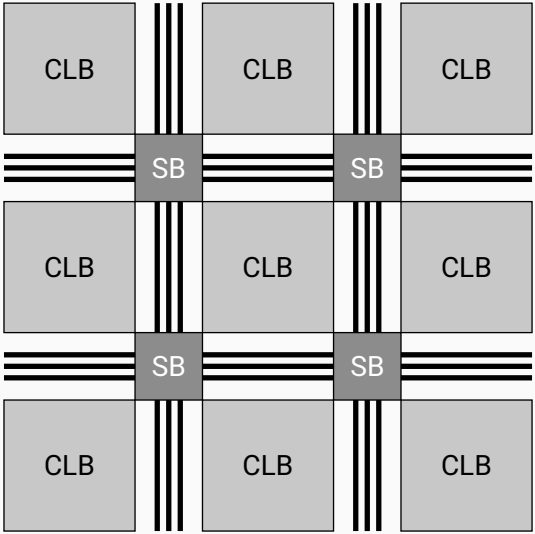
Timing-Driven Placement for FPGA Architectures with Dedicated Routing Paths

S. Nikolić, G. Zgheib*, and P. Ienne

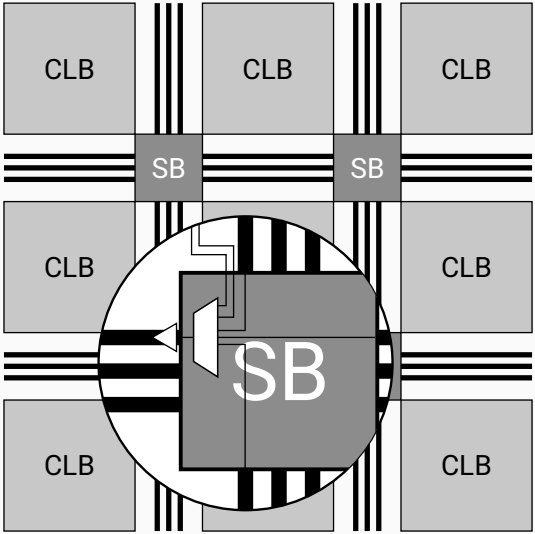
FPL'20, Göteborg, 01.09.2020

École Polytechnique Fédérale de Lausanne
*Intel Corporation

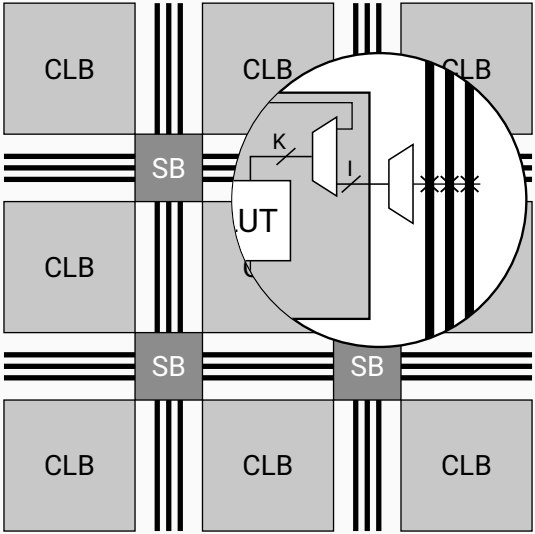
Field-Programmable Gate Array



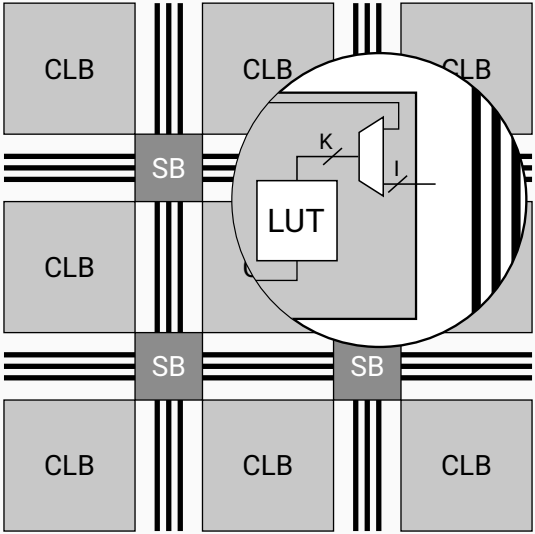
Price of Programmability: Switch Block MUX



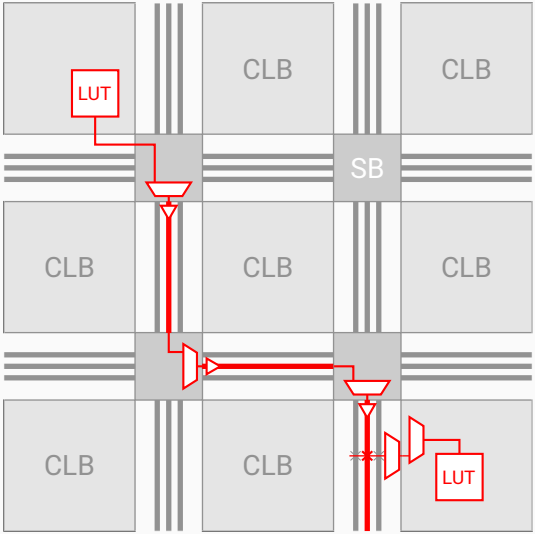
Price of Programmability: Connection Block MUX



Price of Programmability: Crossbar MUX



Many MUXes $\Longrightarrow$ Large Delay



Direct Connections: Switch Block-to-Switch Block

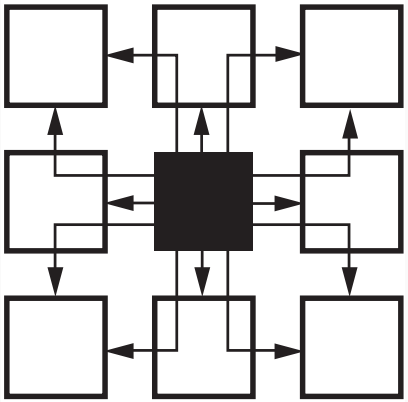
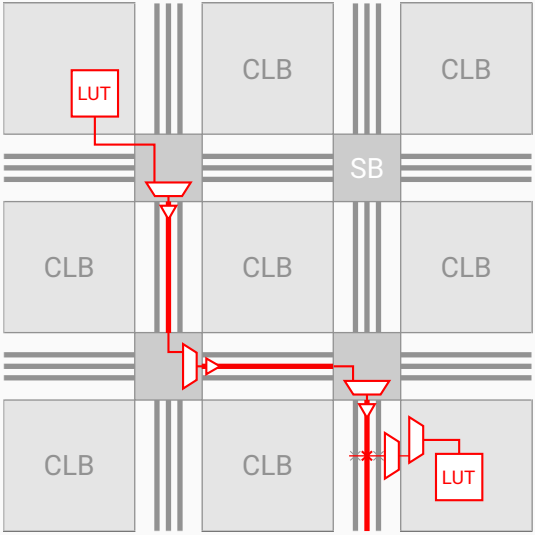


Figure 48: **Hierarchical Routing Resources**



Direct Connections: Switch Block-to-Switch Block

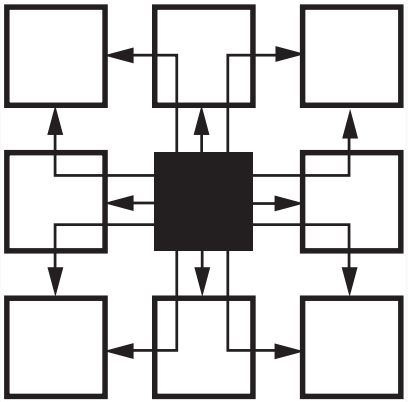
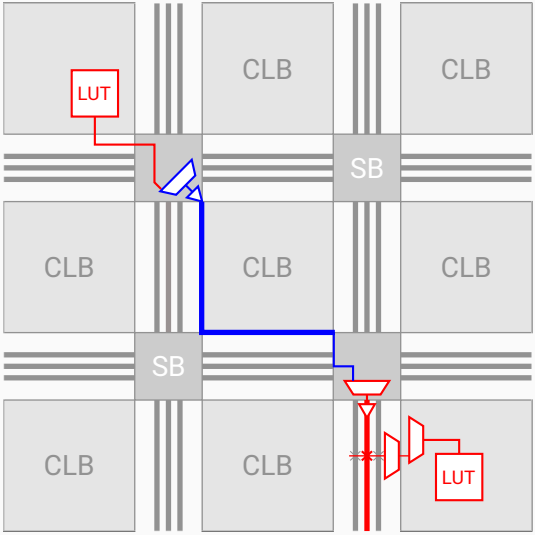


Figure 48: **Hierarchical Routing Resources**



Direct Connections: Cluster-to-Cluster

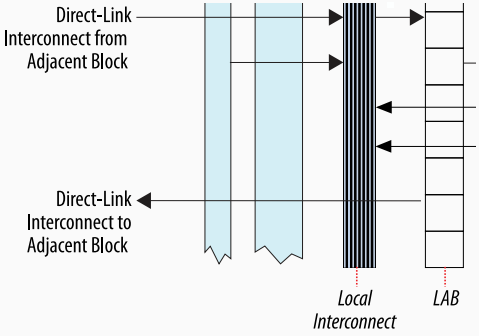
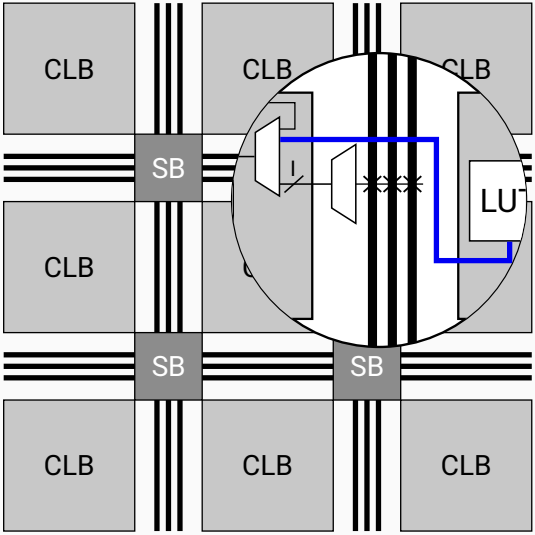


Figure 1. Intel Stratix 10 LAB Structure and Interconnects Overview



Direct Connections: LUT-to-LUT

Xilinx Adaptive Compute Acceleration Platform: Versal™ Architecture

Brian Gaide, Dinesh Gaitonde, Chirag Ravishankar, Trevor Bauer
bgaide@xilinx.com,dineshg@xilinx.com,chiragr@xilinx.com,trevor@xilinx.com
Xilinx Inc.

