Preusser and Rainer G. Spallek

SV: **FPGA-Optimised Uniform Random Number Generators Using LUTs and Shift Registers**
David B. Thomas and Wayne Luk

Community: **ATHENa - Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware Using FPGAs**
Kris Gaj, Jens-Peter Kaps, Venkata Amirineni, Marcin Rogawski, Ekawat Homsirikamol and Benjamin Y. Brewster

And this still continues today

H-Saber: An FPGA-Optimized Version for Designing Fast and Efficient Post-Quantum Cryptography Hardware Accelerators

Andrea Guerrieri[‡*], Gabriel Da Silva Marques[‡], Francesco Regazzoni[†§], and Andres Upegui[‡]
[‡]University of Applied Sciences and Arts Western Switzerland, Geneva, Switzerland
[*]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[§]University of Amsterdam, Amsterdam, The Netherlands
[†]Università della Svizzera Italiana, Lugano, Switzerland

High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber

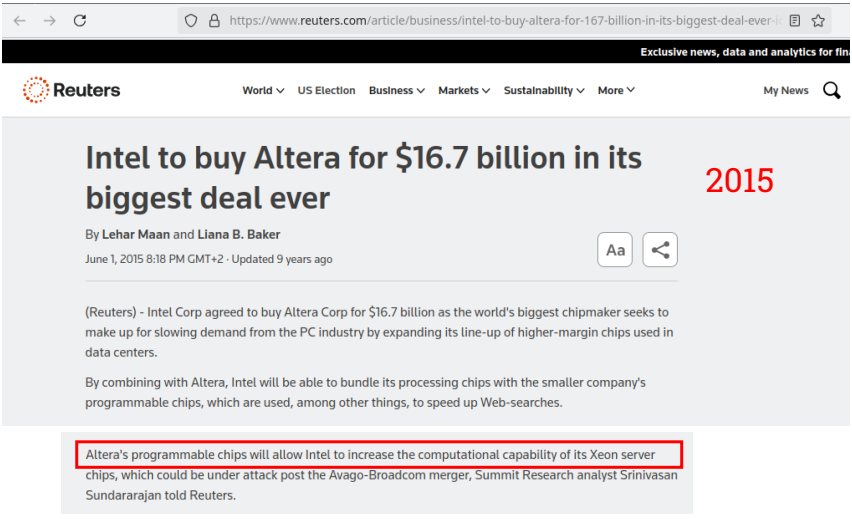
Viet Ba Dang, Kamyar Mohajerani, and Kris Gaj

Abstract—Post-Quantum Cryptography (PQC) has emerged as a response of the cryptographic community to the danger of attacks performed using quantum computers. All PQC schemes can be implemented in software and hardware using conventional (non-quantum) computing systems. PQC is the biggest revolution in cryptography since the invention of public-key schemes in the mid-1970 s. Lattice-based key exchange schemes have emerged as leading candidates in the NIST PQC standardization process due to their relatively short public keys and ciphertexts. This paper presents novel high-speed hardware architectures for four lattice-based Key Encapsulation Mechanisms (KEMs) representing three NIST PQC finalists: NTRU (with two distinct variants, NTRU-HPS and NTRU-HRSS), CRYSTALS-Kyber, and Saber. We benchmark these candidates in terms of their performance and resource utilization in today's FPGAs. Our best architectures outperform the best designs from other groups reported to date in terms of the area-time product by factors ranging from 1.01 to 2.88, depending on the algorithm and security level. Additionally, our study demonstrates that CRYSTALS-Kyber and Saber have very similar hardware performance. Both outperform NTRU in terms of execution time by a factor 36-62 for key generation and 3-7 for decapsulation, assuming the same security level.

Index Terms—FPGA, hardware, high-speed, key encapsulation mechanism, lattice-based, post-quantum cryptography

Why isn’t everybody using FPGAs by now?

For some time it appeared that we would all be using them



For some time it appeared that we would all be using them

Reuters World US Election Business Markets Sustainability More My News

AMD closes record chip industry deal with estimated $50 billion purchase of Xilinx

2022

By Jane Lee

February 14, 2022 7:51 PM GMT+1 · Updated 2 years ago

Fast forward to today

Reuters

World ⌄ US Election Business ⌄ Markets ⌄ Sustainability ⌄ More ⌄ My News

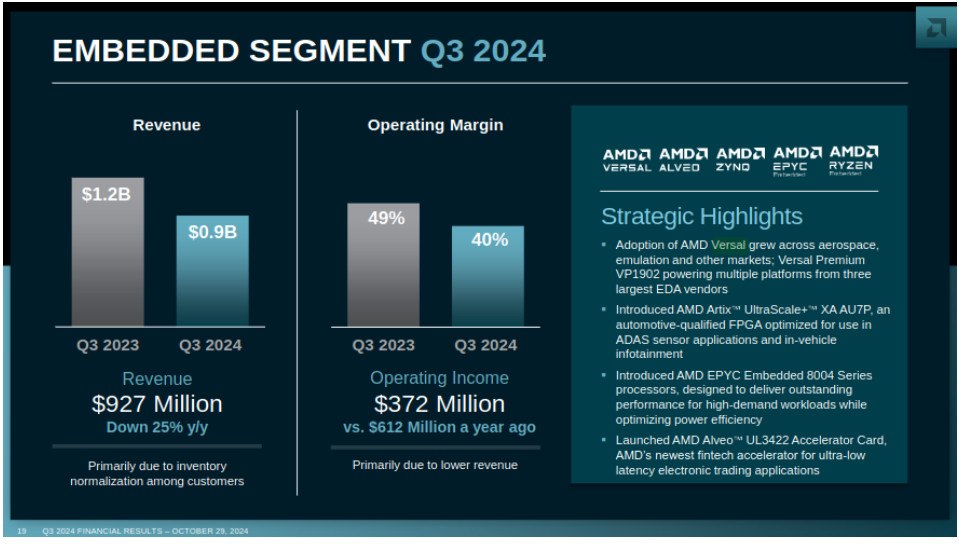
Intel to spin out programmable chip unit, hold IPO; shares rise 2%

