Algorithms for Automated FPGA Interconnect Design

Stefan Nikolić

CUNY Graduate Center, 17.02.2023

École Polytechnique Fédérale de Lausanne

What is an FPGA?

What is an FPGA?

An FPGA is a **RECONFIGURABLE** integrated circuit (chip)

What does “reconfigurable” mean?

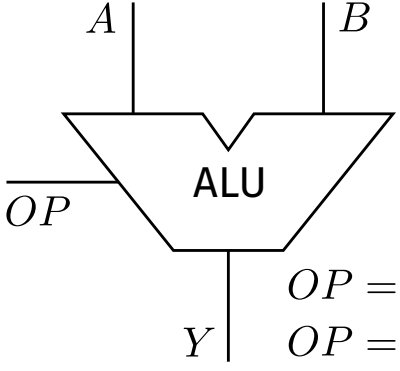
A **RECONFIGURABLE** circuit can change functionality after fabrication

What does “reconfigurable” mean?

A **RECONFIGURABLE** circuit can change functionality after fabrication

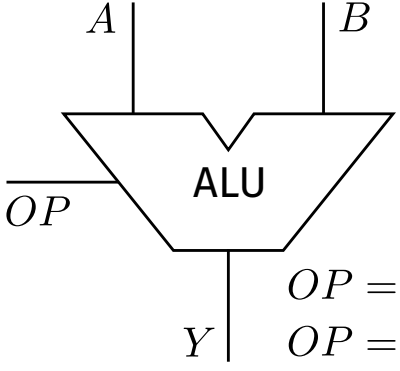
A **NON-CONFIGURABLE** circuit has one functionality fixed during fabrication

What is reconfigurability: A non-configurable circuit (a toy ALU)



$$OP = 0 \implies Y = A \overline{\wedge} B$$
$$OP = 1 \implies Y = A \vee B$$

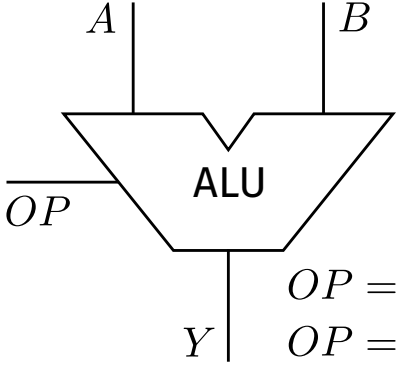
What is reconfigurability: A non-configurable circuit (a toy ALU)



$$OP = 0 \implies Y = A \overline{\wedge} B$$
$$OP = 1 \implies Y = A \vee B$$

Can it compute $A \oplus B$?

What is reconfigurability: A non-configurable circuit (a toy ALU)

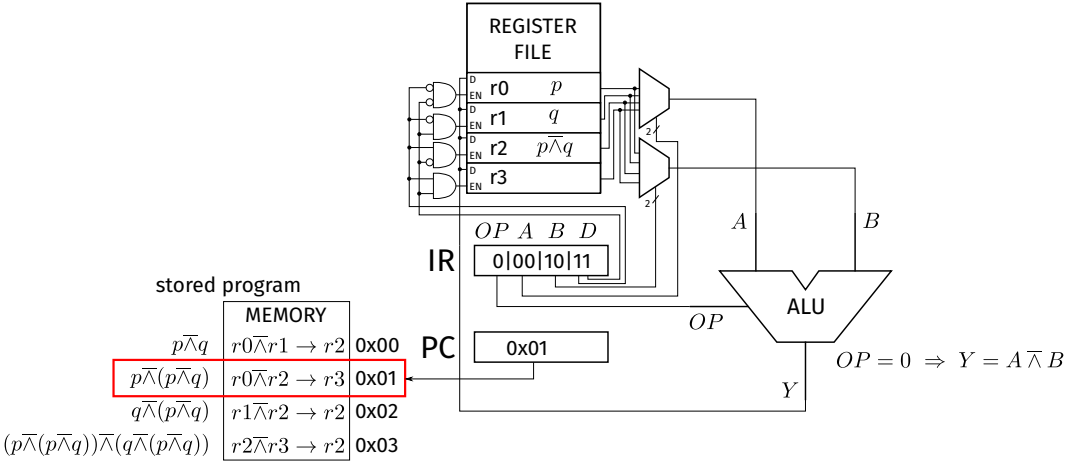


$$OP = 0 \implies Y = A \overline{\wedge} B$$
$$OP = 1 \implies Y = A \vee B$$

Can it compute $A \oplus B$?

Never on its own

What is reconfigurability: Stored-program computer



Can compute compute $A \oplus B$ as a sequence of 4 instructions

What is reconfigurability: Stored-program computer

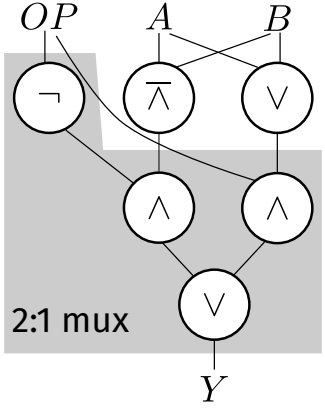
Functionality of a stored-program computer can be changed
by changing the instruction sequence, but this is not what we mean by
RECONFIGURABILITY

What is reconfigurability: Boolean circuits

- A graph $G = (V, E)$ where each node is some Boolean operator
- in-degree $\leqslant K, K = const. \geqslant 2$

What is reconfigurability: Boolean circuits

- A graph $G = (V, E)$ where each node is some Boolean operator
- in-degree $\leqslant K, K = const. \geqslant 2$



Boolean circuit of the toy ALU

Non-configurable circuit

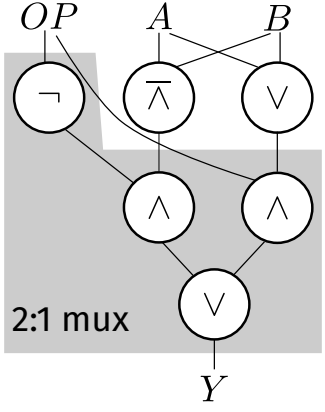
A non-configurable circuit implements only **ONE FIXED** Boolean circuit

(i.e., neither E nor the operator of any $v \in V$ can change)

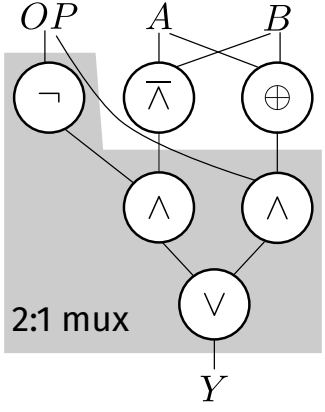
Reconfigurable circuit

In a reconfigurable circuit both E and the operator of any $v \in V$ can change

Reconfigurable toy ALU



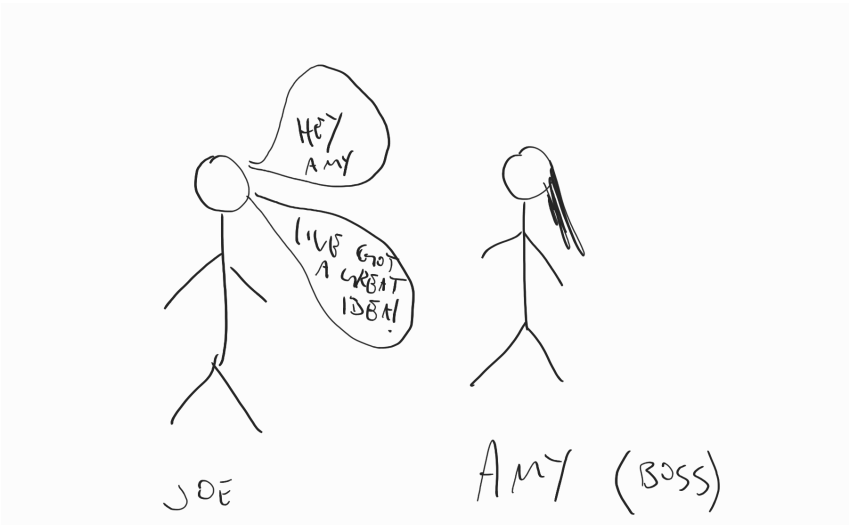
Reconfigurable toy ALU



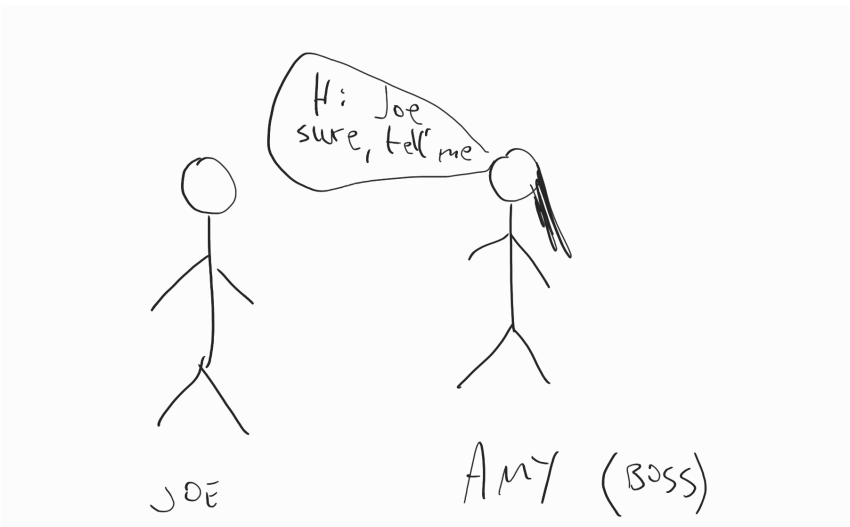
When is reconfigurability useful?

Avoiding chip production cost

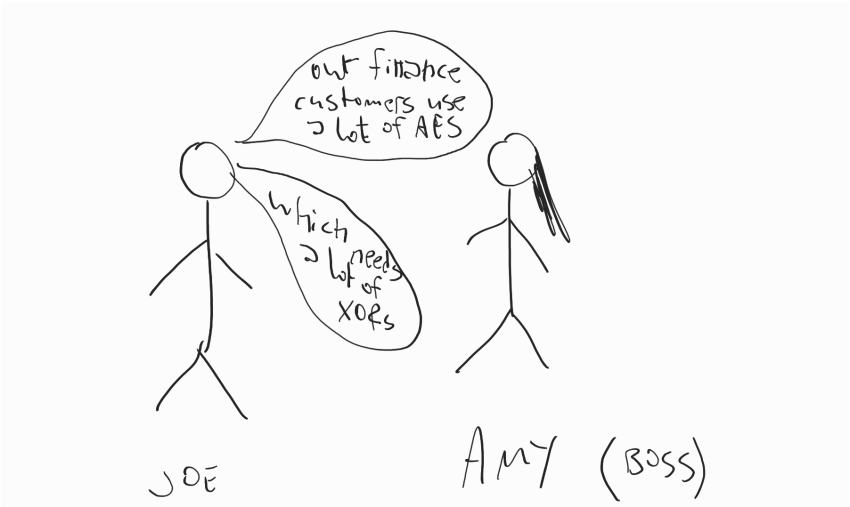
Avoiding chip production cost



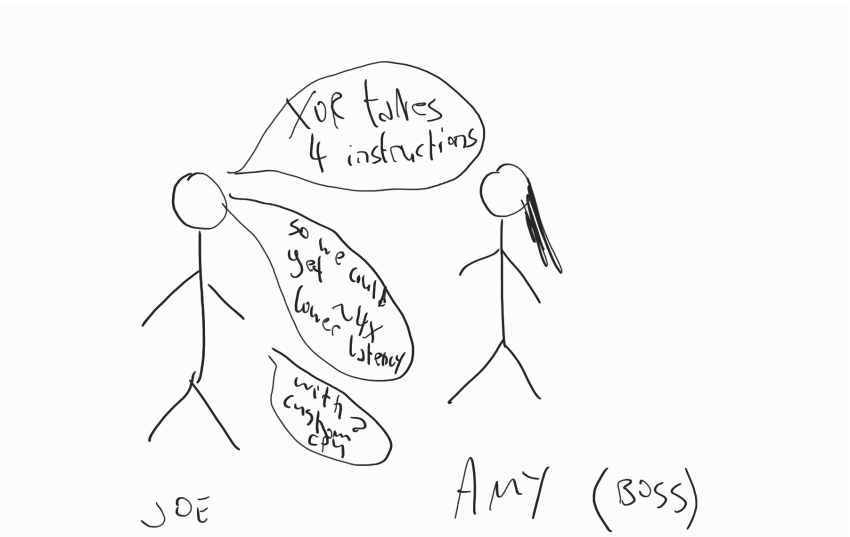
Avoiding chip production cost



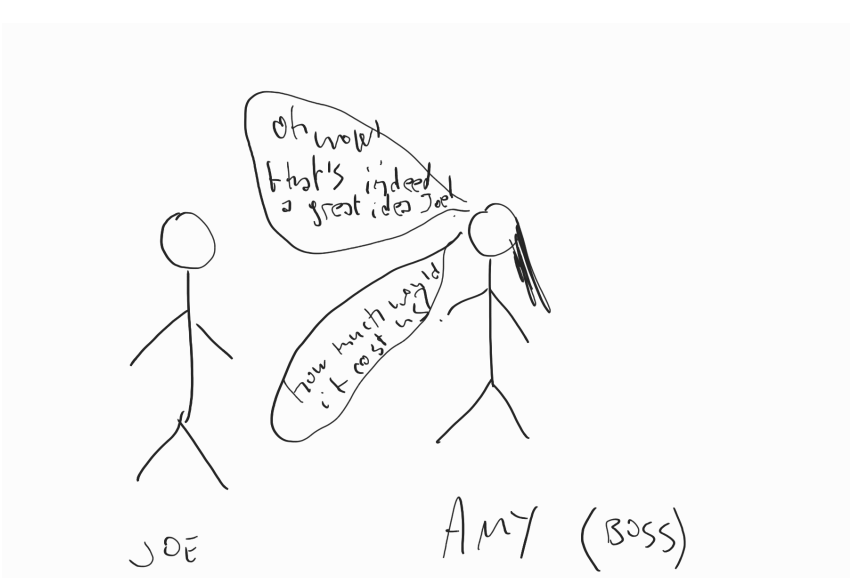
Avoiding chip production cost



Avoiding chip production cost



Avoiding chip production cost



Avoiding chip production cost

Avoiding chip production cost

https://semiengineering.com/facing-off-against-growing-chip-design-complexity/

LOW POWER-HIGH PERFORMANCE

OPINION

Facing Off Against Growing Chip Design Complexity

Modeling advanced process features to improve the precision of simulation results.

