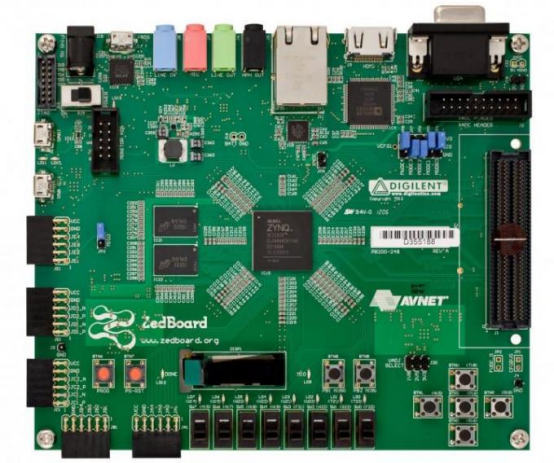
A ~~Brief~~ BLIS Affair with FPGAs

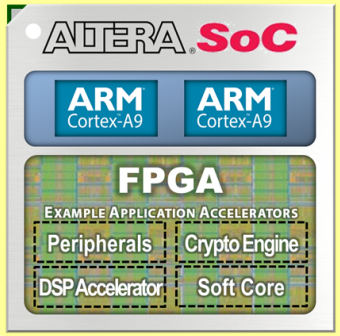
Zhipeng Zhao, Qi Guo, Tze Meng Low

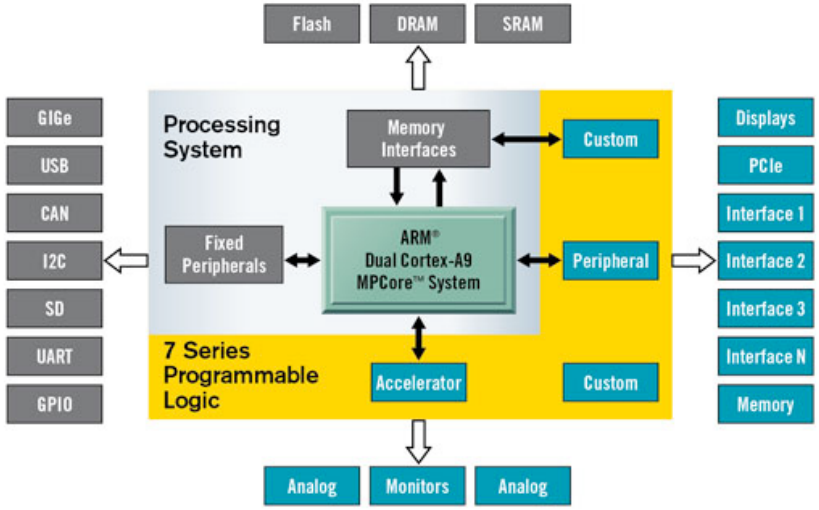
Carnegie Mellon University

FPGAs



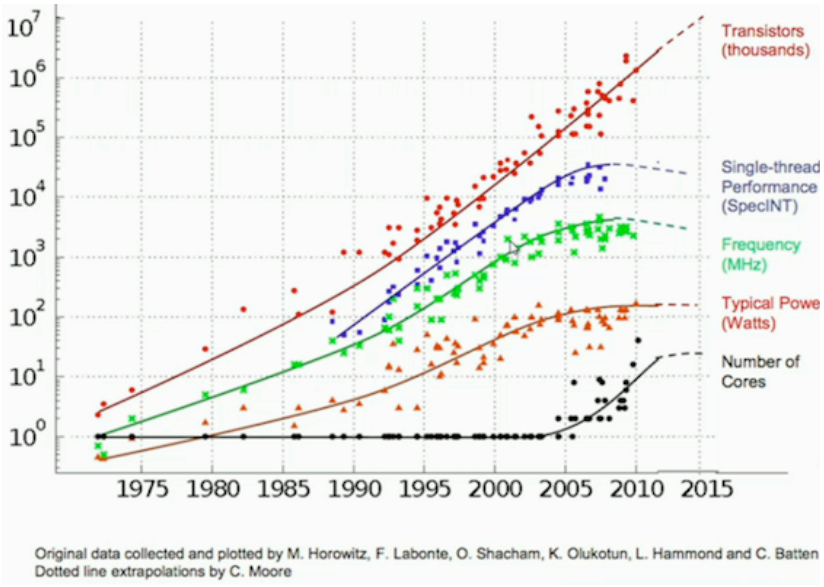
System on Chip (SoC)



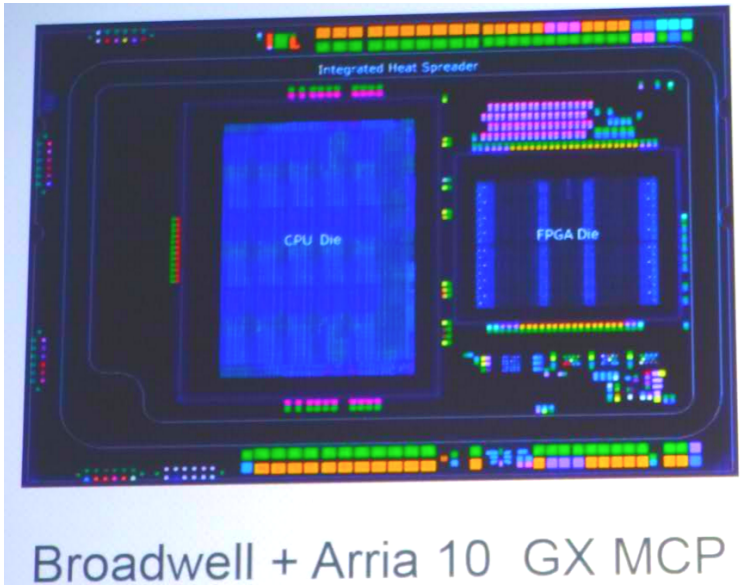


DIY Hardware

Energy-efficient computing



Heterogeneous architecture



Used in Datacenters and Cloud Services

High Level Synthesis

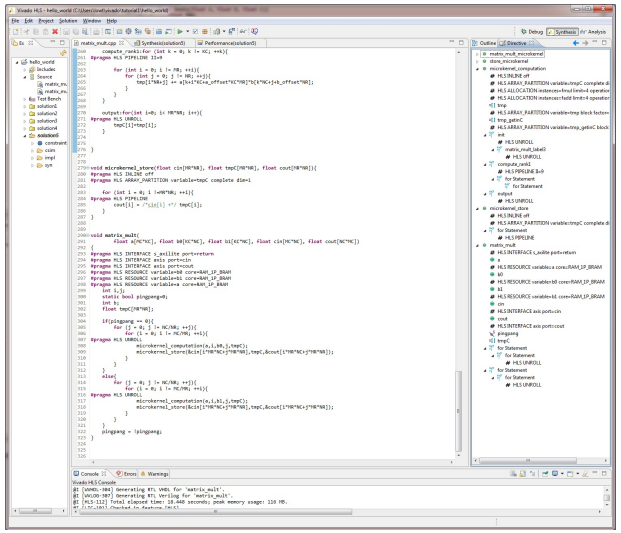
C / OpenCL

High Level Synthesis (HLS)

Altera® SDK for OpenCL

Vivado™ HLS

RTL



Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.41	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
10979	10979	10980	10980	none

Detail

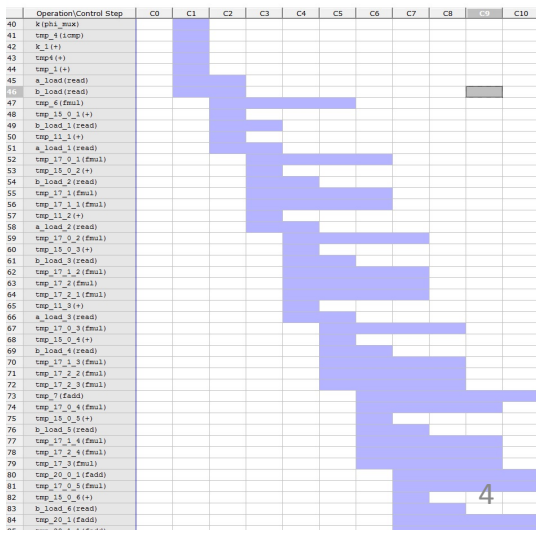
Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	2
FIFO	-	-	-	-
Instance	0	20	5389	5729
Memory	-	-	-	-
Multiplexer	-	-	-	1269
Register	-	-	2350	-
Total	0	20	7739	7000
Available	280	220	106400	53200
Utilization (%)	0	9	7	13



High Level Synthesis

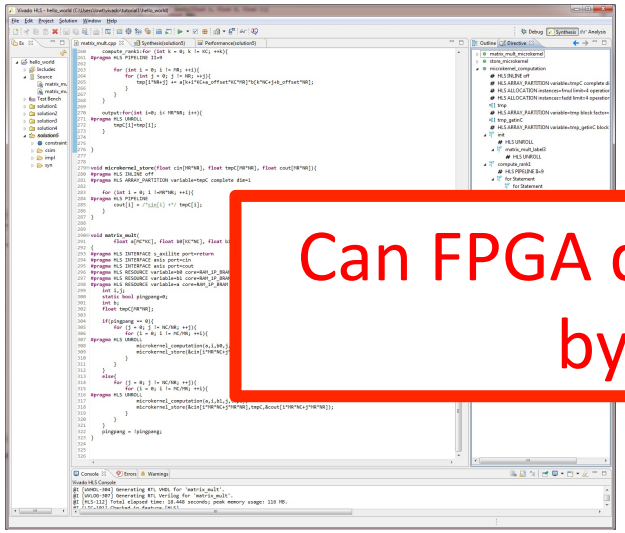
C / OpenCL

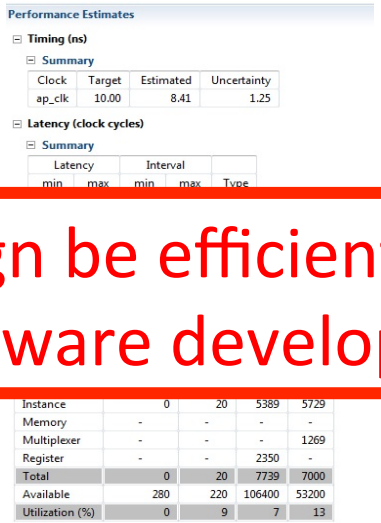
High Level Synthesis (HLS)

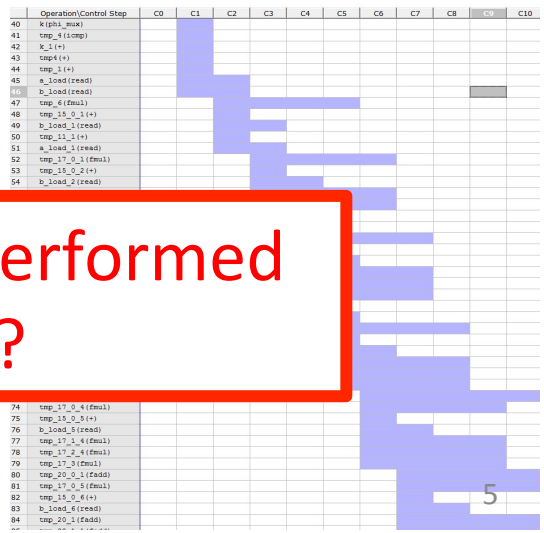
Altera® SDK for OpenCL

Vivado™ HLS

RTL







Can FPGA design be efficiently performed by software developers?