By **Stephen Nellis** and **Samrhitha A**

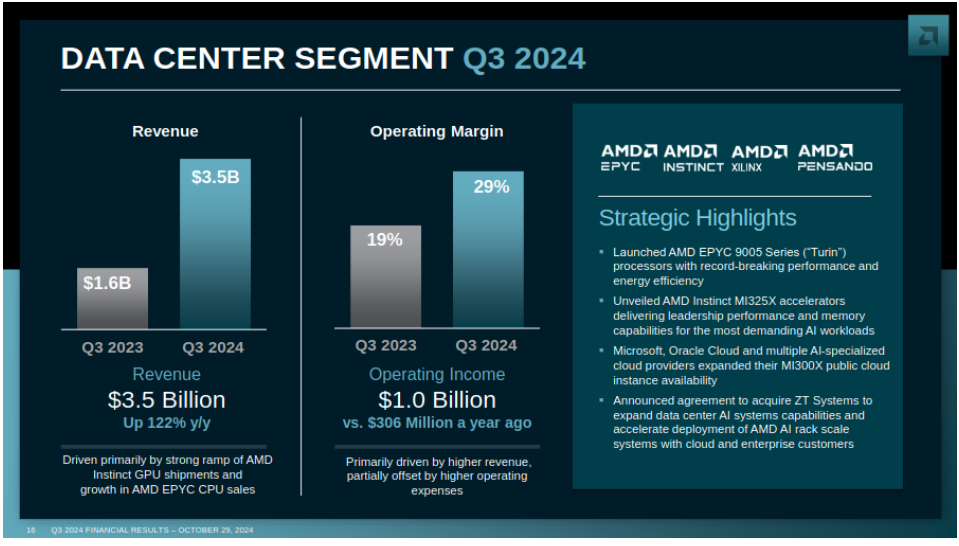
October 4, 2023 12:06 AM GMT+2 · Updated a year ago

Aa

Intel is struggling, but situation is not great at AMD either

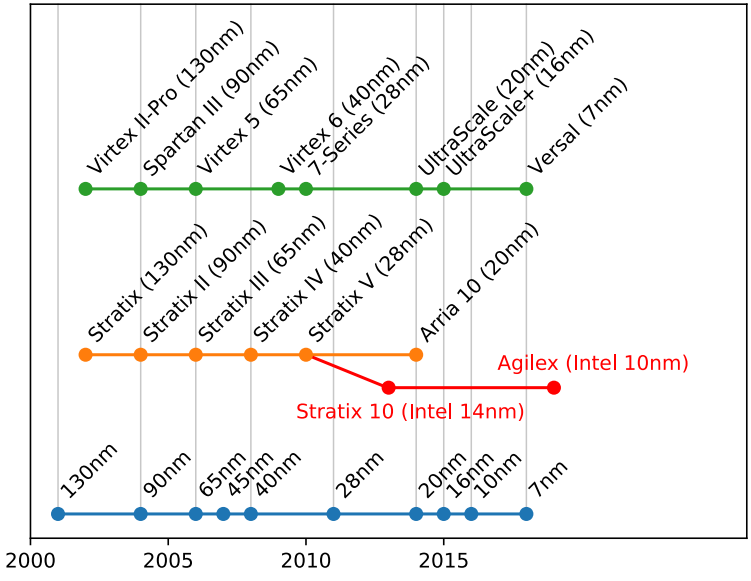


Meanwhile, the GPUs are doing great

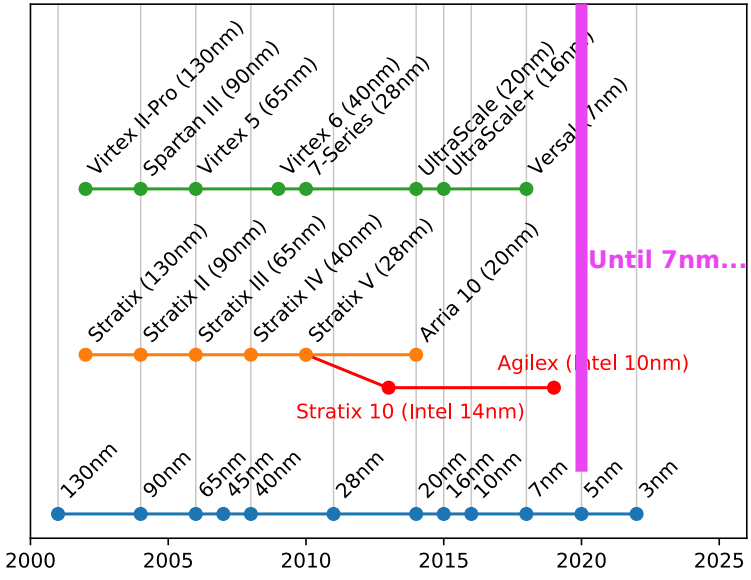


Why is that so?

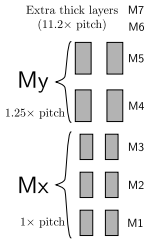
FPGAs really were at the technological forefront



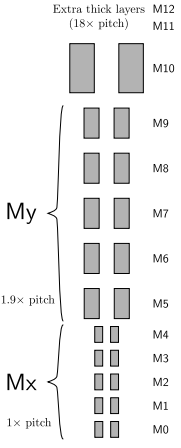
FPGAs really were at the technological forefront



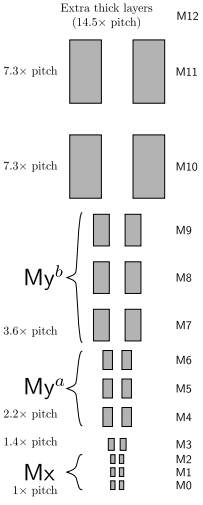
Shrinking wires are to blame



16nm

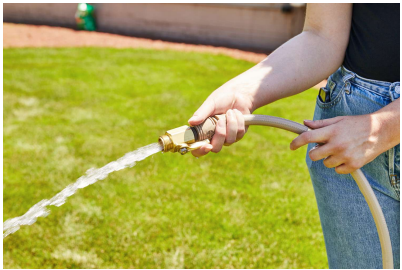


7nm



3nm

Wire-Hose Analogy



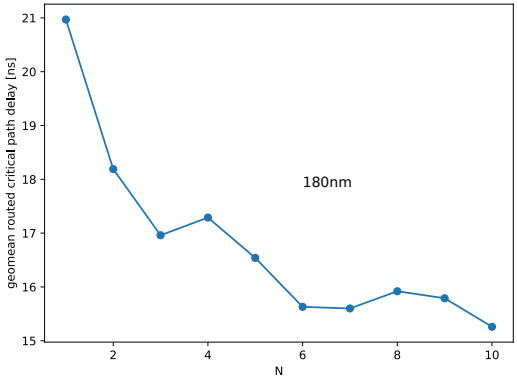
smaller cross-sectional area = higher resistance
(more time to fill the bucket)

larger cross-sectional area = lower resistance
(less time to fill the bucket)