DECEMBER 19TH, 2022 - BY: **KELLY DAMALOU**

The semiconductor industry continues to face incredible pressures to deliver higher levels of performance in a smaller area, with lower power demands. From high-performance systems-on-chip for 5G mobile devices and network infrastructure to the radio-frequency transceivers that enable autonomous vehicles and the industrial Internet of Things, today's applications demand a reduced profile, paired with greater power output. This "downsizing" trend has affected every part of the system, but especially the integrated circuits (ICs) that represent the foundational building blocks.

The industry's transition from a standard 7nm chip size to a 5nm design has delivered profile and power gains, but this trend also brings challenges. As chip size decreases, the risk of defects and failure mechanisms grows — which means chipmakers now have to conduct more thorough inspections, which is costing them time and money. And we're not even close to seeing the end of downsizing. With 3nm chips coming in 2022 and 2nm designs predicted for 2024, the engineering risk, and the associated verification workload, is only increasing. While a 7nm chip design is predicted to cost $223.3 million, a 5nm design represents $463.3 million in expenses. That number skyrockets to $650 million for a next-generation 3nm chip design.

Avoiding chip production cost

Avoiding chip production cost

Avoiding chip production cost

Avoiding chip production cost

Further benefits

- Reduced time to market
- Fixing hardware bugs
- Extending product lifetime
- Accelerating different tasks

Further benefits

https://www.nippon.com/en/news/kd884763688642519041

nippon.com Your Doorway to Japan

Latest | In-depth | Japan Data | News | Guide | Video/Live | Sections | Popular

Home > News > Tesla to recall 120,000 units in China due to faulty chip components

Tesla to recall 120,000 units in China due to faulty chip components

Economy Apr 7, 2022

Read in other languages

English | 日本語 | 简体字 | 繁體字 | Français | Español | العربية | Русский

Further benefits

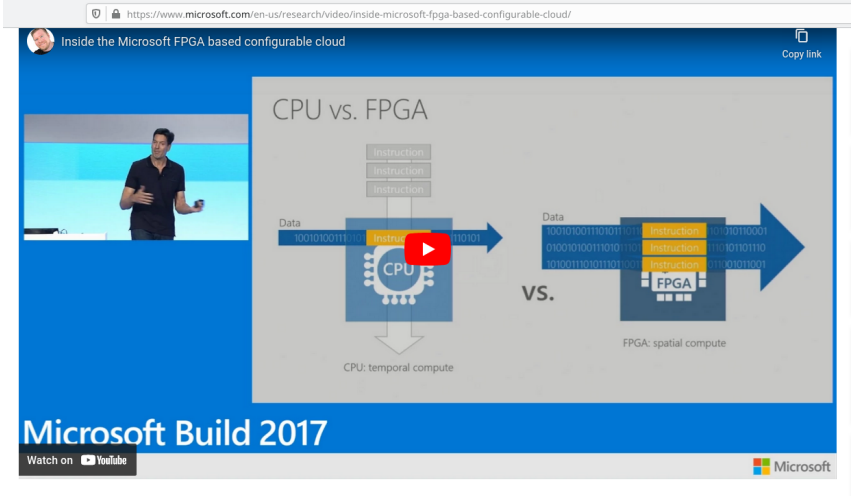
- Reduced time to market
- Fixing hardware bugs
- Extending product lifetime
- Accelerating different tasks

Further benefits

Further benefits

- Reduced time to market
- Fixing hardware bugs
- Extending product lifetime
- Accelerating different tasks

Further benefits



Inside the Microsoft FPGA-based configurable cloud

Source: Channel 9 | Build 2017

Microsoft has been deploying FPGAs in every Azure server over the last several years, creating a cloud that can be reconfigured to optimize a diverse set of applications and functions. This configurable cloud provides more efficient execution than CPUs for many scenarios without the inflexibility of fixed-function ASICs at scale. Today,

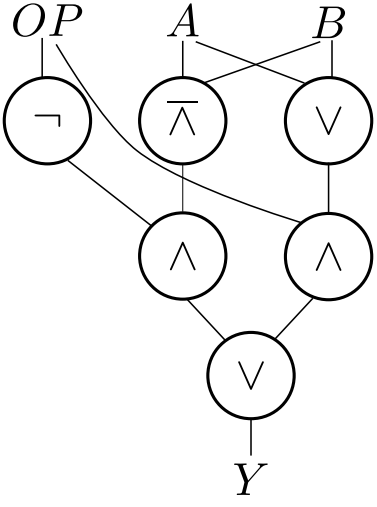
Date: May 8, 2017

Speakers: Mark Russinovich

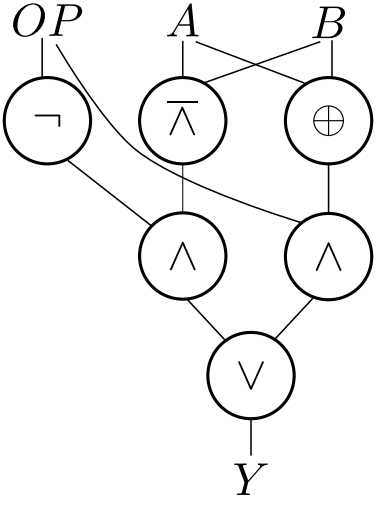
Affiliation: Microsoft

How is reconfigurability achieved?

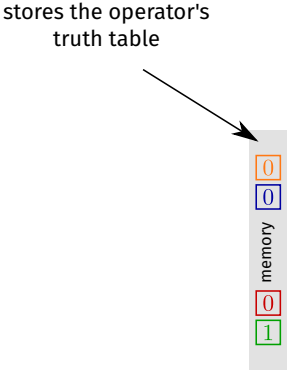
Two tasks: Changing node operators



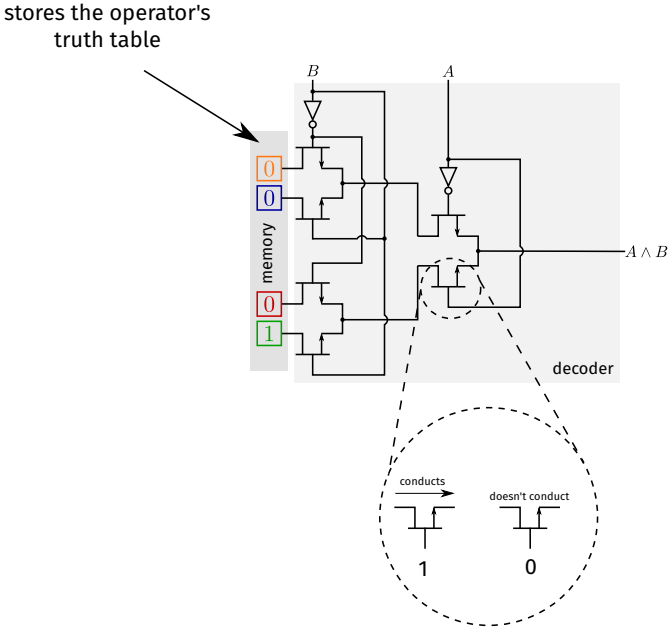
Two tasks: Changing node operators



Look-Up Table (LUT)—A universal operator

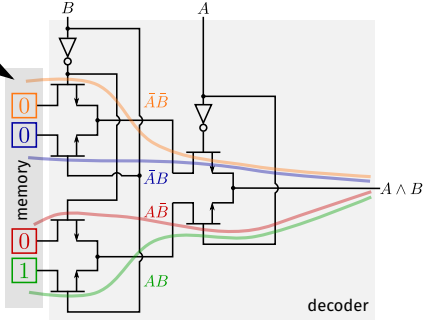


Look-Up Table (LUT)—A universal operator



Look-Up Table (LUT)—A universal operator

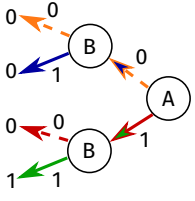
stores the operator's truth table



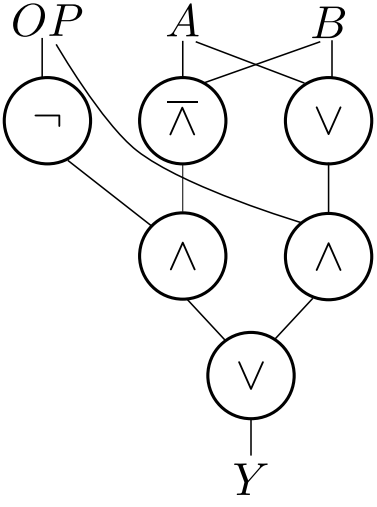
Any Binary Decision Diagram (BDD)

$\subseteq$

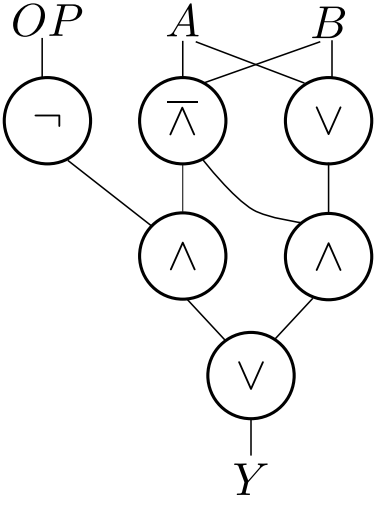
Full binary decision tree



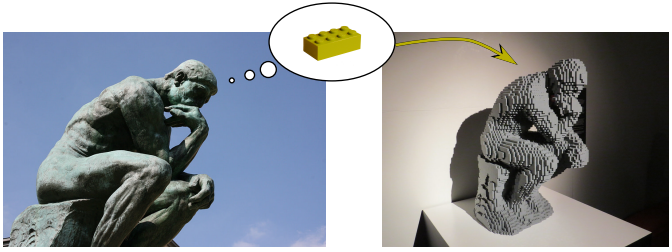
Two tasks: Changing edges



Two tasks: Changing edges



Programmable Interconnect



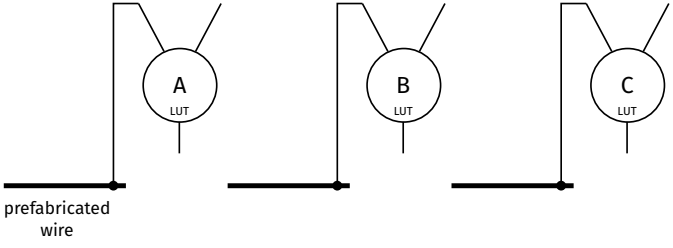
When designing **non-configurable** interconnect:

- produce exactly what is needed

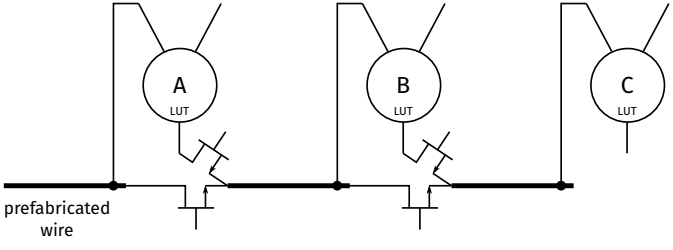
When designing **reconfigurable** interconnect:

- think of something generic enough
- produce it in large quantities
- connect many of these together to build more complex things **after production**

