Stanford | ENGINEERING
Electrical Engineering

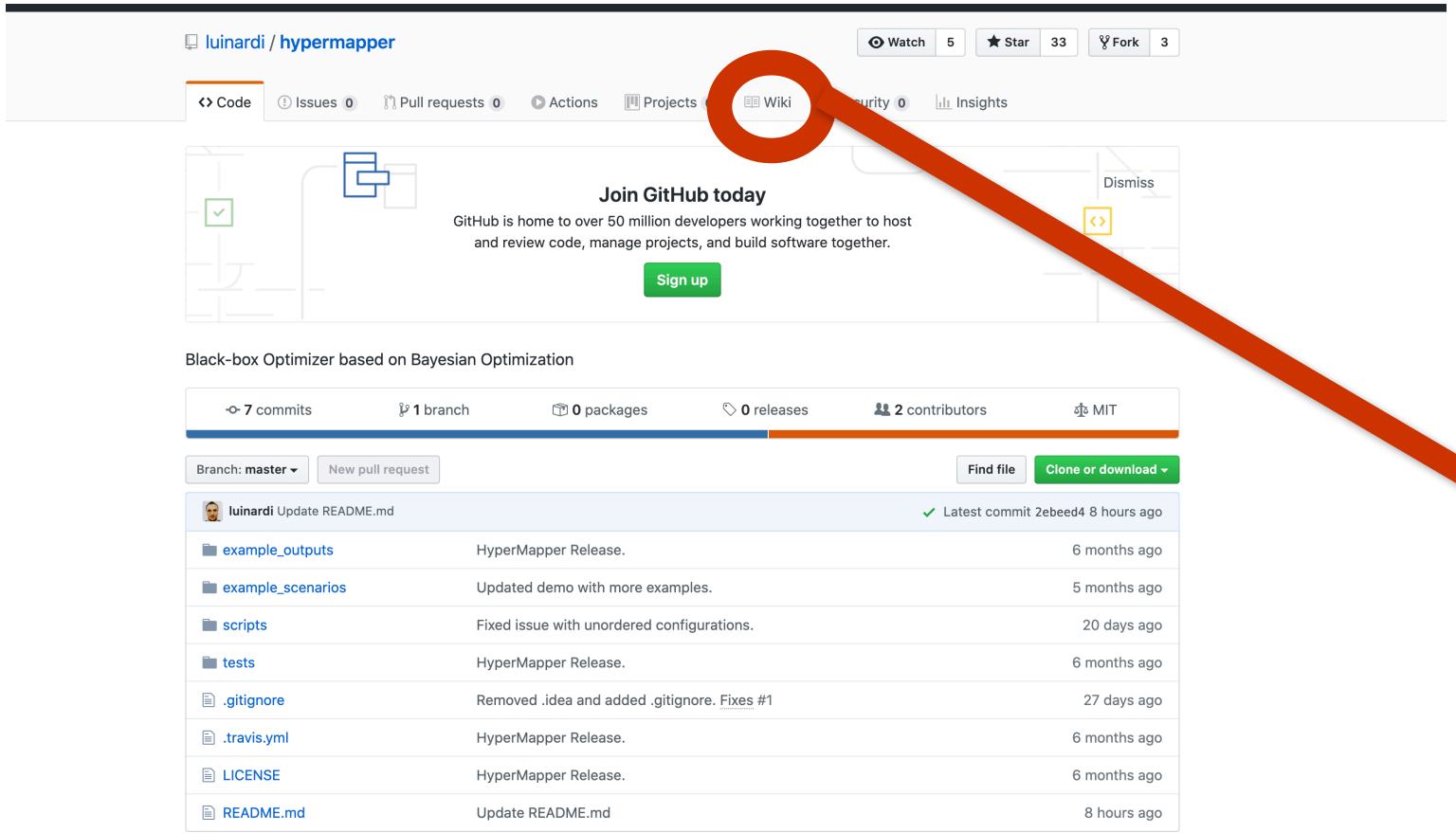
STANFORD DAWN

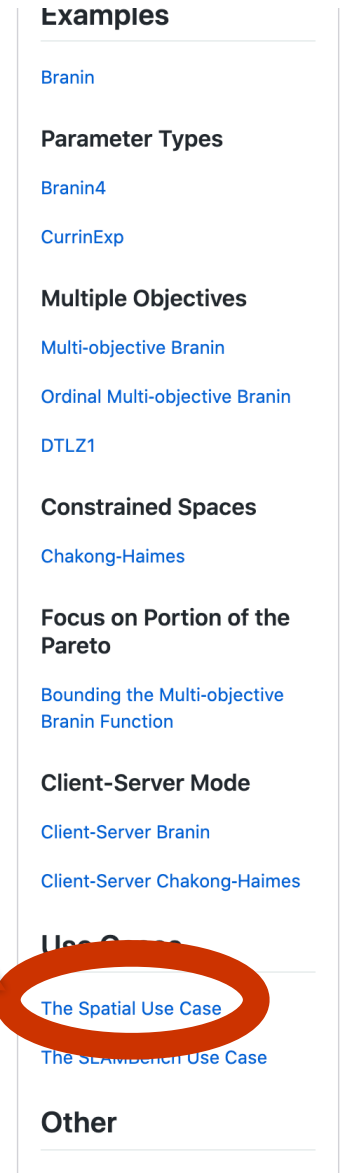
Hands-on: Demo Spatial (GeMM)

Matthew Feldman, Luigi Nardi, Artur Souza, Kunle Olukotun

HyperMapper + Spatial Tutorial

- This GeMM app is ready for you to experiment with!
  - https://github.com/luinardi/hypermapper



