The ATI HD5870 Matrix Factory

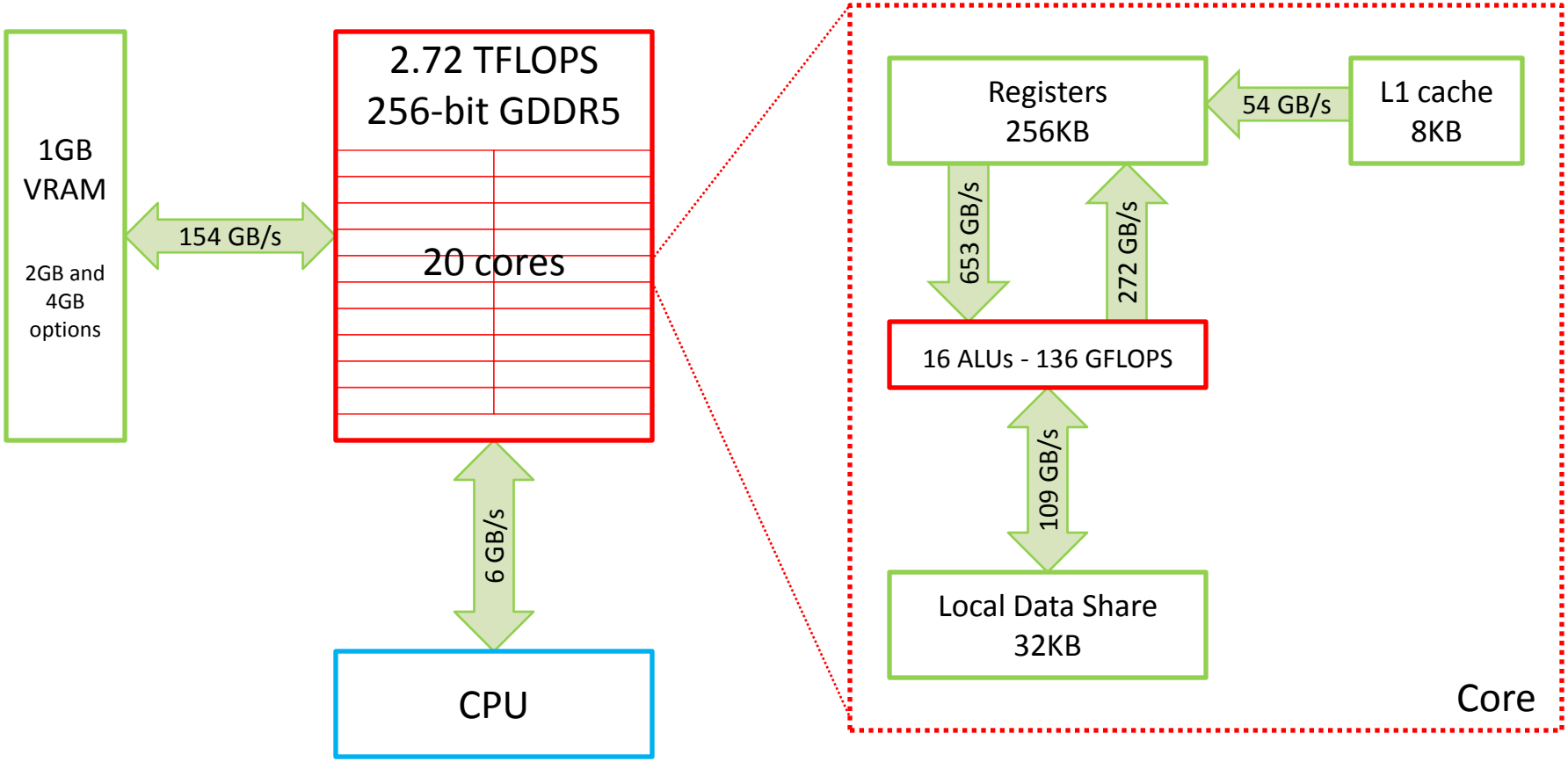
Capacity 1.75 TFLOPS

Jawed Ashraf

OPLib

- **SGEMM and SGEMV as building blocks**
  - **only requires single precision**
- **Data persists on the graphics card**
  - **hence “matrix factory”**
- **Packed matrices and vectors**
  - **amortise kernel launch overheads**
  - **benefits smaller matrices, N=500 to 1000**

