9th Dedekind Project

Structure

- Introduction to the problem
- Core Formula
- Implementation
- Numbers

Dedekind Numbers

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)	?	?

Core “Jump” Formula

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

Modified Formula

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

Modified Formula

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

“Tops”

“Bottoms”

“Permutations”

Modified Formula

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

“Tops”

“Bottoms”

“Permutations”

For D(9): 490’013’148 tops ~91’000’000 bottoms/top ~250/5040 permutations/bottom

Modified Formula

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

“Tops”

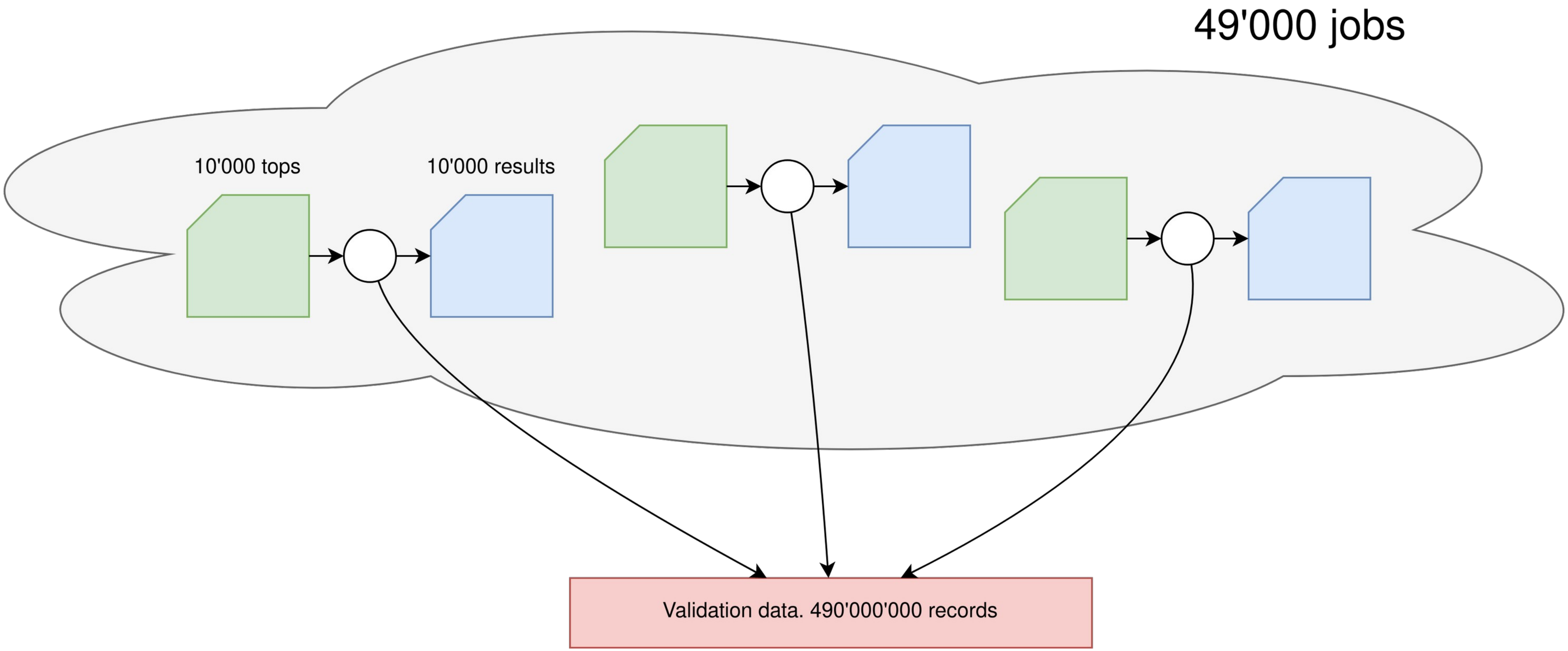
“Bottoms”

“Permutations”