High resistance breaks decades-old rules of thumb

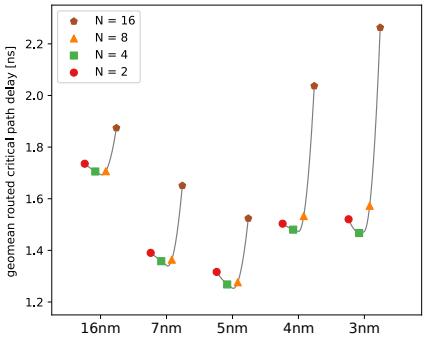
Routed delay as a function of cluster size

Old technology



Ahmed, "The effect of logic block granularity", University of Toronto, 2001

New technologies

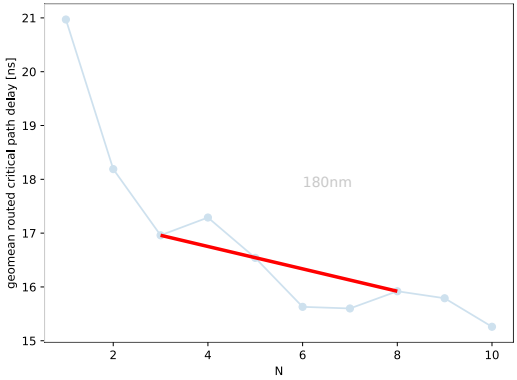


Nikolić, Catthoor, Tőkei, Ienne, "Global is the new local", FPGA'21

High resistance breaks decades-old rules of thumb

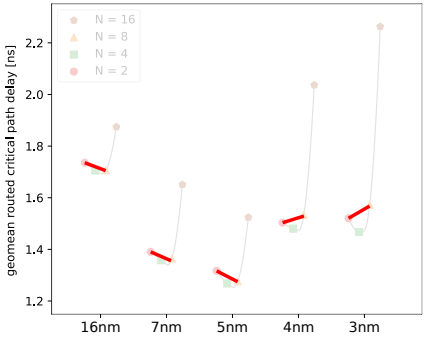
Routed delay as a function of cluster size

Old technology



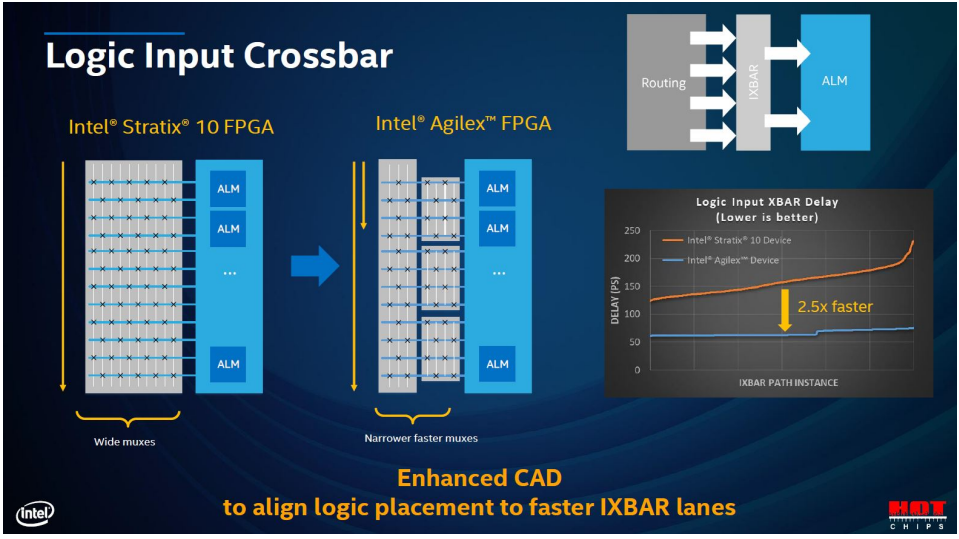
Ahmed, "The effect of logic block granularity", University of Toronto, 2001

New technologies

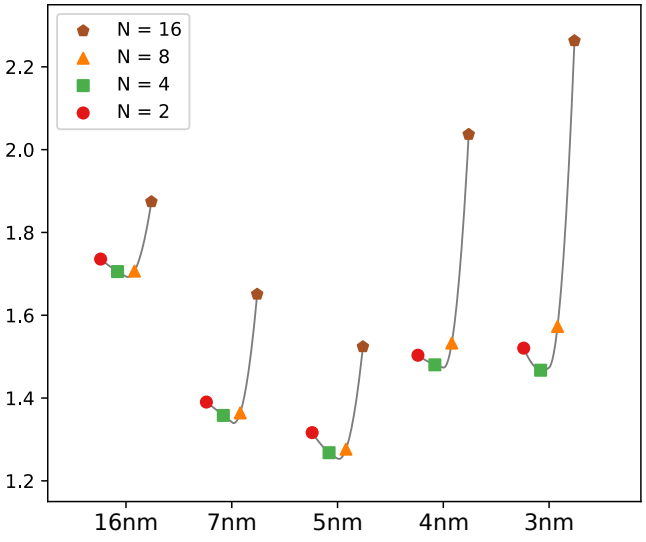


Nikolić, Catthoor, Tőkei, Ienne, "Global is the new local", FPGA'21

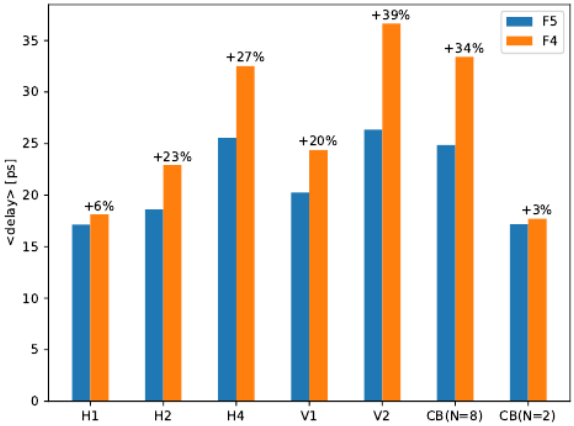
And this can be observed in latest 7nm commercial FPGAs too