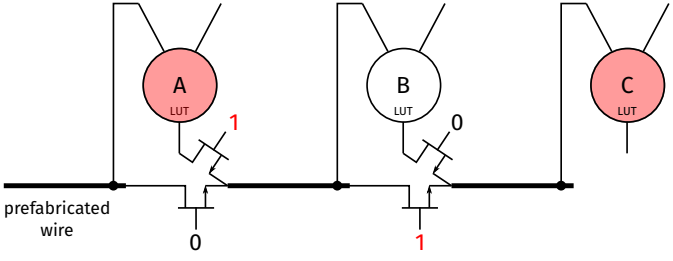
Programmable Interconnect



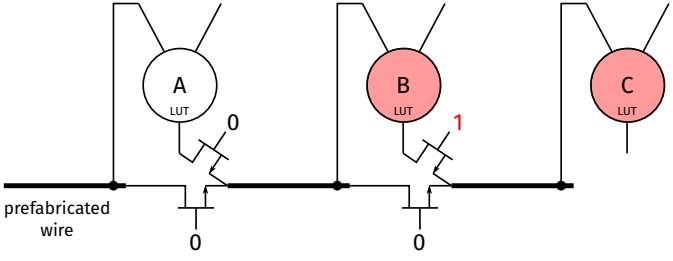
Programmable Interconnect



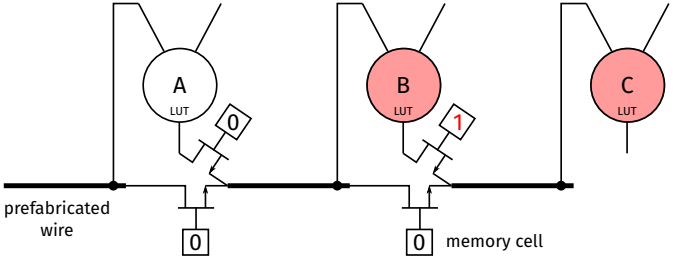
Programmable Interconnect



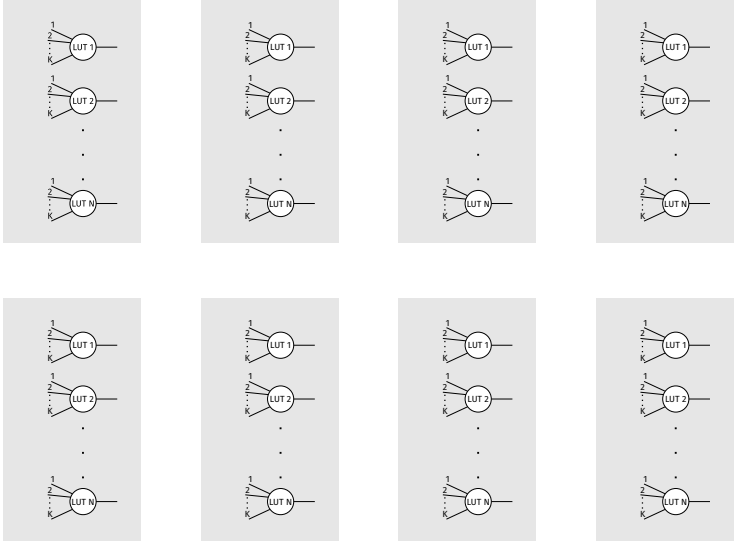
Programmable Interconnect



Programmable Interconnect

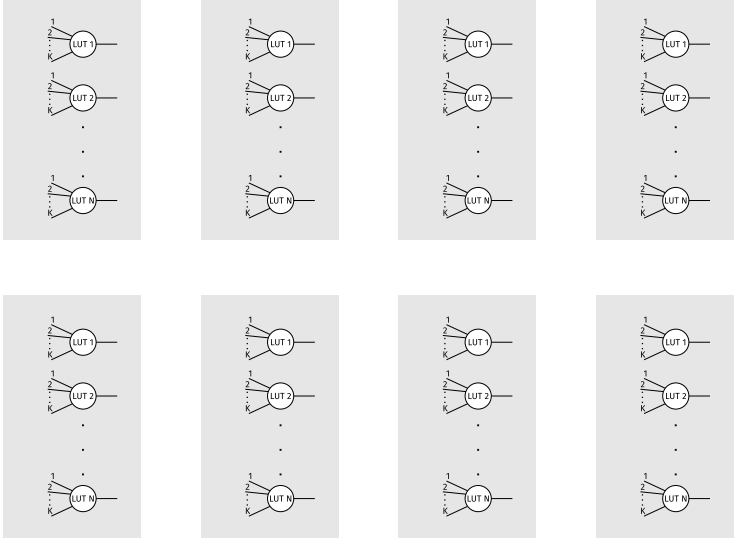


Island-Style Architecture

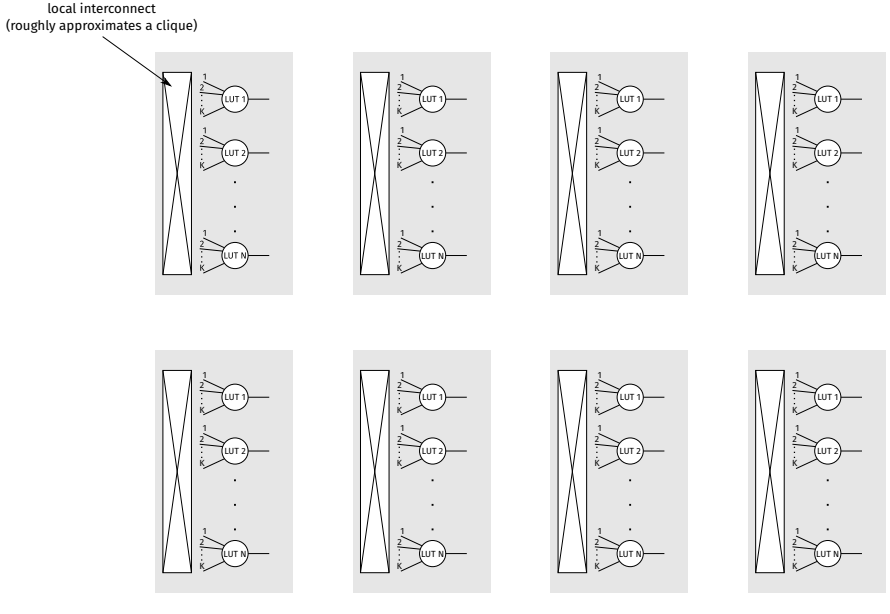


Island-Style Architecture

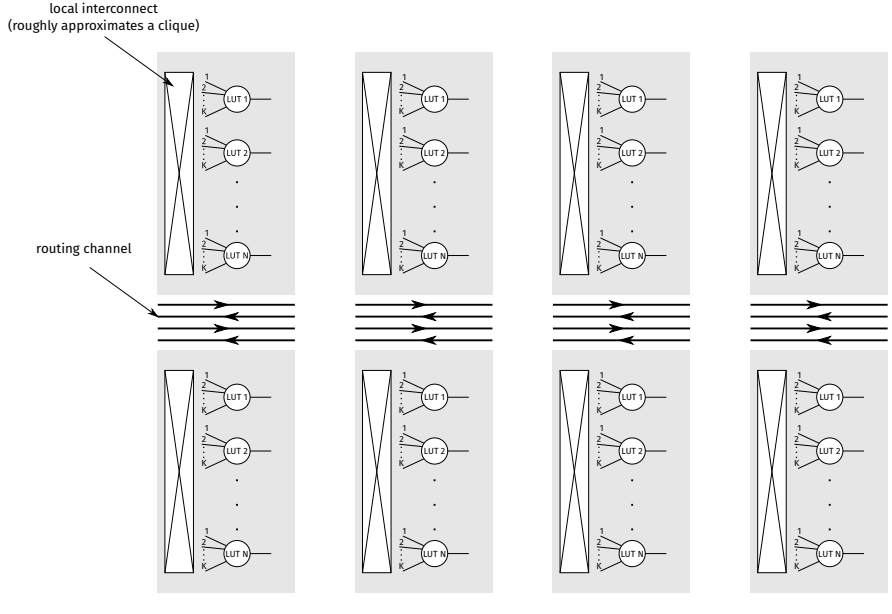
We normally call islands "CLBs"