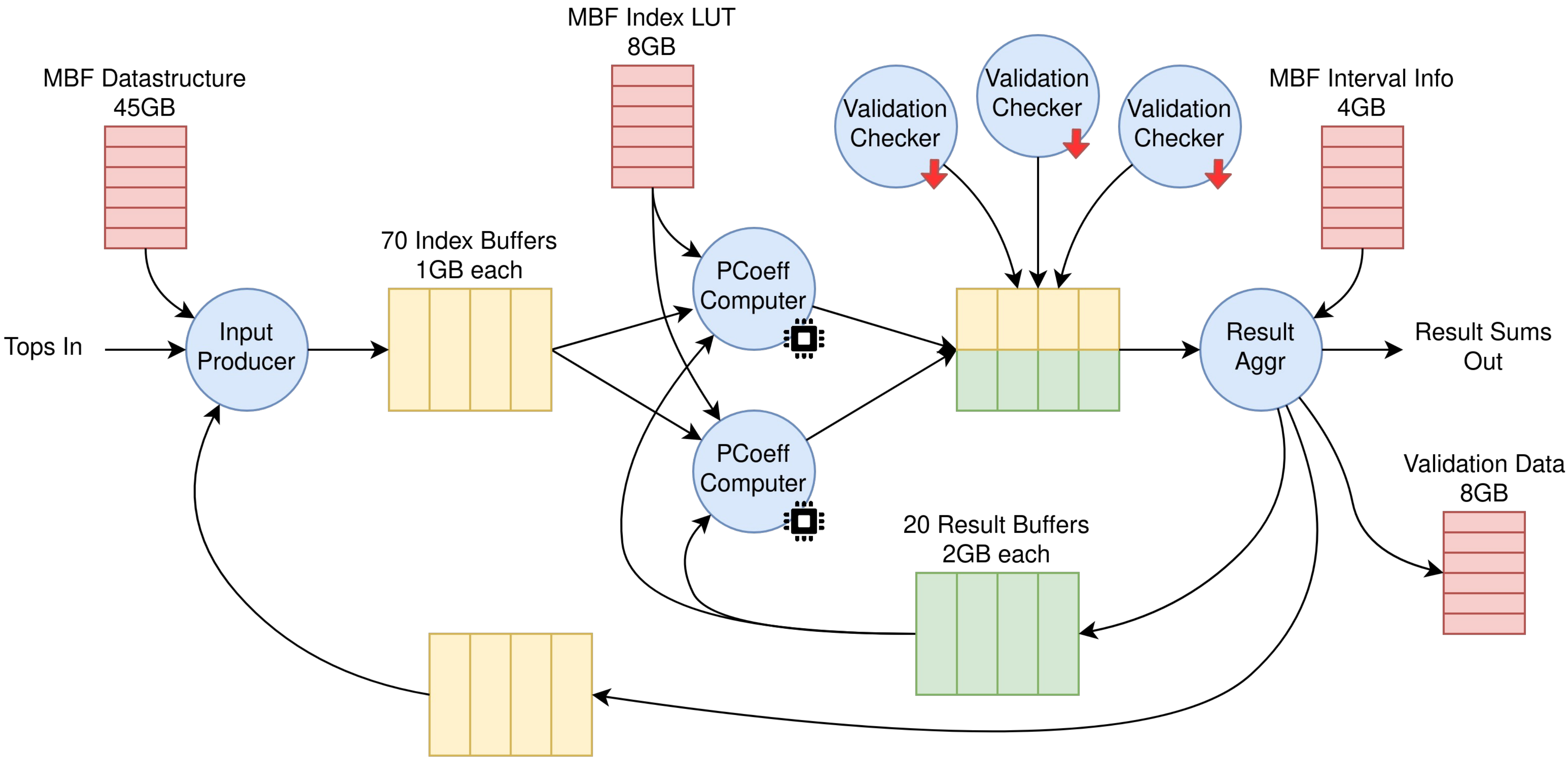
For D(9): 490’013’148 tops ~91’000’000 bottoms/top ~250/5040 permutations/bottom

