A Computation of the 9th Dedekind Number using FPGA Supercomputing

Lennart Van Hirtum

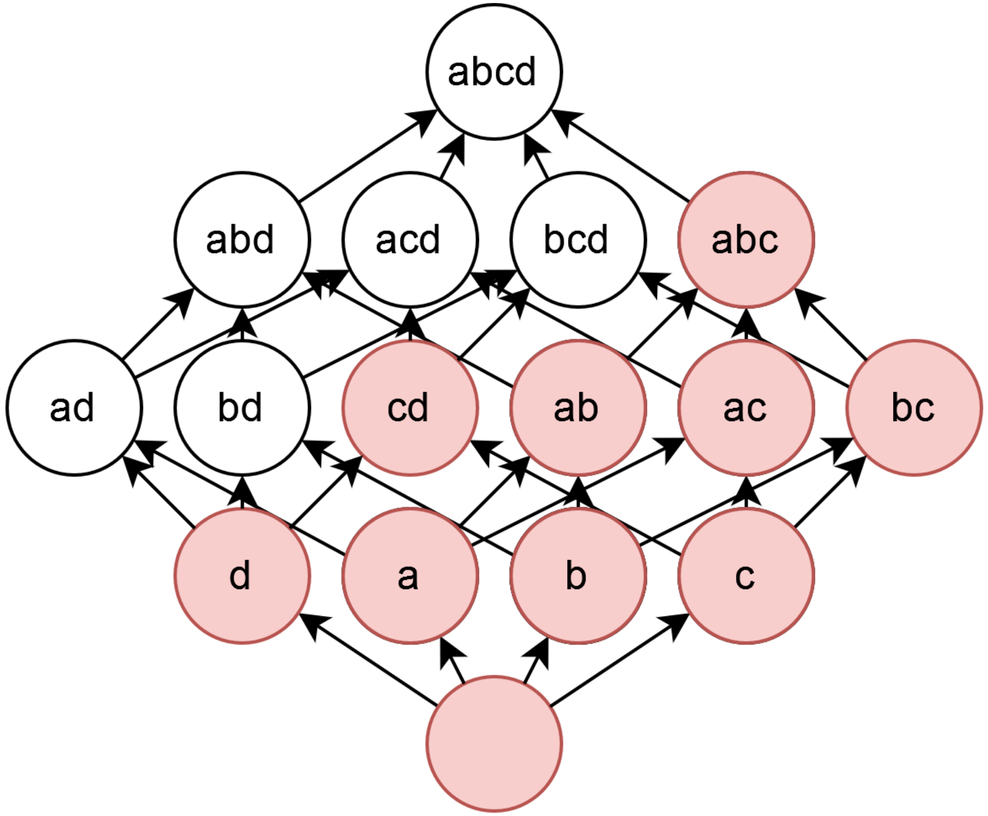
Patrick De Causmaecker, Jens Goemaere, PC2

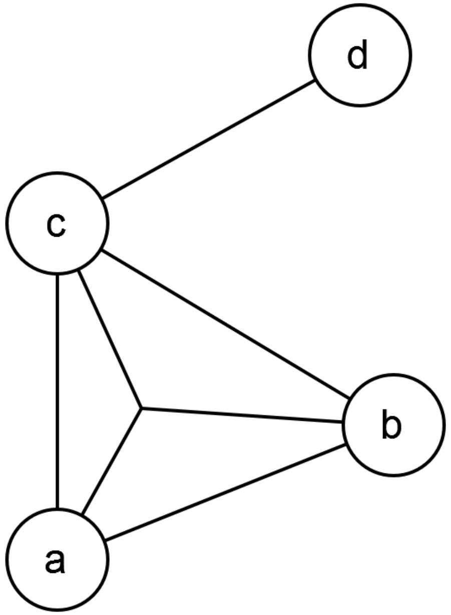
Paderborn University, Germany
Paderborn Center for Parallel Computing

KULAK, 22 May 2023

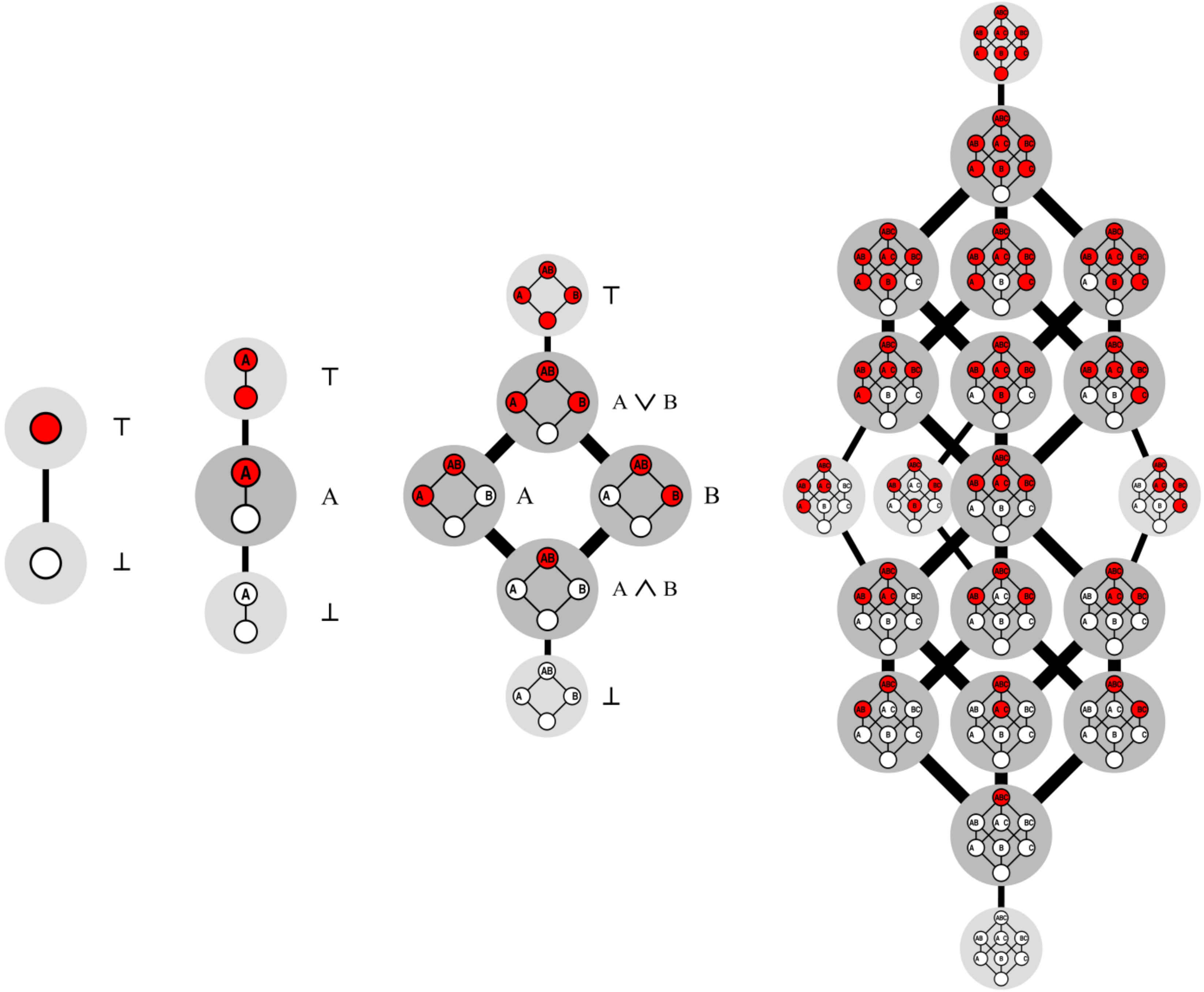
D(0) =	2	Dedekind (1897)
D(1) =	3	Dedekind (1897)
D(2) =	6	Dedekind (1897)
D(3) =	20	Dedekind (1897)
D(4) =	168	Dedekind (1897)
D(5) =	7581	Church (1940)
D(6) =	7828354	Ward (1946)
D(7) =	2414682040998	Church (1965)
D(8) =	56130437228687557907788	Wiedemann (1991)
D(9) =	286386577668298411128469151667598498812366 (2023)	

Monotone Boolean Functions





abcd	bcd	acd	cd	abd	bd	ad	d	abc	bc	ac	c	ab	b	a	

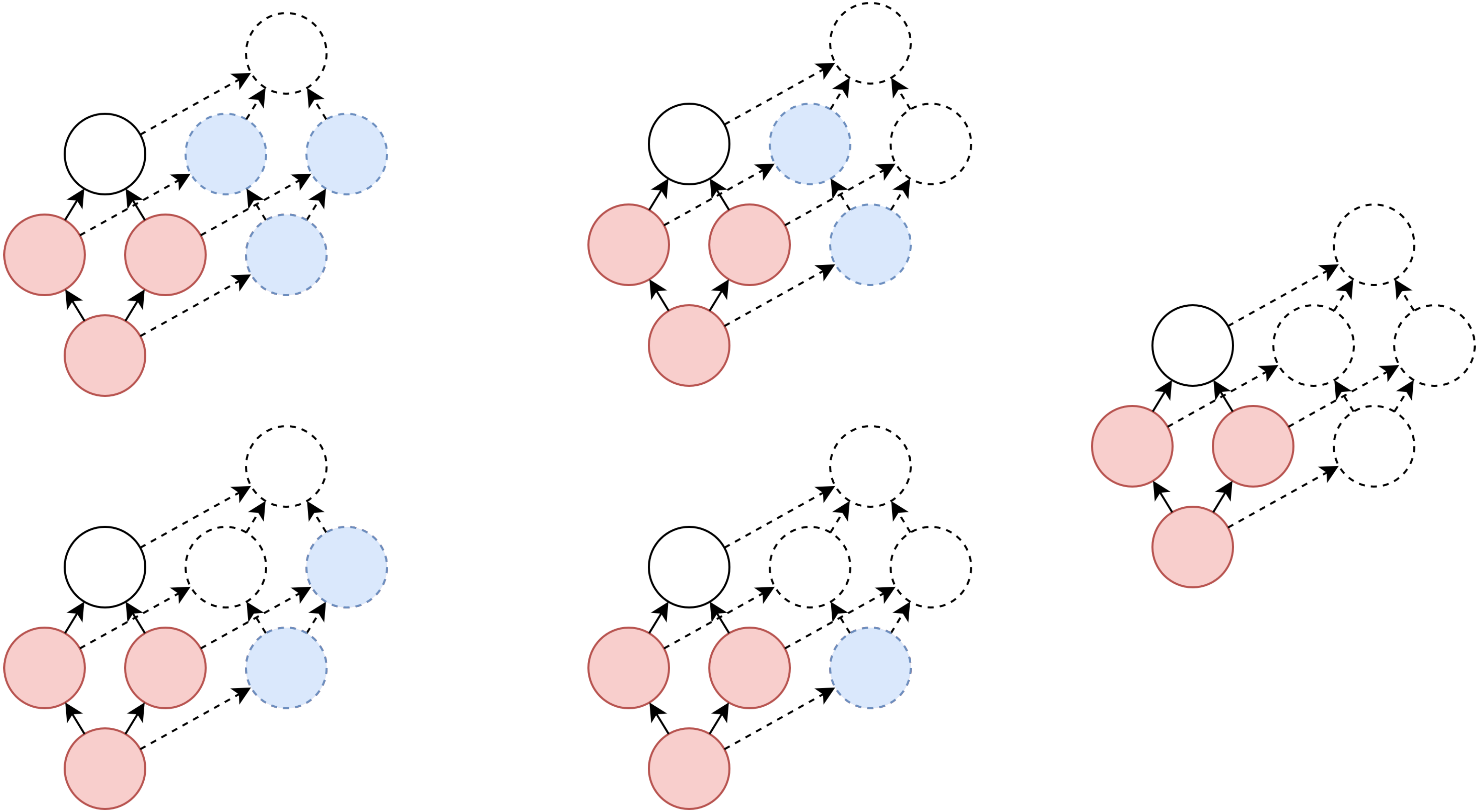

⊤
⊥
⊤
A
⊥
⊤
A ∨ B
A
B
A ∧ B
⊥
A
B
AB
C
AC
BC
ABC

Jumping Formulas

1-Jump

$$D(n+1) = \sum_{\alpha \in A_n} |[\perp, \alpha]|$$

Core Idea



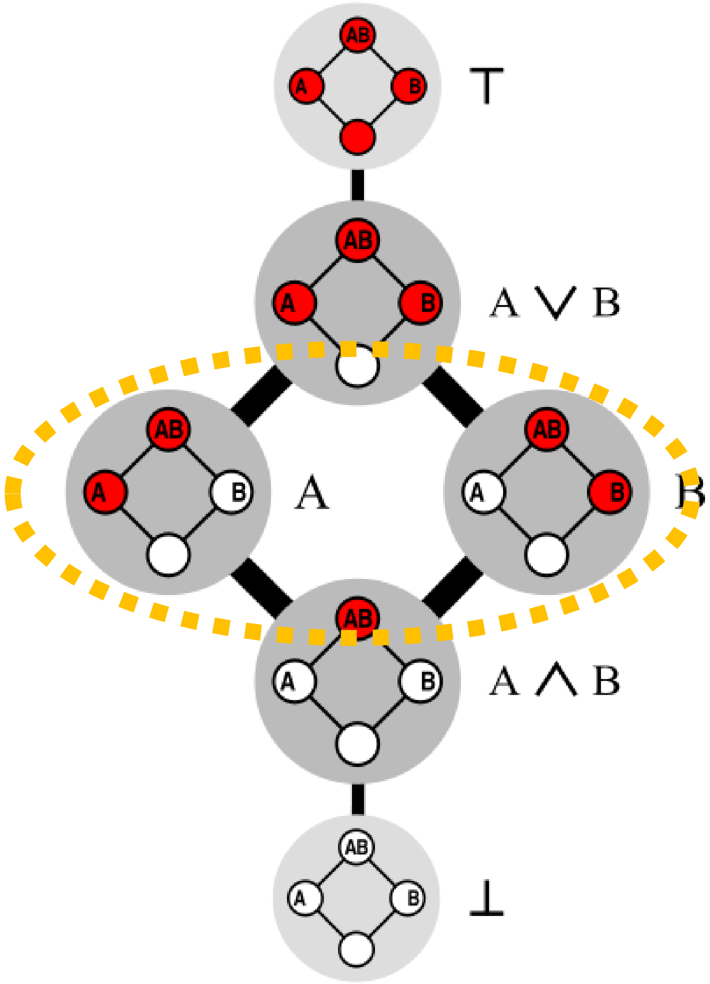
1-Jump

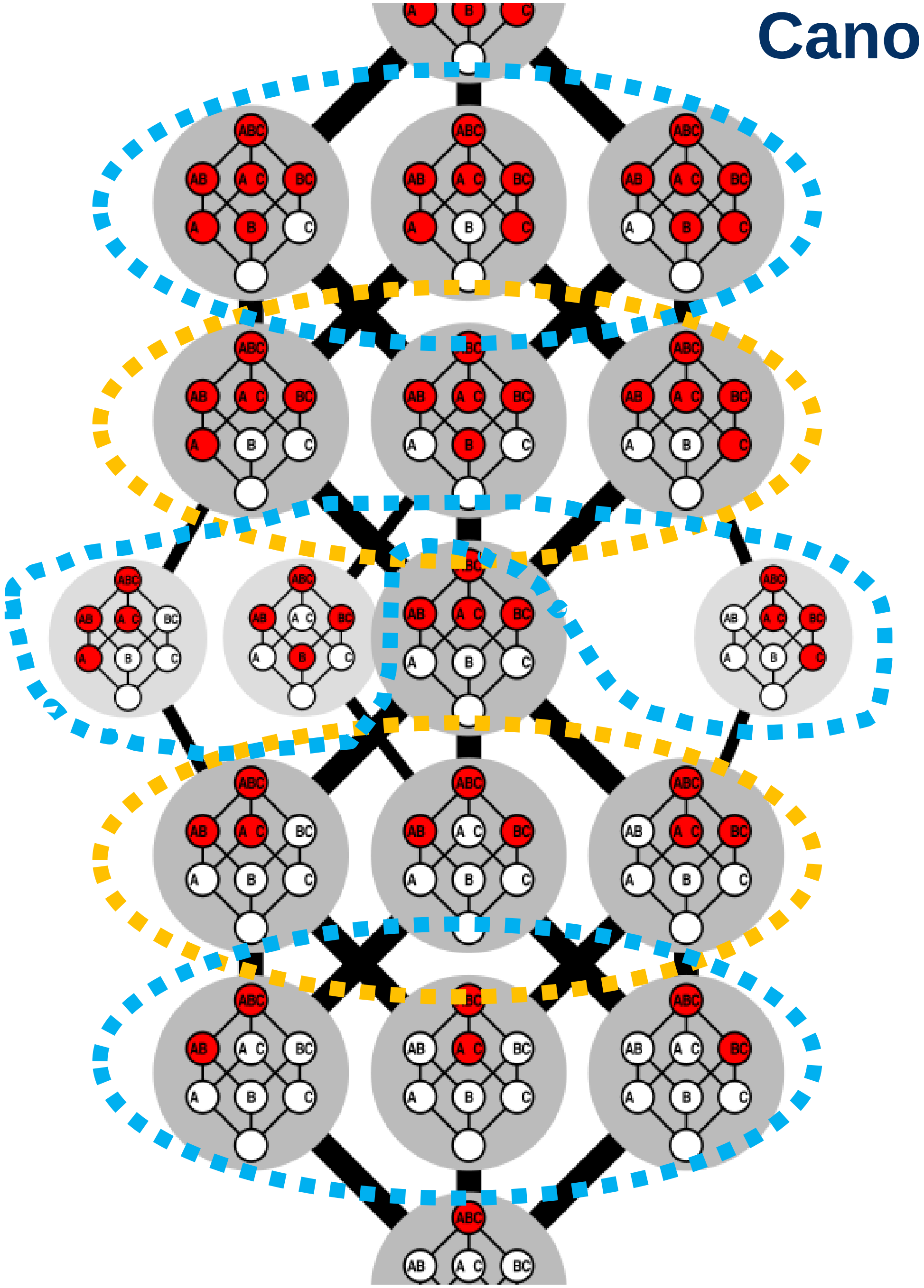
$$D(n+1) = \sum_{\alpha \in A_n} |[\perp, \alpha]|$$