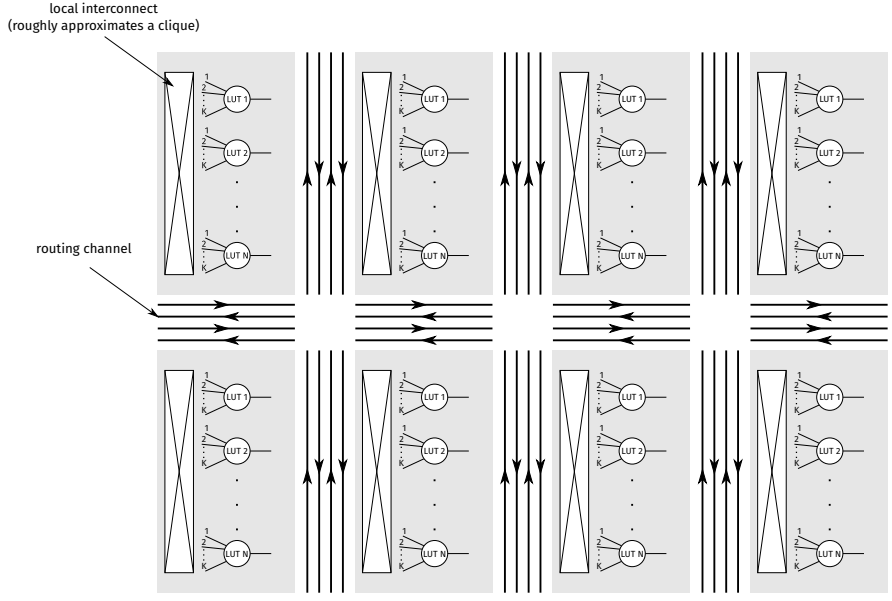
Island-Style Architecture



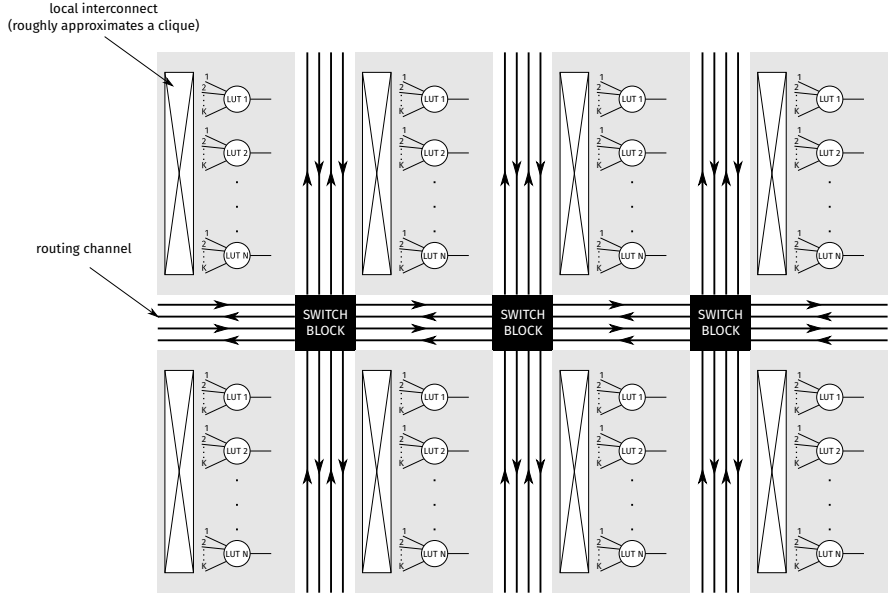
Island-Style Architecture



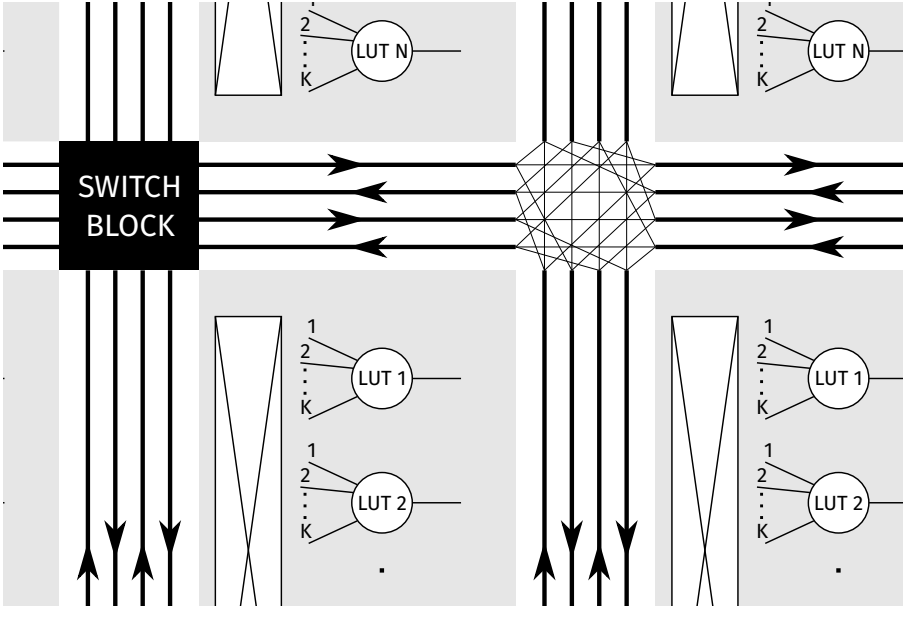
Island-Style Architecture



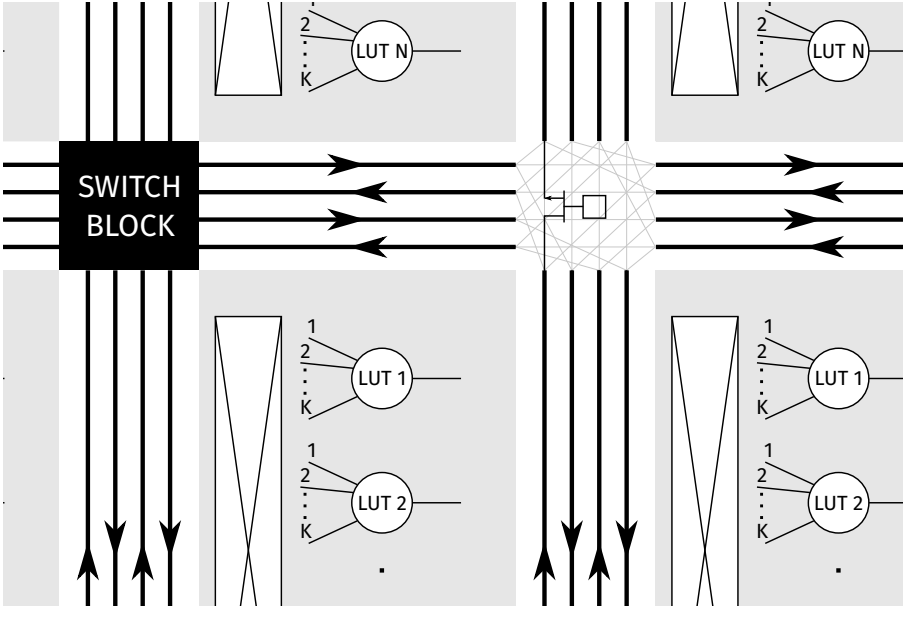
Island-Style Architecture



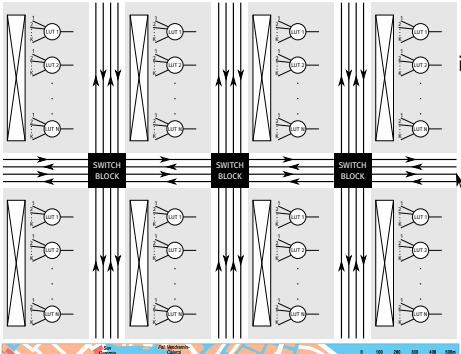
Island-Style Architecture



Island-Style Architecture



Island-Style Architecture



island

channel

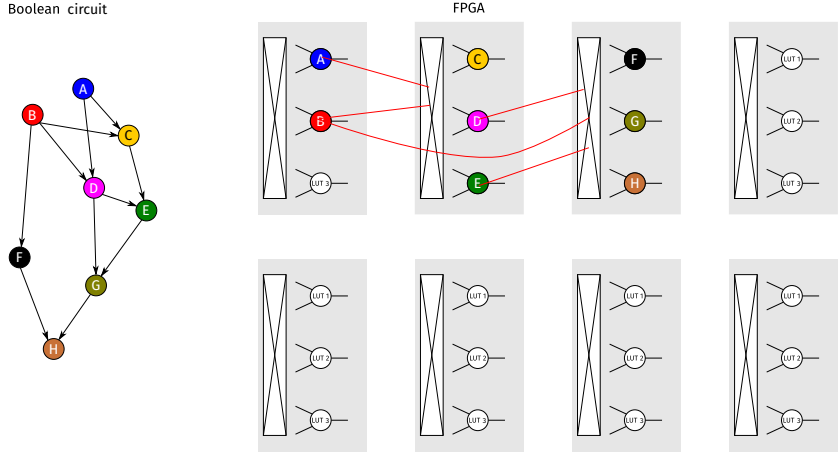
Generating a configuration

Two steps:

1. placement
2. routing

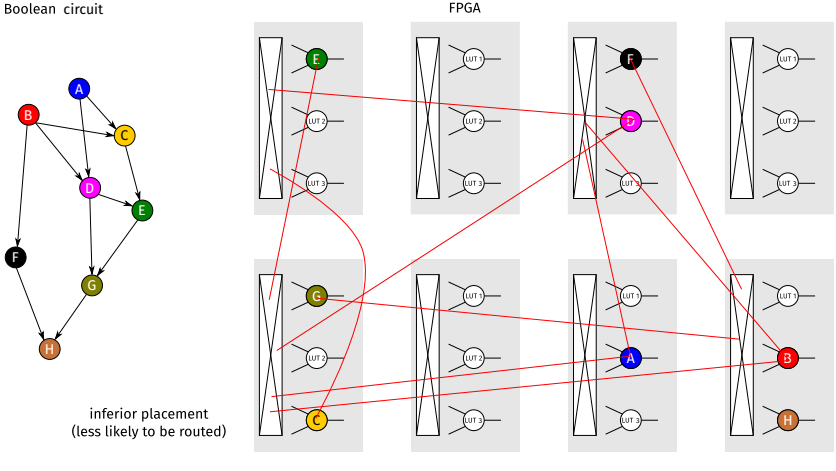
Generating a configuration: Placement

Assign each node a physical location so that external connection length is minimized



Generating a configuration: Placement

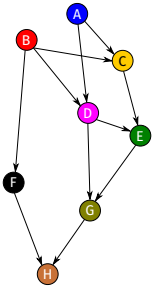
Assign each node a physical location so that external connection length is minimized



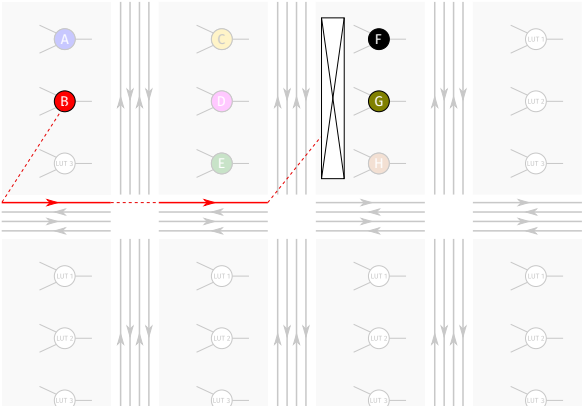
Generating a configuration: Routing

- implement each edge as a path of prefabricated wires and programmable switches
- paths of edges with different tail node must not intersect

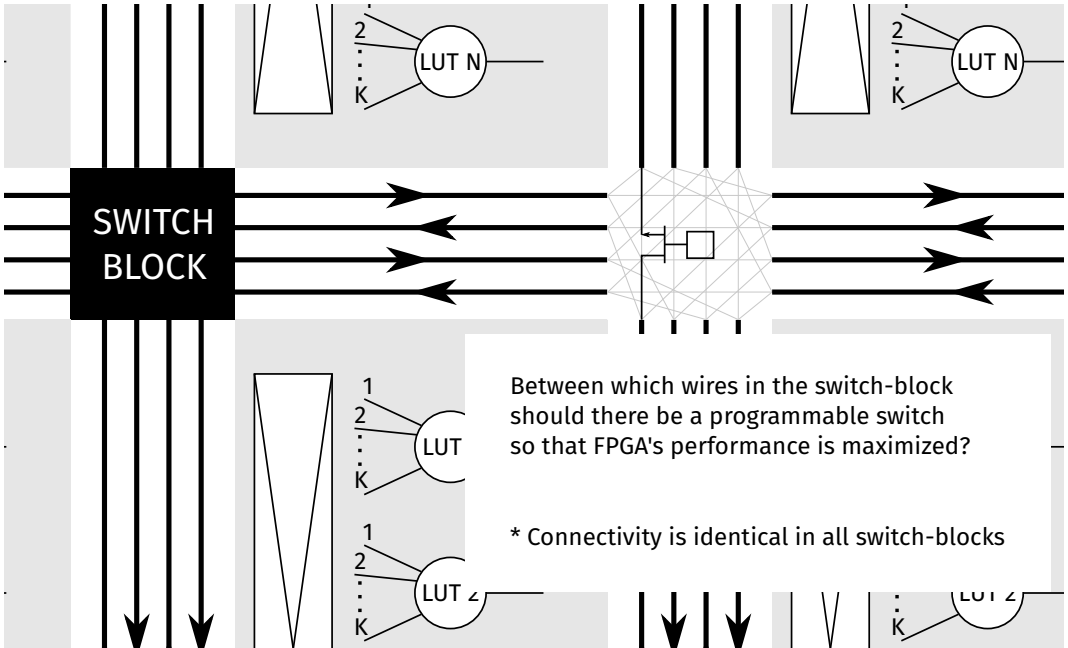
Boolean circuit



FPGA



Fundamental question: To switch or not to switch?



Between which wires in the switch-block should there be a programmable switch so that FPGA's performance is maximized?

* Connectivity is identical in all switch-blocks

Haven't we learned how to design switch-blocks since 1984?

2019 International Conference on Field-Programmable Technology (ICFPT)

A Study on Switch Block Patterns for Tileable FPGA Routing Architectures

Xifan Tang, Edouard Giacomin, Aurélien Alacchi and Pierre-Emmanuel Gaillardon
University of Utah
Email: xifan.tang@utah.edu

***Abstract*—Following the rapid growth of *Field Programmable Gate Arrays* (FPGAs) sizes, the regularity of architectures has become a critical feature, leading to the development of million-of-LUT devices. While the routing architecture plays a dominant role in the area, delay and power of modern FPGAs, most of previously published works focus on improving the routability and performance of FPGAs while very few studied tileable (highly-regular) routing architectures. In this paper, we provide a detailed analysis between tileable and popular non-tileable FPGAs considering modern routing architectures. First, we upgrade VPR to generate tileable routing architecture, which can support different switch block patterns for (1) the routing tracks that start/end in a tile and (2) the routing tracks that pass through a tile. Then, we evaluate the performance of mixed switch blocks patterns in the context of a Stratix IV-like FPGA architecture, by considering the most representative patterns, i.e., Subset, Universal and Wilton. Experimental results show that averaged over the MCNC and VTR benchmarks, when compared to the well-optimized non-tileable architectures, the tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2%. In particular, our results showed that in the context of tileable FPGA, a mix of Universal and Wilton switch block patterns lead to the best trade-**

Experimental results show that compared to VPR, our RRG generator can reduce the number of unique tiles by $8.8\times$ and $5.5\times$ for homogeneous and heterogeneous FPGAs respectively, even considering 128×128 array size.
(2) More than tileable FPGA, our RRG generator also supports different switch block patterns for (a) the routing tracks that start/end in a tile and (b) the routing tracks that pass a tile. We evaluate the performance of mixed switch blocks patterns in the context of a Stratix IV-like FPGA architecture, by considering the most representative patterns, i.e., Subset [20], Universal [19], Wilton [21] and Imran [22]. Experimental results show that averaged over the MCNC and VTR benchmarks, when compared to the well-optimized non-tileable architectures, the tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2%. In particular, our results showed that in the context of tileable FPGA, a mix of Universal and Wilton switch block patterns leads to the best trade-off in area, delay and routability, while Wilton switch block was the best choice in non-tileable FPGAs.

Haven't we learned how to design switch-blocks since 1984?

2019 International Conference on Field-Programmable Technology (ICFPT)

A Study on Switch Block Patterns for Tileable FPGA Routing Architectures

Xifan Tang, Edouard Giacomin, Aurélien Alacchi and Pierre-Emmanuel Gaillardon
University of Utah
Email: xifan.tang@utah.edu

***Abstract*—Following the rapid growth of *Field Programmable Gate Arrays* (FPGAs) sizes, the regularity of architectures has become a critical feature, leading to the development of million-of-LUT devices. While the routing architecture plays a dominant role in the area, delay and power of modern FPGAs, most of previously published works focus on improving the routability and performance of FPGAs while very few studied tileable (highly-regular) routing architectures. In this paper, we provide a detailed analysis between tileable and popular non-tileable FPGAs considering modern routing architectures. First, we upgrade VPR to generate tileable routing architecture, which can support different switch block patterns for (1) the routing tracks that start/end in a tile and (2) the routing tracks that pass through a tile. Then, we evaluate the performance of mixed switch blocks patterns in the context of a Stratix IV-like FPGA architecture, by considering the most representative patterns, i.e., Subset, Universal and Wilton. Experim… that averaged over the MCNC and VTR … compared to the well-optimized non-tileabl… tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2%. In particular, our results showed that in the context of tileable FPGA, a mix of Universal and Wilton switch block patterns lead to the best trade-…

Experimental results show that compared to VPR, our RRG generator can reduce the number of unique tiles by 8.8× and 5.5× for homogeneous and heterogeneous FPGAs respectively, even considering 128 × 128 array size.
(2) More than tileable FPGA, our RRG generator a… different switch block patterns for (a) the routing… start/end in a tile and (b) the routing tracks that pass a tile. We evaluate the performance of mixed switch blocks patterns in the context of a Stratix IV-like FPGA architecture, by considering the most representative patterns, i.e., Subset [20], Universal [19], Wilton [21] and Imran [22]. Experimental results show that averaged over the MCNC and VTR benchmarks, when compared to the well-optimized non-tileable architectures, … In … a mix of Uni… Wilton switch b… o the best trade… a, delay and rout… Wilton switch block was the best choice in non-tileable FPGAs.