			Naïve			
M	N	K	Cycle	DSP	FF	LUT
4	4	4	882	5	533	884
4	4	8	1626	5	537	889
4	8	4	1714	5	542	895
4	8	8	3162	5	547	900
8	4	4	1754	5	545	895
8	4	8	3234	5	550	900
8	8	4	3418	5	550	906
8	8	8	6306	5	554	911

```
for (int i = 0; i != m; ++i)
   for (int j = 0; j != n; ++j)
      for (int p = 0; p != k; ++p)
           c[i][j] += a[i][p]*b[p][j]
```

			Naïve			
M	N	K	Cycle	DSP	FF	LUT
4	4	4	882	5	533	884
4	4	8	1626	5	537	889
4	8	4	1714	5	542	895
4	8	8	3162	5	547	900
8	4	4	1754	5	545	895
8	4	8	3234	5	550	900
8	8	4	3418	5	550	906
8	8	8	6306	5	554	911

```
for (int i = 0; i != m; ++i)
   for (int j = 0; j != n; ++j)
      for (int p = 0; p != k; ++p)
           c[i][j] += a[i][p]*b[p][j]
```

0.162 FLOPS / cycle

			Naïve			
M	N	K	Cycle	DSP	FF	LUT
4	4	4	882	5	533	884
4	4	8	1626	5	537	889
4	8	4	1714	5	542	895
4	8	8	3162	5	547	900
8	4	4	1754	5	545	895
8	4	8	3234	5	550	900
8	8	4	3418	5	550	906
8	8	8	6306	5	554	911

```
for (int i = 0; i != m; ++i)
    for (int j = 0; j != n; ++j)
        for (int p = 0; p != k; ++p)
            c[i][j] += a[i][p]*b[p][j]
```

2% of available resources

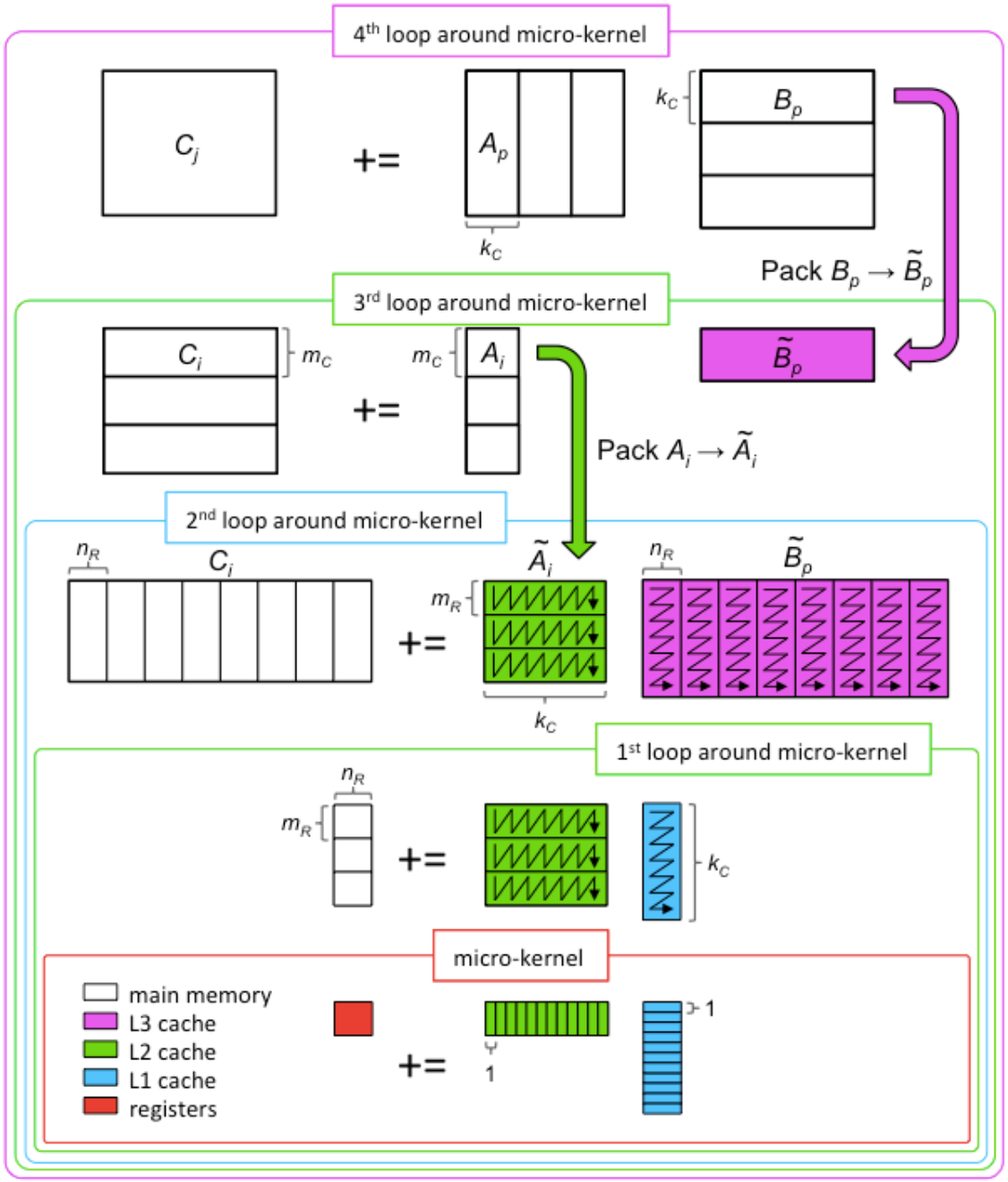
			Naïve				HPC			
M	N	K	Cycle	DSP	FF	LUT	Cycle	DSP	FF	LUT
4	4	4	882	5	533	884	33	44	5407	8964
4	4	8	1626	5	537	889	53	44	5411	8970
4	8	4	1714	5	542	895	34	85	10908	17578
4	8	8	3162	5	547	900	54	85	10912	17585
8	4	4	1754	5	545	895	34	85	10748	17581
8	4	8	3234	5	550	900	54	85	10752	17589
8	8	4	3418	5	550	906	34	167	21142	34547
8	8	8	6306	5	554	911	54	167	21146	34555

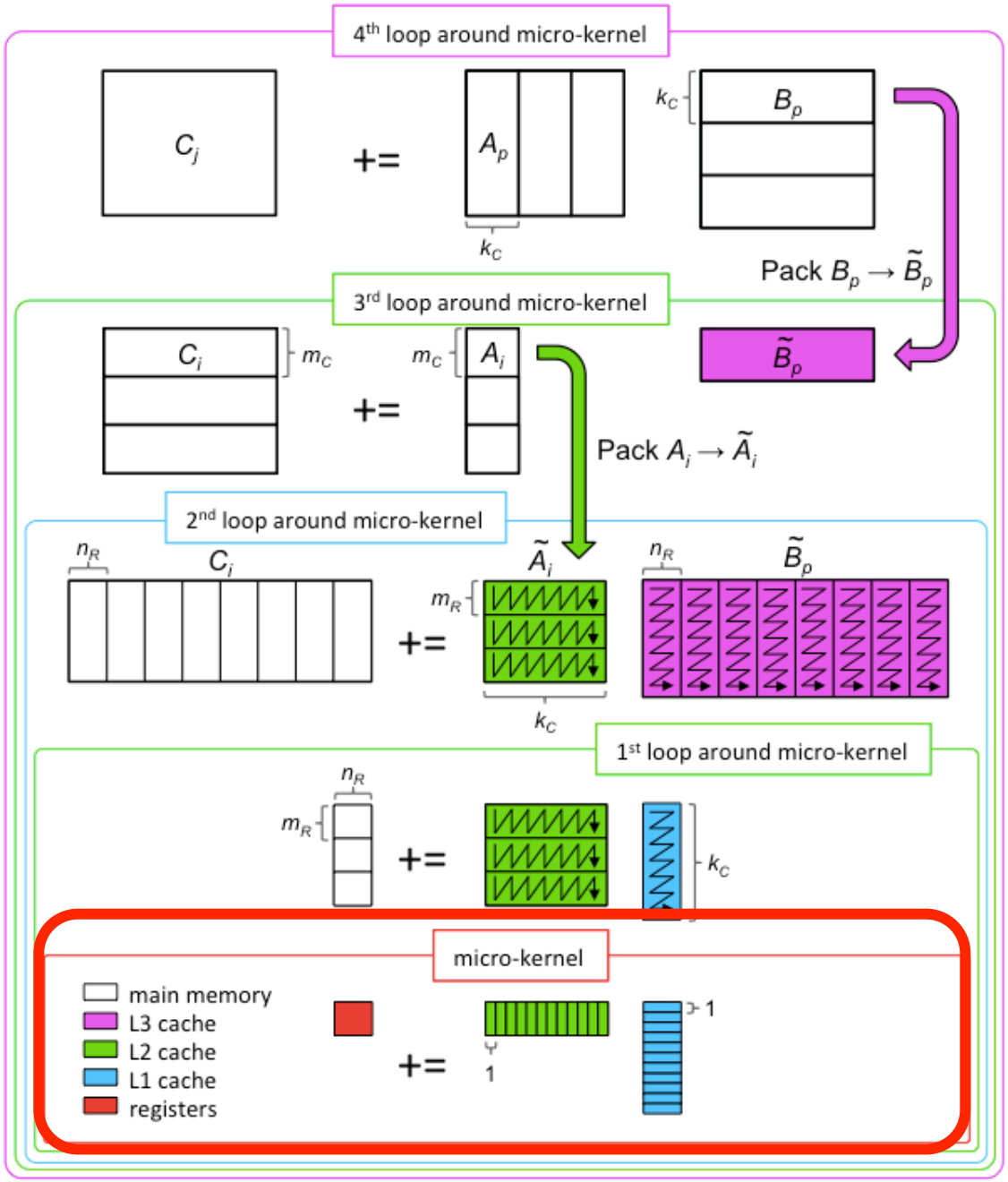
18.96 FLOPS / cycle

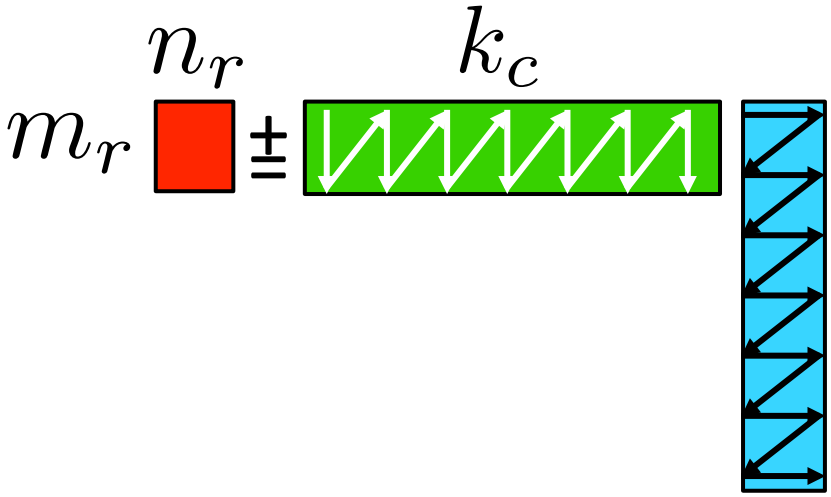
```
#pragma HLS ARRAY_PARTITION
     variable=C block factor=4 dim=1
```