⬇

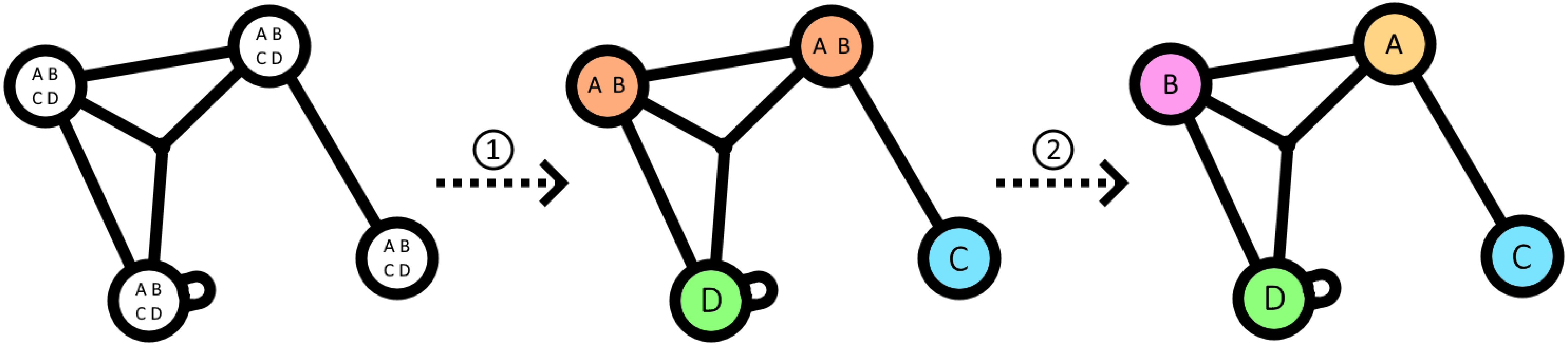
$$D(n+1) = \sum_{\alpha \in R_n} D_\alpha |[\perp, \alpha]|$$

Canonization





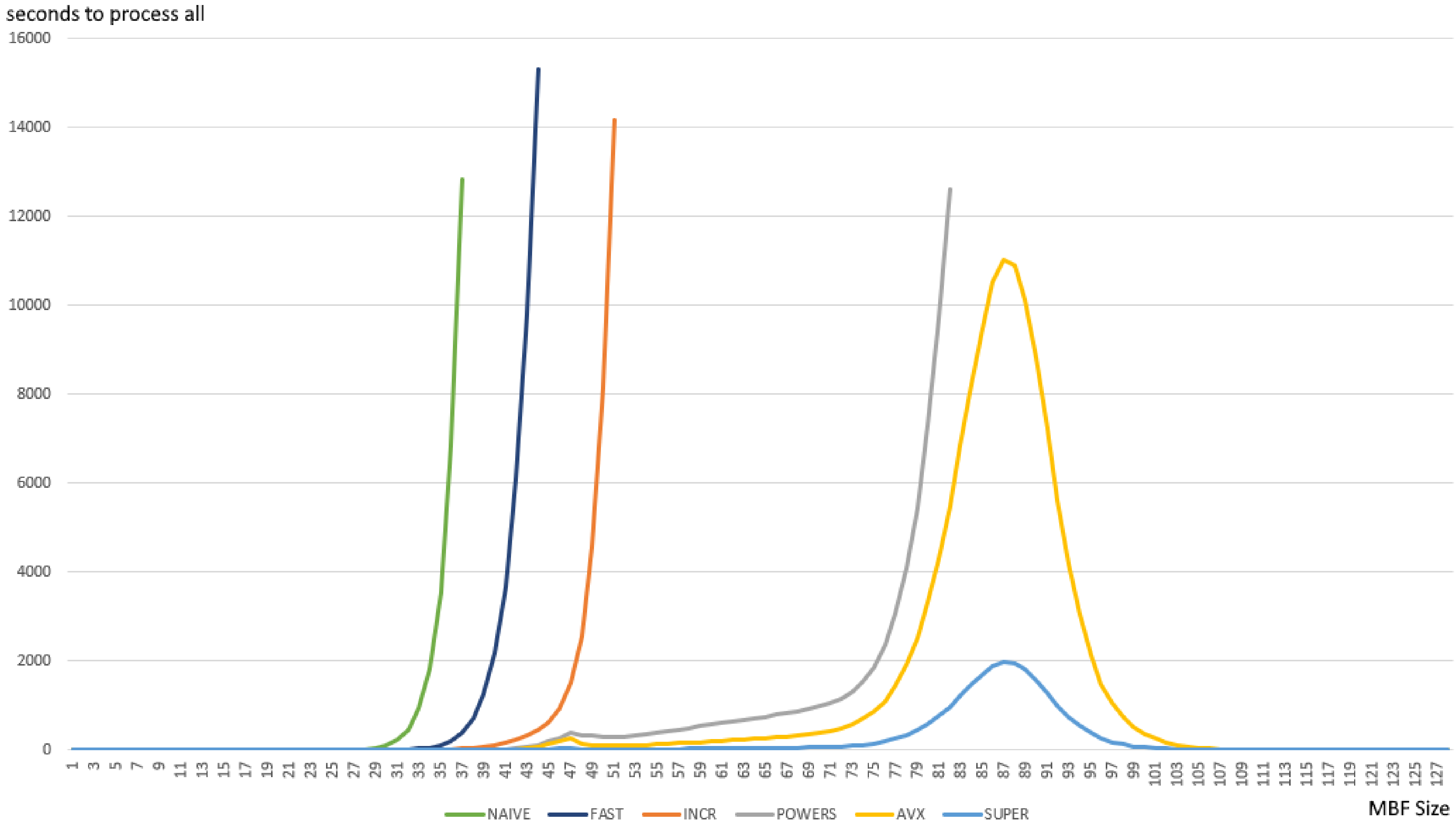
Canonization



R(0) =	2	
R(1) =	3	
R(2) =	5	
R(3) =	10	
R(4) =	30	
R(5) =	210	
R(6) =	16353	
R(7) =	490013148	Yusan (2012)
R(8) =	1392195548889993358	Pawelski (2021)
R(9) =	789204635842035040527740846300252680 (Paw 2023)	

P-Coëfficient Formula

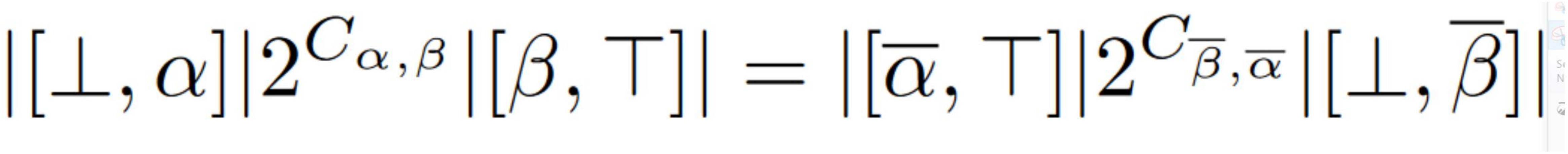
$$D(n+2) = \sum_{\substack{\alpha,\beta\in D_n \\ \alpha\leq\beta}} |[\bot,\alpha]|2^{C_{\alpha,\beta}}|[\beta,\top]|$$


seconds to process all
16000
14000
12000
10000
8000
6000
4000
2000
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85
87
89
91
93
95
97
99
101
103
105
107
109
111
113
115
117
119
121
123
125
127
NAIVE
FAST
INCR
POWERS
AVX
SUPER
MBF Size

Expanded P-Coëfficient Formula

$$D(n+2) = \sum_{\substack{\alpha,\beta\in D_n\\ \alpha\leq\beta}} |[\bot,\alpha]| 2^{C_{\alpha,\beta}} |[\beta,\top]|$$

$$D(n+2) = \sum_{\alpha\in R_n} |[\bot,\alpha]| D_\alpha \sum_{\substack{\beta\in R_n\\ \exists\delta\simeq\beta:\alpha\leq\delta}} |[\beta,\top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma\in Permut_\beta\\ \alpha\leq\gamma}} 2^{C_{\alpha,\gamma}}$$



$$|[\bot, \alpha]|2^{C_{\alpha,\beta}}|[\beta, \top]| = |[\overline{\alpha}, \top]|2^{C_{\overline{\beta},\overline{\alpha}}}|[\bot, \overline{\beta}]|$$

Expanded P-Coëfficient Formula

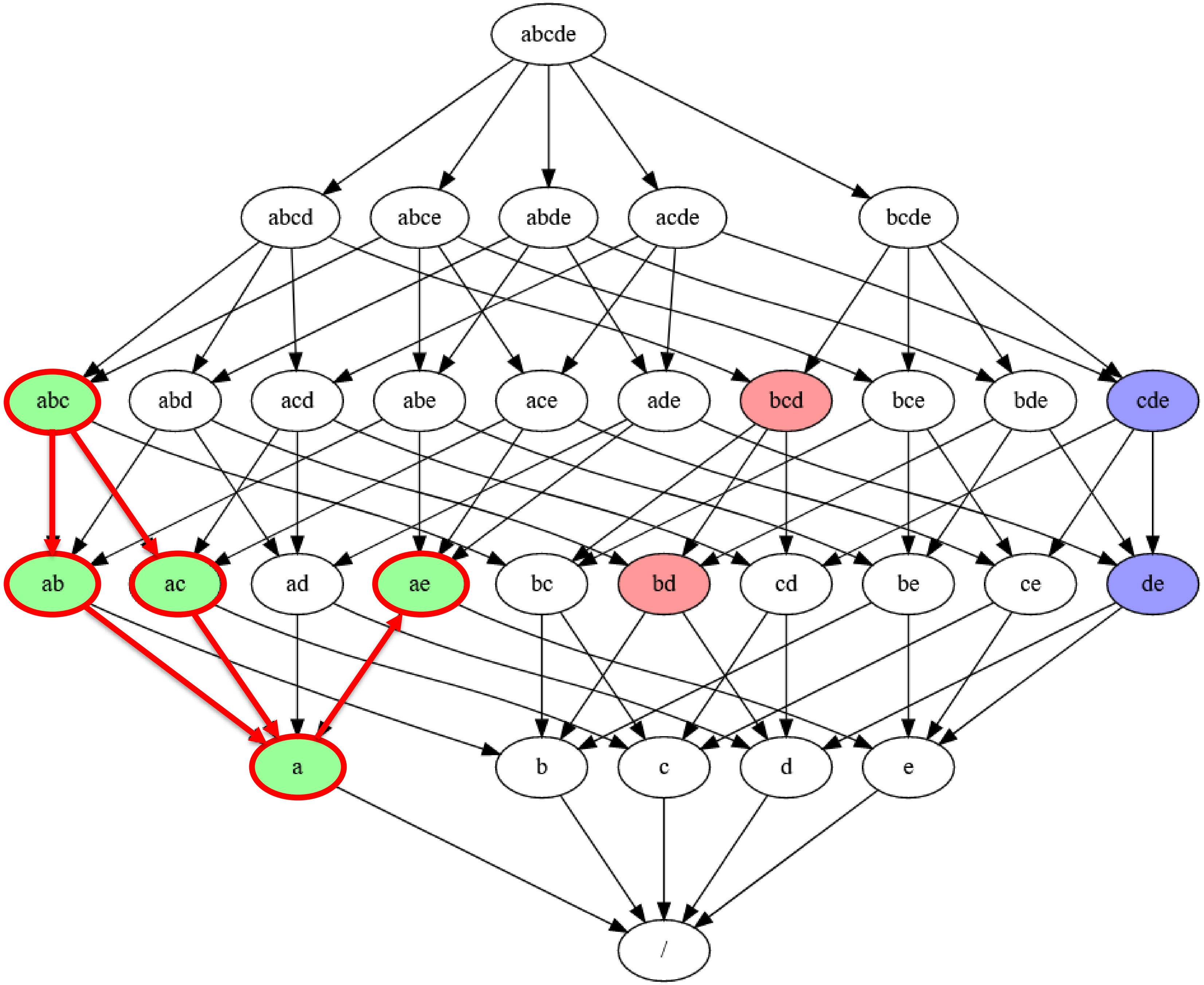
$$D(n+2) = \sum_{\alpha\in R_n} |[\bot,\alpha]| D_\alpha \sum_{\substack{\beta\in R_n \\ \exists\delta\simeq\beta:\alpha\leq\delta}} |[\beta,\top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma\in Permut_\beta \\ \alpha\leq\gamma}} 2^{C_{\alpha,\gamma}}$$

490M *45M *5040

$5.57 * 10^{18}$ $C_{\alpha,\gamma}$ values in total!

