Sparse Matrix-Vector Multiply on the Keystone II Digital Signal Processor

Yang Gao, Fan Zhang and Dr. Jason D. Bakos

2014 IEEE High Performance Extreme Computing Conference(HPEC '14)

UNIVERSITY OF SOUTH CAROLINA

Heterogeneous Computing on Embedded Platforms

- GPU/FPGA/DSP(KeyStone)
- ARM + Embedded GPUs/FPGA/DSP(KeyStone II)

	Embedded GPU			FPGA	DSP
Platforms	PowerVR SGX 544MP3	PowerVR G6430	GK20A (Kepler)	Zynq-7000	KeyStone II
	Exynos 5 Octa	Apple A7	Tegra K1(Jetson)	7Z045	TCI6638
Theoretical Peak Performance (Gflops)	51.1	115.2	@864 MHz 331	65	@1.35GHz 172.8

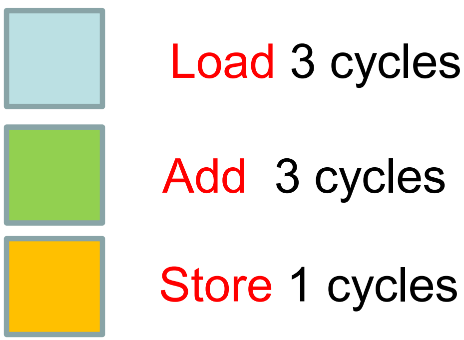
UNIVERSITY OF SOUTH CAROLINA

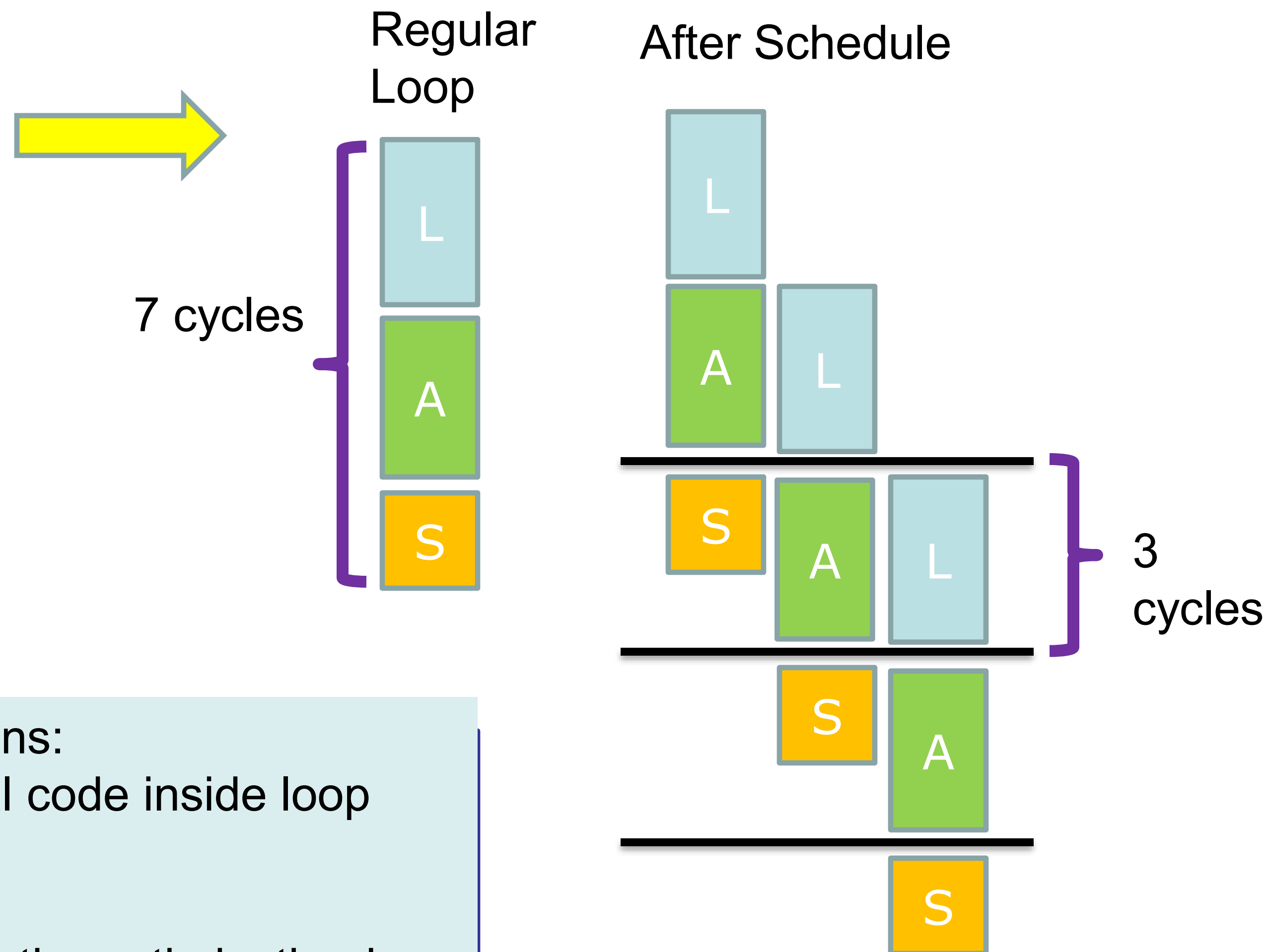
Key Features of KeyStone II

- In-order execution, static scheduled VLIW processor
- 8 symmetric VLIW cores with SIMD instructions, up to 16 flops/cycle
- No shared last level cache
- 6MB on-chip shared RAM (MSMC)
- L1D and L2 can be configured as cache, scratchpad, or both
- DMA engine for parallel loading/flushing scratchpads
- High performance, Low power consumption design

VLIW and Software Pipeline

```
for i = 0 to n
  $1 = load A[i]
  add $2,$2,$1
  A[i] = store $2
endfor
```



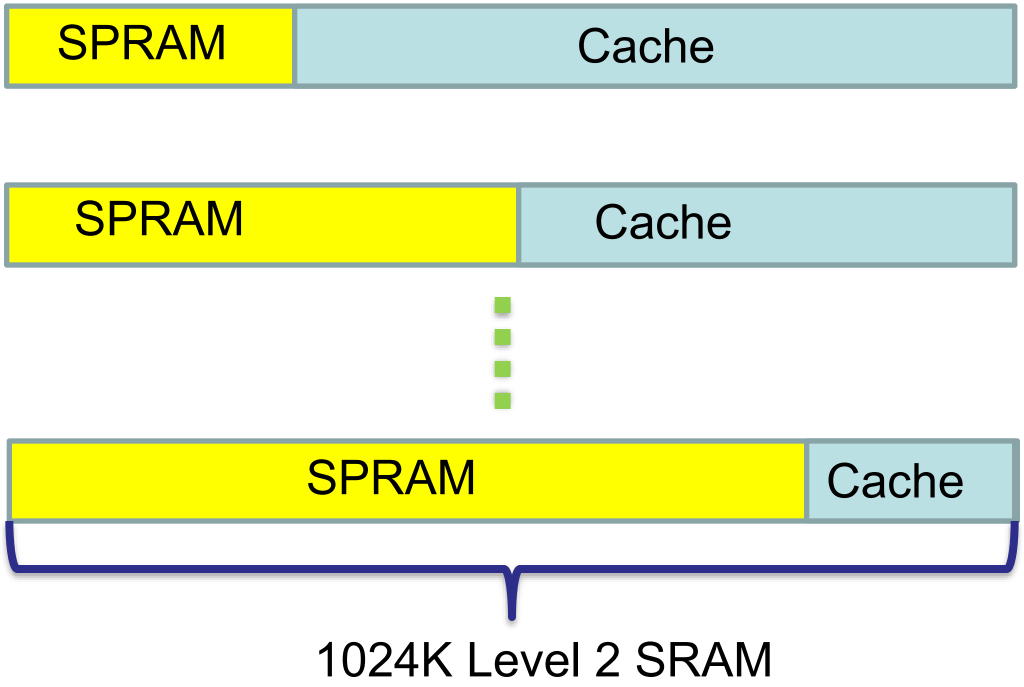


Software Pipeline Restrictions:

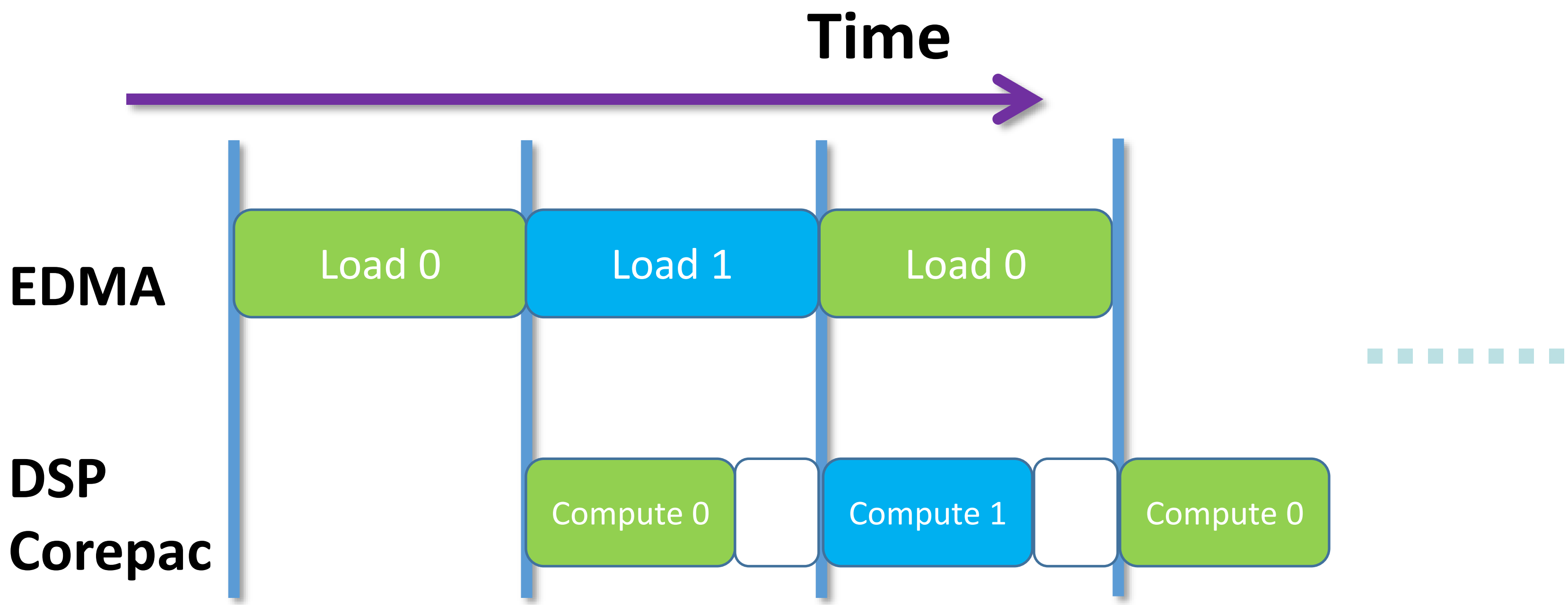
- Not friendly to conditional code inside loop body
- Register Limitations
- Rely on compiler to exert the optimization in low level

DSP Memory Hierarchy

	Size (KB)	Cache Capacity(KB)
L1P	32	0/4/8/16/32
L1D	32	0/4/8/16/32
L2	1024	0/32/64/128/ 256/512/1024
MSMC	6144	0



SPM Method and Double Buffer



Sparse Matrices

- We evaluated the Keystone II using a SpMV kernel
- Sparse Matrices can be very large but contain few non-zero elements
- Compressed formats are often used, e.g. Compressed Sparse Row (CSR)

$$\begin{pmatrix} 1 & -1 & 0 & -3 & 0 \\ -2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ -4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{pmatrix}$$

val	(1	-1	-3	-2	5	4	6	4	-4	2	7	8	-5)
col	(0	1	3	0	1	2	3	4	0	2	3	1	4)
ptr	(0	3	5	8	11	13)							

