Custom Tailored FPGA Boson Sampling

Gregory Morse [1]
morse@inf.elte.hu

Tamás Kozsik [1]
kto@elte.hu

Péter Rakyta [2]
peter.rakyta@ttk.elte.hu

[1] Department of Programming Languages and Compilers,

[2] Department of Physics of Complex Systems, Eötvös Loránd Tudományegyetem/University (ELTE), Budapest, Hungary

Project Collaboration

ELTE EÖTVÖS LORÁND
TUDOMÁNYEGYETEM

Eötvös Loránd University
Faculty of Informatics

Eötvös Loránd University
Faculty of Science

QNL Quantum Information
National Laboratory
HUNGARY

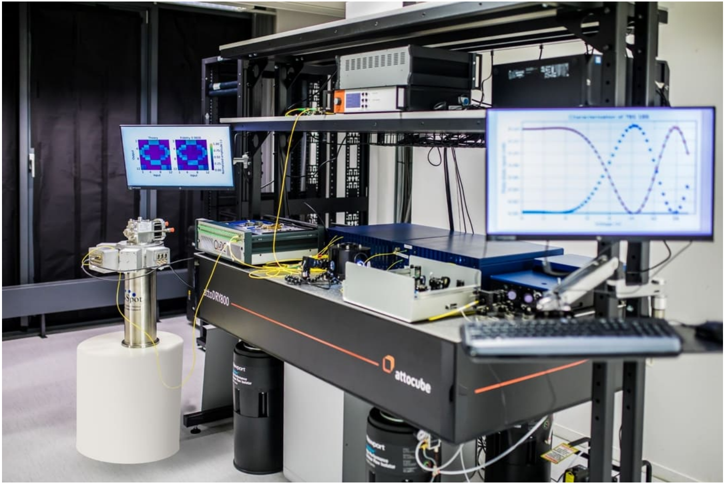
WIGNER

Introduction

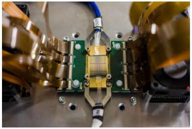
- Computing the permanent of a matrix finds an important application in the context of boson sampling
- BB/FG permanent formula with a reflected binary Gray code $\mathcal{O}(n.2^{n-1})$
- Run it in parallel on 4 SLRs (Super Logic Regions) $\mathcal{O}(n.2^{n-3})$
- Up to 40x40 matrix permanents @ 280MHz
- Dual FPGA - specialized kernel with twice as fast operation
- Generalize to repeated rows/columns up to 40 photons @ 240MHz

Keywords: Boson Sampling, Matrix Permanent, FPGA, dataflow, Repeated Row and Column Permanent, Reflected N-ary Gray code

Real World Boson Sampling Setup



QUANTUM



- QuiX Photonic Quantum Computer

Optimally Efficient Classical Permanent Algorithms

- BB/FG formula:

$$\mathrm{perm}(A) = \frac{1}{2^{m-1}} \sum_{\delta} \left(\prod_{k=1}^{m} \delta_k \right) \prod_{j=1}^{m} \sum_{i=1}^{m} \delta_i a_{i,j}, \tag{1}$$

where $\boldsymbol{A}$ is an $m \times m$ square matrix describing the interferometer and $\boldsymbol{\delta}$ is a binary Gray code. Adaptable to sub-computations of repeated-row rectangular permanents.

- With independent repeated rows and columns:

$$\mathrm{perm}(\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{N}) = \frac{1}{2^{n-1}} \sum_{\boldsymbol{\Delta}} \left(\prod_{k=1}^{m} (-1)^{\Delta_k} \binom{M_k}{\Delta_k} \right) \prod_{j=1}^{m} \left(\sum_{k=1}^{m} (M_k - 2\Delta_k)\, a_{k,j} \right)^{N_j} \tag{2}$$

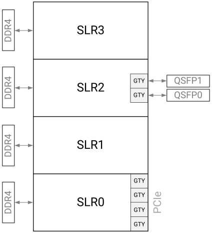
$\boldsymbol{M}$ and $\boldsymbol{N}$ are the row and column multiplicities respectively such that the photon count $n = \sum M_i = \sum N_j$ and $\boldsymbol{\Delta}$ is the n-ary Gray code, required for efficient computation.