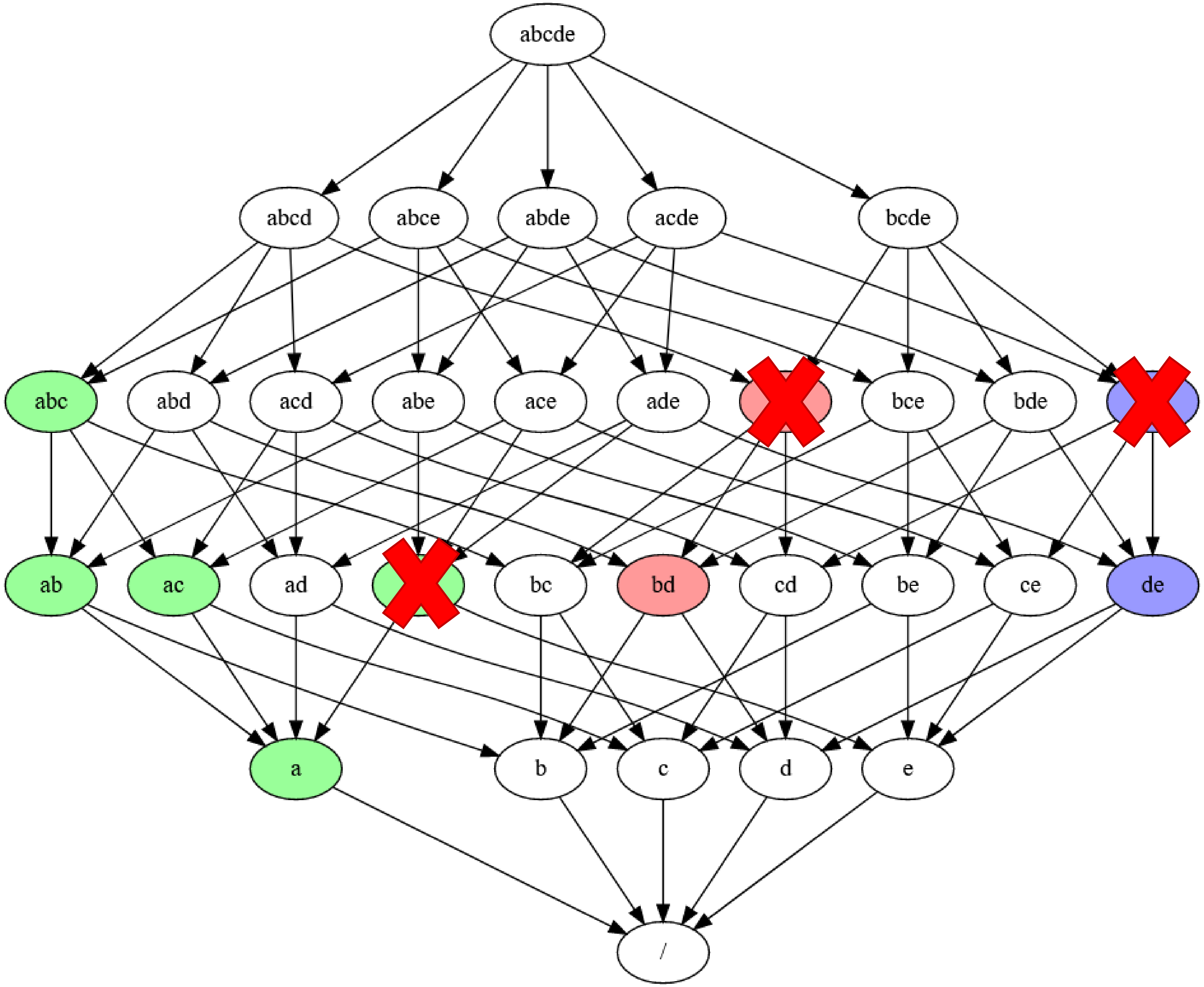
FloodFill Algorithm

FloodFill Algorithm



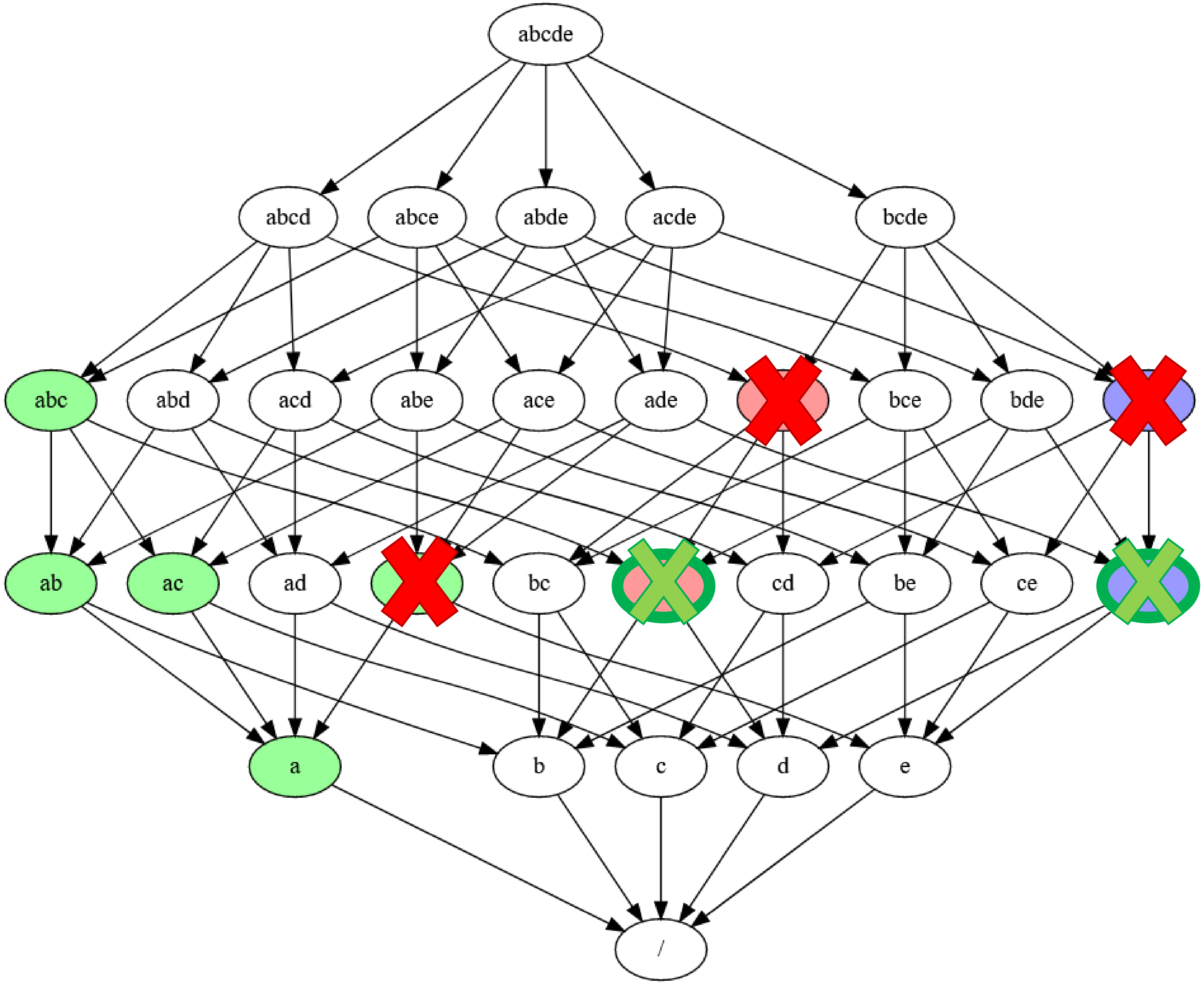
$$C_{\alpha,\beta} = 3$$

Leaf Elimination



- LE Up
- LE Down

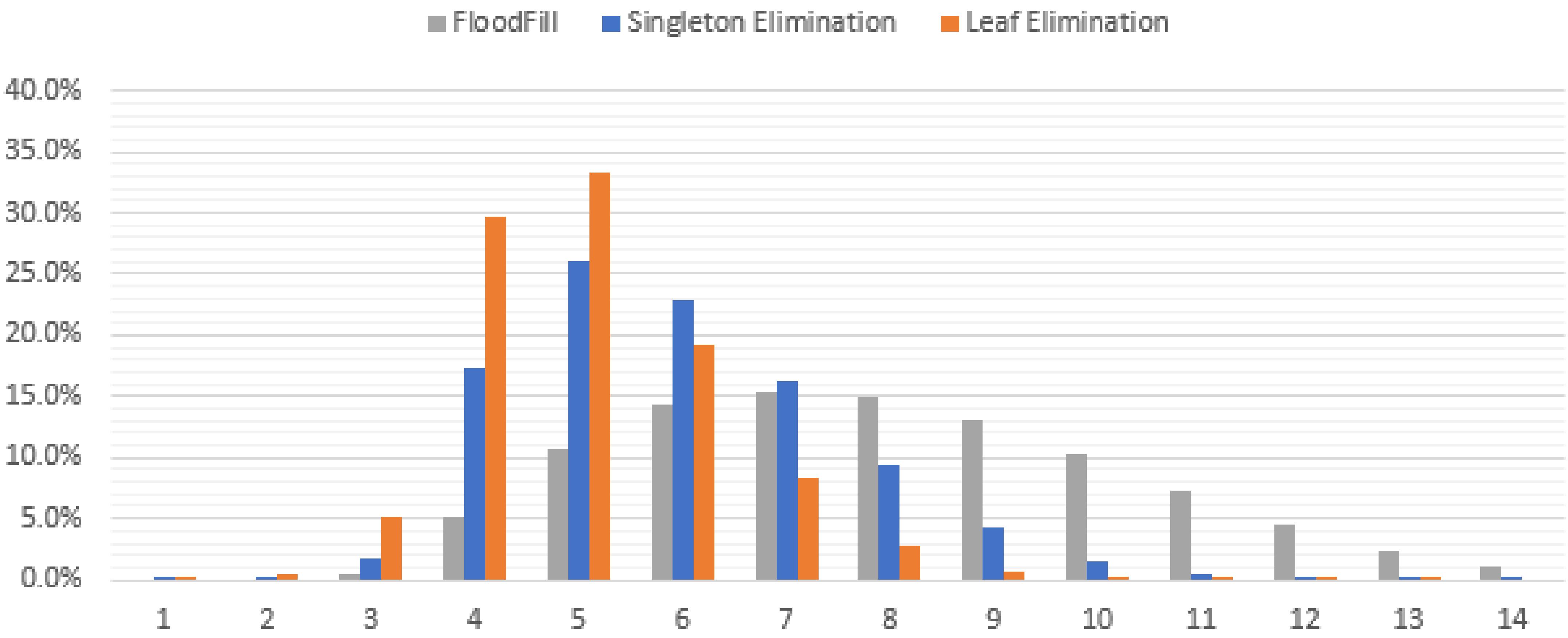
Singleton Elimination



FloodFill Algorithm

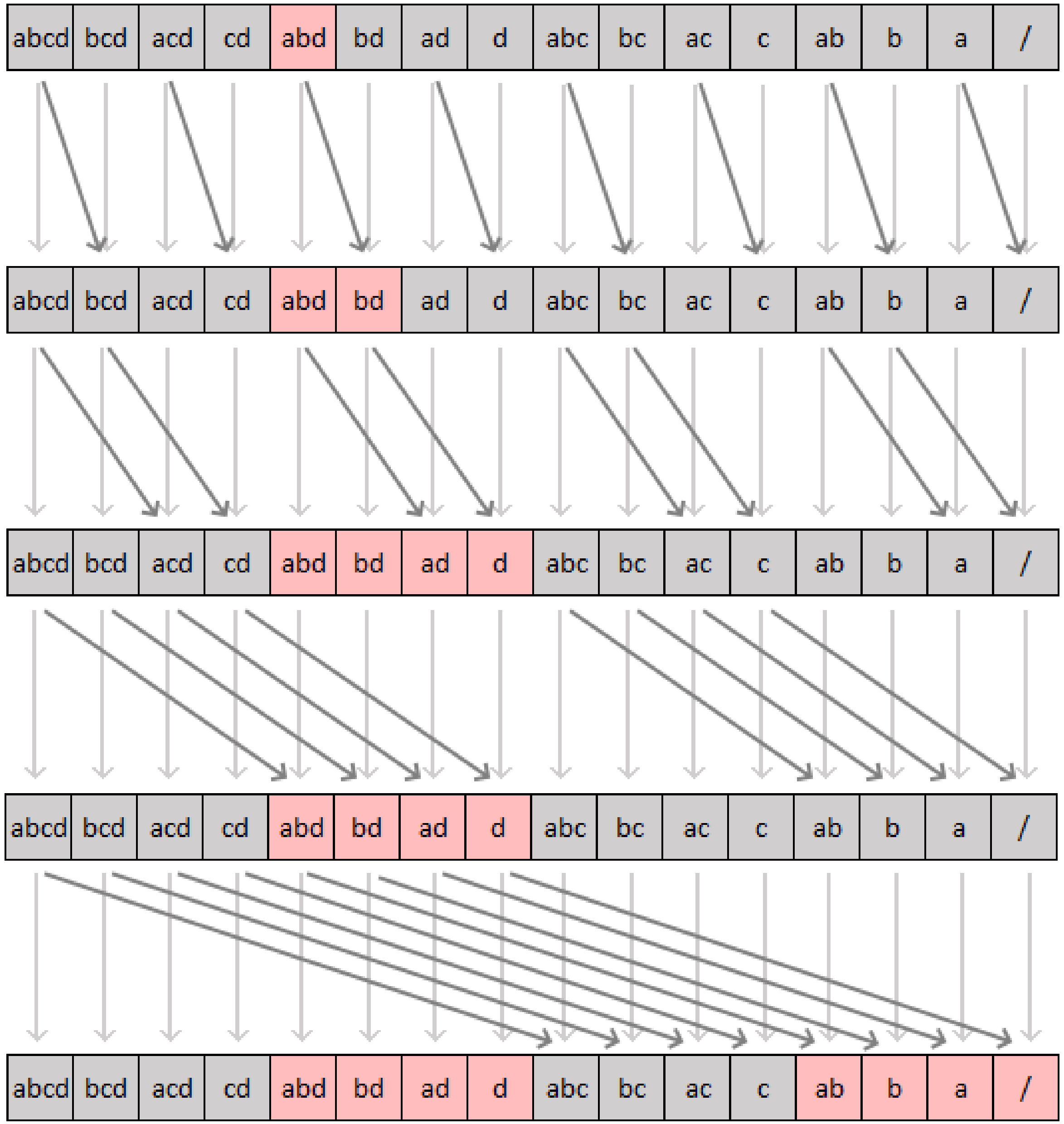
```
def countConnected(MBF $\alpha$, MBF $\gamma$):
    BF graph = $\alpha \& \neg\gamma$
    graph = eliminateLeafesUp(graph)
    graph, int count = eliminateSingletons(graph)
    while graph not empty:
        BF seed = firstNode(graph)
        do:
            BF seedUp = monotonizeUp(seed) & graph
            seed = monotonizeDown(seedUp) & graph
        while seedUp != seed
        graph = graph & ¬ seed
        count++
    return count
```

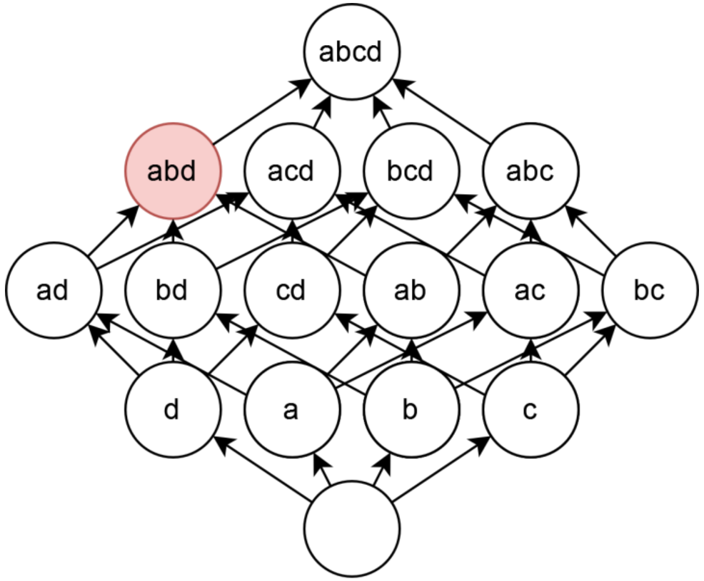
FloodFill Cycles Distribution

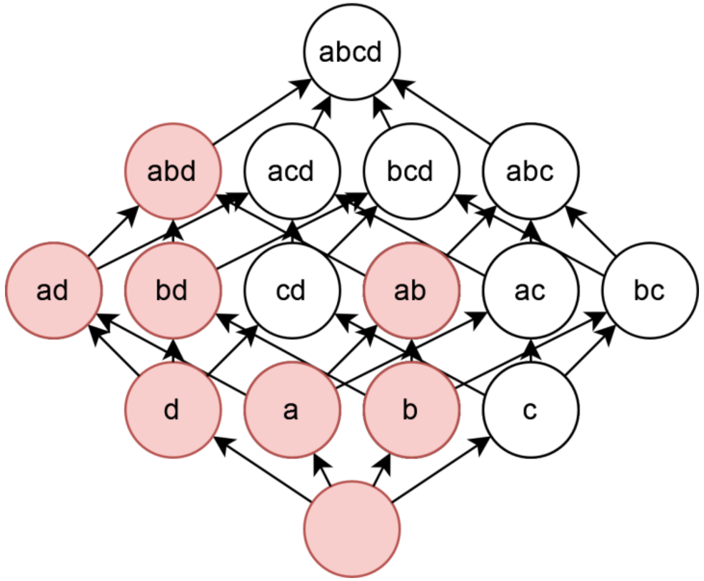


Hardware Implementation

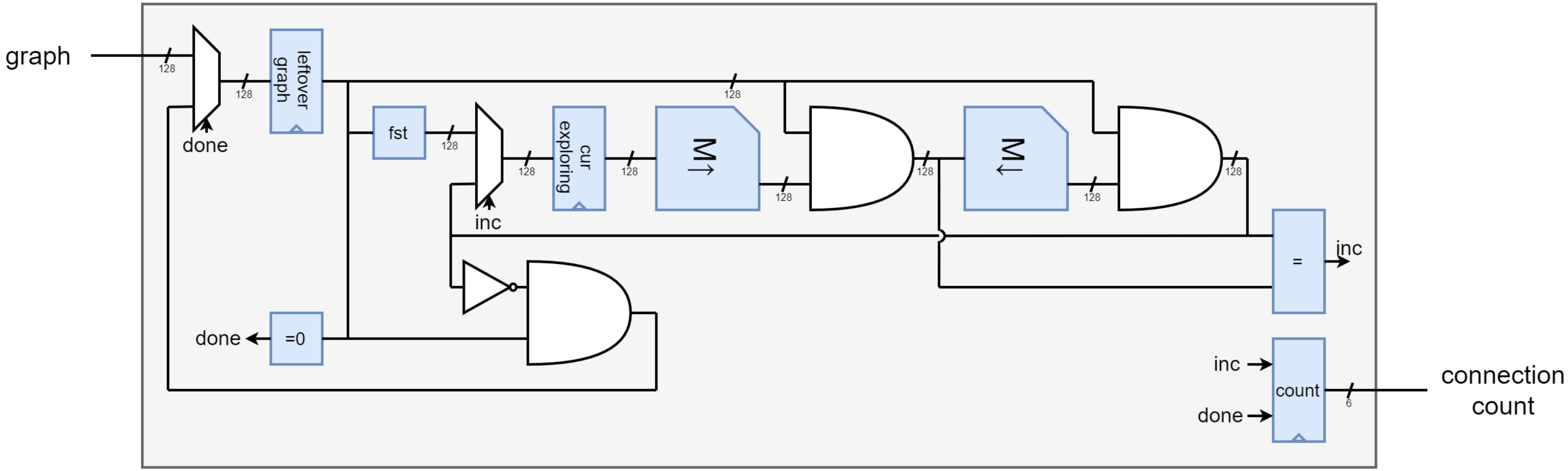
Monotonization


abcd bcd acd cd abd bd ad d abc bc ac c ab b a /
abcd bcd acd cd abd bd ad d abc bc ac c ab b a /
abcd bcd acd cd abd bd ad d abc bc ac c ab b a /
abcd bcd acd cd abd bd ad d abc bc ac c ab b a /
abcd bcd acd cd abd bd ad d abc bc ac c ab b a /