Without further redesign, FPGA performance worsens in new technologies



Longer wires suffer larger delay increase with technology scaling



Since FPGAs are much less dense than ASIC, they suffer more from scaling effects

Measuring the Gap Between FPGAs and ASICs

FPGA'06

Ian Kuon and Jonathan Rose
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON
{ikuon,jayar}@eecg.utoronto.ca

ABSTRACT

This paper presents experimental measurements of the differences between a 90nm CMOS FPGA and 90nm CMOS Standard Cell ASICs in terms of logic density, circuit speed and power consumption. We are motivated to make these measurements to enable system designers to make better informed choices between these two media and to give insight to FPGA makers on the deficiencies to attack and thereby improve FPGAs. In the paper, we describe the methodology by which the measurements were obtained and we show that, for circuits containing only combinational logic and flip-flops, the ratio of silicon area required to implement them in FPGAs and ASICs is on average 40. Modern FPGAs also contain "hard" blocks such as multiplier/accumulators and block memories and we find that these blocks reduce this average area gap significantly to as little as 21. The ratio

Mohamed S. Abdelfattah
Vaughn Betz
University of Toronto

The Case for Embedded Networks on Chip on Field-Programmable Gate Arrays

Micro'14

Table 1. Summary of mixed and hard FPGA NoCs at 65 nm.

Feature		Mixed NoCs	Hard NoCs
Description		Hard routers, soft links	Hard routers, hard links
Special feature		Configurable topology	Low-voltage (low-V) mode
Comparison to soft NoCs	Area	20× smaller	23× smaller
	Speed	5× faster	6× faster
	Power	9× less power	11× less power (15× low-V)
Frequency		730 MHz	910 MHz
Critical path		Soft interconnect	Switch allocator in router

For FPGAs to be able to continue to democratize access to new technologies

We need to do something to make the wires faster

Automating programmable interconnect design

Design questions

When designing an FPGA, we need to answer questions such as how many wires of each type a routing channel should have and which wires should be connectable by a programmable switch

How have these questions been answered before in industry?

- Primary goal: never be late for the next technology node

How have these questions been answered before in industry?

- Primary goal: never be late for the next technology node
- Rely heavily on intuition
- Minimize changes in the interconnect as much as possible

How have these questions been answered before in industry?

Architectural Enhancements in Stratix V™

David Lewis*, David Cashman*, Mark Chan, Jeffery Chromczak*, Gary Lai, Andy Lee, Tim Vanderhoek*, Haiming Yu
Altera Corporation, 101 Innovation Drive, San Jose, CA, 95134
(*) Altera Corporation, 150 Bloor St W., Suite 400, Toronto, Ont., Canada M5S 2X9
{dlewis,dcashman,mchan,jchrmomcz,glai,alee,tvanderh,hyu}@altera.com

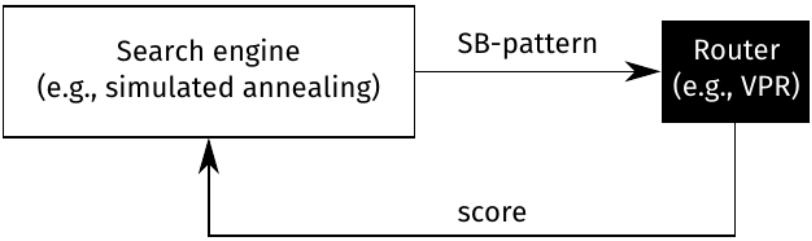
Although relatively small, the modularity results in a desire to accommodate a moderate increase in routing demand with adjustments to wire length, rather than the coarser quantization of increasing columns of routing multiplexers. Consequently rather than completely re-architect the routing, we explored minor variations that could keep pace with the increase in routing demand as well as obtain performance improvement.

How come did nobody try to automate programmable interconnect design before?

EDA tools were always developed for designing many chips

FPGA families traditionally share the same programmable interconnect architecture and there were always only a few vendors $\Longrightarrow$ no incentive to develop algorithms

What about academia?



[1] M. Lin, J. Wawrzynek, and A. El Gamal, “Exploring FPGA routing architecture stochastically”, TCAD’10

Pitfalls of the black-box approach

- Search space contains a huge number of possible solutions (e.g., 512 switches $\implies$ 2^{512} potential switch-patterns)
- Running router in the loop is very slow (usually minutes or hours; sometimes even more)

$\implies$ Covering any sizable portion of the search space is not feasible

How can we fix this?

Turning PathFinder Upside-Down: Exploring FPGA Switch-Blocks by Negotiating Switch Presence

Stefan Nikolić and Paolo Ienne
École Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences, 1015 Lausanne, Switzerland
{stefan.nikolic, paolo.ienne}@epfl.ch

FPL'21 (Best Paper Award)

Key observation

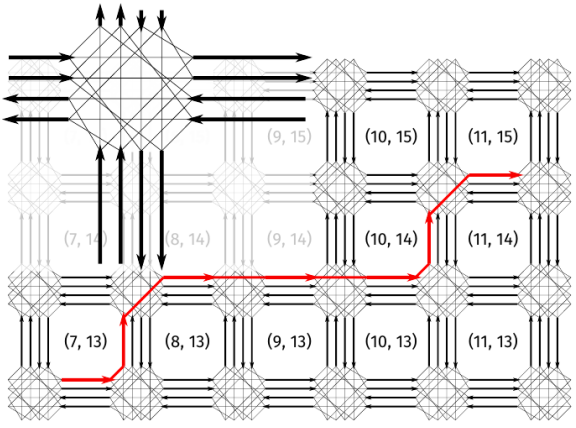
Router generates a lot of very useful information that is discarded if it is used as a black box

In the limit, we can “fool” it to make it design exactly the pattern that it needs

What does the router actually do?