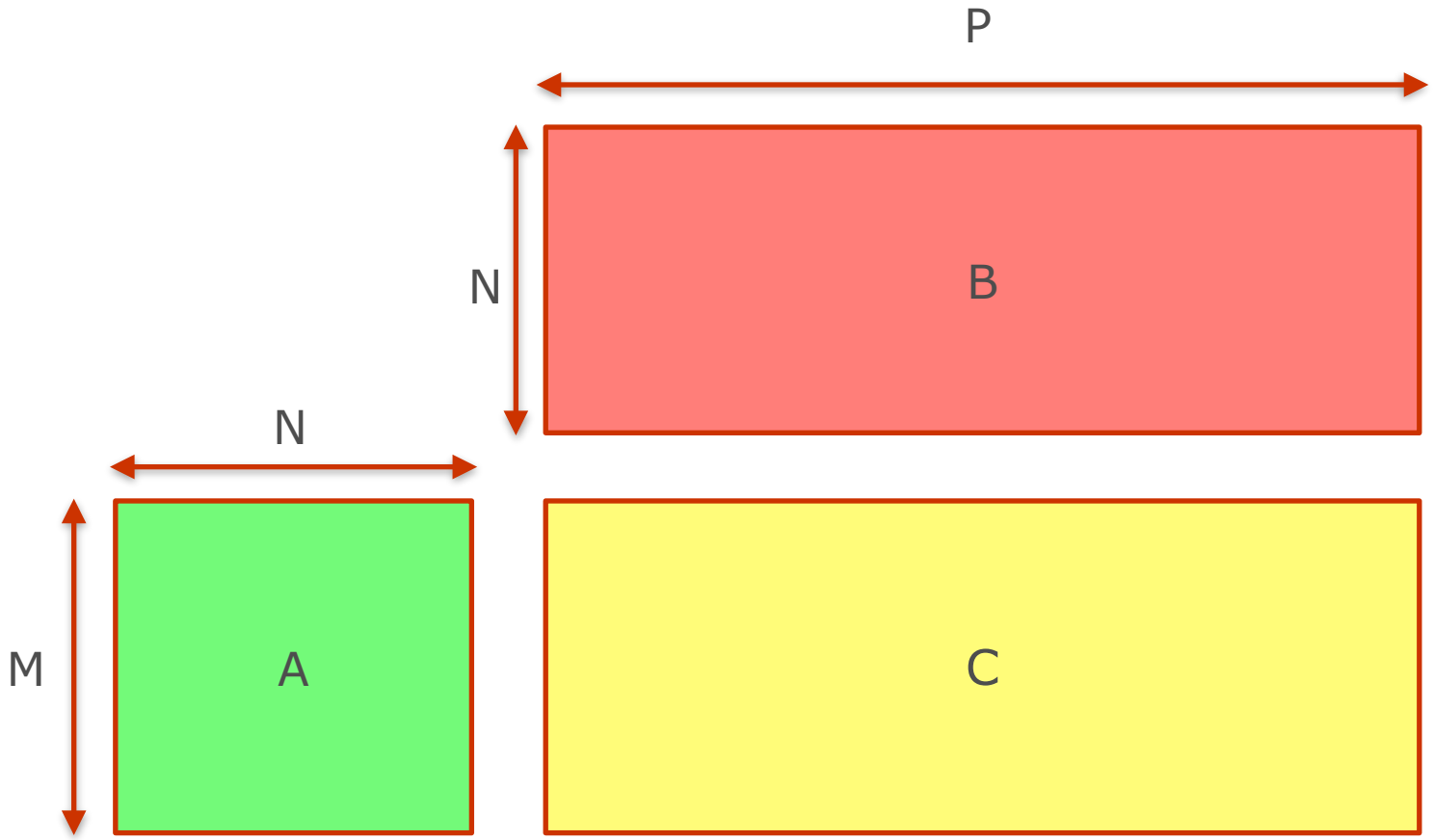


GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*

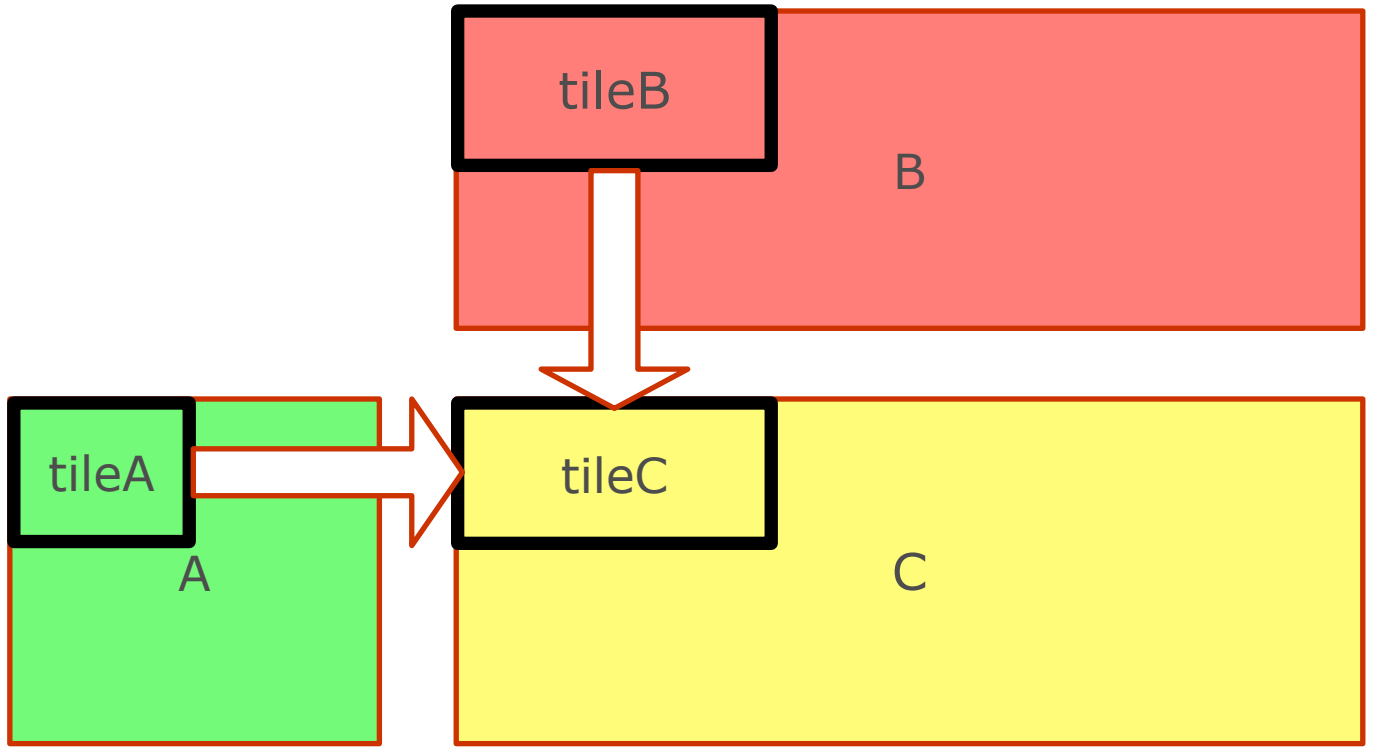
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