Total bottoms: 4.59 * 10^16 Total terms: 11483553838459169213 ≈ 1.148 * 10^19

Jobs

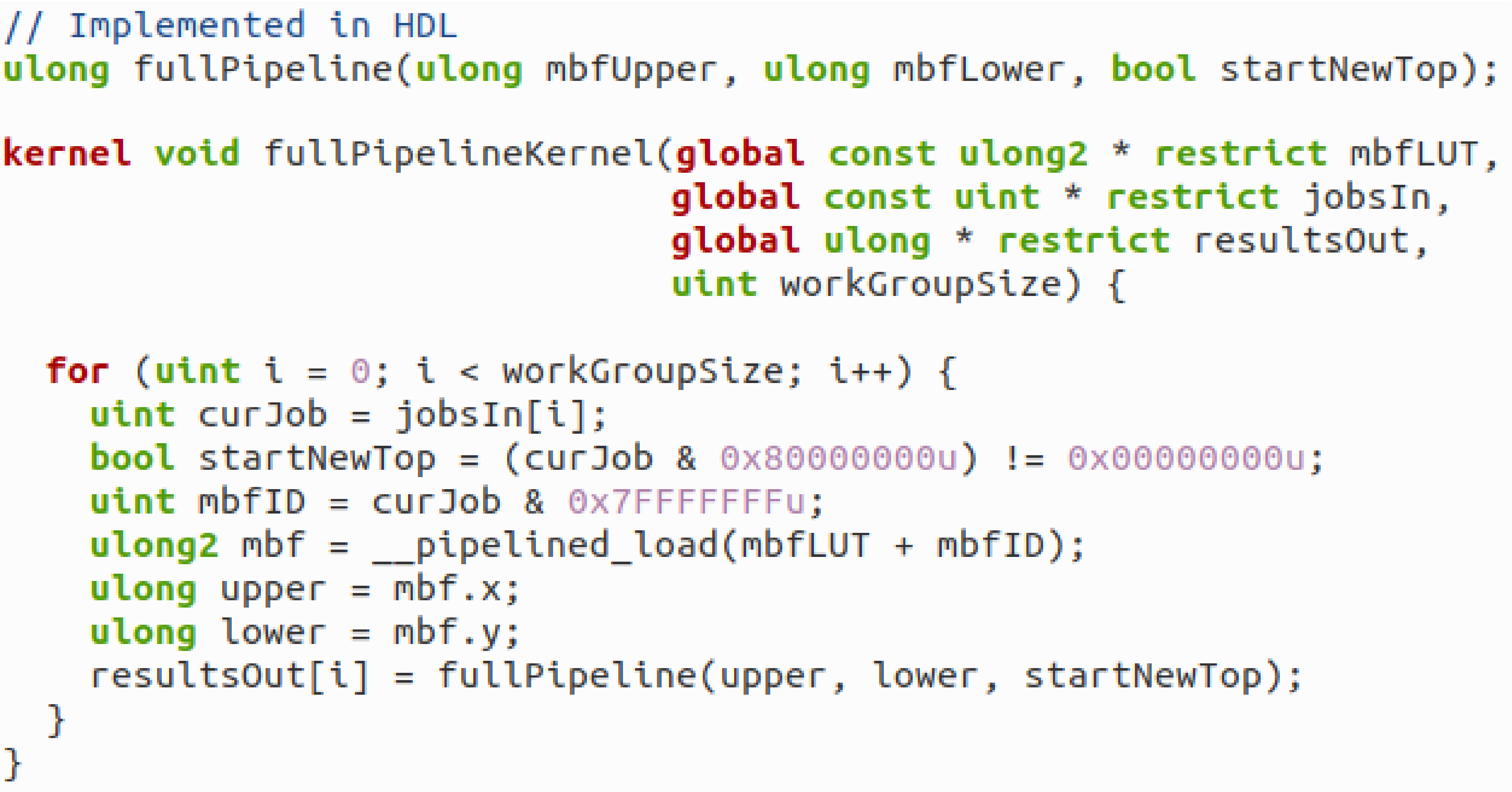

49'000 jobs
10'000 tops
10'000 results
Validation data. 490'000'000 records

Threads



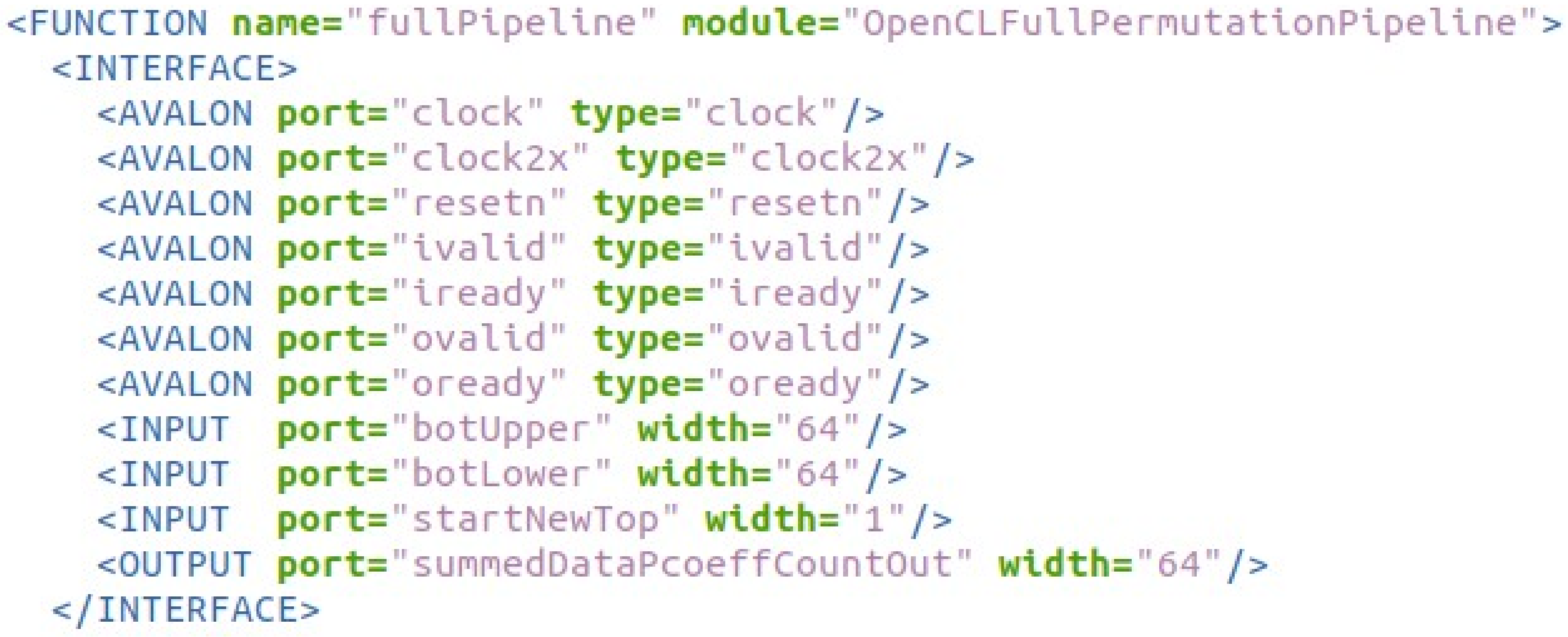
MBF Index LUT
8GB
MBF Datastructure
45GB
Validation Checker
Validation Checker
Validation Checker
MBF Interval Info
4GB
70 Index Buffers
1GB each
PCoeff Computer
Tops In
Input Producer
Result Aggr
Result Sums Out
PCoeff Computer
Validation Data
8GB
20 Result Buffers
2GB each