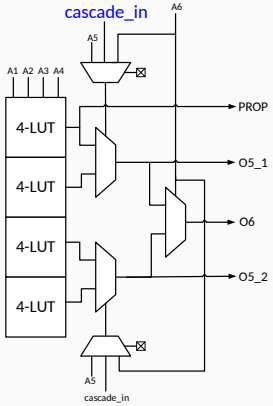
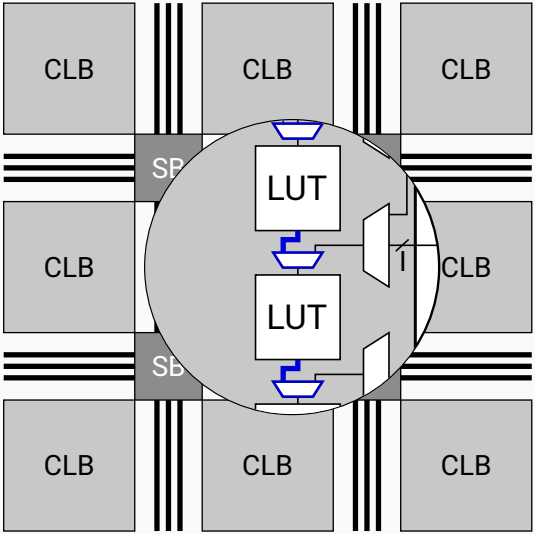


Figure 5: 6LUT Comparison between UltraScale and Versal



Direct Connections: Two Questions

1. Where to put them?
   (metal and area cost, increased capacitive loading, etc.)

Direct Connections: Two Questions

1. Where to put them?
   (metal and area cost, increased capacitive loading, etc.)

2. How to use them effectively?

Direct Connections: Two Questions

1. Where to put them?
   (metal and area cost, increased capacitive loading, etc.)

   Our work at FPGA'20

2. How to use them effectively?

Direct Connections: Two Questions

1. Where to put them?
   (metal and area cost, increased capacitive loading, etc.)

   Our work at FPGA'20

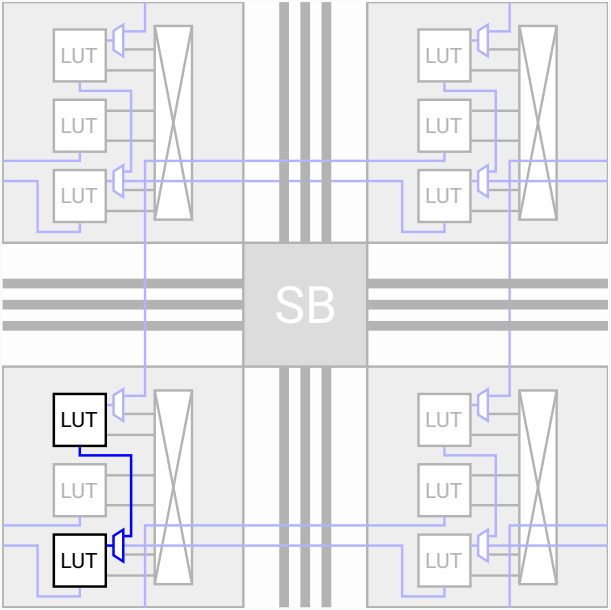
2. How to use them effectively?

   This work

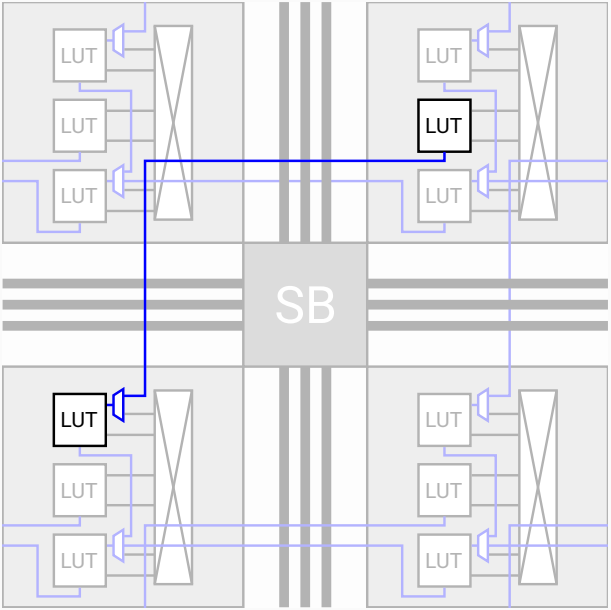
Outline

Target Architectures

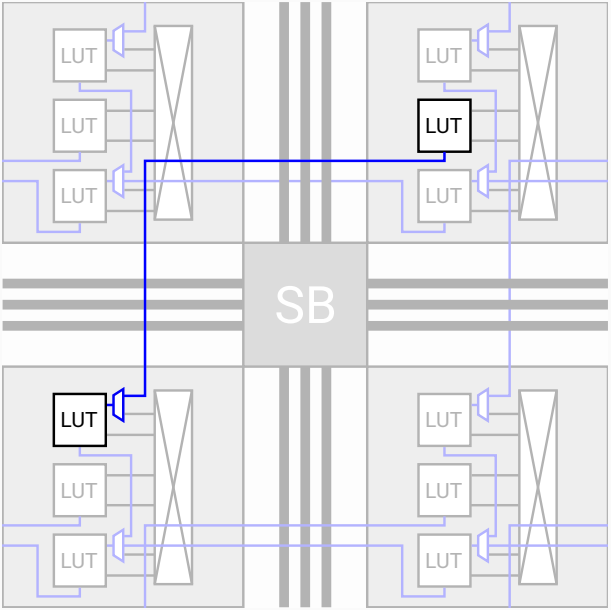
- LUT-to-LUT connections



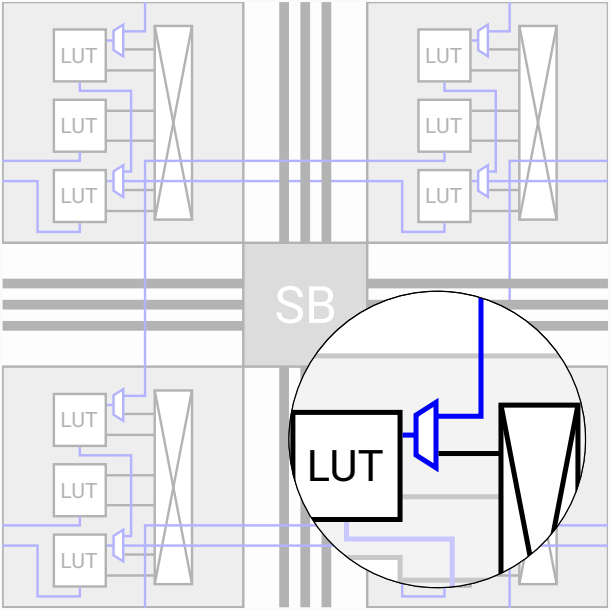
- LUT-to-LUT connections
- Can span multiple clusters



- LUT-to-LUT connections
- Can span multiple clusters
- Optionally used $\Longrightarrow$
  keeps all the flexibility of the programmable interconnect

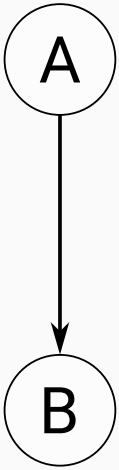


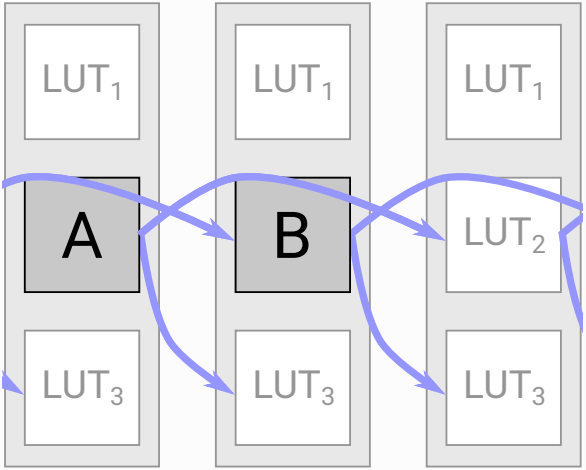
- LUT-to-LUT connections
- Can span multiple clusters
- Optionally used $\Longrightarrow$
  keeps all the flexibility of the programmable interconnect