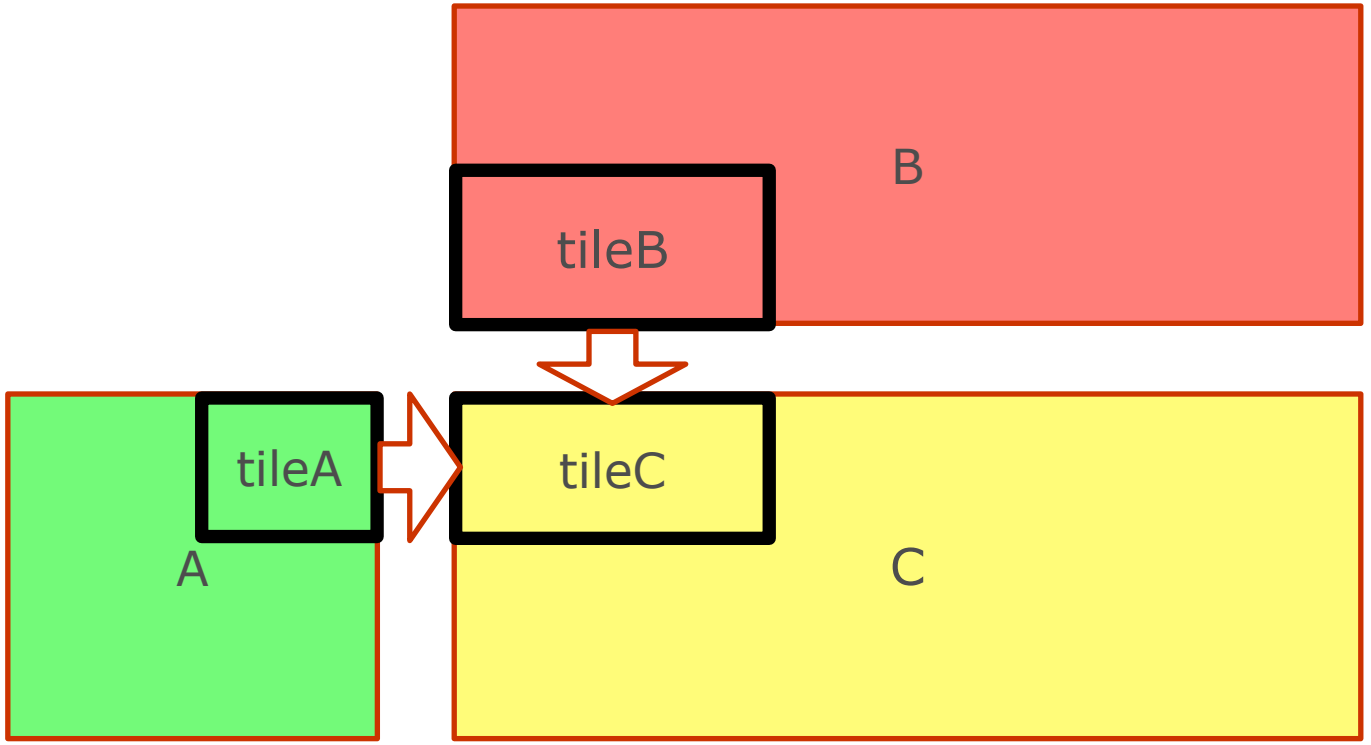
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