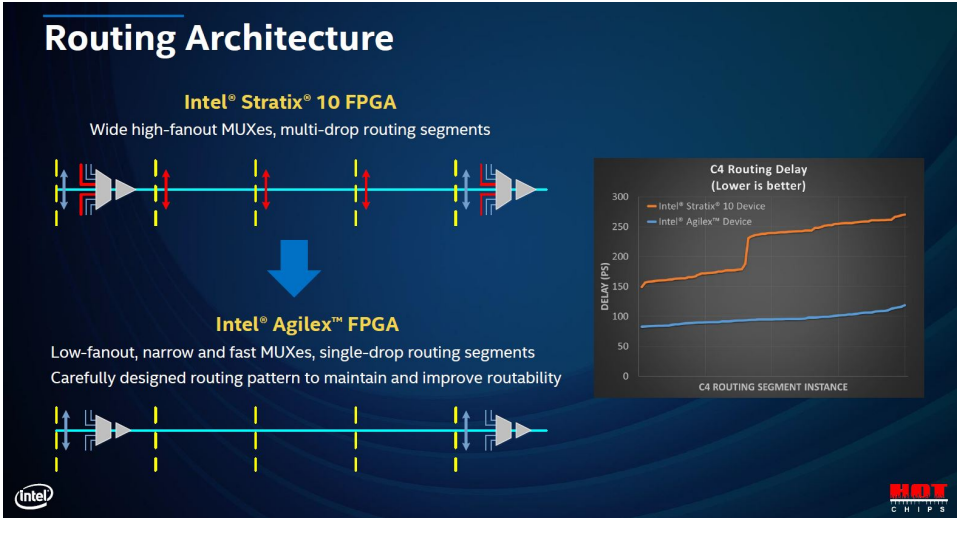
Subset [20] 1997

Universal [19] 1996

Wilton [21] 1997

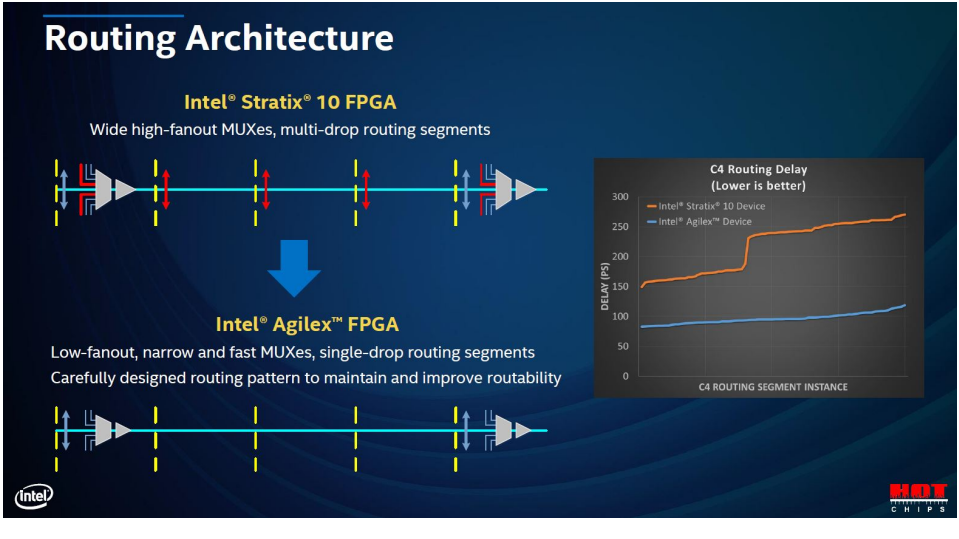
Imran [22] 1999

Meanwhile in industry: technology driving change



[1] Ganusov and Iyer. Agilex Generation of Intel FPGAs. Hot Chips'20

Meanwhile in industry: technology driving change



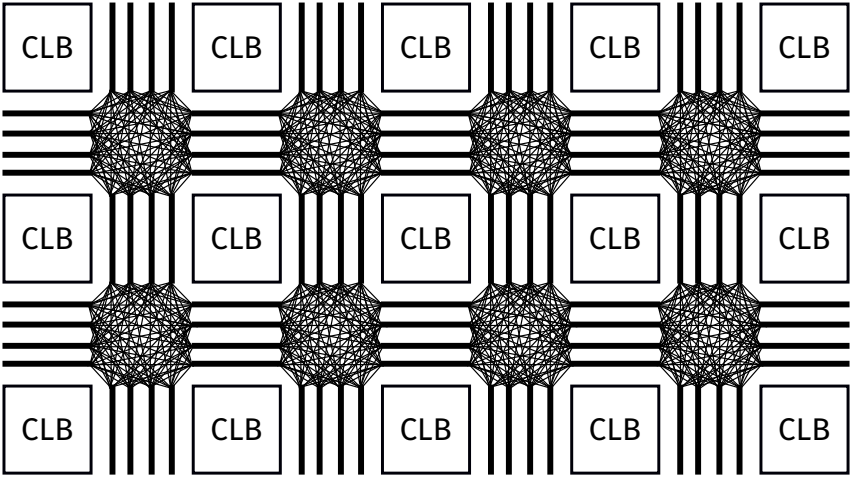
"Carefully designed routing pattern to maintain and improve routability"

Can we automatically design switch-patterns optimized for technology X?

What do we expect from a switch-block?

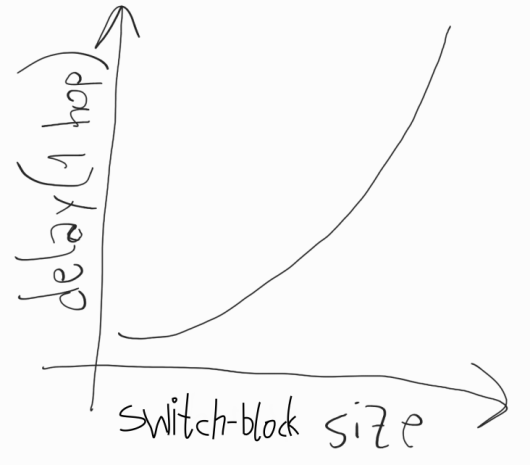
Route many different connections
of many different circuits
with minimal delay

What do we expect from a switch-block?



fully-connected $\Longrightarrow$ hops always minimal

What do we expect from a switch-block?



What do we expect from a switch-block?

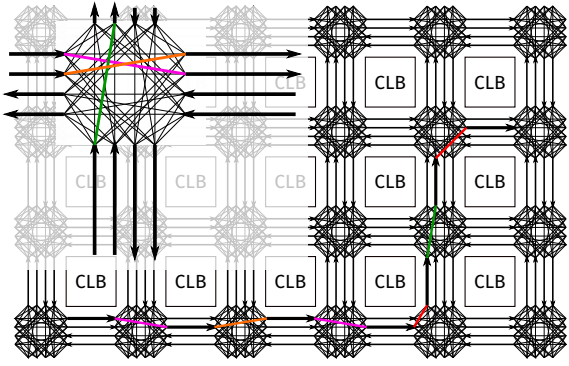
So we need

- something sparse
- that still minimizes hops
- allows paths to avoid intersecting (congestion resolution)

Negotiating switches

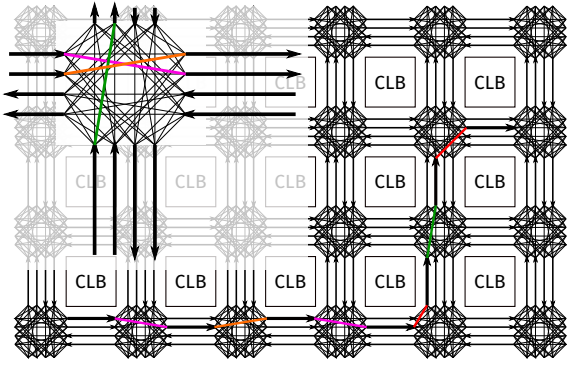
FPGA router vs mosaic artist

Disclaimer: authors have nothing to do with Damanhur; photo had a creative commons license



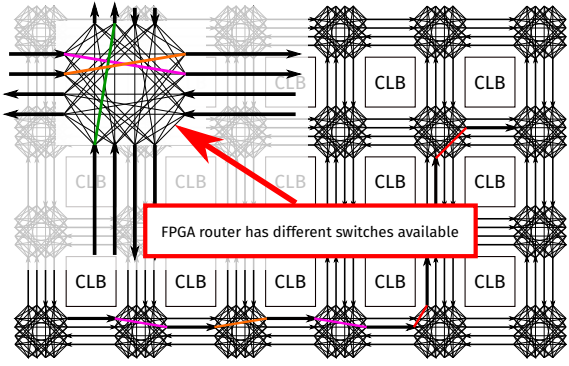
FPGA router vs mosaic artist

Disclaimer: authors have nothing to do with Damanhur; photo had a creative commons license



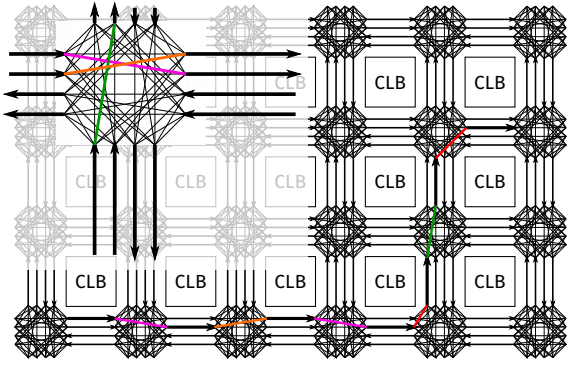
FPGA router vs mosaic artist

Disclaimer: authors have nothing to do with Damanhur; photo had a creative commons license



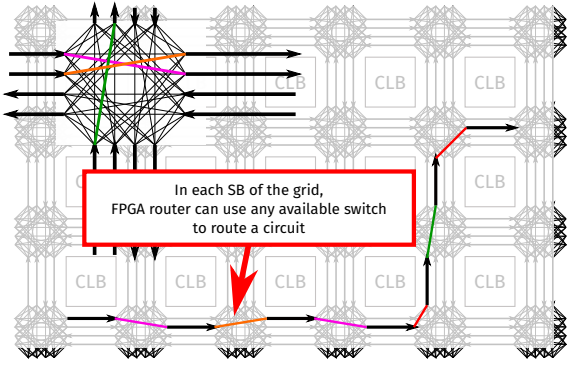
FPGA router vs mosaic artist

Disclaimer: authors have nothing to do with Damanhur; photo had a creative commons license



FPGA router vs mosaic artist

Disclaimer: authors have nothing to do with Damanhur; photo had a creative commons license



FPGA architect vs seller of mosaic stones

FPGA architect's problem

- Too many different switches = expensive, poor performance

FPGA architect vs seller of mosaic stones

FPGA architect's problem

- Too many different switches = expensive, poor performance

Mosaic stone seller's problem

- Too many different stones = hard to search through inventory

FPGA architect vs seller of mosaic stones

FPGA architect's problem

- Too many different switches = expensive, poor performance

FPGA architect's goal

- Minimize the number of different switches
- While making the router happy (so that it creates fast circuits)

Mosaic stone seller's problem

- Too many different stones = hard to search through inventory

FPGA architect vs seller of mosaic stones

FPGA architect's problem

- Too many different switches = expensive, poor performance

FPGA architect's goal

- Minimize the number of different switches
- While making the router happy (so that it creates fast circuits)

Mosaic stone seller's problem

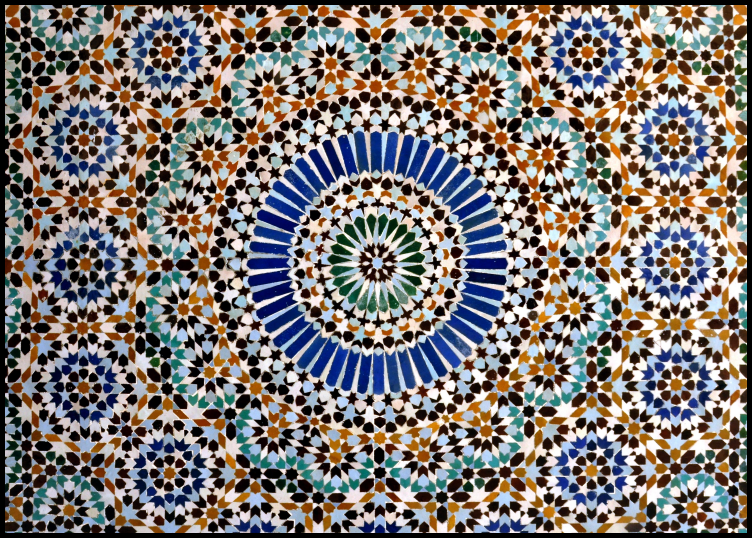
- Too many different stones = hard to search through inventory

Mosaic stone seller's goal

- Minimize the number of different stones
- While making the artist happy (so that they create nice mosaics)

Mosaic stone sellers existed centuries before FPGA architects...

Ahmed is a 16th century mosaic artist



Ahmed's ideal design made of 1000 different kinds of stones

Ahmed sails to Constantinople to buy the stones

He goes to a bazaar to find the store selling 1000 kinds of stones

… And presents his list to Mustafa the shop owner

1. Ultramarine medium vedge (576 pieces)
2. Alabaster needle (229 pieces)
3. Coral hexagon (143 pieces)

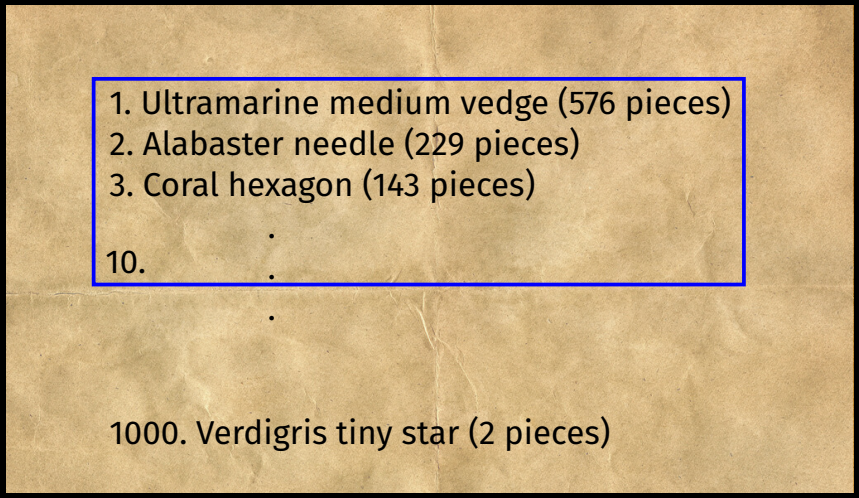
.
.
.