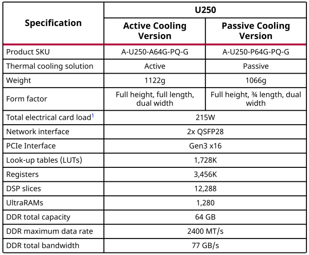
Xilinx Alveo U250 FPGA and Maxeler MaxCompiler 2021.1

PCIExpress clock operates at 250MHz, initialization limitation 16nm Ultrascale+ architecture, Vivado compiler supporting up to 500MHz

Active Option

Figure 3: **Floorplan of the XCU250 Device**





Specification	U250	
	Active Cooling Version	**Passive Cooling Version**
Product SKU	A-U250-A64G-PQ-G	A-U250-P64G-PQ-G
Thermal cooling solution	Active	Passive
Weight	1122g	1066g
Form factor	Full height, full length, dual width	Full height, ¾ length, dual width
Total electrical card load[1]	215W	
Network interface	2x QSFP28	
PCIe Interface	Gen3 x16	
Look-up tables (LUTs)	1,728K	
Registers	3,456K	
DSP slices	12,288	
UltraRAMs	1,280	
DDR total capacity	64 GB	
DDR maximum data rate	2400 MT/s	
DDR total bandwidth	77 GB/s	

MAXELER™ a groq company

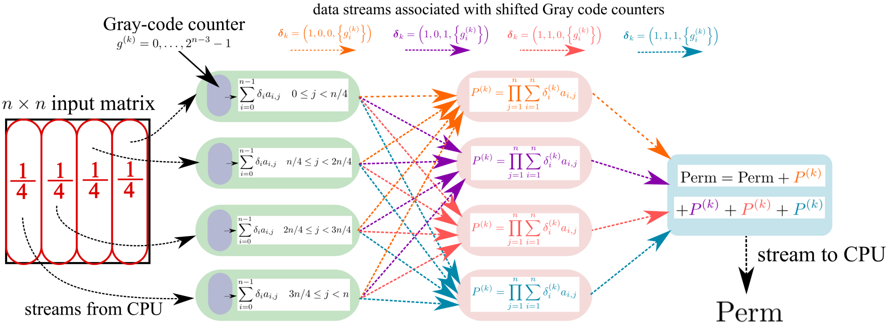
groq™

Accuracy of single and double precision floating vs. normalized fixed point

- We developed a GNU MPFR (multi-precision floating point reliability library) infinite precision wrapper to check accuracy with realistic data as part of **Piquasso Boost** extension to piquasso
- Single/double precision CPU variants use 4M+2A complex multiplication $(a+bi)(c+di)=(ac-bd)+(bc+ad)i$
- Infinite Precision and FPGA variants use Knuth 3M+5A complex multiplication $x=c(a+b),(x-b(c+d))+(x+a(d-c))i$
- Complex number normalization computed by worst-case column sums using a Euclidean vector inspired technique, keeping all computations $-1\leq a,b,|a+bi|\leq 1$
- Outer sum of BB/FG precise number of bits determined $\frac{\max(\sum_{k=0}^{\lfloor (n-1)/2\rfloor}(2(2k+2)-n)^n\binom{n-1}{2k+1},\sum_{k=0}^{\lfloor (n-1)/2\rfloor}(2(2k+1)-n)^n\binom{n-1}{2k})}{n^n}$ for $n=40$, then ± 27 is maximum partial sum requiring 6 integer bits (including sign)