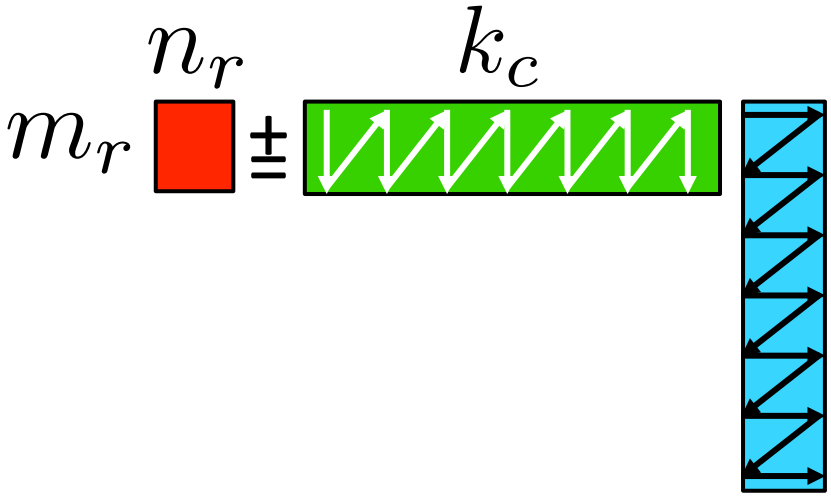
			Naïve				HPC				HLS			
M	N	K	Cycle	DSP	FF	LUT	Cycle	DSP	FF	LUT	Cycle	DSP	FF	LUT
4	4	4	882	5	533	884	33	44	5407	8964	33	44	5407	8964
4	4	8	1626	5	537	889	53	44	5411	8970	53	44	5411	8970
4	8	4	1714	5	542	895	34	85	10908	17578	34	85	10908	17578
4	8	8	3162	5	547	900	54	85	10912	17585	54	85	10912	17585
8	4	4	1754	5	545	895	34	85	10748	17581	34	85	10748	17581
8	4	8	3234	5	550	900	54	85	10752	17589	54	85	10752	17589
8	8	4	3418	5	550	906	34	167	21142	34547	34	167	21142	34547
8	8	8	6306	5	554	911	54	167	21146	34555	54	167	21146	34555

```
#pragma HLS ARRAY_PARTITION
     variable=C block factor=4 dim=1
#pragma HLS PIPELINE II=5
#pragma HLS UNROLL
```

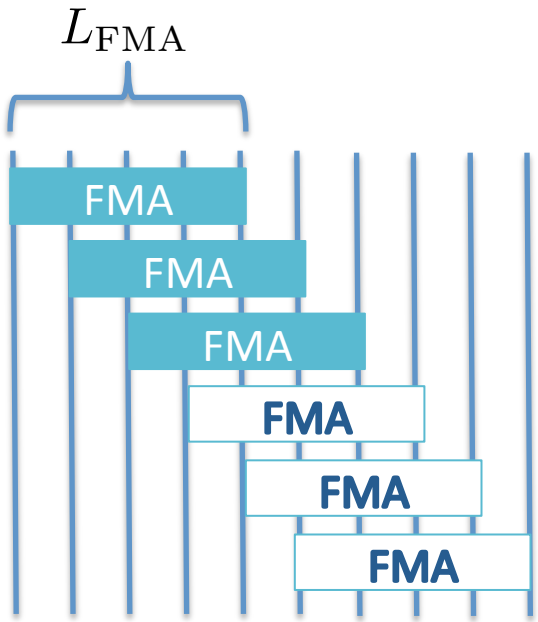

4th loop around micro-kernel
C_j
+=
A_p
k_C
k_C
B_p
Pack $B_p \rightarrow \tilde{B}_p$
$\tilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
m_C
A_i
+=
Pack $A_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
n_R
C_i
+=
$\tilde{A}_i$
m_R
k_C
n_R
$\tilde{B}_p$
1st loop around micro-kernel
n_R
m_R
+=
k_C
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
+=
1
1

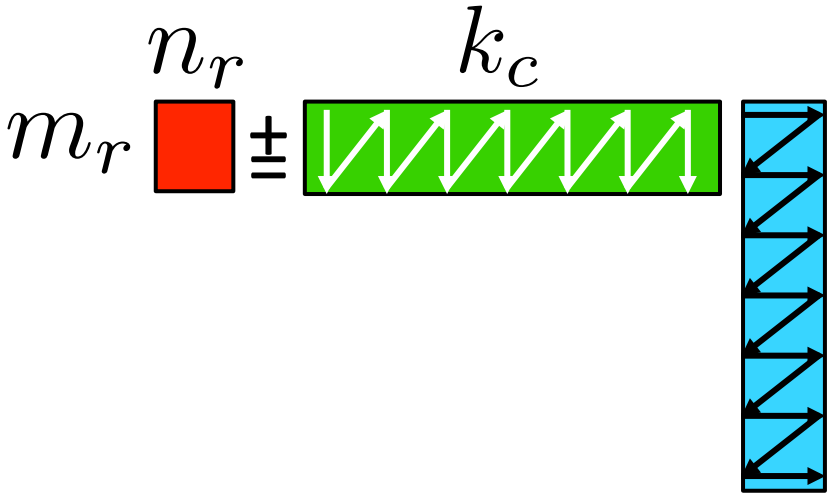

4th loop around micro-kernel
C_j
+=
A_p
k_C
k_C
B_p
Pack $B_p \rightarrow \tilde{B}_p$
$\tilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
m_C
A_i
+=
Pack $A_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
n_R
C_i
+=
$\tilde{A}_i$
m_R
k_C
n_R
$\tilde{B}_p$
1st loop around micro-kernel
n_R
m_R
+=
k_C
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
+=
1
1