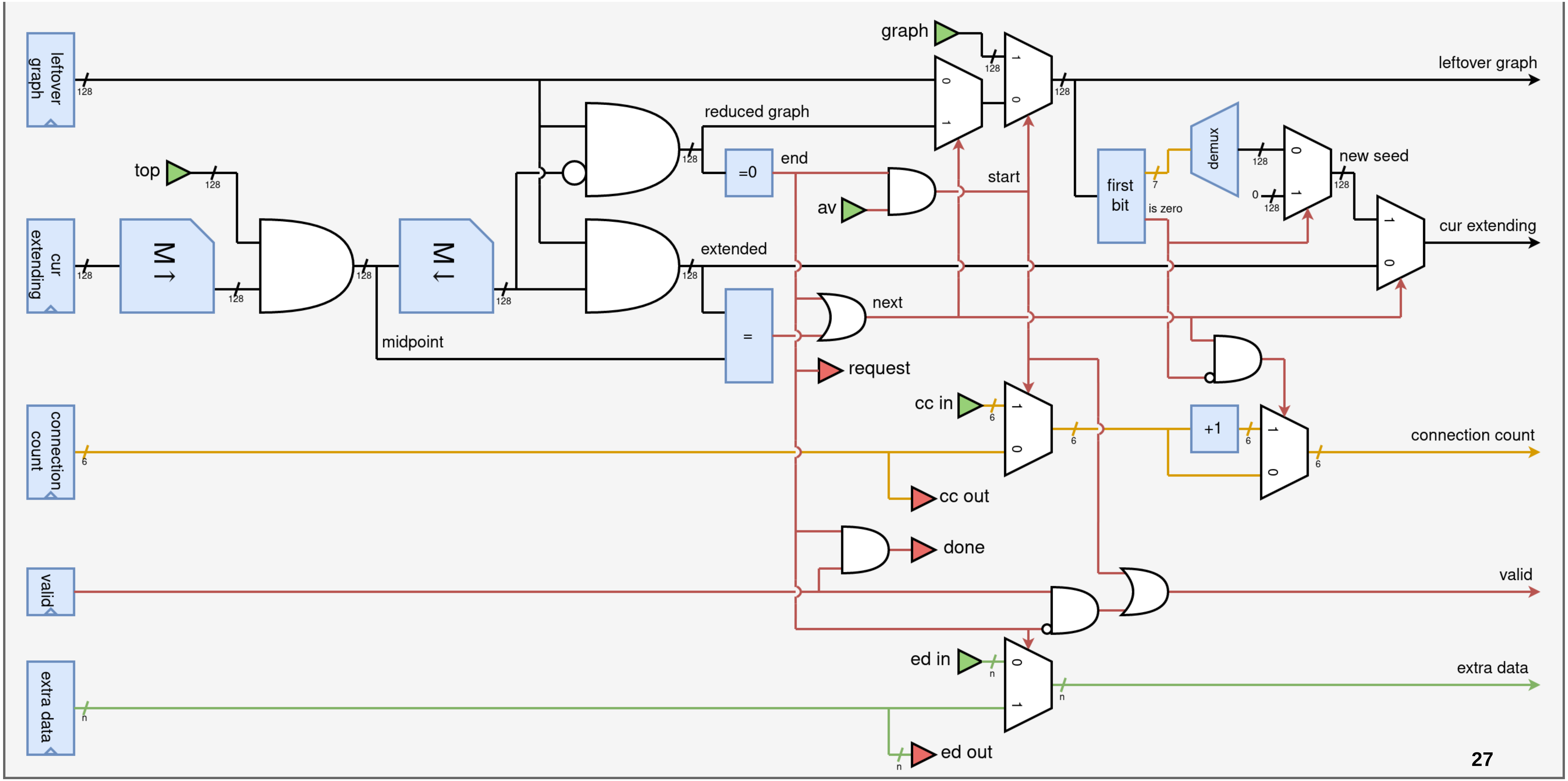




Count Connected Core

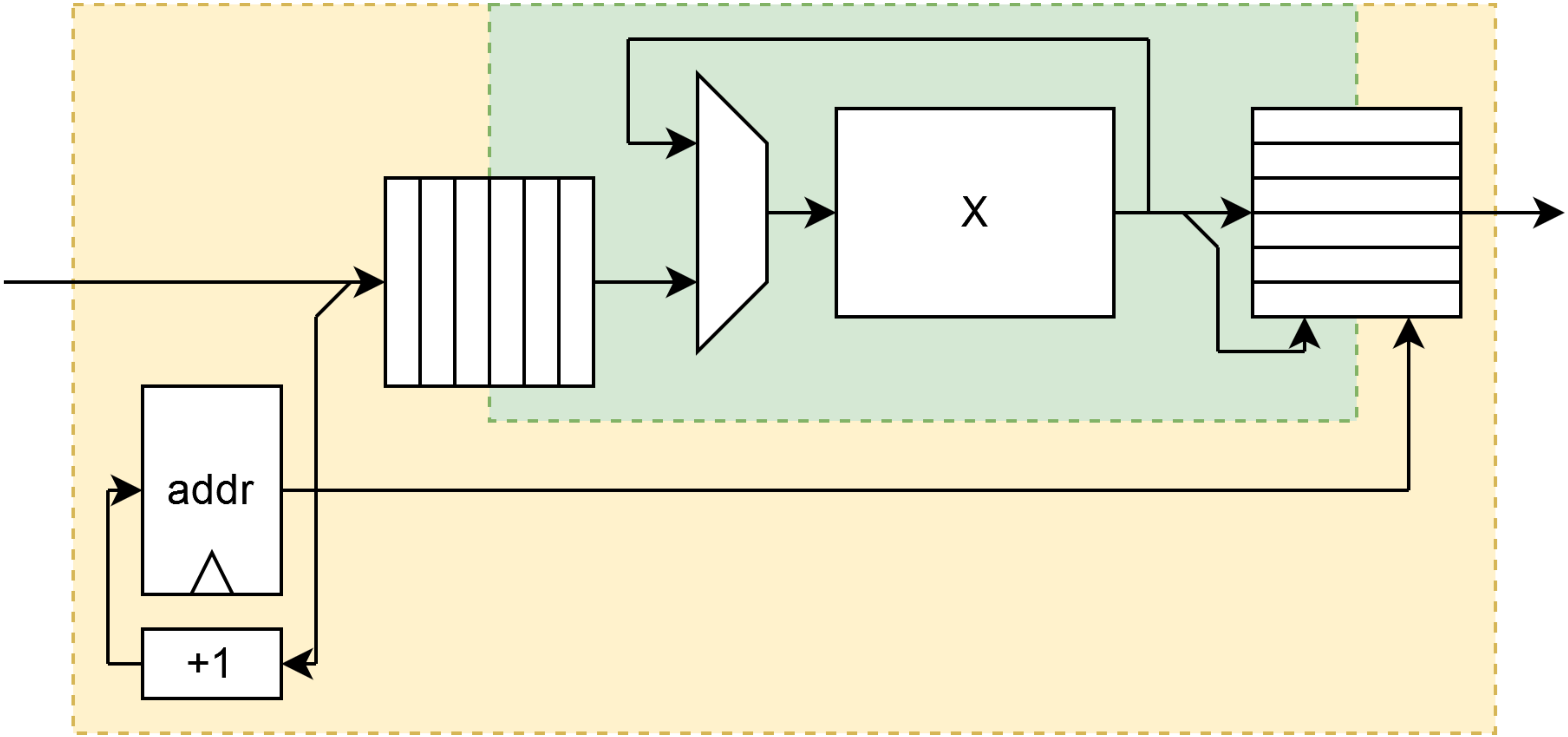


Pipelined

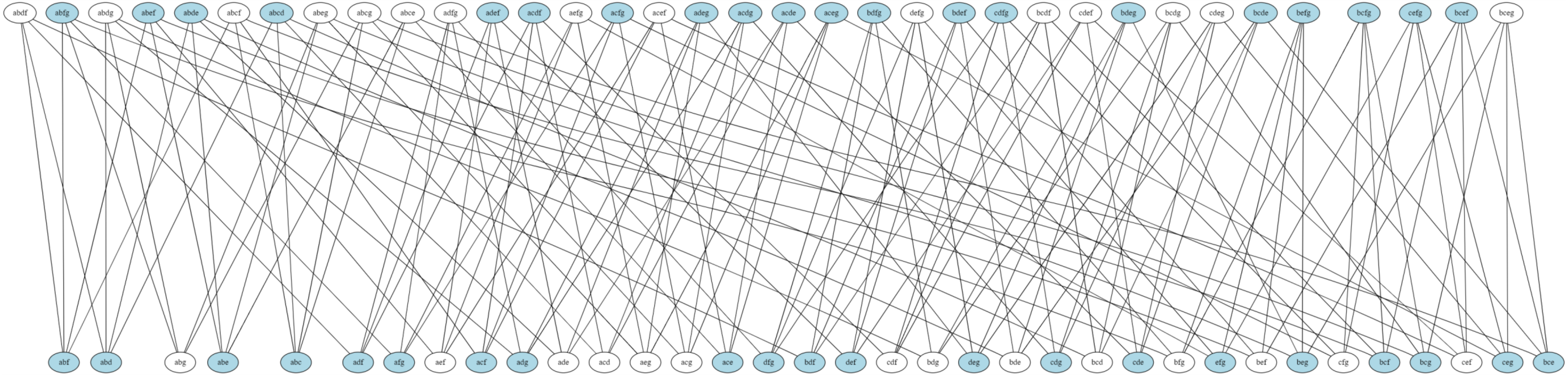


leftover graph
cur extending
connection count
valid
extra data
graph
top
av
cc in
ed in
reduced graph
end
start
extended
next
midpoint
request
cc out
done
ed out
=0
=
M↑
M↓
first bit
is zero
demux
new seed
+1
leftover graph
cur extending
connection count
valid
extra data
128
7
6
n
0
1

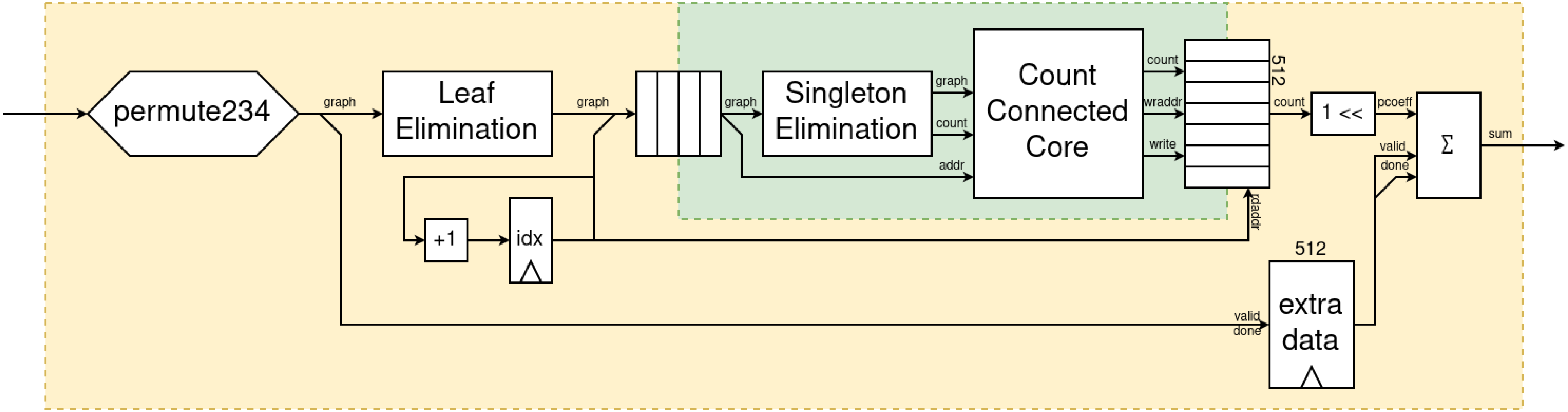
Loop implementation

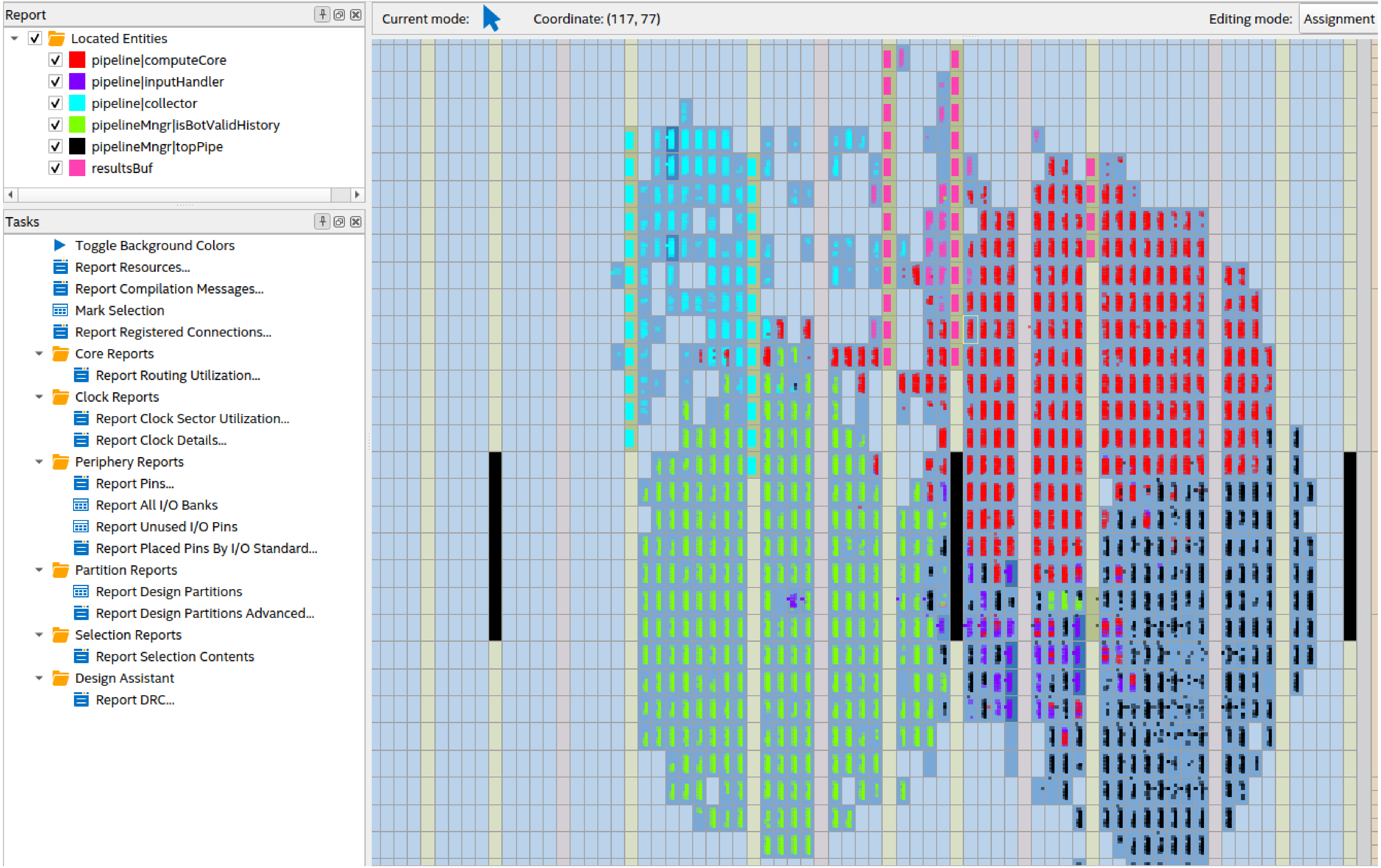


Worst Case


abdf
abfg
abdg
abef
abde
abcf
abcd
abeg
abcg
abce
adfg
adef
acdf
aefg
acfg
acef
adeg
acdg
acde
aceg
bdfg
defg
bdef
cdfg
bcdf
cdef
bdeg
bcdg
cdeg
bcde
befg
bcfg
cefg
bcef
bceg
abf
abd
abg
abe
abc
adf
afg
aef
acf
adg
ade
acd
aeg
acg
ace
dfg
bdf
def
cdf
bdg
deg
bde
cdg
bcd
cde
bfg
efg
bef
beg
cfg
bcf
bcg
cef
ceg
bce

Processing Module





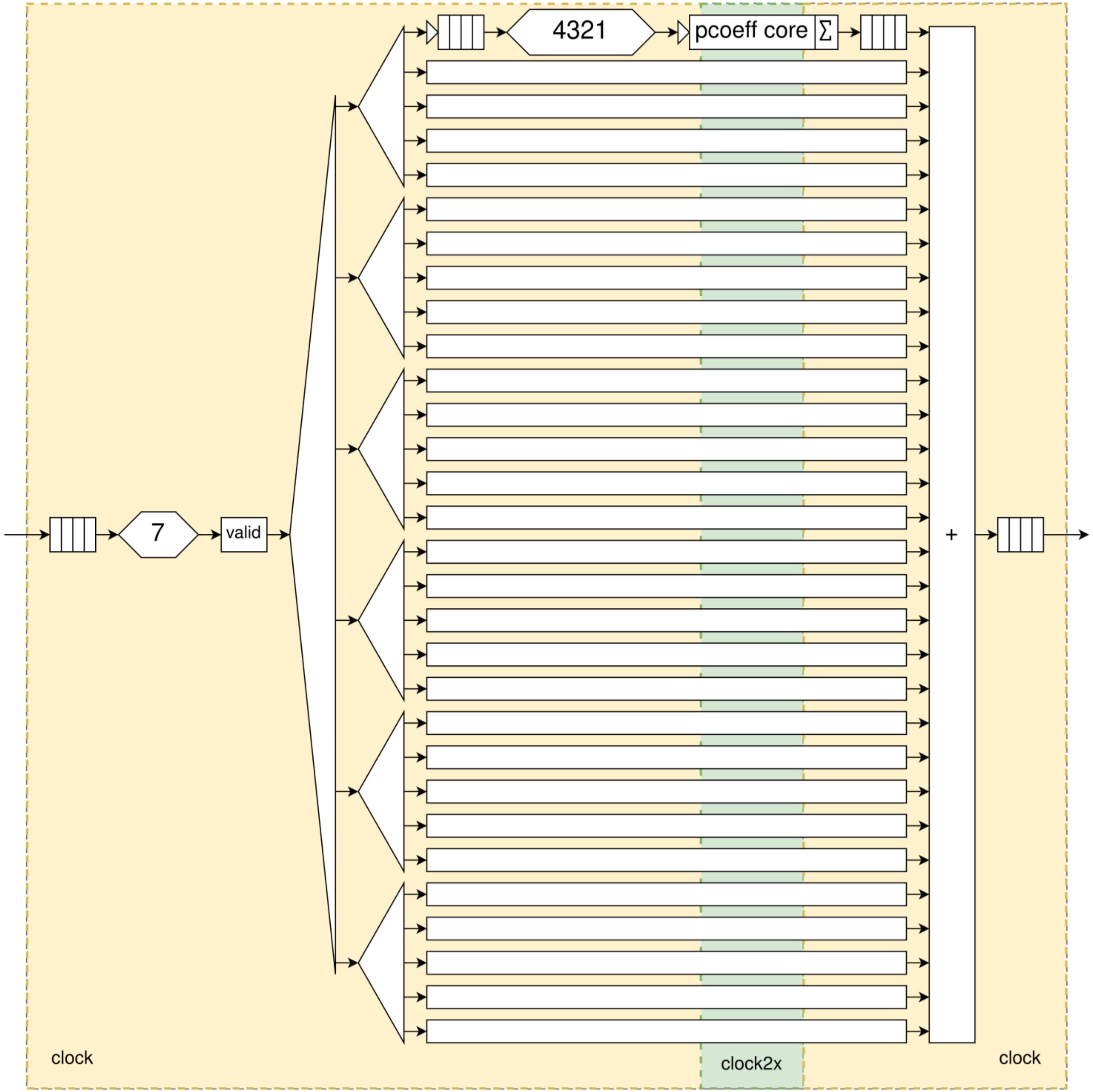
Report

- ☑ Located Entities
  - ☑ 🟥 pipeline|computeCore
  - ☑ 🟪 pipeline|inputHandler
  - ☑ 🟦 pipeline|collector
  - ☑ 🟩 pipelineMngr|isBotValidHistory
  - ☑ ⬛ pipelineMngr|topPipe
  - ☑ 🟥 resultsBuf

Tasks

- Toggle Background Colors
- Report Resources...
- Report Compilation Messages...
- Mark Selection
- Report Registered Connections...
- Core Reports
  - Report Routing Utilization...
- Clock Reports
  - Report Clock Sector Utilization...
  - Report Clock Details...
- Periphery Reports
  - Report Pins...
  - Report All I/O Banks
  - Report Unused I/O Pins
  - Report Placed Pins By I/O Standard...
- Partition Reports
  - Report Design Partitions
  - Report Design Partitions Advanced...
- Selection Reports
  - Report Selection Contents
- Design Assistant
  - Report DRC...