Cypress GPU


1GB
VRAM
2GB and
4GB
options
154 GB/s
2.72 TFLOPS
256-bit GDDR5
20 cores
6 GB/s
CPU
Registers
256KB
54 GB/s
L1 cache
8KB
653 GB/s
272 GB/s
16 ALUs - 136 GFLOPS
109 GB/s
Local Data Share
32KB
Core

Basic Parameters

- **Chip**
  - **2.72 TFLOPS**
  - **154 GB/s**
  - **0.057 bytes:FLOP**
- **Core – 850MHz**
  - **136 GFLOPS**
  - **762 GB/s load**
  - **5.6 bytes:FLOP**
  - **MAD requires 6 bytes:FLOP**
- **Register file – 256KB**
  - **6 hardware threads**
  - **160 floats per work item**
- **ALU per cycle:**
  - **5 MADs = 10 FLOP**
  - **L1 4 bytes load**
  - **LDS 8 bytes load/store**
  - **registers load 48 bytes**
  - **registers store 20 bytes**

OPCAL

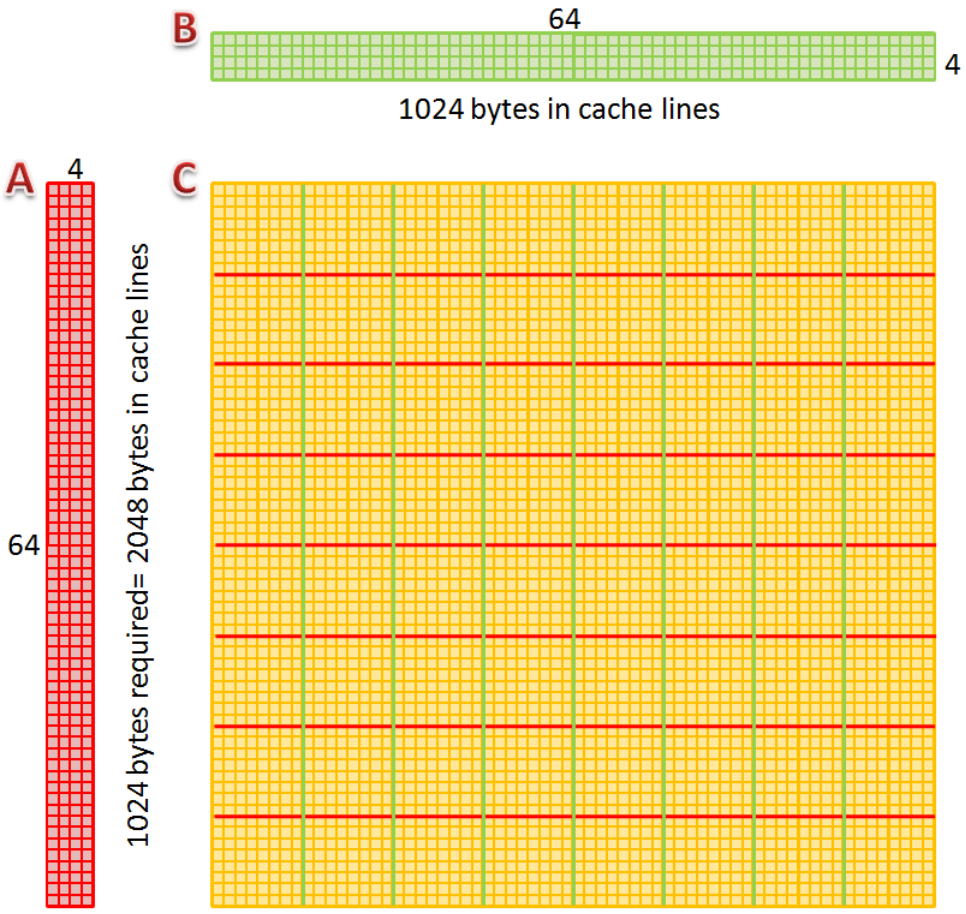
- OpenCL was original target
  - started with AMD's OpenCL just after Stream SDK 2.0 was released
  - disaster zone
    - major features missing
    - terrible compilation quality
  - appears to be improving quickly
    - image support recently added
- Currently using CAL
  - eventually port OPCAL to OpenCL?
    - in 6 months or 1 year?
    - would sacrifice some performance
    - would make library more accessible/extensible by others

