HISQy Business

Evan Weinberg, Senior Developer Technology Compute Engineer, NVIDIA
Lattice2023, July 31, 2023
(In collaboration with Venkitesh Ayyar, Richard Brower, Kate Clark)

“There’s going to be a GPU raffle!
The drawing will be during the Lattice 2024/2025 announcement on Friday.”

- Me

Agenda

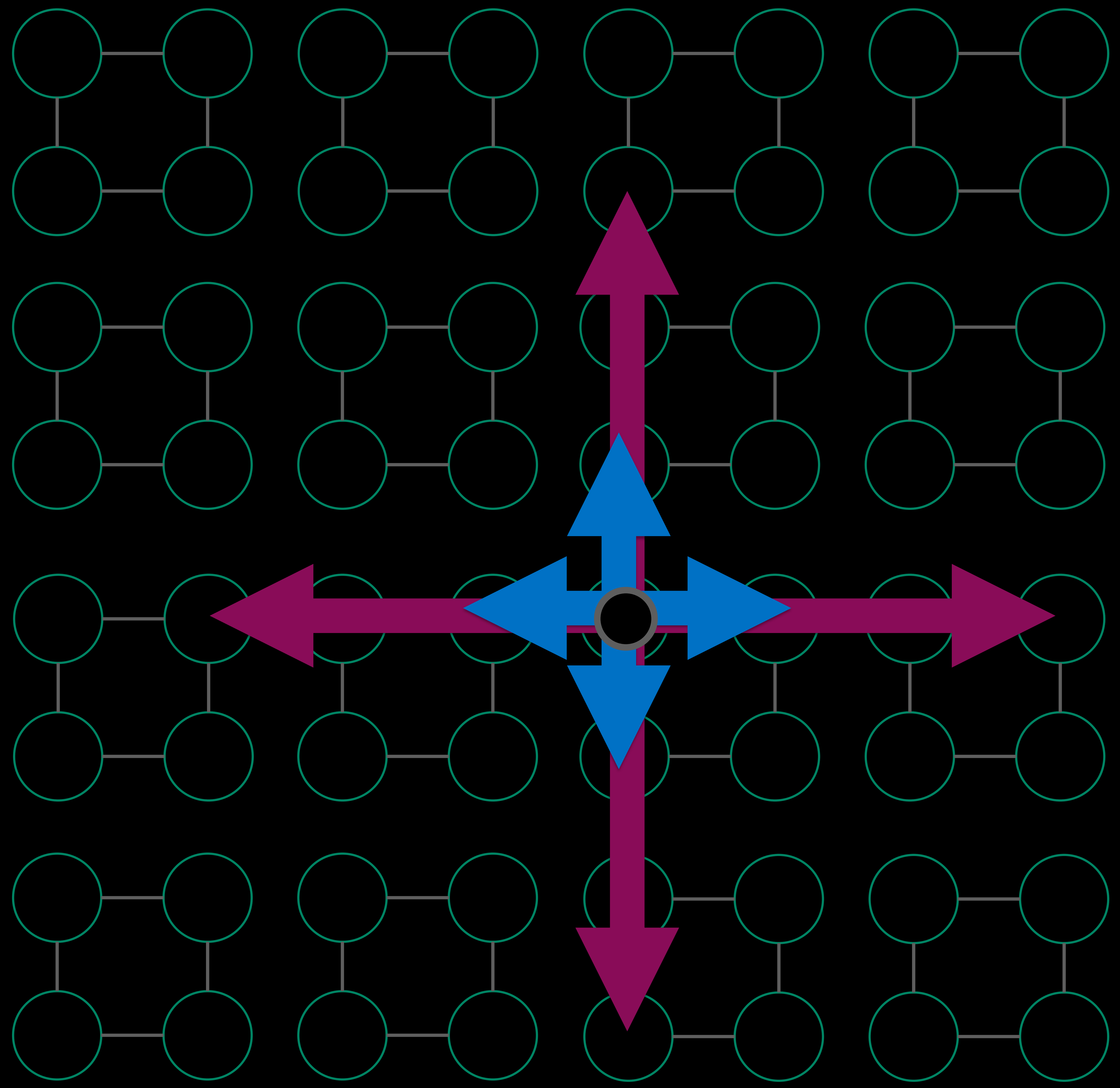
- Takeaways & Challenges
- HISQ Crash Course
- HISQ Force
- HISQ Domain-Decomposed Preconditioning
- Future Work

Takeaways & Challenges

Takeaways

Speeding up HISQ workflows

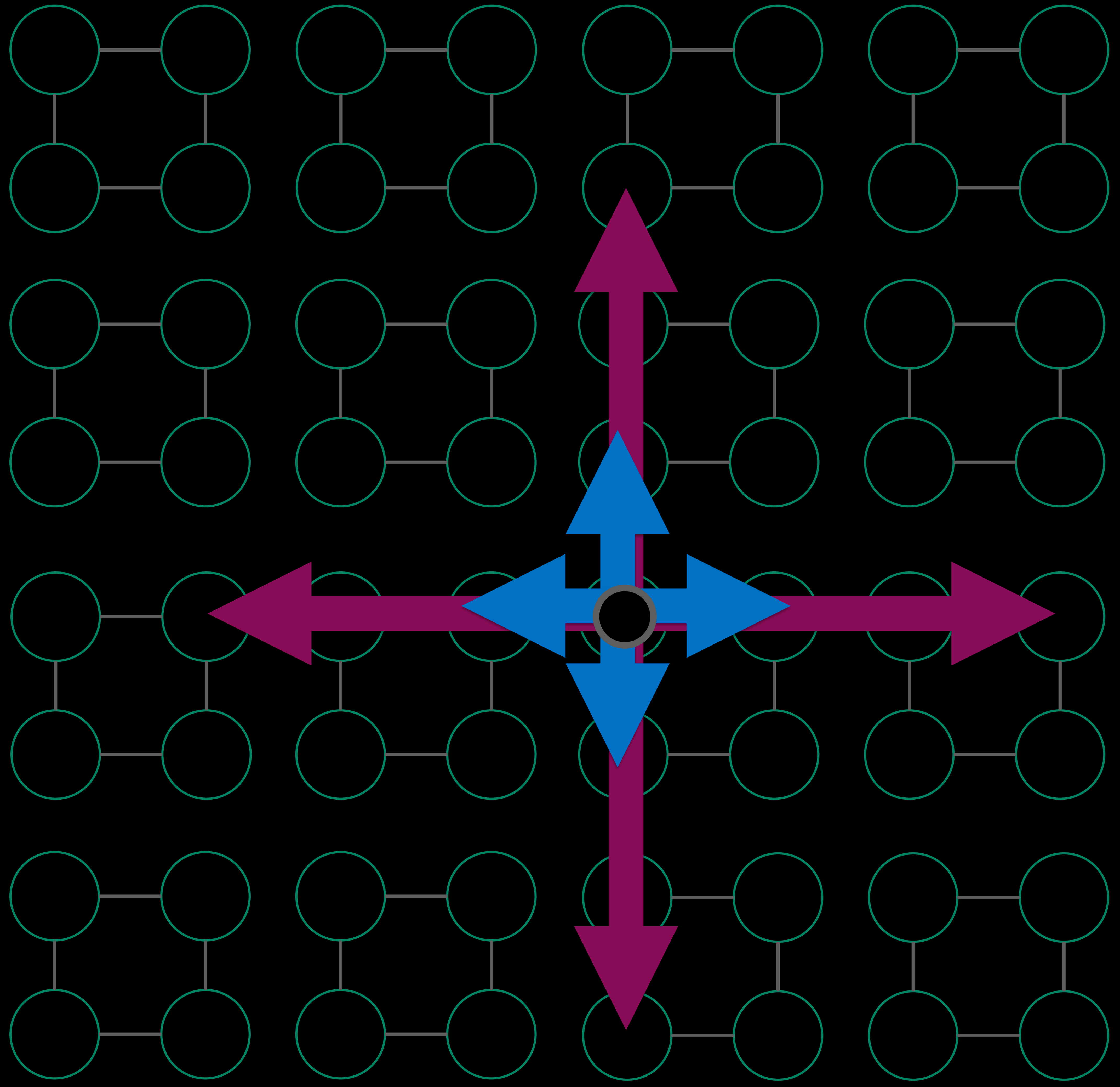
- HISQ: highly improved staggered quarks
  - Smeared links: lots of locality to exploit
- **New:** hugely fused HISQ force implementation in **QUDA**
  - **Merged:** **https://github.com/lattice/quda/pull/1367**



Takeaways

Speeding up HISQ workflows

- HISQ: highly improved staggered quarks
  - Smeared links: lots of locality to exploit
- **New:** hugely fused HISQ force implementation in **QUDA**
  - **Merged:** https://github.com/lattice/quda/pull/1367
- Modern machines have varying degrees of network performance
  - Domain-decomposition algorithms become increasingly important
  - HISQ's distance one and three terms introduce conceptual challenges
- **New:** (mostly-)optimized implementation of a local HISQ preconditioner in **QUDA**
  - We have demonstrated *numerical stability*
  - And, in some cases, faster propagator solves---with performance successes and failures understood
  - **WIP branch, constantly in flux:** https://github.com/lattice/quda/tree/feature/stag-invert-cleanup



**ECP benchmark applications

QUDA

- "QCD on CUDA" – http://lattice.github.com/quda (open source, BSD license)
  - Not just CUDA anymore
- Effort started at Boston University in 2008, now in wide use as the GPU backend for BQCD, Chroma**, CPS**, MILC**, TIFR, etc.
- Provides solvers for major fermionic discretizations, pure gauge algorithms, etc.
- Maximize performance
  - Mixed-precision methods
  - Autotuning for high performance on all architectures
  - Multigrid solvers for optimal convergence
  - NVSHMEM for improving strong scaling
- Portable: HIP (merged), SYCL (in review) and OpenMP (in development)
  - **A research tool for how to reach the exascale (and beyond)**
  - **Optimally mapping the problem to hierarchical processors and node topologies**

Challenges

Or: the state of the network

- LQCD simulations are particularly sensitive to network bandwidth

Challenges

Or: the state of the network

- LQCD simulations are particularly sensitive to network bandwidth
- Not all HPC facilities prioritize network bandwidth

Challenges

Or: the state of the network

- LQCD simulations are particularly sensitive to network bandwidth
- Not all HPC facilities prioritize network bandwidth
- Regardless, it’s not always possible (or practical) to control process placement
  - You can’t always take advantage of *all* hierarchies of bandwidths/latencies

Challenges

Or: the state of the network

- LQCD simulations are particularly sensitive to network bandwidth
- Not all HPC facilities prioritize network bandwidth
- Regardless, it's not always possible (or practical) to control process placement
  - You can't always take advantage of *all* hierarchies of bandwidths/latencies
- **Communication reducing or avoiding algorithms** are increasingly important for mitigating these challenges

Challenges

Or: the state of the network

- LQCD simulations are particularly sensitive to network bandwidth
- Not all HPC facilities prioritize network bandwidth
- Regardless, it's not always possible (or practical) to control process placement
  - You can't always take advantage of *all* hierarchies of bandwidths/latencies
- **Communication reducing or avoiding algorithms** are increasingly important for mitigating these challenges
- Our community has been and continues to be fully aware of this:
  - Communication-reducing solvers
  - Domain-decomposed preconditioners
  - Domain-decomposed HMC

HISQ Crash Course

Why Staggered Fermions?

Aka Kogut-Susskind Fermions

$$D^{stag}_{x,y} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[U_\mu(x)\delta_{x,y-1} - U^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right] + 2m\delta_{x,y}$$

- Staggered fermions
  - Spin-diagonalize the discrete Dirac matrix
  - Lose shift-by-one translational invariance, but preserve a shift-by-two
  - Phases $\eta_\mu(x)$ preserve the Dirac structure

Why Staggered Fermions?

Aka Kogut-Susskind Fermions

$$D_{x,y}^{stag} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[U_\mu(x)\delta_{x,y-1} - U_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right] + 2m\delta_{x,y}$$

- Staggered fermions
  - Spin-diagonalize the discrete Dirac matrix
  - Lose shift-by-one translational invariance, but preserve a shift-by-two
  - Phases $\eta_\mu(x)$ preserve the Dirac structure
- The lack of spin degrees of freedom make them relatively inexpensive
  - Beneficial for all types of bandwidth: memory bandwidth, cache bandwidth, communications bandwidth

Why Staggered Fermions?

Aka Kogut-Susskind Fermions

$$D_{x,y}^{stag} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[U_\mu(x)\delta_{x,y-1} - U_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right] + 2m\delta_{x,y}$$

- Staggered fermions
  - Spin-diagonalize the discrete Dirac matrix
  - Lose shift-by-one translational invariance, but preserve a shift-by-two
  - Phases $\eta_\mu(x)$ preserve the Dirac structure
- The lack of spin degrees of freedom make them relatively inexpensive
  - Beneficial for all types of bandwidth: memory bandwidth, cache bandwidth, communications bandwidth
- In contrast to other discretizations…
  - There's an exact chiral symmetry in contrast to Wilson/clover/twisted/etc
  - There's no extra dimension in contrast to domain wall/Mobius/etc

Why Staggered Fermions?

Aka Kogut-Susskind Fermions

$$D^{stag}_{x,y} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[U_\mu(x)\delta_{x,y-1} - U^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right] + 2m\delta_{x,y}$$

- Staggered fermions
  - Spin-diagonalize the discrete Dirac matrix
  - Lose shift-by-one translational invariance, but preserve a shift-by-two
  - Phases $\eta_\mu(x)$ preserve the Dirac structure
- The lack of spin degrees of freedom make them relatively inexpensive
  - Beneficial for all types of bandwidth: memory bandwidth, cache bandwidth, communications bandwidth
- In contrast to other discretizations…
  - There's an exact chiral symmetry in contrast to Wilson/clover/twisted/etc
  - There's no extra dimension in contrast to domain wall/Mobius/etc
- Like the continuum operator, it's just a symmetric first derivative: **anti-Hermitian** and **normal**

Why Staggered Fermions?

Continued

- Huge secondary benefit: the even/odd preconditioned operator is **Hermitian Positive-Definite**
- Anti-Hermitian + normal: $D_{eo} = -D_{oe}^{\dagger}$

$$\begin{bmatrix} 2m & D_{eo} \\ D_{oe} & 2m \end{bmatrix} \begin{bmatrix} x_e \\ x_o \end{bmatrix} = \begin{bmatrix} b_e \\ b_o \end{bmatrix}$$

$$(4m^2 - D_{eo}D_{oe})x_e = 2mb_e - D_{eo}b_o$$

Why Staggered Fermions?

Continued

- Huge secondary benefit: the even/odd preconditioned operator is **Hermitian Positive-Definite**
- Anti-Hermitian + normal: $D_{eo} = -D_{oe}^{\dagger}$

$$\begin{bmatrix} 2m & D_{eo} \\ D_{oe} & 2m \end{bmatrix} \begin{bmatrix} x_e \\ x_o \end{bmatrix} = \begin{bmatrix} b_e \\ b_o \end{bmatrix}$$

$$(4m^2 - D_{eo} D_{oe}) x_e = 2m b_e - D_{eo} b_o$$

- Obviously, there's no free lunch
  - There is a residual doubling: $2^{d/2}$ doublers (as opposed to 2^d)
  - "Taste-breaking" effects: only one of the "pions" feels the *exact* lattice chiral symmetry

Enter HISQ

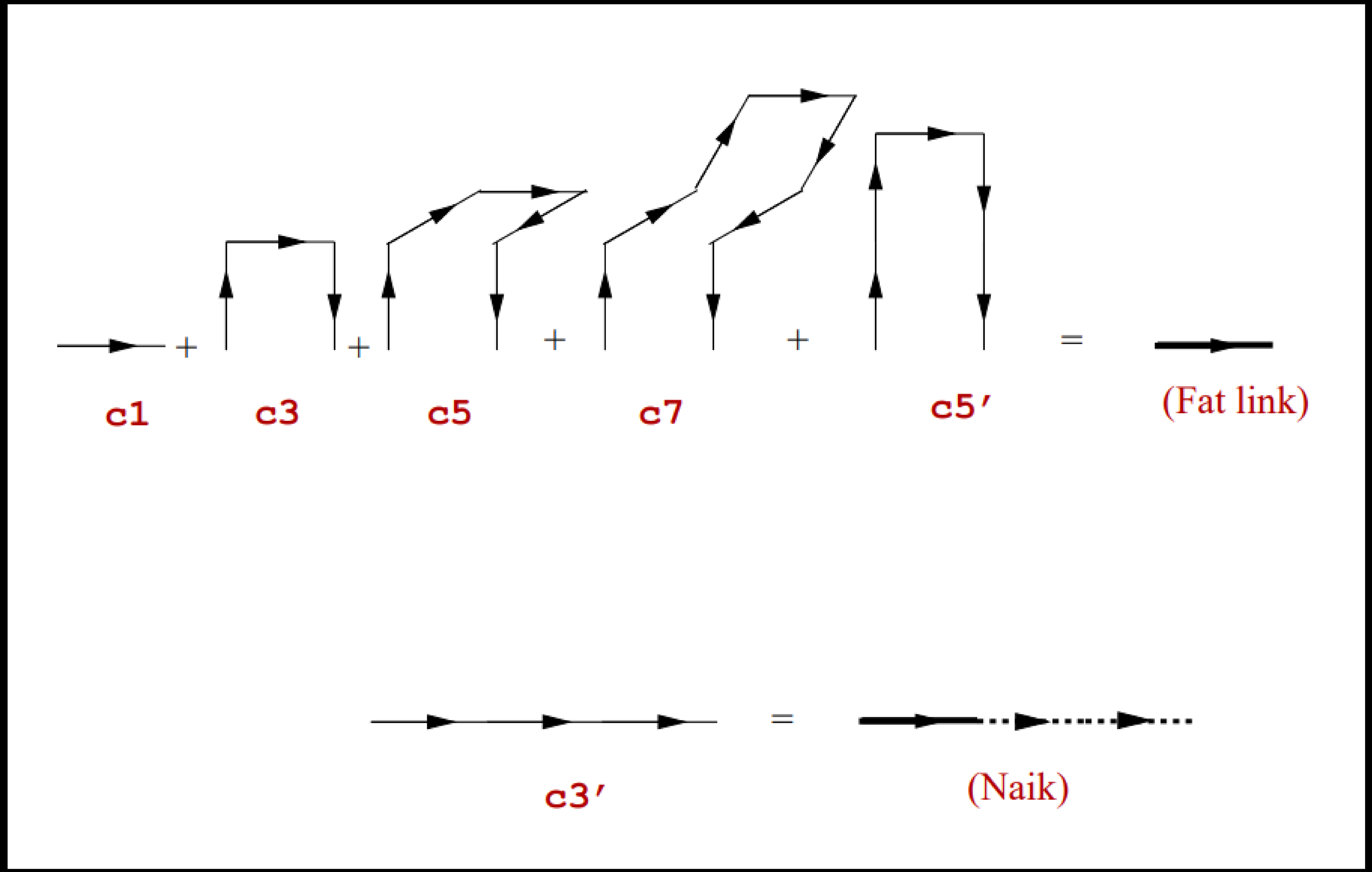
“Highly Improved Staggered Quarks”

- HISQ takes staggered fermions and addresses the issues:
  - Smooths the fields
  - Suppresses taste-breaking effects
  - *Additionally* performs Symanzik improvement

Enter HISQ

"Highly Improved Staggered Quarks"

- HISQ takes staggered fermions and addresses the issues:
  - Smooths the fields
  - Suppresses taste-breaking effects
  - *Additionally* performs Symanzik improvement
- Core kernel: ASQTAD smearing
  - "fat7" + Lepage term to suppress taste-breaking

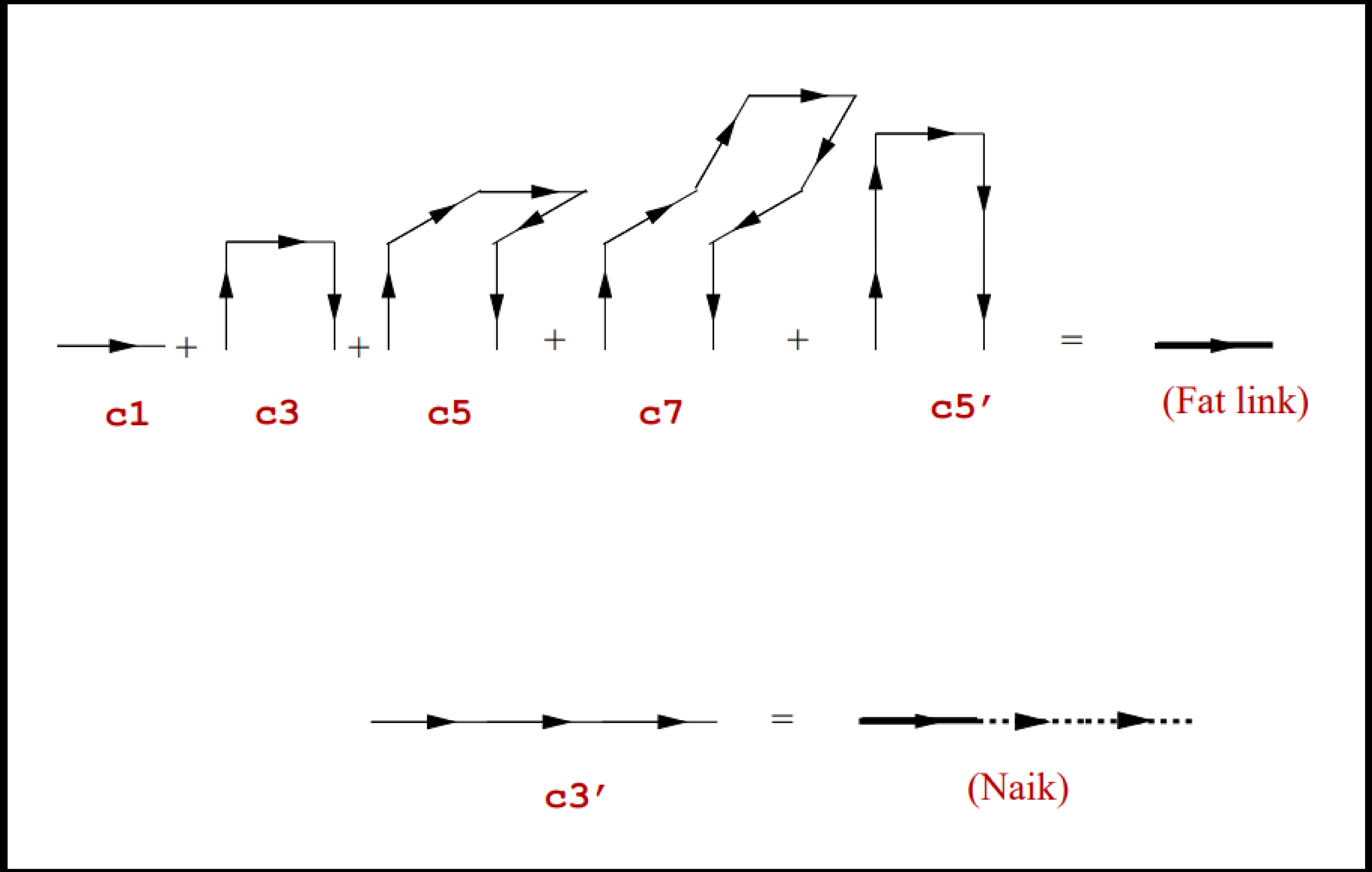


From Follana et al; arxiv:0507011

Enter HISQ

"Highly Improved Staggered Quarks"

- HISQ takes staggered fermions and addresses the issues:
  - Smooths the fields
  - Suppresses taste-breaking effects
  - *Additionally* performs Symanzik improvement
- Core kernel: ASQTAD smearing
  - "fat7" + Lepage term to suppress taste-breaking
- Full workflow: ASQTAD + re-unitarization + ASQTAD
  - Equations can be re-written to remove Lepage term from first step

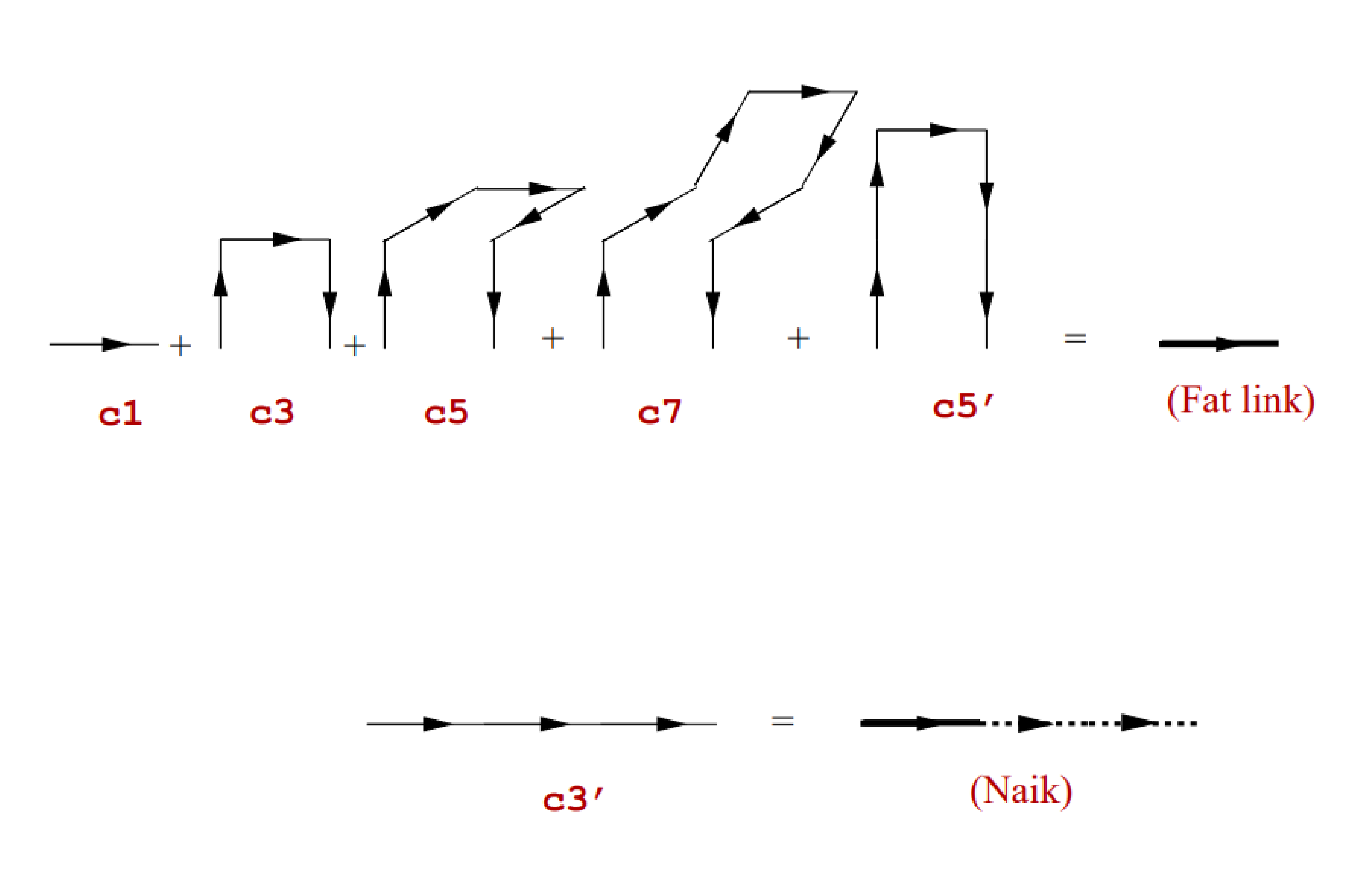


From Follana et al; arxiv:0507011

Enter HISQ

"Highly Improved Staggered Quarks"

- HISQ takes staggered fermions and addresses the issues:
  - Smooths the fields
  - Suppresses taste-breaking effects
  - *Additionally* performs Symanzik improvement
- Core kernel: ASQTAD smearing
  - "fat7" + Lepage term to suppress taste-breaking
- Full workflow: ASQTAD + re-unitarization + ASQTAD
  - Equations can be re-written to remove Lepage term from first step
- Addition of "long links" for Symanzik improvement