Motivation

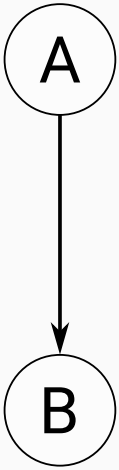
FPGA'20: Swapping LUTs within Clusters

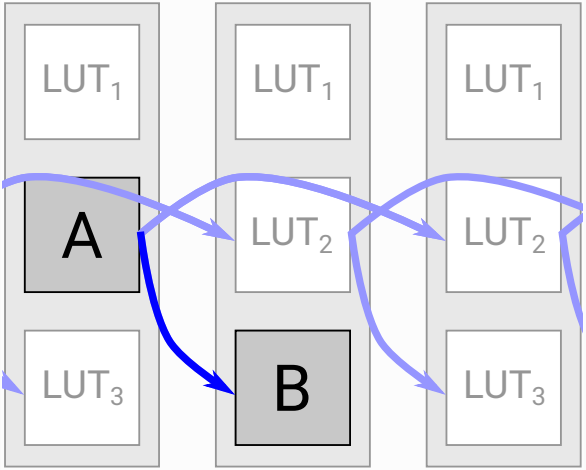




No direct connection between A and B

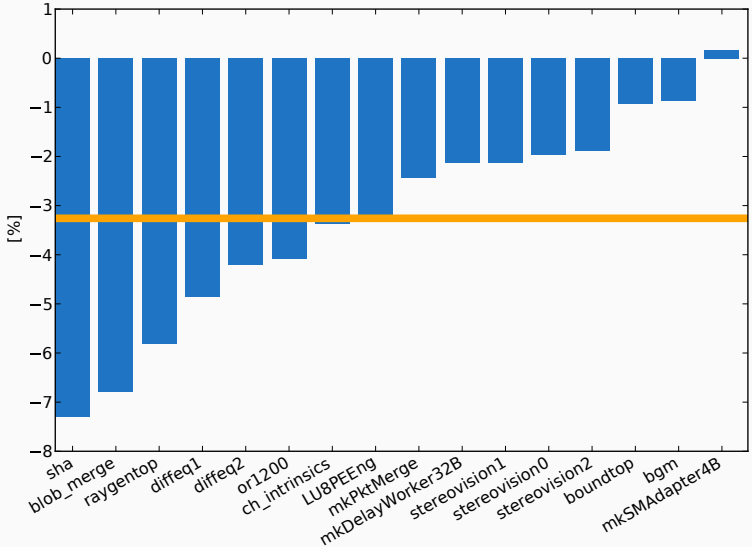
FPGA'20: Swapping LUTs within Clusters



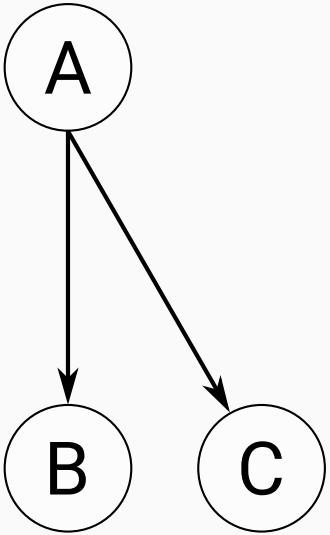


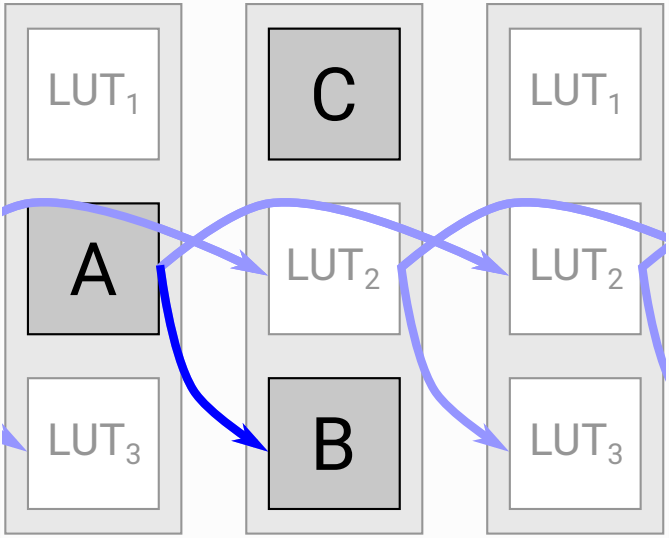
Direct connection between A and B

FPGA'20: Delay Improvement due to Direct Connections

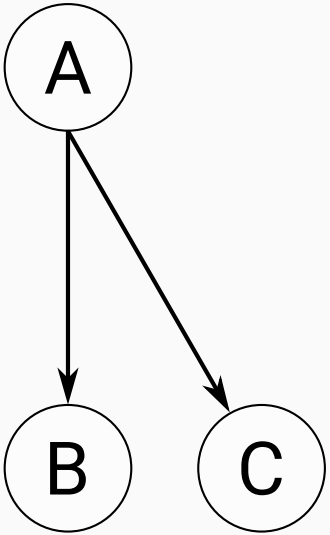


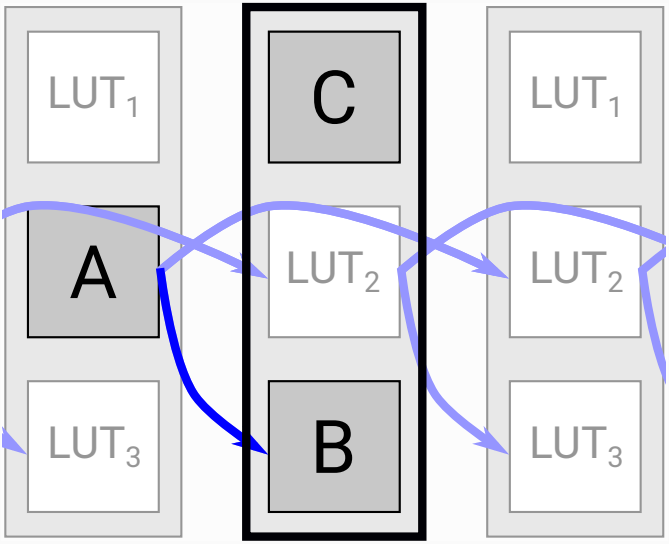
FPGA'20: Missed Opportunities



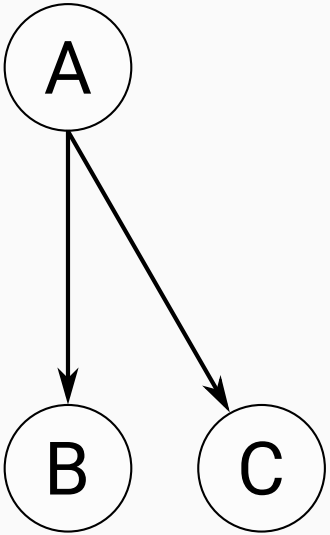


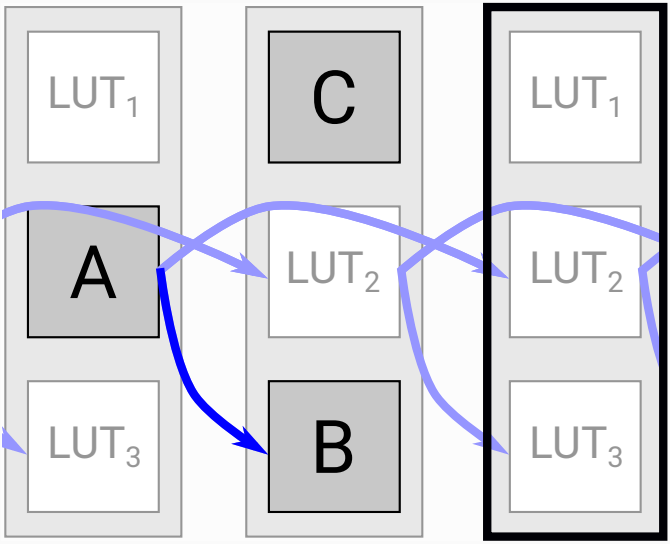
FPGA'20: Missed Opportunities



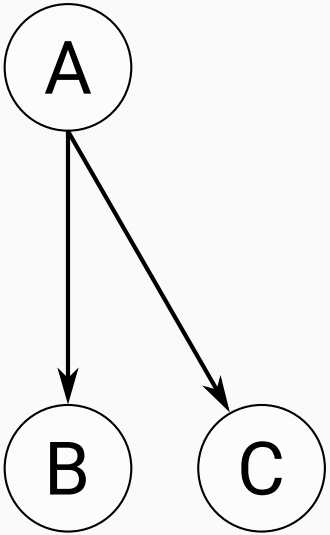


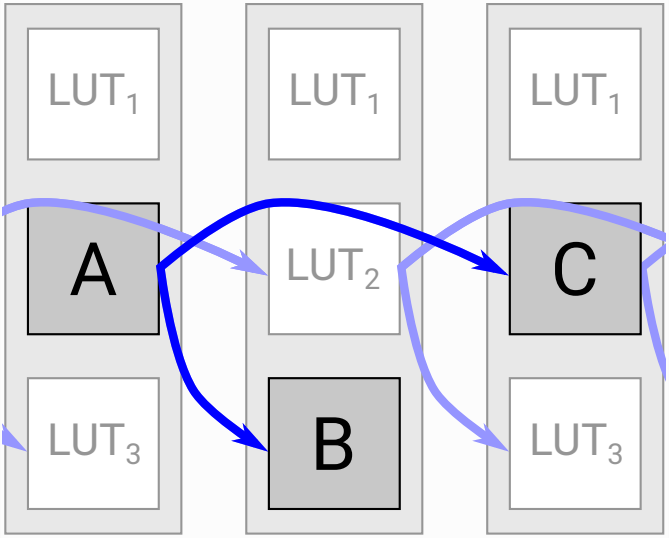
FPGA'20: Missed Opportunities





FPGA'20: Missed Opportunities





How much could we gain?

FPGA'20: Missed Opportunities

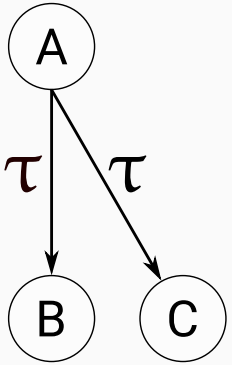
$\tau = \langle t_d(u, v) \rangle,$

$\forall (u, v) : (u, v)$ is a direct connection

FPGA'20: Missed Opportunities

$\tau = \langle t_d(u, v) \rangle$,

$\forall (u, v) : (u, v)$ is a direct connection

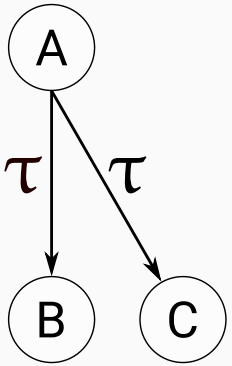


FPGA'20: Missed Opportunities

$\tau = \langle t_d(u, v) \rangle$,

$\forall (u, v) : (u, v)$ is a direct connection

$\sim 19\%$ lower geomean delay



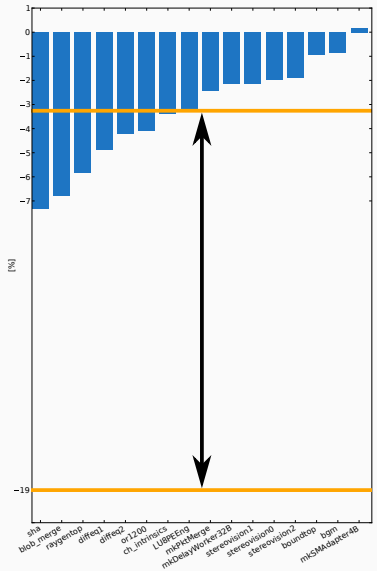
FPGA'20: Missed Opportunities

Unlikely to meet in practice...

FPGA'20: Missed Opportunities

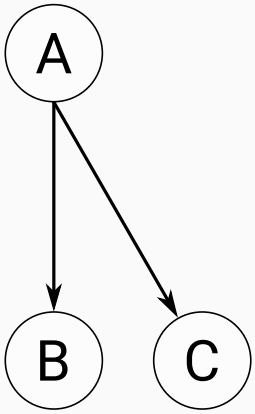
Unlikely to meet in practice...

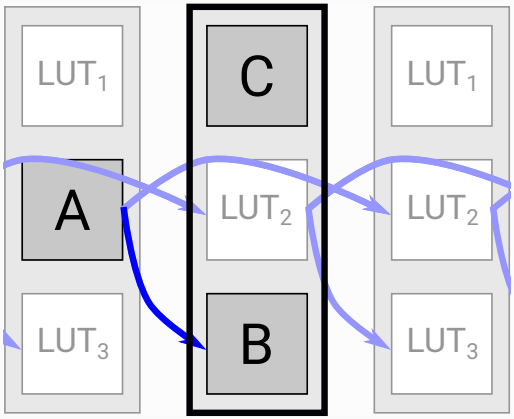
But, leaves a big
margin for improvement



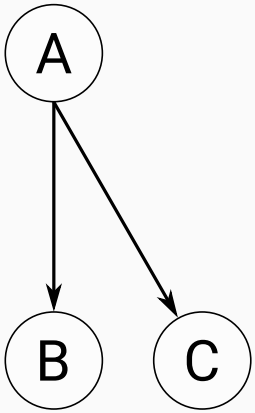
General Approach

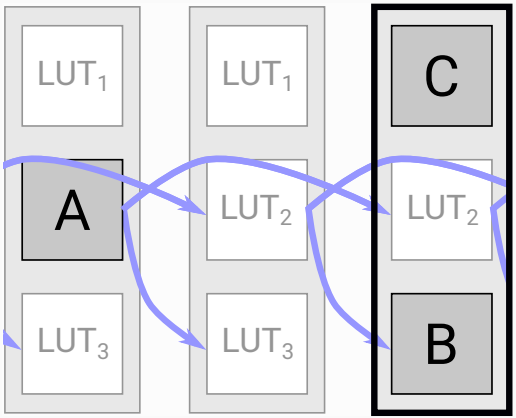
Placing Clusters is not Sufficient





Placing Clusters is not Sufficient



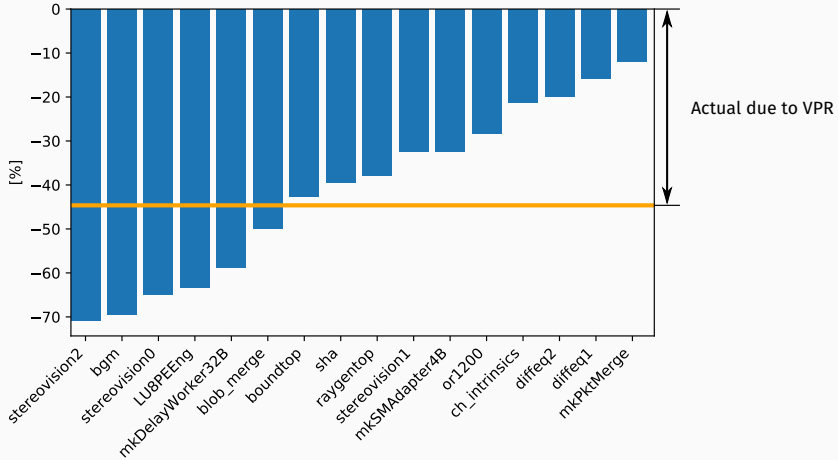


Flat placement of LUTs

Flat placement of LUTs

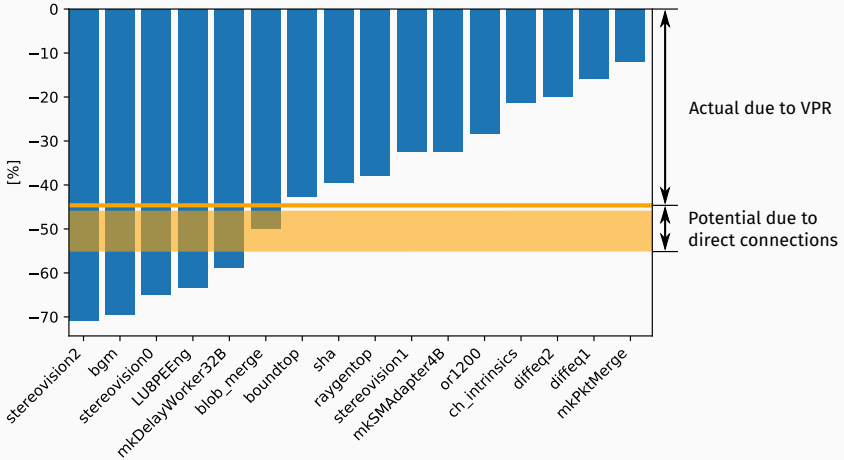
An order of magnitude more
placeable objects and placement positions

Generic vs Dedicated Placement



Delay improvement over initial random placement

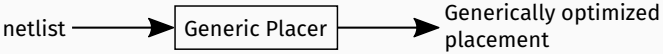
Generic vs Dedicated Placement



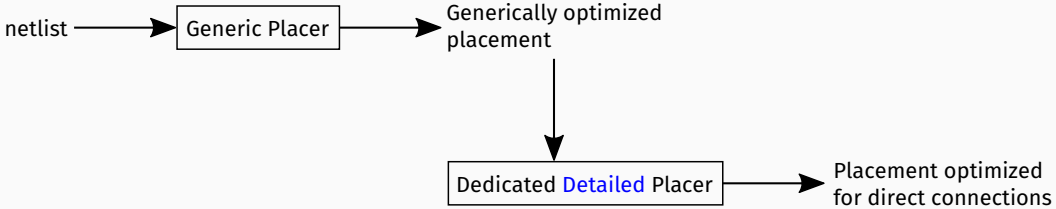
Delay improvement over initial random placement

When to Consider Direct Connections?