n_r
k_c
m_r
±

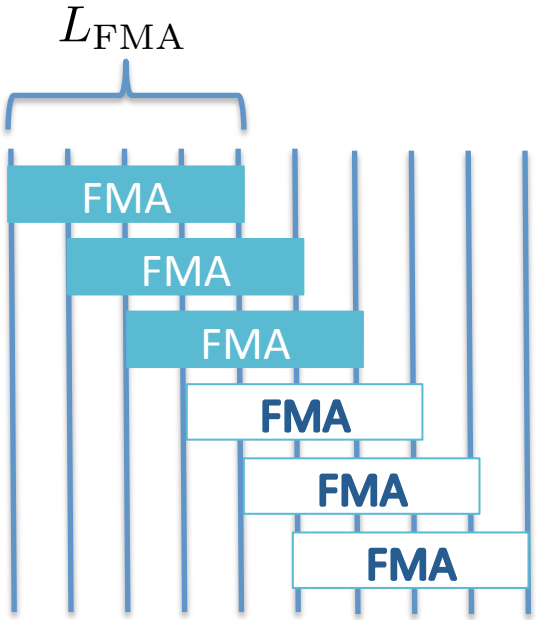


$$m_r n_r \geq N_{\mathrm{FMA}} L_{\mathrm{FMA}} N_{vec}$$

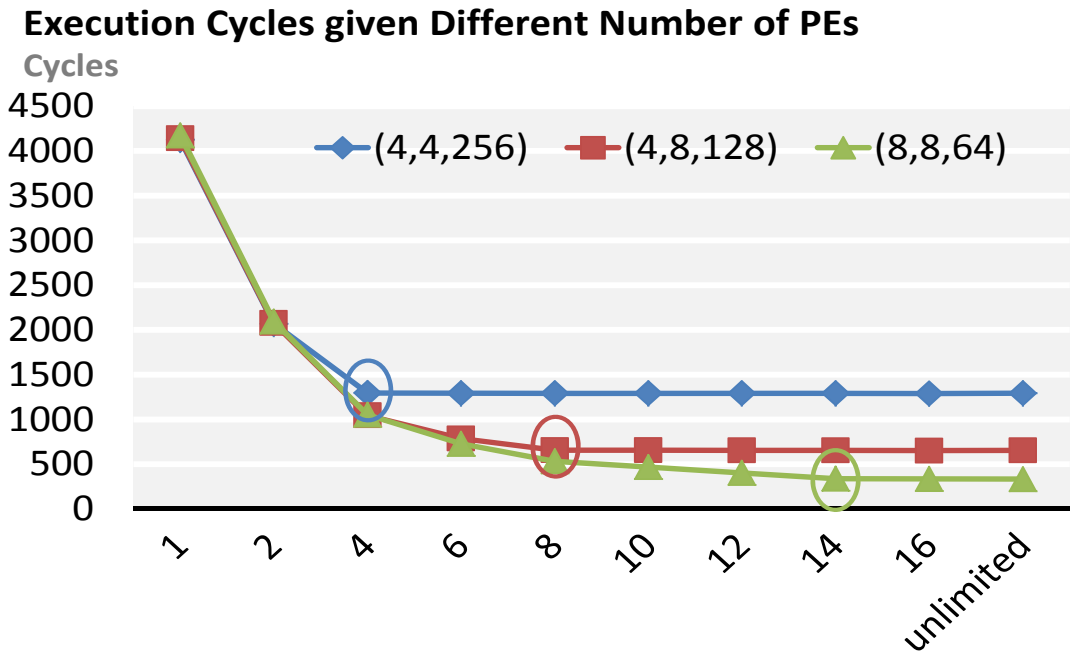


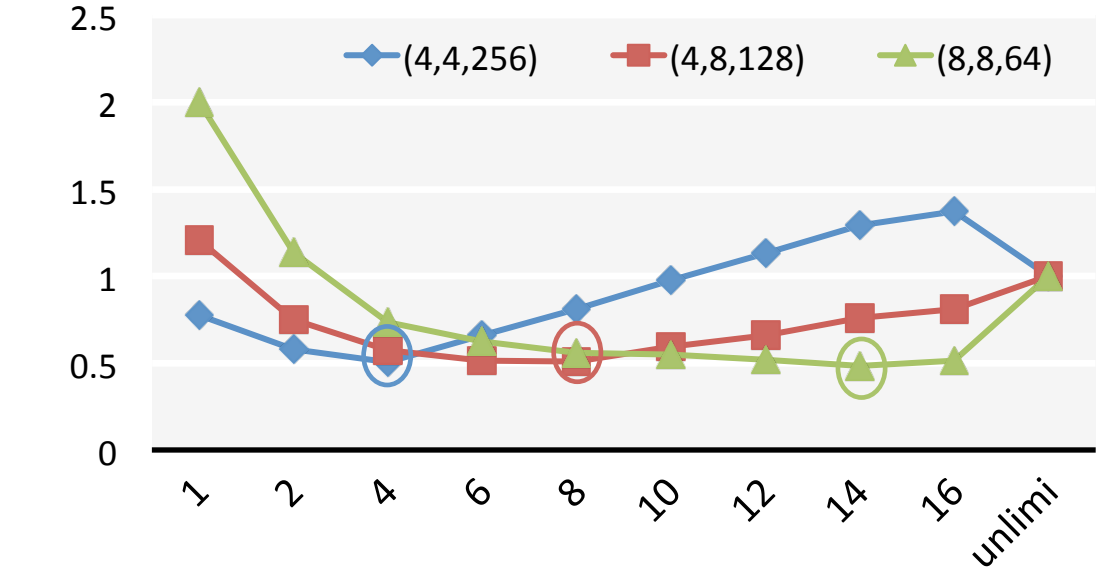


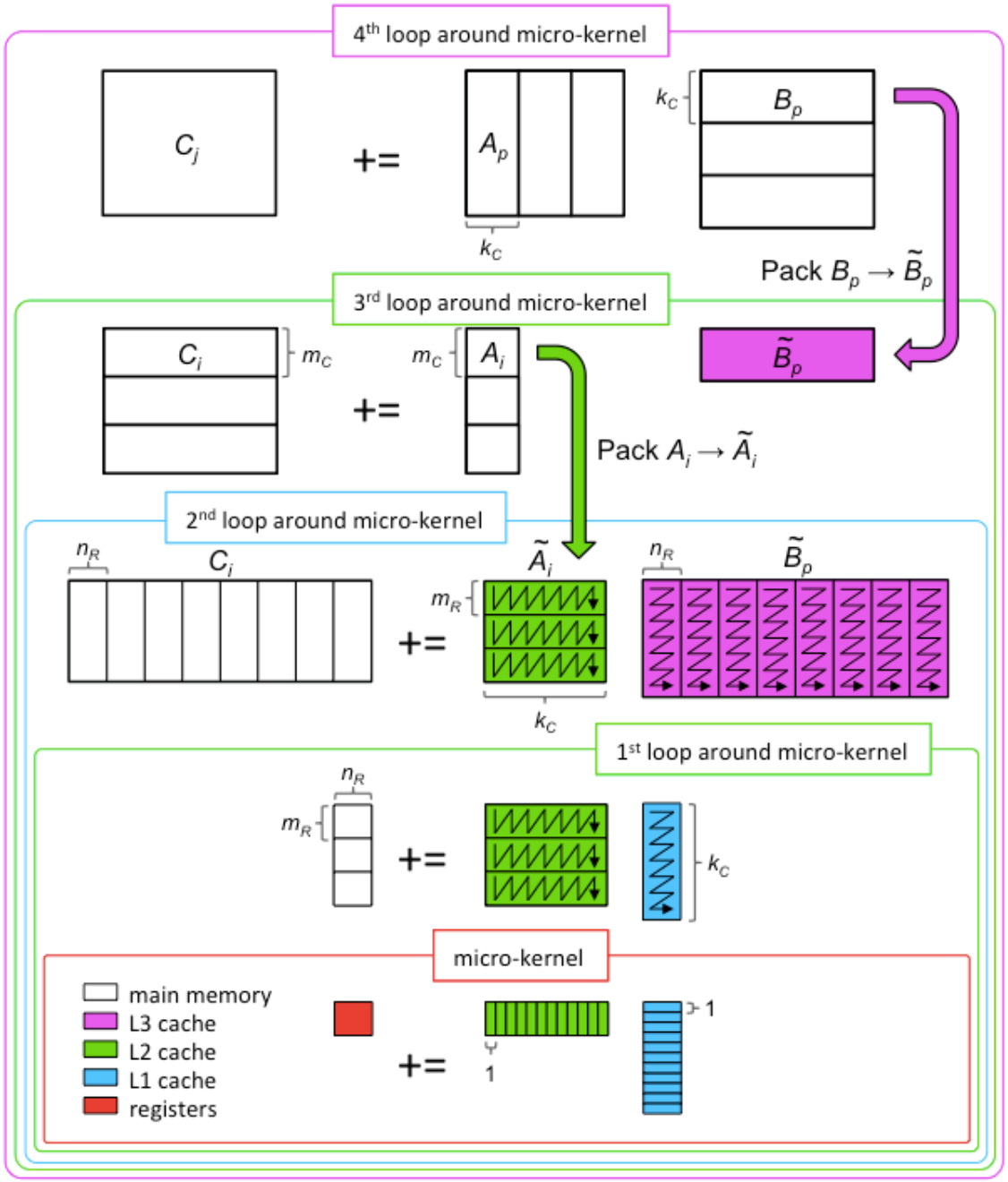
$$m_r n_r \geq N_{\mathrm{FMA}} L_{\mathrm{FMA}} N_{vec}$$



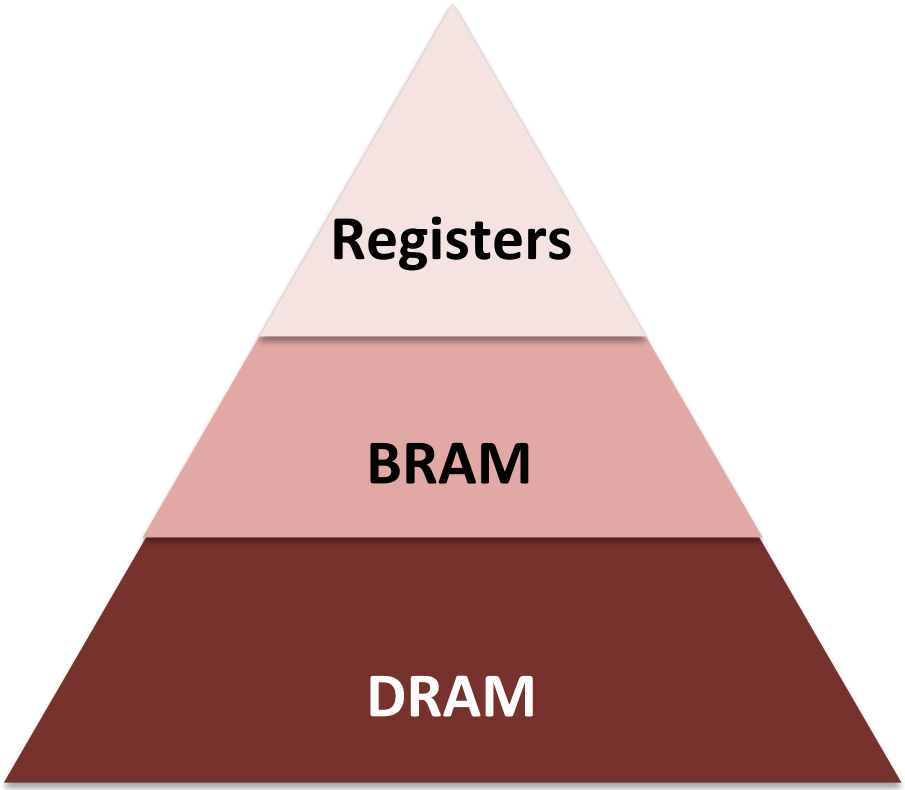
- Key Differences
  - Variable architecture
  - Fixed problem size $m_r \quad n_r$
  - Dependent on HLS implementation for L_{FMA}


Execution Cycles given Different Number of PEs
Cycles
4500
4000
3500
3000
2500
2000
1500
1000
500
0
(4,4,256)
(4,8,128)
(8,8,64)
1
2
4
6
8
10
12
14
16
unlimited

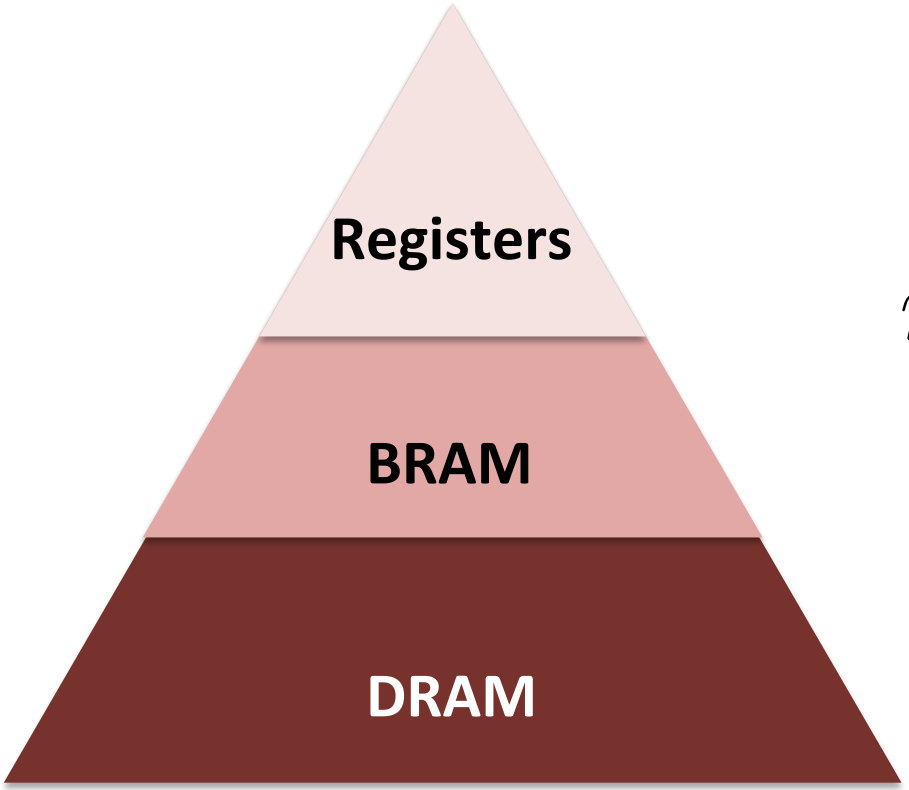

Power Efficiency given Different Number of PEs
Normalized Cycle*Power Number
2.5
2
1.5
1
0.5
0
(4,4,256)
(4,8,128)
(8,8,64)
1
2
4
6
8
10
12
14
16
unlimi