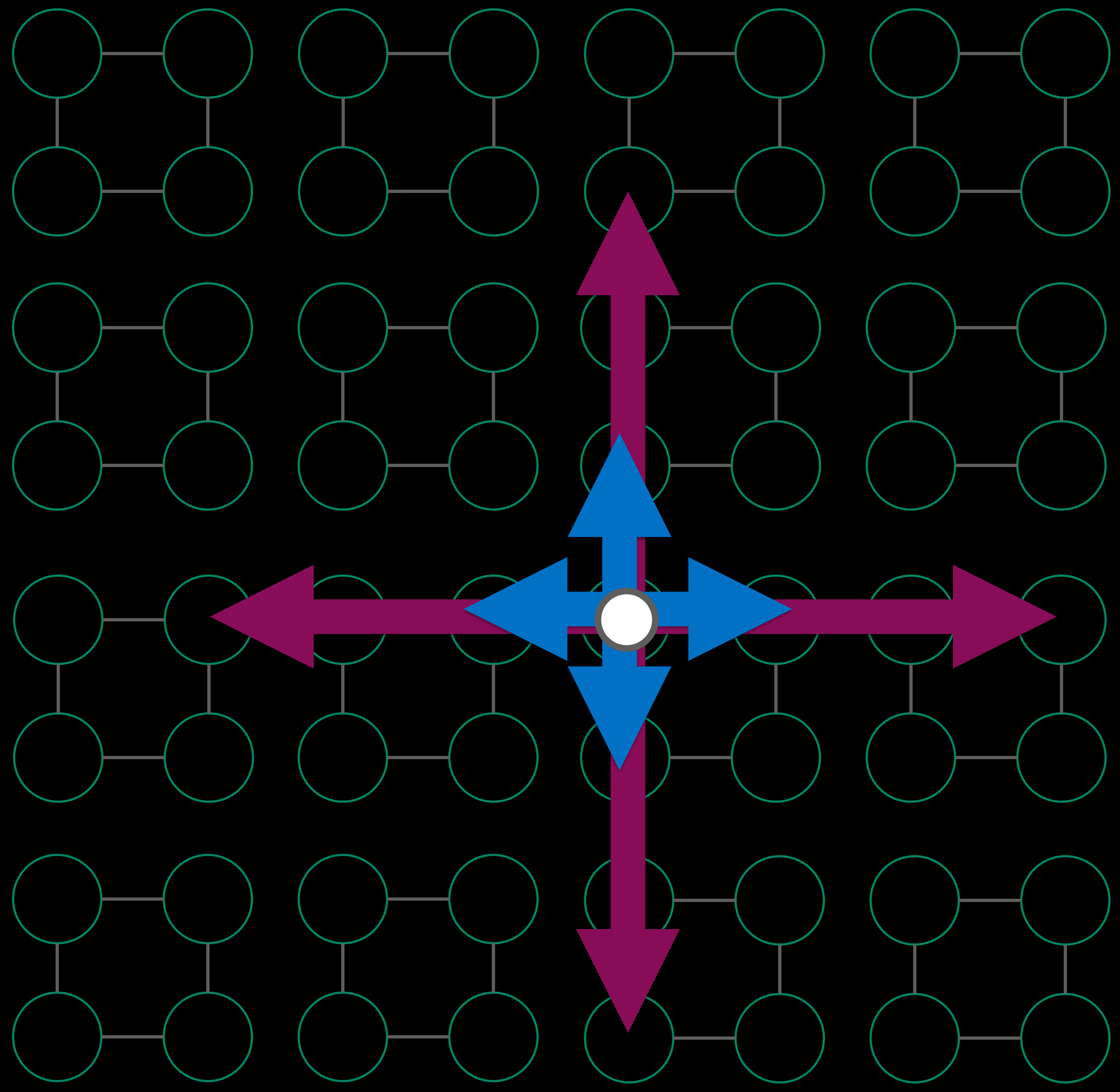


From Follana et al; arxiv:0507011

The HISQ Stencil

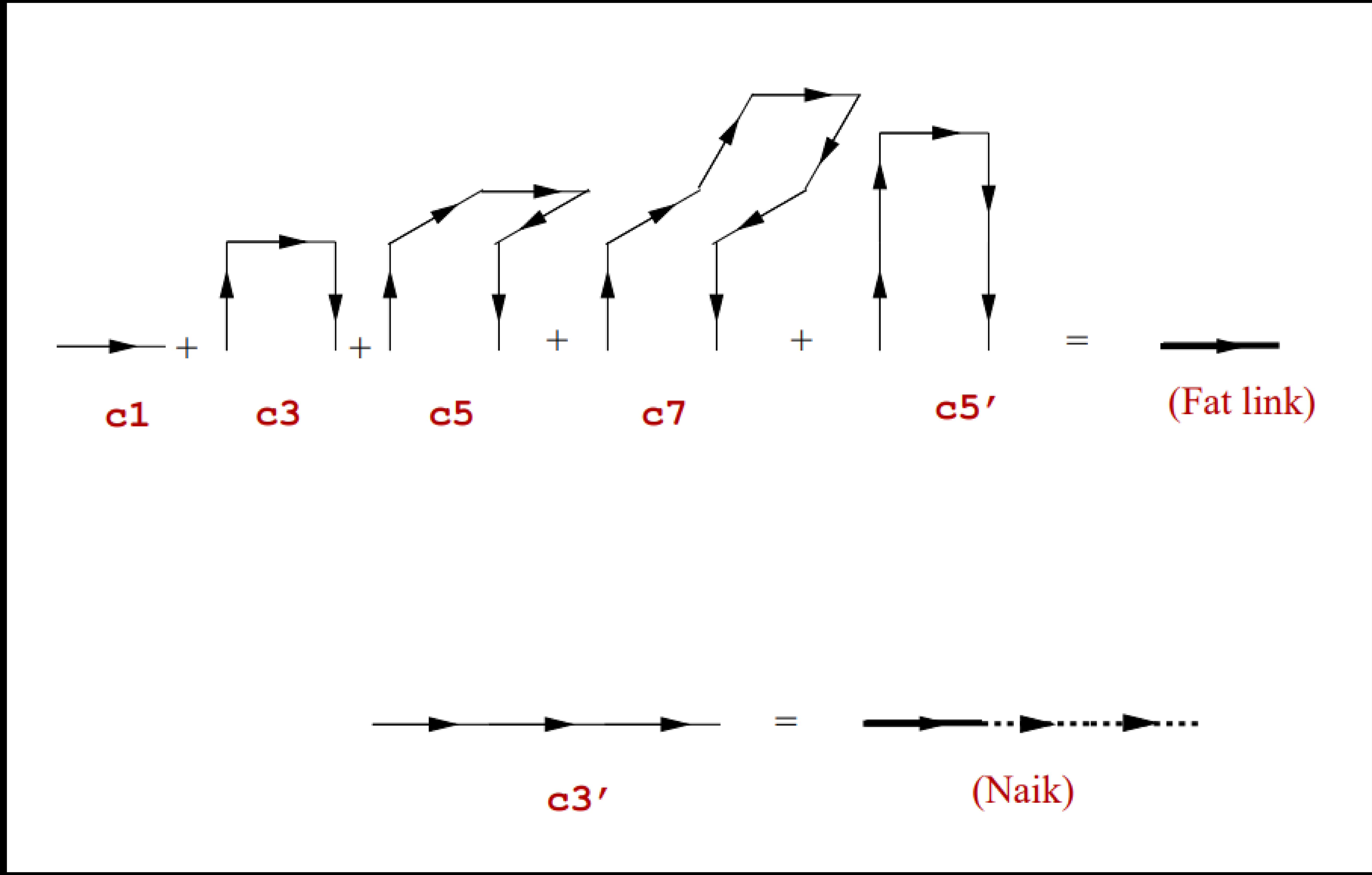
17 points for Lattice Gryffindor

- The final (massive) HISQ stencil is a 17-point stencil
- One local mass term
- Eight distance-1 “fat link” terms: “general” **Nc x Nc** matrices
- Eight distance-3 “long link” terms: **U(Nc)** matrices



Recursive Link Fattening

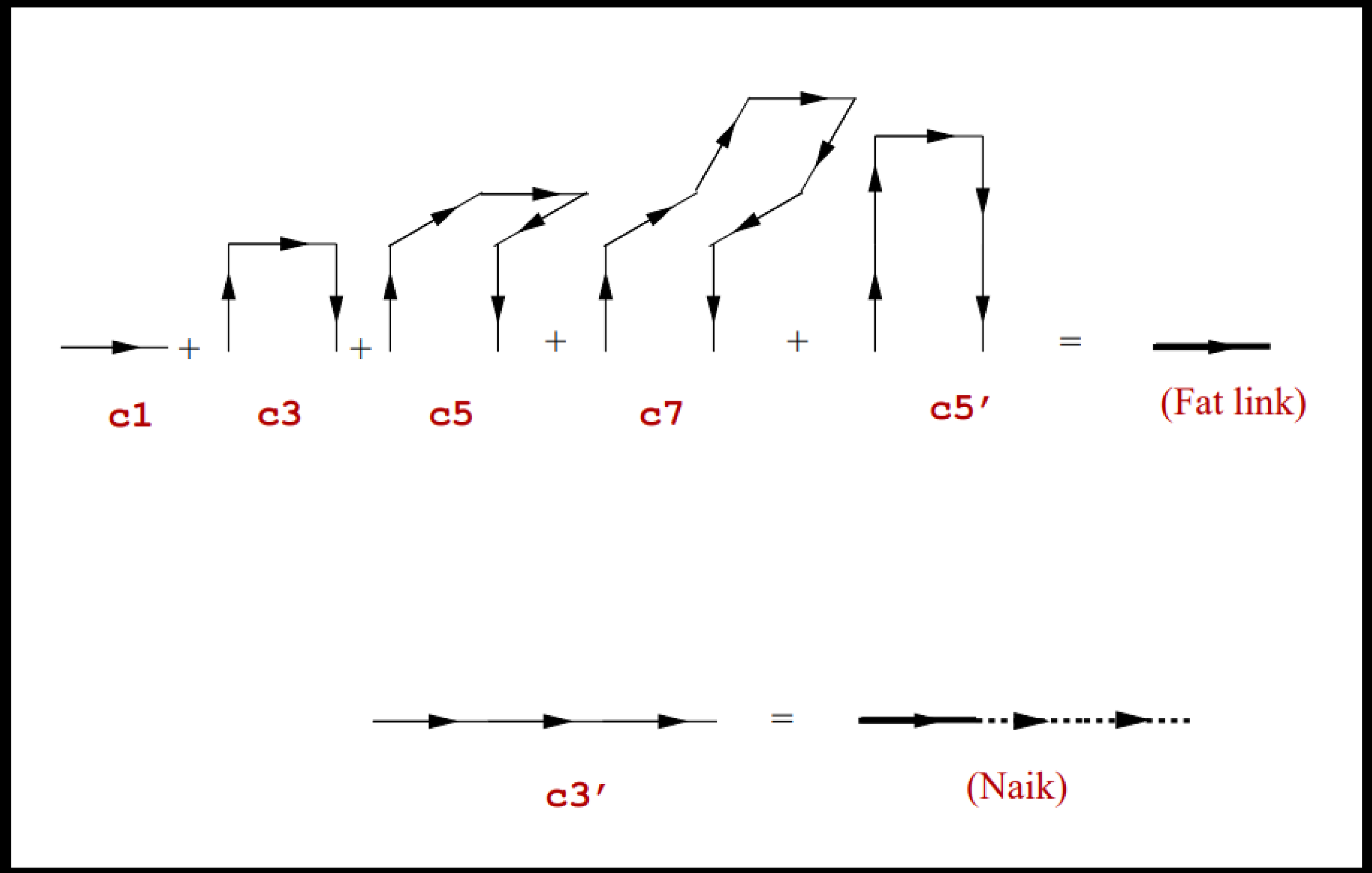
- Constructing the fat links is inherently recursive
- 3-link terms are built from 1-link terms
- 5-link terms can be built from 3-link terms
  - As can the Lepage (c5') staple
- 7-link terms can be built from 5-link terms



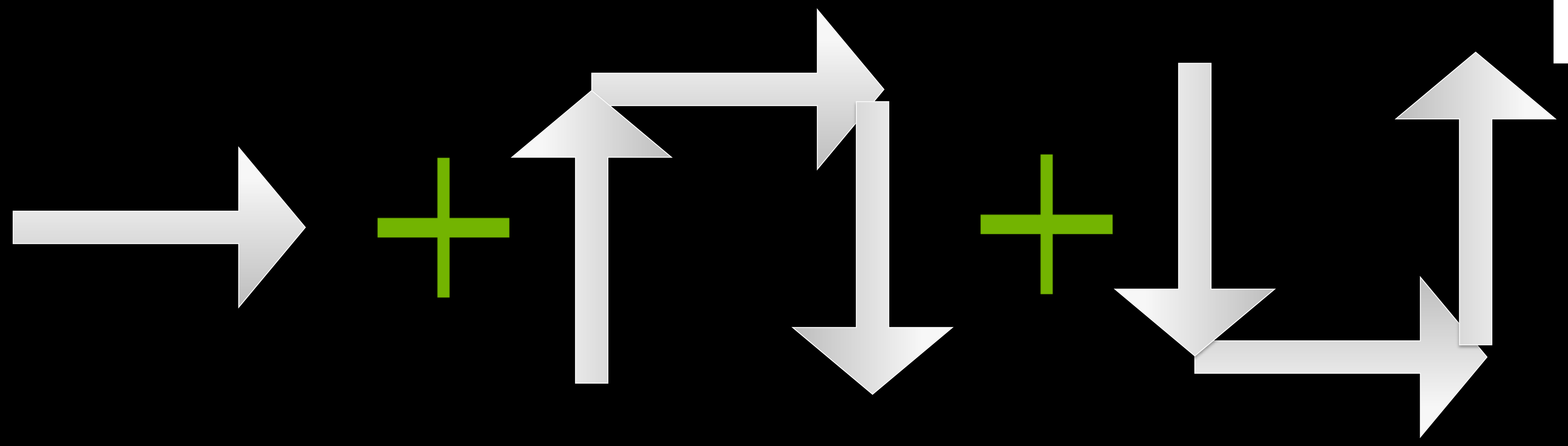
From Follana et al; arxiv:0507011

Data Reuse

Caches exist for a reason

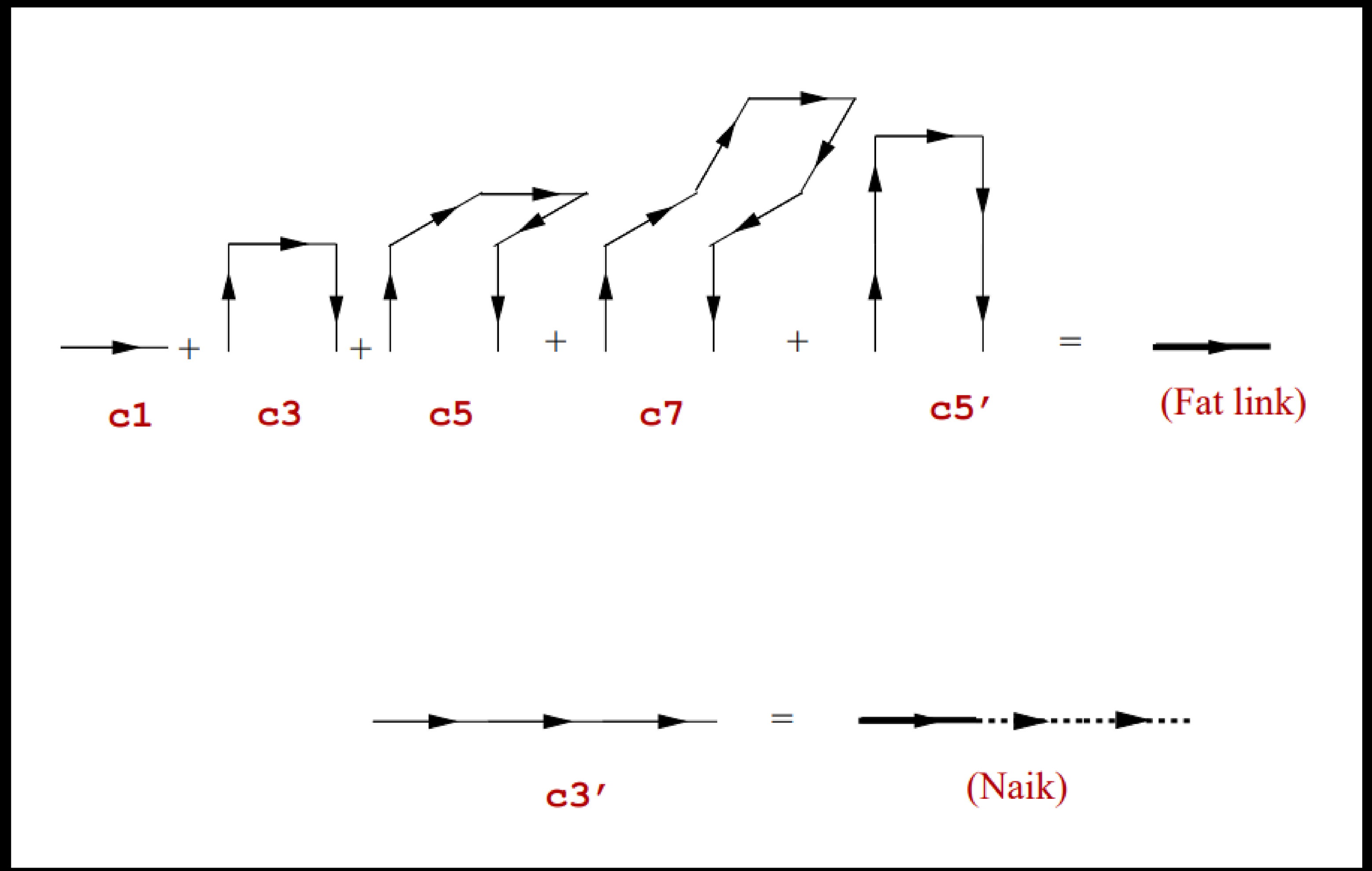


- In one kernel: accumulate c1 + c3, store c3 term



Save this sum to a temporary accumulator

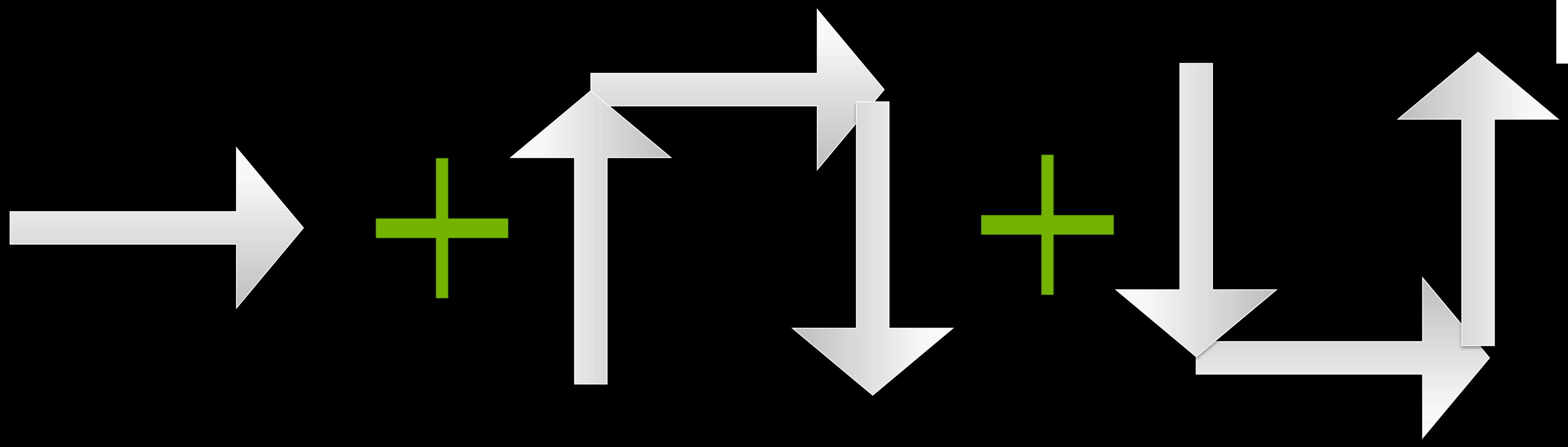
Save each length three staple



Data Reuse

Caches exist for a reason

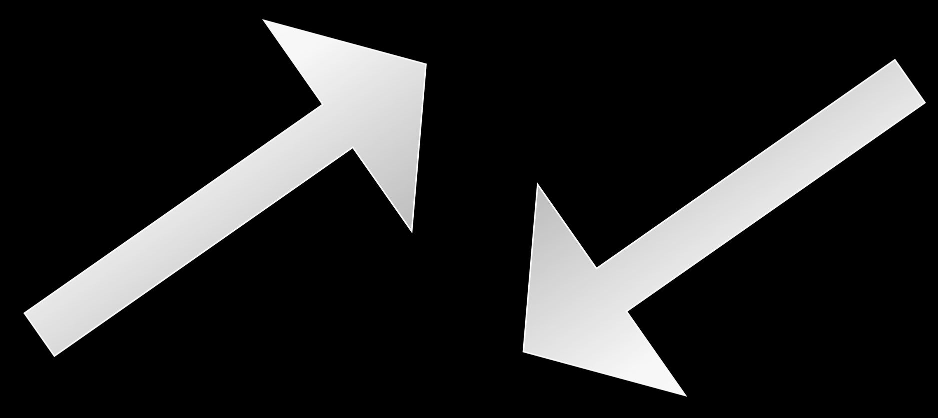
- In one kernel: accumulate c1 + c3, store c3 term



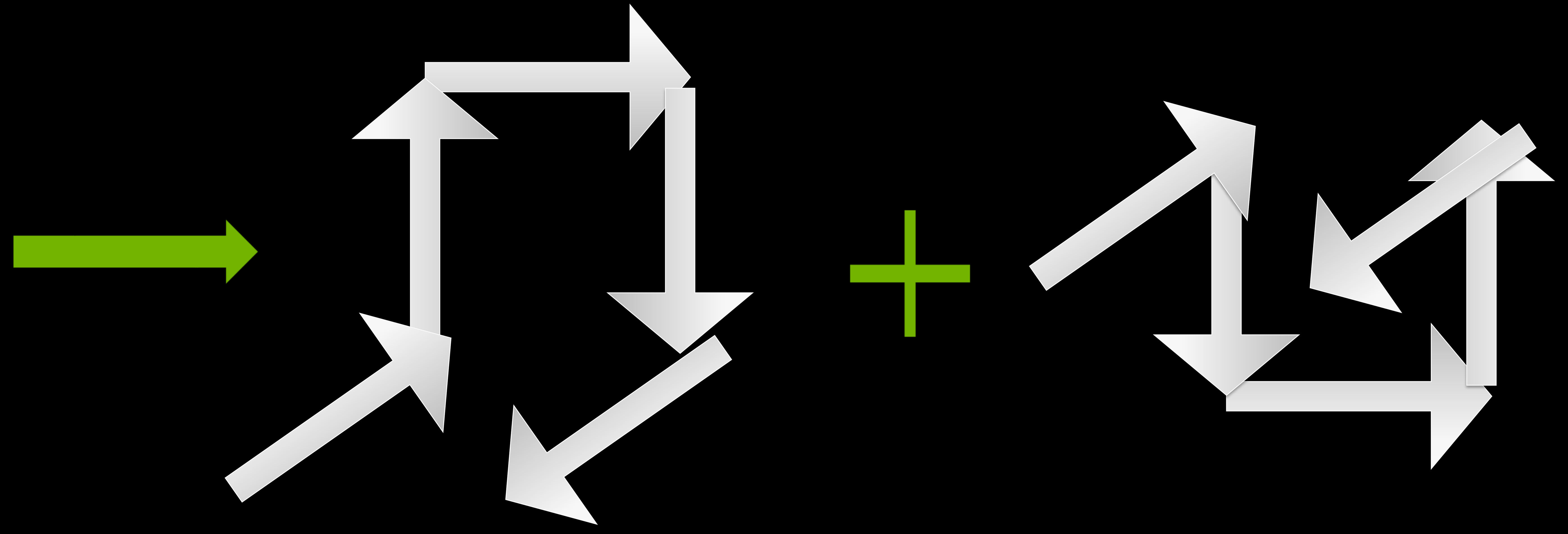
Save this sum to a temporary accumulator

Save each length three staple

- In the next kernel: load gauge links, load two staples, construct five-link terms, accumulate c5s into force, save five-link terms



Load these two links

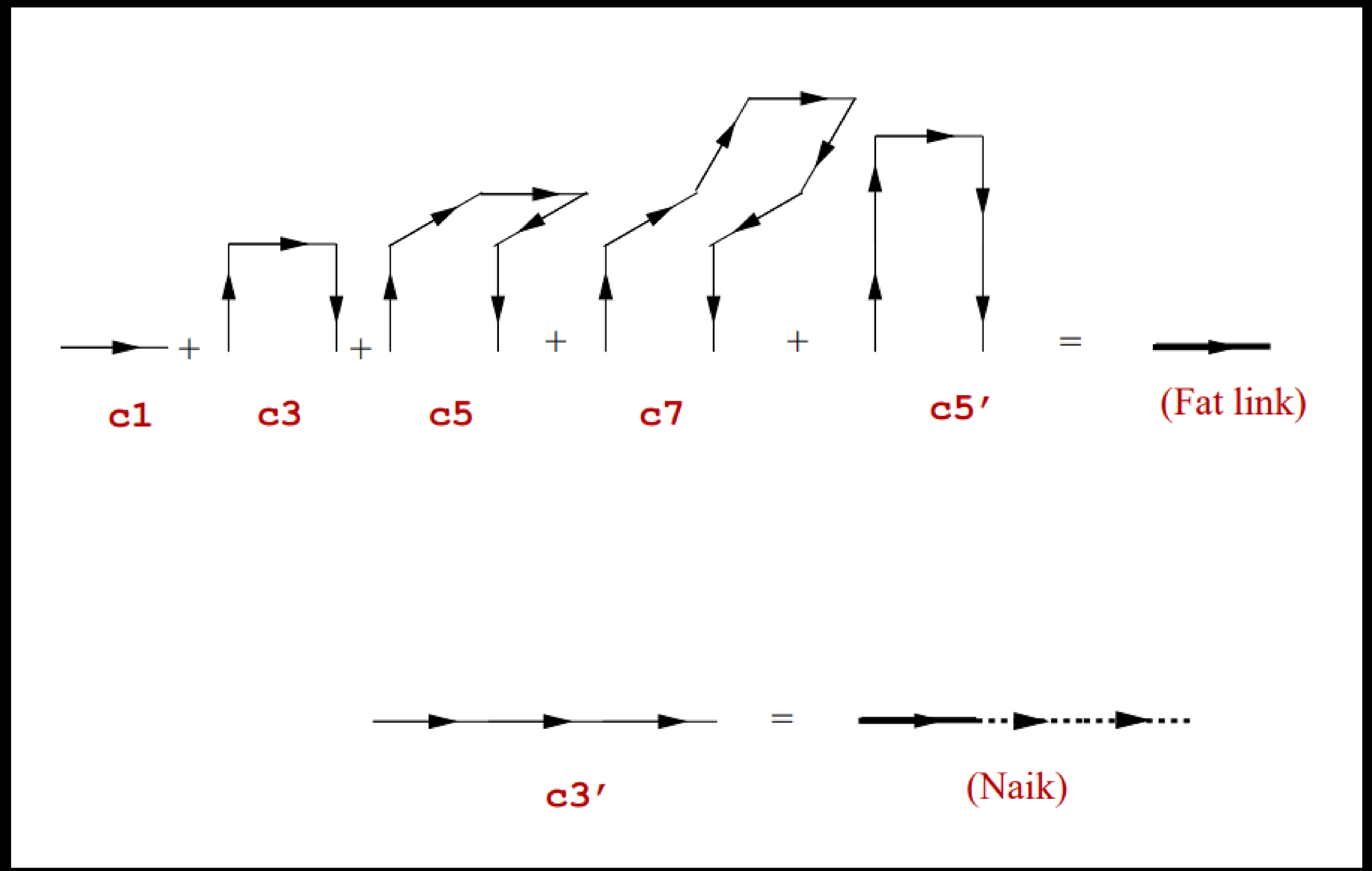


Increment five-link staples into accumulator

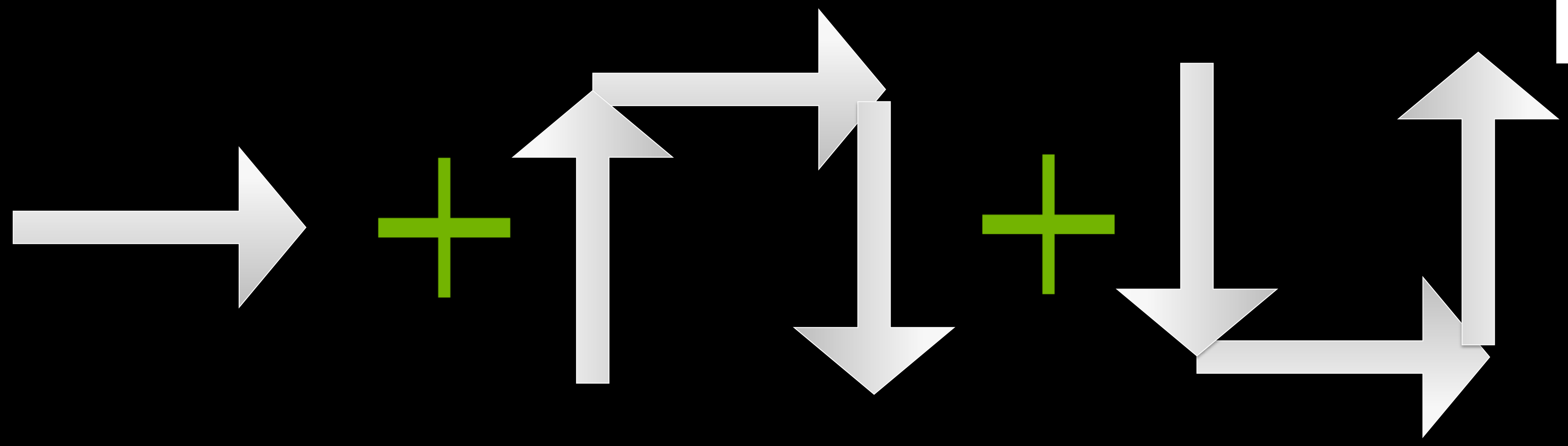
Save each five-link staple separately

Data Reuse

Caches exist for a reason



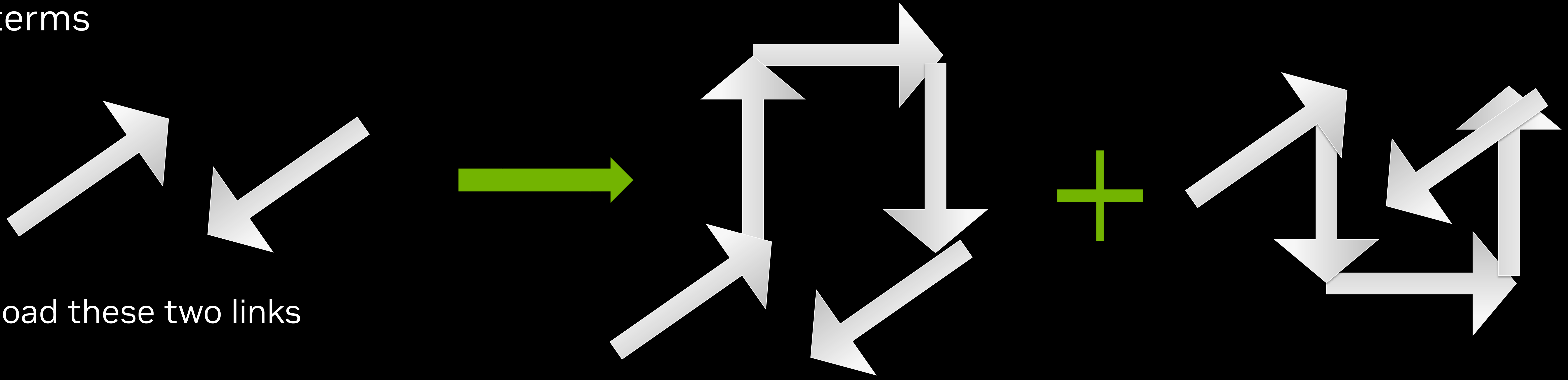
- In one kernel: accumulate c1 + c3, store c3 term