OpenCL



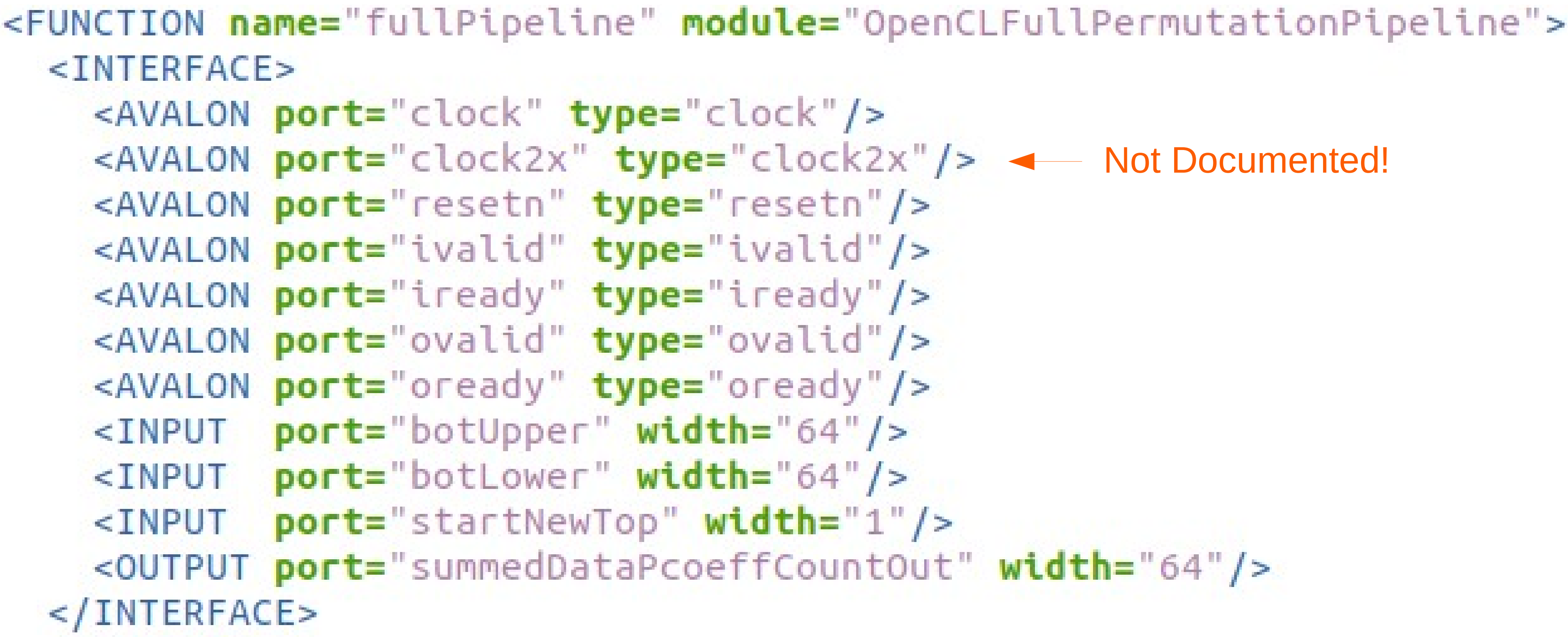
```
// Implemented in HDL
ulong fullPipeline(ulong mbfUpper, ulong mbfLower, bool startNewTop);

kernel void fullPipelineKernel(global const ulong2 * restrict mbfLUT,
                               global const uint * restrict jobsIn,
                               global ulong * restrict resultsOut,
                               uint workGroupSize) {

  for (uint i = 0; i < workGroupSize; i++) {
    uint curJob = jobsIn[i];
    bool startNewTop = (curJob & 0x80000000u) != 0x00000000u;
    uint mbfID = curJob & 0x7FFFFFFFu;
    ulong2 mbf = __pipelined_load(mbfLUT + mbfID);
    ulong upper = mbf.x;
    ulong lower = mbf.y;
    resultsOut[i] = fullPipeline(upper, lower, startNewTop);
  }
}
```