Sparse Matrix-Vector Multiply

- Code for y = **A**αx + βy

conditional execution

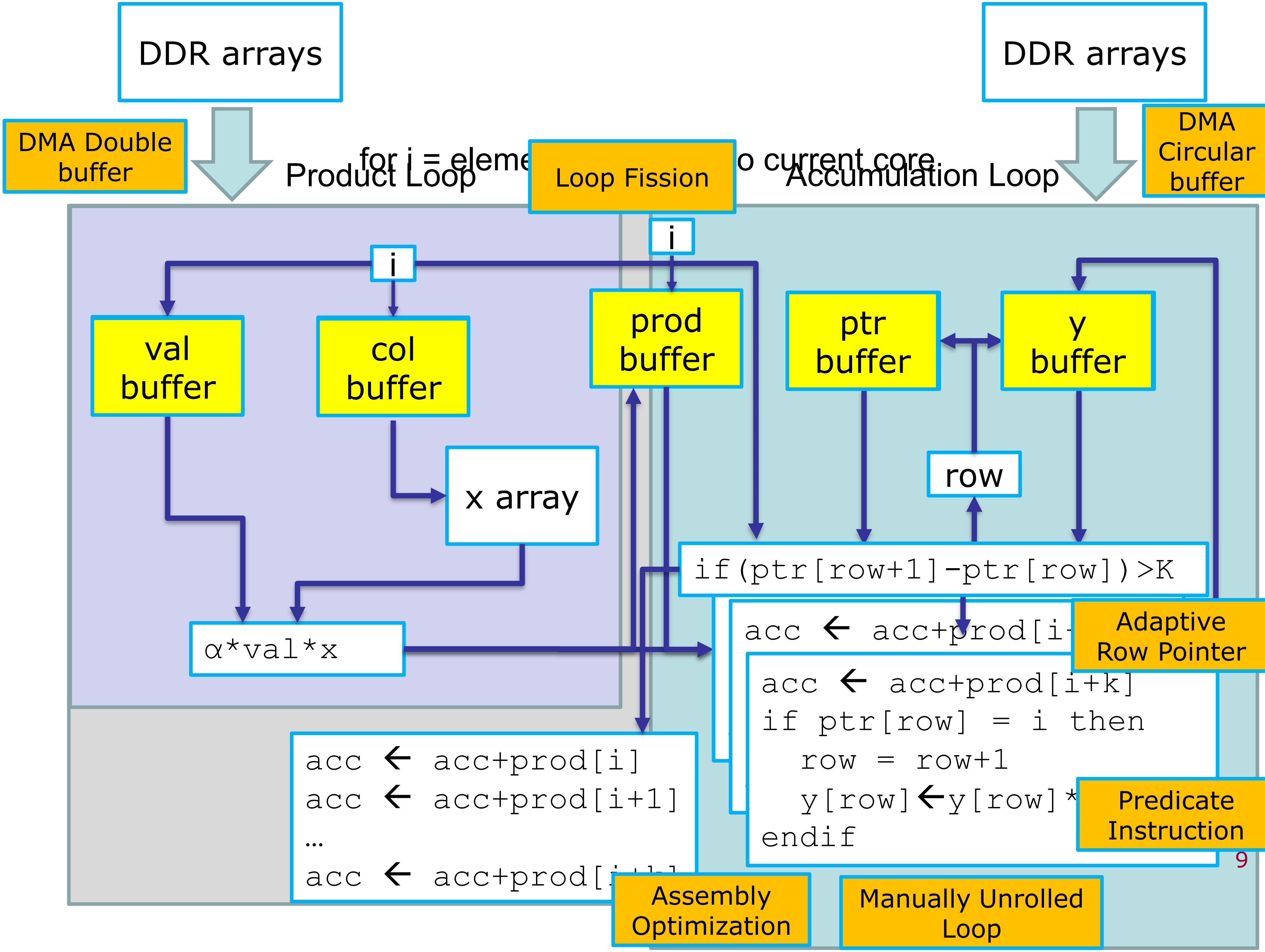
```
row = 0
for i = 0 to number_of_nonzero_elements do
   if i == ptr[row+1] then row=row+1, y[row]*=beta;
   y[row] = y[row] + alpha * A[i] * x[col[i]]
end
```