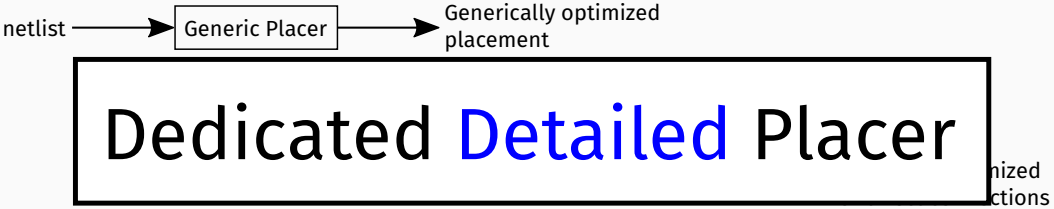
When to Consider Direct Connections?



When to Consider Direct Connections?



When to Consider Direct Connections?



Placement Algorithm

Timing-Driven Detailed Placement

0. All nodes (LUTs) are assigned a starting position

Timing-Driven Detailed Placement

0. All nodes (LUTs) are assigned a starting position

1. Select a subset of nodes

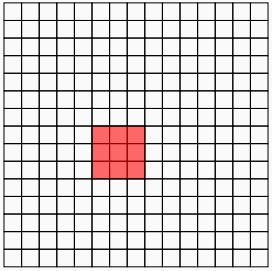
Timing-Driven Detailed Placement

0. All nodes (LUTs) are assigned a starting position

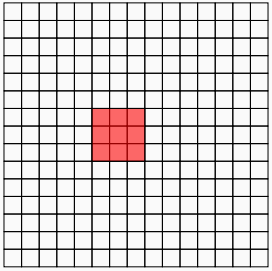
1. Select a subset of nodes
2. Move them to reduce the critical path delay

Which Nodes to Move?

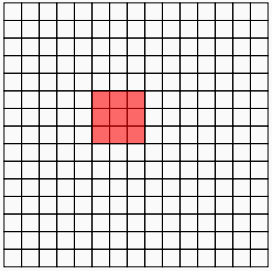
Determining Movable Nodes: Sliding Window



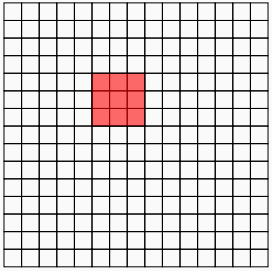
Determining Movable Nodes: Sliding Window



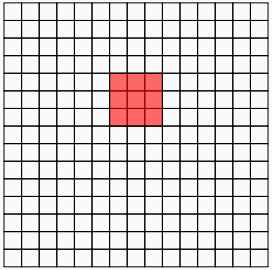
Determining Movable Nodes: Sliding Window



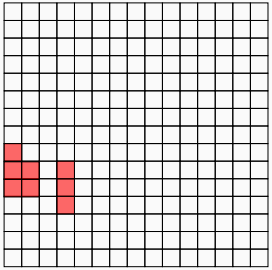
Determining Movable Nodes: Sliding Window



Determining Movable Nodes: Sliding Window

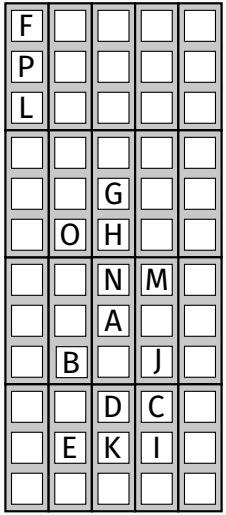


Determining Movable Nodes: Critical Path

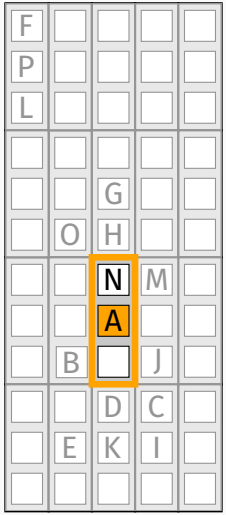


Generalization

Movement Constraints

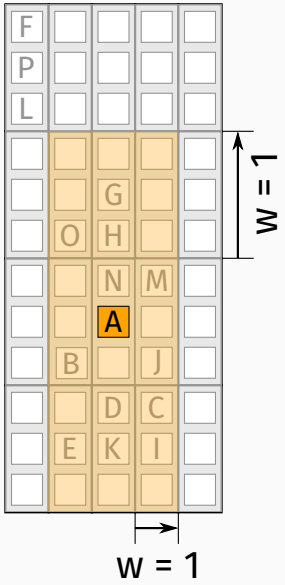


Movement Constraints



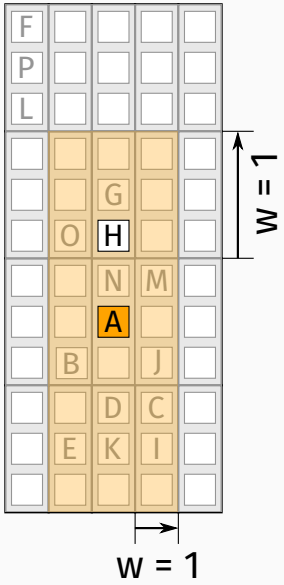
- Each node can move to any position in the *w*-bounded square around its starting cluster

Movement Constraints



- Each node can move to any position in the *w*-bounded square around its starting cluster

Movement Constraints



- Each node can move to any position in the *w*-bounded square around its starting cluster
- Overlaps with stationary nodes removed by postprocessing

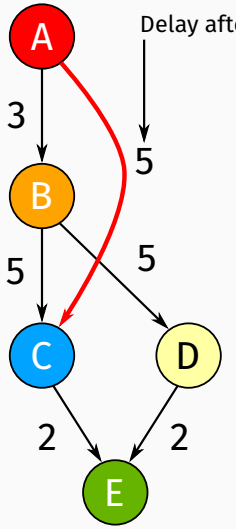
Improving Connection Delays

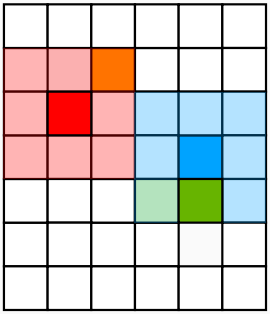
Each circuit connection (u, v) has initial delay $\tau_{u,v}$

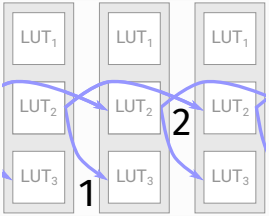
Improving Connection Delays

Implementing by a direct connection can improve it by $0 \leqslant imp_{u,v} \leqslant l_{u,v} = const.$

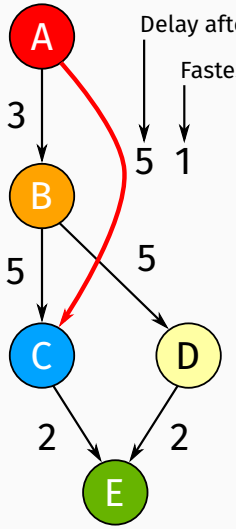
Improving Connection Delays

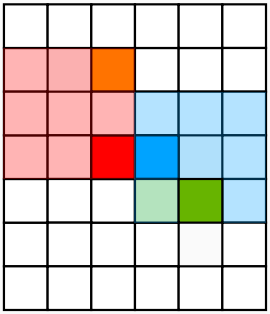


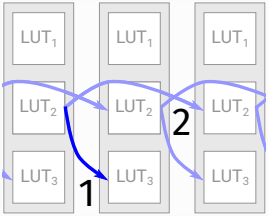




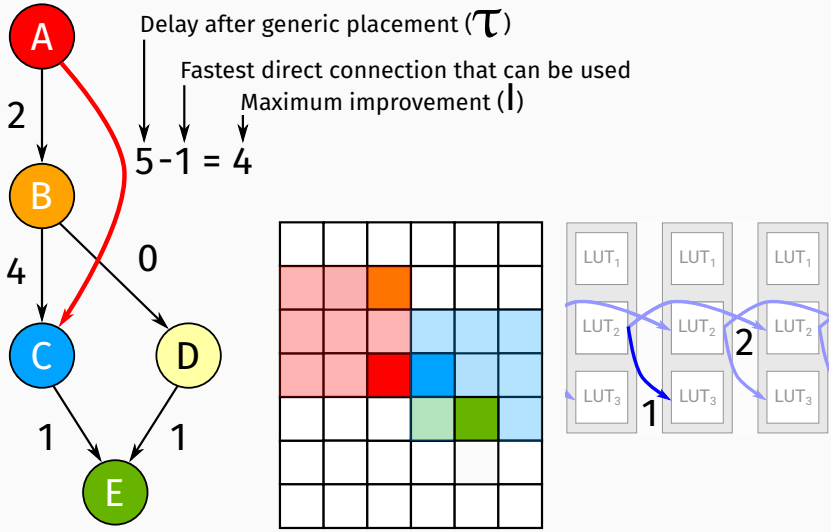
Improving Connection Delays







Improving Connection Delays



Improving Connection Delays

1. Assign *imp*-variables values, s.t. critical path delay $\leqslant$ some target D

Improving Connection Delays

1. Assign *imp*-variables values, s.t. critical path delay $\leqslant$ some target *D*
2. All nodes incident to an edge with $imp \neq 0$ are movable