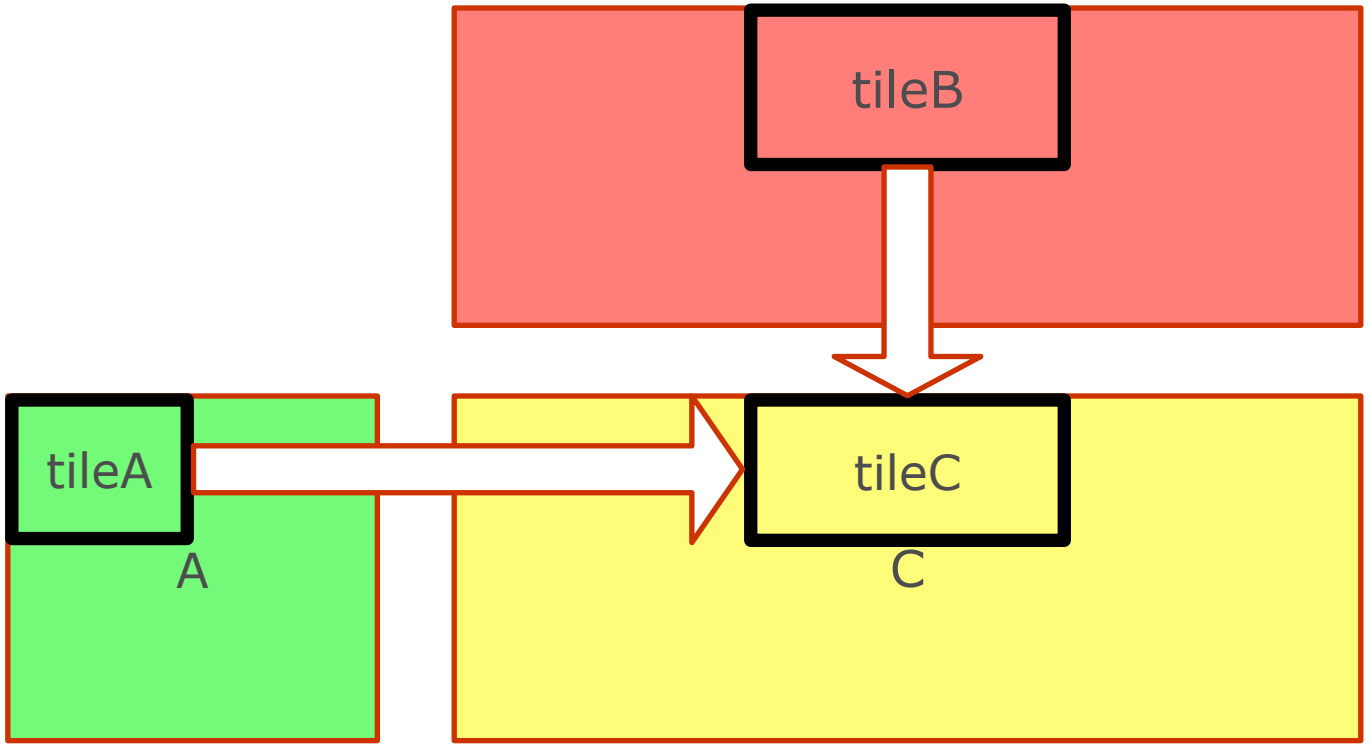
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



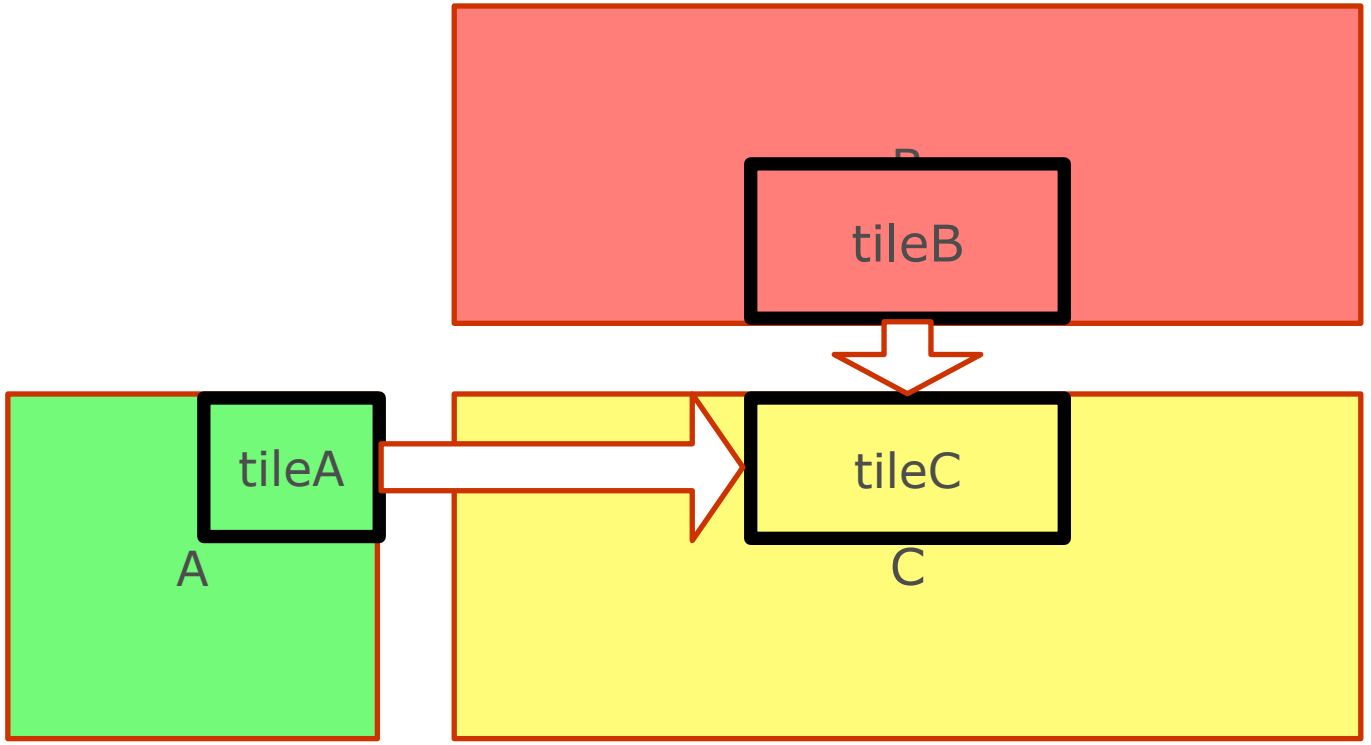
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