The router implements all signals of the placed user circuit using nonintersecting paths composed of prefabricated wires and programmable switches

Normally, it sees only the wires and switches that exist in one particular FPGA



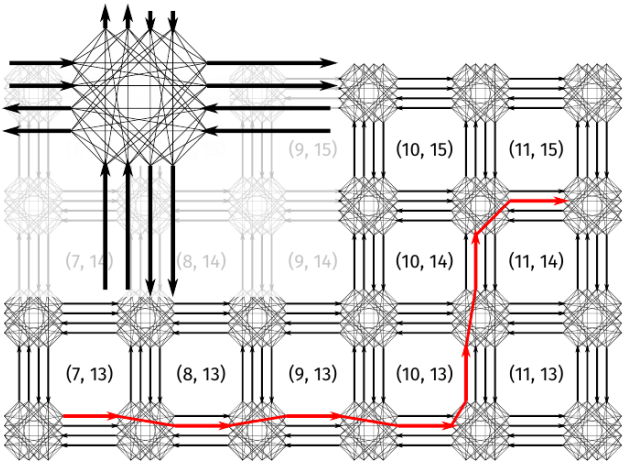
We could try to sparsify the architecture by doing the following

1. Count in how many switch-blocks a particular switch was used
2. Discard the least used ones
3. Until all relevant circuits are still routable

But how would we know if we should add some switches that do not exist in the particular architecture?

Classical approach would be to look at the results and rely on intuition or let simulated annealing make the modification

But if we implicitly represent all switches, the router can make the choices for us



This is a bit similar to deciding where to place pavements in a park

TU Delft OpenCourseWare

Themes | Programs | All Courses | DelftX MOOCs | More

Home > Courses > Next Generation Infrastructures > Course materials > Lectures > Inverse Infrastructures

Next Generation Infrastructures

Inverse Infrastructures

Self-organizing Inverse Infrastructures

We need to re-think infrastructures as necessarily being large-scale, centrally governed and owned by governments or companies. Not only that. As you have learned during this course, we may need to adapt to the idea that in the future infrastructures will increasingly be ad hoc and temporary in nature and emerge from bottom-up initiatives and from self-organization. You might remember the animation in week 1: we call this *inverse infrastructures*.

Inverse infrastructures is a term coined in 2003 by professor Wim Vree to denote *user-driven* and *self-organizing infrastructures*. Let's look at an example. You are cycling along a road that turns left. But instead of following the bicycle path, you take a short cut across the grass. You are not the first one to do so. There's a desire path. Without any prompting, you and those before you have spontaneously created this path. It results from self-organization.

r/DesirePath • 2 yr. ago
thsvnlwn

Technical University Delft (NL) paved all desire paths.

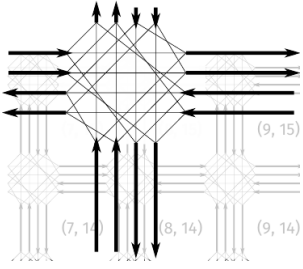
2.3K | 40 | Share | Report

For this to work, there must be choice

To have people self-optimize pavements you have to give them choices

Not much choice here

Same holds for the FPGA router optimizing the interconnect architecture

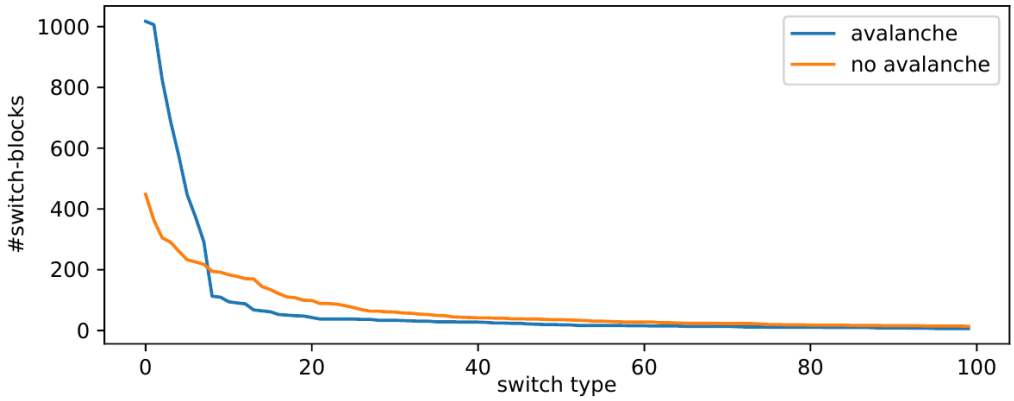


Nor here

Yet, if everyone can do what they want for free, it will be a mess

Hence we modify the cost function of the router to make all switches initially expensive and have their cost drop in proportion to the number of tiles in which they are being used

This is a great aid in switch-pattern minimization



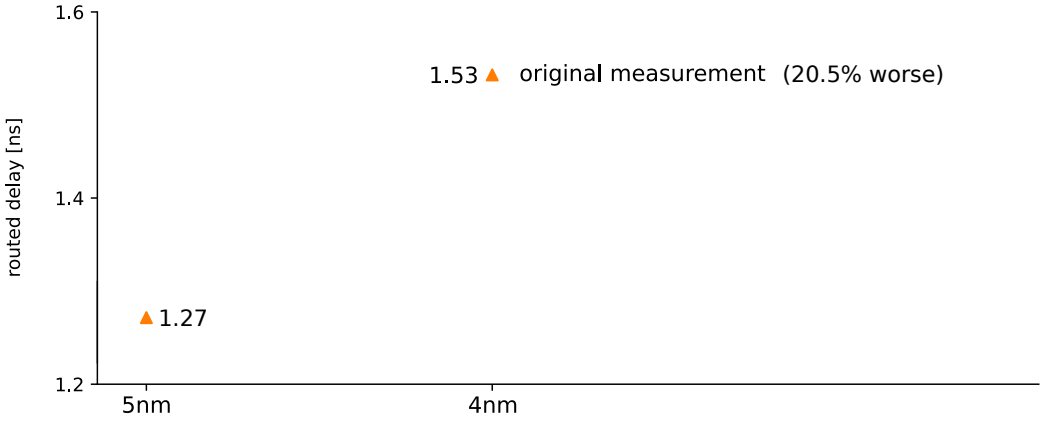
Results

	avalanche			manual [Nik21]			annealed		
#iterations	36						10 000 moves		
#sw. types	78			180			210		
avgerage	fi	fo	t[ps]	fi	fo	t[ps]	fi	fo	t[ps]
H1	5	3	13.9	10	10	16.0	13	13	19.6
H2	5	4	16.8	11	11	21.3	14	11	24.1
H4	4	7	27.4	11	11	30.8	16	12	32.1
H6	5	5	35.7	11	11	43.1	9	13	47.3
V1	7	6	21.8	12	12	24.6	14	15	29.2
V4	2	5	70.1	13	13	74.3	13	15	86.8
W(tile)	6792 nm			7464 nm			7488 nm		
CPD	1.38 ns			1.46 ns			1.55 ns		