Kernel



```
<FUNCTION name="fullPipeline" module="OpenCLFullPermutationPipeline">
  <INTERFACE>
    <AVALON port="clock" type="clock"/>
    <AVALON port="clock2x" type="clock2x"/>
    <AVALON port="resetn" type="resetn"/>
    <AVALON port="ivalid" type="ivalid"/>
    <AVALON port="iready" type="iready"/>
    <AVALON port="ovalid" type="ovalid"/>
    <AVALON port="oready" type="oready"/>
    <INPUT  port="botUpper" width="64"/>
    <INPUT  port="botLower" width="64"/>
    <INPUT  port="startNewTop" width="1"/>
    <OUTPUT port="summedDataPcoeffCountOut" width="64"/>
  </INTERFACE>
```

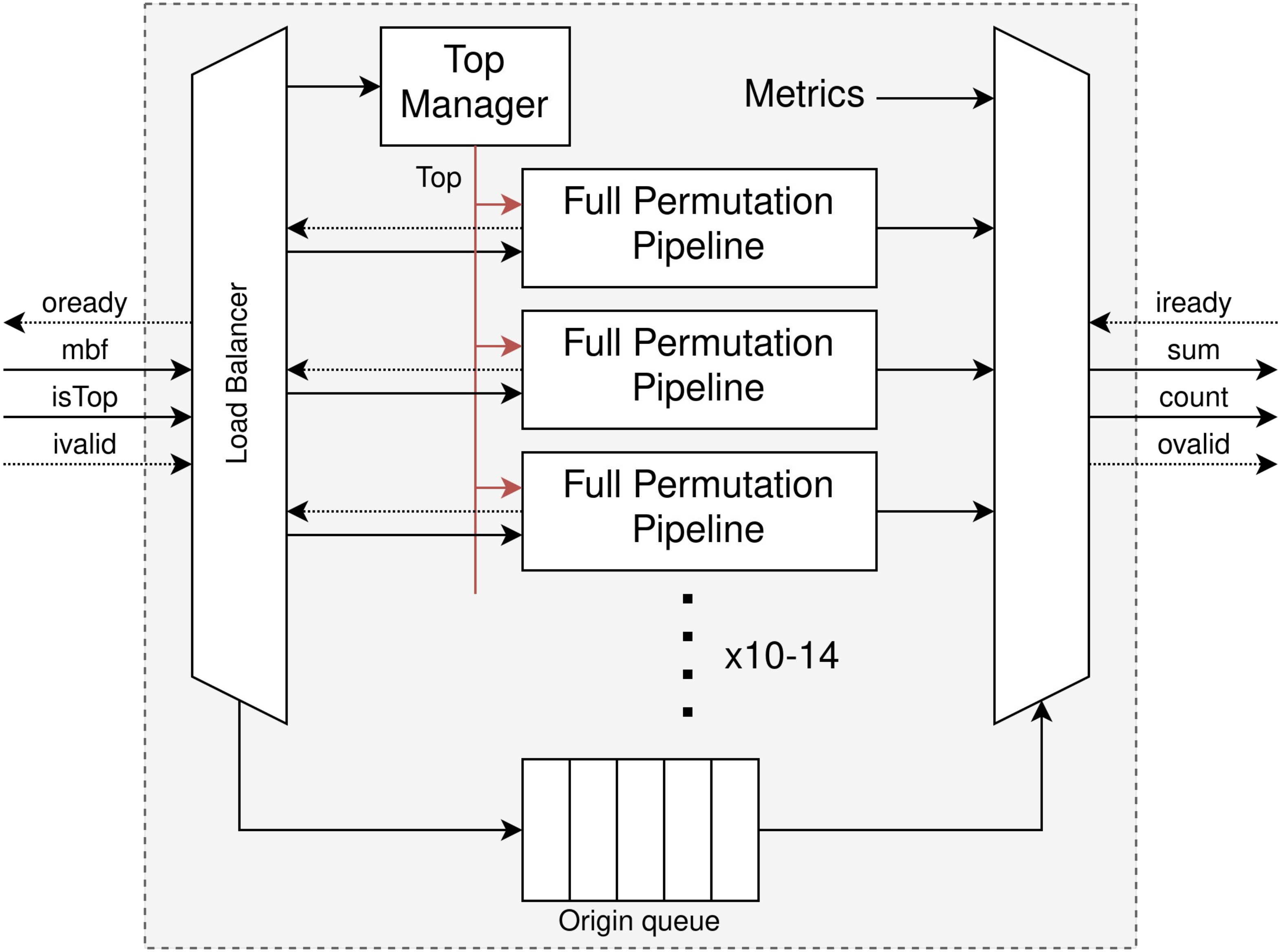
Kernel