1000. Verdigris tiny star (2 pieces)

When Mustafa saw “1000. Verdigris tiny star (2 pieces)”

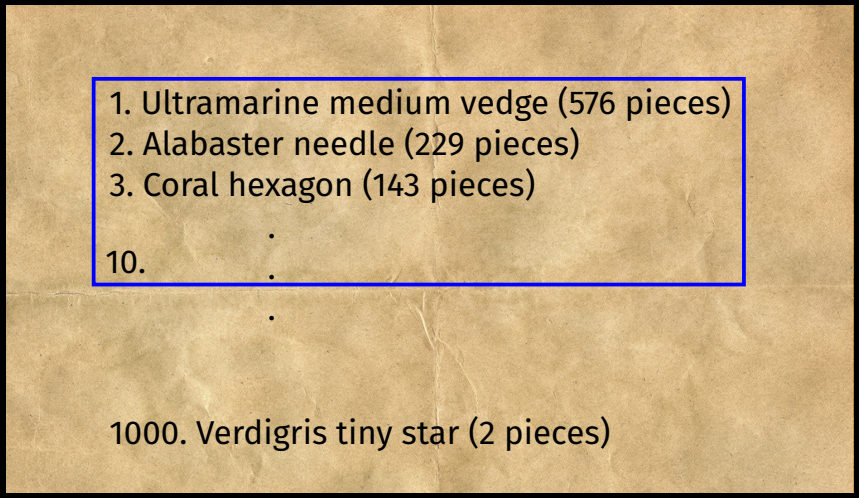
“Oh my, will I really have to search through all these bags?”

Mustafa has an idea



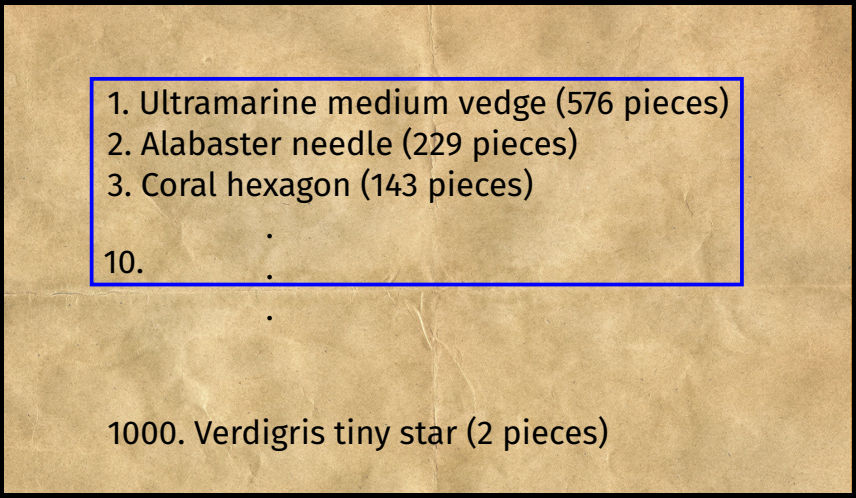
“Here, you seem to need these 10 kinds the most.”

Mustafa has an idea



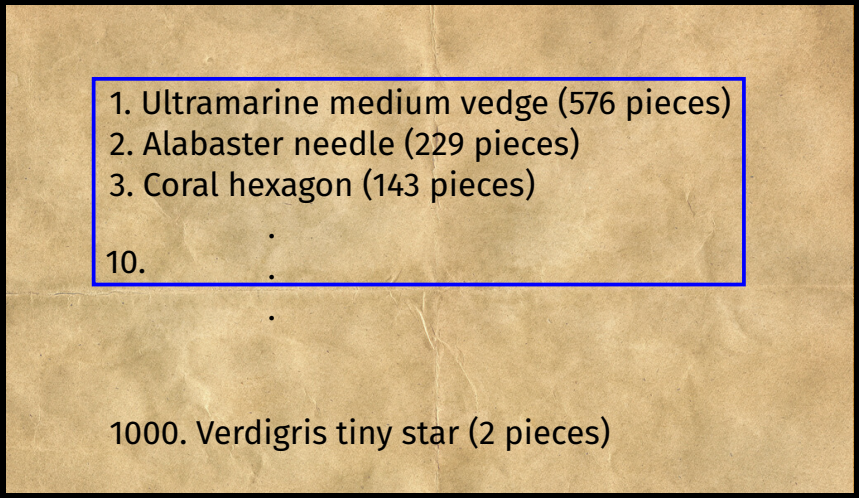
“I give you a big discount on any number of them!”

Mustafa has an idea



“But once I take the bags out, you have to buy. Deal?”

Mustafa has an idea

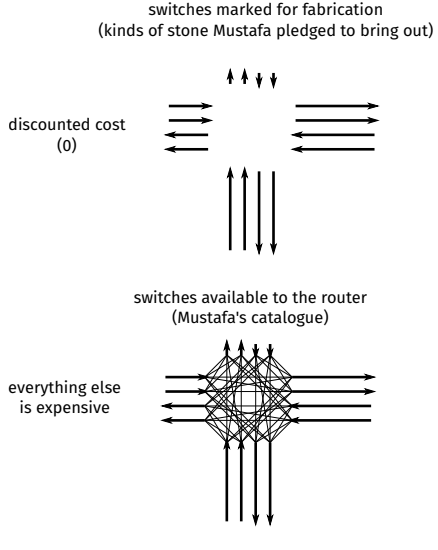


“You can come tomorrow with a new design.”

Ahmed agrees and tries to redesign the mosaic
to maximize the usage of discounted stones

Let’s see how this works for switch-block exploration

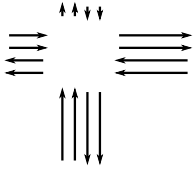
Initial setting



PathFinder routes the first time (Ahmed creates his ideal mosaic)

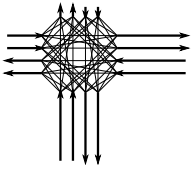
switches marked for fabrication
(kinds of stone Mustafa pledged to bring out)

discounted cost
(0)

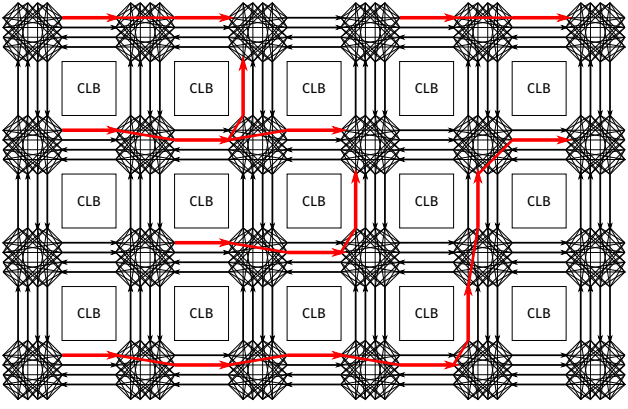


switches available to the router
(Mustafa's catalogue)

everything else
is expensive

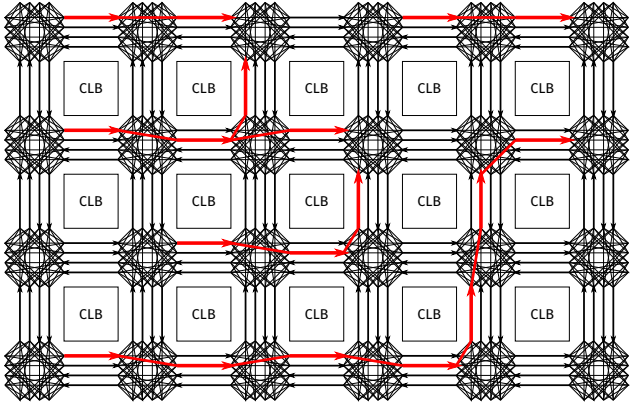


PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)

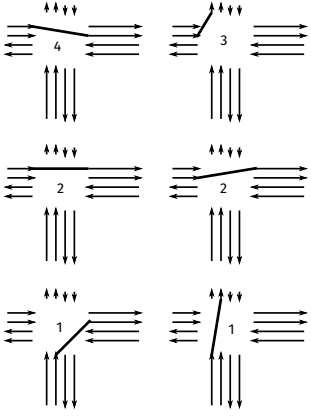


Count in how many SBs each switch is used (Ahmed writes his list)

PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)

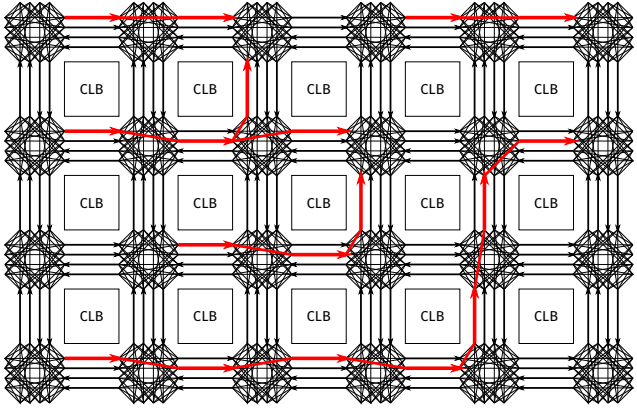


Count in how many switch-blocks each switch was used
(Ahmed counts different stones)

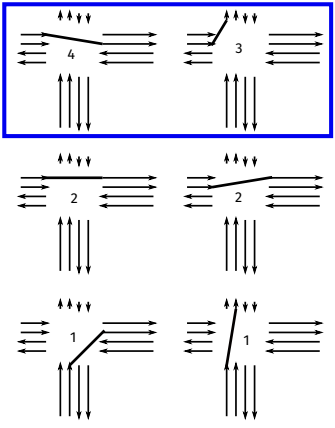


Give discounts and mark for fabrication (Mustafa gives discounts)

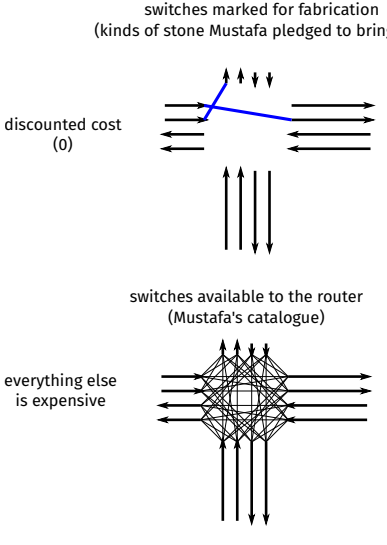
PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)



Give discounts and mark for fabrication
(Mustafa gives discount and takes out the bags)



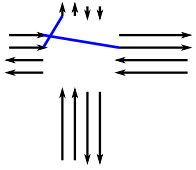
Give discounts and mark for fabrication (Mustafa gives discounts)



PathFinder routes again, maximizing usage of discounted switches (Ahmed redesigns his mosaic)

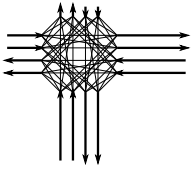
switches marked for fabrication
(kinds of stone Mustafa pledged to bring out)

discounted cost
(0)

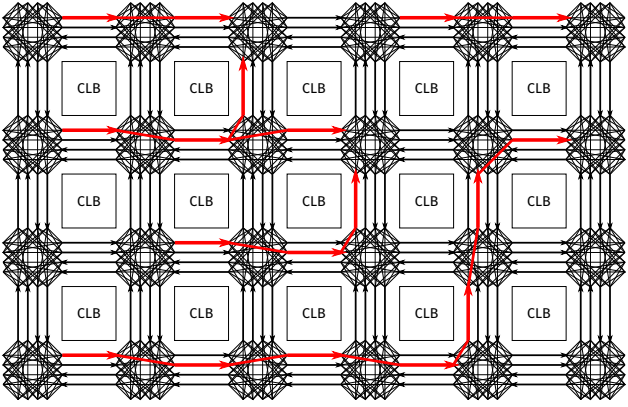


switches available to the router
(Mustafa's catalogue)

everything else
is expensive



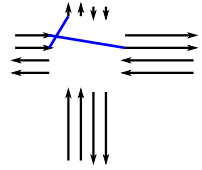
PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)



PathFinder routes again, maximizing usage of discounted switches (Ahmed redesigns his mosaic)

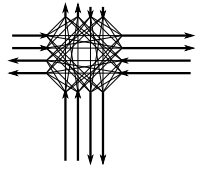
switches marked for fabrication
(kinds of stone Mustafa pledged to bring out)

discounted cost
(0)

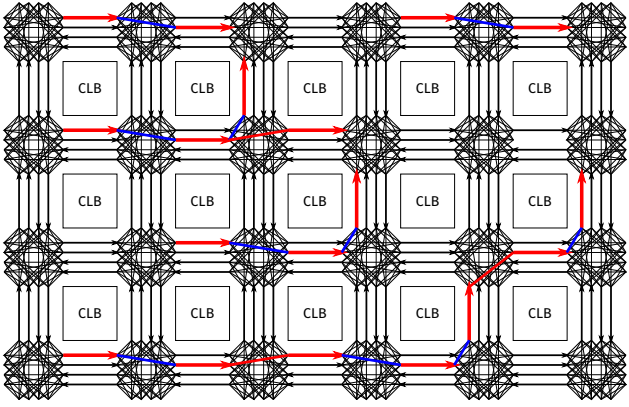


switches available to the router
(Mustafa's catalogue)

everything else
is expensive

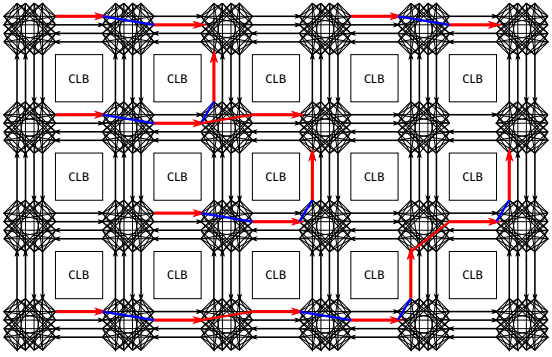


PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)

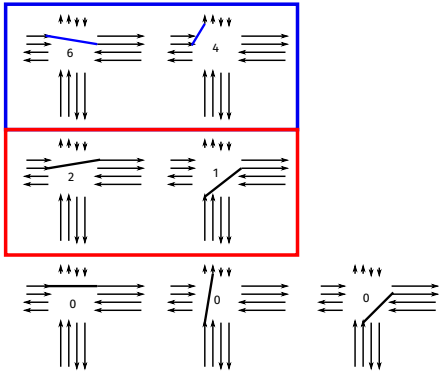


PathFinder still wants non-discounted switches (Ahmed's list is still long)

PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)



Count in how many switch-blocks each switch was used
(Ahmed counts different stones)

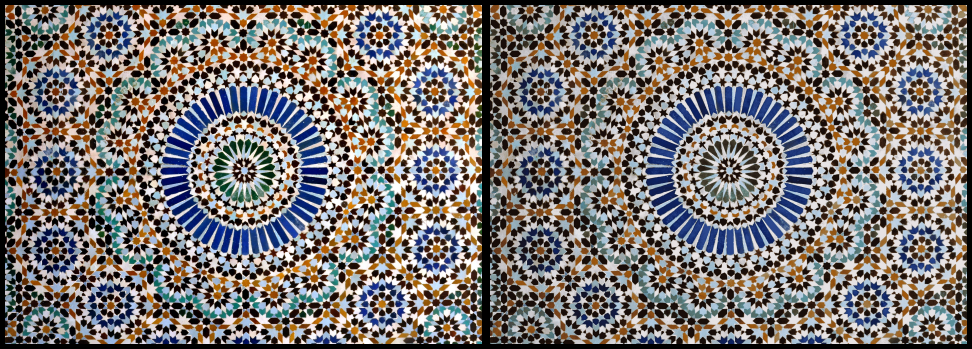


Could be required for connectivity, congestion, or critical paths

What do we do now?