Current mode: Coordinate: (117, 77) Editing mode: Assignment

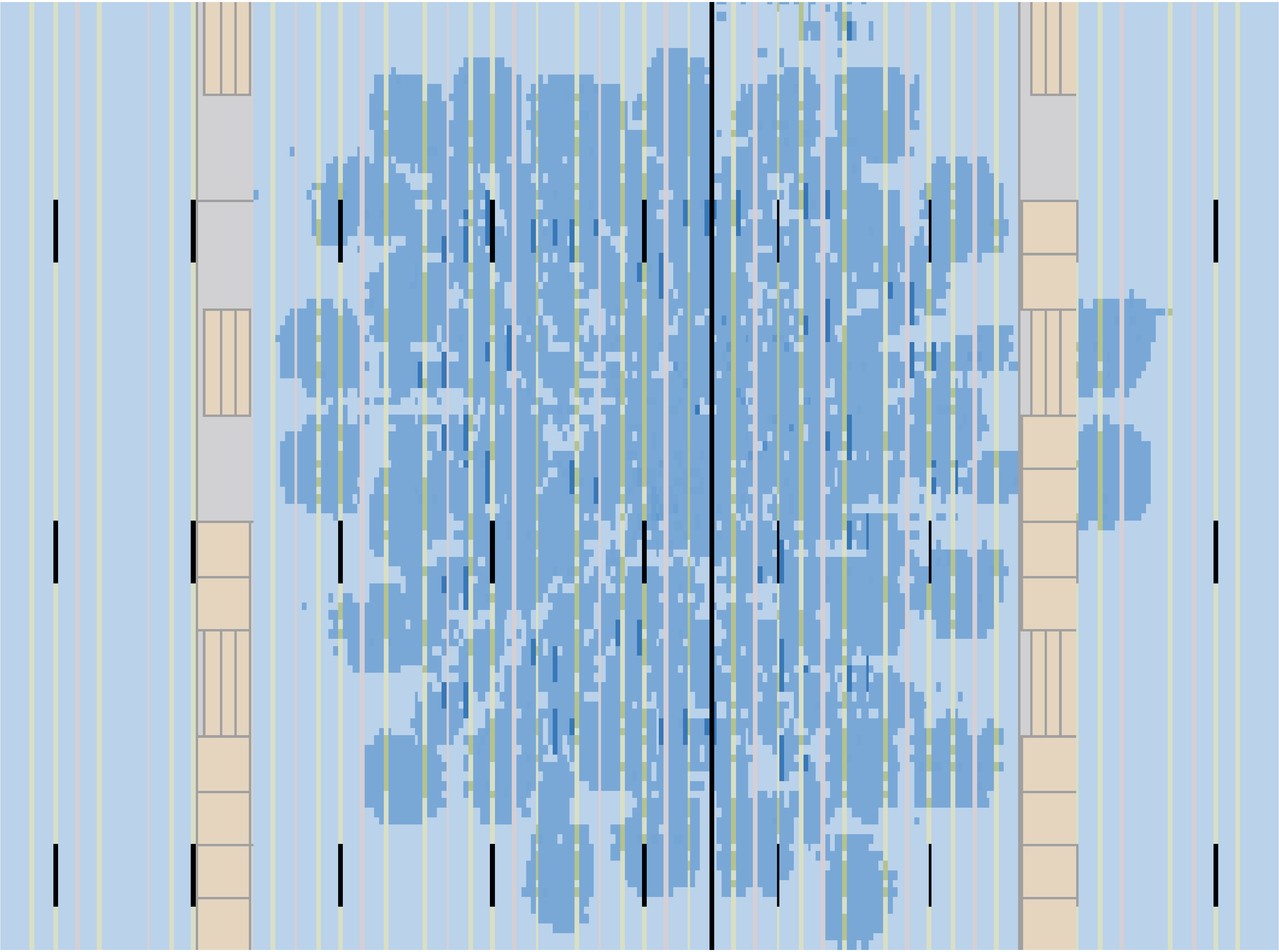


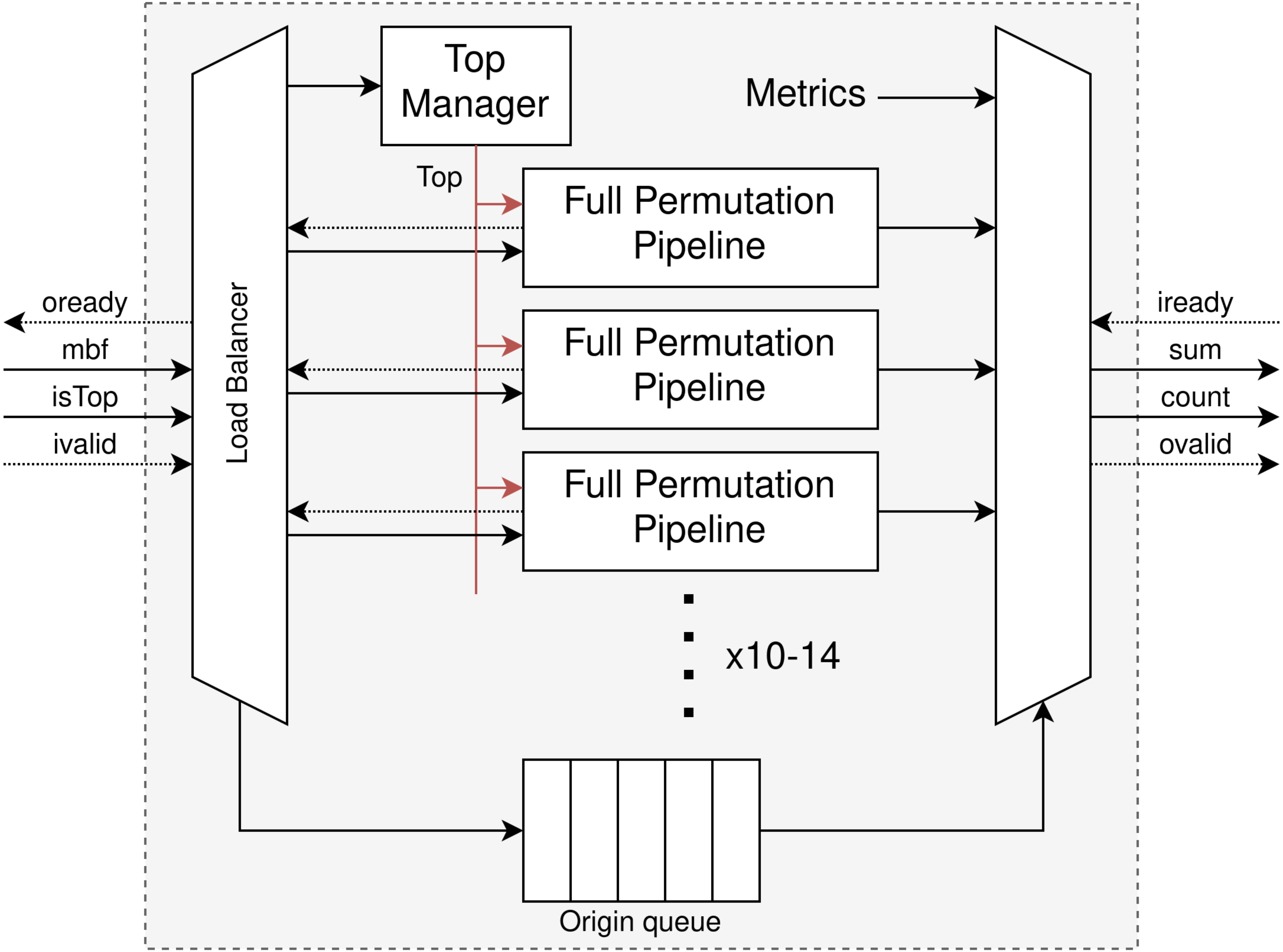
5040 Permutation splitter

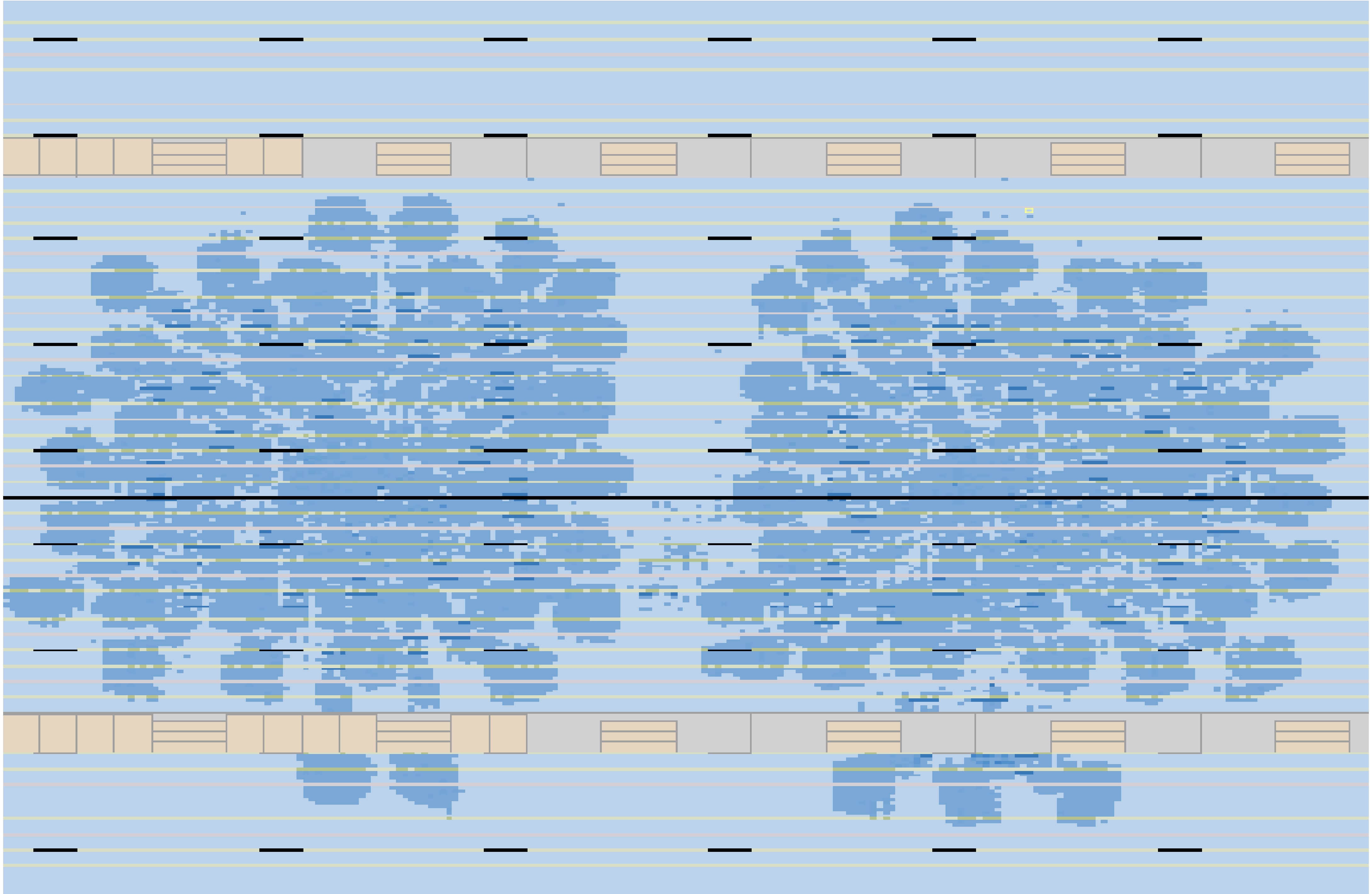
$$\sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} 2^{C_{\alpha,\gamma}}$$