Save this sum to a temporary accumulator

Save each length three staple

- In the next kernel: load gauge links, load two staples, construct five-link terms, accumulate c5s into force, save five-link terms



Load these two links

Increment five-link staples into accumulator

Save each five-link staple separately

- …etc

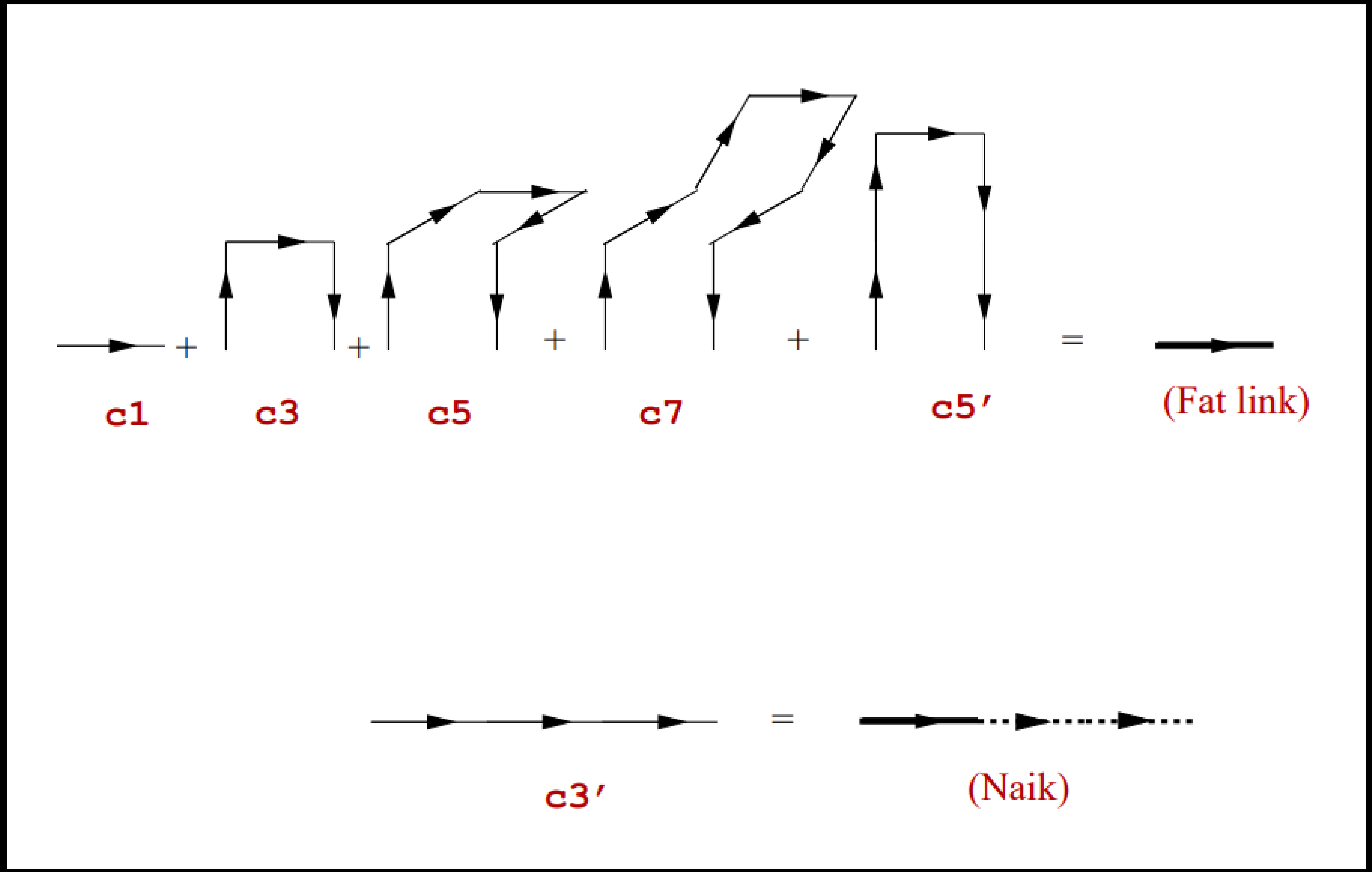
HISQ Force

HISQ Force

"Highly Improved Staggered Quarks"

- The HISQ force is a beast: three-stage chain rule
- Similar to the fat link construction, there are a lot of opportunities for…
  - Reuse of intermediates
  - Kernel fusion
  - **Cache Reuse**

$$\frac{d}{dU}$$



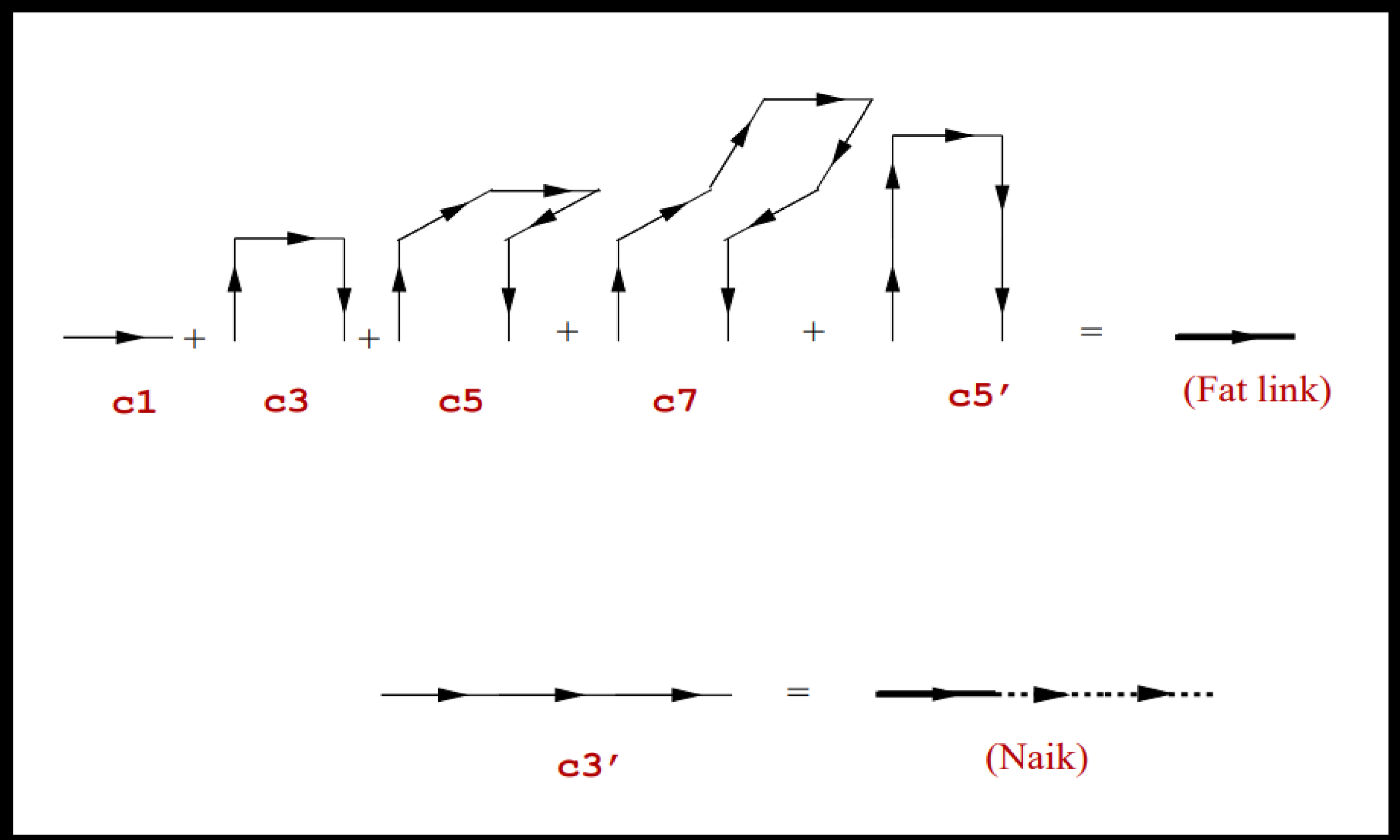
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward



End loop (sig)
```

$$\frac{d}{dU}$$



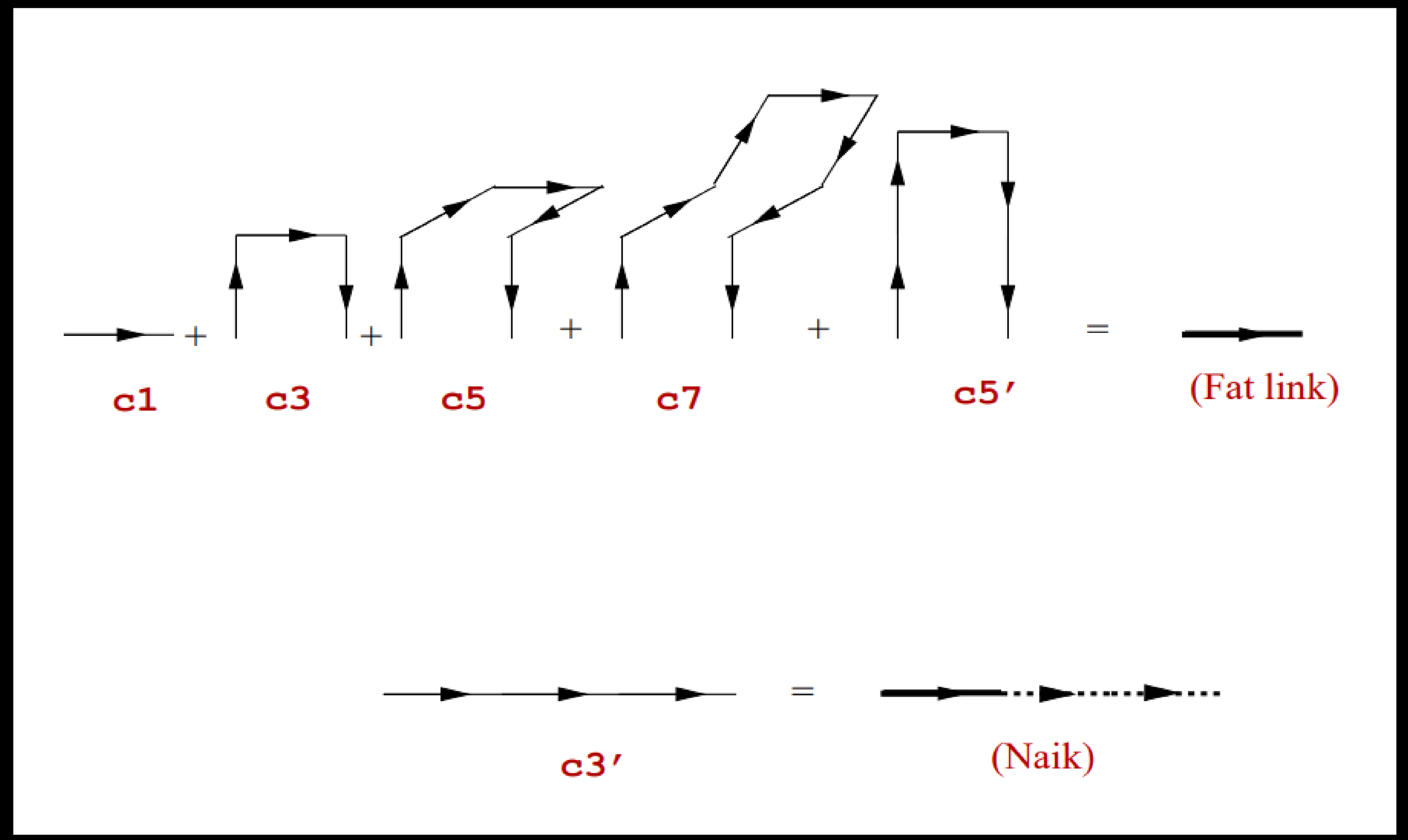
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward









  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



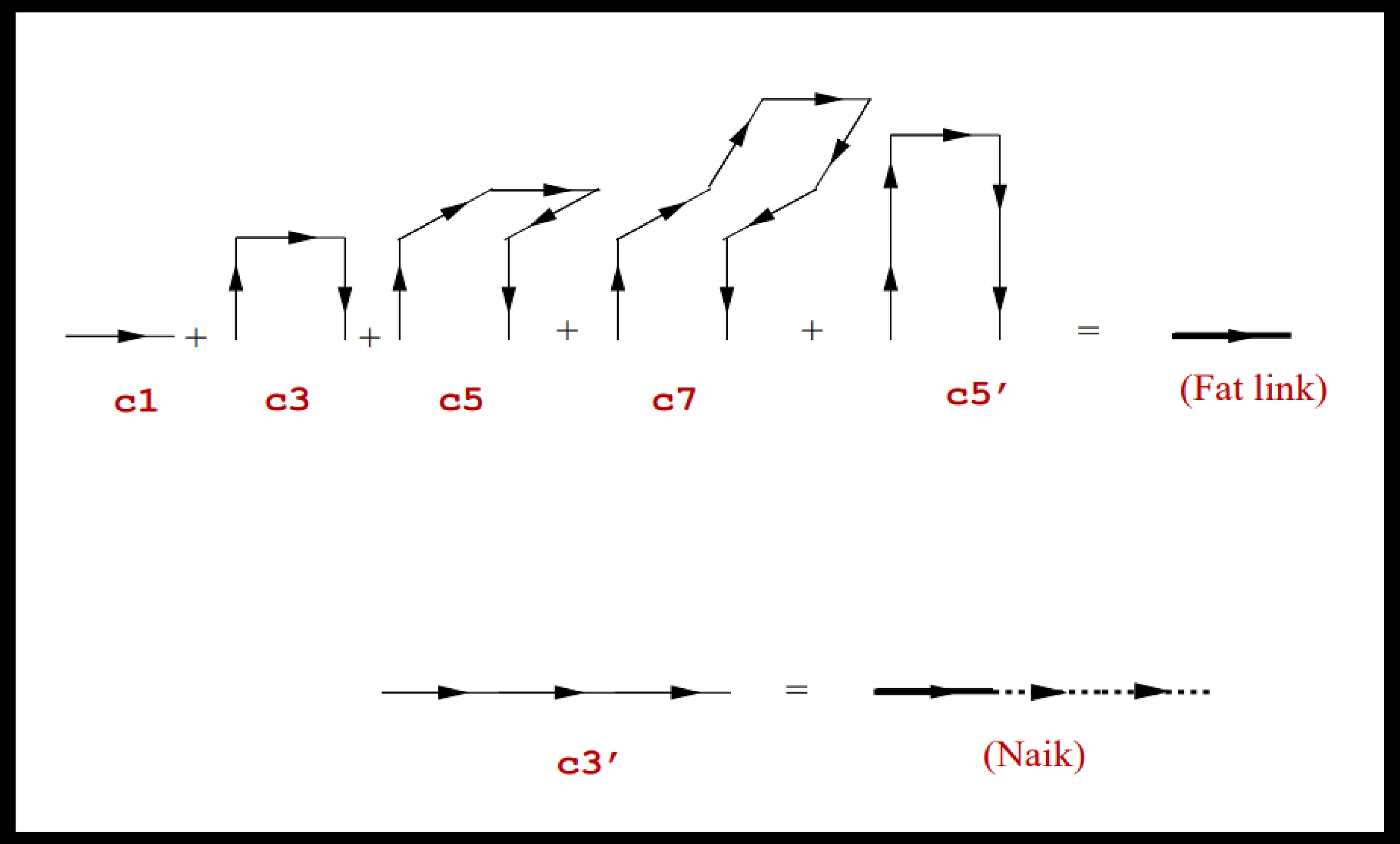
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force: Accumulate and store intermediates




  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



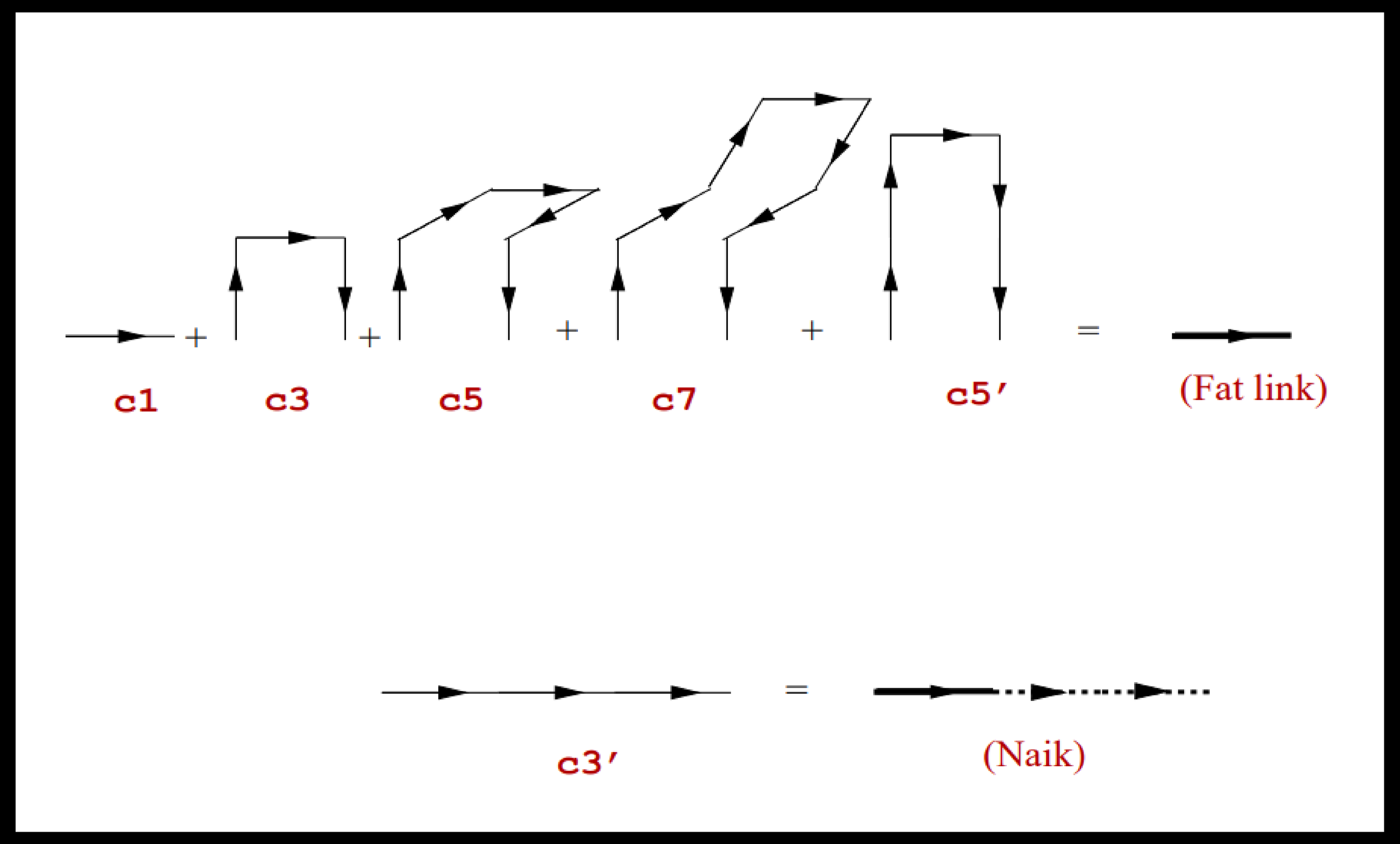
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force
    Loop over nu != |sig|,|mu|; forward/backward
      Compute sig,mu,nu 5-link middle force: reuse intermediates from before




    End loop (nu)



  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



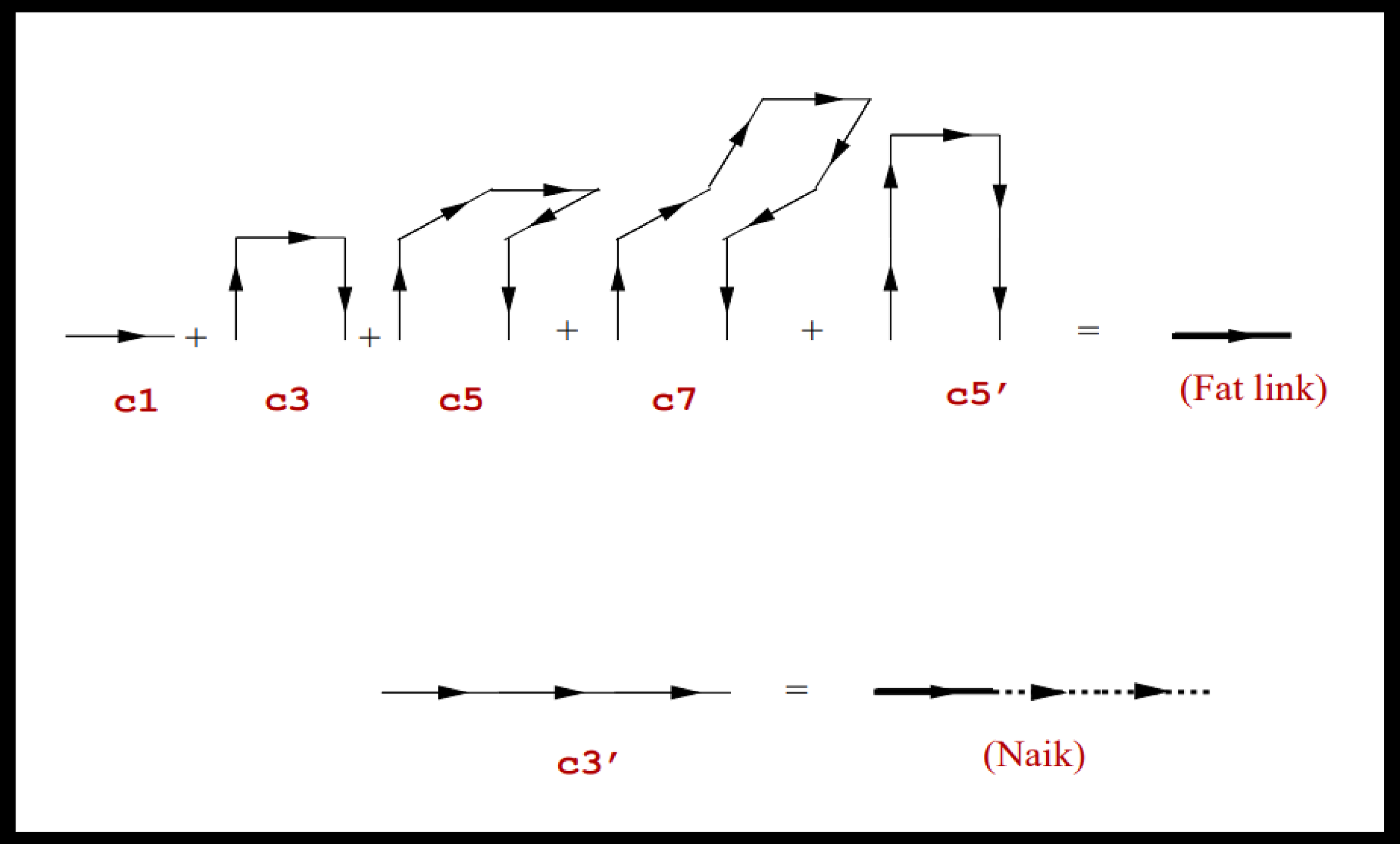
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force
    Loop over nu != |sig|,|mu|; forward/backward
      Compute sig,mu,nu 5-link middle force
      Loop over rho != |sig|,|mu|,|nu|, forward/backward
        Compute sig,mu,nu,rho 7-link middle force, side force
      End loop (rho)

    End loop (nu)



  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



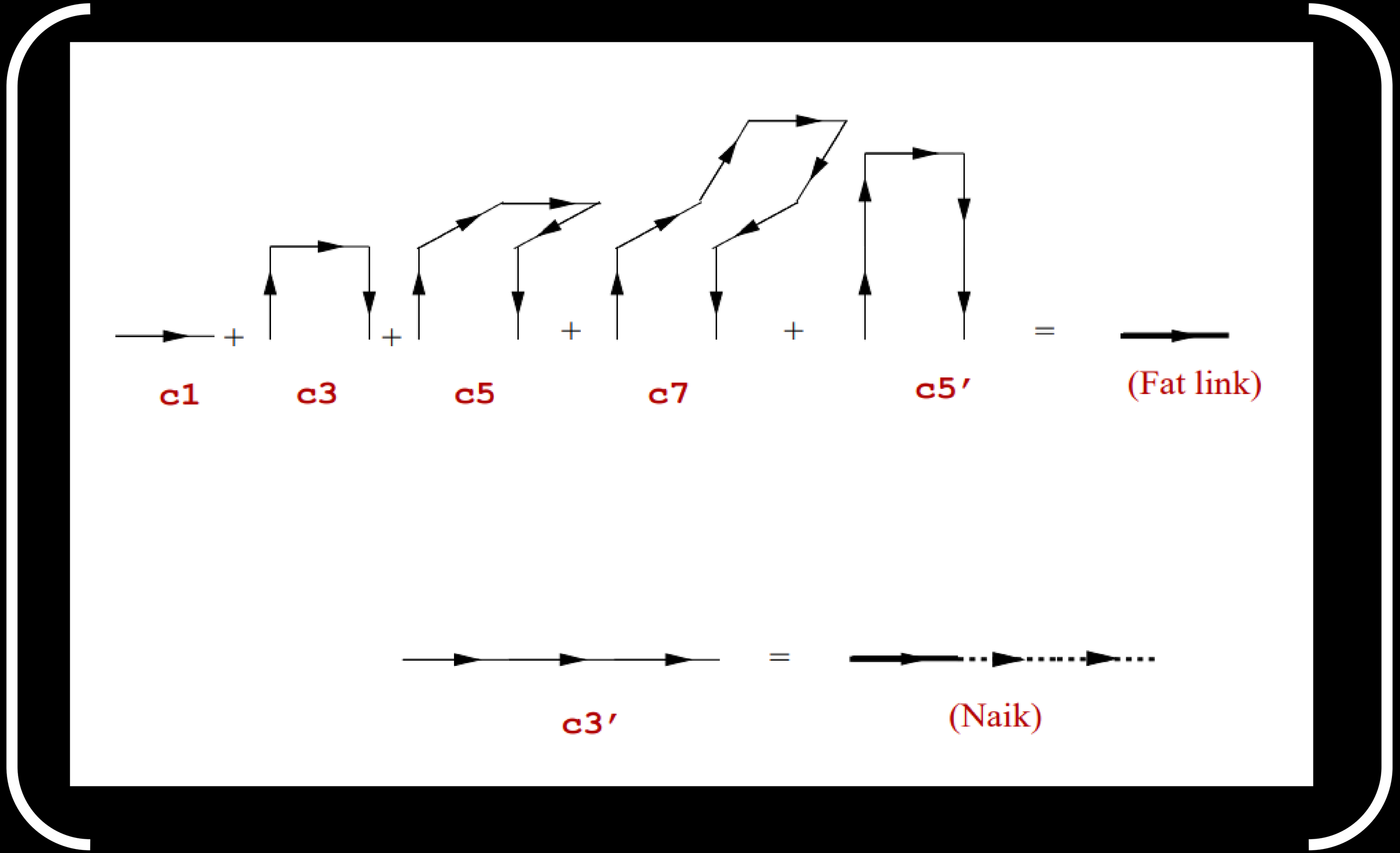
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force
    Loop over nu != |sig|,|mu|; forward/backward
      Compute sig,mu,nu 5-link middle force
      Loop over rho != |sig|,|mu|,|nu|, forward/backward
        Compute sig,mu,nu,rho 7-link middle force, side force
      End loop (rho)
      Compute sig,mu,nu 5-link side force
    End loop (nu)



  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



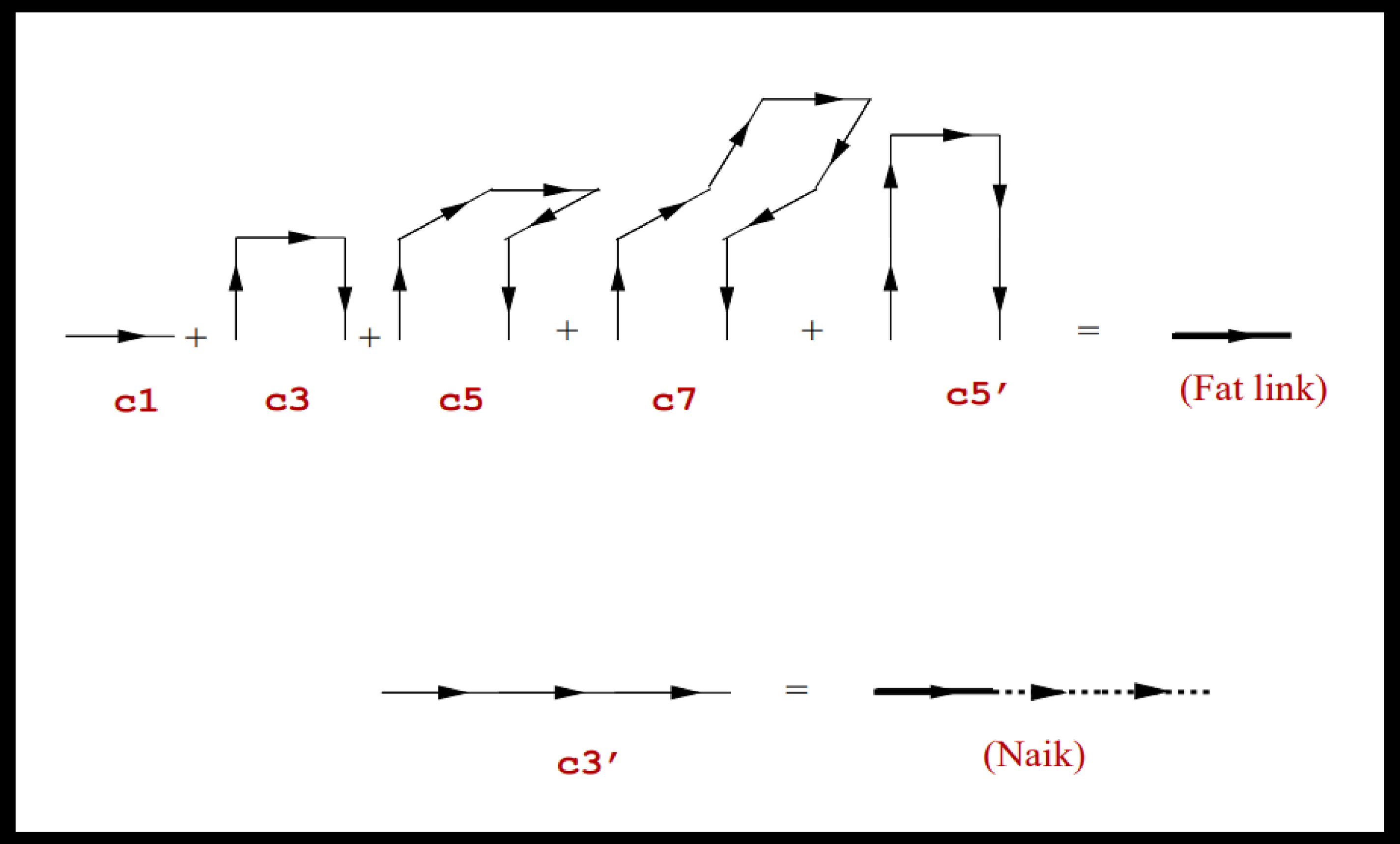
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force
    Loop over nu != |sig|,|mu|; forward/backward
      Compute sig,mu,nu 5-link middle force
      Loop over rho != |sig|,|mu|,|nu|, forward/backward
        Compute sig,mu,nu,rho 7-link middle force, side force
      End loop (rho)
      Compute sig,mu,nu 5-link side force
    End loop (nu)
    Compute sig,mu 3-link side force
    Compute sig,mu Lepage middle force
    Compute sig,mu Lepage side force
  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



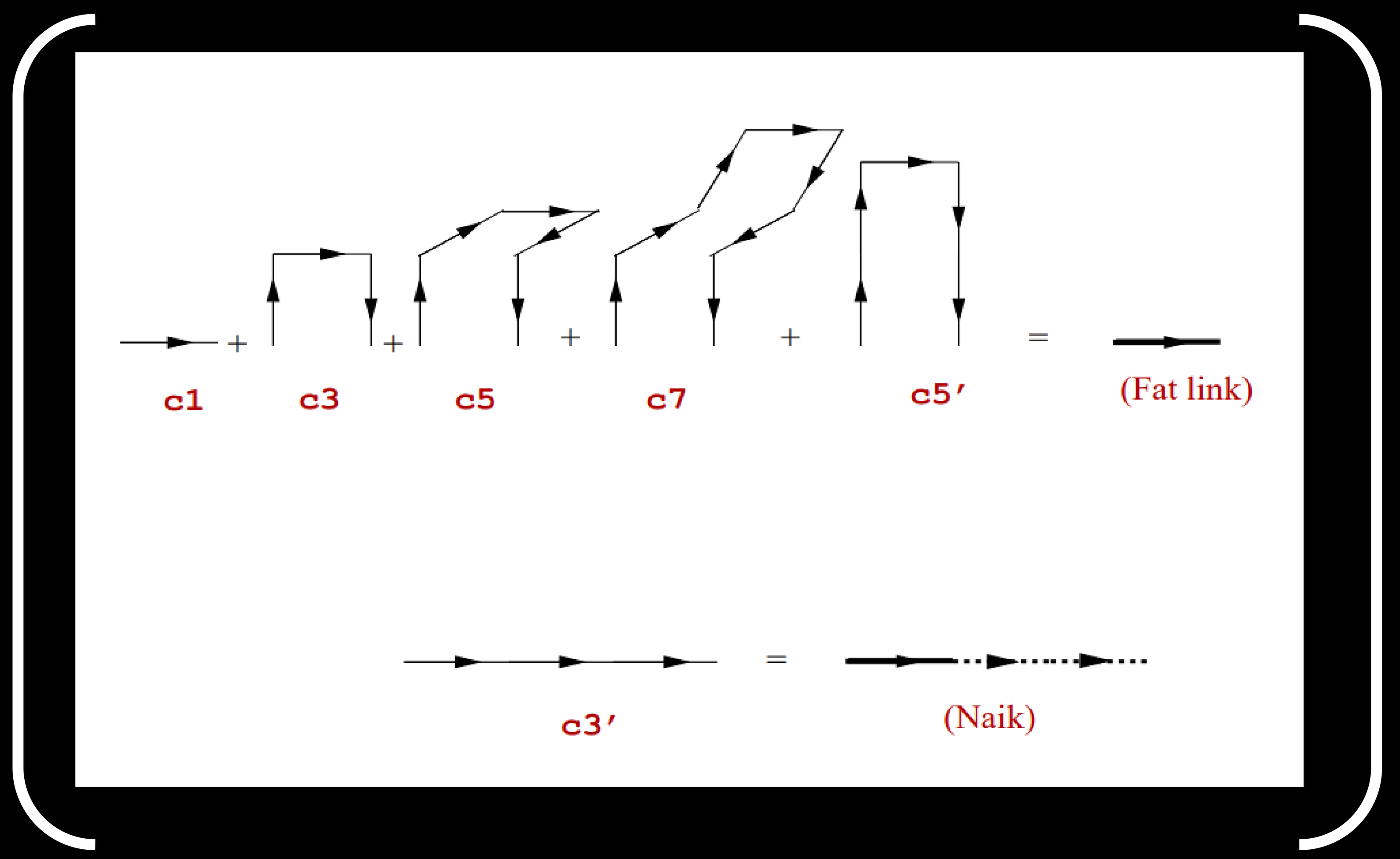
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force
    Loop over nu != |sig|,|mu|; forward/backward
      Compute sig,mu,nu 5-link middle force
      Loop over rho != |sig|,|mu|,|nu|, forward/backward
        Compute sig,mu,nu,rho 7-link middle force, side force
      End loop (rho)
      Compute sig,mu,nu 5-link side force
    End loop (nu)
    Compute sig,mu 3-link side force
    Compute sig,mu Lepage middle force
    Compute sig,mu Lepage side force
  End loop (mu)
End loop (sig)
```