reduction

Low arithmetic intensity
(~3 flops / 24 bytes)
Memory bound kernel

indirect
indexing

UNIVERSITY OF SOUTH CAROLINA



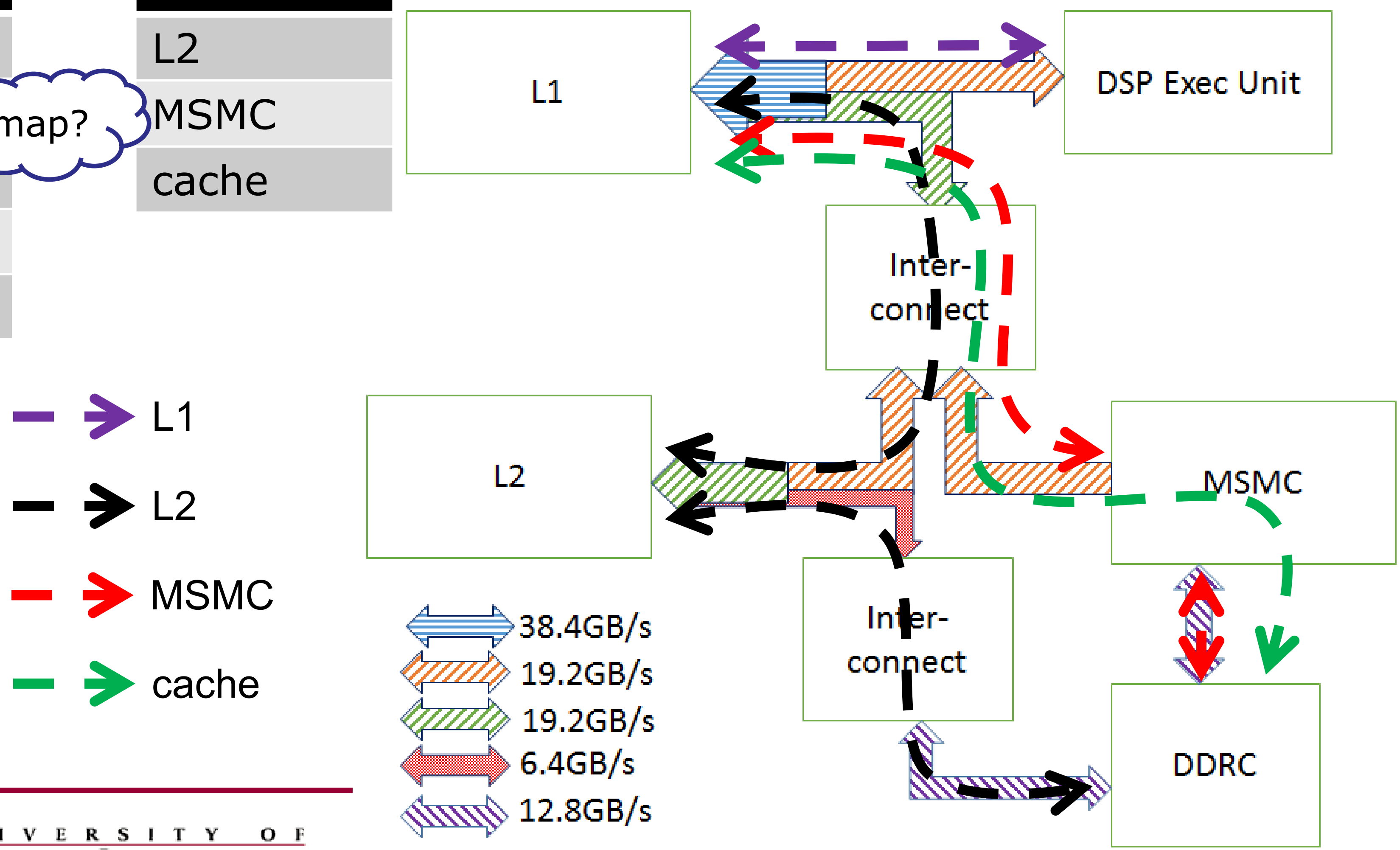
DDR arrays
DDR arrays
DMA Double buffer
for i = eleme
o current core
Product Loop
Loop Fission
Accumulation Loop
DMA Circular buffer
i
i
val buffer
col buffer
prod buffer
ptr buffer
y buffer
x array
row
if(ptr[row+1]-ptr[row])>K
α*val*x
acc ← acc+prod[i-
Adaptive Row Pointer
acc ← acc+prod[i+k]
if ptr[row] = i then
row = row+1
y[row]←y[row]*
Predicate Instruction
endif
acc ← acc+prod[i]
acc ← acc+prod[i+1]
…
acc ← acc+prod[i+k]
Assembly Optimization
Manually Unrolled Loop

Buffer Location and Performance

Buffer
val
col
ptr
y
prod

map?

Locations
L2
MSMC
cache



Matrix

- Tri-diagonal

$$\begin{bmatrix} 2, & 4, & 0, & 0, & 0, & 0 \\ 5, & 4, & 7, & 0, & 0, & 0 \\ 0, & 6, & 2, & 4, & 0, & 0 \\ 0, & 0, & 3, & 10, & 1, & 0 \\ 0, & 0, & 0, & 4, & 6, & 8 \\ 0, & 0, & 0, & 0, & 2, & 12 \end{bmatrix}$$

- N-diagonal
  3, 151

Buffer Location and Performance

Non-zeros per row	val	col	ptr	y	prod	Gflops	Norm. Perf	Note
3	S S L2 C	L2 L2 L2 C	L2 L2 C C	L2 L2 C C	L1 L2 S C	2.26 1.84 1.23 1.44	1.57 1.28 0.85 1	Best Median Worst[2] All cache
151	L2 S C C	S C C C	L2 L2 C C	L2 L2 C C	L2 S L2 C	3.76 3.55 2.66 2.51	1.50 1.41 1.06 1	Best Median Worst[2] All cache

L1: level 1 SPRAM, L2:level 2 SPRAM, S:MSMC, C:cache

1: The results are normalized to the all cache configuration

2: The worst amongst the configurations with SPRAM

Matrix

- Tri-diagonal

$$\begin{bmatrix} 2, & 4, & 0, & 0, & 0, & 0 \\ 5, & 4, & 7, & 0, & 0, & 0 \\ 0, & 6, & 2, & 4, & 0, & 0 \\ 0, & 0, & 3, & 10, & 1, & 0 \\ 0, & 0, & 0, & 4, & 6, & 8 \\ 0, & 0, & 0, & 0, & 2, & 12 \end{bmatrix}$$

- N-diagonal
  3, 151
- University of Florida sparse matrix collection
- Matrix Market