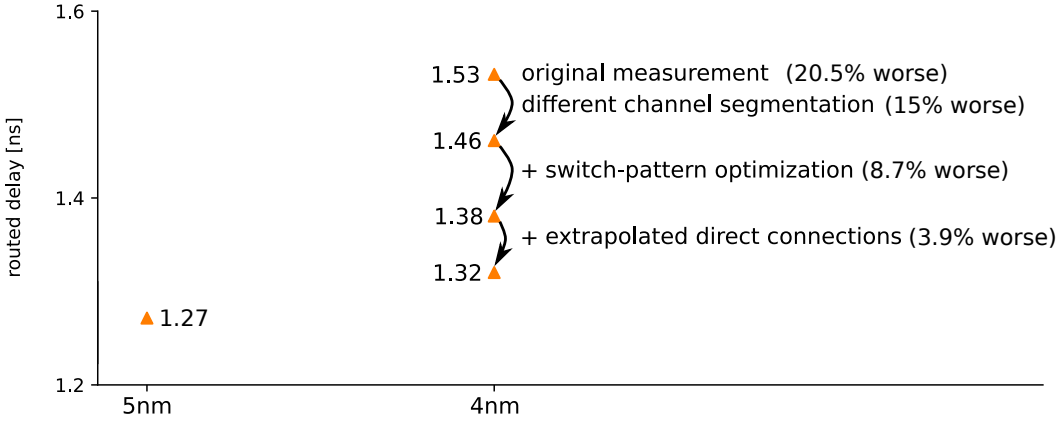
5.48% smaller average routed delay than
our best hand-optimized solution (from a previous paper)

10.97% smaller average routed delay than
simulated-annealing-based optimization

All our architectural work combined



All our architectural work combined



Conclusions

We demonstrated that difficult tasks in programmable interconnect design can be automated and that this can bring great benefits to FPGA performance in new technologies

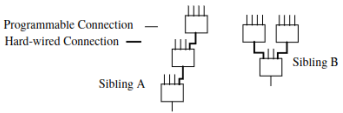
Yet, while necessary, this is not sufficient to bring FPGAs back to the technological forefront

Next step: domain-specific reconfigurable architectures

Using Architectural "Families" to Increase FPGA Speed and Density

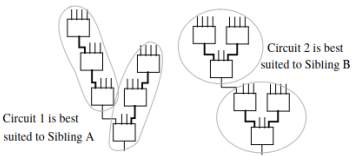
Vaughn Betz and Jonathan Rose
University of Toronto
Toronto, ON, Canada
M5S 1A4
vaughn@eecg.toronto.edu

FPGA'95



(a) A Two Sibling Family

Not a new idea



(b) Mapping of Two Example Circuits into This Family

Again, it's like Lego

General-purpose Lego

Domain-specific, but still reconfigurable Lego

But now the time is ripe for it

- General-purpose FPGAs are difficult to scale
- There are application domains that could call for significant production volumes
- We have design automation algorithms to actually do this customization

We need to understand the requirements of the different application domains

Easing the pain of using FPGAs

Raising the level of abstraction of circuit description is important

XILINX
ALL PROGRAMMABLE.

A Software Developer's Journey into a Deeply Heterogeneous World

Tomas Evensen, CTO Embedded Software, Xilinx

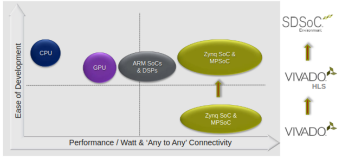
FPGA – Reaching New Developers

- **Limited pool of FPGA developers**
  - Need to reach software developers
  - Software developers are different!
- **Key to reach software developers**
  1. Create libraries so they can utilize accelerators written by others
  2. Create tools so they can utilize FPGA without RTL

Page 7 © Copyright 2016 Xilinx XILINX ALL PROGRAMMABLE.

SDSoC: FPGA Development through Software

FPGA Productivity with Technology Advancement



Page 18 © Copyright 2016 Xilinx XILINX ALL PROGRAMMABLE.

Page 19 © Copyright 2016 Xilinx XILINX ALL PROGRAMMABLE.

But it neglects another major productivity bottleneck

Premise for successfully speeding up modern CAD algorithms

Improving highly general algorithms is still likely possible

But, knowing low-level architectural details and having the ability to change them makes the task much easier

IIBLAST: Speeding Up Commercial FPGA Routing by Decoupling and Mitigating the Intra-CLB Bottleneck

Shashwat Shrivastava*, Stefan Nikolić*, Chirag Ravishankar†, Dinesh Gaitonde†, and Mirjana Stojilović*
*EPFL, †AMD
{shashwat.shrivastava, stefan.nikolic, mirjana.stojilovic}@epfl.ch, {chirag.ravishankar, dinesh.gaitonde}@amd.com

ICCAD'23

Key observation: we lack a good A* heuristic for the sparse intra-CLB routing architecture

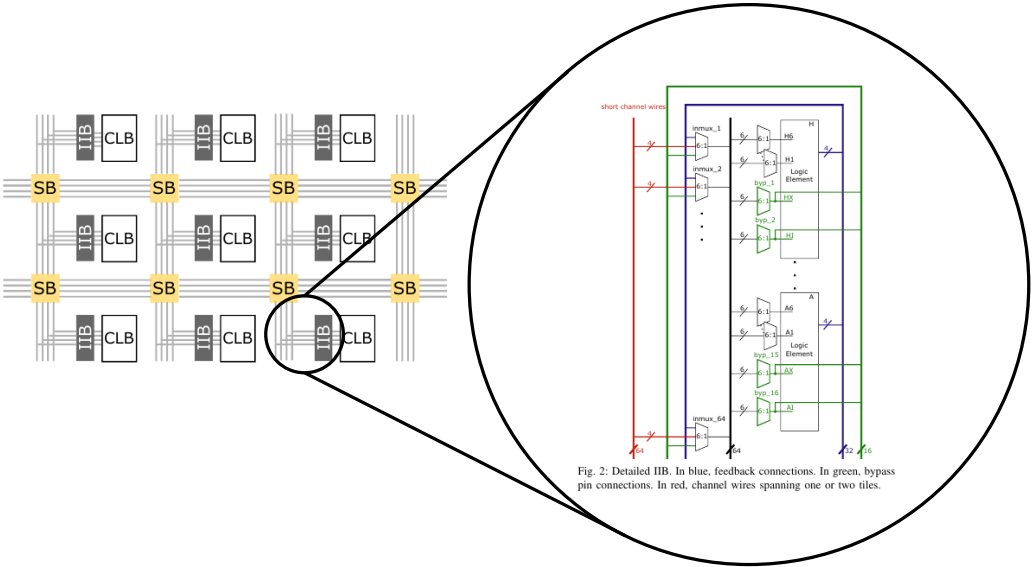


Fig. 2: Detailed IIB. In blue, feedback connections. In green, bypass pin connections. In red, channel wires spanning one or two tiles.