$7! = 5040$

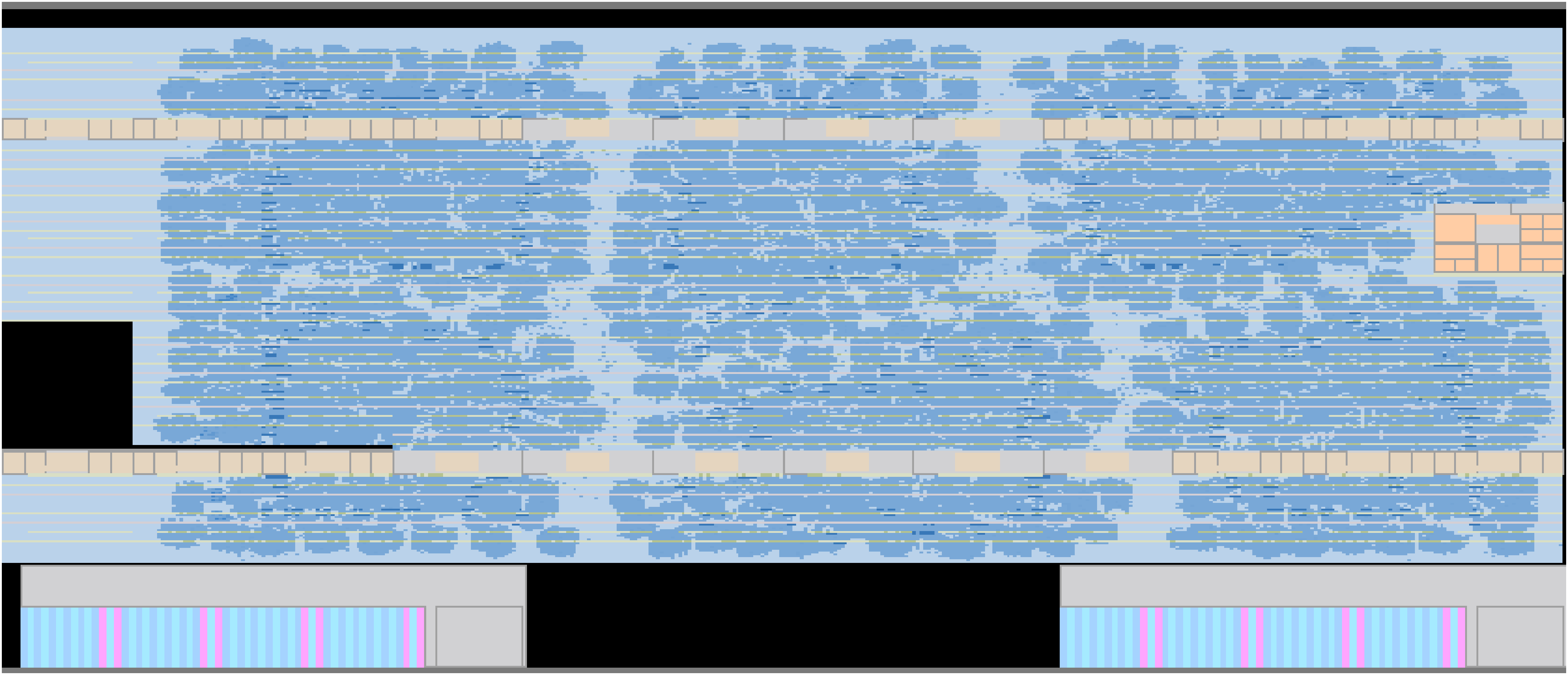
permutations



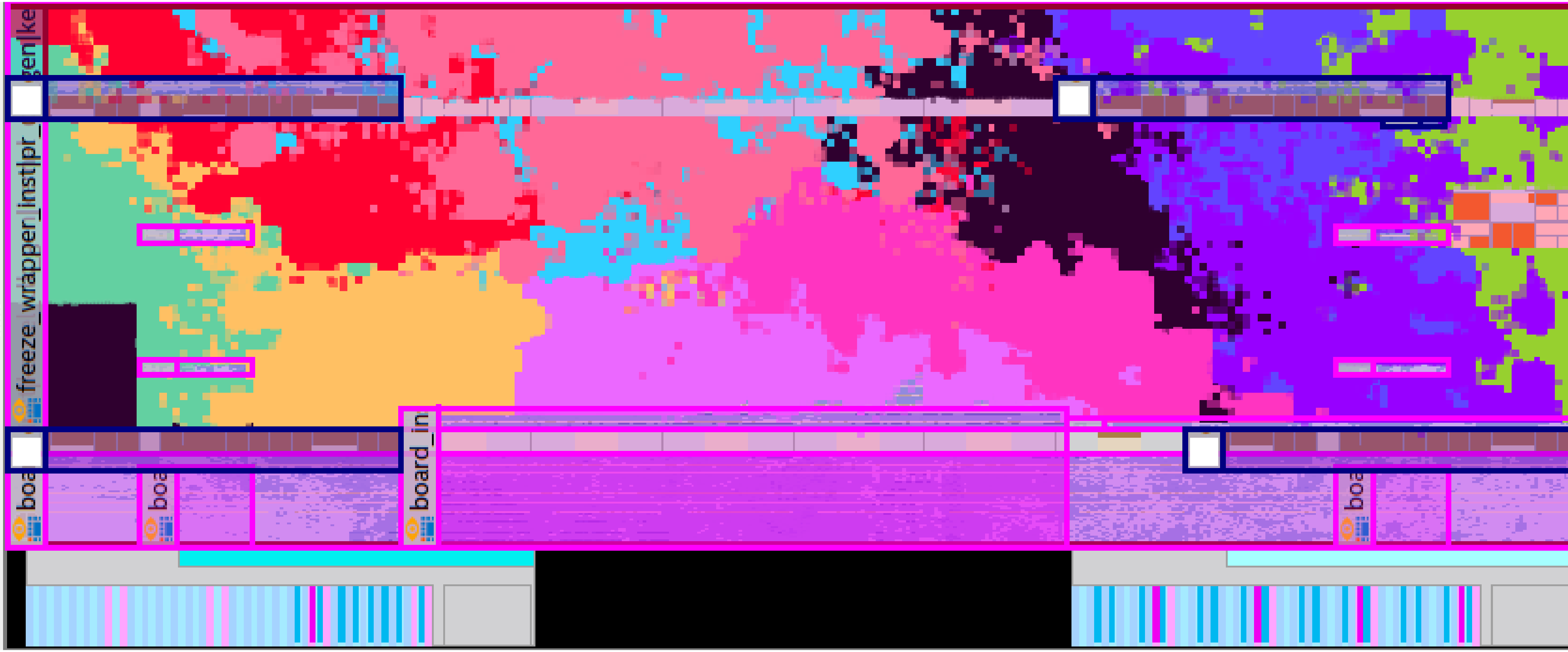

Top Manager
Metrics
Top
Full Permutation Pipeline
Full Permutation Pipeline
Full Permutation Pipeline
x10-14
Load Balancer
oready
mbf
isTop
ivalid
iready
sum
count
ovalid
Origin queue



6 pipelines

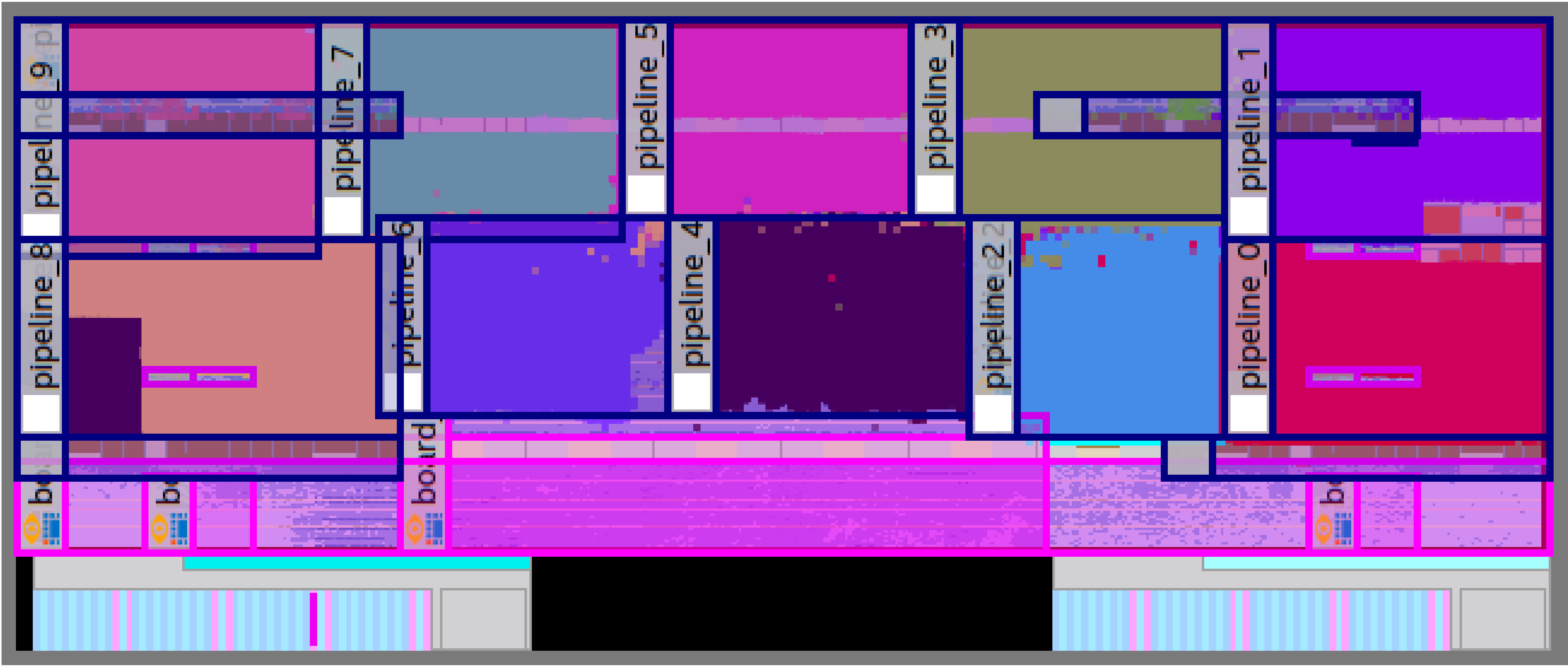


10 pipelines



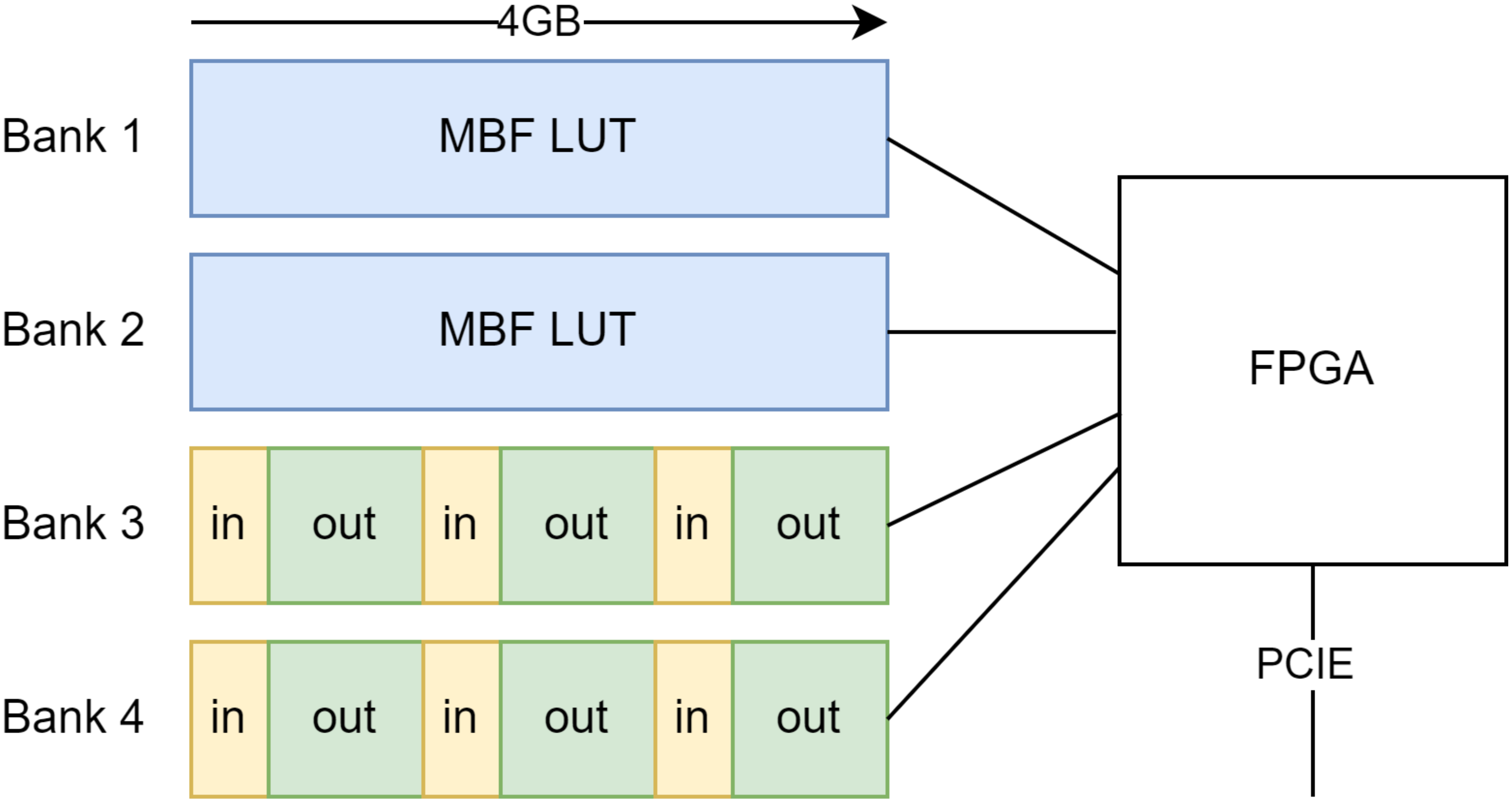
Terrible performance!

Adding logic lock regions

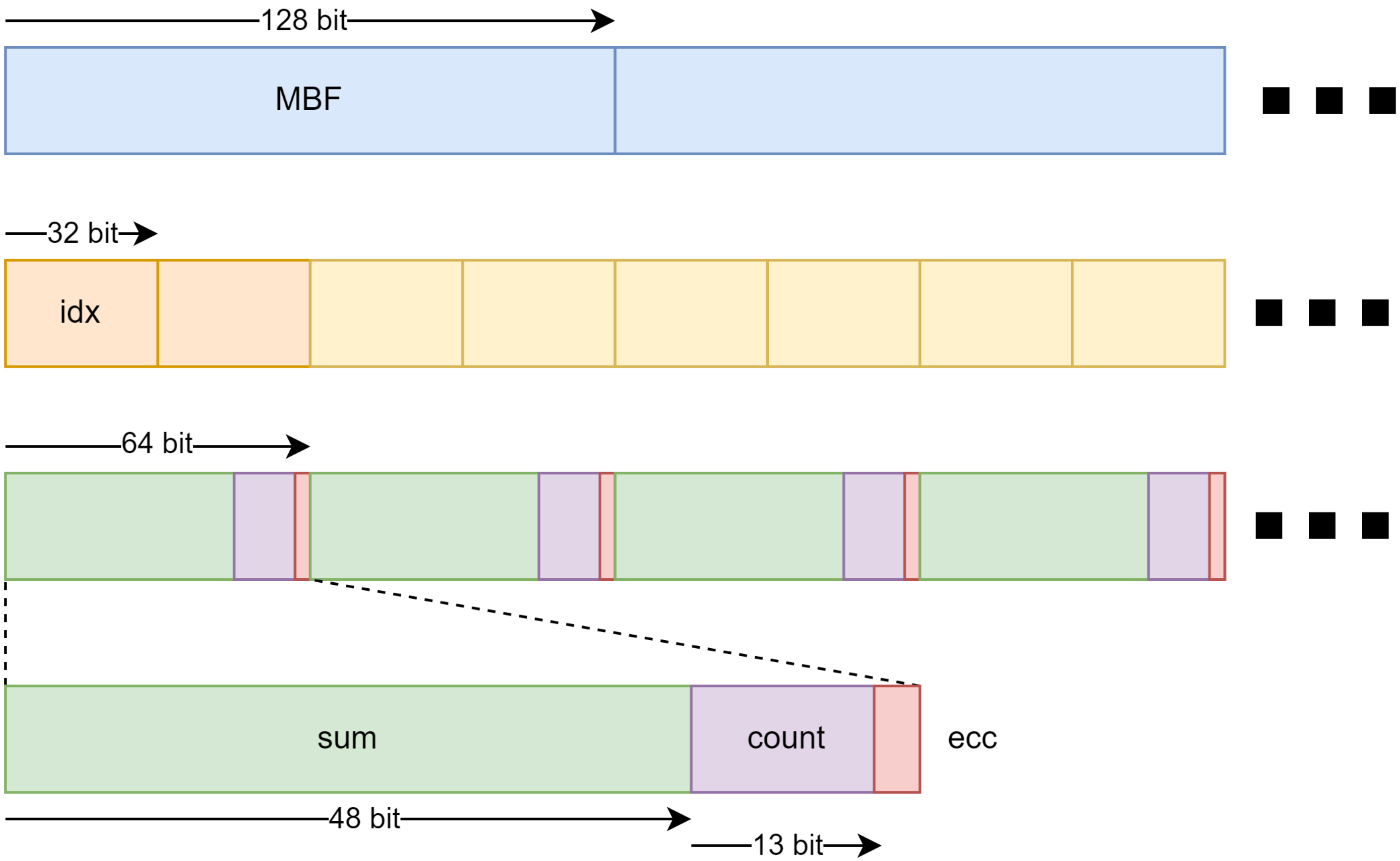


450 MHz! 92% Logic Density 300 Mbots/s

FPGA Card Memory



Buffer Blocks



$$\sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} 2^{C_{\alpha,\gamma}} \qquad \sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} 1$$

Supercomputing

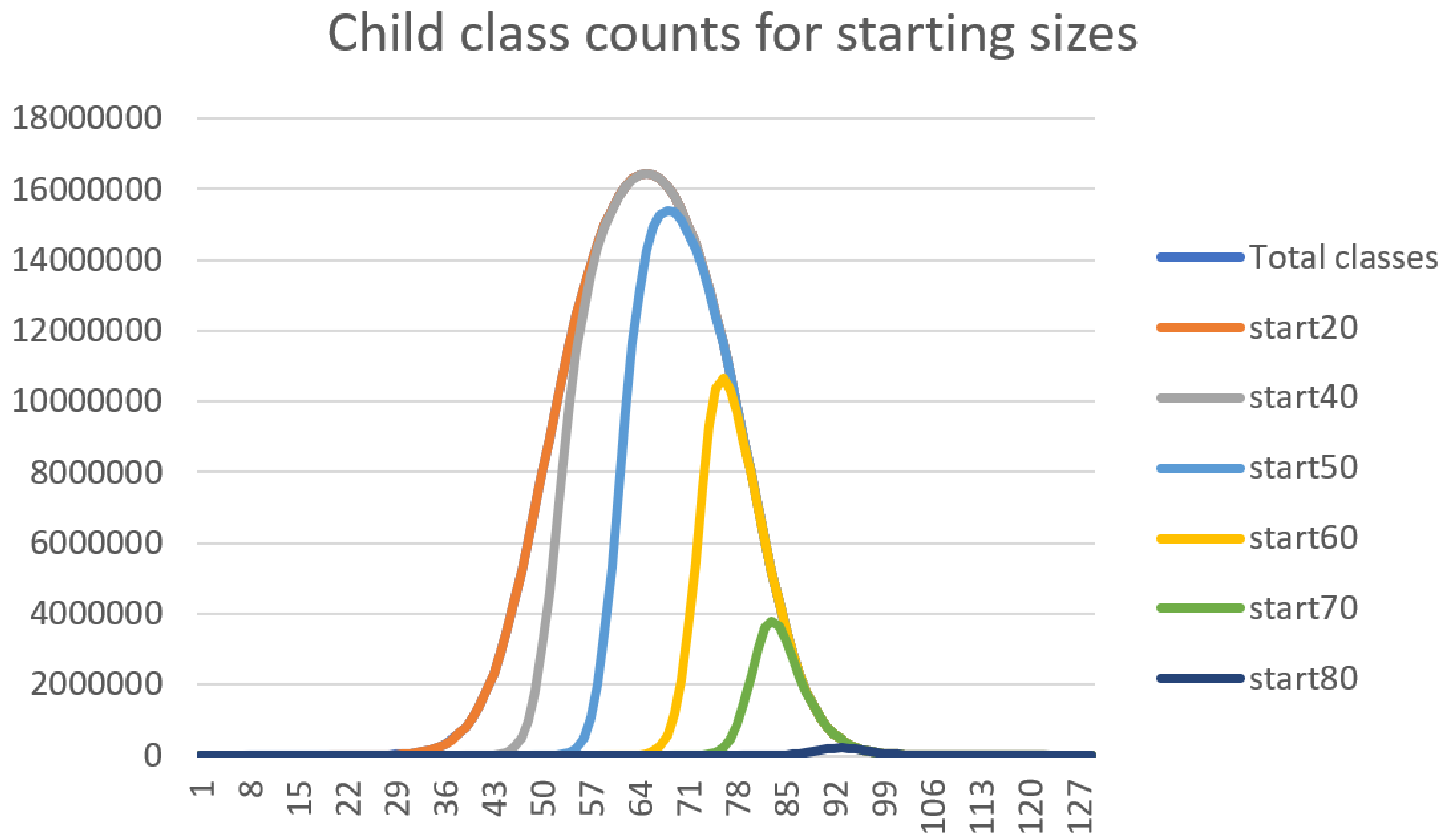
Pcoëfficiënt Recap

$$D(n+2) = \sum_{\alpha \in R_n} |[\bot, \alpha]| D_\alpha \sum_{\substack{\beta \in R_n \\ \exists \delta \simeq \beta : \alpha \leq \delta}} |[\beta, \top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} 2^{C_{\alpha,\gamma}}$$