$\frac{d}{dU}$



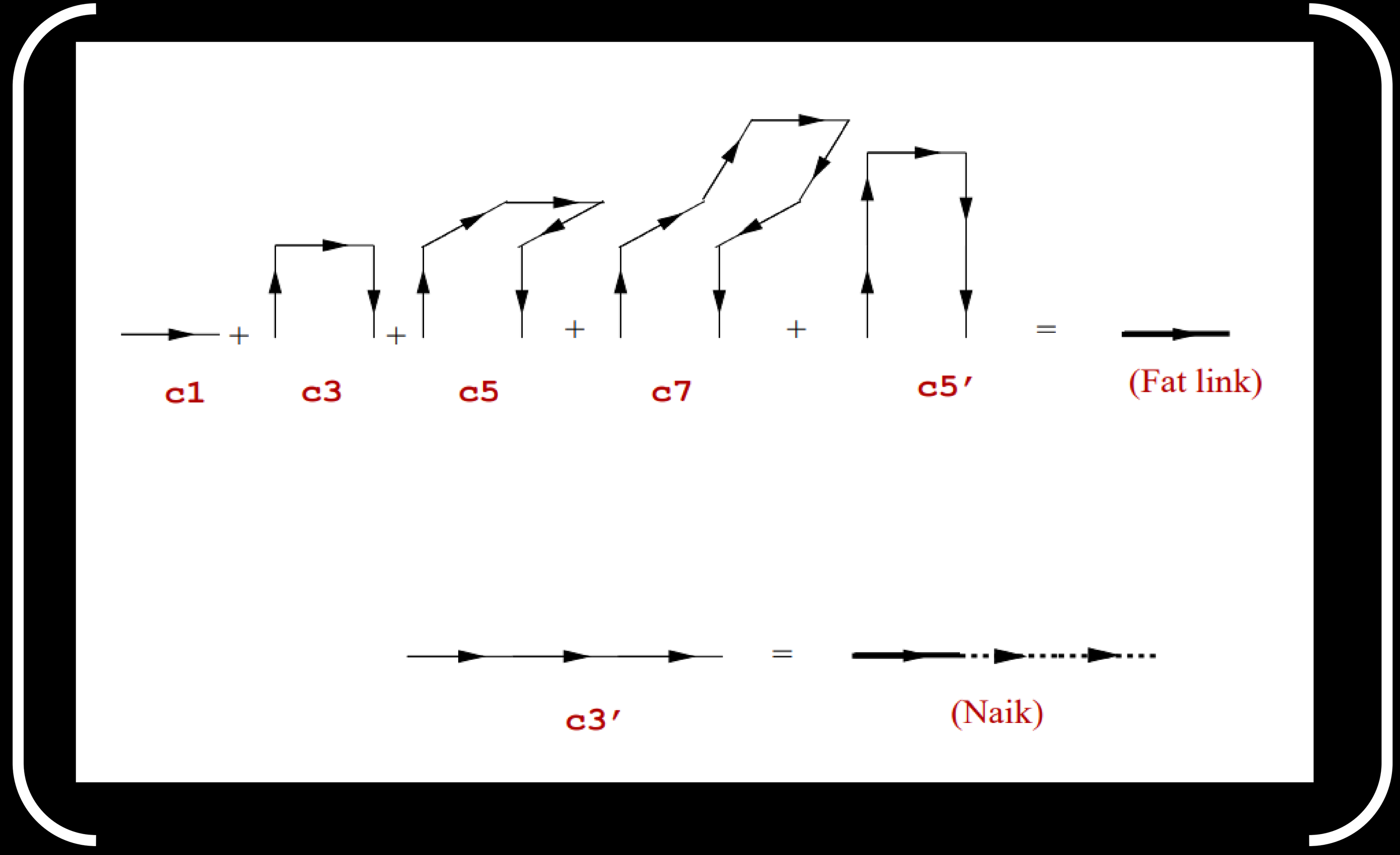
HISQ Force

Sorry about the pseudocode

- Original implementation:

```
Loop over sig = {x, y, z, t}; forward/backward
  Loop over mu != |sig|; forward/backward
    Compute sig,mu 3-link middle force
    Loop over nu != |sig|,|mu|; forward/backward
      Compute sig,mu,nu 5-link middle force
      Loop over rho != |sig|,|mu|,|nu|, forward/backward
        Compute sig,mu,nu,rho 7-link middle force, side force
      End loop (rho)
      Compute sig,mu,nu 5-link side force... + next middle force
    End loop (nu)
    Compute sig,mu 3-link side force
    Compute sig,mu Lepage middle force
    Compute sig,mu Lepage side force
  End loop (mu)
End loop (sig)
```

$$\frac{d}{dU}$$



HISQ Force

Sorry about the pseudocode

- Original implementation:

Loop over sig = {x, y, z, t}; forward/backward

  Loop over mu != |sig|; forward/backward

    ~~Compute sig,mu 3-link middle force~~

    Loop over nu != |sig|,|mu|; forward/backward

      ~~Compute sig,mu,nu 5-link middle force~~

      Loop over rho != |sig|,|mu|,|nu|, forward/backward

        Compute sig,mu,nu,rho 7-link middle force, side force

      End loop (rho)

      Compute sig,mu,nu 5-link side force... + next 5-link middle force

    End loop (nu)

    Compute sig,mu 3-link side force

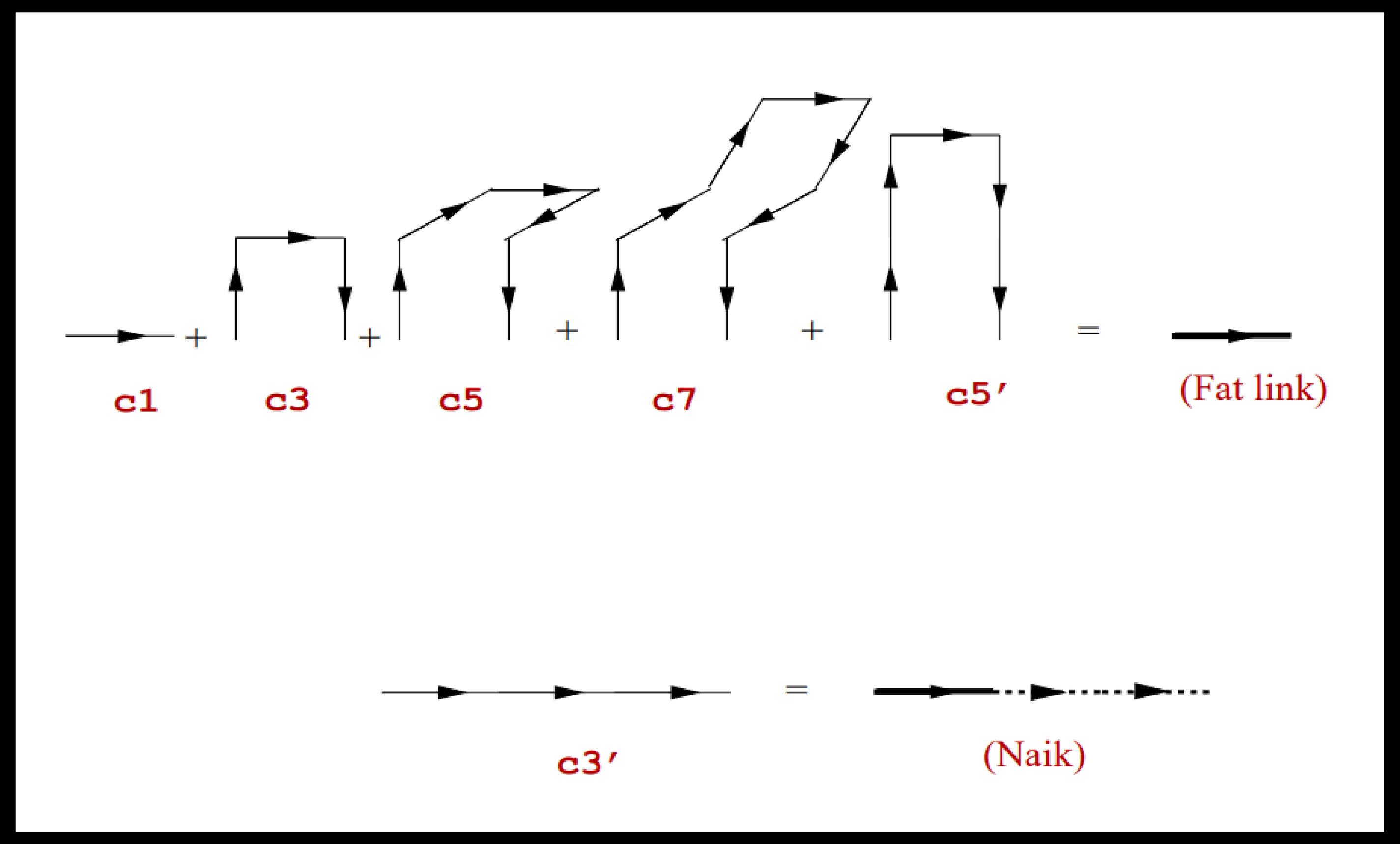
    Compute sig,mu Lepage middle force

    Compute sig,mu Lepage side force... + next 3-link middle force

  End loop (mu)

End loop (sig)

$$\frac{d}{dU}$$

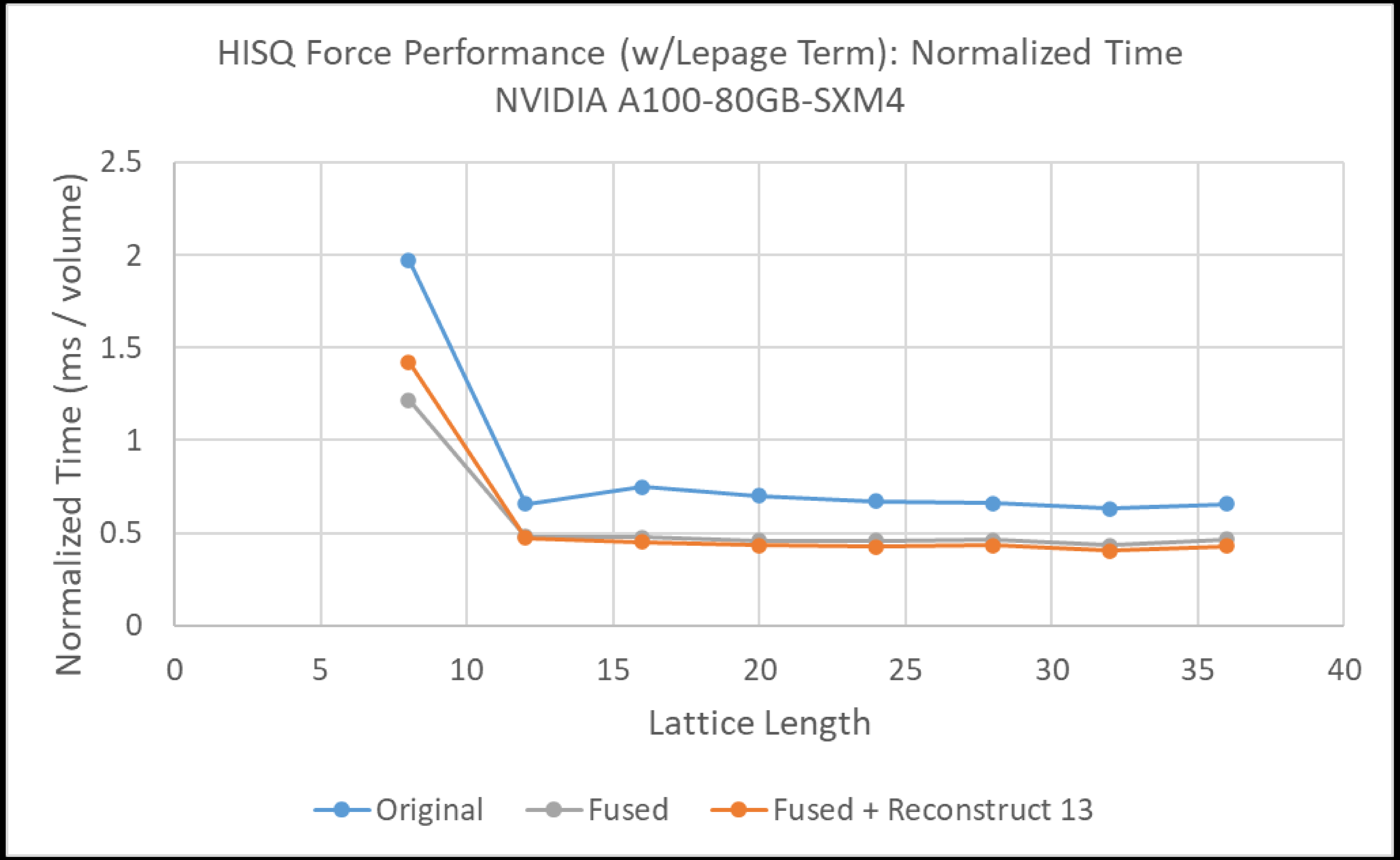


Use Your Symmetries

- While the staples are general matrices, all base gauge links are U(3)
- Take advantage of this symmetry to reduce memory traffic: store as 13 reals, recompute as needed

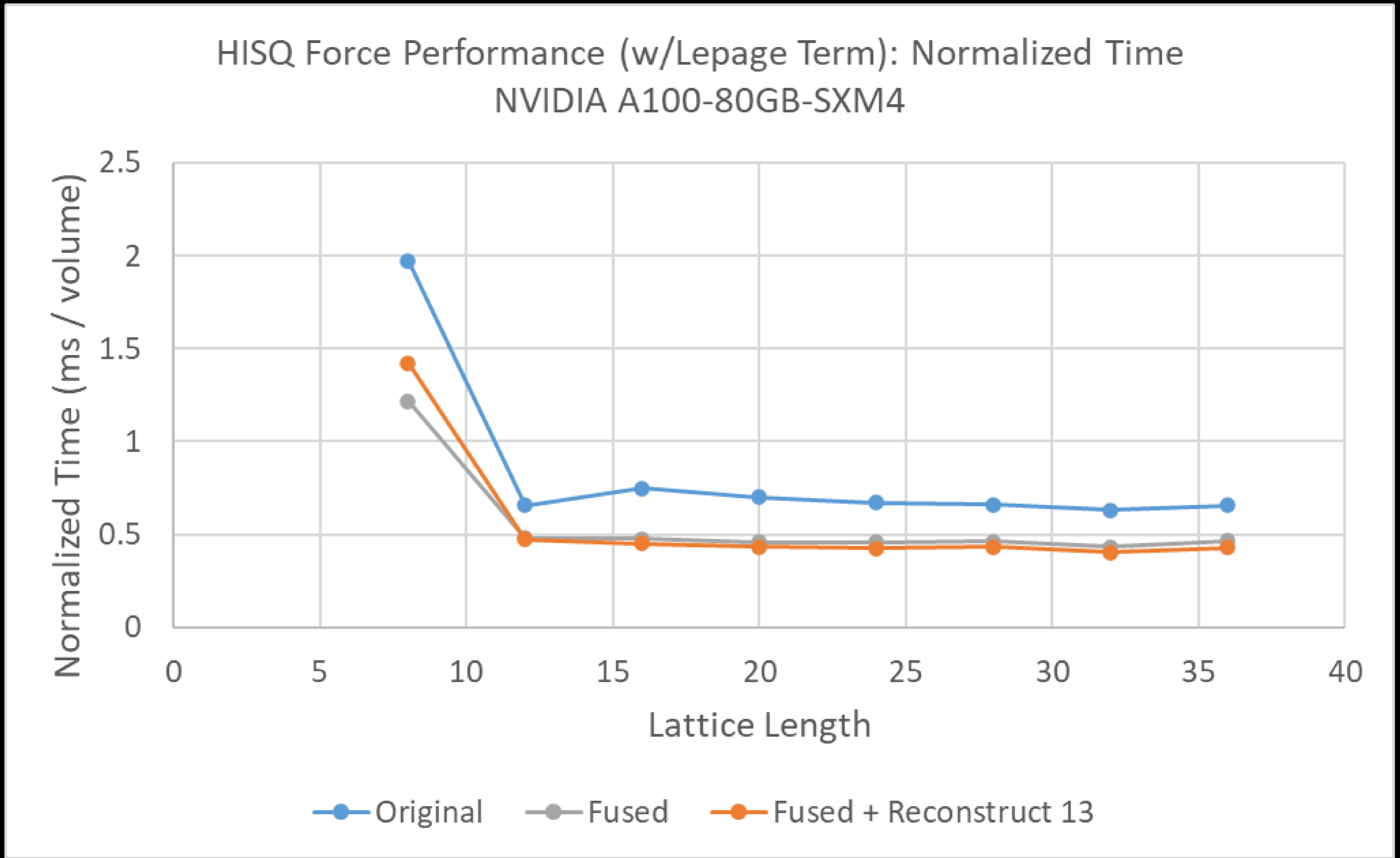
Use Your Symmetries

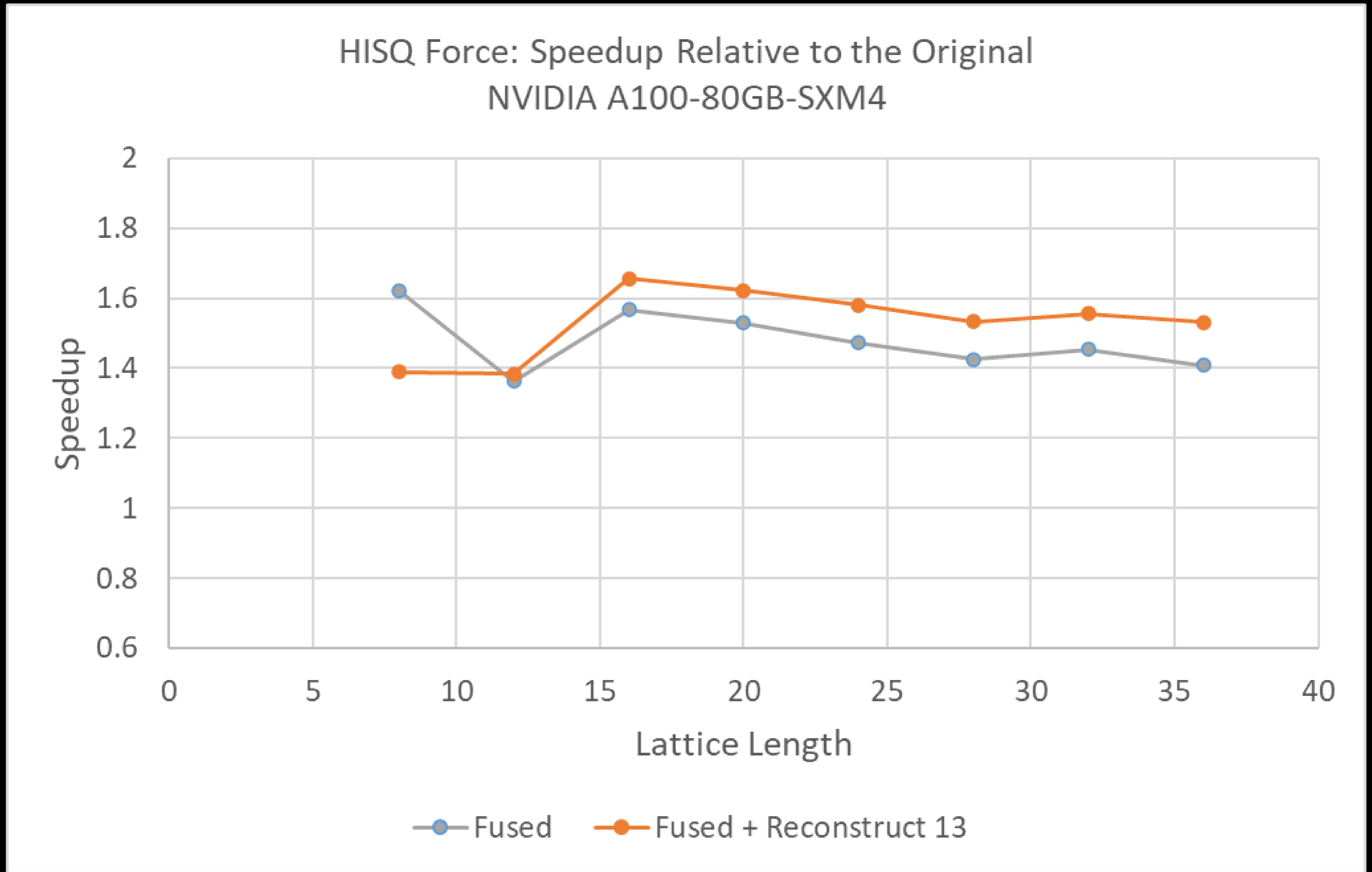
- While the staples are general matrices, all base gauge links are U(3)
- Take advantage of this symmetry to reduce memory traffic: store as 13 reals, recompute as needed



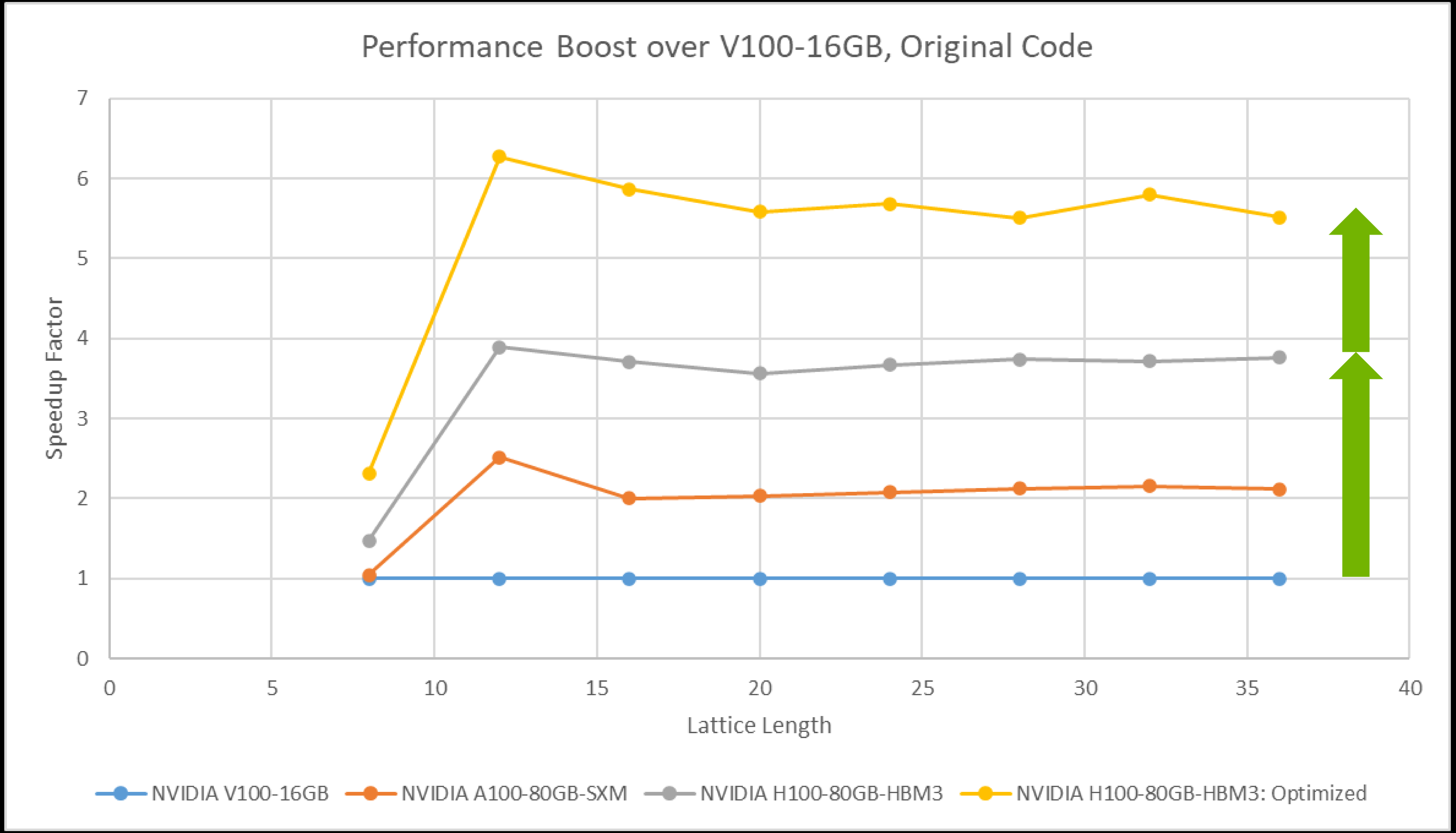
Use Your Symmetries

- While the staples are general matrices, all base gauge links are U(3)
- Take advantage of this symmetry to reduce memory traffic: store as 13 reals, recompute as needed