Design

- Time complexity: $\mathcal{O}(2^{n-1-k})$ where $k = 2$ for single, $k = 3$ for dual
- Area: $\mathcal{O}(n^2)$ dominated by matrix storage in FFs (flip-flops)
- Multiplication area based on a product tree with Karatsuba rectangular tiling to match the 18x25 signed DSP multipliers of the FPGA $\mathcal{O}(b \log b)$ where tree depth is 6: 20 (b=64-bit) -> 10 (93-bit) -> 5 (110-bit) -> 2 (127-bit) -> 1 (127-bit) -> 1 (127-bit)



data streams associated with shifted Gray code counters

Gray-code counter
$g^{(k)} = 0, \ldots, 2^{n-3} - 1$

$\boldsymbol{\delta}_k = \left(1, 0, 0, \left\{g_i^{(k)}\right\}\right)$

$\boldsymbol{\delta}_k = \left(1, 0, 1, \left\{g_i^{(k)}\right\}\right)$

$\boldsymbol{\delta}_k = \left(1, 1, 0, \left\{g_i^{(k)}\right\}\right)$

$\boldsymbol{\delta}_k = \left(1, 1, 1, \left\{g_i^{(k)}\right\}\right)$

$n \times n$ input matrix

$\frac{1}{4}$ $\frac{1}{4}$ $\frac{1}{4}$ $\frac{1}{4}$

streams from CPU

$\sum_{i=0}^{n-1} \delta_i a_{i,j} \quad 0 \leq j < n/4$

$\sum_{i=0}^{n-1} \delta_i a_{i,j} \quad n/4 \leq j < 2n/4$

$\sum_{i=0}^{n-1} \delta_i a_{i,j} \quad 2n/4 \leq j < 3n/4$

$\sum_{i=0}^{n-1} \delta_i a_{i,j} \quad 3n/4 \leq j < n$

$P^{(k)} = \prod_{j=1}^{n} \sum_{i=1}^{n} \delta_i^{(k)} a_{i,j}$

$P^{(k)} = \prod_{j=1}^{n} \sum_{i=1}^{n} \delta_i^{(k)} a_{i,j}$

$P^{(k)} = \prod_{j=1}^{n} \sum_{i=1}^{n} \delta_i^{(k)} a_{i,j}$

$P^{(k)} = \prod_{j=1}^{n} \sum_{i=1}^{n} \delta_i^{(k)} a_{i,j}$

$\mathrm{Perm} = \mathrm{Perm} + P^{(k)}$
$+ P^{(k)} + P^{(k)} + P^{(k)}$

stream to CPU

Perm

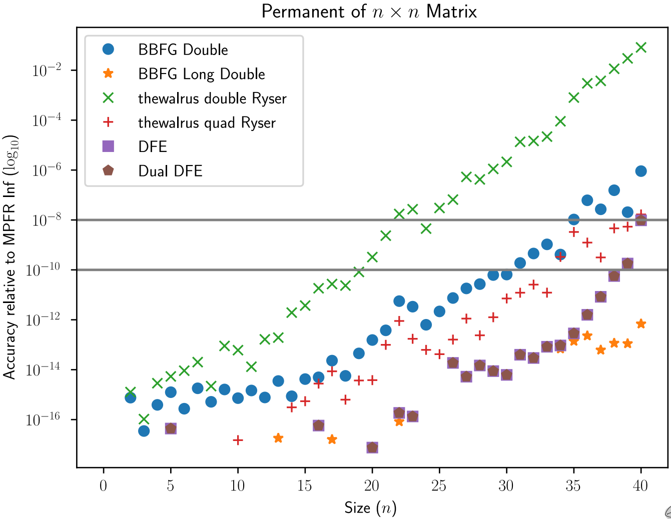
Repeated Row/Column Design

- Reflected N-ary Gray code (using direction encoding (DE))
- Binomial coefficients computed with a loop length=9, division by "magic number" multiplication
    - Incremental update by Gray code decreasing $b_i = \frac{b_{i-1}\times k}{n-k+1}$ otherwise $b_i = \frac{b_{i-1}\times(n-k)}{k+1}$
- Staggering the Gray code at even intervals