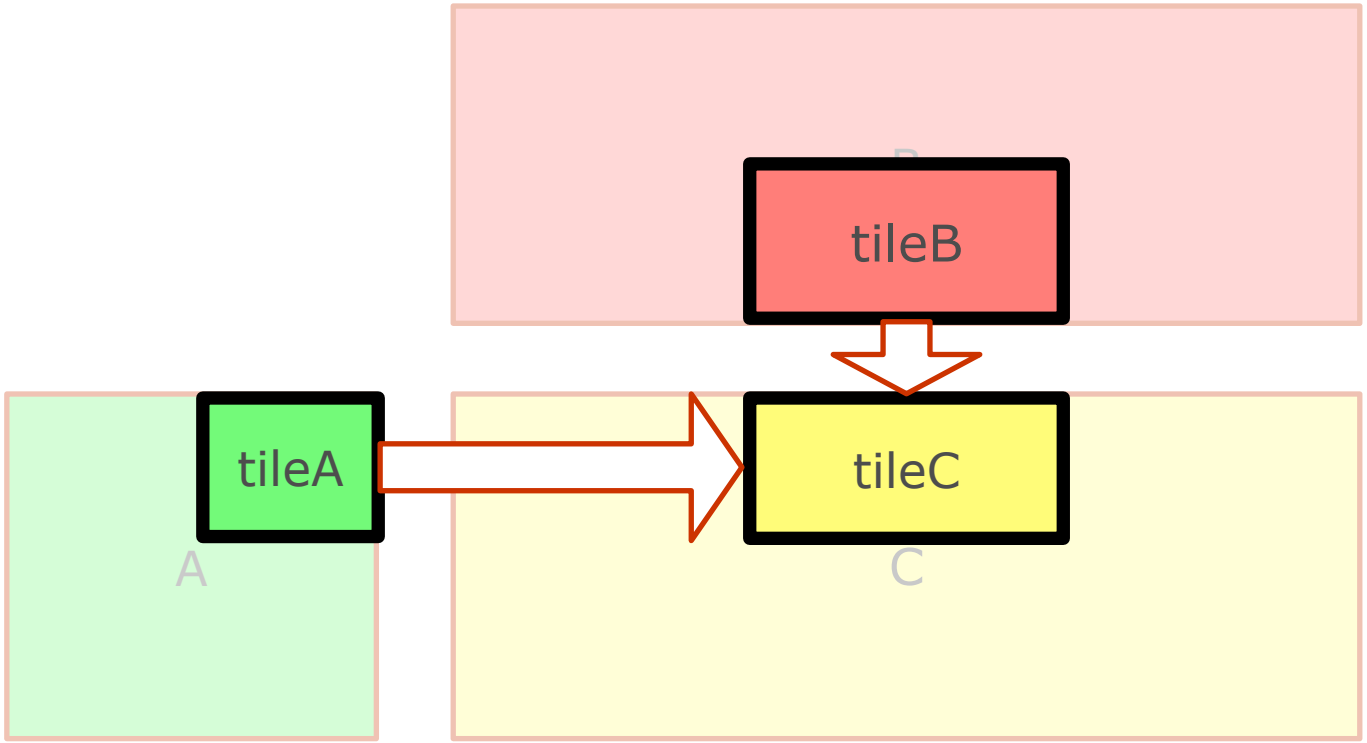
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



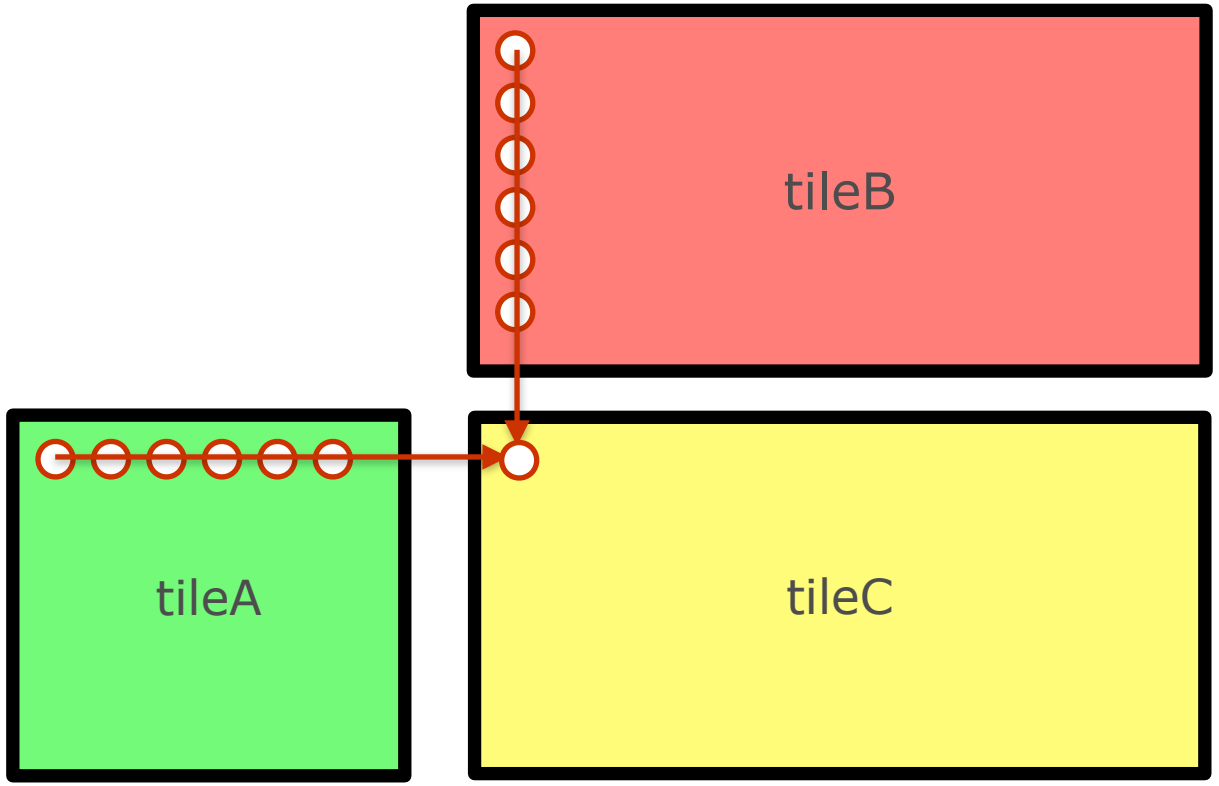
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