Improvements are algorithmic and architectural



Algorithm:
1.5x

Architecture:
3.75x

Architecture and algorithm boosts multiply: ~5.6x

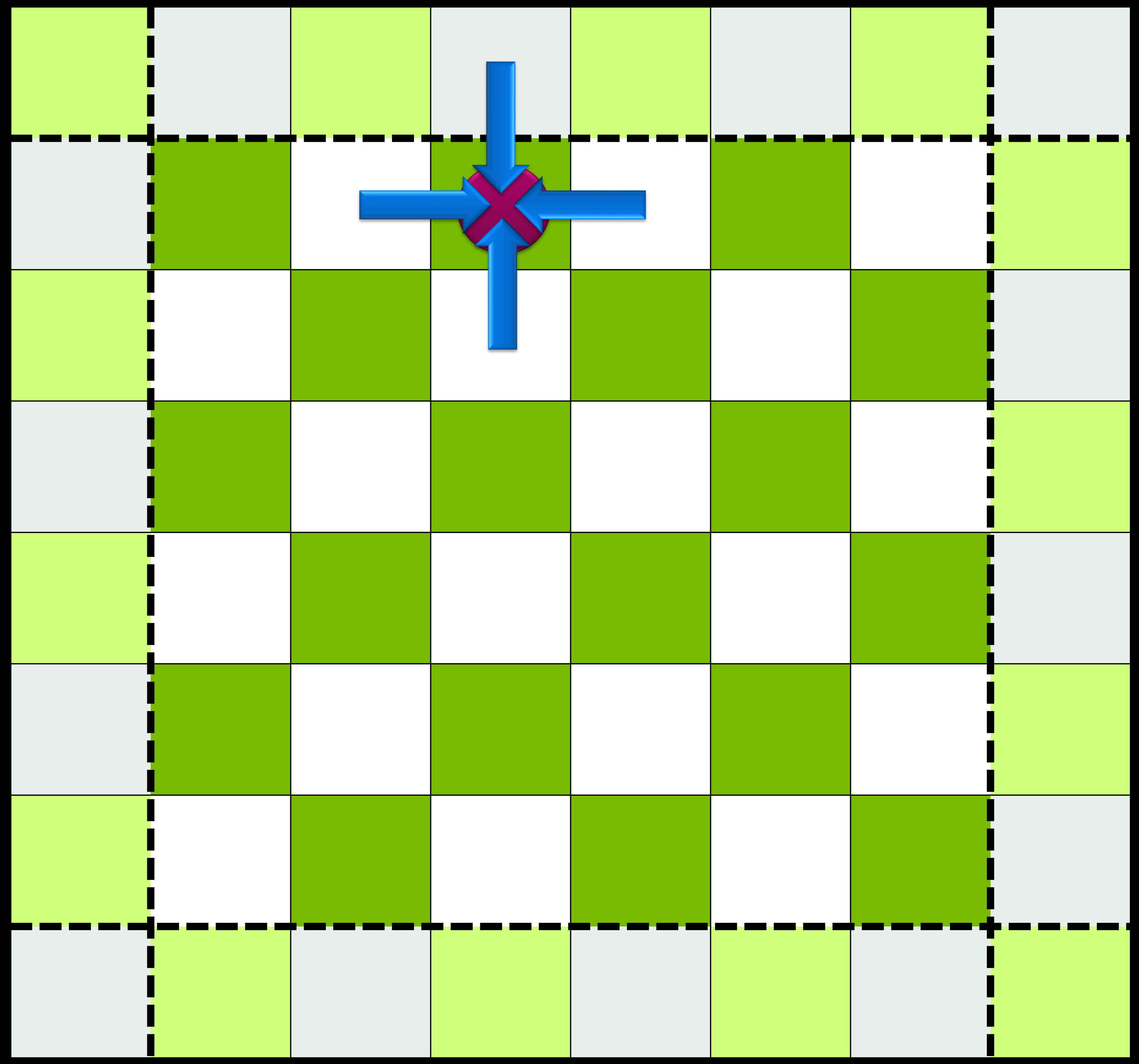
NVIDIA

HISQ Domain-Decomposed Preconditioning

Additive Schwarz Preconditioning with Non-Overlapping Blocks

Speeding up HISQ inversions

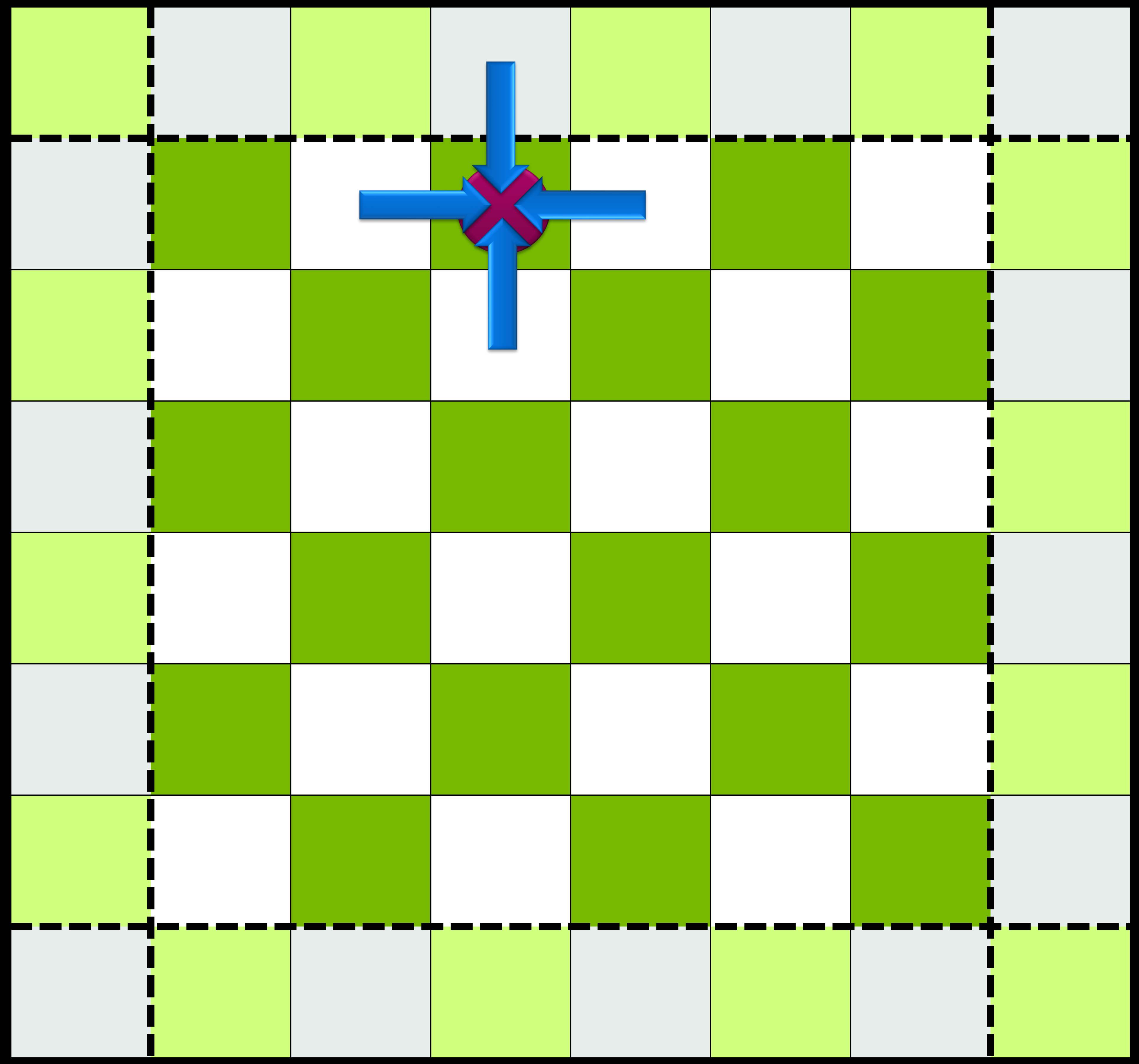
- Simple idea: expand the idea of site-local preconditioning…
  - Preconditioning (twisted-)clover with the (twisted-)clover inverse
  - Example B: 4-d preconditioning of Mobius fermions



Additive Schwarz Preconditioning with Non-Overlapping Blocks

Speeding up HISQ inversions

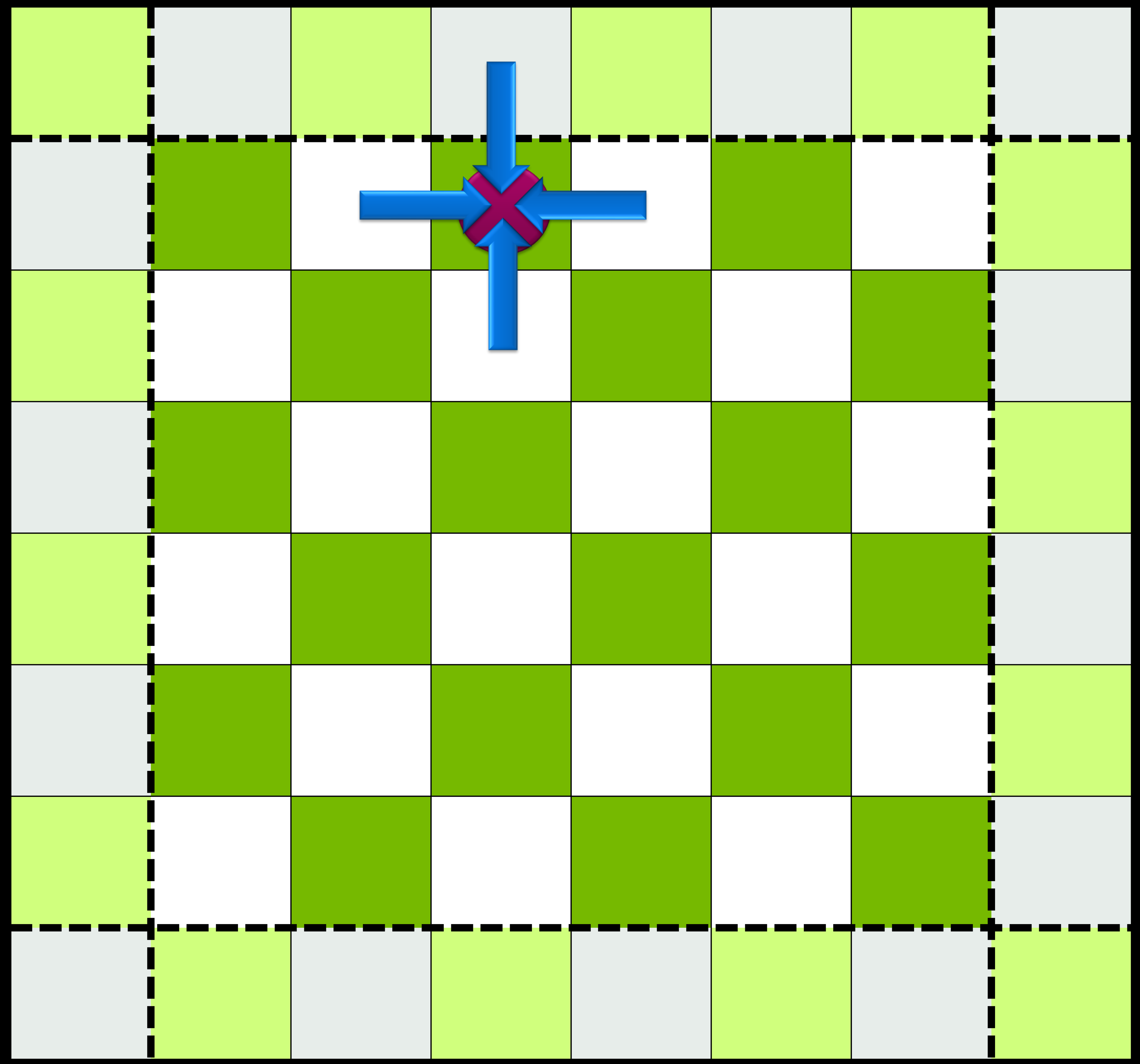
- Simple idea: expand the idea of site-local preconditioning…
  - Preconditioning (twisted-)clover with the (twisted-)clover inverse
  - Example B: 4-d preconditioning of Mobius fermions
- …to larger domains: Schwarz preconditioning



Additive Schwarz Preconditioning with Non-Overlapping Blocks

Speeding up HISQ inversions

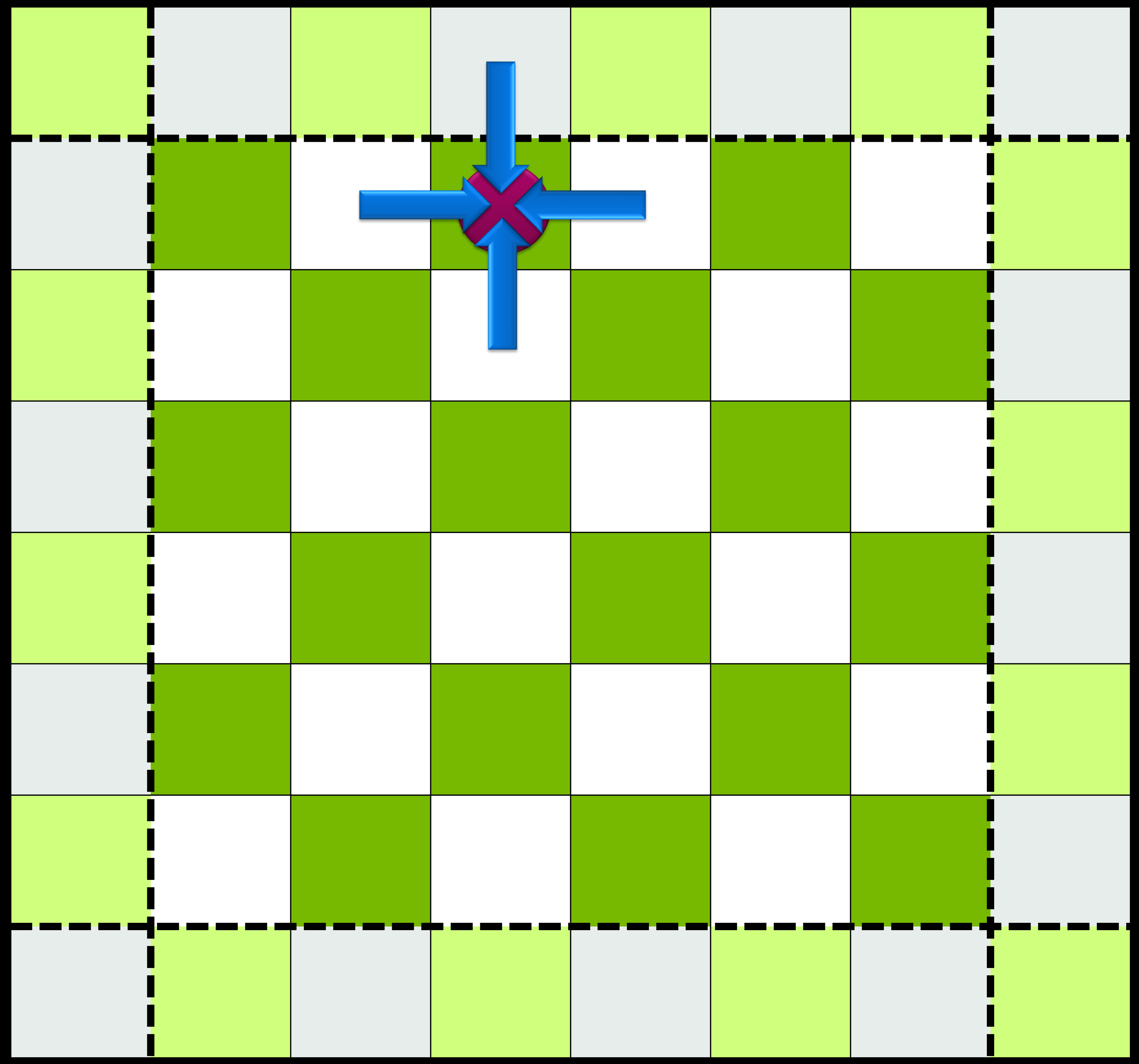
- Simple idea: expand the idea of site-local preconditioning…
  - Preconditioning (twisted-)clover with the (twisted-)clover inverse
  - Example B: 4-d preconditioning of Mobius fermions
- …to larger domains: Schwarz preconditioning
- Additive Schwarz is analogous to Jacobi Iterations, but for domains



Additive Schwarz Preconditioning with Non-Overlapping Blocks

Speeding up HISQ inversions

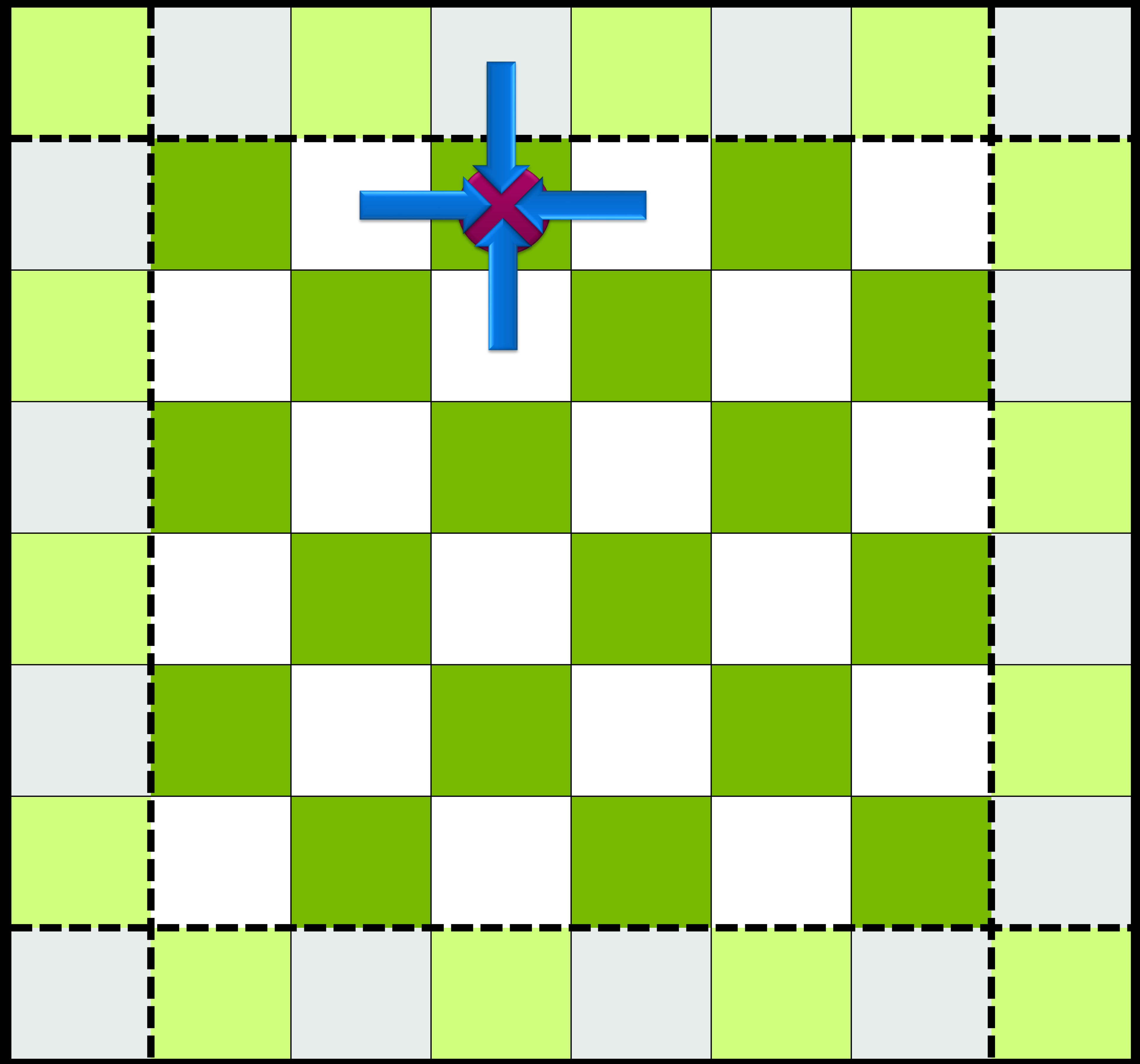
- Simple idea: expand the idea of site-local preconditioning…
  - Preconditioning (twisted-)clover with the (twisted-)clover inverse
  - Example B: 4-d preconditioning of Mobius fermions
- …to larger domains: Schwarz preconditioning
- Additive Schwarz is analogous to Jacobi Iterations, but for domains
- For this talk: domains are non-overlapping



Additive Schwarz Preconditioning with Non-Overlapping Blocks

Speeding up HISQ inversions

- Simple idea: expand the idea of site-local preconditioning…
  - Preconditioning (twisted-)clover with the (twisted-)clover inverse
  - Example B: 4-d preconditioning of Mobius fermions
- …to larger domains: Schwarz preconditioning
- Additive Schwarz is analogous to Jacobi Iterations, but for domains
- For this talk: domains are non-overlapping
- Here: one domain per MPI rank (== one GPU)
  - This is a person-hour coding and debugging constraint
  - There's no inherent algorithmic or machine constraint



Existing Work

Mobius Fermions

- The theory and use of Schwarz preconditioners is long-lived and exhaustive---the idea isn’t anything new-fangled
- The challenge is constructing the algorithm and the implementation

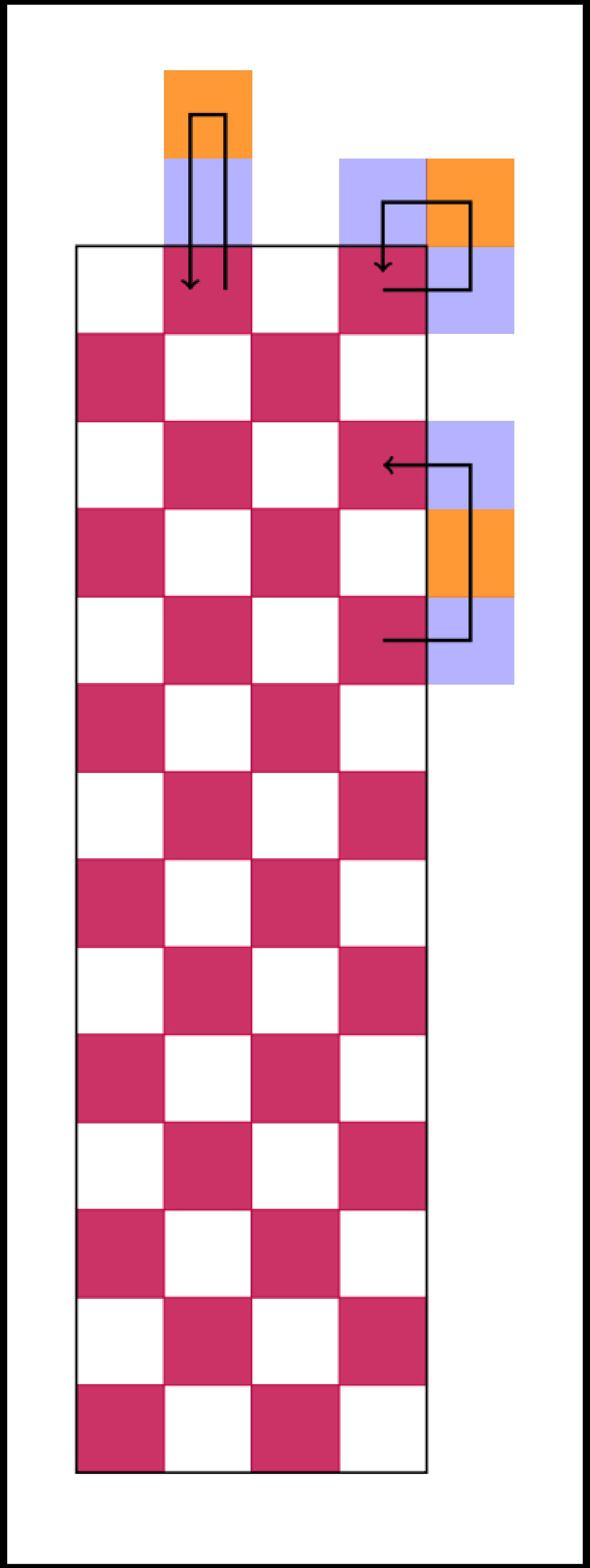
Existing Work

Mobius Fermions

- The theory and use of Schwarz preconditioners is long-lived and exhaustive---the idea isn't anything new-fangled
- The challenge is constructing the algorithm and the implementation
- A recent example in LQCD is Multi-Splitting Preconditioned Conjugate Gradient (MSPCG)
  - [arxiv:2104.05615]
- For Mobius fermions, the relevant HPC operator is the *normal* 4-d preconditioned operator

$$(1 - D_{eo}D_{oe})^{\dagger}(1 - D_{eo}D_{oe})$$

- The product of four Ds generates so-called **snake terms**



Zero Boundaries

"Boundary clovers"

- Let's consider the massless staggered operator… in one dimension, for extreme simplicity

$$D_{x,y}^{stag} \approx \left[M_\mu(x)\delta_{x,y-1} - M_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right]$$

- The stencil gathers from two sites: one on the left, and one on the right

Zero Boundaries

"Boundary clovers"

- Let's consider the massless staggered operator… in one dimension, for extreme simplicity

$$D^{stag}_{x,y} \approx \left[M_\mu(x)\delta_{x,y-1} - M^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right]$$

- The stencil gathers from two sites: one on the left, and one on the right

Exterior domain

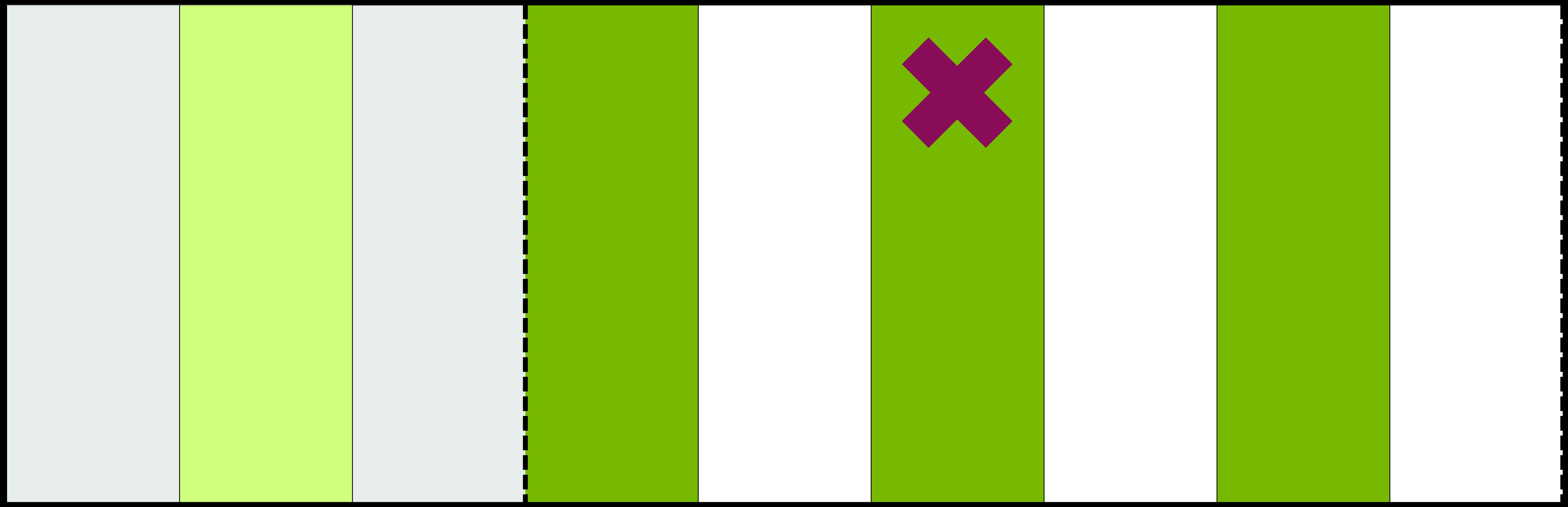
Interior domain

- For non-overlapping blocks, there's no contribution from outside the domain
- Above: contribution from the left is zero
- For this simple stencil, this is equivalent to zeroing out the hopping term itself…
  - **…that thinking is trouble**

Squared operator

- Let’s consider the massless operator **squared**… in one dimension, to keep bookkeeping easy

$$D^{stag}_{x,y} \approx \left[M_\mu(x)\delta_{x,y-1} - M^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right]$$