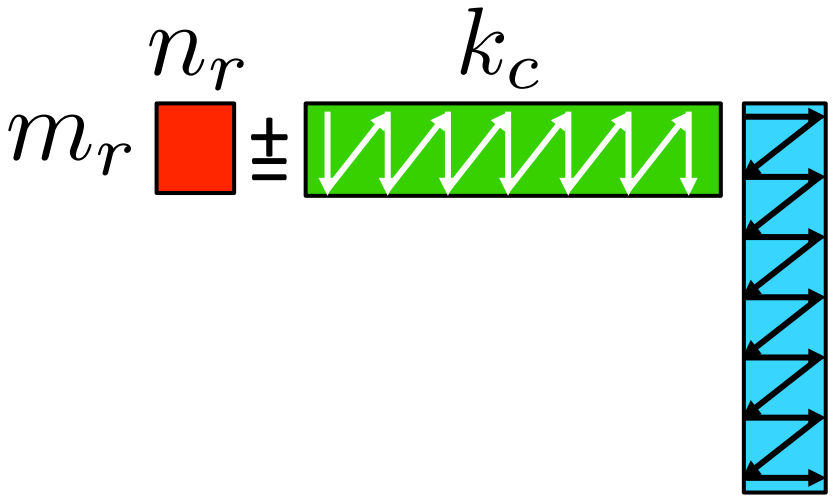

4th loop around micro-kernel
C_j
+=
A_p
k_C
k_C
B_p
Pack $B_p \rightarrow \tilde{B}_p$
$\tilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
m_C
A_i
+=
Pack $A_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
n_R
C_i
+=
$\tilde{A}_i$
m_R
k_C
n_R
$\tilde{B}_p$
1st loop around micro-kernel
n_R
m_R
+=
k_C
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
+=
1
1