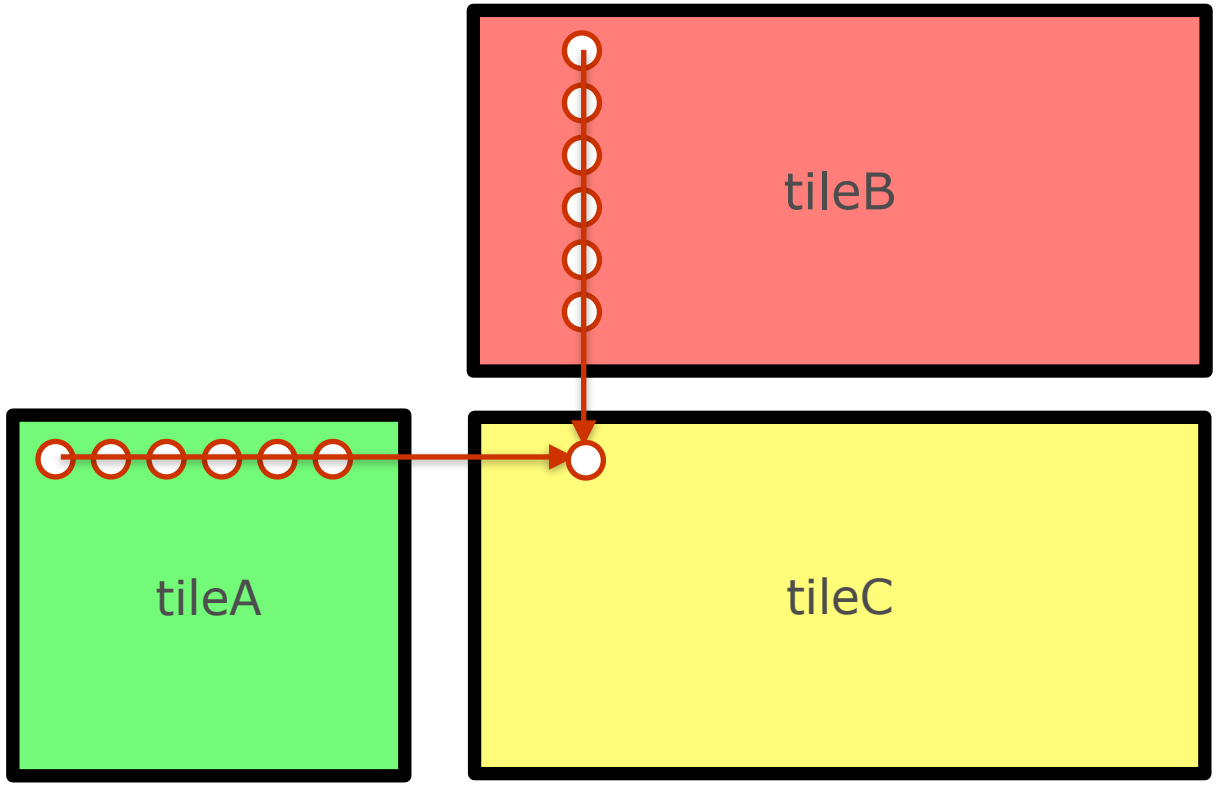
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



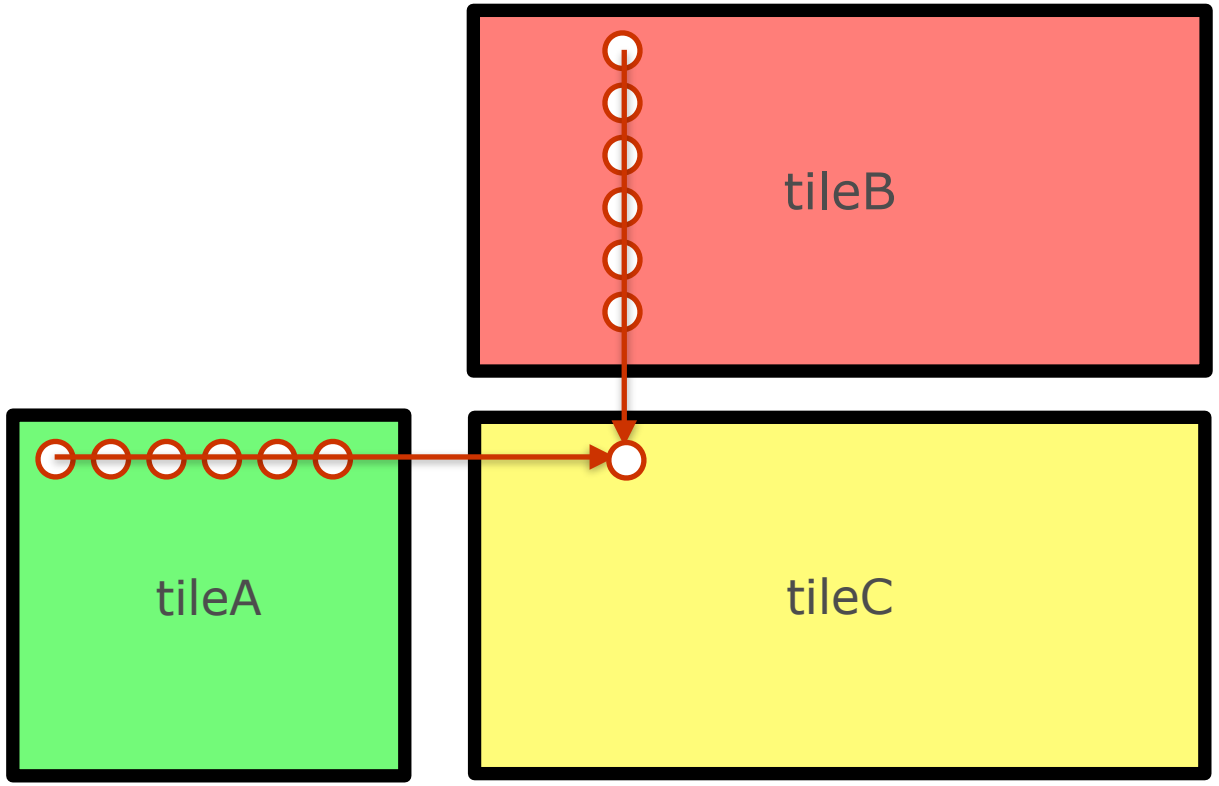
GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*