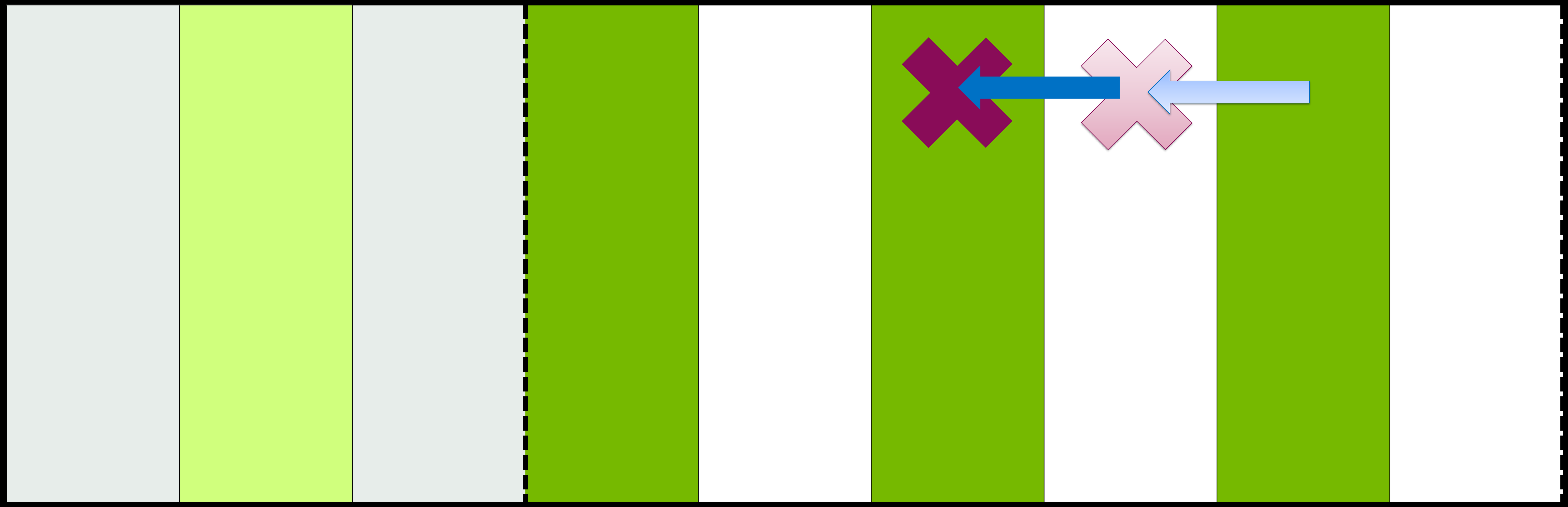


Squared operator

- Let's consider the massless operator **squared**… in one dimension, to keep bookkeeping easy

$$D_{x,y}^{stag} \approx \left[M_\mu(x)\delta_{x,y-1} - M_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right]$$

$$\approx \underbrace{M_\mu(x)M_\mu(x+\hat{\mu})\delta_{x,y-2}}_{From\ the\ right}$$

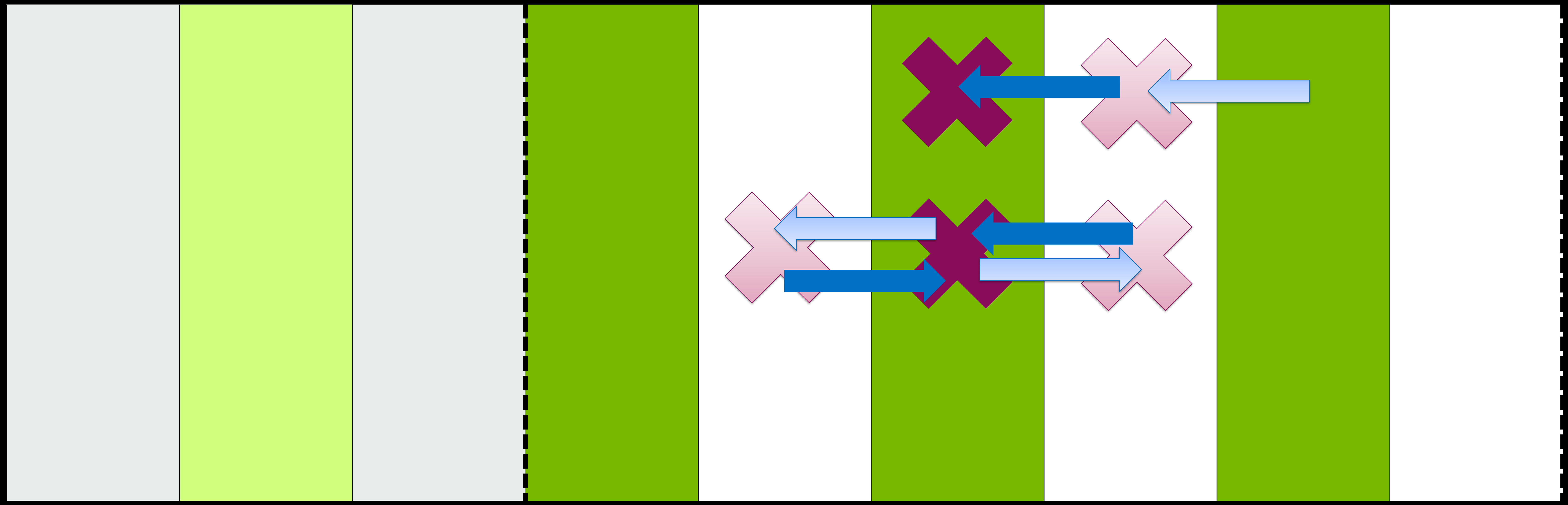


Squared operator

- Let's consider the massless operator **squared**... in one dimension, to keep bookkeeping easy

$$D^{stag}_{x,y} \approx \left[M_\mu(x)\delta_{x,y-1} - M^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right]$$

$$\approx \underbrace{M_\mu(x)M_\mu(x+\hat{\mu})\delta_{x,y-2}}_{From\ the\ right} - \underbrace{\left[M_\mu(x)M^\dagger_\mu(x) + M_\mu(x-\hat{\mu})M^\dagger_\mu(x-\hat{\mu})\right]\delta_{y,z}}_{From\ self}$$

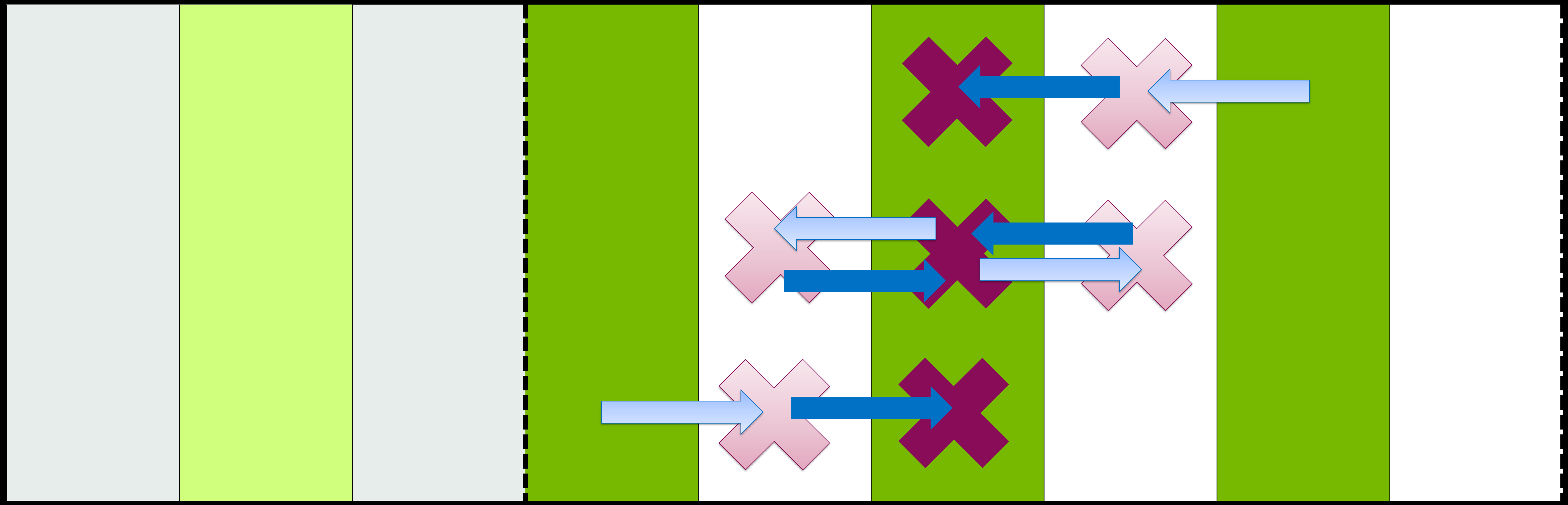


Squared operator

- Let's consider the massless operator **squared**… in one dimension, to keep bookkeeping easy

$$D^{stag}_{x,y} \approx \left[M_\mu(x)\delta_{x,y-1} - M^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right]$$

$$\approx \underbrace{M_\mu(x)M_\mu(x+\hat{\mu})\delta_{x,y-2}}_{From\ the\ right} - \underbrace{\left[M_\mu(x)M^\dagger_\mu(x) + M_\mu(x-\hat{\mu})M^\dagger_\mu(x-\hat{\mu})\right]\delta_{y,z}}_{From\ self} + \underbrace{M^\dagger_\mu(x-\hat{\mu})\ M^\dagger_\mu(x-2\hat{\mu})\delta_{x,y+2}}_{From\ the\ left}$$



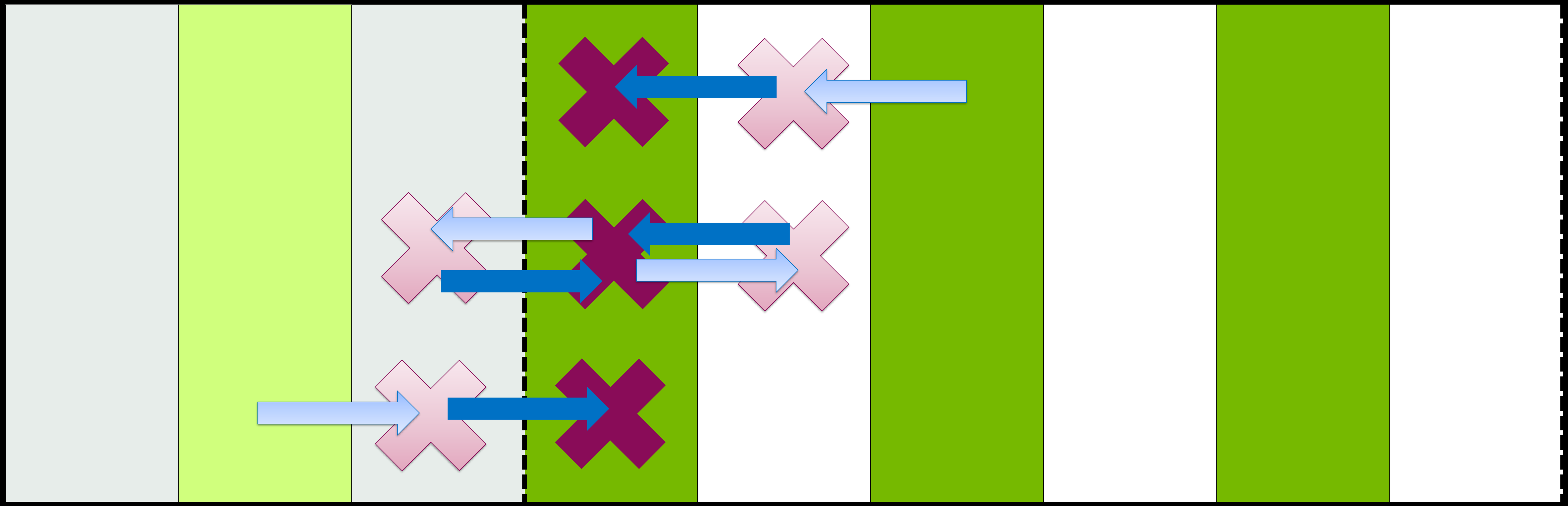
Squared operator on the Boundary

There's always a catch

- Let's consider the massless operator **squared**… in one dimension, to keep bookkeeping easy

$$D^{stag}_{x,y} \approx \left[M_\mu(x)\delta_{x,y-1} - M^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right]$$

$$\approx \underbrace{M_\mu(x)M_\mu(x+\hat{\mu})\delta_{x,y-2}}_{\text{From the right}} - \underbrace{\left[M_\mu(x)M^\dagger_\mu(x) + M_\mu(x-\hat{\mu})M^\dagger_\mu(x-\hat{\mu})\right]\delta_{y,z}}_{\text{From self}} + \underbrace{M^\dagger_\mu(x-\hat{\mu})\, M^\dagger_\mu(x-2\hat{\mu})\delta_{x,y+2}}_{\text{From the left}}$$

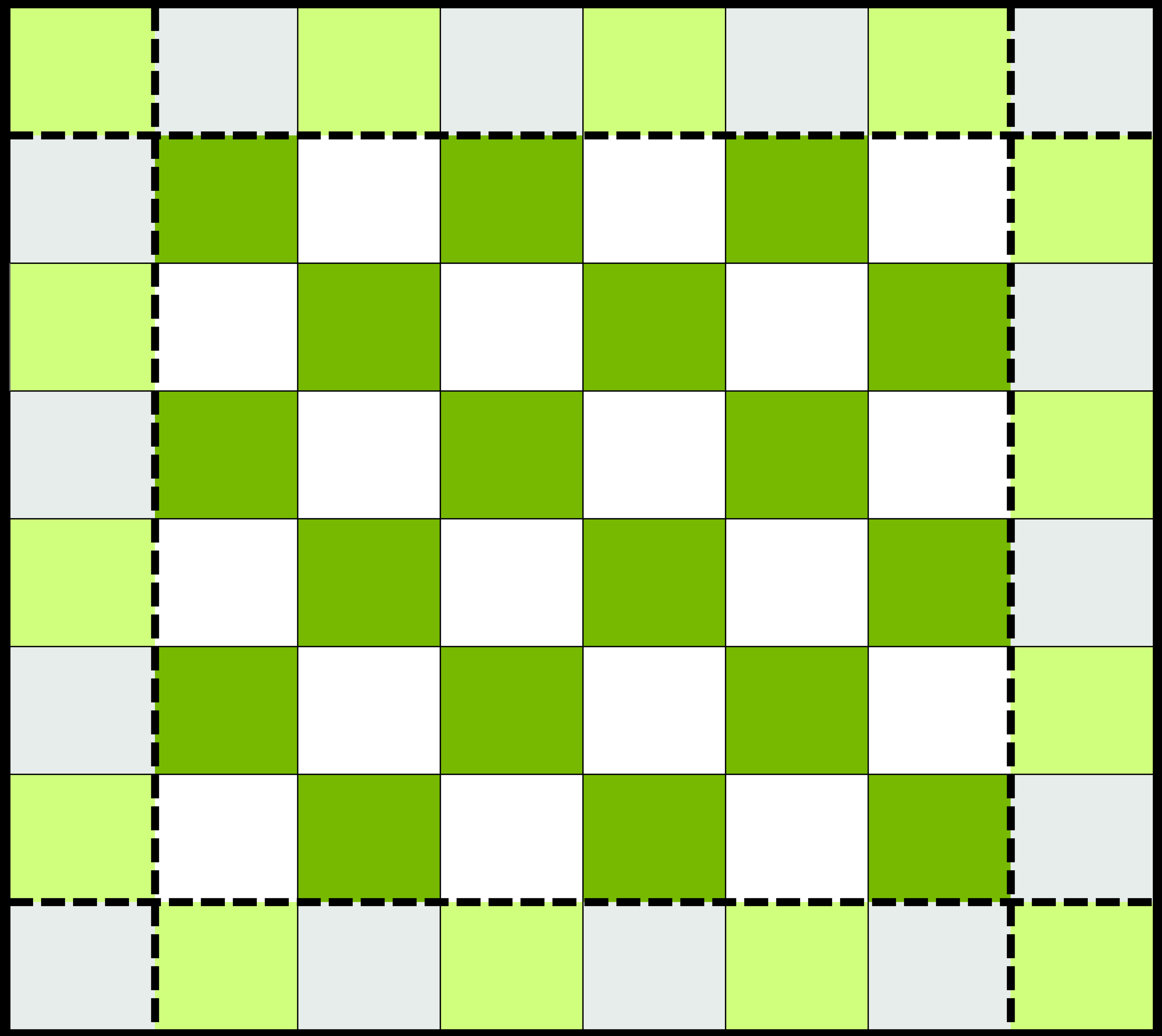


Sidebar: MSPCG Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D^\dagger_{oe} D^\dagger_{eo} D_{eo} D_{oe}$$



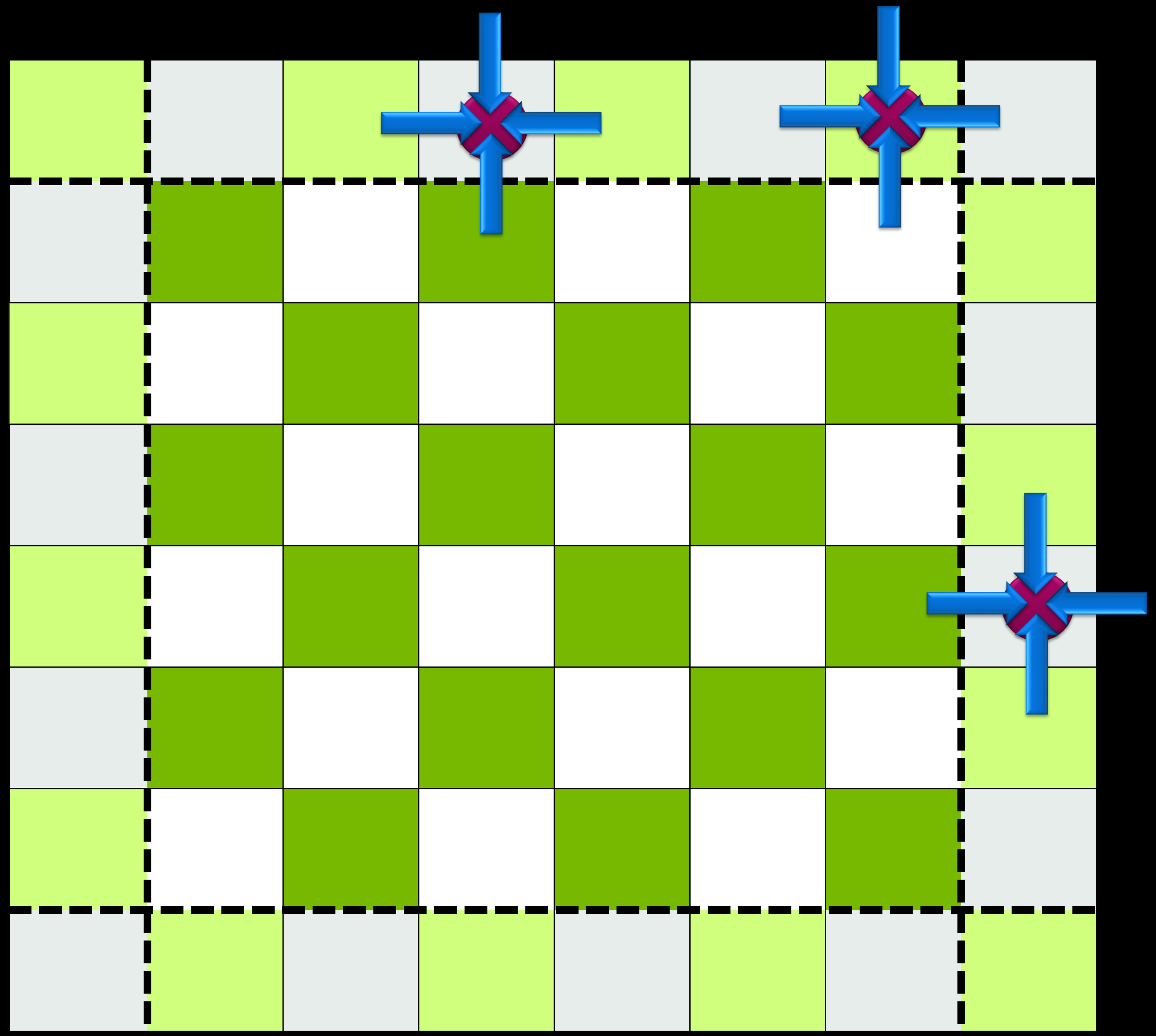
Existing Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D_{oe}^{\dagger} D_{eo}^{\dagger} D_{eo} D_{oe}$$

- Four steps, one for each operator application
  1. D_{oe} on $(L+2)^4$ volume



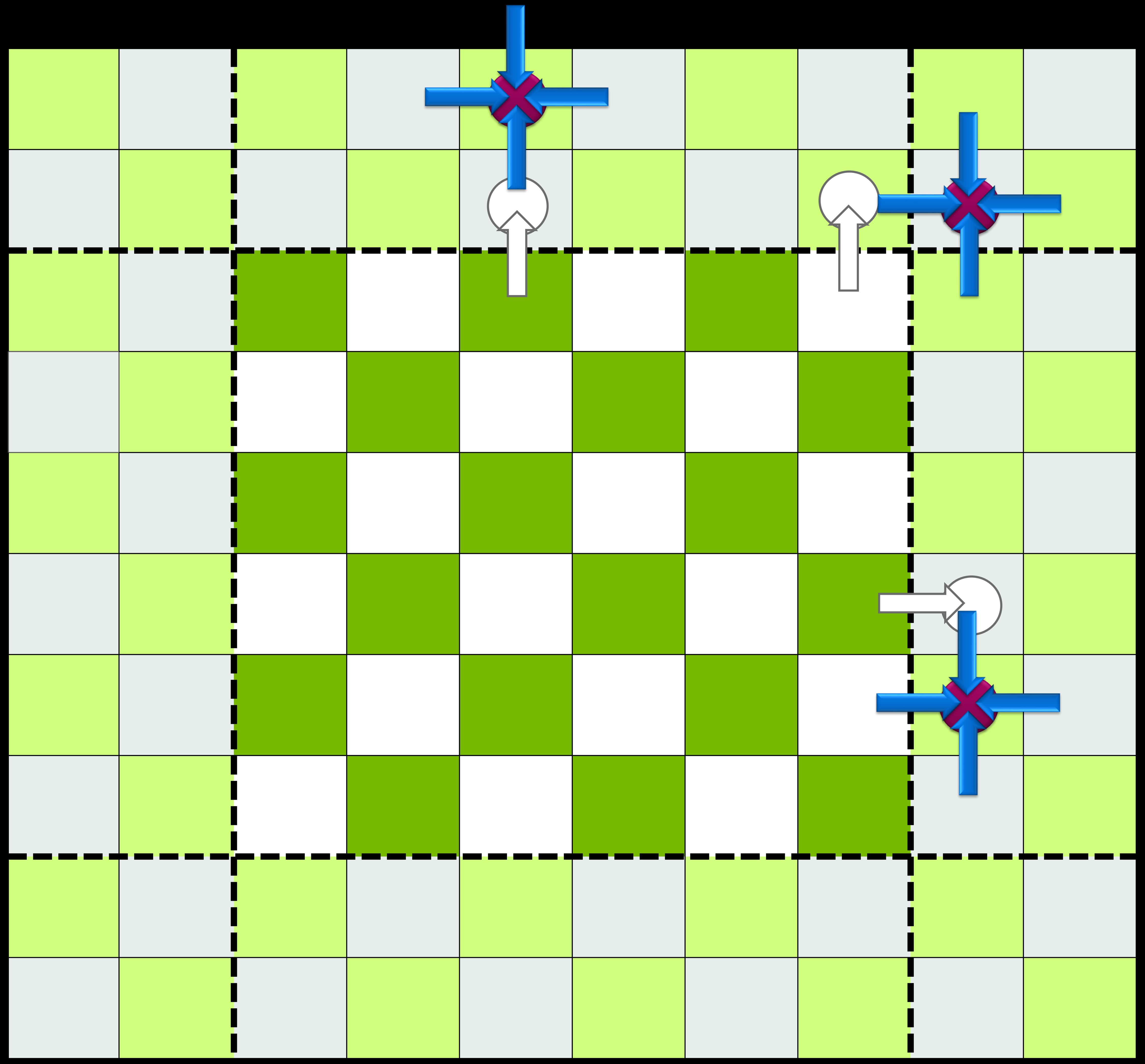
Existing Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D_{oe}^{\dagger} D_{eo}^{\dagger} D_{eo} D_{oe}$$

- Four steps, one for each operator application
  1. D_{oe} on $(L+2)^4$ volume
  2. D_{eo} on $(L+4)^4$ volume



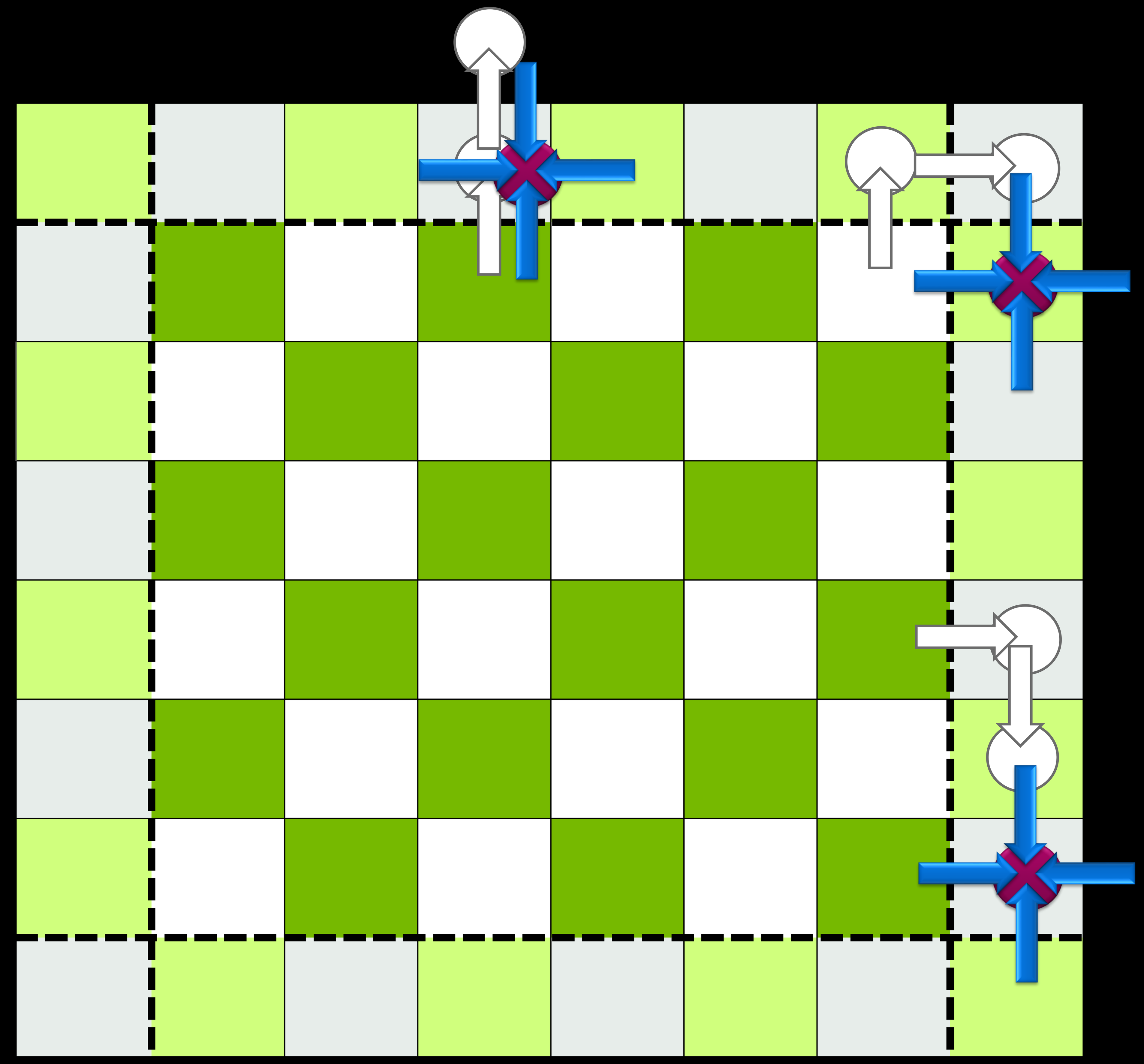
Existing Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D_{oe}^{\dagger} D_{eo}^{\dagger} D_{eo} D_{oe}$$

- Four steps, one for each operator application
  1. D_{oe} on $(L+2)^4$ volume
  2. D_{eo} on $(L+4)^4$ volume
  3. $D_{eo}^{\dagger}$ on $(L+2)^4$ volume



NVIDIA

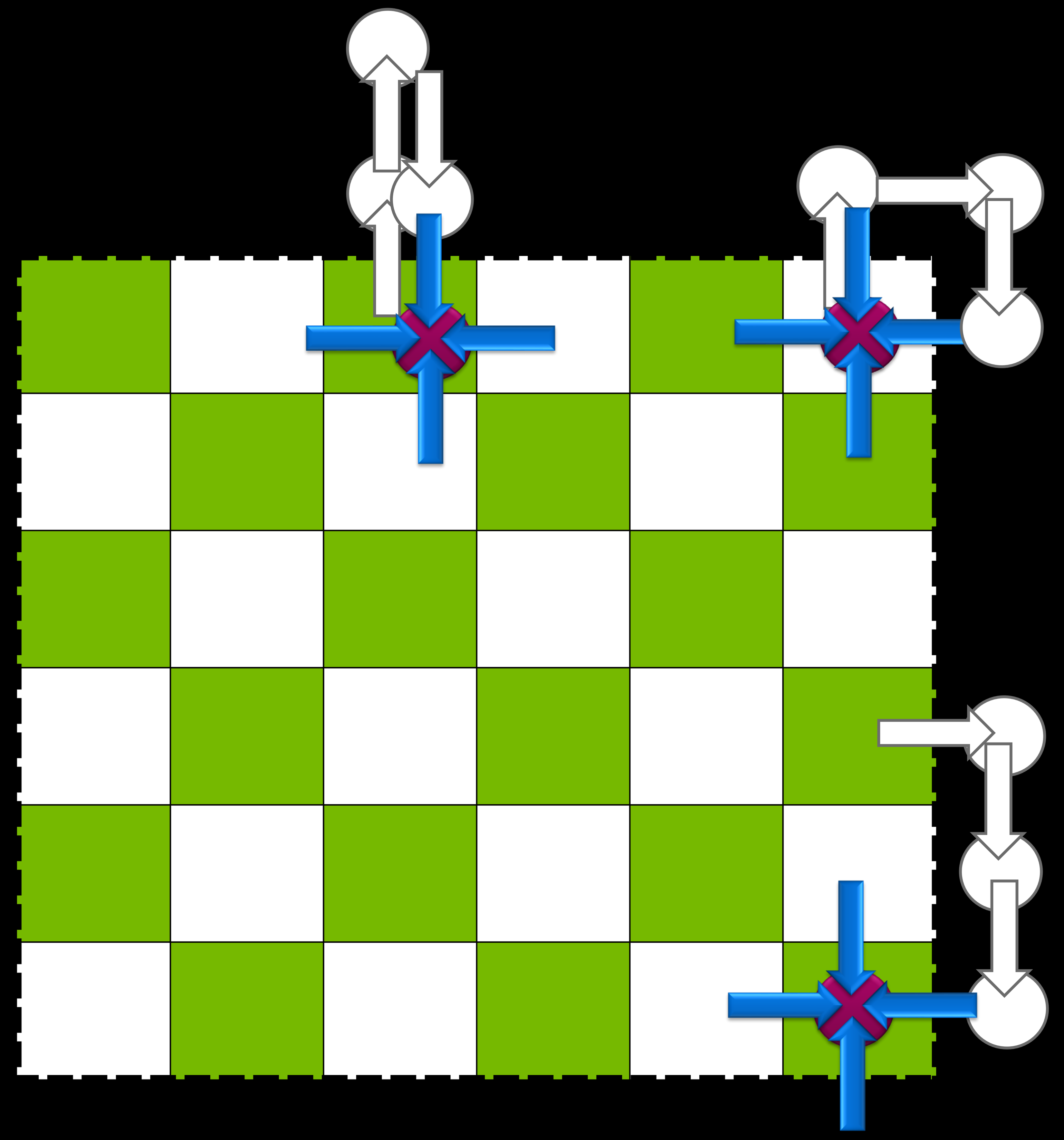
Existing Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D_{oe}^{\dagger} D_{eo}^{\dagger} D_{eo} D_{oe}$$