Data Movement

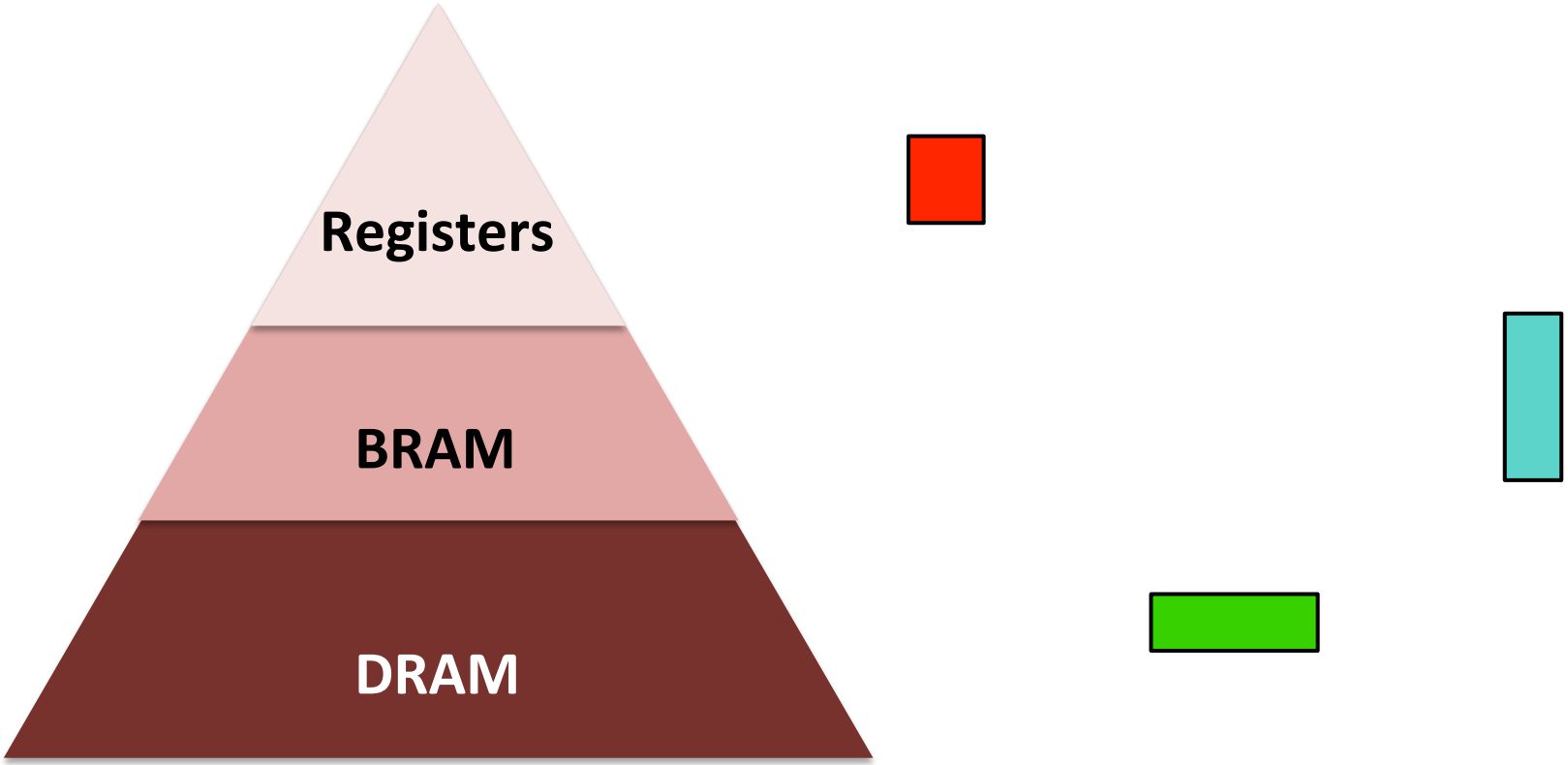


Data Movement

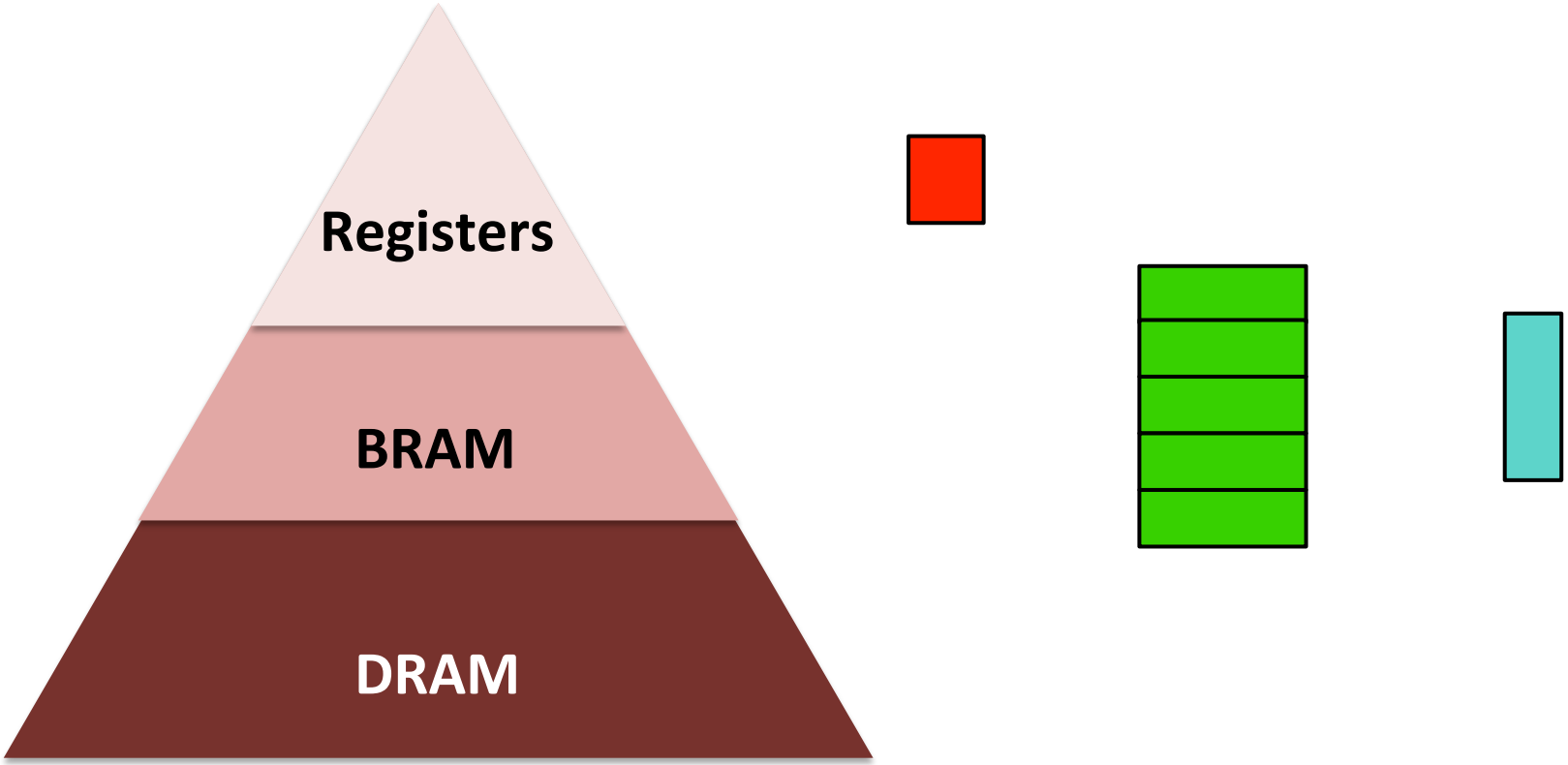


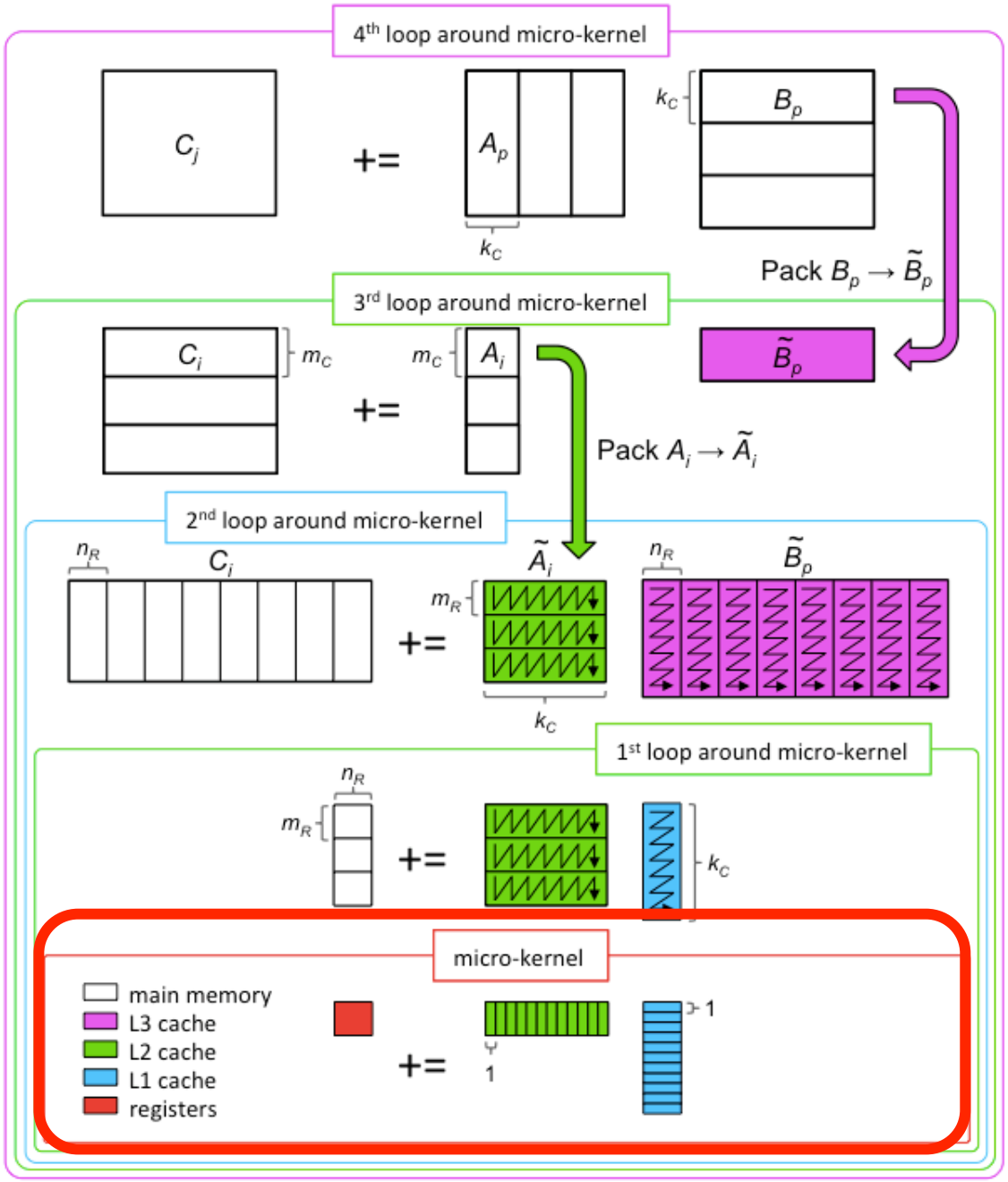


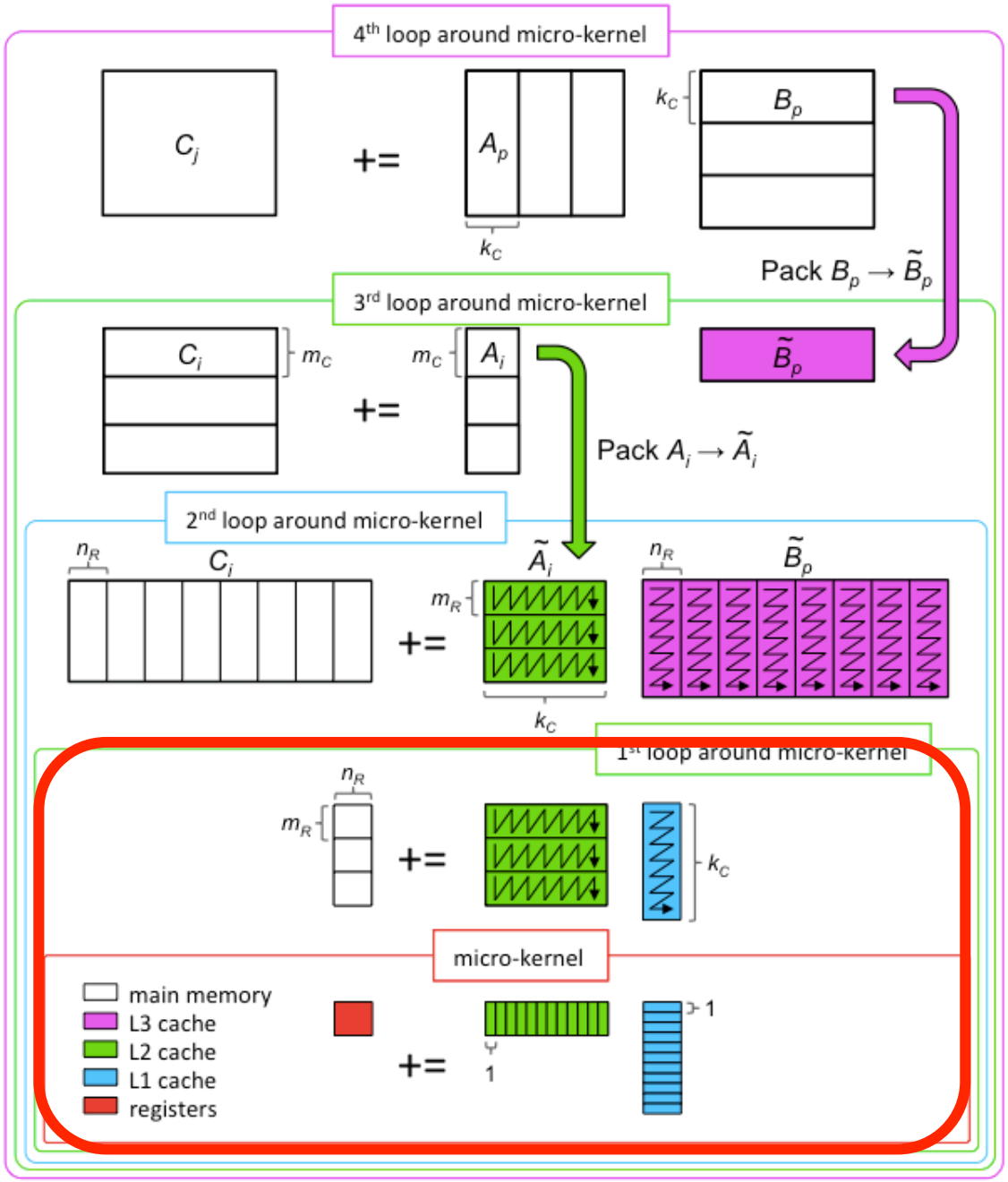
Data Movement

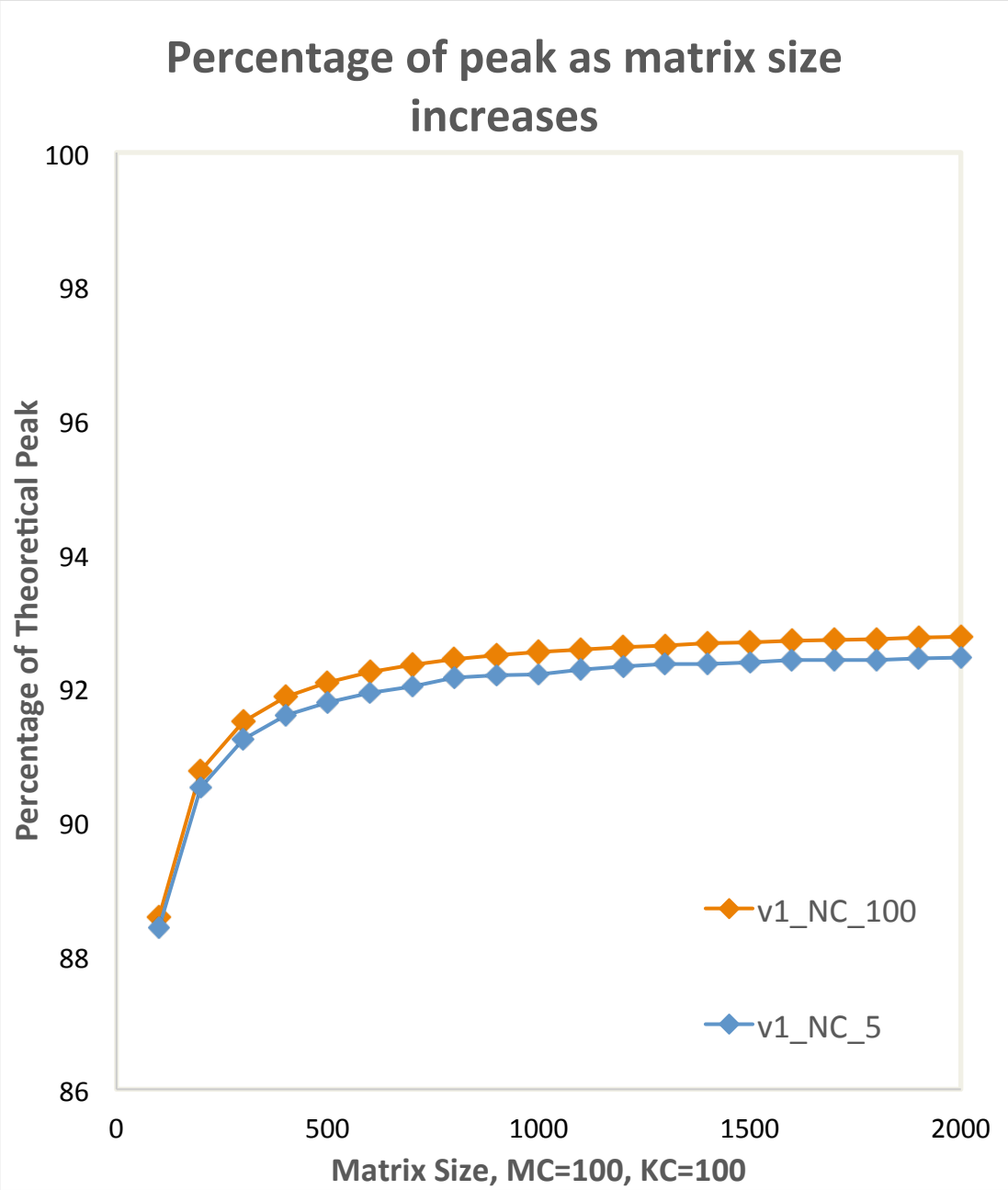


Data Movement




4th loop around micro-kernel
C_j
+=
A_p
k_C
k_C
B_p
Pack $B_p \rightarrow \tilde{B}_p$
$\tilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
m_C
A_i
+=
Pack $A_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
n_R
C_i
+=
$\tilde{A}_i$
m_R
k_C
n_R
$\tilde{B}_p$
1st loop around micro-kernel
n_R
m_R
+=
k_C
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
+=
1
1


4th loop around micro-kernel
C_j
+=
A_p
k_C
k_C
B_p
Pack $B_p \rightarrow \tilde{B}_p$
$\tilde{B}_p$
3rd loop around micro-kernel
C_i
m_C
m_C
A_i
+=
Pack $A_i \rightarrow \tilde{A}_i$
2nd loop around micro-kernel
n_R
C_i
+=
$\tilde{A}_i$
m_R
k_C
n_R
$\tilde{B}_p$
1st loop around micro-kernel
n_R
m_R
+=
k_C
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers
+=
1
1