GeMM in Hardware

- **Task**: Compute $A_{MxN} * B_{NxP} = C_{MxP}$
- Let's solve this using *tiling* and *inner products*

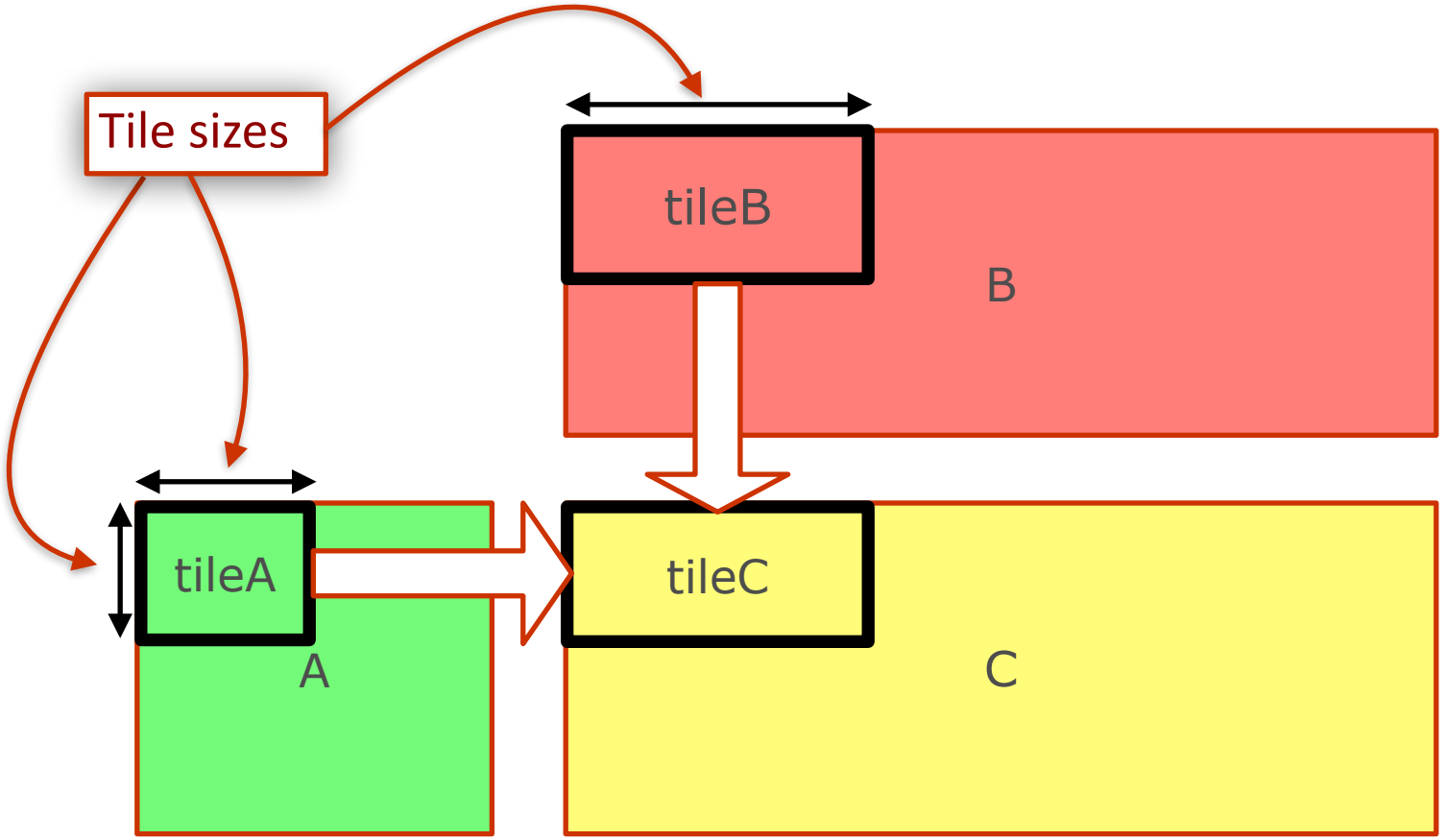


GeMM in Hardware

- There are a few obvious design choices…