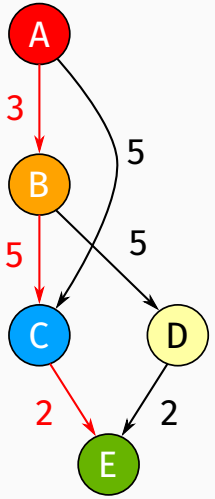
Improving Connection Delays

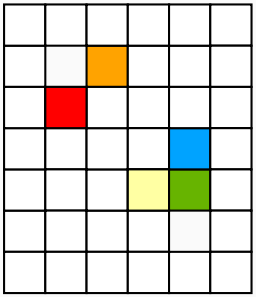
1. Assign *imp*-variables values, s.t. critical path delay $\leqslant$ some target D
2. All nodes incident to an edge with $imp \neq 0$ are movable

$$\min |\{(u, v) : imp_{u,v} \neq 0\}|$$

Improving Connection Delays: An Example

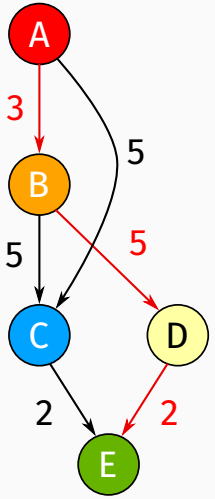
Two critical paths with delay 10

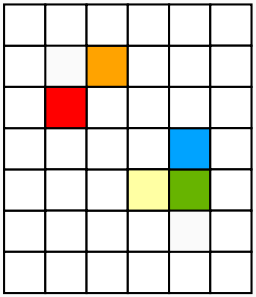




Improving Connection Delays: An Example

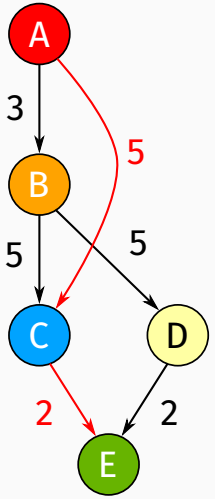
Two critical paths with delay 10

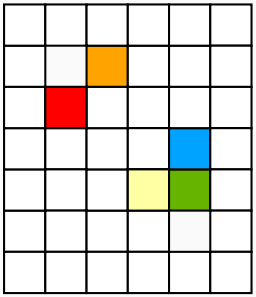




Improving Connection Delays: An Example

Two critical paths with delay 10
One path with delay 7

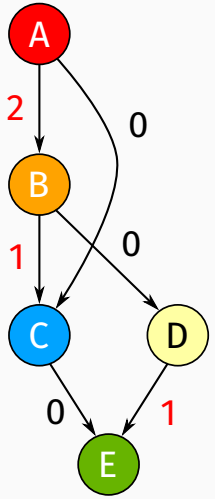


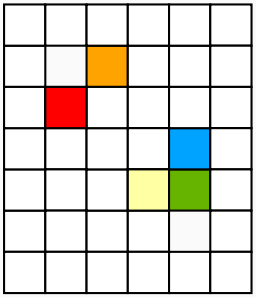


Improving Connection Delays: An Example

Two critical paths with delay 10
One path with delay 7

$D = 7$





Improving Connection Delays: Selection LP

1. ... s.t. critical path delay $\leqslant D$
2. ...

Improving Connection Delays: Selection LP

1. ... s.t. critical path delay $\leqslant D$
2. ...

$$
\begin{aligned}
\text{s.t.} \quad & t_{u,v} = \tau_{u,v} - imp_{u,v} \\
& ta_v \geqslant ta_u + t_{u,v} \\
& ta_u \leqslant ta_{max} \\
& ta_{max} \leqslant D
\end{aligned}
$$

Improving Connection Delays: Selection LP

1. ... s.t. critical path delay $\leqslant D$
2. ...

$$\min |\{(u,v) : imp_{u,v} \neq 0\}|$$

$$\begin{aligned}
\text{s.t.} \quad & t_{u,v} = \tau_{u,v} - imp_{u,v} \\
& ta_v \geqslant ta_u + t_{u,v} \\
& ta_u \leqslant ta_{max} \\
& ta_{max} \leqslant D
\end{aligned}$$

Improving Connection Delays: Selection LP

1. ... s.t. critical path delay $\leqslant D$
2. ...

~~$\min |\{(u,v) : imp_{u,v} \neq 0\}|$~~

$\min \sum_{(u,v)} imp_{u,v}$

$$\text{s.t.} \quad t_{u,v} = \tau_{u,v} - imp_{u,v}$$

$$ta_v \geqslant ta_u + t_{u,v}$$

$$ta_u \leqslant ta_{max}$$

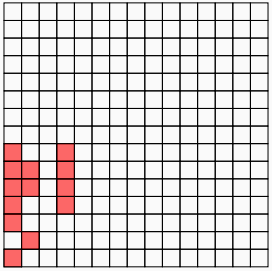
$$ta_{max} \leqslant D$$

Improving Connection Delays: Selection LP

$$
\begin{aligned}
&\min \textstyle\sum_{(u,v)} imp_{u,v} \\
&\text{s.t.} \quad t_{u,v} = \tau_{u,v} - imp_{u,v} \\
&\qquad\quad ta_v \geqslant ta_u + t_{u,v} \\
&\qquad\quad ta_u \leqslant ta_{max} \\
&\qquad\quad ta_{max} \leqslant D
\end{aligned}
$$

[1]Hambrusch and Tu, “Edge weight reduction problems in directed acyclic graphs”, *J. Algorithms*, 1997

Determining Movable Nodes: Selection LP



How to Move the Selected Nodes?

Different Options

Heuristic Methods:

Different Options

Heuristic Methods:

...

...

...

Different Options

Heuristic Methods:

...

...

...

Exact Methods:

Different Options

Heuristic Methods:	Exact Methods:
...	SAT
...	SMT
...	ILP

Different Options

Exact Methods:

SAT

SMT

ILP

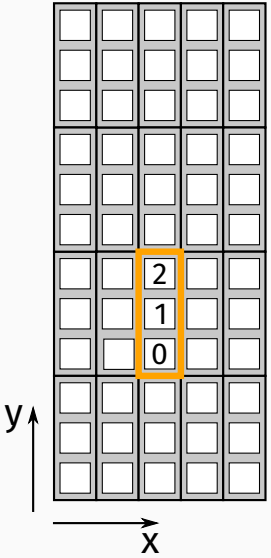
Different Options

Exact Methods:

ILP

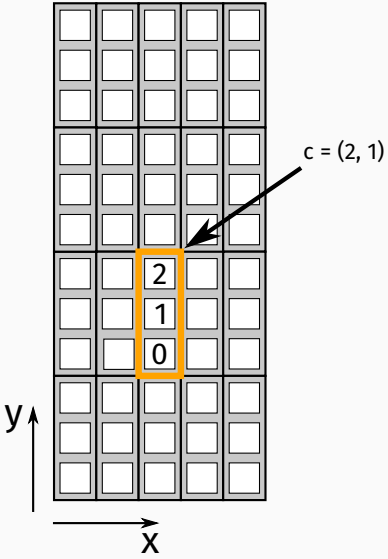
Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$



Naive ILP: Describing a Particular Placement

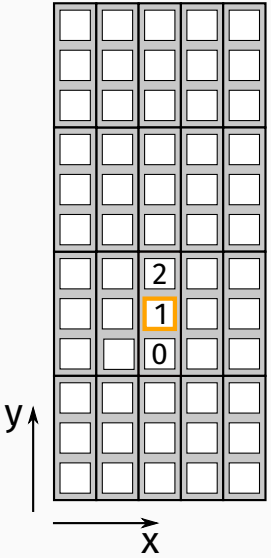
Cluster position: $c = (x, y)$



Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$

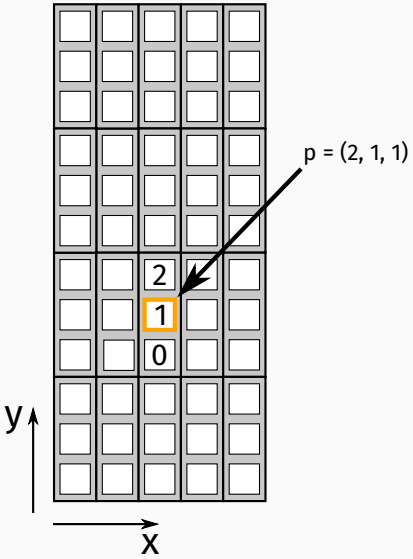
LUT position: $p = (x, y, i)$



Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$

LUT position: $p = (x, y, i)$

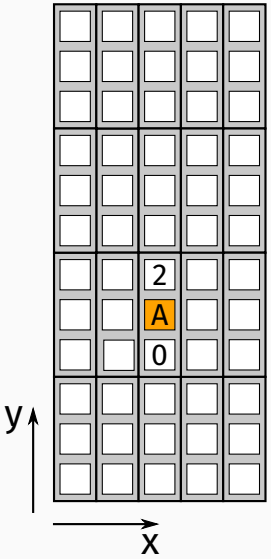


Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$

LUT position: $p = (x, y, i)$

Introduce: $x_{u,p} \in \{0, 1\}$

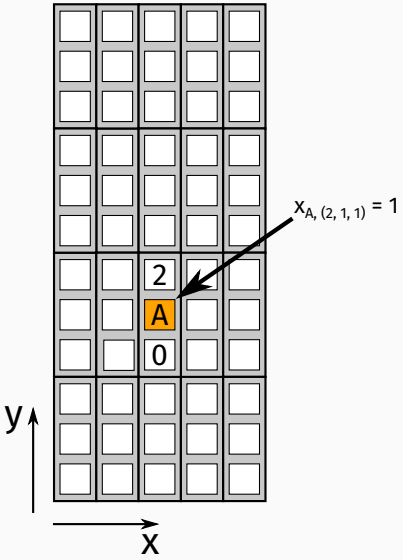


Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$

LUT position: $p = (x, y, i)$

Introduce: $x_{u,p} \in \{0, 1\}$

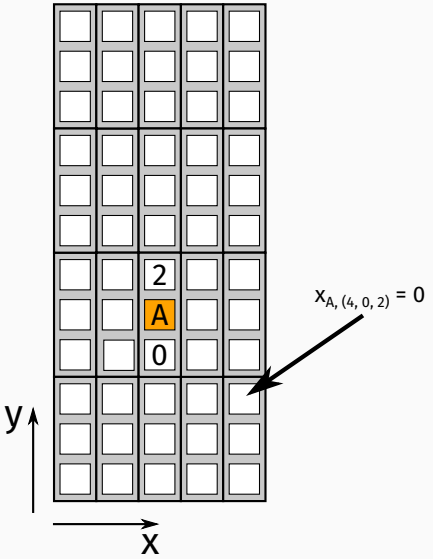


Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$

LUT position: $p = (x, y, i)$

Introduce: $x_{u,p} \in \{0, 1\}$

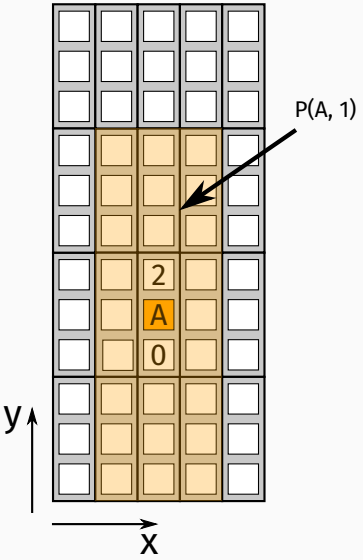


Naive ILP: Describing a Particular Placement

Cluster position: $c = (x, y)$

LUT position: $p = (x, y, i)$

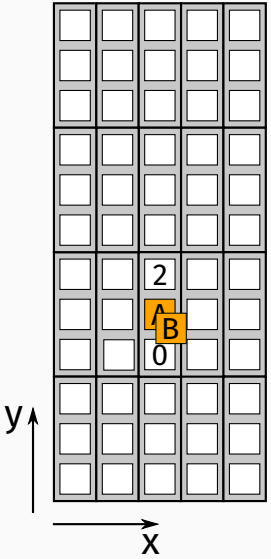
Introduce: $x_{u,p} \in \{0, 1\}, \forall p \in P(u, w)$



Naive ILP: Describing Any Legal Placement

No overlaps between movable nodes:

$\sum_{u \in V_m} x_{u,p} \leqslant 1, \forall p$



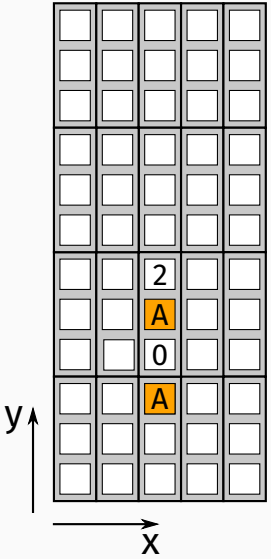
Naive ILP: Describing Any Legal Placement