CAL with IL

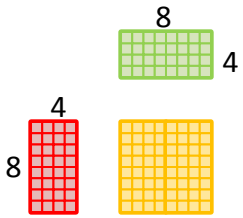
- **CAL is AMD's underlying GPU technology**
- **IL is like D3D10's assembly language**
- **Generally hellish – it's the 1970s!**
- **Features such as functions and macros help**
- **SKA (or GPUSA) is a key tool**
  - **disassembly is essential for optimisation and insightful for debugging**
  - **aspects of performance are deterministic**

Brute SGEMM

Workgroup computes 64x64 block in C:



64 work items each compute 8x8:



- 156 floats (or equivalent)
- 6 hardware threads
- 16 TEX – 256 bytes
- 256 MADs – 512 FLOP
- 60 cycles
- nominally 2.32 TFLOPS
- requires L1 = 4.3 bytes:cycle
- achieves 1.45 TFLOPS – 62%

Zero-Padding

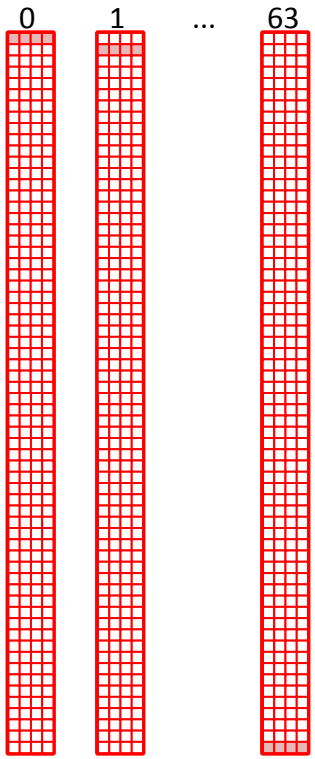
- **Packing of floats into float4s requires padding**
- **Running sum is in strides of 4 or 8 floats**
- **Compute shader writes prefer 64-element alignment**
- **Matrix writes have to be 256-element aligned if the matrix is to be used later as input**
- **Host application doesn't notice padding**
  - **$0 < \{M, K\} < 16385$**
  - **$0 < N < 65537$**

LDS – Killing two birds with one stone

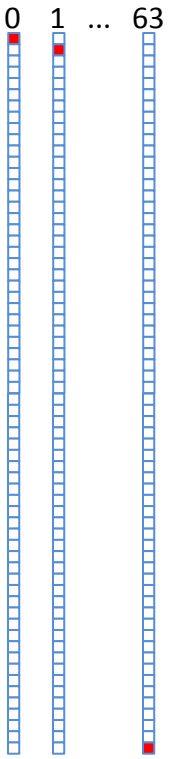
- **Use LDS to reduce TEX count**
  - **reduces required byte:cycle**
  - **reduces register allocation, hiding more load latency**
- **Transpose A through LDS**
  - **64 work items each fetch a single float4 from A**
  - **each work item sends 1st float from float4 to LDS**
  - **each work item loads 8 floats from LDS and performs 64 MADs**
  - **repeat for 2nd, 3rd and 4th floats from A's float4**
- **LDS acts as a column selector for A**
  - **single column minimises load/store address registers**
  - **single column avoids additional address generation arithmetic**

Transposing A through LDS

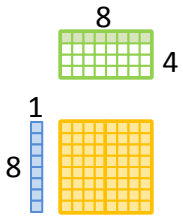
1. Each work item fetches a single float4 from A:



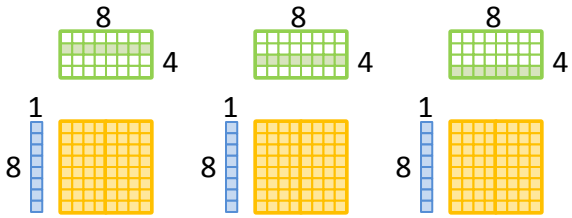
2. Each work item writes a float to LDS:



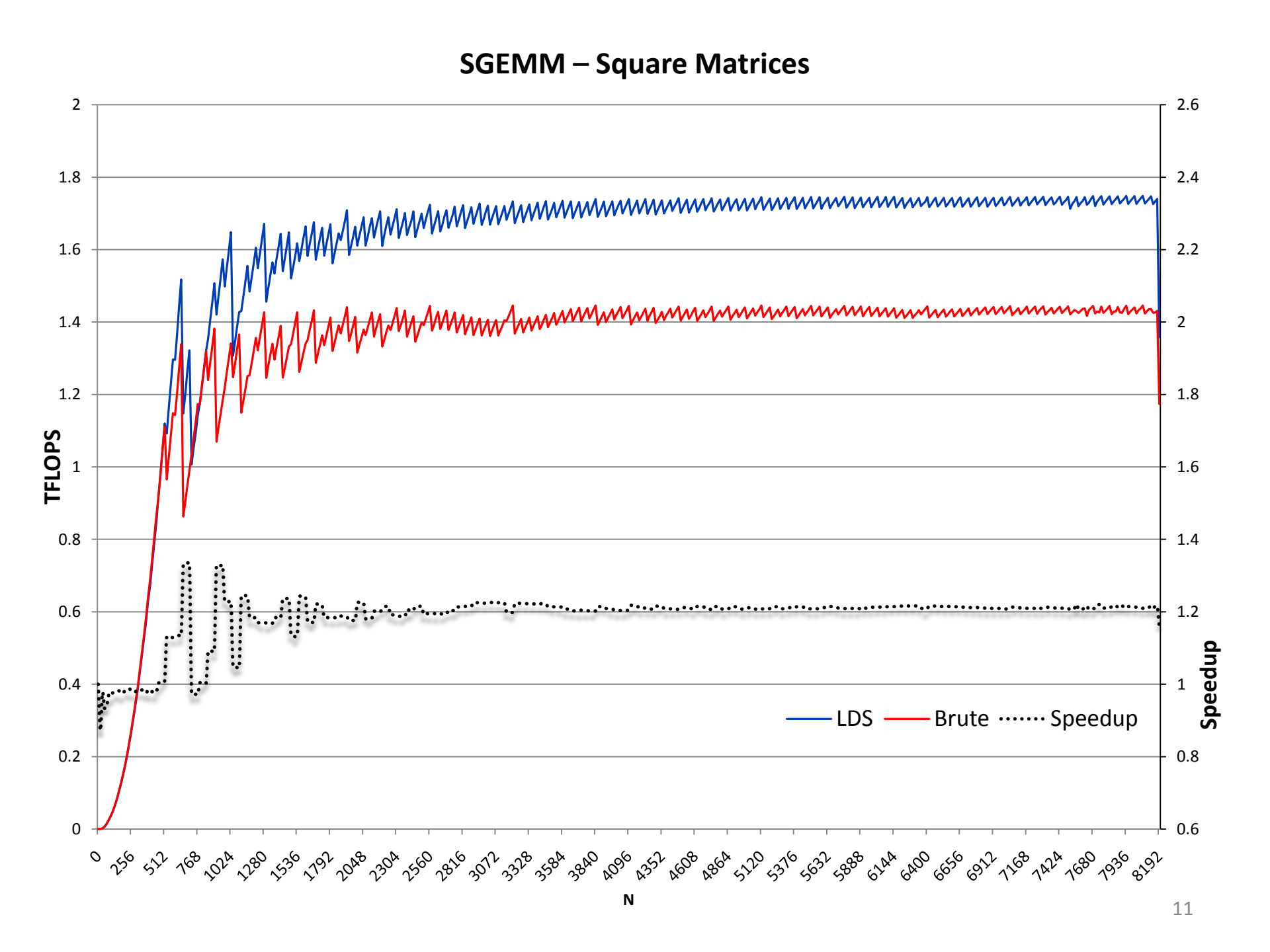
3. Each work item computes 8x8:



4. Repeat steps 2 and 3 for 2nd, 3rd and 4th floats:

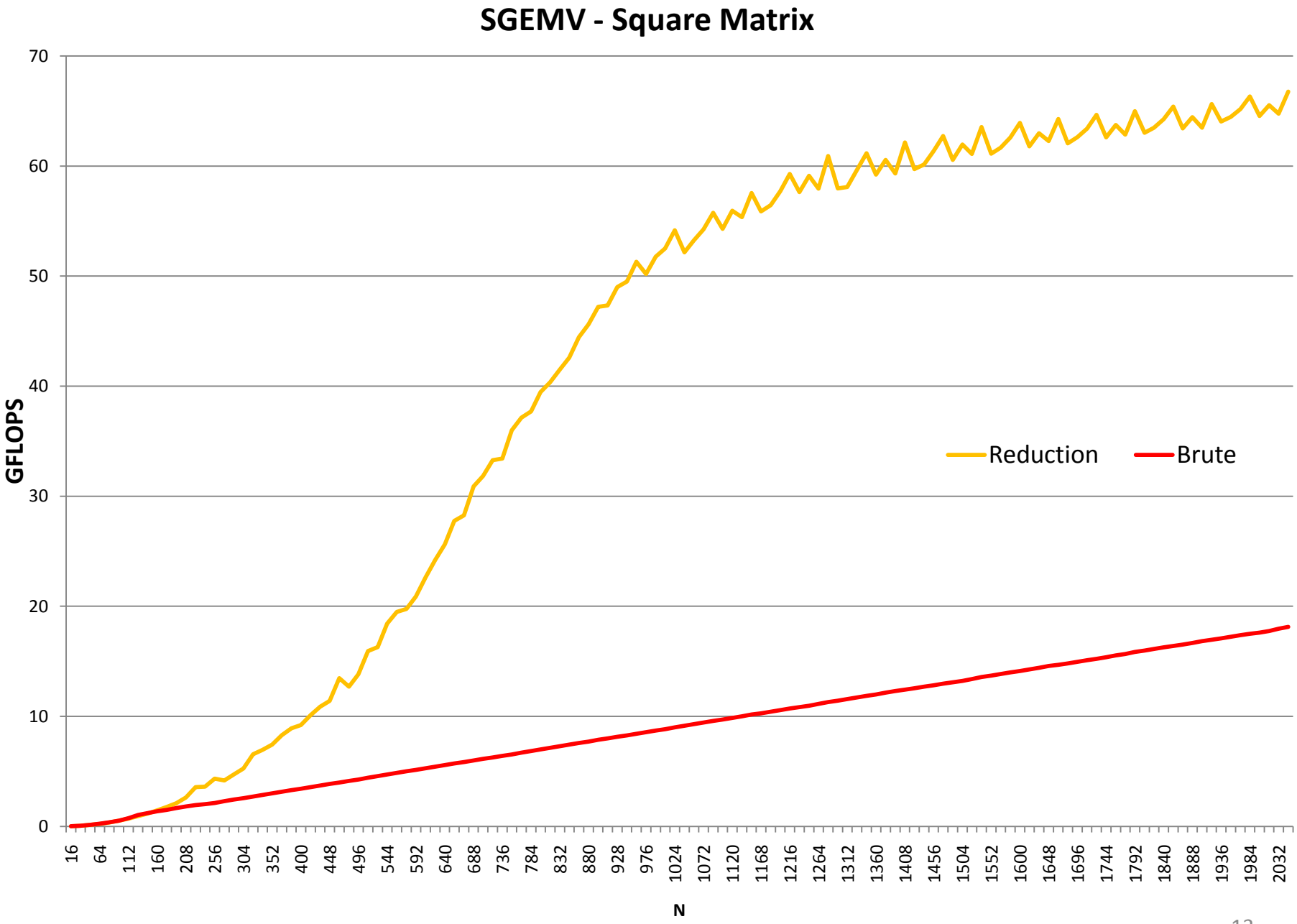


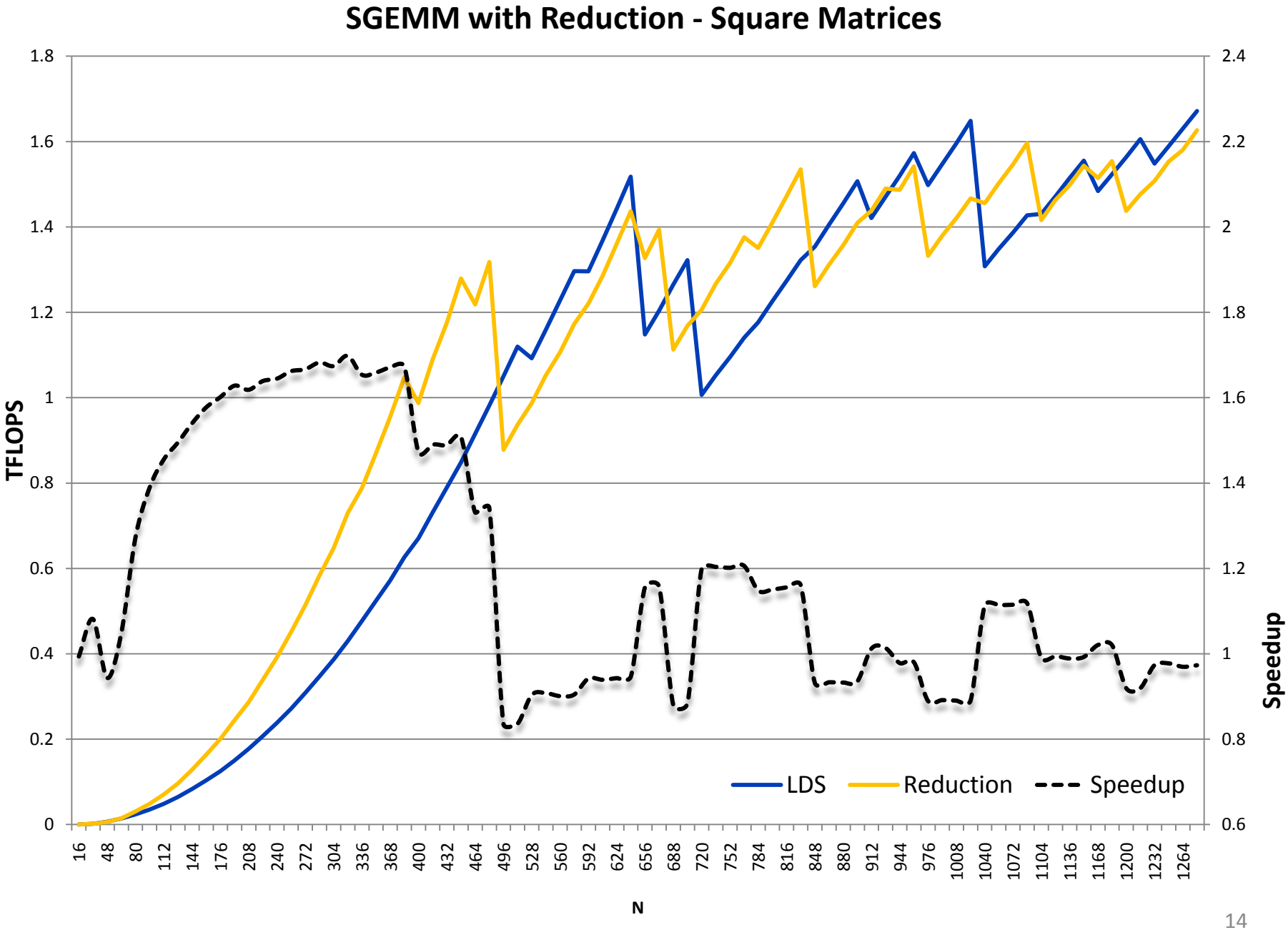
- 132 floats (or equivalent)
- 7 hardware threads
- 9 TEX – 144 bytes
- 256 MADs – 512 FLOP
- 75 cycles
- nominally 1.86 TFLOPS
- requires L1 = 1.9 bytes:cycle
- achieves 1.75 TFLOPS – 94%


SGEMM – Square Matrices
TFLOPS
Speedup
N
LDS
Brute
Speedup
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
0
256
512
768
1024
1280
1536
1792
2048
2304
2560
2816
3072
3328
3584
3840
4096
4352
4608
4864
5120
5376
5632
5888
6144
6400
6656
6912
7168
7424
7680
7936
8192

SGEMV - Reduction

- **Pathetic arithmetic intensity**
- **GPU hardly wakes up before going back to sleep**
- **Try increased size of execution domain**
  - **2 or more work items compute each result**
  - **reduction used to sum the 2 or more partial results**
  - **LDS is perfect for reduction**
  - **increases cache hit rate**
  - **small increase in average accuracy**
- **Using 4-way reduction**
  - **16 masters and 48 slaves**


SGEMV - Square Matrix
GFLOPS
70
60
50
40
30
20
10
0
Reduction
Brute
16
64
112
160
208
256
304
352
400
448
496
544
592
640
688
736
784
832
880
928
976
1024
1072
1120
1168
1216
1264
1312
1360
1408
1456
1504
1552
1600
1648
1696
1744
1792
1840
1888
1936
1984
2032
N


SGEMM with Reduction - Square Matrices
TFLOPS
Speedup
N
LDS
Reduction
Speedup
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
16
48
80
112
144
176
208
240
272
304
336
368
400
432
464
496
528
560
592
624
656
688
720
752
784
816
848
880
912
944
976
1008
1040
1072
1104
1136
1168
1200
1232
1264

Future

- **Integration into OPLib**
  - **C# interface prepared**
  - **requires additional glue kernels – not performance-critical**
- **More performance optimisations**
  - **packed matrix (SGEMM4) performance currently in the range of 450-1340 GFLOPS for N=512**
  - **evaluate layout of matrices in 2D buffers on GPU**
  - **cache behaviour versus quantity of packed matrices**

Contact

Jawed Ashraf

OPCAL@cupidity.f9.co.uk

OPLib

www.albanese.co.uk