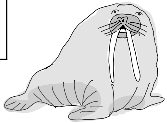
Counter Chain	DEGC	Gray Code (GC)	b_i
(0, 0, 0)	**(0, 0, 0)**	**(0, 0, 0)**	**1**
(1, 0, 0)	(1, 0, 0)	(1, 0, 0)	1
(0, 1, 0)	**(2, 1, 0)**	**(1, 1, 0)**	**2**
(1, 1, 0)	(3, 1, 0)	(0, 1, 0)	2
(0, 2, 0)	**(0, 2, 0)**	**(0, 2, 0)**	**1**
(1, 2, 0)	(1, 2, 0)	(1, 2, 0)	1
(0, 0, 1)	**(2, 3, 1)**	**(1, 2, 1)**	**2**
(1, 0, 1)	(3, 3, 1)	(0, 2, 1)	2
(0, 1, 1)	**(0, 4, 1)**	**(0, 1, 1)**	**4**
(1, 1, 1)	(1, 4, 1)	(1, 1, 1)	4
(0, 2, 1)	**(2, 5, 1)**	**(1, 0, 1)**	**2**
(1, 2, 1)	(3, 5, 1)	(0, 0, 1)	2
(0, 0, 2)	**(0, 0, 2)**	**(0, 0, 2)**	**1**
(1, 0, 2)	(1, 0, 2)	(1, 0, 2)	1
(0, 1, 2)	**(2, 1, 2)**	**(1, 1, 2)**	**2**
(1, 1, 2)	(3, 1, 2)	(0, 1, 2)	2
(0, 2, 2)	**(0, 2, 2)**	**(0, 2, 2)**	**1**
(1, 2, 2)	(1, 2, 2)	(1, 2, 2)	1

Table: Example for Non-Anchor Row Multiplicities (1, 2, 2) with 3x3x2=18 values

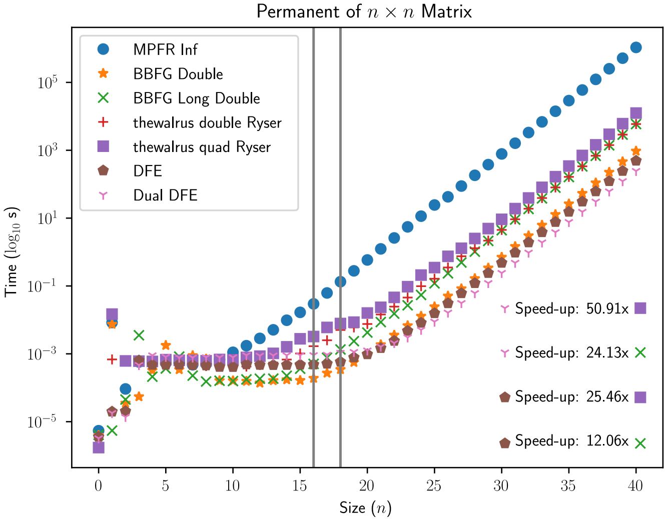
Accuracy CPU vs Digital Front End (DFE) to FPGA