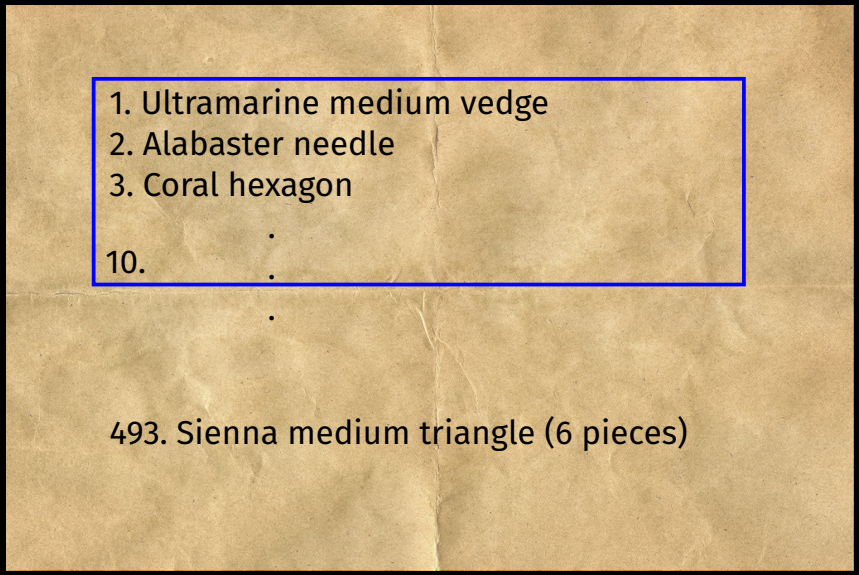
Let’s see what Mustafa did

Ahmed agrees and tries to rework the design thinking of the discount



With only 493 kinds of stones, difference is barely visible

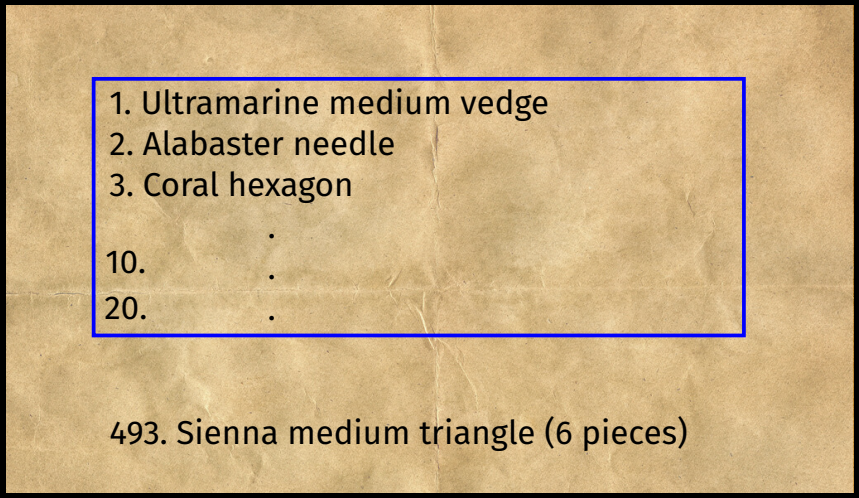
So Ahmed goes to bargain with Mustafa again



When Mustafa saw “493. Sienna medium triangle (6 pieces)”

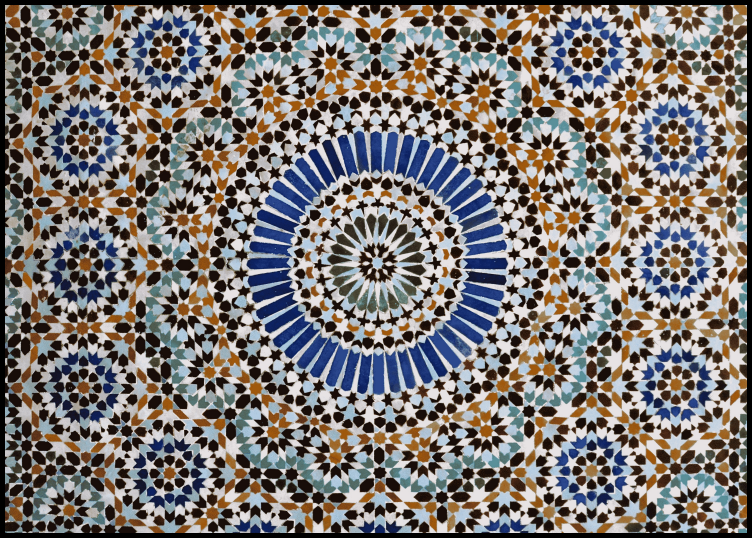
“Oh my, here we go again…”

But he still wants Ahmed to be happy



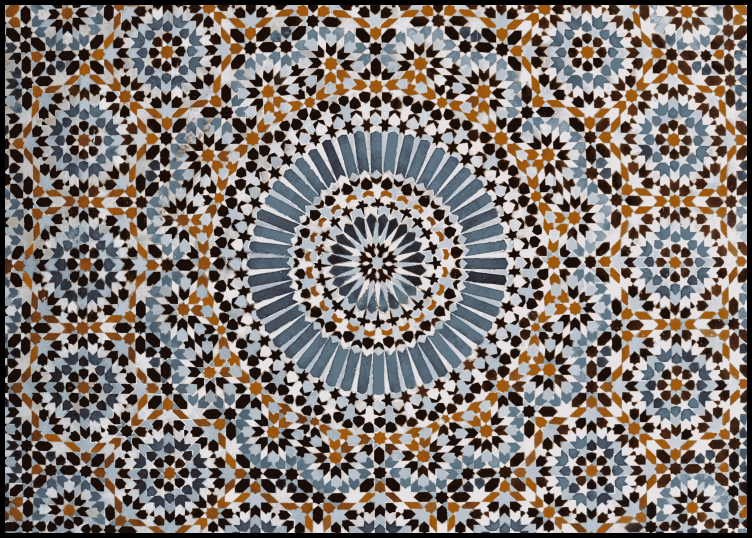
“Alright, alright, I give you these next ten with a large discount too.”

Ahmed goes to rework the mosaic further



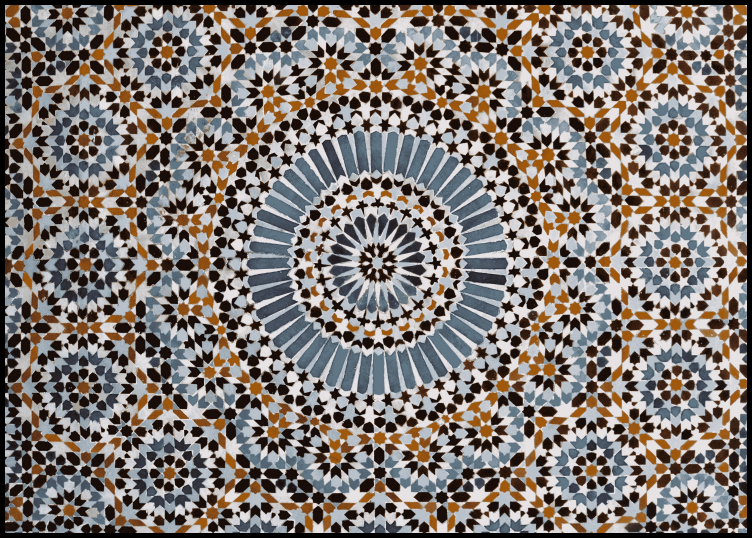
Then he bargains with Mustafa some more

An agreement is made



Eventually, Ahmed is happy with his new design taking advantage of Mustafa's discounts

An agreement is made

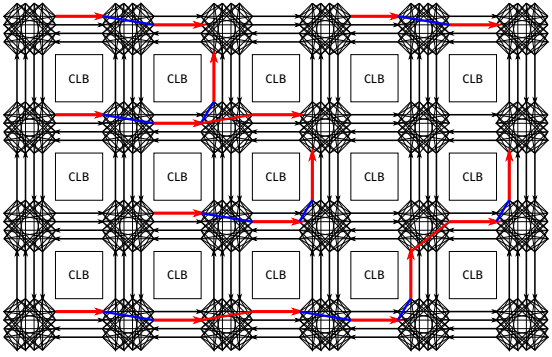


By then, Mustafa took out 87 bags—quite some work, but $\ll 1000$

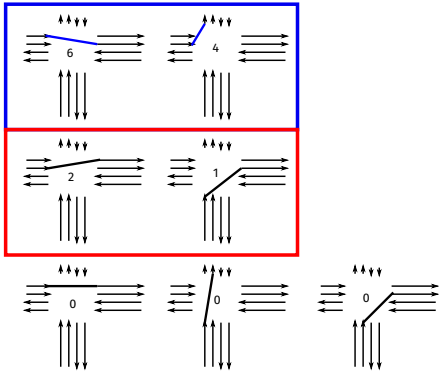
What do we do now?

Give new discounts

PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)

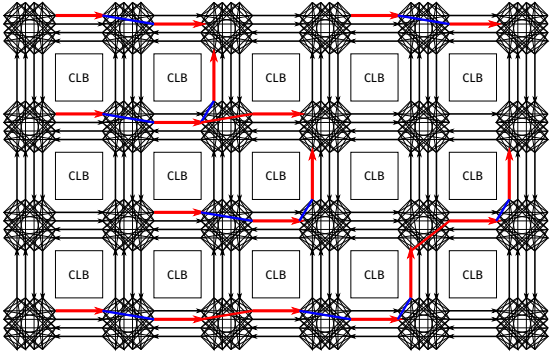


Count in how many switch-blocks each switch was used
(Ahmed counts different stones)

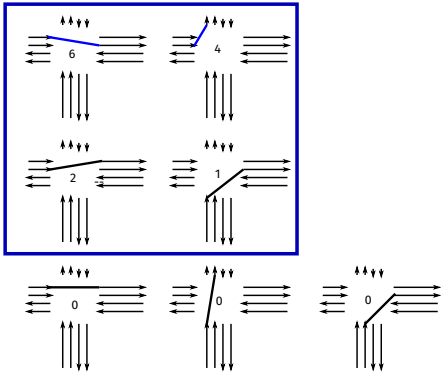


Give new discounts

PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)



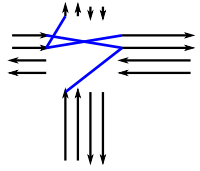
Give discounts and mark for fabrication
(Mustafa gives discount and takes out the bags)



PathFinder uses only switches marked for fabrication (blue)

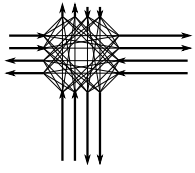
switches marked for fabrication
(kinds of stone Mustafa pledged to bring out)

discounted cost
(0)

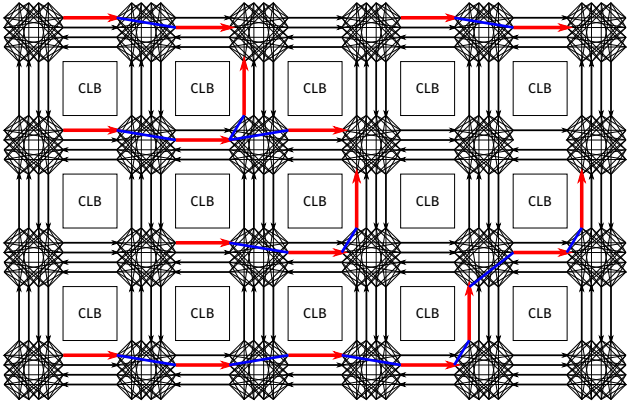


switches available to the router
(Mustafa's catalogue)

everything else
is expensive

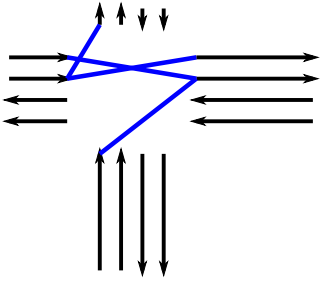


PathFinder routes circuits (each connection by a shortest path)
(Ahmed designs the mosaic)