No overlaps between movable nodes:

$\sum_{u \in V_m} x_{u,p} \leqslant 1, \forall p$

Each node uniquely placed:

$\sum_{p \in P(u,w)} x_{u,p} = 1, \forall u$



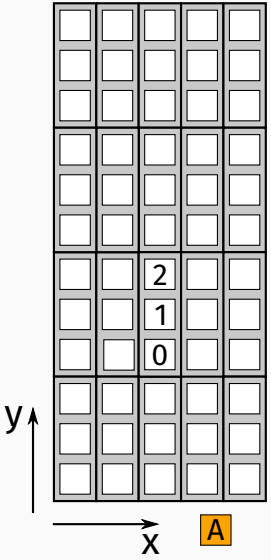
Naive ILP: Describing Any Legal Placement

No overlaps between movable nodes:

$\sum_{u \in V_m} x_{u,p} \leqslant 1, \forall p$

Each node uniquely placed:

$\sum_{p \in P(u,w)} x_{u,p} = 1, \forall u$



Naive ILP: Timing

Arrival times: same as Selection LP

Naive ILP: Timing

Arrival times: same as Selection LP

Connection delay: $t_{u,v} = \sum_{p_u \in P(u,w), p_v \in P(v,w)} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$

Naive ILP: Timing

Arrival times: same as Selection LP

Connection delay: $t_{u,v} = \sum_{p_u \in P(u,w), p_v \in P(v,w)} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$

Naive ILP: Encoding Efficiency

$$t_{u,v} = \sum_{p_u \in P(u,w), p_v \in P(v,w)} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

Naive ILP: Encoding Efficiency

$$t_{u,v} = \sum_{p_u \in P(u,w), p_v \in P(v,w)} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

$(2w + 1)^2 N$

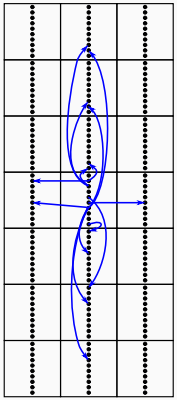
Naive ILP: Encoding Efficiency

$$t_{u,v} = \sum_{p_u \in P(u,w), p_v \in P(v,w)} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

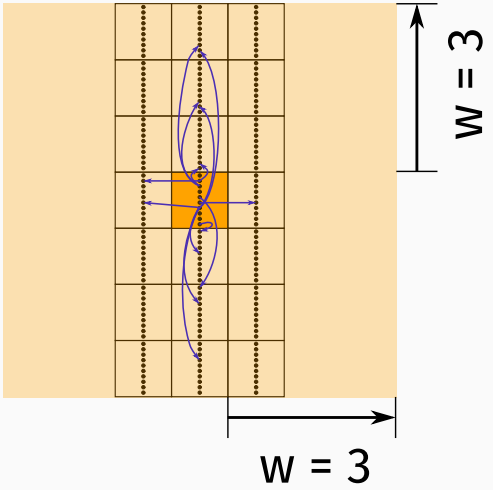
$(2w+1)^2 N$

$((2w+1)^2 N)^2$

An Example Target Architecture (FPGA'20)



An Example Target Architecture (FPGA'20)



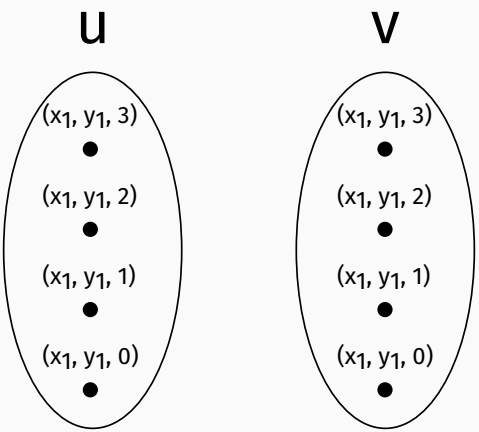
Naive ILP: Encoding Efficiency

$$t_{u,v} = \sum_{p_u \in P(u,w), p_v \in P(v,w)} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

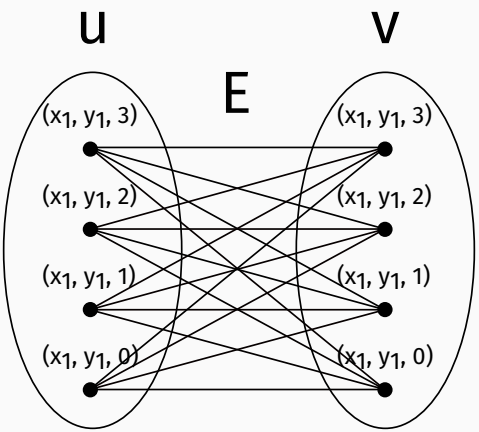
490

240,100

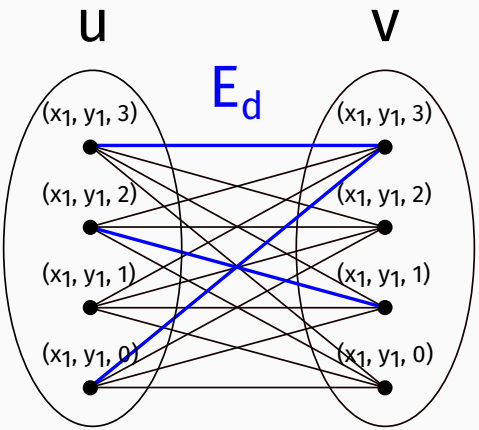
Improved ILP



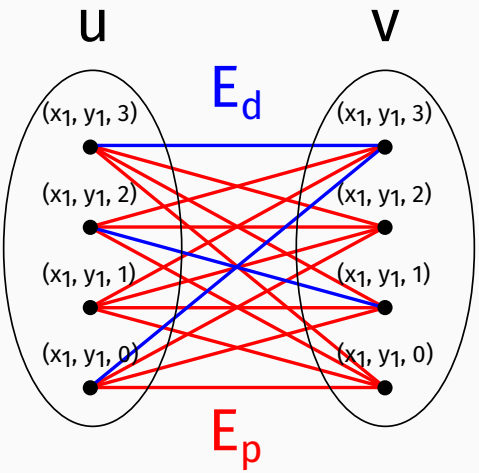
Improved ILP



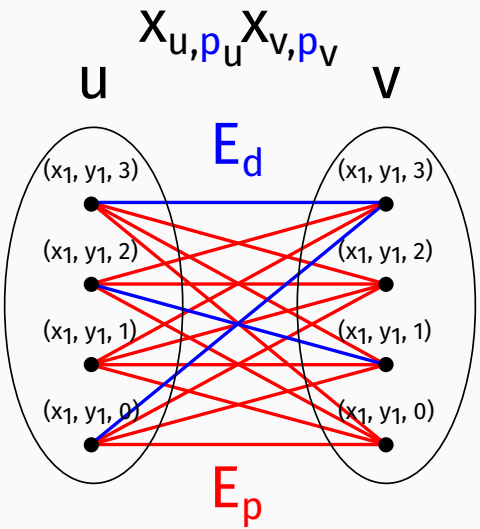
Improved ILP



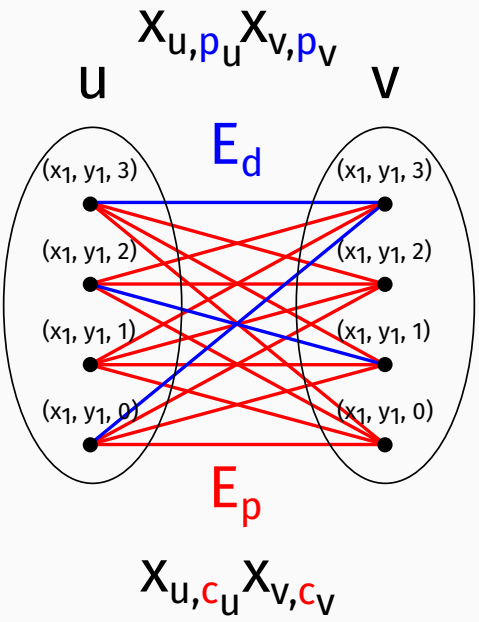
Improved ILP



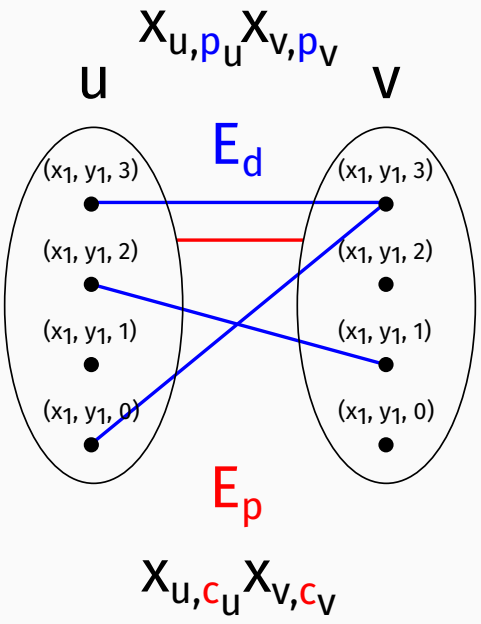
Improved ILP



Improved ILP



Improved ILP



Improved ILP

$$t_{u,v} = \sum_E \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

Improved ILP

$$t_{u,v} = \sum_{E} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

$$t_{u,v} = \sum_{E_d} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

Improved ILP

$$t_{u,v} = \sum_{E} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

$$t_{u,v} = \sum_{E_d} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v} + \sum_{E_p} \tau_{c_u,c_v} x_{u,c_u} x_{v,c_v}$$

Improved ILP

$$\cancel{t_{u,v} = \sum_{E} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}}$$

$$t_{u,v} = y \sum_{E_d} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v} + (1-y) \sum_{E_p} \tau_{c_u,c_v} x_{u,c_u} x_{v,c_v}$$

Improved ILP

$$t_{u,v} = \sum_{E} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

$$t_{u,v} = y \sum_{E_d} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v} + (1-y) \sum_{E_p} \tau_{c_u,c_v} x_{u,c_u} x_{v,c_v}$$

$((2w+1)^2 N)^2$

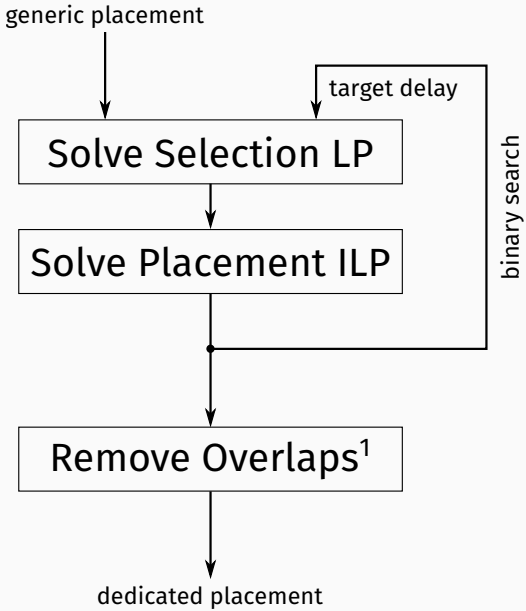
Improved ILP

$$t_{u,v} = \sum_{E} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v}$$

$$t_{u,v} = y \sum_{E_d} \tau_{p_u,p_v} x_{u,p_u} x_{v,p_v} + (1-y) \sum_{E_p} \tau_{c_u,c_v} x_{u,c_u} x_{v,c_v}$$

$((2w+1)^2 M)^2$

Complete Flow



[1]Darav et al., “Multi-commodity flow-based spreading in a commercial analytic placer”, *FPGA’19*