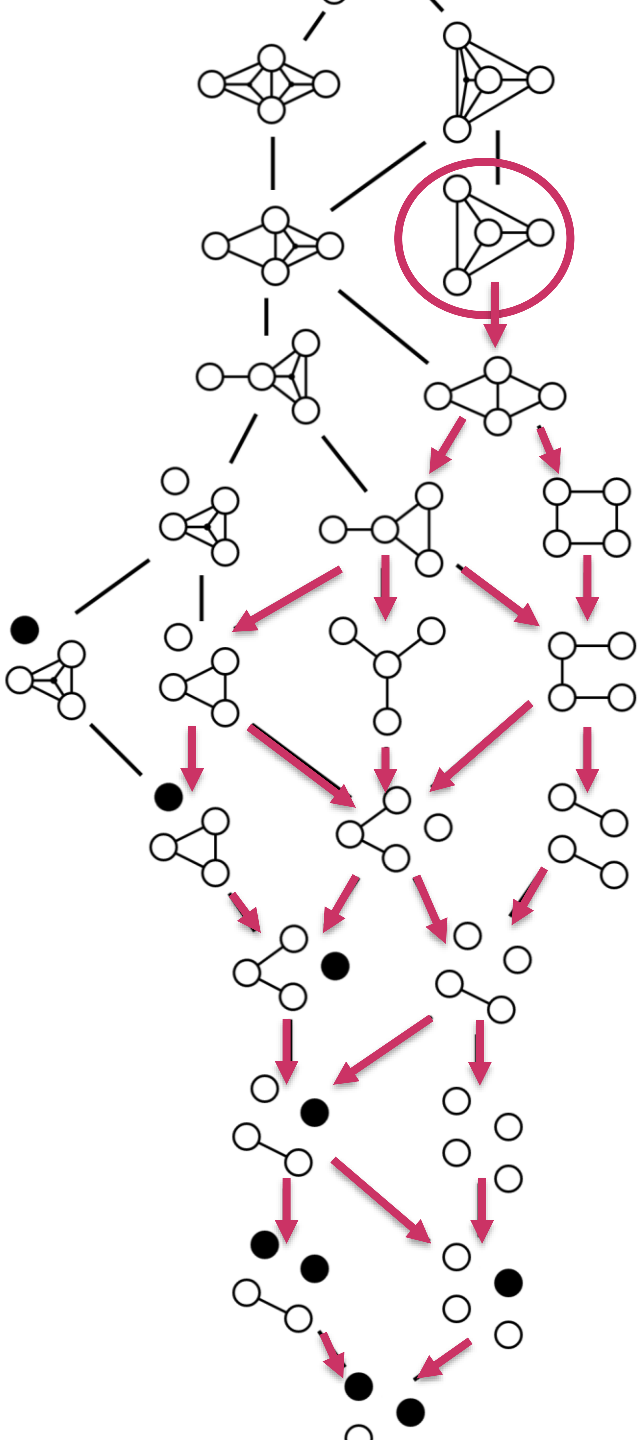
Pcoëfficiënt Recap

$$D(n+2) = \sum_{\alpha \in R_n} |[\bot, \alpha]| D_\alpha \sum_{\substack{\beta \in R_n \\ \underline{\exists \delta \simeq \beta : \alpha \leq \delta}}} |[\beta, \top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} 2^{C_{\alpha,\gamma}}$$

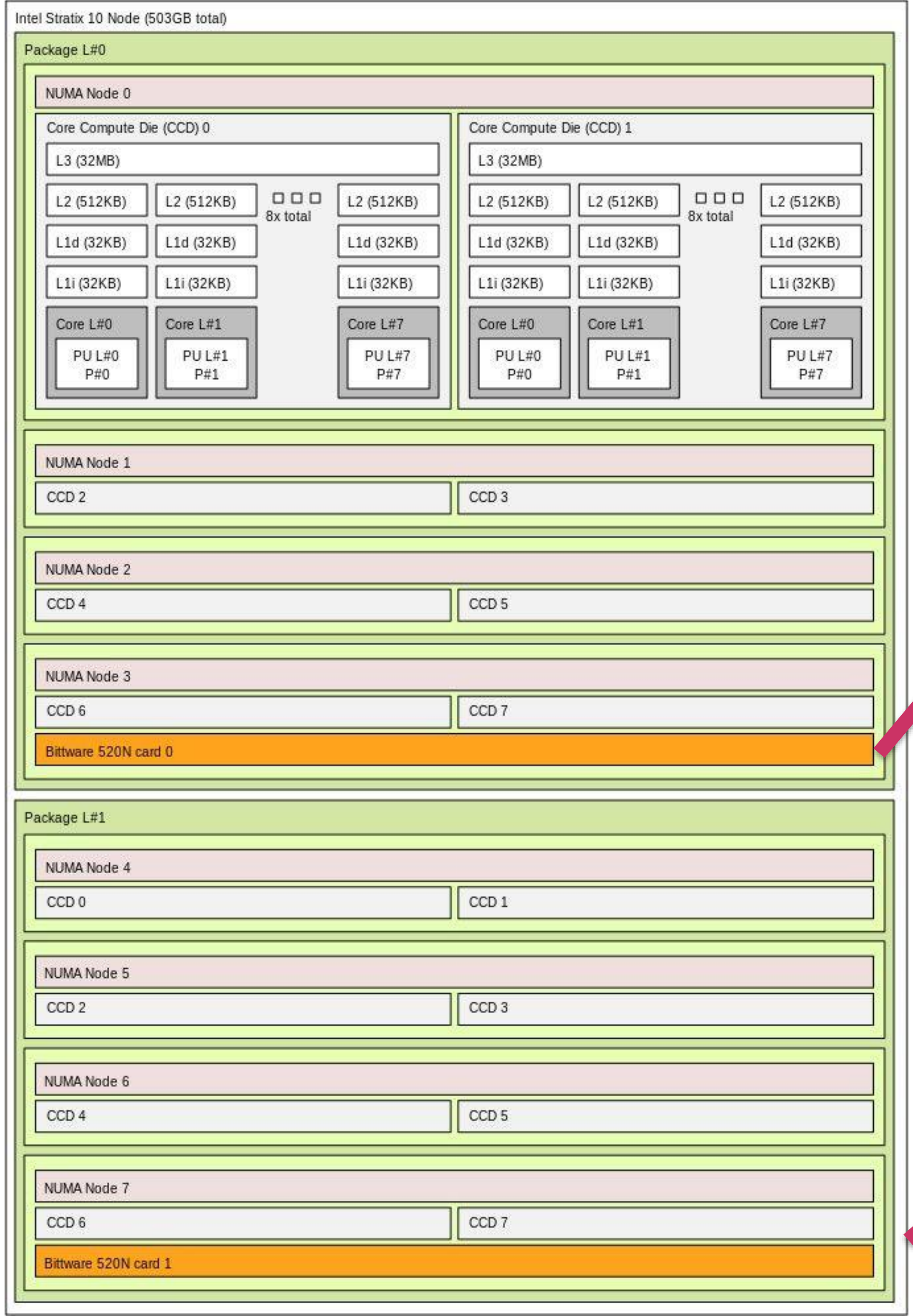
Beta Generation



Beta Generation



Noctua 2 FPGA Node



Intel Stratix 10 Node (503GB total)

Package L#0

NUMA Node 0

Core Compute Die (CCD) 0

L3 (32MB)

L2 (512KB) | L2 (512KB) | 8x total | L2 (512KB)

L1d (32KB) | L1d (32KB) | L1d (32KB)

L1i (32KB) | L1i (32KB) | L1i (32KB)

Core L#0 PU L#0 P#0 | Core L#1 PU L#1 P#1 | Core L#7 PU L#7 P#7

Core Compute Die (CCD) 1

L3 (32MB)

L2 (512KB) | L2 (512KB) | 8x total | L2 (512KB)

L1d (32KB) | L1d (32KB) | L1d (32KB)

L1i (32KB) | L1i (32KB) | L1i (32KB)

Core L#0 PU L#0 P#0 | Core L#1 PU L#1 P#1 | Core L#7 PU L#7 P#7

NUMA Node 1

CCD 2 | CCD 3

NUMA Node 2

CCD 4 | CCD 5

NUMA Node 3

CCD 6 | CCD 7

Bittware 520N card 0

Package L#1

NUMA Node 4

CCD 0 | CCD 1

NUMA Node 5

CCD 2 | CCD 3

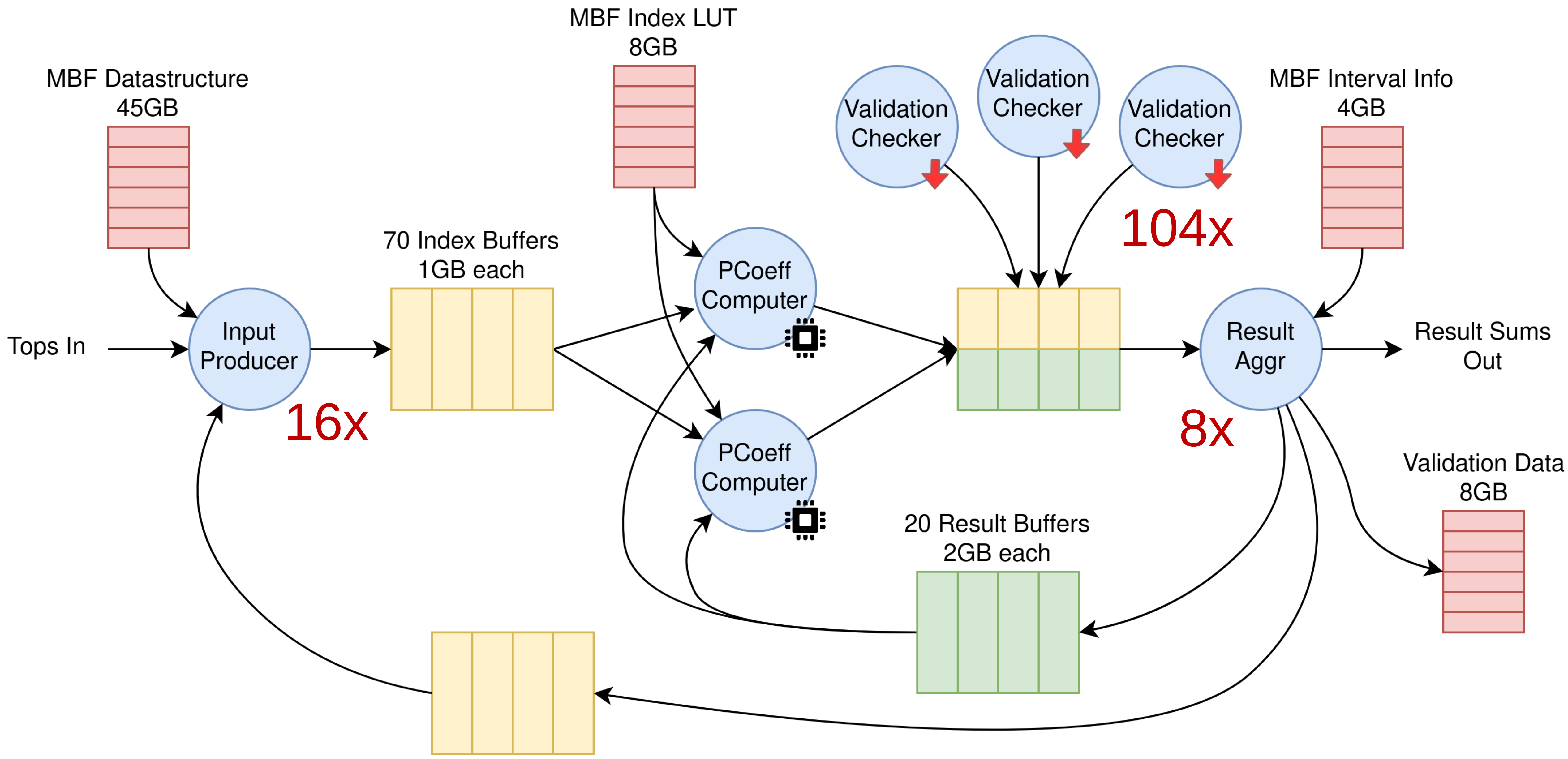
NUMA Node 6

CCD 4 | CCD 5

NUMA Node 7

CCD 6 | CCD 7

Bittware 520N card 1


MBF Datastructure
45GB
MBF Index LUT
8GB
Validation Checker
Validation Checker
Validation Checker
MBF Interval Info
4GB
104x
70 Index Buffers
1GB each
PCoeff Computer
Tops In
Input Producer
Result Aggr
Result Sums Out
16x
PCoeff Computer
8x
Validation Data
8GB
20 Result Buffers
2GB each

Pcoëfficiënt Recap

$$D(n+2) = \sum_{\alpha\in R_n} |[\bot,\alpha]| D_\alpha \sum_{\substack{\beta\in R_n \\ \underline{\exists\delta\simeq\beta:\alpha\leq\delta}}} |[\beta,\top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma\in Permut_\beta \\ \alpha\leq\gamma}} 2^{C_{\alpha,\gamma}}$$

Pcoëfficiënt Recap

$$D(n+2) = \sum_{\alpha \in R_n} |[\bot, \alpha]| D_\alpha \sum_{\substack{\beta \in R_n \\ \exists \delta \simeq \beta : \alpha \leq \delta}} |[\beta, \top]| \frac{D_\beta}{n!} \sum_{\substack{\gamma \in Permut_\beta \\ \alpha \leq \gamma}} 2^{C_{\alpha,\gamma}}$$

5.8 α per second

490 million in total

Computation on Noctua 2

- 15’000 jobs
- 33’000 tops/job
- 100 mins / job
- 16 FPGA servers

Computation on Noctua 2

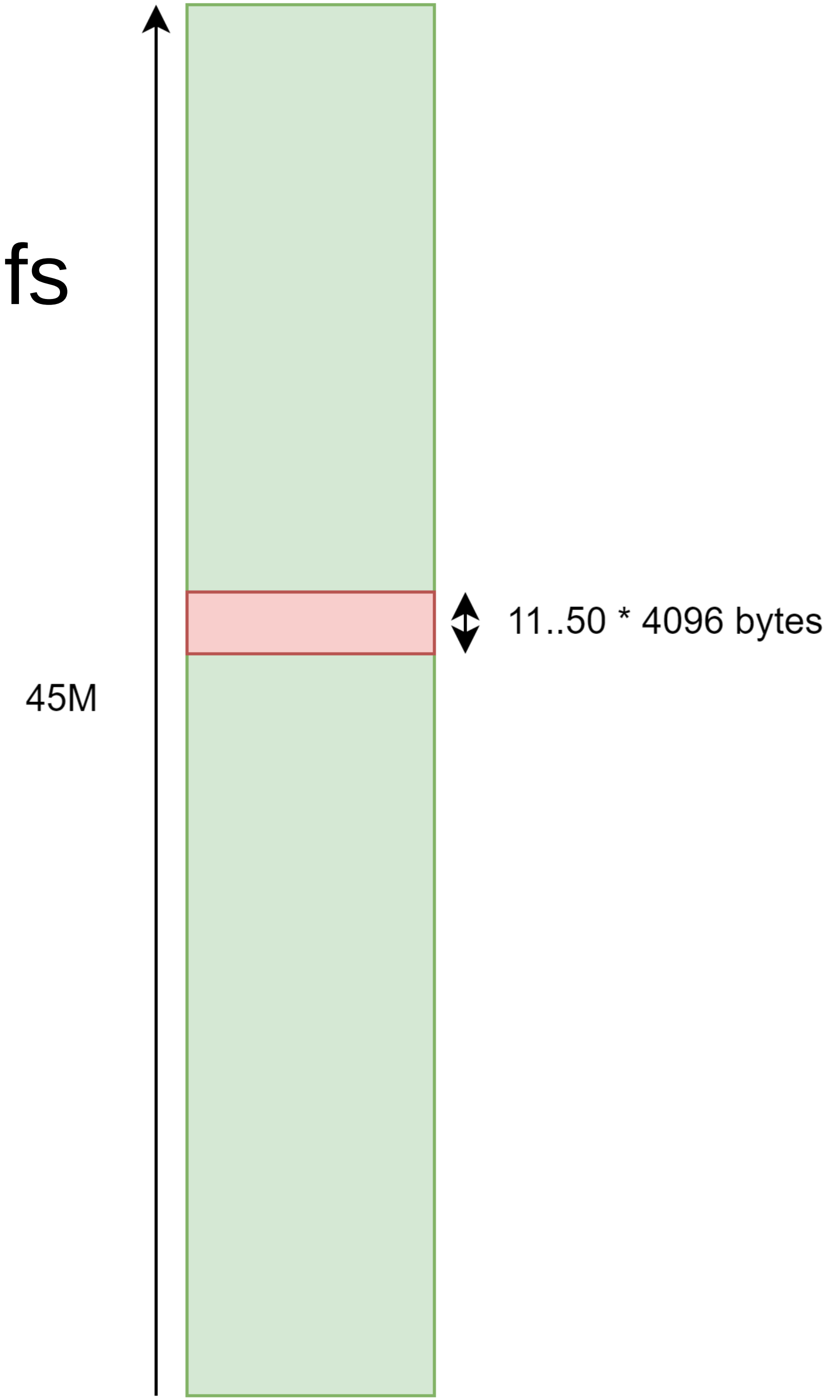
- After 4 months on Noctua 2
- 47’000 FPGA hours in total