Matrix	Rows	Columns	Nonzeros	Nonzeros /Row
TSOPF_FS_b300_c3	84414	84414	13135930	155.6
pdb1HYS	36417	36417	4344765	119.3
m_t1	97578	97578	9753570	99.9
audikw_1	943695	943695	77651847	82.3
consph	83334	83334	6010480	72.1
cant	62451	62451	4007383	64.2
pwtk	217918	217918	11524432	52.9
shipsec1	140874	140874	3568176	25
ldoor	952203	952203	23737339	24.9
lhr71c	70304	70304	1528092	21.7
thermal1	82654	82654	574458	6.9
mac_econ_fwd500	206500	206500	1273389	6.1
ASIC_100ks	99190	99190	578890	5.8
scircuit	170998	170998	958936	5.6
shyy161	76480	76480	329762	4.3
mc2depi	525825	525825	2100225	4.0

Testing Platforms

	Intel i7 3770K MKL	NVIDIA GTX 680 cuSparse	NVIDIA Tegra K1 cuSparse	TI 6638K2K
Arch	Ivy Bridge	Kepler	Kepler	KeyStone II
Memory B/W(GB/s)	25.6	192.3	17.1	12.8
SPRAM KB/core	n/a	64/64[1]	64/64[1]	32/1024/768[2]
Single Precision Peak Throughput (Gflops)	448	3090	365	@1.35GHz 172.8(DSP) 44.8(ARM)
TDP(W)	77	195	~10	~15