- Four steps, one for each operator application
  1. D_{oe} on $(L+2)^4$ volume
  2. D_{eo} on $(L+4)^4$ volume
  3. $D_{eo}^{\dagger}$ on $(L+2)^4$ volume
  4. $D_{oe}^{\dagger}$ on on L^4 volume



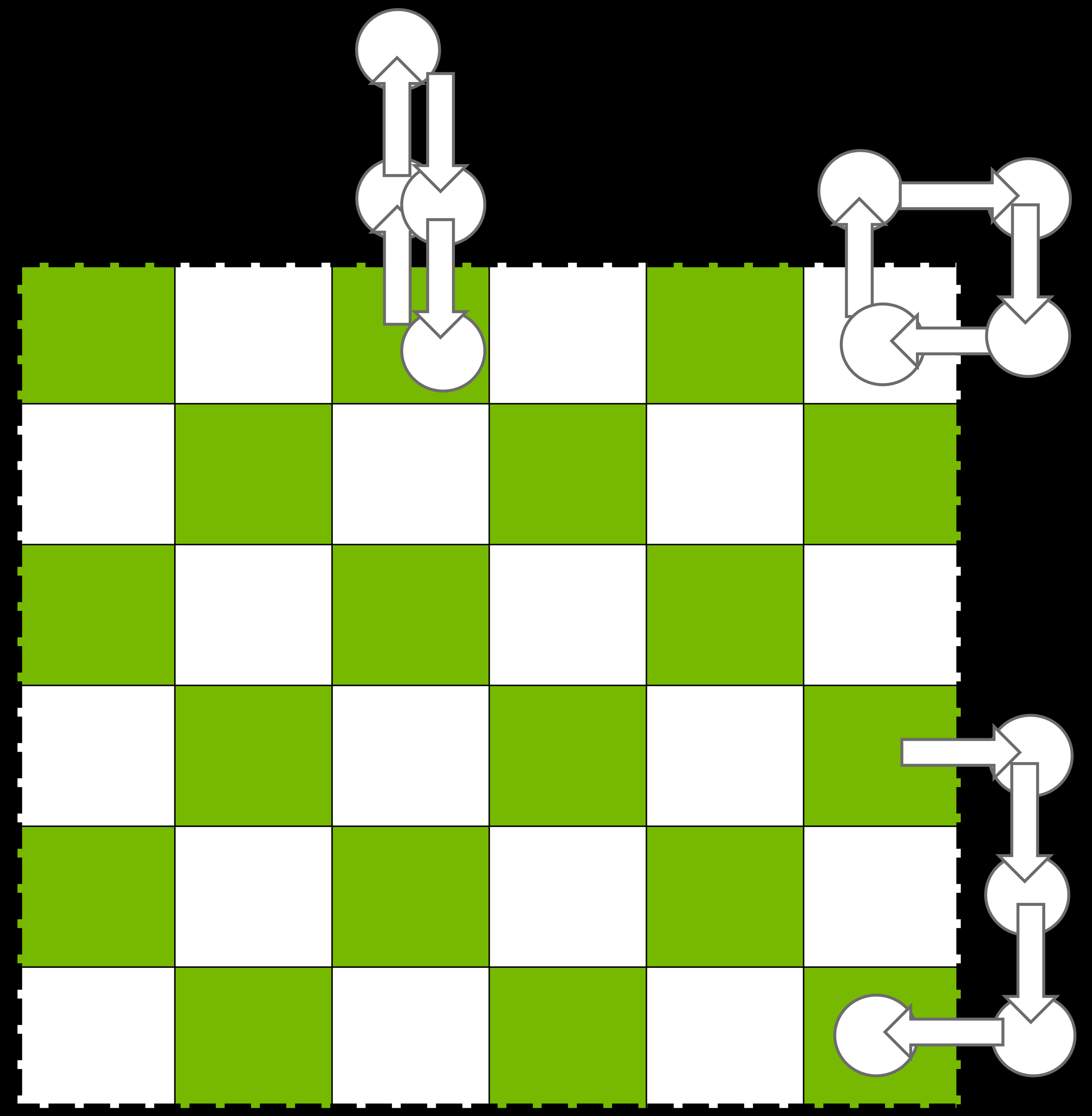
Existing Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D_{oe}^{\dagger} D_{eo}^{\dagger} D_{eo} D_{oe}$$

- Four steps, one for each operator application
  1. D_{oe} on $(L+2)^4$ volume
  2. D_{eo} on $(L+4)^4$ volume
  3. $D_{eo}^{\dagger}$ on $(L+2)^4$ volume
  4. $D_{oe}^{\dagger}$ on on L^4 volume
- This extra work can be very expensive; non-trivially so for small local domains (strong-scaling regime)



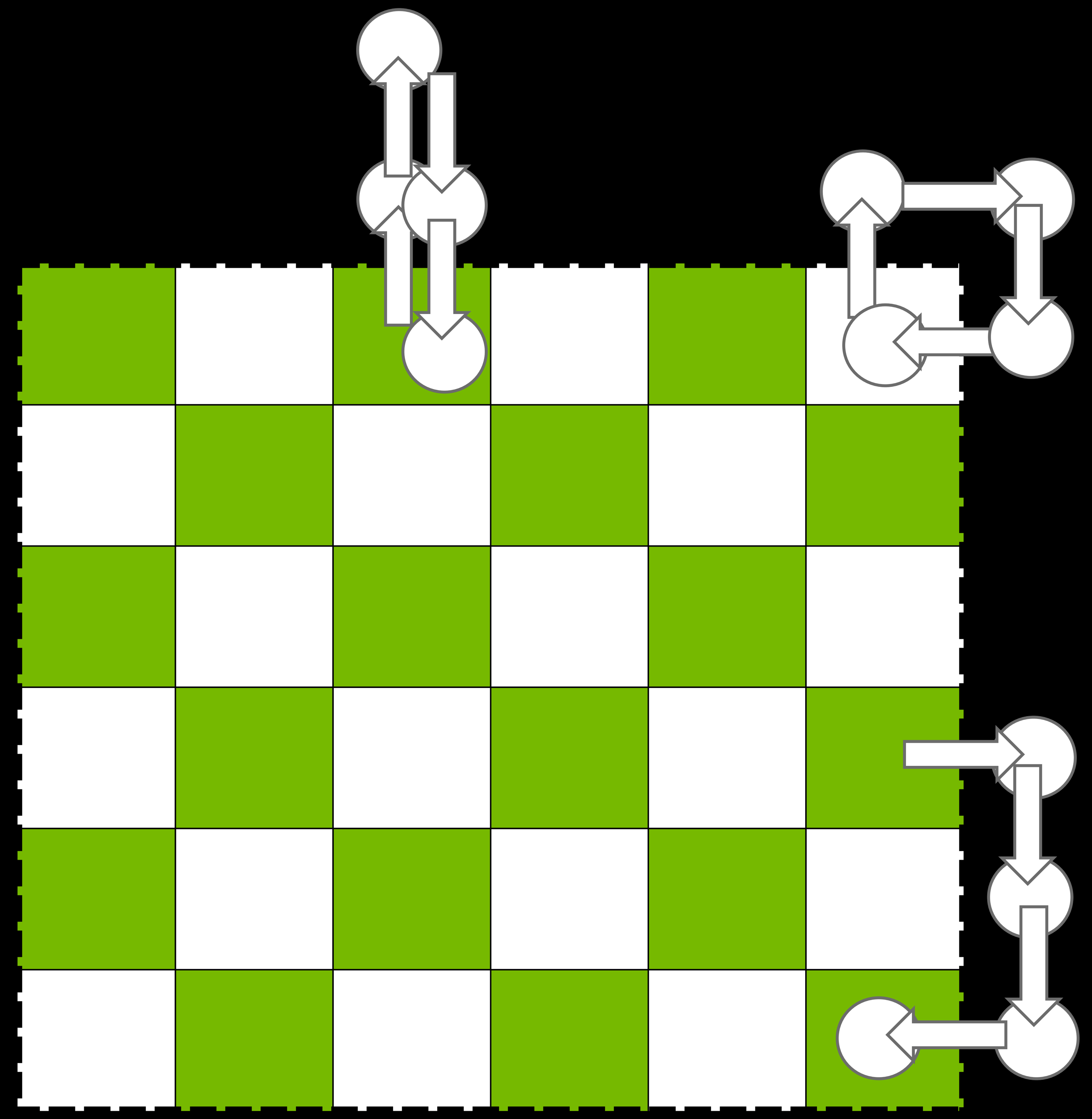
Existing Work

Mobius Fermions

- The MSPCG work took advantage of extended domains

$$D_{oe}^{\dagger} D_{eo}^{\dagger} D_{eo} D_{oe}$$

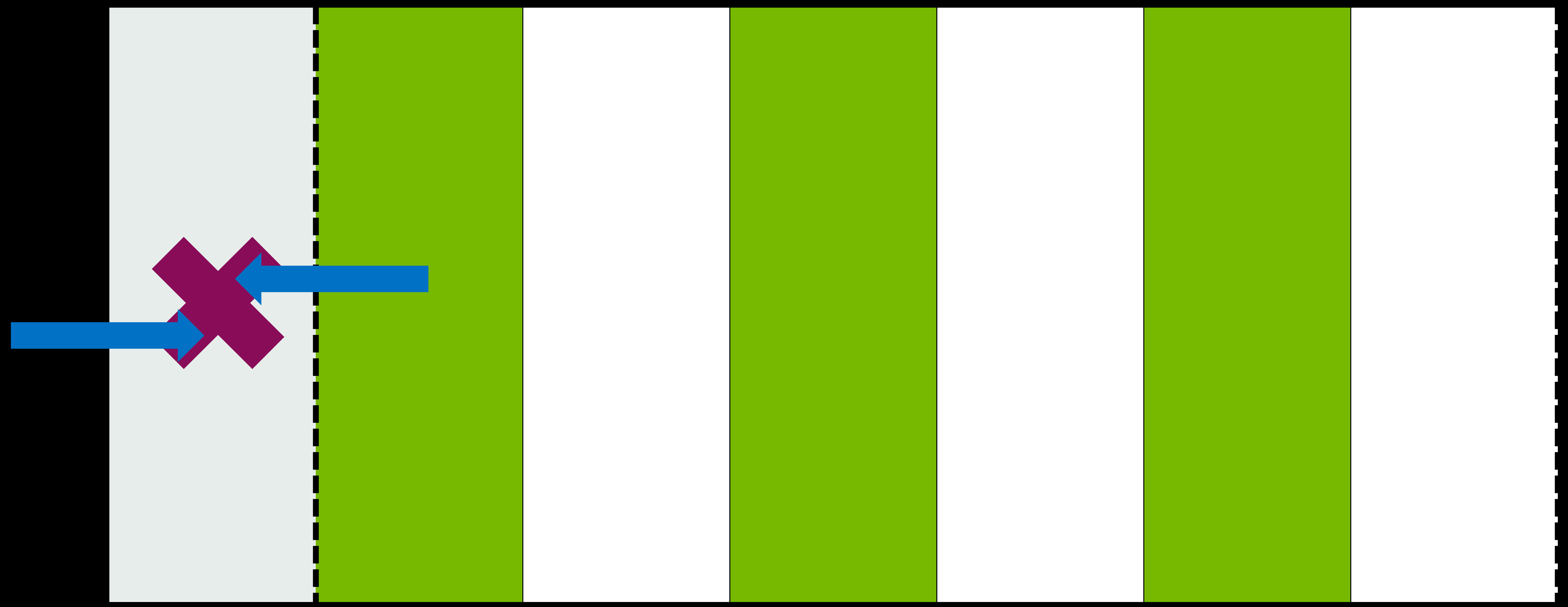
- Four steps, one for each operator application
  1. D_{oe} on $(L+2)^4$ volume
  2. D_{eo} on $(L+4)^4$ volume
  3. $D_{eo}^{\dagger}$ on $(L+2)^4$ volume
  4. $D_{oe}^{\dagger}$ on on L^4 volume
- This extra work can be very expensive; non-trivially so for small local domains (strong-scaling regime)
- HISQ fermions have relative benefits and challenges
  - Only $D_{eo} D_{oe}$
  - Need to bookkeep distance-1 and distance-3 terms
  - Distance-3 terms would necessitate an $(L+6)^4$ volume



Application to 1-d Staggered

Extended domains

$$D_{x,y}^{stag} \approx \left[M_\mu(x)\delta_{x,y-1} - M_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right]$$

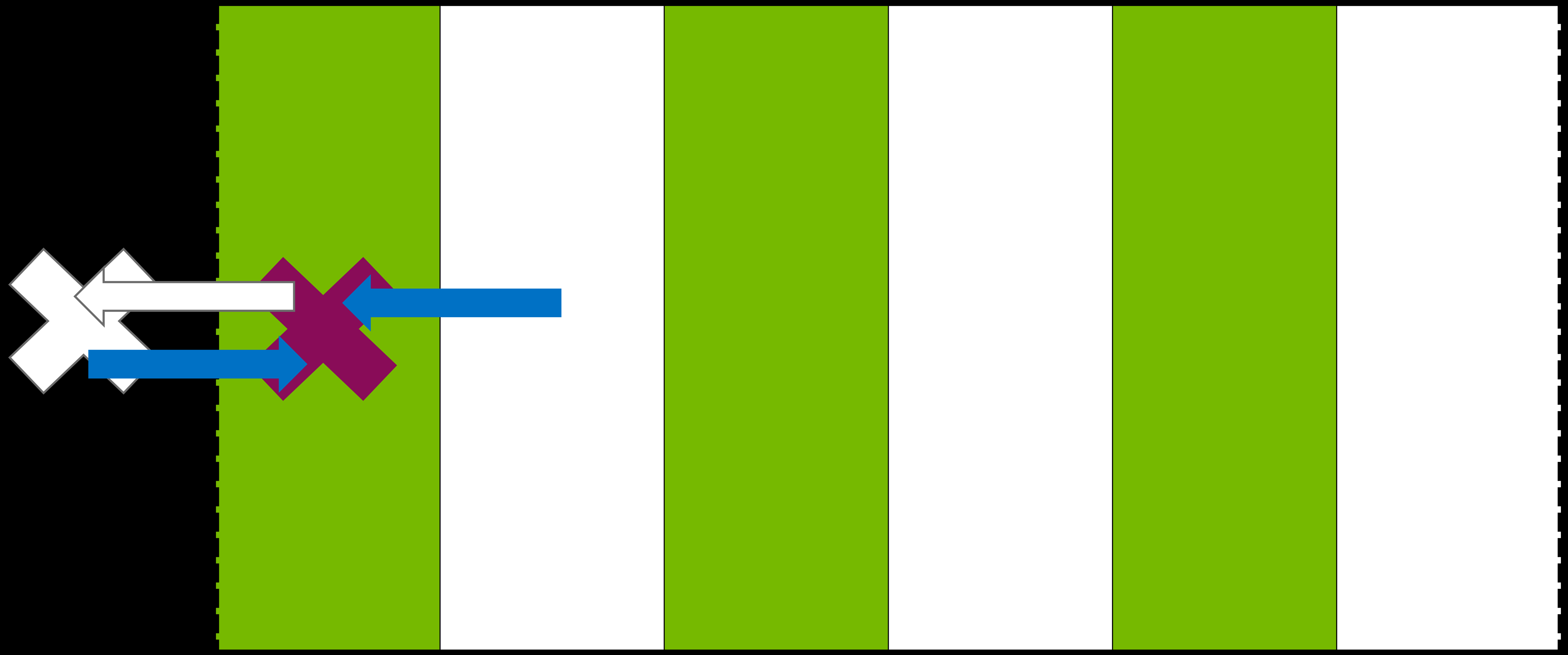


- Step one: calculate including the extended domain

Application to 1-d Staggered

Extended domains

$$D_{x,y}^{stag} \approx \left[M_\mu(x)\delta_{x,y-1} - M_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right]$$

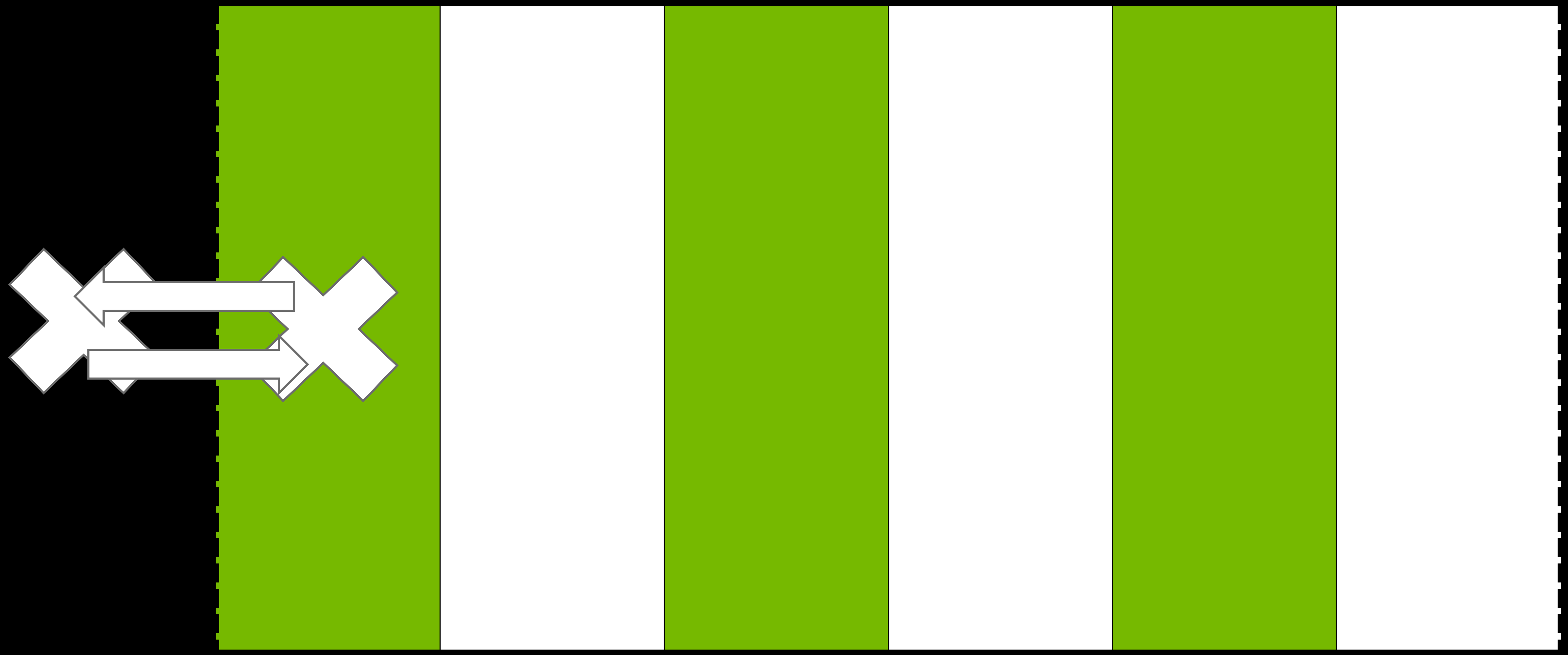


- Step two: only calculate within the interior

Application to 1-d Staggered

$$D^{stag}_{x,y} \approx \left[M_\mu(x)\delta_{x,y-1} - M^\dagger_\mu(x-\hat{\mu})\delta_{x,y+1}\right]$$

$$\approx \underbrace{M_\mu(x)M_\mu(x+\hat{\mu})\delta_{x,y-2}}_{From\ the\ right} - \underbrace{\left[M_\mu(x)M^\dagger_\mu(x) + M_\mu(x-\hat{\mu})M^\dagger_\mu(x-\hat{\mu})\right]\delta_{y,z}}_{From\ self} + \underbrace{M^\dagger_\mu(x-\hat{\mu})\ M^\dagger_\mu(x-2\hat{\mu})\delta_{x,y+2}}_{From\ the\ left}$$

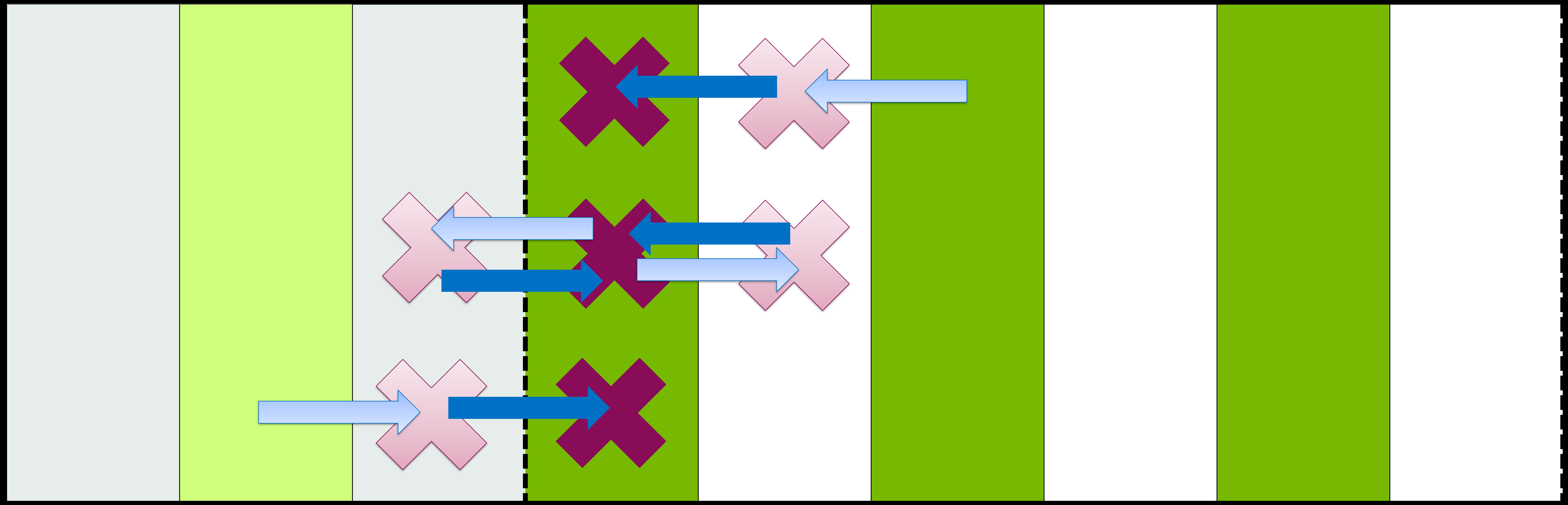


- This also gives you the boundary term

Alternative Form: “Boundary Clover”

$$\approx \underbrace{M_\mu(x)M_\mu(x+\hat{\mu})\delta_{x,y-2}}_{From\ the\ right} - \underbrace{\left[M_\mu(x)M_\mu^\dagger(x) + M_\mu(x-\hat{\mu})M_\mu^\dagger(x-\hat{\mu})\right]\delta_{y,z}}_{From\ self} + \underbrace{M_\mu^\dagger(x-\hat{\mu})\, M_\mu^\dagger(x-2\hat{\mu})\delta_{x,y+2}}_{From\ the\ left}$$

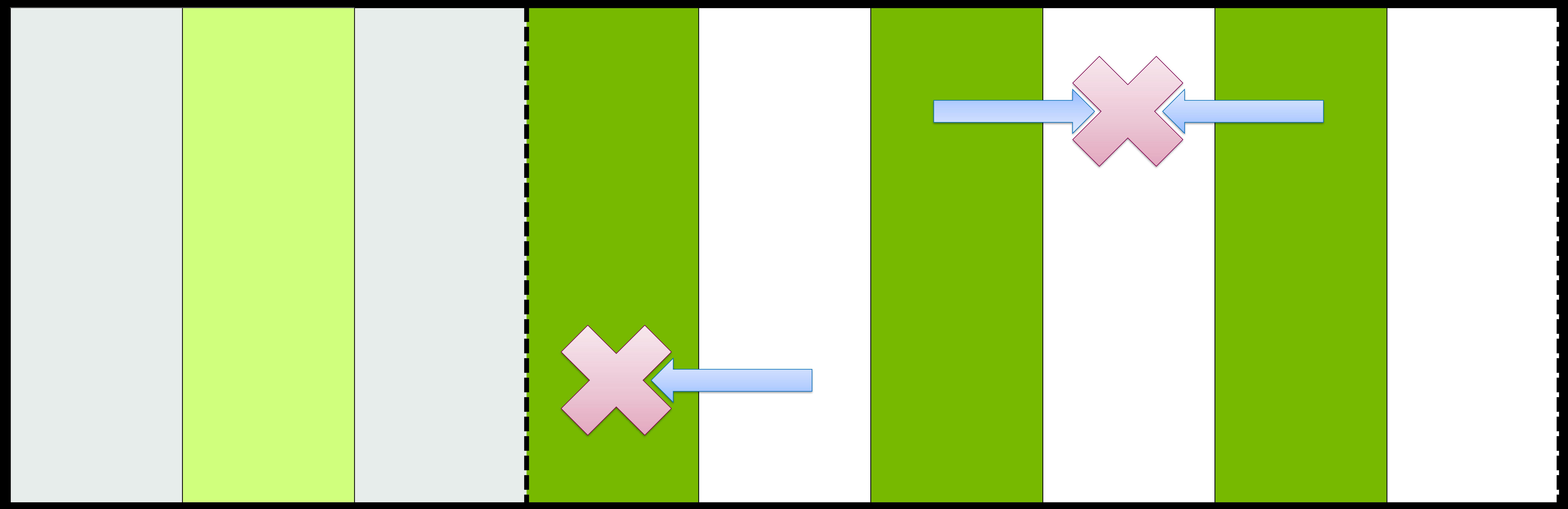
- Alternative approach: what if we “just” calculated the self-contribution (“boundary clover”) directly?



Implementing a Boundary Clover Workflow

Step 1

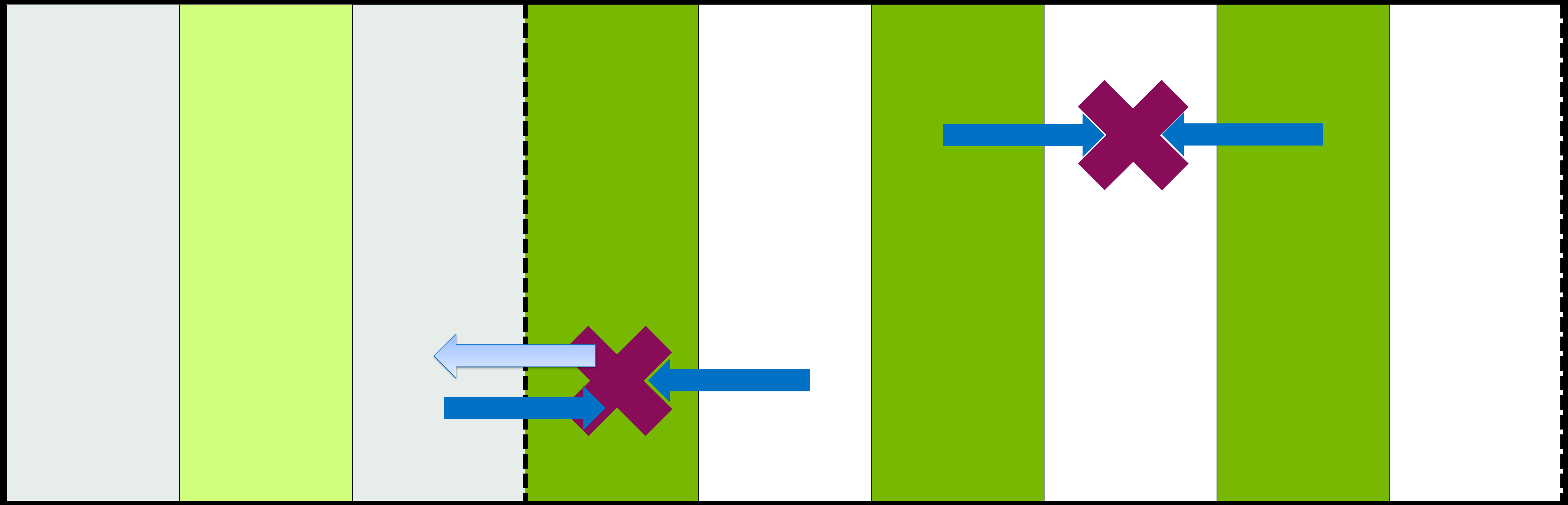
- An implementation in two parts:
- Step 1: Apply the operator *with Dirichlet boundary conditions*
  - For operators in the interior, this is nothing interesting
  - For operators on the boundary, it's a quick snip



Boundary Clover

Step 2

- An implementation in two parts:
- Step 2: Apply the operator *with “clover” computations on the boundary*
  - For operators on the interior, this is nothing special
  - For operators on the boundary, in the direction of the boundary, compute the full hop “out and in”
- Key optimizations:
  - We can reuse the same link for the “out” as the “in”
  - We could create a custom field with this pre-computed to avoid the multiplication



Application to HISQ

Review: HISQ Stencil

Three hops this time

- On face value, the HISQ stencil has no complications relative to the naïve staggered example

$$D^{HISQ}_{x,y} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[\left(F_\mu(x)\delta_{x,y-1} - F_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right) + \left(L_\mu(x)\delta_{x,y-3} - L_\mu^\dagger(x-3\hat{\mu})\delta_{x,y+3}\right)\right] + 2m\delta_{x,y}$$

- Here, F is the distance 1 “fat link” and L is the distance 3 “long link”

Review: HISQ Stencil

Three hops this time

- On face value, the HISQ stencil has no complications relative to the naïve staggered example

$$D_{x,y}^{HISQ} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[\left(F_\mu(x)\delta_{x,y-1} - F_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right) + \left(L_\mu(x)\delta_{x,y-3} - L_\mu^\dagger(x-3\hat{\mu})\delta_{x,y+3}\right)\right] + 2m\delta_{x,y}$$

- Here, F is the distance 1 "fat link" and L is the distance 3 "long link"
- This *does* lead to extra bookkeeping at the boundary
  - Site at [0]: There are neither fat nor long link contributions from the "left": outside the domain

Review: HISQ Stencil

Three hops this time

- On face value, the HISQ stencil has no complications relative to the naïve staggered example

$$D_{x,y}^{HISQ} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[\left(F_\mu(x)\delta_{x,y-1} - F_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right) + \left(L_\mu(x)\delta_{x,y-3} - L_\mu^\dagger(x-3\hat{\mu})\delta_{x,y+3}\right)\right] + 2m\delta_{x,y}$$

- Here, F is the distance 1 "fat link" and L is the distance 3 "long link"
- This *does* lead to extra bookkeeping at the boundary
  - Site at [0]: There are neither fat nor long link contributions from the "left": outside the domain

Review: HISQ Stencil

Three hops this time

- On face value, the HISQ stencil has no complications relative to the naïve staggered example

$$D^{HISQ}_{x,y} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[\left(F_\mu(x)\delta_{x,y-1} - F_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right) + \left(L_\mu(x)\delta_{x,y-3} - L_\mu^\dagger(x-3\hat{\mu})\delta_{x,y+3}\right)\right] + 2m\delta_{x,y}$$

- Here, F is the distance 1 “fat link” and L is the distance 3 “long link”
- This *does* lead to extra bookkeeping at the boundary
  - Site at [0]: There are neither fat nor long link contributions from the “left”: outside the domain
  - Sites at [1] or [2]: There is no long link contribution from the “left”, but there’s still a fat link contribution!