March 8th 2023, at 5pm:

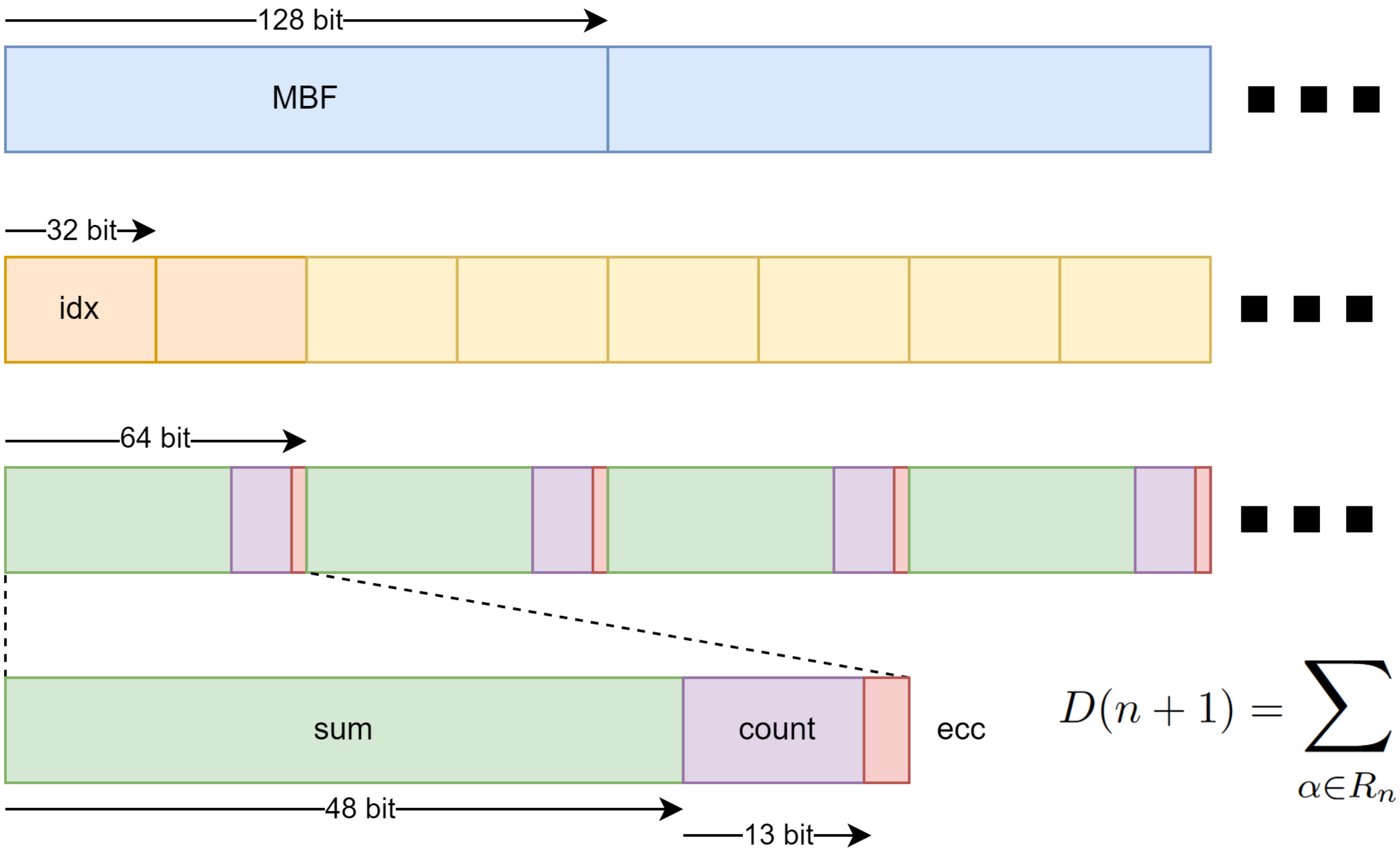
286386577668298411128469151667598498812366

Errors

- Large error blocks in result bufs
- Aligned to 4096 bytes
- “Forgot to copy”?
- Fixed now



Errors



$$\sum_{\substack{\gamma\in Permut_\beta\\ \alpha\le\gamma}} 2^{C_{\alpha,\gamma}}$$

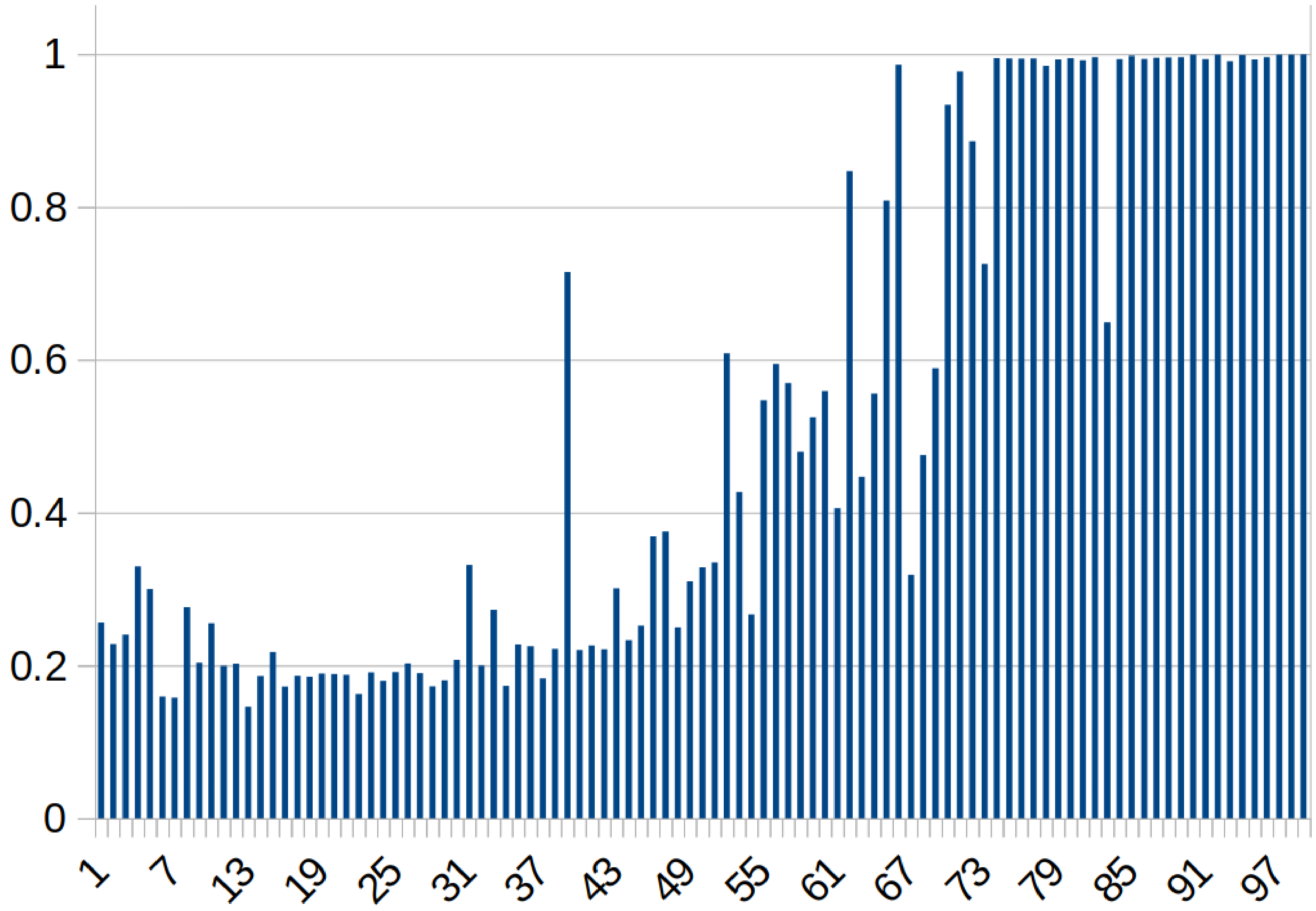
$$\sum_{\substack{\gamma\in Permut_\beta\\ \alpha\le\gamma}} 1$$

$$D(n+1) = \sum_{\alpha\in R_n} D_\alpha \sum_{\substack{\beta\in R_n\\ \exists\delta\simeq\beta:\alpha\le\delta}} \frac{D_\beta}{n!} \sum_{\substack{\gamma\in Permut_\beta\\ \alpha\le\gamma}} 1$$

Lucky Checksum!

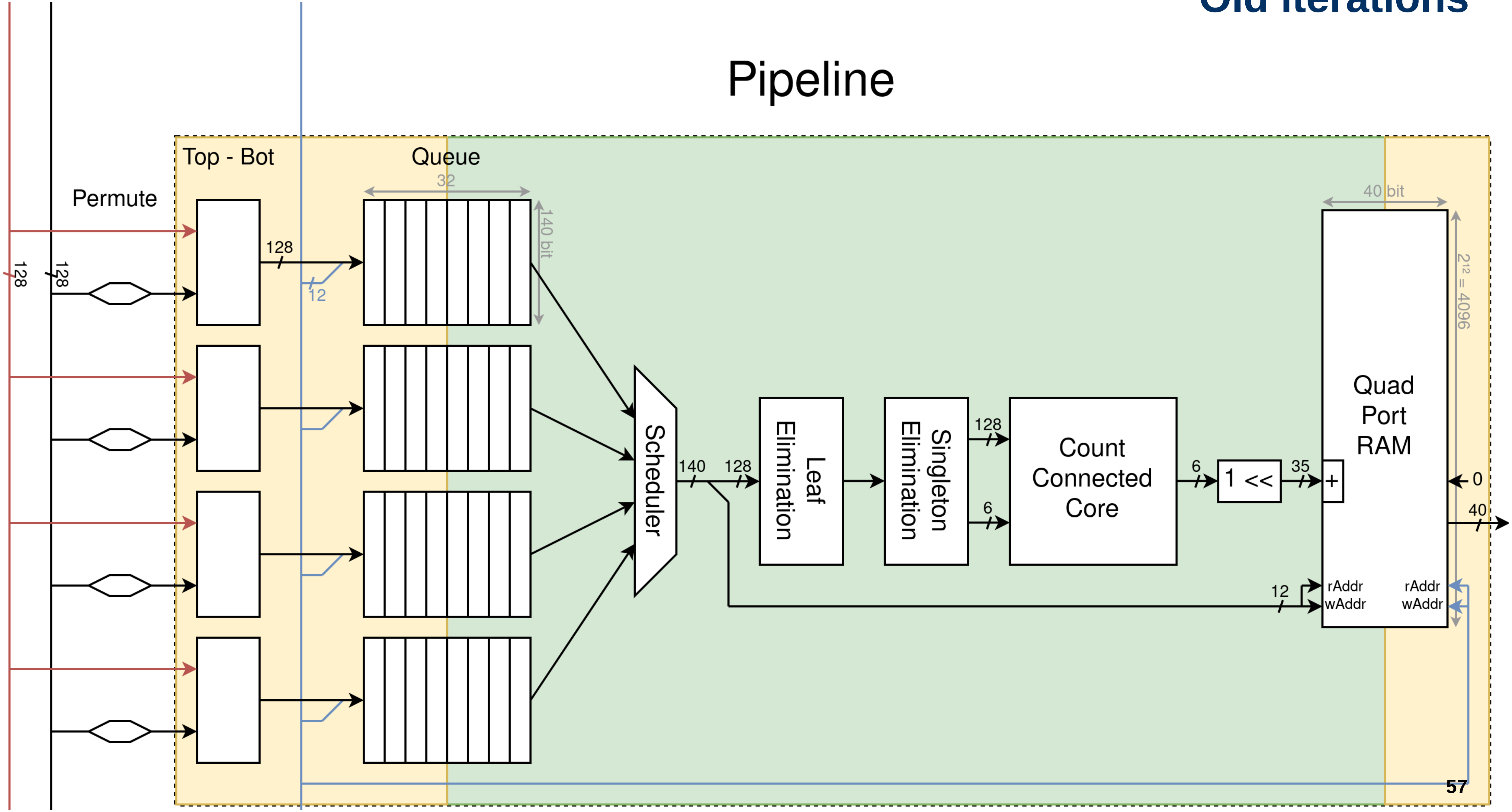
Appendix

Efficiency Distribution

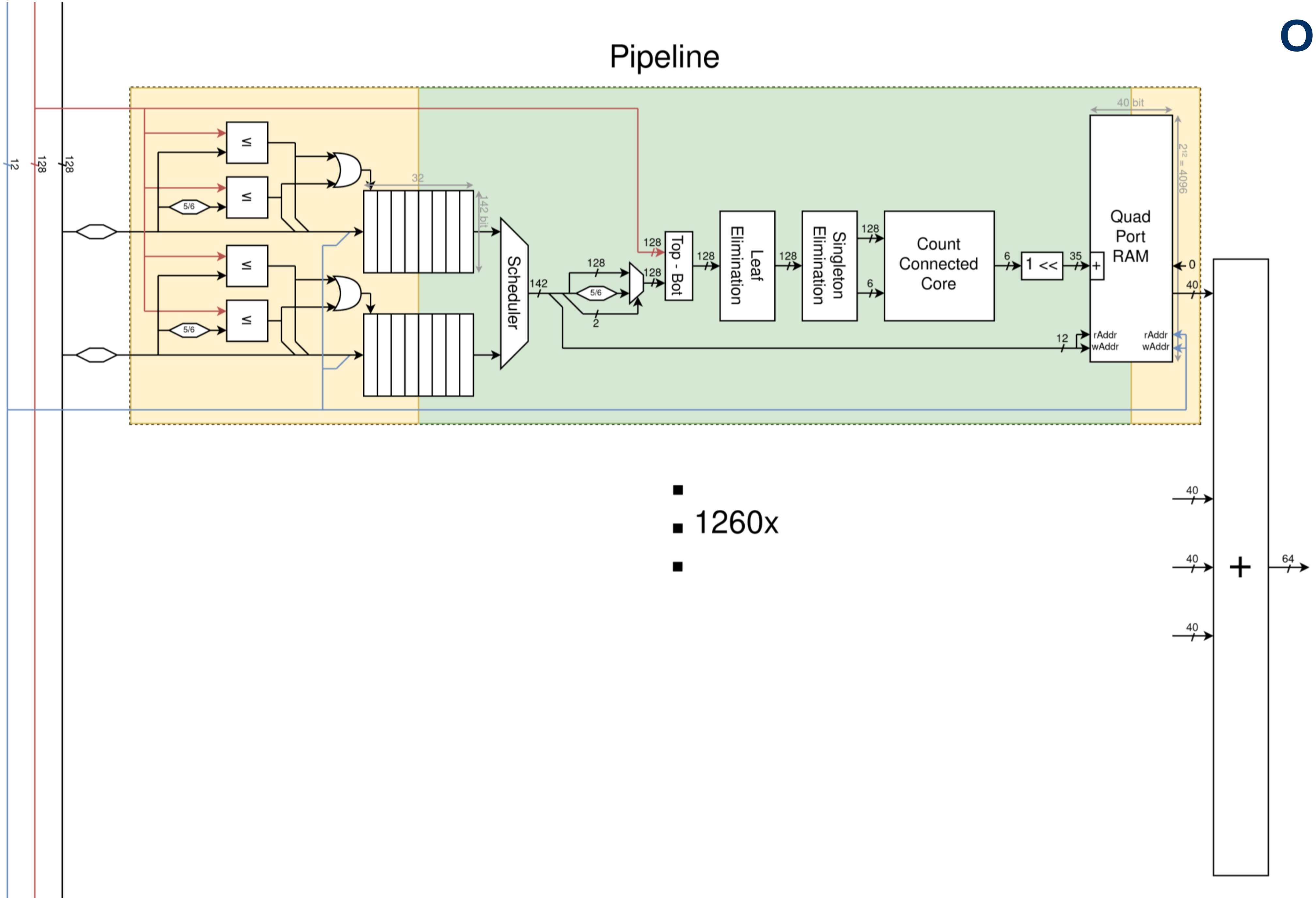


Old iterations

Pipeline



Old iterations



-
- 1260x
-