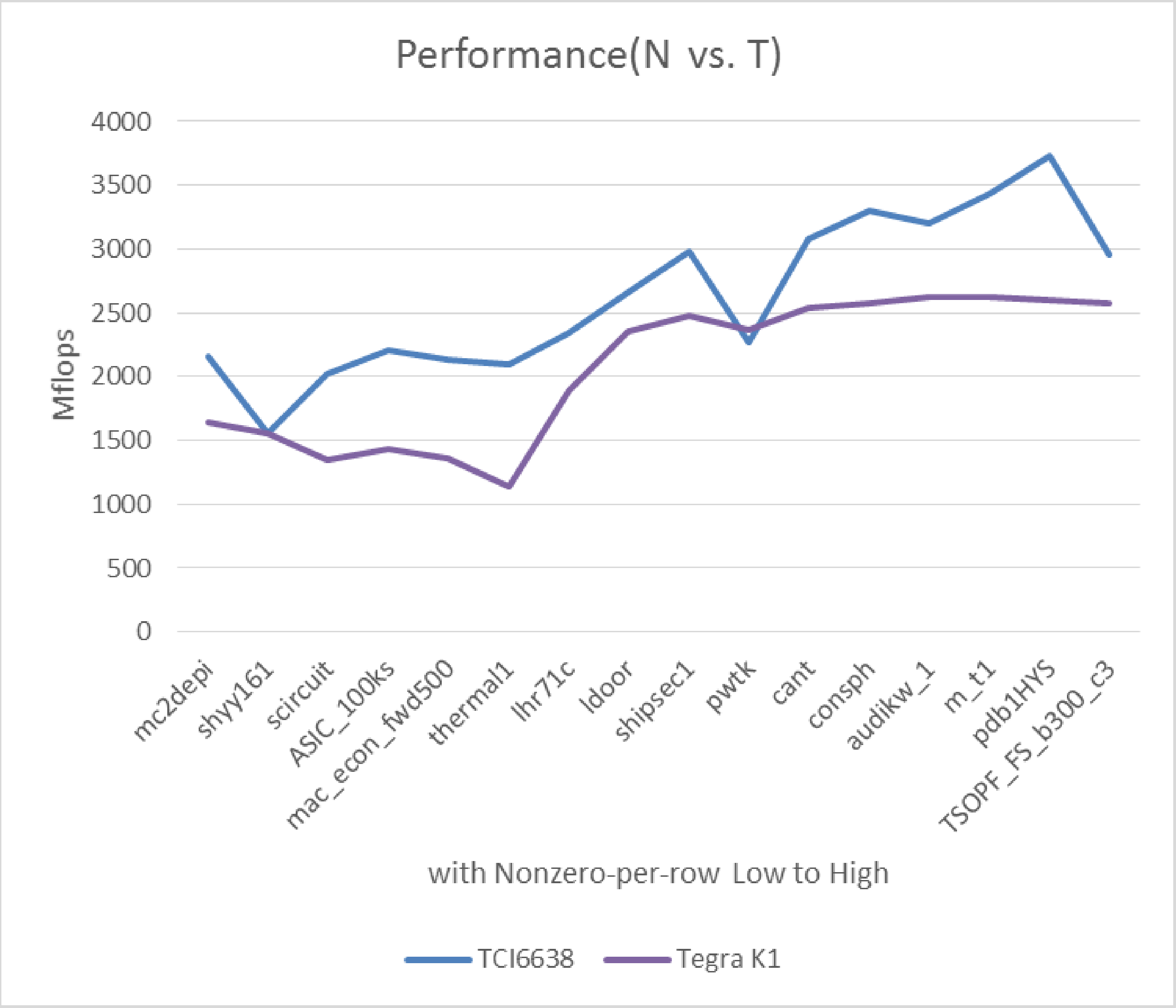
UNIVERSITY OF SOUTH CAROLINA

1. Register file/allocable share memory or L1
2. L1 SRAM / L2 SRAM / MSMC per core

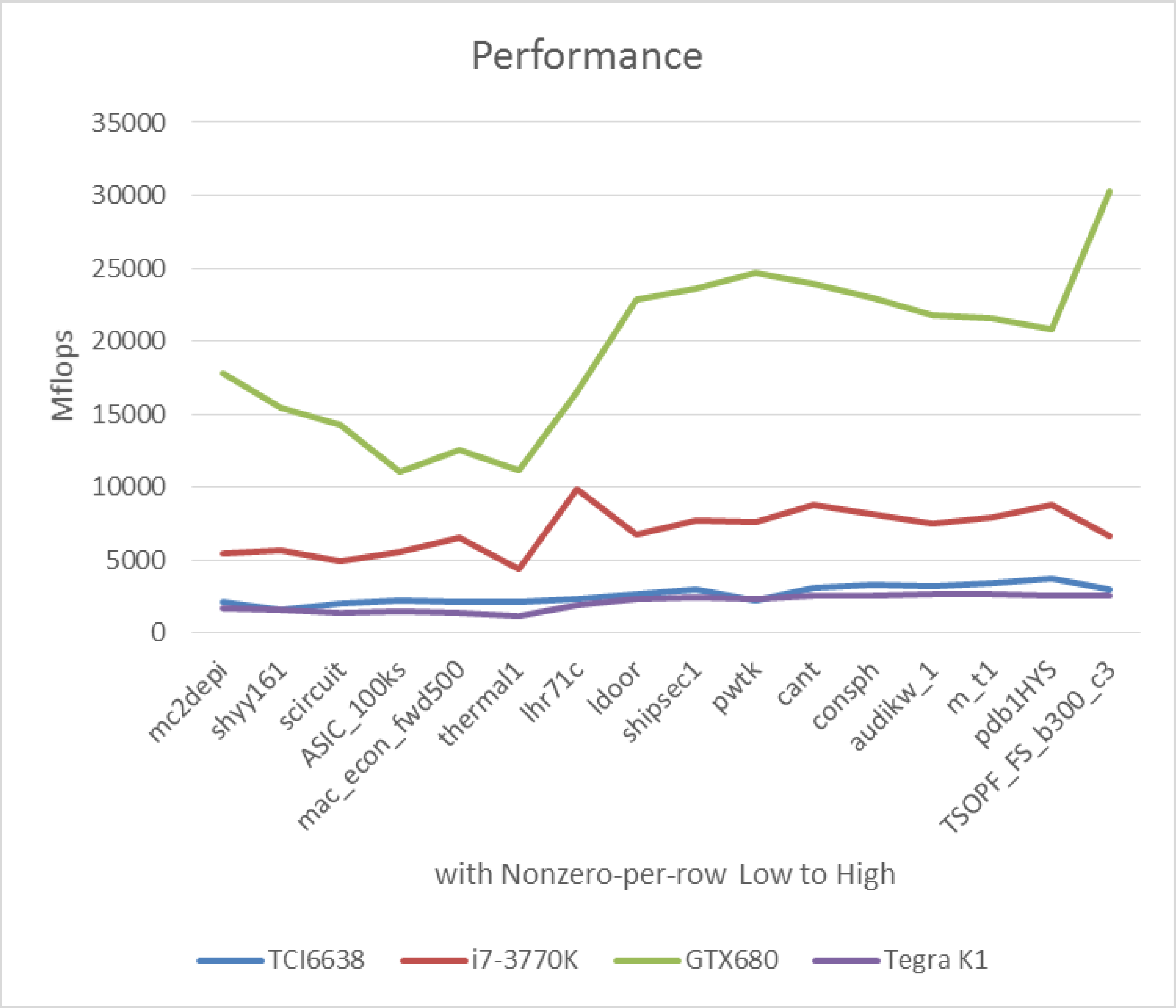
Close Comparison to NVIDIA Tegra K1

	Tegra K1	TI 6638K2K
Architecture	Cortex A15 + Kepler GPU	Cortex A15 + Keystone II DSP
CPU	4 Cores, 2.2 GHz 32K+32K L1 2M L2	4 Cores, 1.4 GHz 32K+32K L1 4M L2
Co-processor	192 CUDA cores @864 MHz 331 Gflops (SP peak)	8 DSP cores @ 1.35GHz 172.8 Gflops (SP peak)
Memory Bandwidth	17.1 GB/s(DDR3-2133)	12.8 GB/s(DDR3-1600)
Power	~ 10 W	~ 15 W