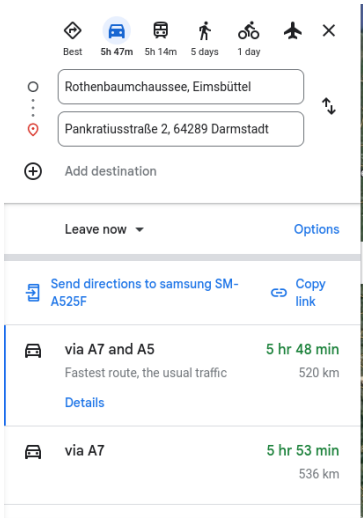
This makes finding paths through the it difficult

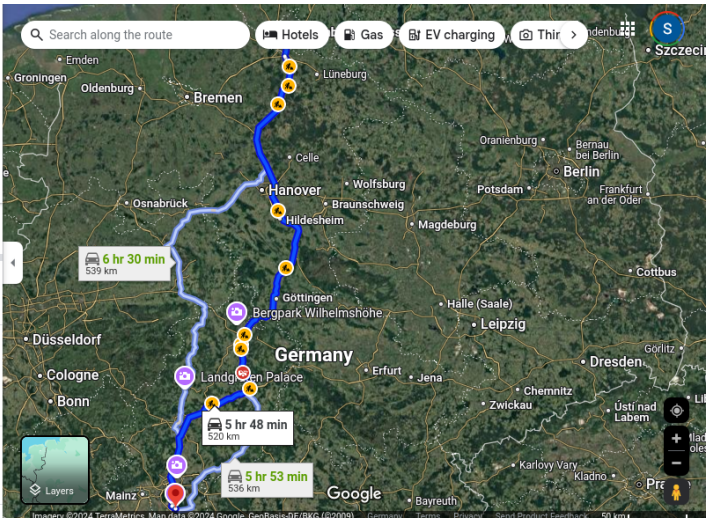
Every time a signal hits congestion on a channel wire, it has to be rerouted
$\Longrightarrow$ the costly intra-CLB path finding has to be repeated again and again

Similar to road-network navigation

Let us assume that we want to go from Rothenbaumchaussee in Hamburg to Pankratiusstraße 2 in Darmstadt

We can use the shortest path (FPGA router would do the same)





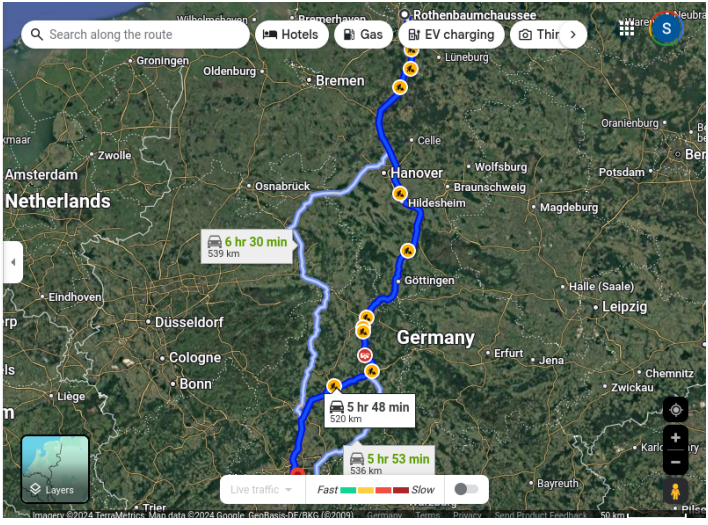
This would also tell us the detailed instructions within Darmstadt

from Rothenbaumchaussee, Eimsbüttel
to Pankratiusstraße 2, 64289 Darmstadt

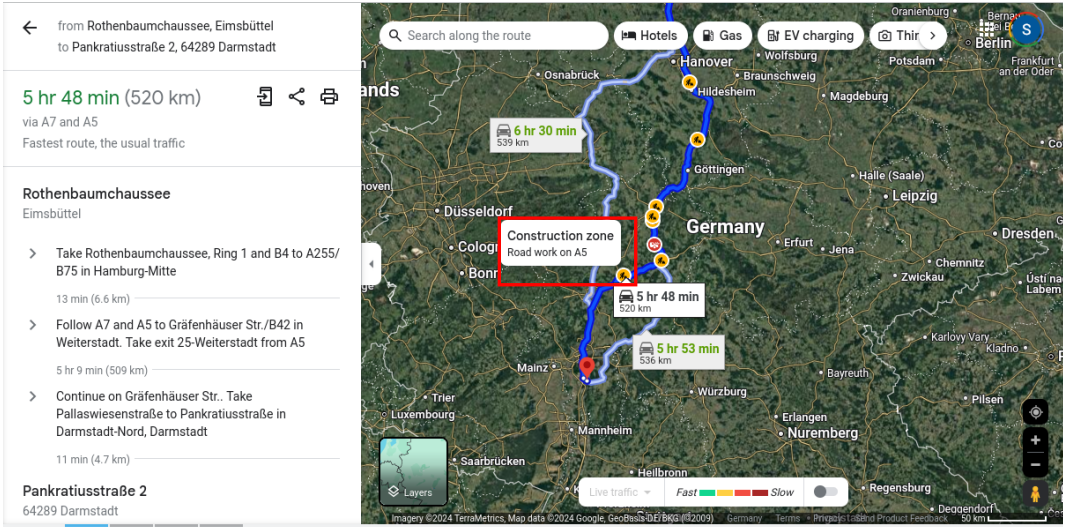
Pallaswiesenstraße to Pankratiusstraße in Darmstadt-Nord, Darmstadt

11 min (4.7 km)