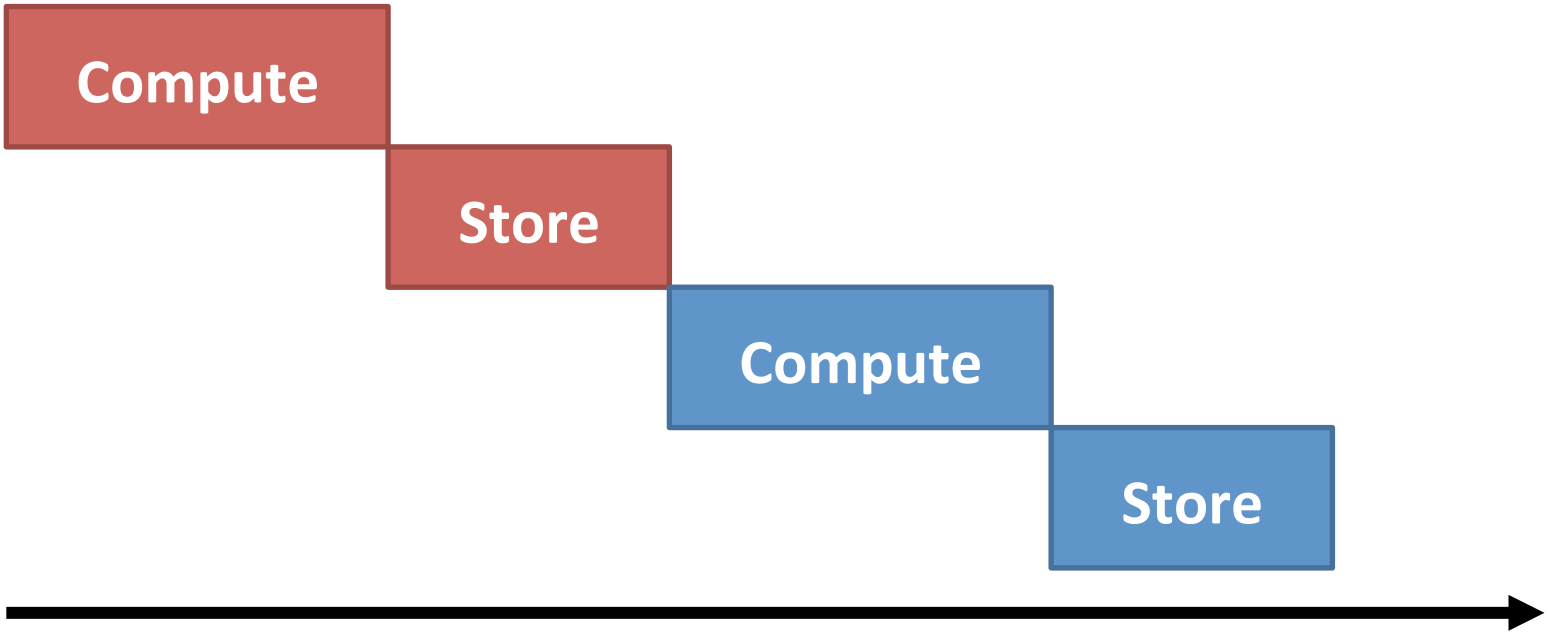
Based on 4 FMA units

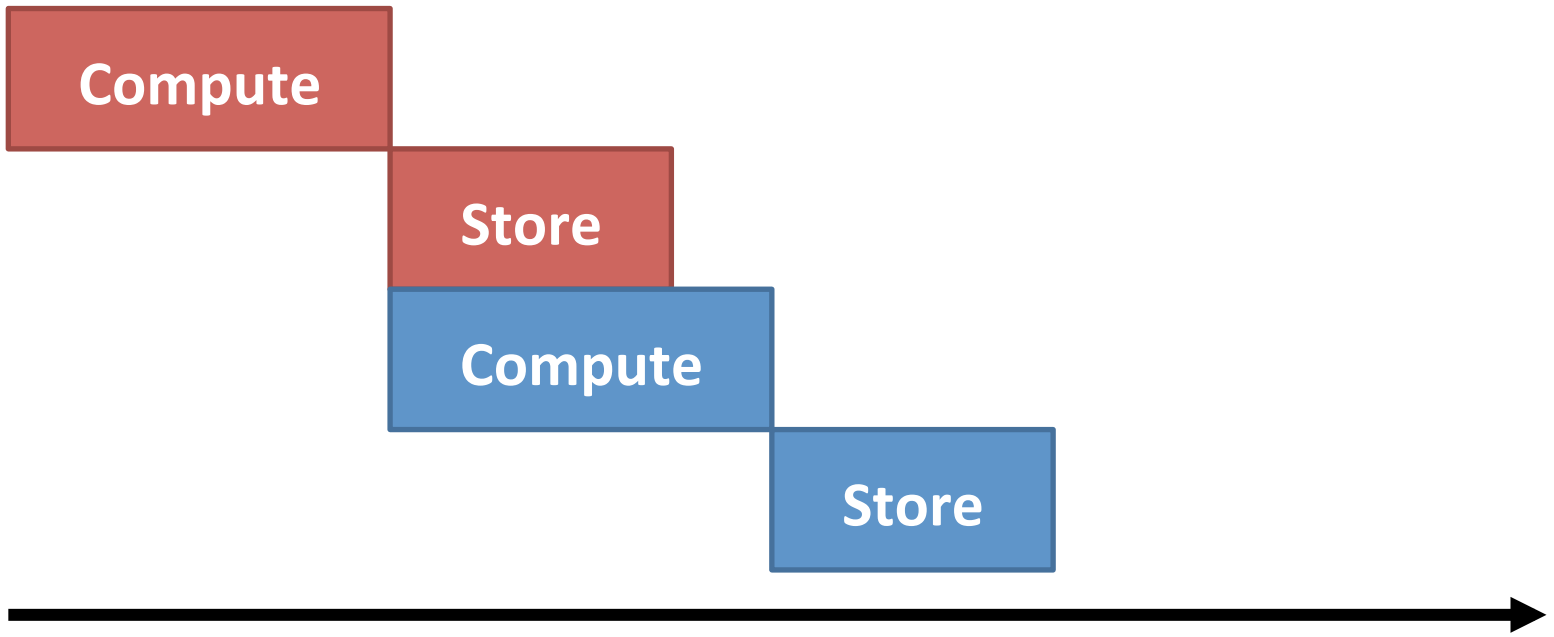
$$m_r = 4$$

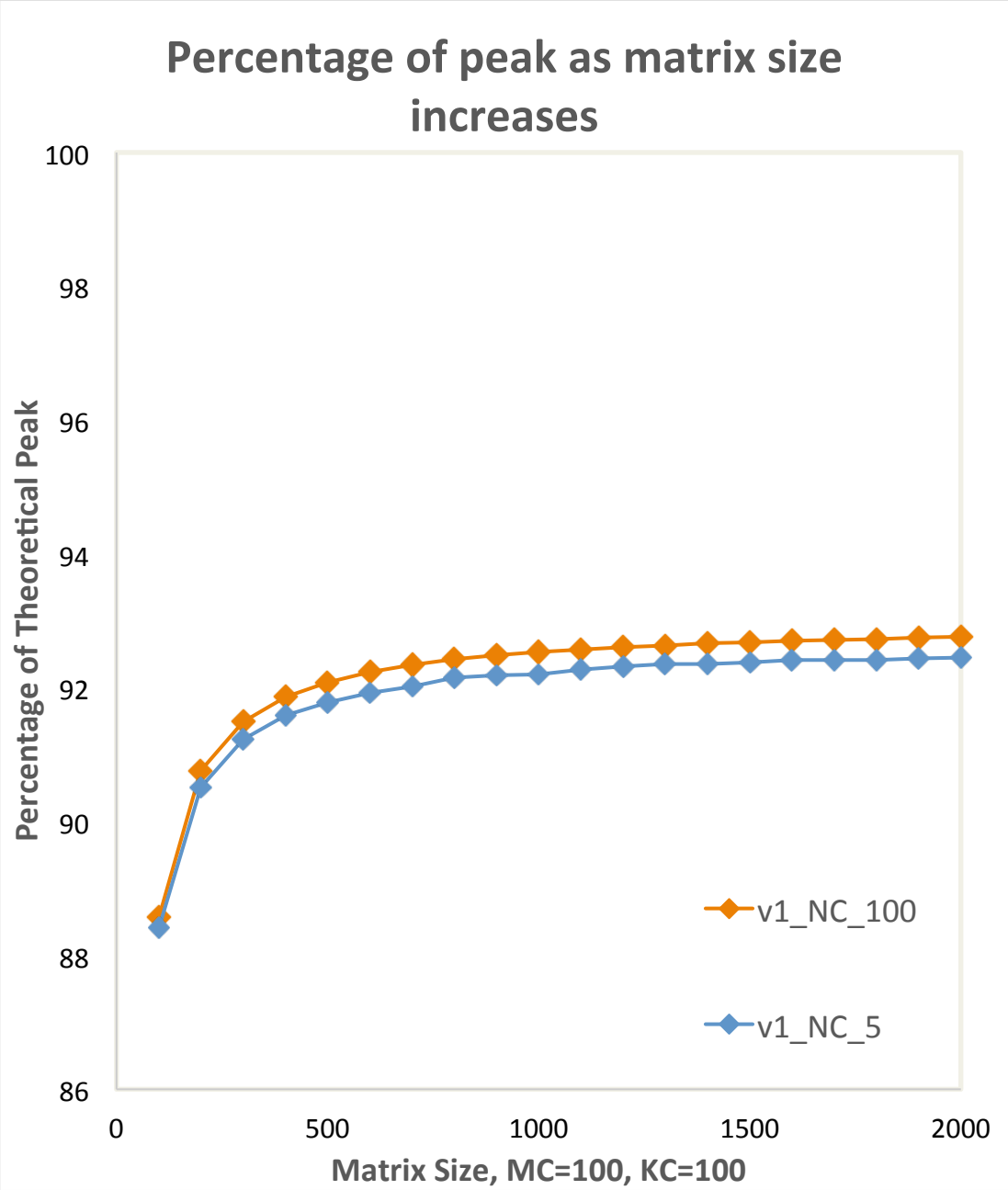
$$n_r = 5$$

$$k_c = 100$$

$$m_c = 20$$


Compute
Store
Compute
Store


Compute
Store
Compute
Store



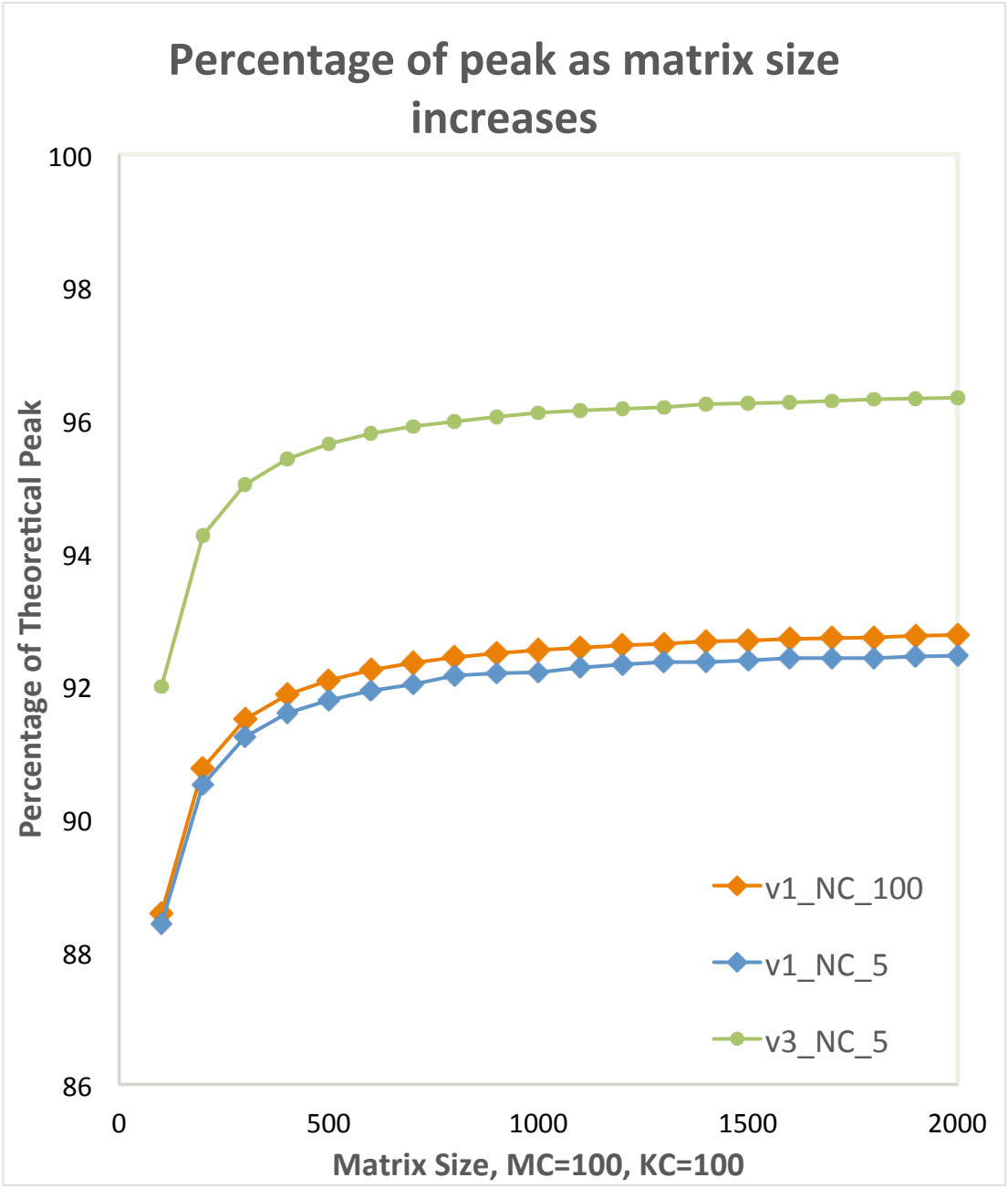
Based on 4 FMA units

$$m_r = 4$$

$$n_r = 5$$

$$k_c = 100$$

$$m_c = 20$$



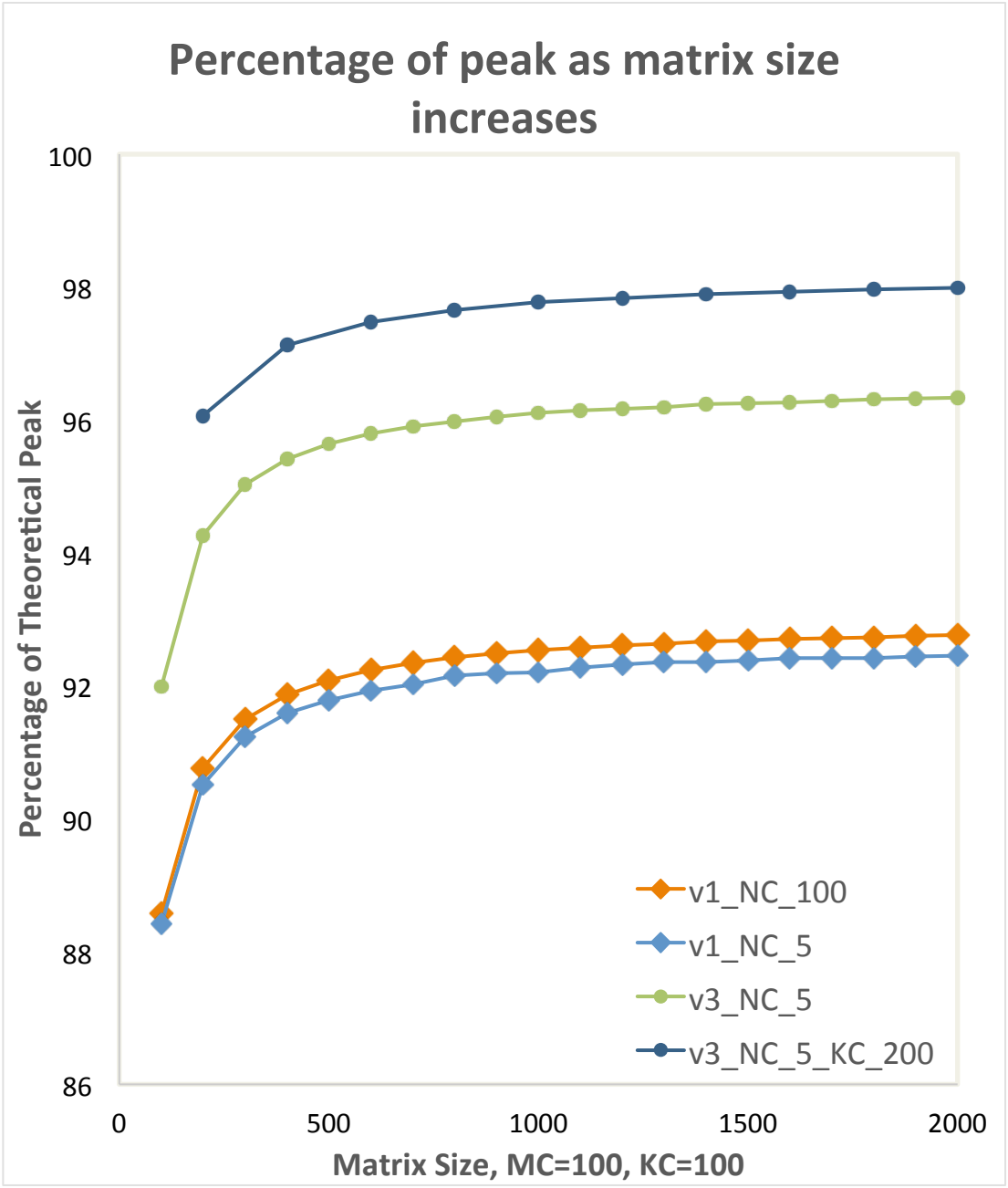
Based on 4 FMA units

$$m_r = 4$$

$$n_r = 5$$