- Vs. The Walrus 0.17 C++ Ryser

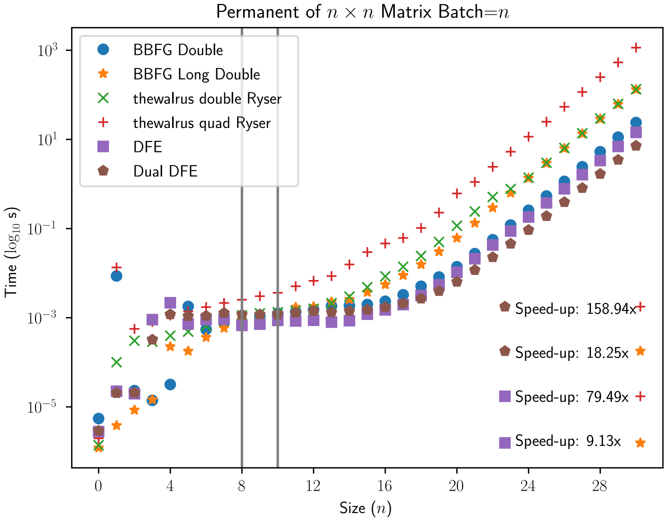


Performance of 280MHz single and dual array



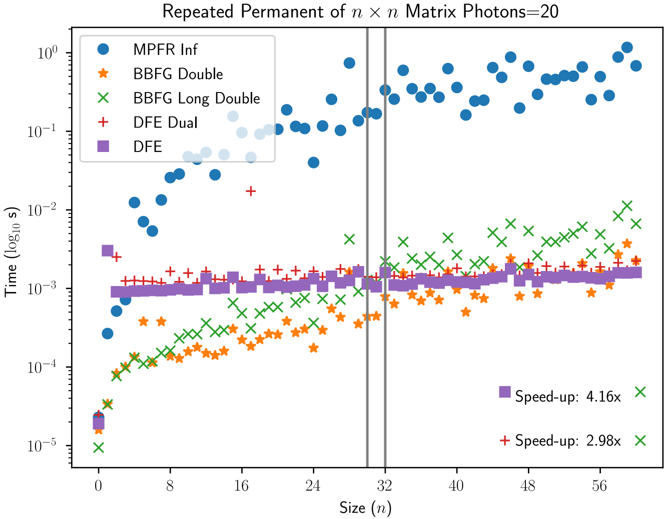
- Initialization delay crossover threshold for single and dual marked based on precise long double calculators

FPGA Batching Advantage



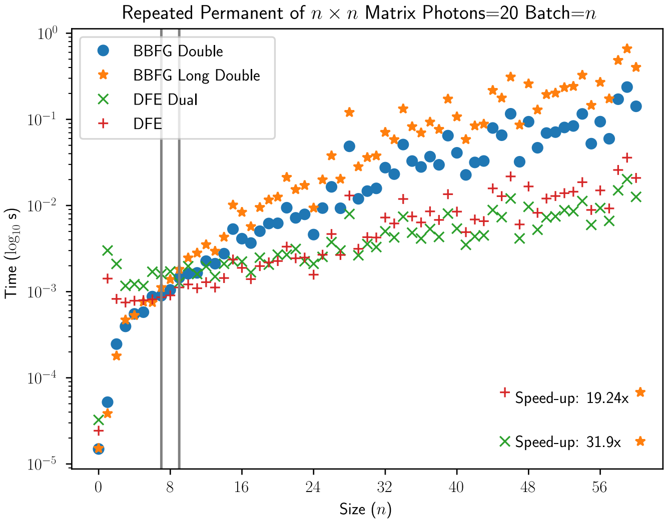
- Batches reduce the cross-over threshold
- Kernels designed for automatic control signal resetting/-counter wrapping

Performance of 240MHz for repeated permanents



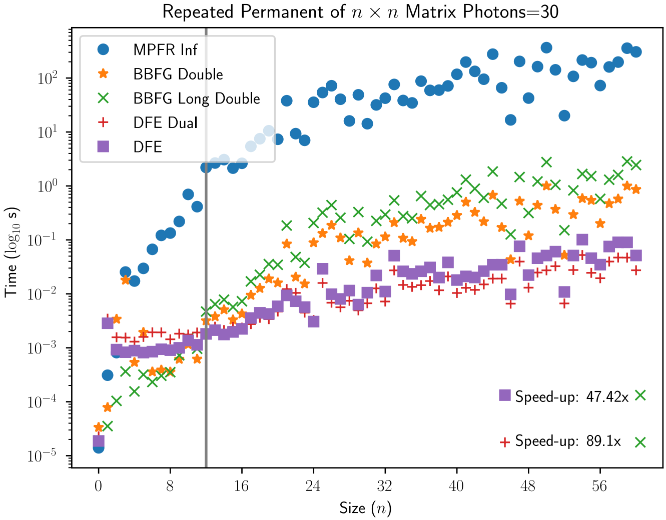
- Significant initialization delay and high crossover threshold
- At 30 photons, this no longer occurs