Review: HISQ Stencil

Three hops this time

- On face value, the HISQ stencil has no complications relative to the naïve staggered example

$$D_{x,y}^{HISQ} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[\left(F_\mu(x)\delta_{x,y-1} - F_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right) + \left(L_\mu(x)\delta_{x,y-3} - L_\mu^\dagger(x-3\hat{\mu})\delta_{x,y+3}\right)\right] + 2m\delta_{x,y}$$

- Here, F is the distance 1 "fat link" and L is the distance 3 "long link"
- This *does* lead to extra bookkeeping at the boundary
  - Site at [0]: There are neither fat nor long link contributions from the "left": outside the domain
  - Sites at [1] or [2]: There is no long link contribution from the "left", but there's still a fat link contribution!

Schur Going to Have a Tough Time

Three hops this time

- The "real" goal is the even/odd preconditioned operator:

$$D^{HISQ}_{x,y} \approx \sum_{\mu=0}^{3} \eta_\mu(x)\left[\left(F_\mu(x)\delta_{x,y-1} - F_\mu^\dagger(x-\hat{\mu})\delta_{x,y+1}\right) + \left(L_\mu(x)\delta_{x,y-3} - L_\mu^\dagger(x-3\hat{\mu})\delta_{x,y+3}\right)\right] + 2m\delta_{x,y}$$

$$\begin{bmatrix} 2m & D_{eo} \\ D_{oe} & 2m \end{bmatrix} \begin{bmatrix} x_e \\ x_o \end{bmatrix} = \begin{bmatrix} b_e \\ b_o \end{bmatrix}$$

$$(4m^2 - D_{eo}D_{oe})x_e = 2mb_e - D_{eo}b_o$$

- The type of bookkeeping noted in the previous slide causes new headaches

Site Zero

Three hops this time

- Let's first consider the site at [0]
- There are three "boundary" contributions:
  - Start at [0]: fat link left, fat link right
  - Start at [0]: long link left, long link right
  - Start at [2]: long link left, fat link right

Site One

Three hops this time

- Let’s first consider the site at [1]
- There is only one boundary condition:
  - Start at [1]: long link left, long link right

Site Two

Three hops this time

- Last, we'll consider the term at [2]
- There are two boundary contributions:
  - Start at [2]: long link left, long link right
  - Start at [0]!: fat link left, long link right

Solver Workflow

Solving at the speed of sound

- For the non-preconditioned solve, we use mixed-precision conjugate gradient (CG) with gauge link reconstruction

Solver Workflow

Solving at the speed of sound

- For the non-preconditioned solve, we use mixed-precision conjugate gradient (CG) with gauge link reconstruction
- Reconstruction reminder:
  - The fat links are general 3x3 matrices
  - The long links are (proportional to) U(3) matrices, which can be represented as 9 or 13 reals

Solver Workflow

Solving at the speed of sound

- For the non-preconditioned solve, we use mixed-precision conjugate gradient (CG) with gauge link reconstruction
- Reconstruction reminder:
  - The fat links are general 3x3 matrices
  - The long links are (proportional to) U(3) matrices, which can be represented as 9 or 13 reals
- Mixed precision solve:
  - Outer operator: Double precision; reconstruct-13 for long links
  - Sloppy operator: “Half” precision (QUDA’s 16-bit fixed point format); reconstruct-9 for long links

Solver Workflow

Solving at the speed of sound

- For the non-preconditioned solve, we use mixed-precision conjugate gradient (CG) with gauge link reconstruction
- Reconstruction reminder:
  - The fat links are general 3x3 matrices
  - The long links are (proportional to) U(3) matrices, which can be represented as 9 or 13 reals
- Mixed precision solve:
  - Outer operator: Double precision; reconstruct-13 for long links
  - Sloppy operator: "Half" precision (QUDA's 16-bit fixed point format); reconstruct-9 for long links
- For the preconditioned solver:
  - We use preconditioned CG (PCG) as the outer solve
  - We use fixed-iteration CG as the inner solve

Solver Workflow

Solving at the speed of sound

- For the non-preconditioned solve, we use mixed-precision conjugate gradient (CG) with gauge link reconstruction
- Reconstruction reminder:
  - The fat links are general 3x3 matrices
  - The long links are (proportional to) U(3) matrices, which can be represented as 9 or 13 reals
- Mixed precision solve:
  - Outer operator: Double precision; reconstruct-13 for long links
  - Sloppy operator: “Half” precision (QUDA’s 16-bit fixed point format); reconstruct-9 for long links
- For the preconditioned solver:
  - We use preconditioned CG (PCG) as the outer solve
  - We use fixed-iteration CG as the inner solve
- Note: PCG on paper requires a stationary preconditioner…
  - But with a Polak–Ribière correction, CG is “no worse than” Gradient Descent…
  - …and seems to work well enough

Reference Configurations, System

Solving at the speed of sound

- Configuration:
  - NERSC Large configuration
  - Volume: 72^3x144
  - Bare light mass $am = 0.001$

Reference Configurations, System

Solving at the speed of sound

- Configuration:
  - NERSC Large configuration
  - Volume: 72^3x144
  - Bare light mass $am = 0.001$
- Machine: Selene
  - DGX-A100-80GB nodes
  - Use 4xGPUs per node
  - 1:1 NIC ratio; HDR 200 (25 GB/s bi-directional)

Reference Configurations, System

Solving at the speed of sound

- Configuration:
  - NERSC Large configuration
  - Volume: 72^3x144
  - Bare light mass $am = 0.001$
- Machine: Selene
  - DGX-A100-80GB nodes
  - Use 4xGPUs per node
  - 1:1 NIC ratio; HDR 200 (25 GB/s bi-directional)
- We consider multiple strong scaling problem sizes

Nodes	GPUs	Local Domain
8	32	36^4
16	64	36^3x18
32	128	36^2x18^2
64	256	36x18^3
128	512	18^4

Reference Configurations, System

Solving at the speed of sound

- Configuration:
  - NERSC Large configuration
  - Volume: 72^3x144
  - Bare light mass $am = 0.001$
- Machine: Selene
  - DGX-A100-80GB nodes
  - Use 4xGPUs per node
  - 1:1 NIC ratio; HDR 200 (25 GB/s bi-directional)
- We consider multiple strong scaling problem sizes
- For networks:
  - 2:1 and 1:1 direct GPU:NIC bindings to emulate different network bandwidths
  - 4:1 GPU:NIC bindings with staging through the CPU

Nodes	GPUs	Local Domain
8	32	36^4
16	64	36^3x18
32	128	36^2x18^2
64	256	36x18^3
128	512	18^4

Reference Configurations, System

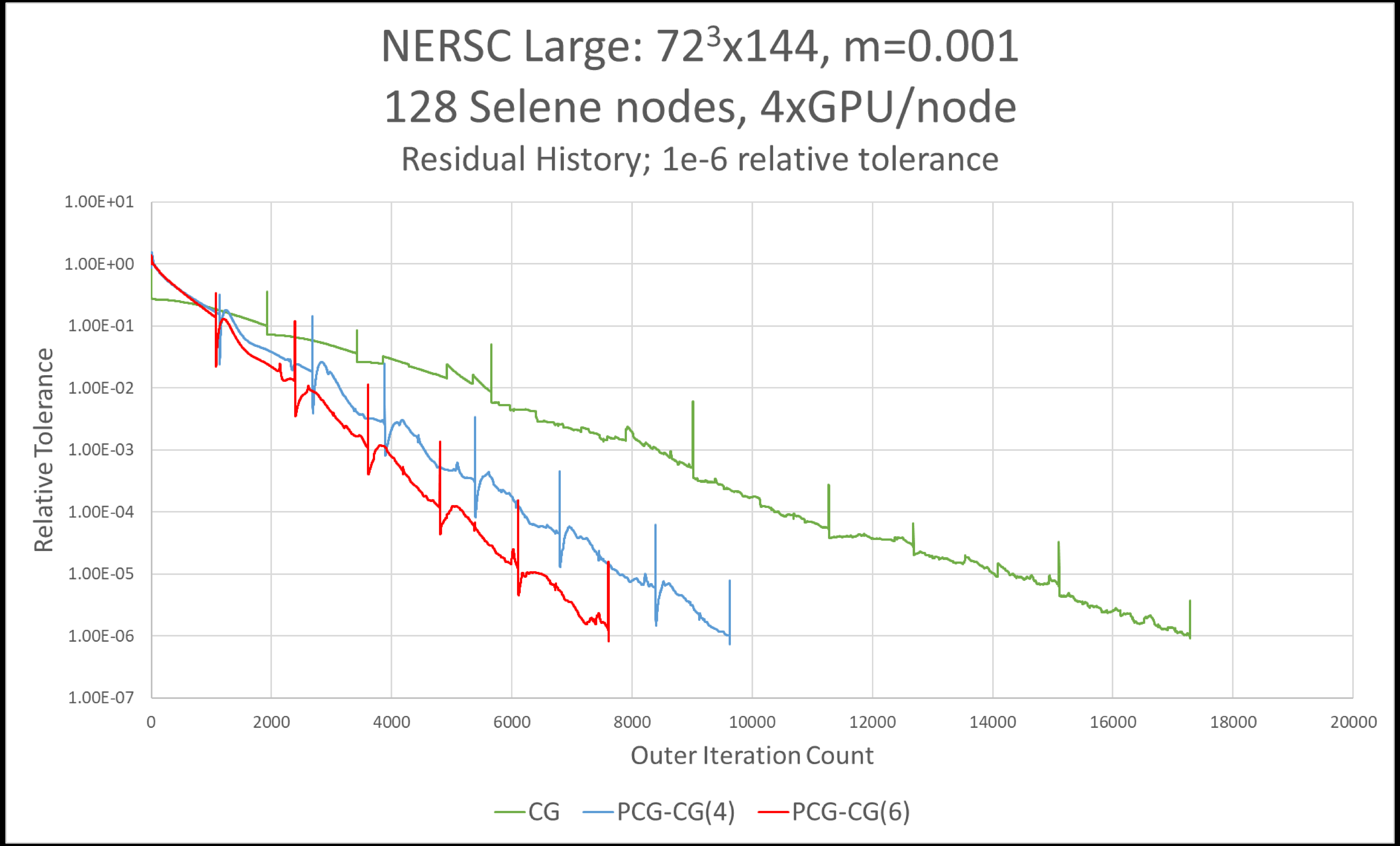
Solving at the speed of sound

- Configuration:
  - NERSC Large configuration
  - Volume: 72^3x144
  - Bare light mass $am = 0.001$
- Machine: Selene
  - DGX-A100-80GB nodes
  - Use 4xGPUs per node
  - 1:1 NIC ratio; HDR 200 (25 GB/s bi-directional)
- We consider multiple strong scaling problem sizes
- For networks:
  - 2:1 and 1:1 direct GPU:NIC bindings to emulate different network bandwidths
  - 4:1 GPU:NIC bindings with staging through the CPU
- All tests utilize **NVSHMEM**, implementations of the HISQ kernel
  - Device-driven communications
  - Reduces latency: no separate packing kernel, no overhead of MPI calls, gets the host out of the way

Nodes	GPUs	Local Domain
8	32	36^4
16	64	36^3x18
32	128	36^2x18^2
64	256	36x18^3
128	512	18^4

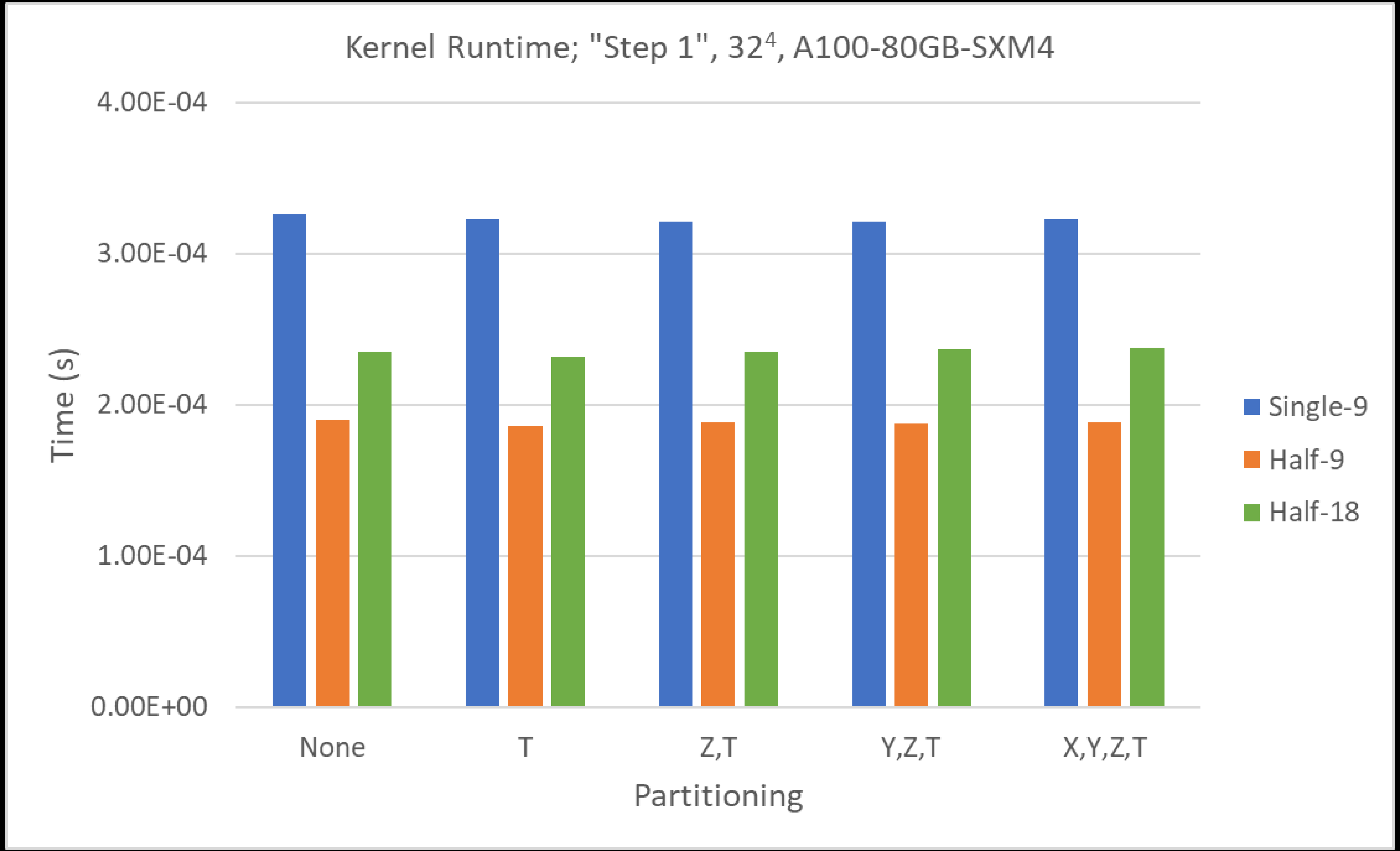
Convergence History

An unstable algorithm is pointless



- CG and PCG each converge in a stable fashion
- The "spikes" are due to residual updates: "every so often" we recompute the *exact* residual and re-inject it into the (P)CG solve

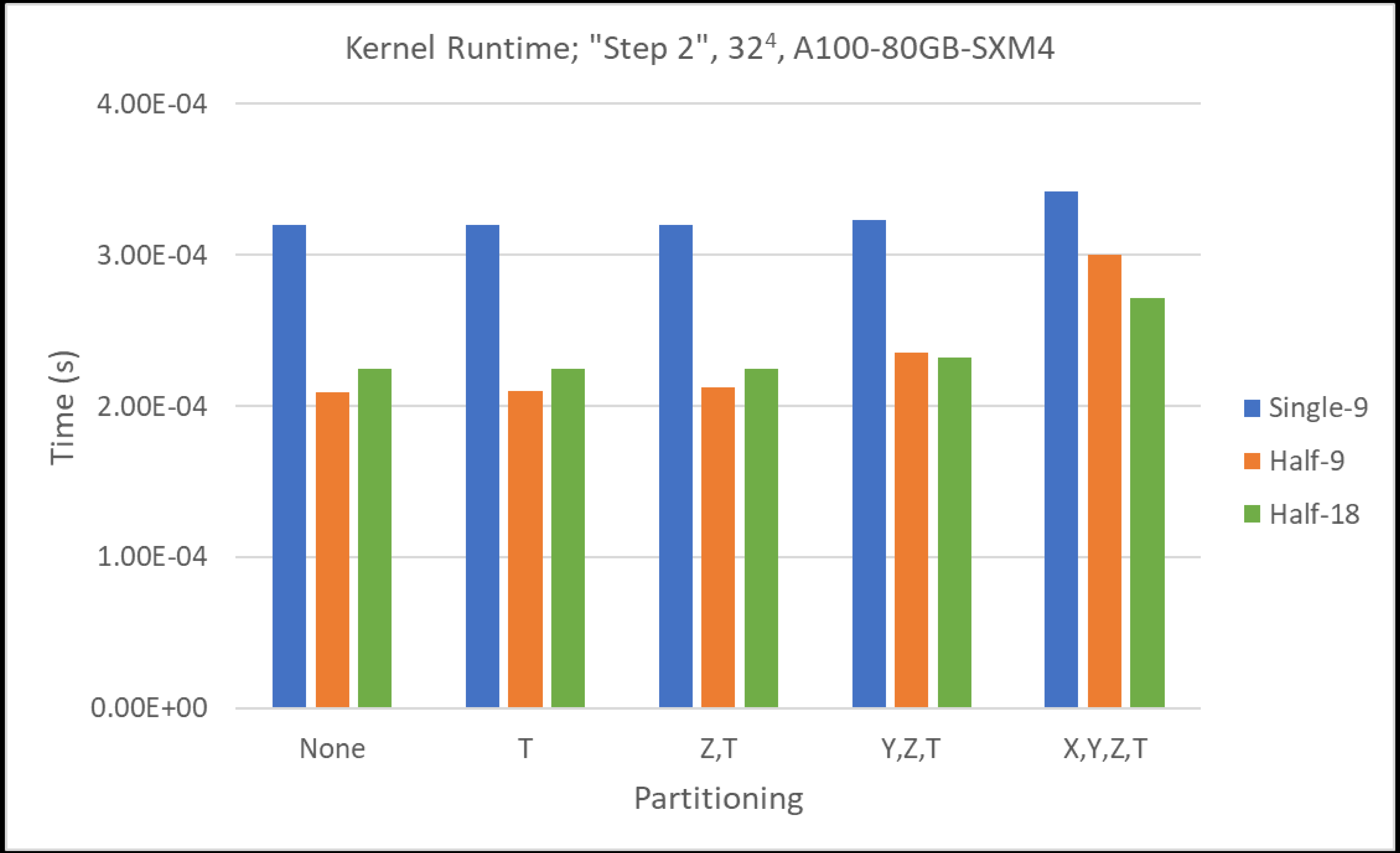
Operator Performance: Zero Boundary Conditions



Performance is essentially independent of the partitioning

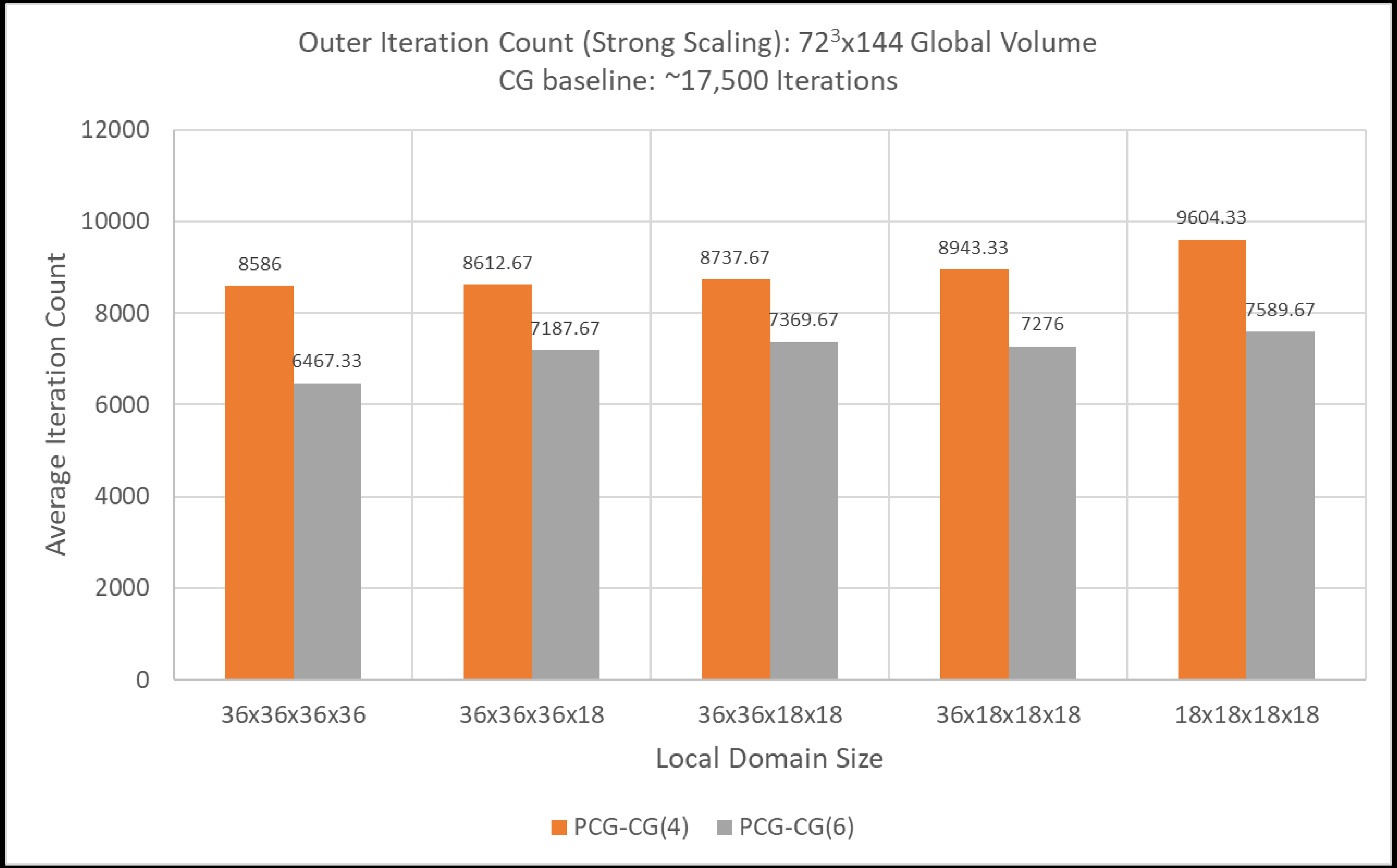
This makes sense: all we're doing is "snipping" away work

Operator Performance: Boundary Clovers



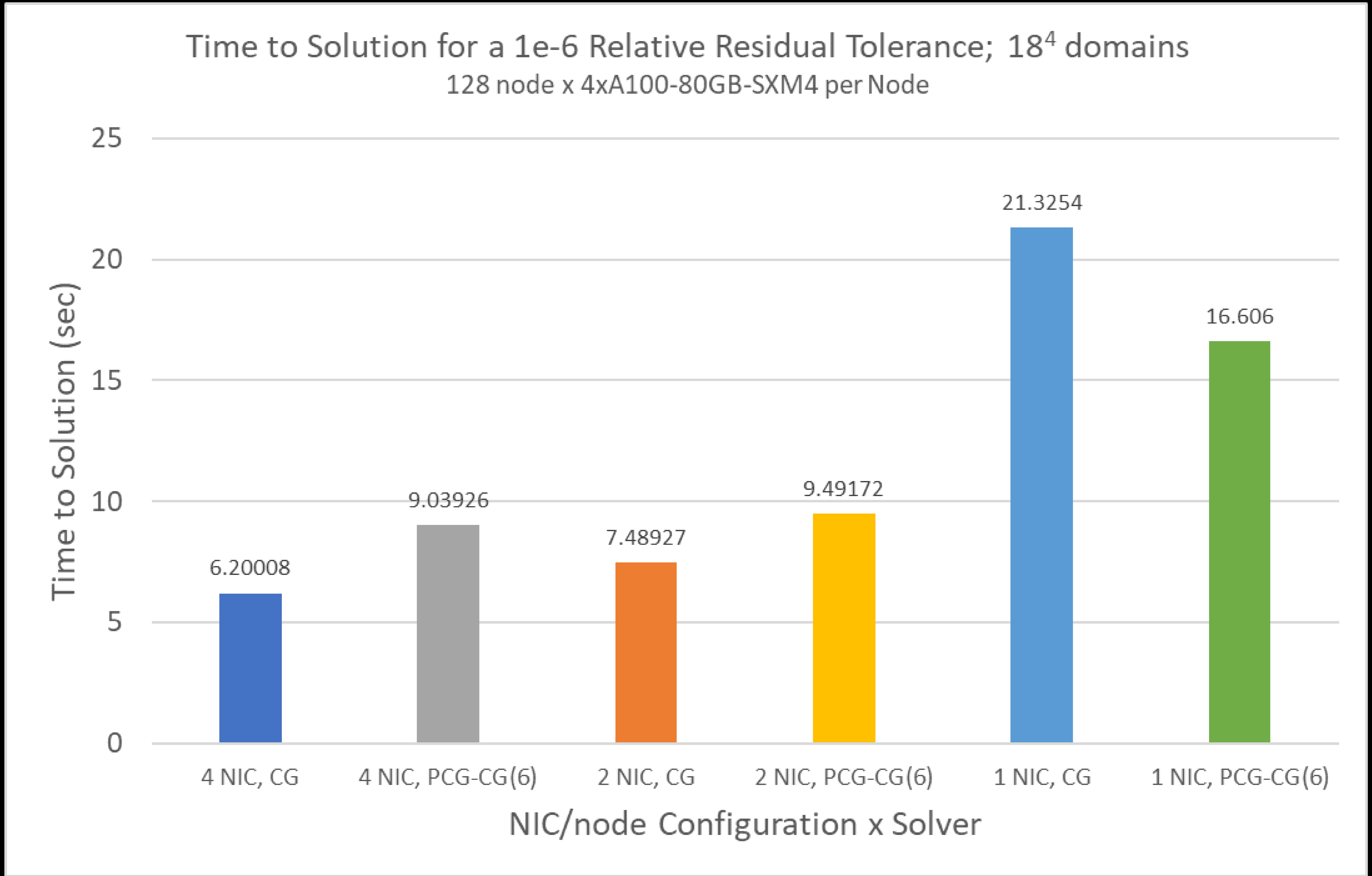
Performance decreases with partitioning
This makes sense: we're adding (divergent) work
Extra note: reconstruct becomes a *detriment:* extra instructions hold up threads

Iteration Counts for each Preconditioner



More preconditioner iterations -> fewer outer iterations (to a point)
Diminishing benefit with smaller partition sizes -> domain is a lower-quality approximation of full domain

Time to Solution (which is all that matters)



Note: 1xNIC includes CPU staging for two GPUs to access a NIC!
There's still outstanding work to be done when the network is strong (25 GB/s bi-directional per NIC)...
...but we also see that the preconditioner is beneficial when the network is slow

Future Work

Future HISQy Business

Same old song and dance

- HISQ Force: no further optimizations
- Schwarz Preconditioner: Pre-computed matrix products to reduce latencies
- HISQ MG + Schwarz Preconditioner:
  - Use the local operator as a smoother on all levels
  - Outer HISQ *and Kahler-Dirac preconditioned operator* have GPU code implementations
  - Even/odd preconditioned coarse operators do not
- …192^3x384 ensemble

NVIDIA®