$$k_c = 100$$

$$m_c = 20$$



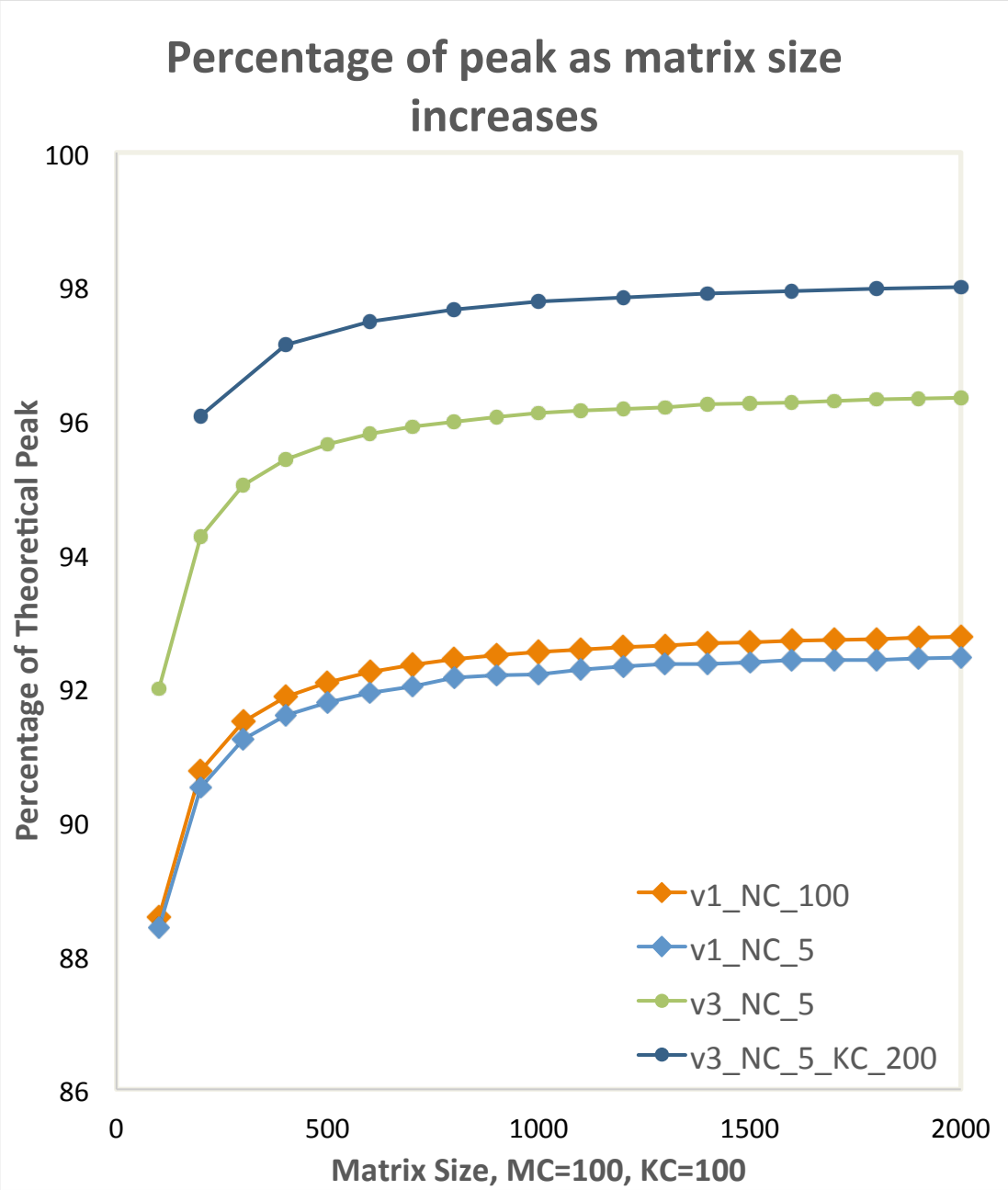
Based on 4 FMA units

$$m_r = 4$$

$$n_r = 5$$

$$k_c = \cancel{100}\ 200$$

$$m_c = 20$$



Based on 4 FMA units

$$m_r = 4$$

$$n_r = 5$$

$$k_c = \cancel{100}\ 200$$

$$m_c = 20$$

Double buffer the movement of A

Summary

- Analytical models for BLIS are good starting points for FPGAs
- Data movement requirements change how kernels are written
- Next, parallel BLIS implementations

Questions?