Experimental Setup

Almost the same as FPGA'20

Almost the same as FPGA'20

- Architecture: best found in FPGA'20
  - 14 direct connections, all crossing clusters
  - 10 6-LUT cluster
  - 40 inputs
  - Complete crossbar
  - No carry chains

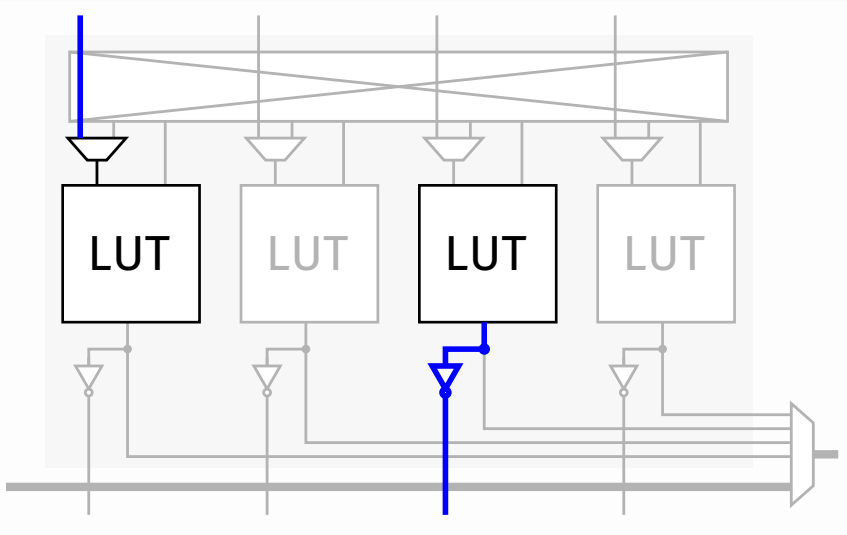
Almost the same as FPGA'20

- Architecture: best found in FPGA'20
  - 14 direct connections, all crossing clusters
  - 10 6-LUT cluster
  - ~~40~~ 60 inputs
  - Complete crossbar
  - No carry chains

Almost the same as FPGA'20

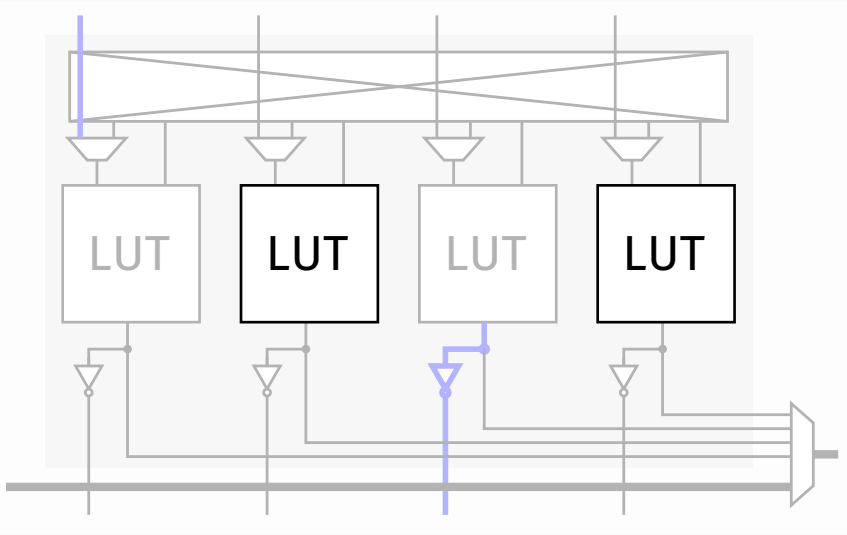
- Architecture: best found in FPGA'20
  - 14 direct connections, all crossing clusters
  - 10 6-LUT cluster
  - ~~40~~ 60 inputs
  - Complete crossbar
  - No carry chains
- VTR 7, with Rubin and DeHon's *delay targeted routing*

Route-time LUT Permutation



Fixed

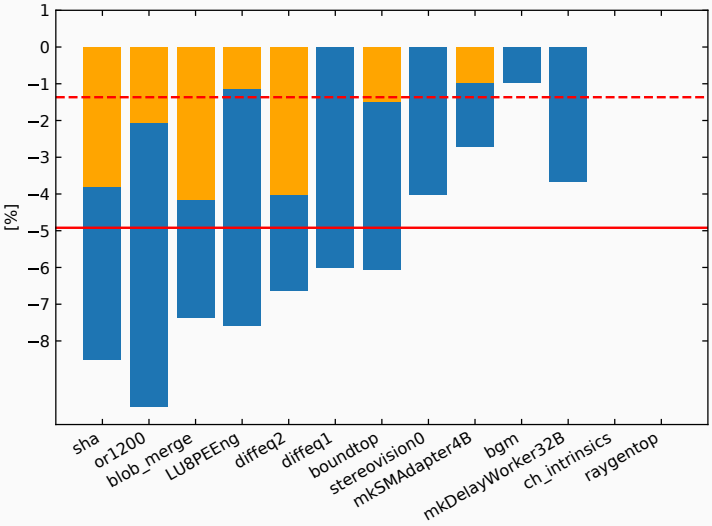
Route-time LUT Permutation



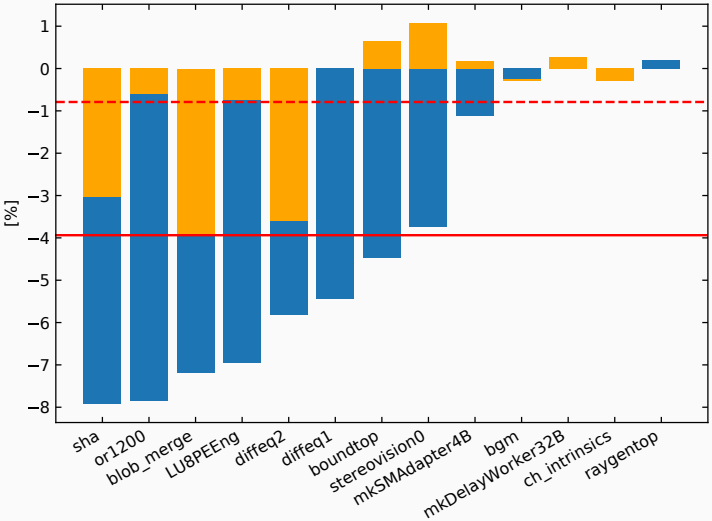
Permutable

Results

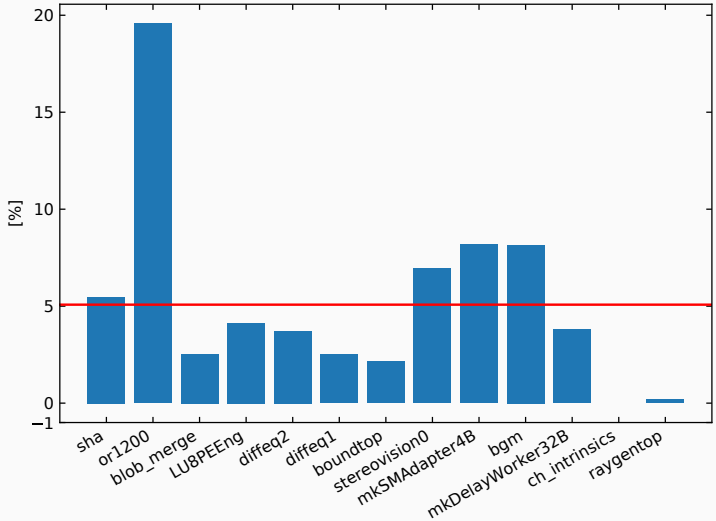
$w = 1$ vs $w = 0$ Delay Change over Baseline: Postplacement

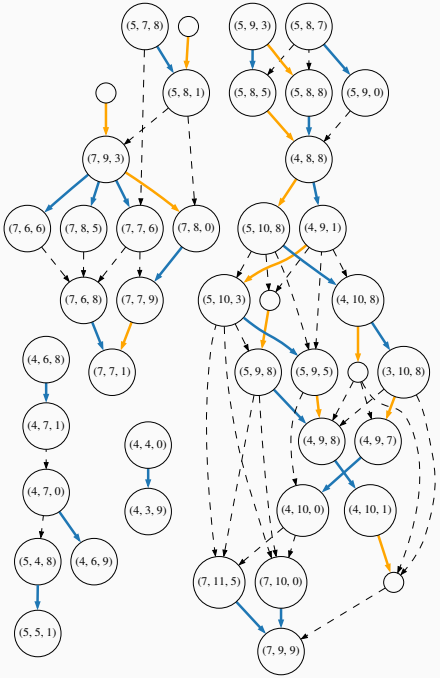


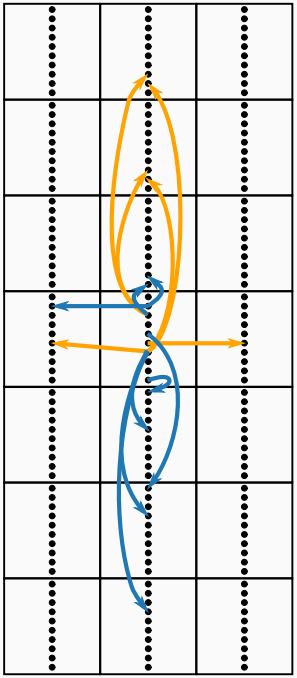
$w = 1$ vs $w = 0$ Delay Change over Baseline: Postrouting



$w = 1$ Delay Change over Baseline, all Programmable




(5, 7, 8)
(5, 8, 1)
(7, 9, 3)
(7, 6, 6)
(7, 8, 5)
(7, 7, 6)
(7, 8, 0)
(7, 6, 8)
(7, 7, 9)
(4, 6, 8)
(7, 7, 1)
(4, 7, 1)
(4, 7, 0)
(5, 4, 8)
(4, 6, 9)
(5, 5, 1)
(4, 4, 0)
(4, 3, 9)
(5, 9, 3)
(5, 8, 7)
(5, 8, 5)
(5, 8, 8)
(5, 9, 0)
(4, 8, 8)
(5, 10, 8)
(4, 9, 1)
(5, 10, 3)
(4, 10, 8)
(5, 9, 8)
(5, 9, 5)
(3, 10, 8)
(4, 9, 8)
(4, 9, 7)
(4, 10, 0)
(4, 10, 1)
(7, 11, 5)
(7, 10, 0)
(7, 9, 9)



Conclusions

We now have an effective dedicated placer for architectures with direct connections