FPGA Batching Advantage for repeated permanents



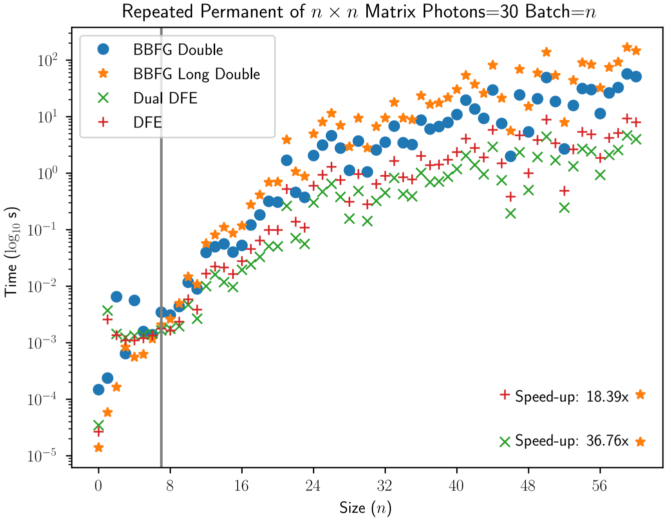
- Batch size n based on realistic use case Faster classical Boson Sampling (Clifford and Clifford 2018)
- Ouput state/row multiplicity fixed over batch of changing input states/column multipicities

Performance of 240MHz for repeated permanents



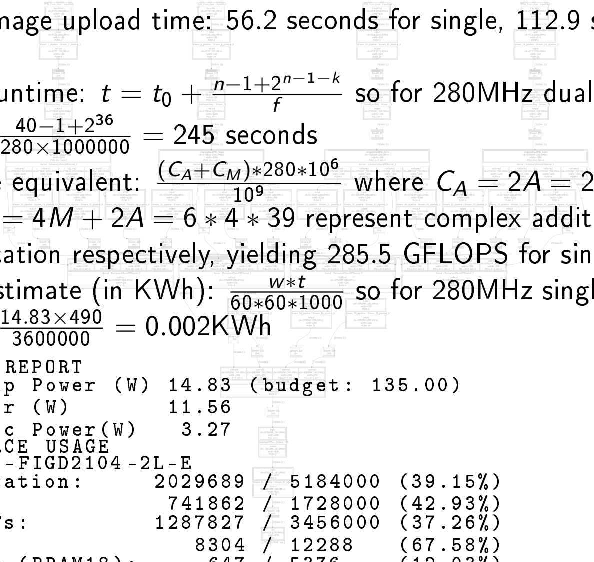
- DFE variants have clearly become best performing