Everybody is happy

And we have the final pattern that will be produced



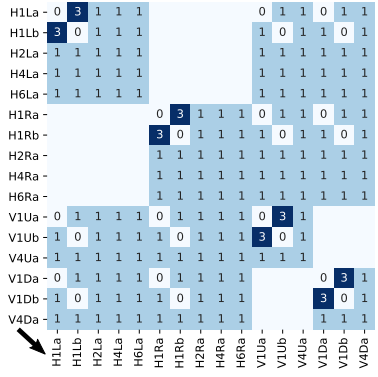
Avalanche Search [Nikolić and Ienne. Turning PathFinder Upside-Down. FPL'21]

1. switches marked for fabrication $\leftarrow \{\}$
2. all possible switches can be used at cost C
3. let PathFinder route the circuits
   (with additional switch-minimization costing; see FPL'21)
4. if no unmarked switches are used, done
5. mark n most-used unmarked switches and set their cost to 0
6. goto 3

Effectiveness of Avalanche Search

Our best manual effort (4nm; 16 wires; 564 possible switches)

<Fs> = 11.25



#switches	180		
average →	fi	fo	t[ps]
H1	10	10	16.0
H2	11	11	21.3
H4	11	11	30.8
H6	11	11	43.1
V1	12	12	24.6
V4	13	13	74.3
W(tile)	7464 nm		
CPD	1.46 ns		

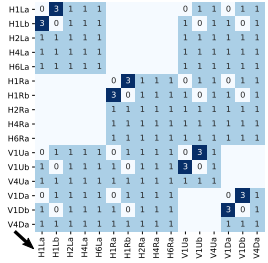
[1] Nikolić, Catthoor, Tőkei, and Ienne. Global Is the New Local. FPGA'21

Avalanche Search

- 3 circuits used in exploration
- 17 circuits used for testing
- converged in 36 iterations

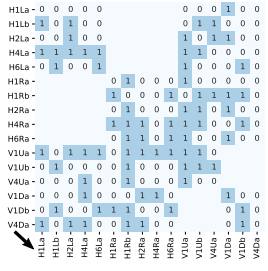
Avalanche Search

<Fs> = 11.25



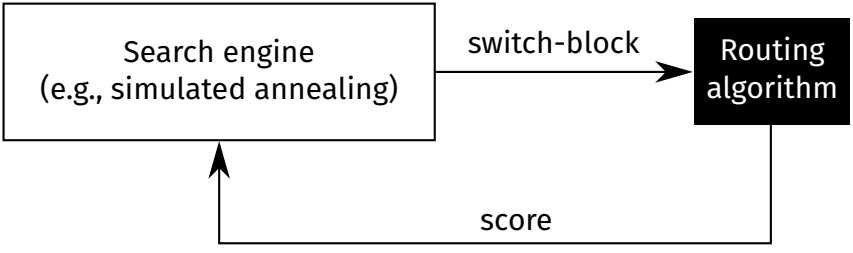
#switches	180		
average →	fi	fo	t[ps]
H1	10	10	16.0
H2	11	11	21.3
H4	11	11	30.8
H6	11	11	43.1
V1	12	12	24.6
V4	13	13	74.3
W(tile)	7464 nm		
CPD	1.46 ns		

<Fs> = 4.875



#switches	78		
average →	fi	fo	t[ps]
H1	5	3	13.9
H2	5	4	16.8
H4	4	7	27.4
H6	5	5	35.7
V1	7	6	21.8
V4	2	5	70.1
W(tile)	6792 nm -9%		
CPD	1.38 ns -5.48%		

And what about the previous state of the art?



[1] Lin, Wawrzynek, El Gamal. Exploring FPGA routing architecture stochastically. TCAD'10

And what about the previous state of the art?

Prior methods

- try to guess a set of switches that would perform well
- use the router as a black box to evaluate performance of each set
- there are exponentially many sets
- each evaluation can take minutes and even hours

Not scalable

And what about the previous state of the art?

Proposed method

- lets the router itself construct the preferred set of switches
- requires no set listing

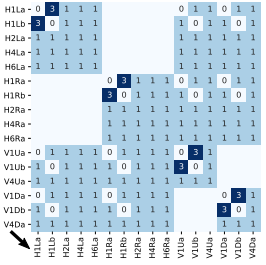
Simulated annealing

- inspired by [1]
- each move is addition or removal of a single switch
- low-temperature anneal of the manual switch-pattern
- 100 temperature changes
- 100 moves per temperature

[1] Lin, Wawrzynek, El Gamal. Exploring FPGA routing architecture stochastically. TCAD'10

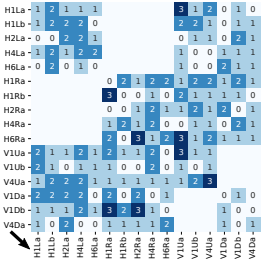
Simulated annealing

<Fs> = 11.25



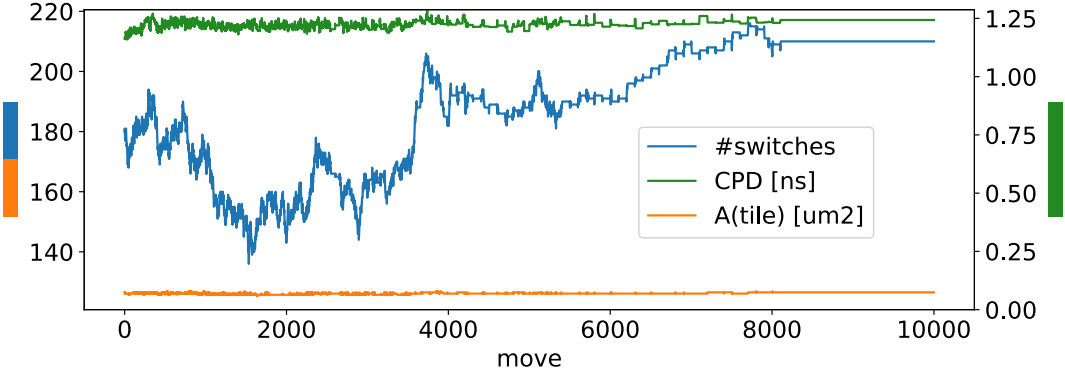
#switches	180		
average →	fi	fo	t[ps]
H1	10	10	16.0
H2	11	11	21.3
H4	11	11	30.8
H6	11	11	43.1
V1	12	12	24.6
V4	13	13	74.3
W(tile)	7464 nm		
CPD	1.46 ns		

<Fs> = 13.125



#switches	210		
average →	fi	fo	t[ps]
H1	13	13	19.6
H2	14	11	24.1
H4	16	12	32.1
H6	9	13	47.3
V1	14	15	29.2
V4	13	15	86.8
W(tile)	7488 nm +0.32%		
CPD	1.55 ns +6.16%		

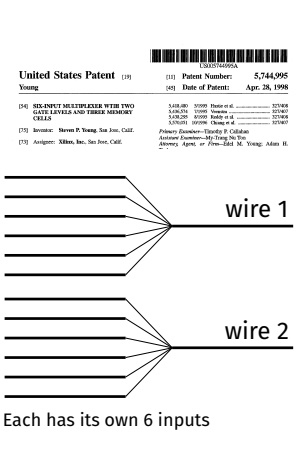
Simulated annealing



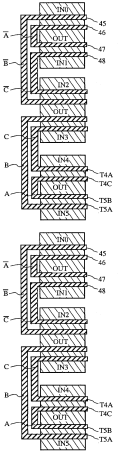
With a better cost function, directed moves, etc., it would likely work better

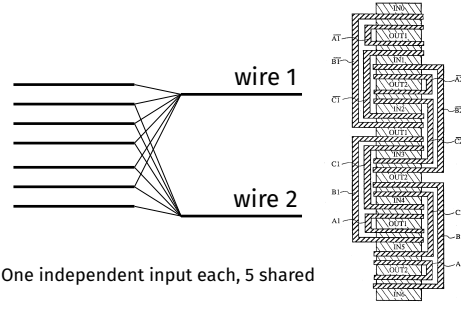
Regularization of Solutions

Some sets of M switches are easy to fabricate, others impossible



Each has its own 6 inputs



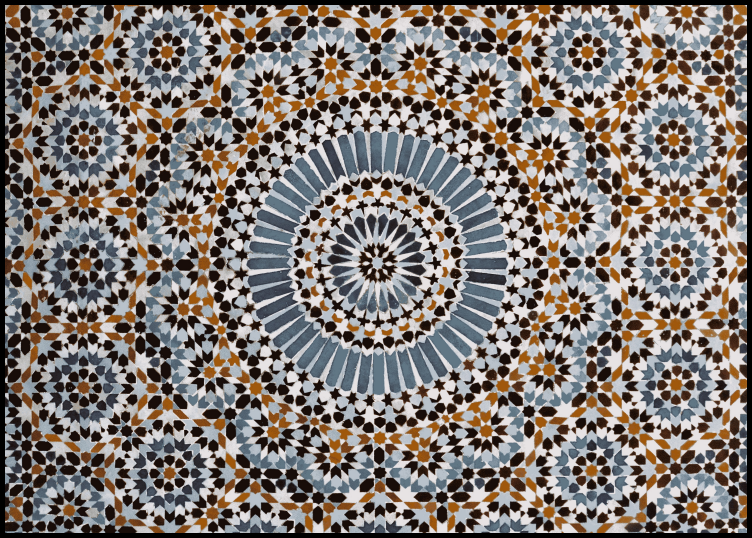


One independent input each, 5 shared

How do we find those M that are as close as possible to the ones that the router wants, but are also easy to fabricate?

Mustafa had a similar problem

An agreement is made



By then, Mustafa took out 87 bags, $\ll 1000$

An agreement is made

But, to take out these 87, he had to move 492 bags in total

An agreement is made

Since the ones he pledged to take out were scattered around the pile

Mustafa's new problem

"How do I give Ahmed 87 kinds of stone that are as close as possible to the ones he wants, but so that I never move more than 150 bags?"

Two years after, Ahmed sails to Constantinople again

But Mustafa is smarter this time

Location of bags matters

1. Ultramarine medium vedge (576 pieces) ✔
2. Alabaster needle (229 pieces)
3. Coral hexagon (143 pieces) ✔

too deep inside and
far from 1. don't discount

8.

117. ✔

11. ✔

134. ✔

23. ✔

135. ✔

79.

113. ✔

1000. Verdigris tiny star (2 pieces)

Mustafa ticks 87 bags that Ahmed desires the most,
but which don't violate his constraints (on moving $\leqslant 150$ bags)

Location of bags matters

1. Ultramarine medium vedge (576 pieces) ✔
2. Alabaster needle (229 pieces)
3. Coral hexagon (143 pieces) ✔

too deep inside and far from 1. don't discount

8.

117. ✔

11. ✔

big discount, take out now

134. ✔

23. ✔

135. ✔

79.

113. ✔

1000. Verdigris tiny star (2 pieces)

“For stone kinds ticked in blue, I give you a big discount.
For any number of pieces!”

Location of bags matters

1. Ultramarine medium vedge (576 pieces) ✔
2. Alabaster needle (229 pieces)
3. Coral hexagon (143 pieces) ✔

too deep inside and far from 1. don't discount

8.

117. ✔

11. ✔

big discount, take out now

134. ✔

23. ✔

135. ✔

79.

113. ✔

1000. Verdigris tiny star (2 pieces)

“But once I take the bags out, you have to buy. Deal?”

Constraints are hard

1. Ultramarine medium vedge (576 pieces) ✔
2. Alabaster needle (229 pieces)
3. Coral hexagon (143 pieces) ✔

too deep inside and far from 1. don't discount

8.

117. ✔

11. ✔

big discount, take out now

134. ✔

23. ✔

135. ✔

79.

113. ✔

1000. Verdigris tiny star (2 pieces)

Mustafa always ends bargaining on a complete ticked list

Important takeaways

Exact constraints which Mustafa applies while ticking Ahmed's list depend on the layout of his pile

Important takeaways

For another merchant they will be different

Important takeaways

But the algorithm is identical
(and hence general)

Let’s see how this works for **REGULAR** switch-block exploration

Key points: constructing a feasible solution

1. Encode **ANY** regularity constraints
   (e.g., for layout or CAD tools)
   as an Integer Linear Program (ILP)

2. Let ILP maximize PathFinder's "desire" (usage of different switches)
   while satisfying the above constraints

Key points: updating costs

For switches in the ILP solution (satisfying regularity constraints)

1. Give a big discount to *n* most-used unmarked switches

For switches not in the ILP solution (violating regularity constraints)

1. Assign full cost with no discount

Key points: ensuring that constraints are met

- Final pattern is the last ILP solution

Regularizing Avalanche Search

1. switches marked for fabrication $\leftarrow \{\}$
2. all possible switches can be used at cost C
3. let PathFinder route the circuits
4. if iterations expanded $\implies$ return the last ILP solution
5. solve the ILP, always retaining marked switches
6. mark n most-used unmarked switches from ILP's solution and set their cost to 0
7. goto 3

Regularizing Avalanche Search

ILP encodings and experimental results
for many forms of regularity available in

[1] Nikolić and Ienne. Regularity Matters. FPGA'23

Conclusions

- we now have a method for automatically designing switch-patterns without explicitly listing and testing individual solutions (previously a fundamental bottleneck)
- we can also make the solutions respect arbitrary constraints (previously not possible)
- the method could be useful for other aspects of interconnect representable as a graph (e.g., connectivity inside CLB, optimizing lengths of channel wires, sharing configuration bits…)

Thank you for attention

stefan.nikolic@epfl.ch

If you found the story of Mustafa and Ahmed helpful

Please consider helping people in their region