Performance Comparison vs. Nvidia TK1



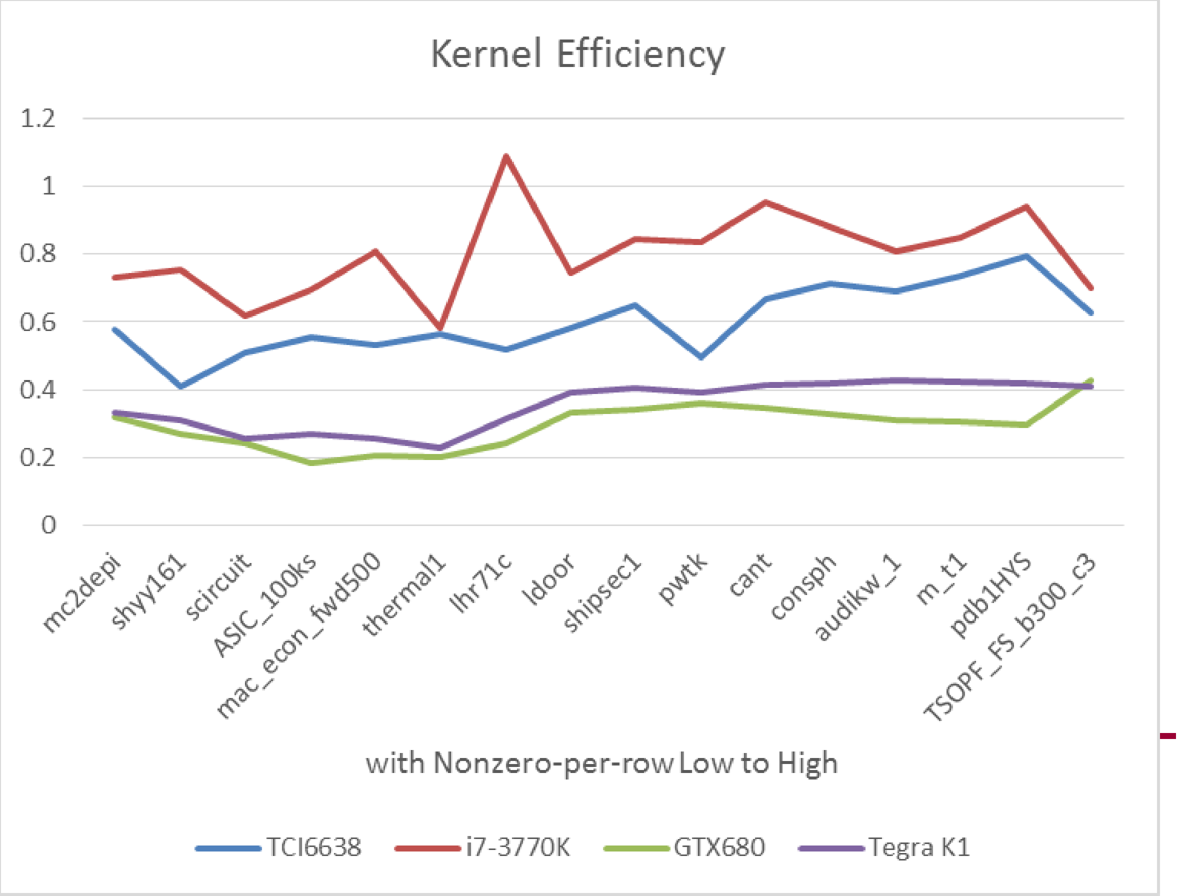
Performance Comparison



Kernel/Memory Efficiency

$$AI = \frac{nnz \times 3ops + rows \times 2ops}{nnz \times 2 \times 4bytes + rows \times 3 \times 4bytes + cols \times 4bytes}$$

efficiency =
observed performance/((Arithmetic Intensity) X (Memory Bandwidth))



Conclusion

- Significant performance gain with loop tuning and SPM optimizations
- Despite the mobile GPU's higher theoretical SP performance and memory bandwidth, the DSP outperforms it with our SpMV implementation.
- Auto-decided SPM buffer allocation/mapping by a model of data access pattern.

Q & A