FPGA Batching Advantage for repeated permanents



- The benefit is further demonstrated

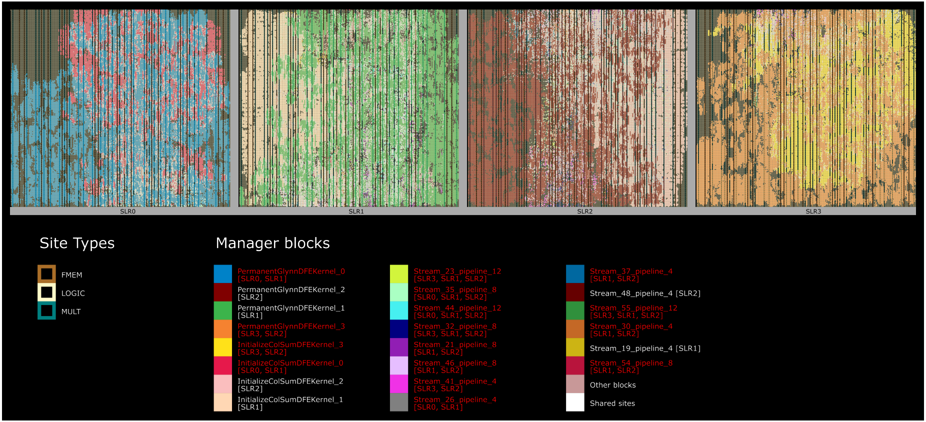
Time, Area and Power Analysis



- FPGA image upload time: 56.2 seconds for single, 112.9 seconds for dual
- Actual runtime: $t = t_0 + \frac{n-1+2^{n-1-k}}{f}$ so for 280MHz dual on a 40x40 matrix: $\frac{40-1+2^{36}}{280\times1000000} = 245$ seconds
- Effective equivalent: $\frac{(C_A+C_M)*280*10^6}{10^9}$ where $C_A = 2A = 2*(40+4)$ and $C_M = 4M + 2A = 6*4*39$ represent complex addition and multiplication respectively, yielding 285.5 GFLOPS for single mode
- Power estimate (in KWh): $\frac{w*t}{60*60*1000}$ so for 280MHz single on a 40x40 matrix: $\frac{14.83\times490}{3600000} = 0.002$KWh

```
FINAL POWER REPORT
Total On-Chip Power (W) 14.83 (budget: 135.00)
Dynamic Power (W)       11.56
Device Static Power(W)   3.27
FINAL RESOURCE USAGE
FPGA: xcU250-FIGD2104-2L-E
Logic utilization:     2029689 / 5184000 (39.15%)
  LUTs:                 741862 / 1728000 (42.93%)
  Primary FFs:         1287827 / 3456000 (37.26%)
DSP blocks:               8304 / 12288   (67.58%)
Block memory (BRAM18):     647 / 5376    (12.03%)
Block memory (URAM):       126 / 1280    (9.84%)
```

Aerial View


SLR0
SLR1
SLR2
SLR3
Site Types
FMEM
LOGIC
MULT
Manager blocks
PermanentGlynnDFEKernel_0 [SLR0, SLR1]
PermanentGlynnDFEKernel_2 [SLR2]
PermanentGlynnDFEKernel_1 [SLR1]
PermanentGlynnDFEKernel_3 [SLR3, SLR2]
InitializeColSumDFEKernel_3 [SLR3, SLR2]
InitializeColSumDFEKernel_0 [SLR0, SLR1]
InitializeColSumDFEKernel_2 [SLR2]
InitializeColSumDFEKernel_1 [SLR1]
Stream_23_pipeline_12 [SLR3, SLR1, SLR2]
Stream_35_pipeline_8 [SLR0, SLR1, SLR2]
Stream_44_pipeline_12 [SLR0, SLR1, SLR2]
Stream_32_pipeline_8 [SLR3, SLR1, SLR2]
Stream_21_pipeline_8 [SLR1, SLR2]
Stream_46_pipeline_8 [SLR1, SLR2]
Stream_41_pipeline_4 [SLR3, SLR2]
Stream_26_pipeline_4 [SLR0, SLR1]
Stream_37_pipeline_4 [SLR1, SLR2]
Stream_48_pipeline_4 [SLR2]
Stream_55_pipeline_12 [SLR3, SLR1, SLR2]
Stream_30_pipeline_4 [SLR1, SLR2]
Stream_19_pipeline_4 [SLR1]
Stream_54_pipeline_8 [SLR1, SLR2]
Other blocks
Shared sites

- After careful pipelining and reducing fanout, streams become the largest routing issue

Conclusion and Future Research

- FPGAs are competitive for multiplication intensive implementations
- State-of-the-art algorithms are often needed for maximizing resources, cannot rely on default implementations
- Computation of the Loop Hafnian using GroqCard in concert with data preparation on the FPGA