- Merge onto Gräfenhäuser Str./B42
  2.4 km
- Use the right 2 lanes to turn right onto Gräfenhäuser Str./B3
  750 m
- Turn left onto Pallaswiesenstraße/B3
  550 m
- Continue straight onto Pallaswiesenstraße
  550 m
- Turn right onto Schloßgartenpl.
  79 m
- Continue onto Schloßgartenstraße
  400 m
- Turn left onto Pankratiusstraße
  Destination will be on the left



However, if we hit construction work before we reach Darmstadt



We may have to detour and enter Darmstadt from another highway exit

from Rothenbaumchaussee, Eimsbüttel
to Pankratiusstraße 2, 64289 Darmstadt

Continue on Alexanderstraße. Take Magdalenenstraße to Pankratiusstraße in Darmstadt-Nord

2 min (750 m)

Slight right onto Alexanderstraße/Karolinenpl.
Continue to follow Alexanderstraße

350 m

Turn left onto Magdalenenstraße

300 m

Turn left onto Kantpl.

39 m

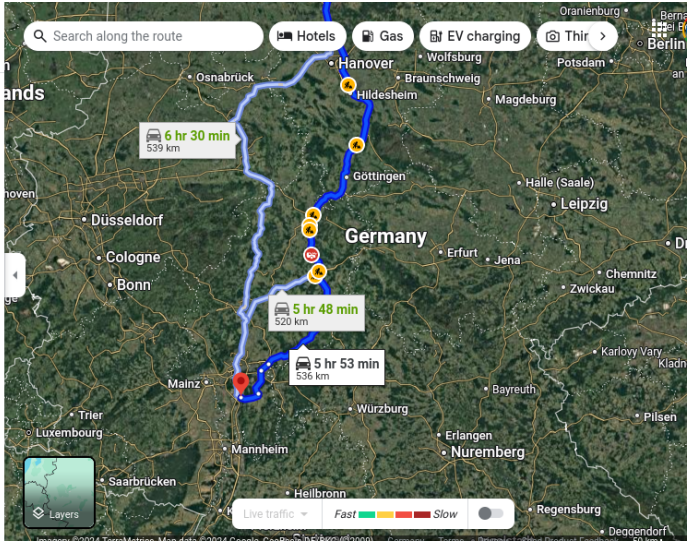
Turn right toward Pankratiusstraße

21 m

Turn right onto Pankratiusstraße
Destination will be on the left

57 m

Pankratiusstraße 2
64289 Darmstadt



At that point, our old intra-city instructions become useless

from Rothenbaumchaussee, Eimsbüttel
to Pankratiusstraße 2, 64289 Darmstadt

Continue on Alexanderstraße. Take Magdalenenstraße to Pankratiusstraße in Darmstadt-Nord

2 min (750 m)

Slight right onto Alexanderstraße/Karolinenpl.
Continue to follow Alexanderstraße

350 m

Turn left onto Magdalenenstraße

300 m

Turn left onto Kantpl.

39 m

Turn right toward Pankratiusstraße

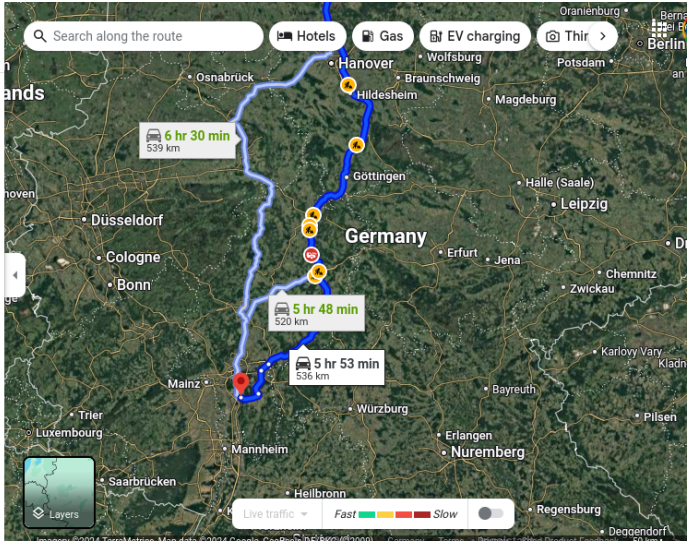
21 m

Turn right onto Pankratiusstraße
Destination will be on the left

57 m

New highway exit = new intra-city instructions

Pankratiusstraße 2
64289 Darmstadt



Lesson learned

If we are repeatedly going to hit congestion and have to reroute, we may save a lot of computation by ignoring street-level navigation before we have reached a highway exit that brings us to the target city

Approach in the paper

1. Neglect the intra-CLB routing until the inter-CLB routing has converged to a legal solution
2. Route the intra-CLB part of all CLBs in parallel (they are independent)
3. If some CLBs cannot route with the chosen “highway exists”, recompose the routing problem and legalize with a single pass of a generic incremental router

Results with a small architectural modification to maximize step 2 success

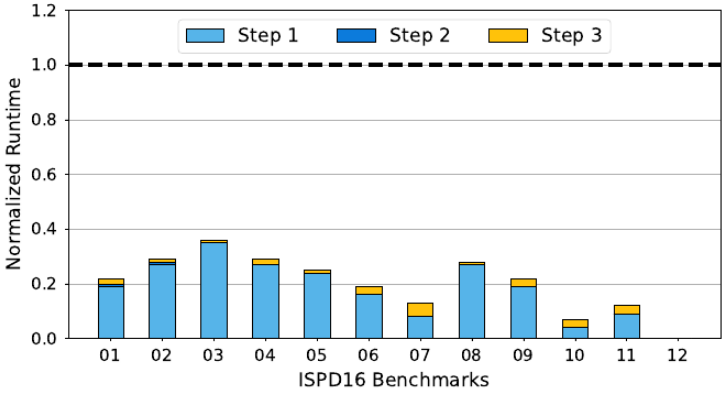


Fig. 10: Runtime comparison: Multi-Stage with IIB-8-2L and LUT-1 vs. Single-Stage with IIB-6-2L. Geomean speedup is 4.94×.

Conclusions

By co-designing architecture and CAD algorithms, we can achieve major speedups

This particular work was aimed at the ASIC emulation application domain (possible to sacrifice some fmax to reduce compilation time)

Other domains may offer a possibility for providing better tools as well

Thank you for attention

stefan.nikolic@dmi.uns.ac.rs