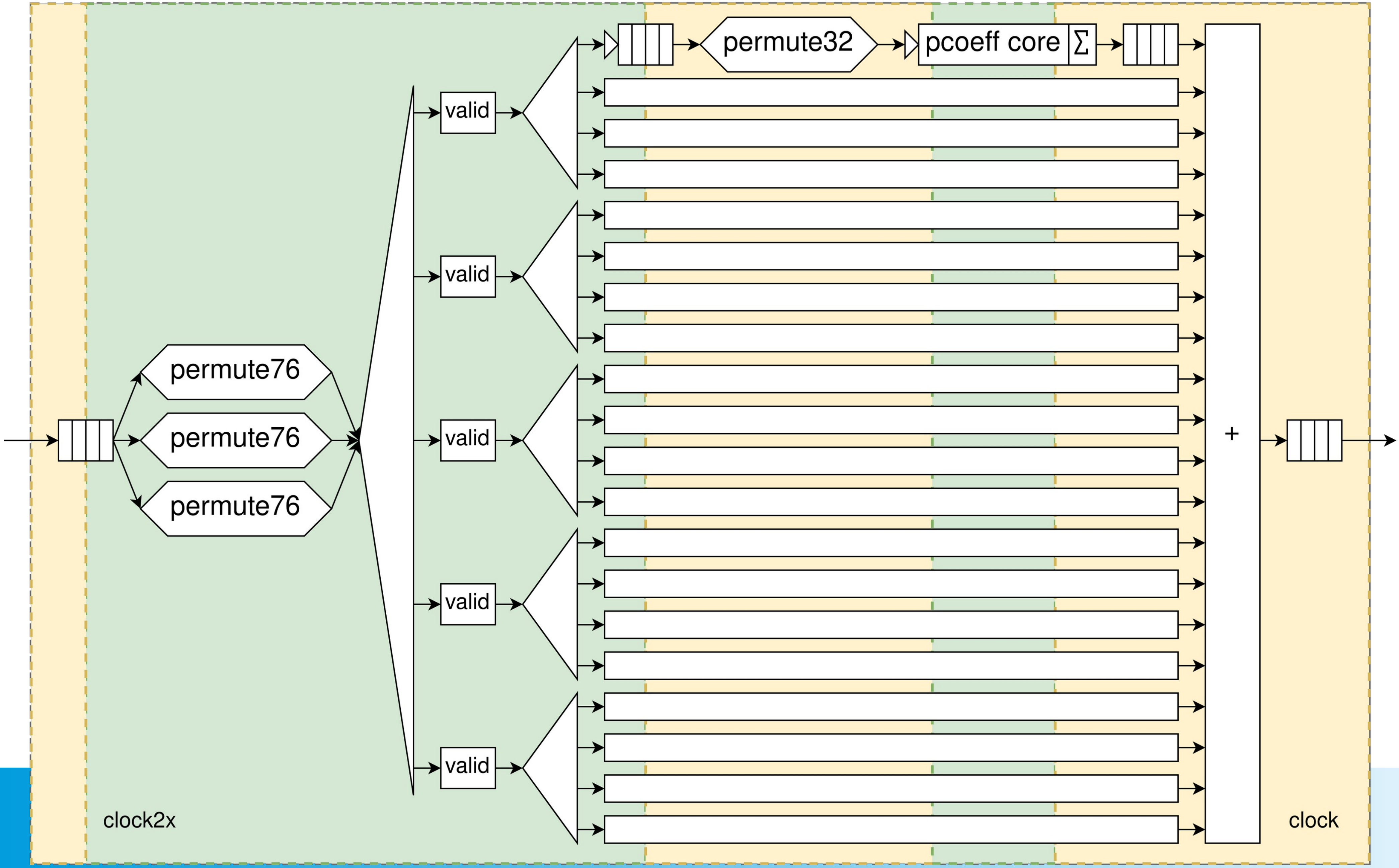
```
<FUNCTION name="fullPipeline" module="OpenCLFullPermutationPipeline">
  <INTERFACE>
    <AVALON port="clock" type="clock"/>
    <AVALON port="clock2x" type="clock2x"/>
    <AVALON port="resetn" type="resetn"/>
    <AVALON port="ivalid" type="ivalid"/>
    <AVALON port="iready" type="iready"/>
    <AVALON port="ovalid" type="ovalid"/>
    <AVALON port="oready" type="oready"/>
    <INPUT  port="botUpper" width="64"/>
    <INPUT  port="botLower" width="64"/>
    <INPUT  port="startNewTop" width="1"/>
    <OUTPUT port="summedDataPcoeffCountOut" width="64"/>
  </INTERFACE>
```

Not Documented!