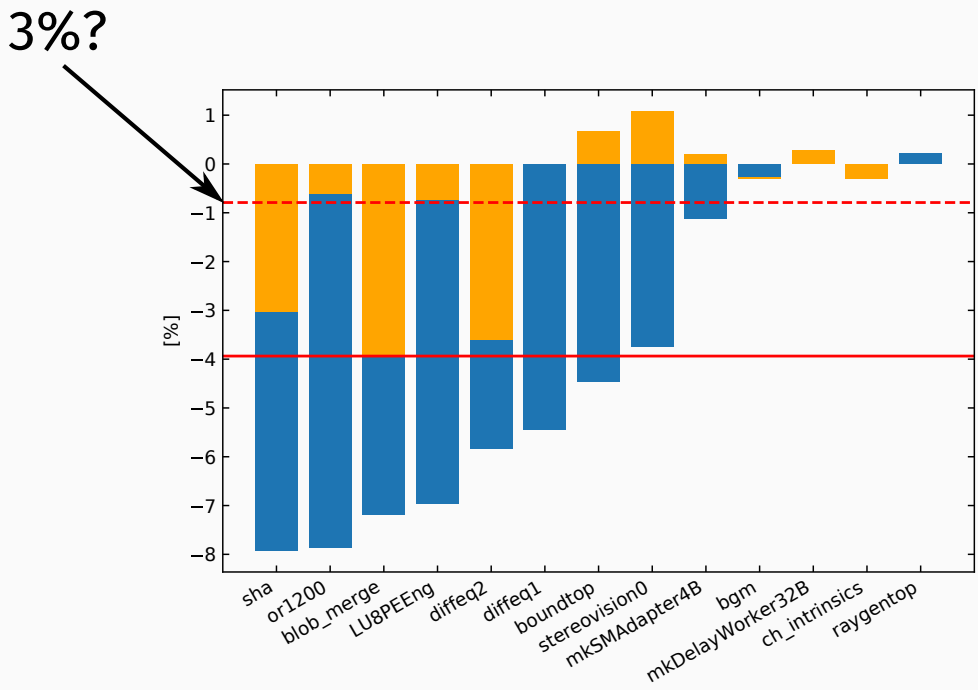
Future Work

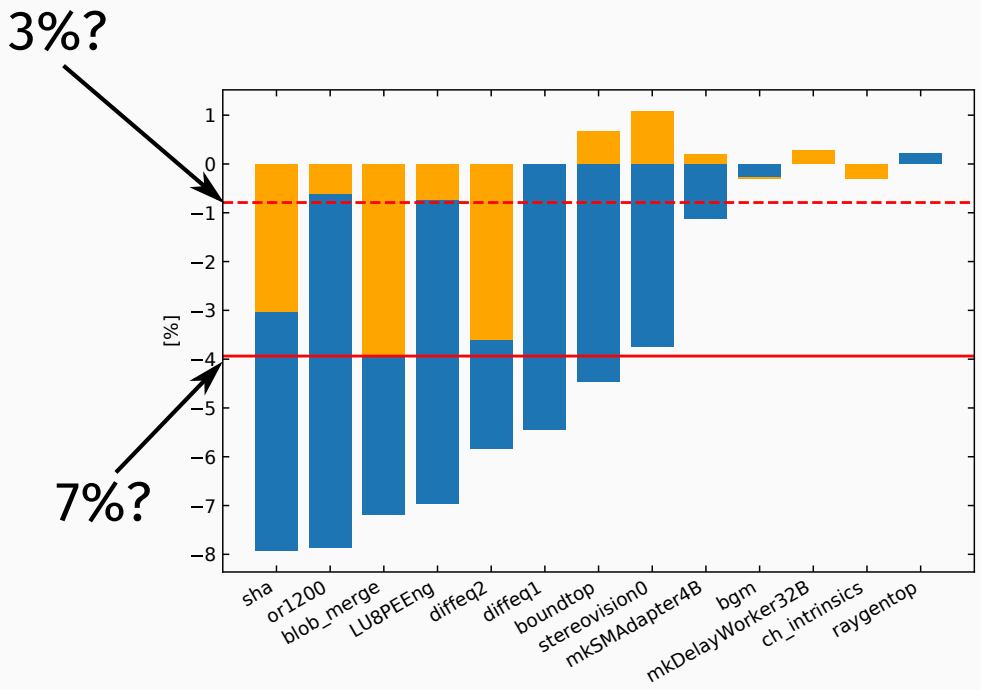
Address scalability issues to extend movement freedom

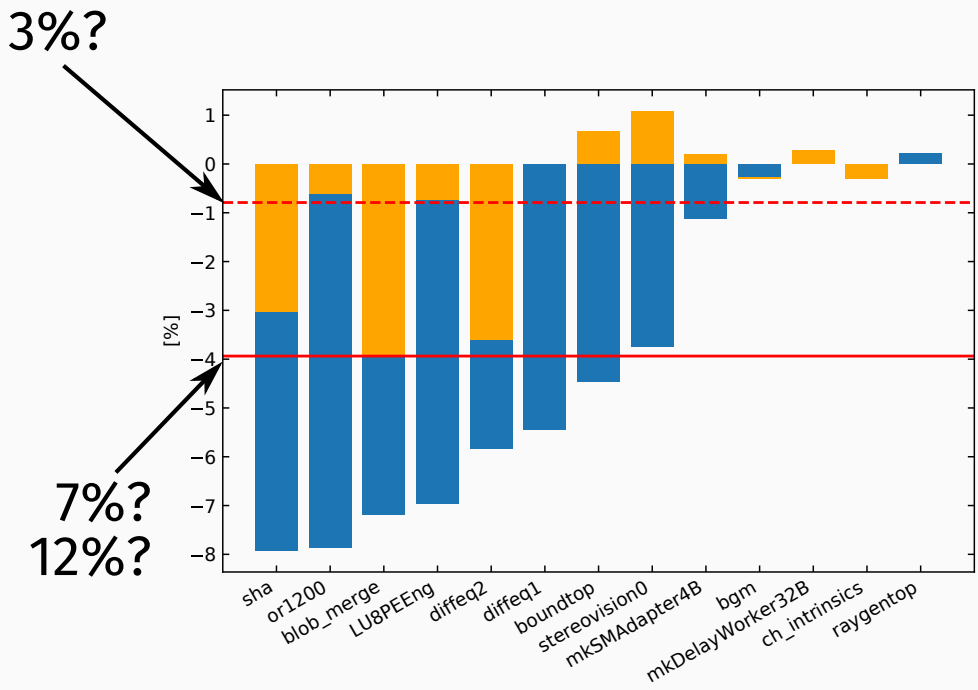
Future Work

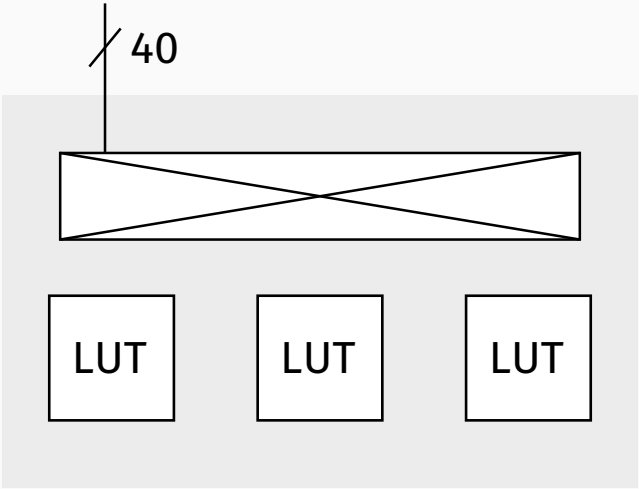
Address scalability issues to extend movement freedom

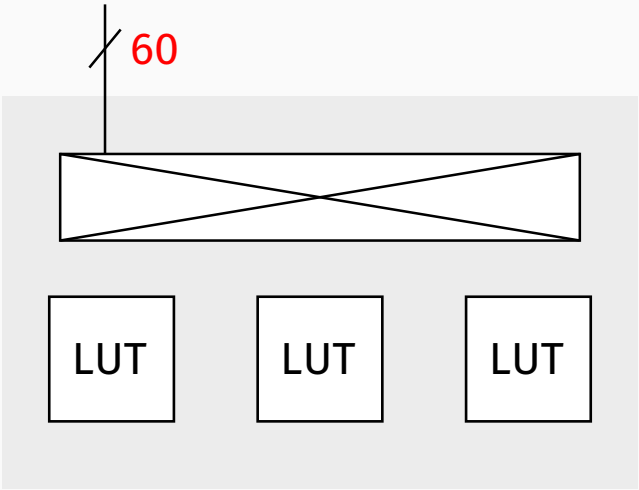
Or allocate the existing freedom more wisely


3%?
1
0
−1
−2
−3
−4
−5
−6
−7
−8
[%]
sha
or1200
blob_merge
LU8PEEng
diffeq2
diffeq1
boundtop
stereovision0
mkSMAdapter4B
bgm
mkDelayWorker32B
ch_intrinsics
raygentop

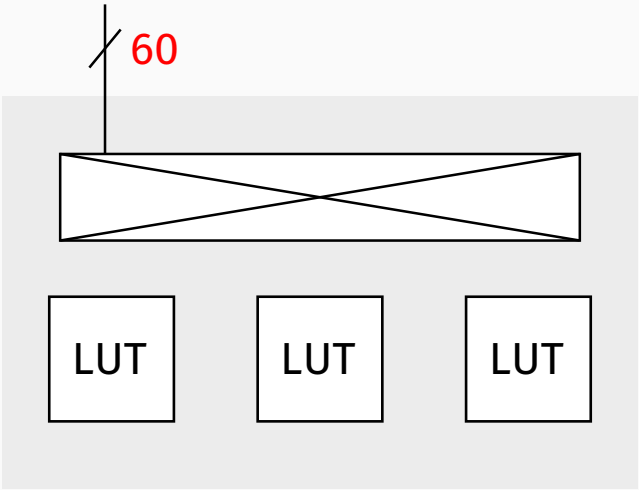

3%?
7%?
[%]
1
0
−1
−2
−3
−4
−5
−6
−7
−8
sha
or1200
blob_merge
LU8PEEng
diffeq2
diffeq1
boundtop
stereovision0
mkSMAdapter4B
bgm
mkDelayWorker32B
ch_intrinsics
raygentop


3%?
7%?
12%?
[%]
1
0
−1
−2
−3
−4
−5
−6
−7
−8
sha
or1200
blob_merge
LU8PEEng
diffeq2
diffeq1
boundtop
stereovision0
mkSMAdapter4B
bgm
mkDelayWorker32B
ch_intrinsics
raygentop


40
LUT
LUT
LUT

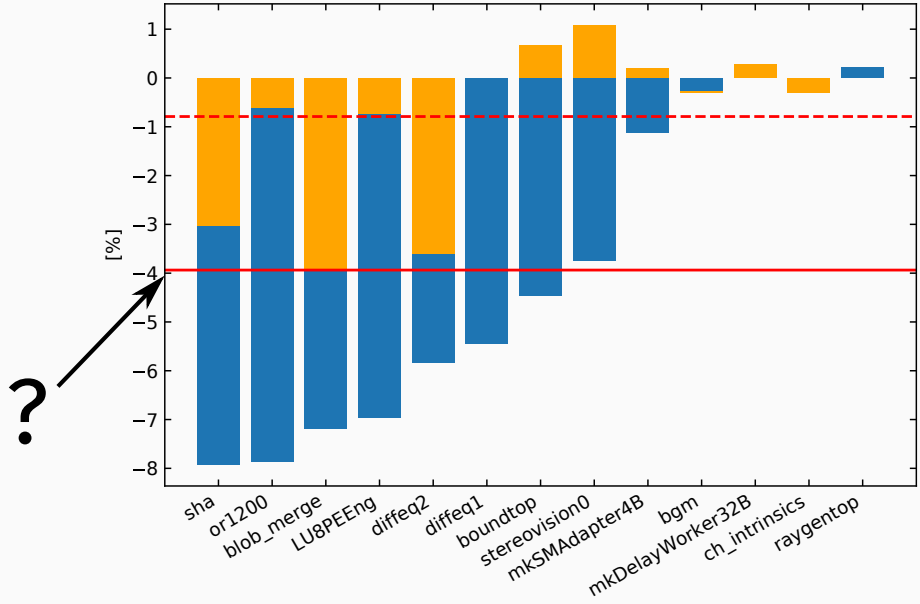


Denser packing



Denser packing + no local direct connections

$\Longrightarrow$ less chance for optimization


1
0
−1
−2
−3
−4
−5
−6
−7
−8
[%]
?
sha
or1200
blob_merge
LU8PEEng
diffeq2
diffeq1
boundtop
stereovision0
mkSMAdapter4B
bgm
mkDelayWorker32B
ch_intrinsics
raygentop

Future Work

Address scalability issues to extend movement freedom

Or allocate the existing freedom more wisely

Extensive architectural exploration

Thank you for attention

https://github.com/stefannikolicns/fpl20-placement