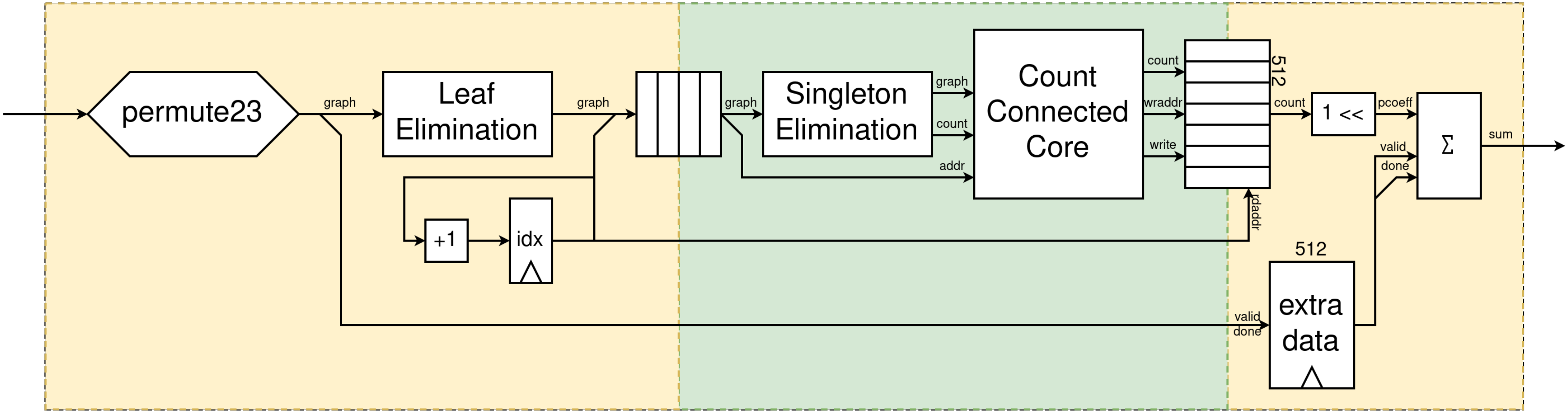
fullPipeline



FullPermutationPipeline



Pcoeff Core



permute23

graph

Leaf Elimination

graph

+1

idx

graph

Singleton Elimination

graph

count

addr

Count Connected Core

count

wraddr

write

512

rdaddr

count

1 <<

pcoeff

valid

done

Σ

sum

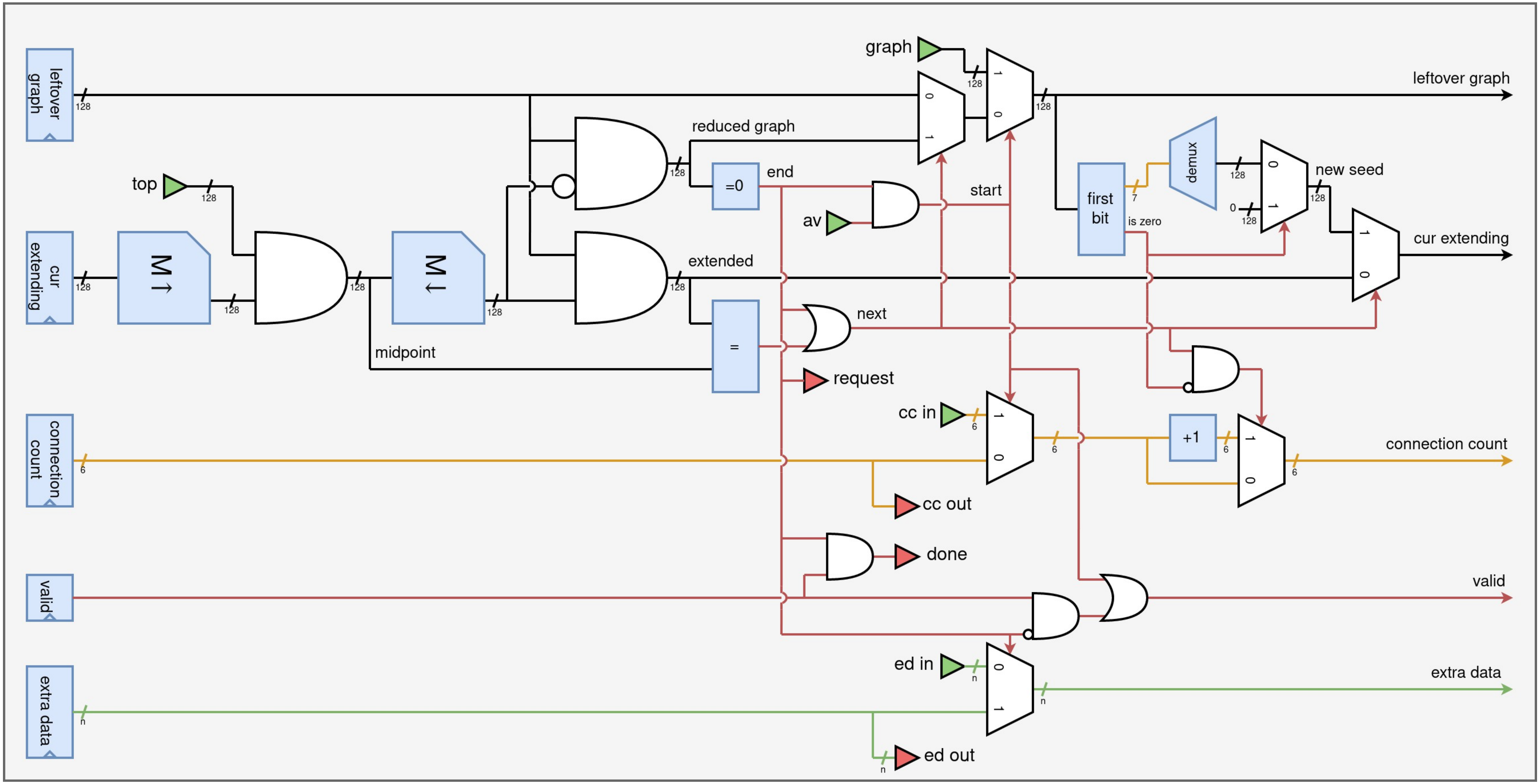
512

extra data

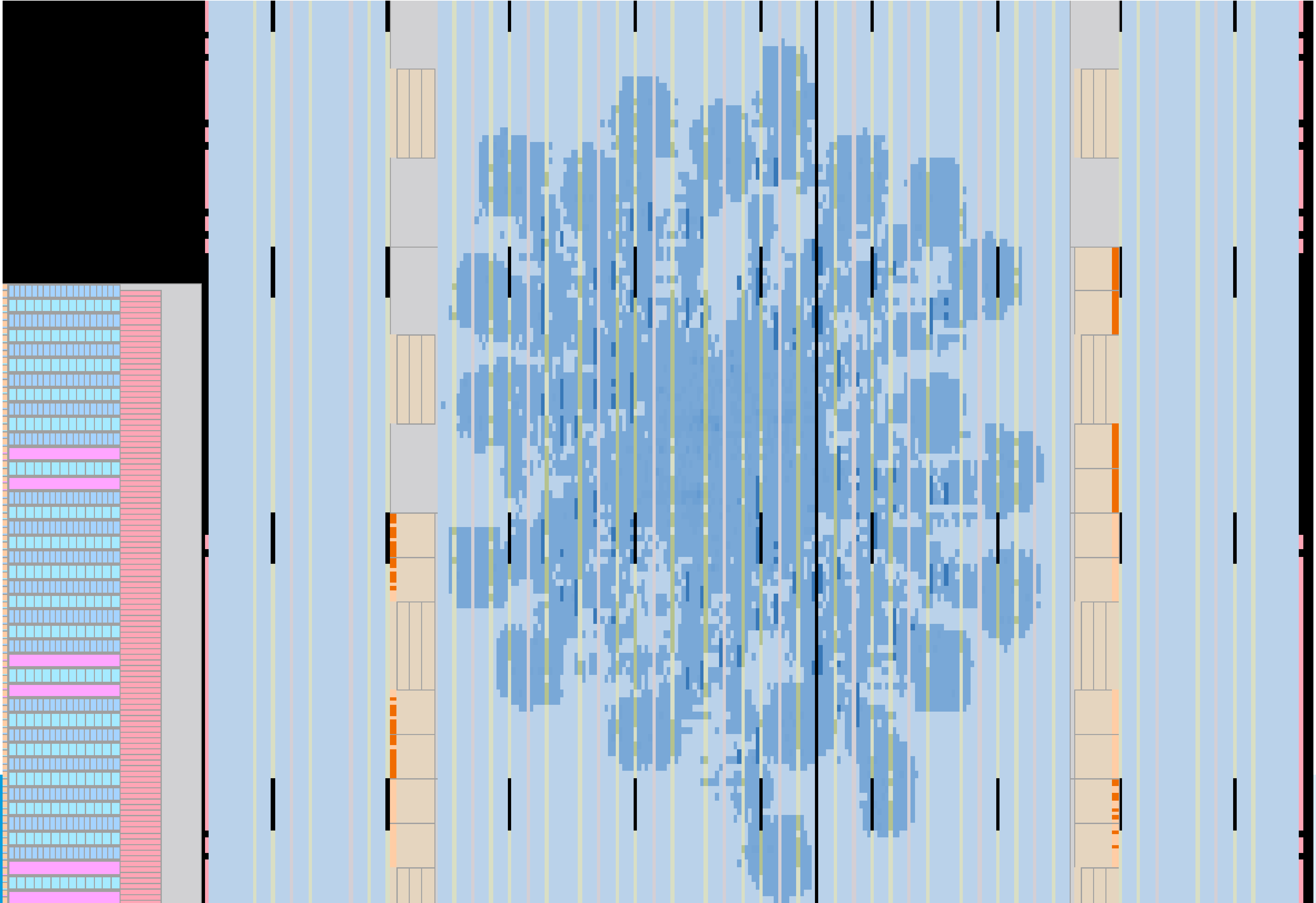
valid

done

Pipelined Count Connected Core



fullPipeline On Chip



The Numbers

- 2.3 * 10^16 bottoms (half due to deduplication)
- 5.74 * 10^18 total permutations
- At 100% Efficiency:
  - 1 CCC processes 0.5 permutations/cycle.
  - 280 CCC: ~140 permutations/cycle
    - = 35’000’000’000 permutations/s at 250MHz
    - = 46000 FPGA Hours
    - Budget 50000 FPGA Hours

Questions to you

- NDRangeKernel __global_id() ordered?
- Best job size?
- Submitting large (10’000) number of jobs?