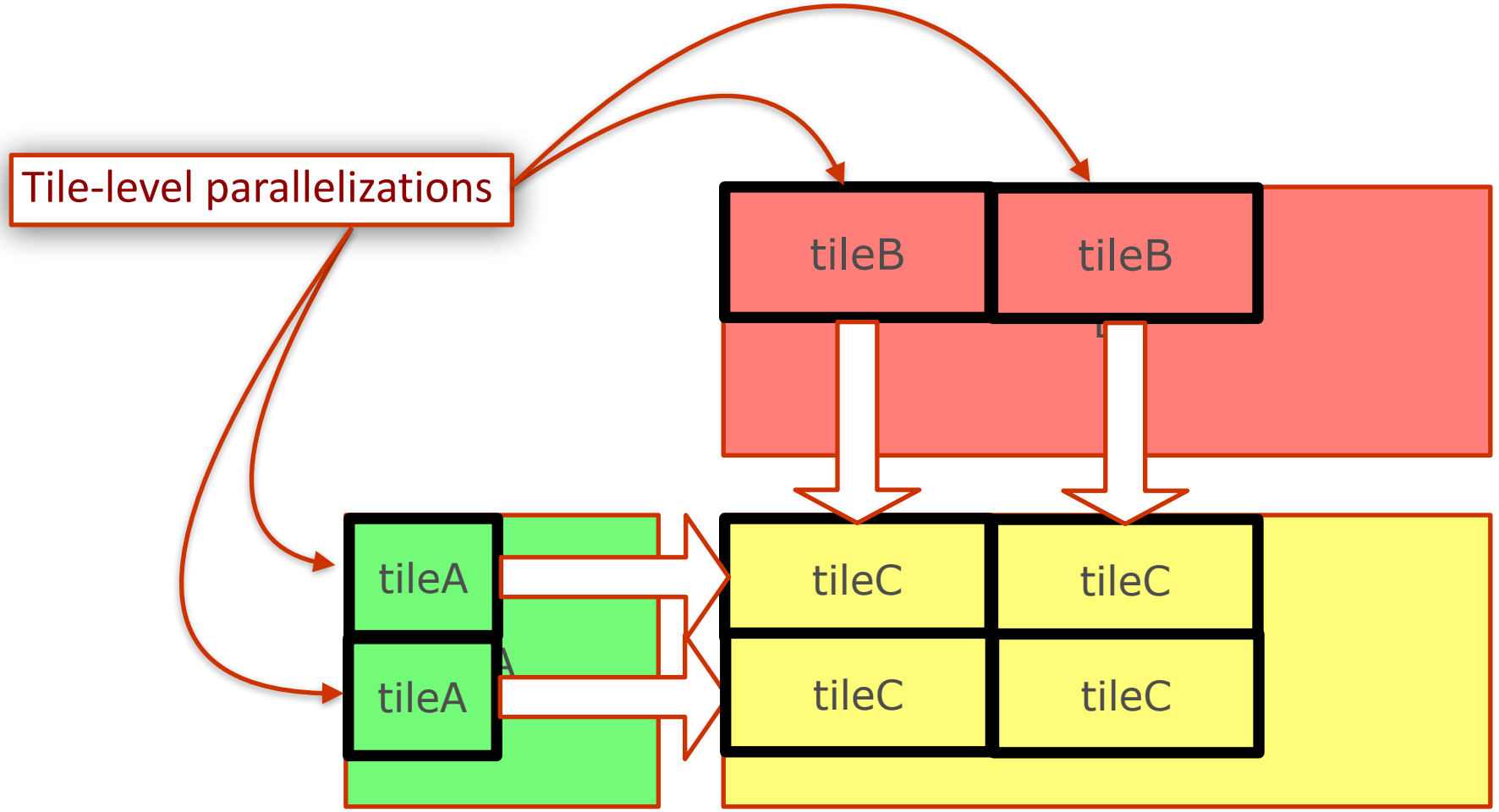
GeMM in Hardware

- There are a few obvious design choices…



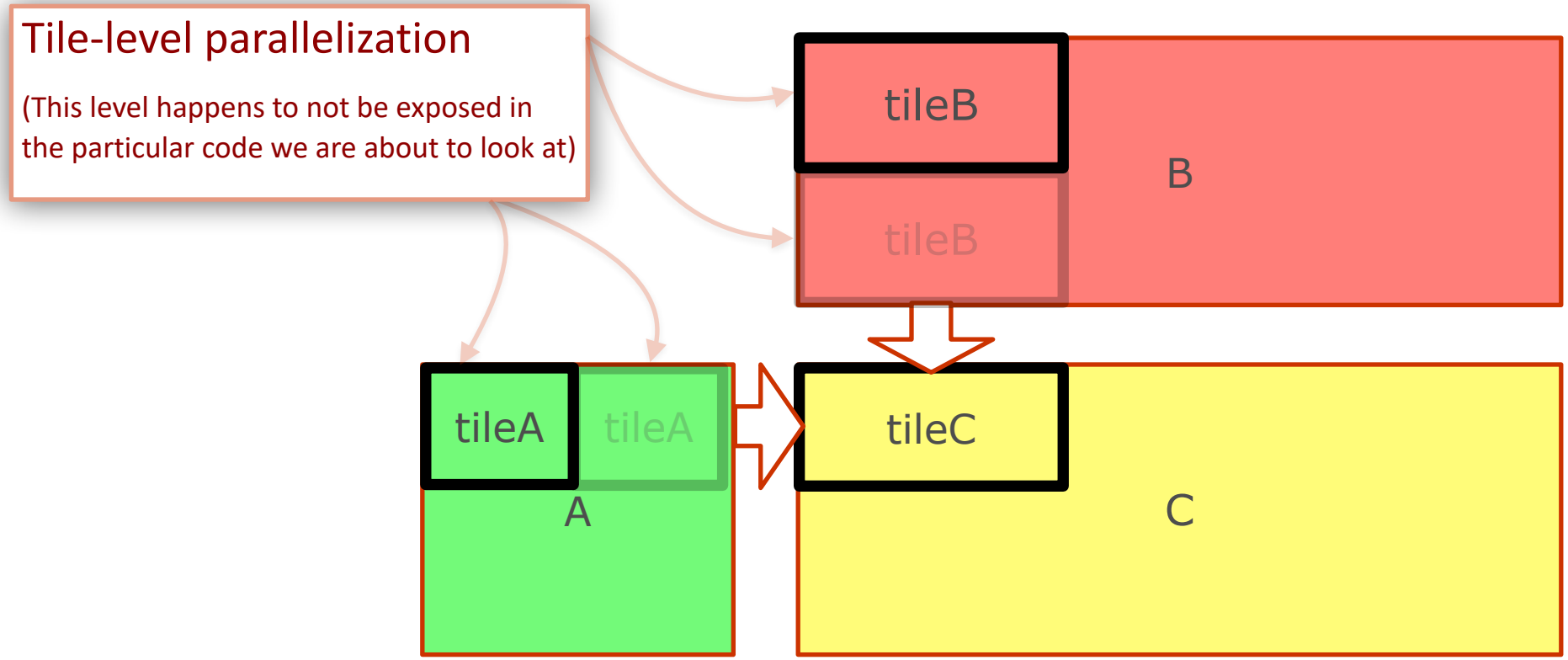
GeMM in Hardware

- There are a few obvious design choices…



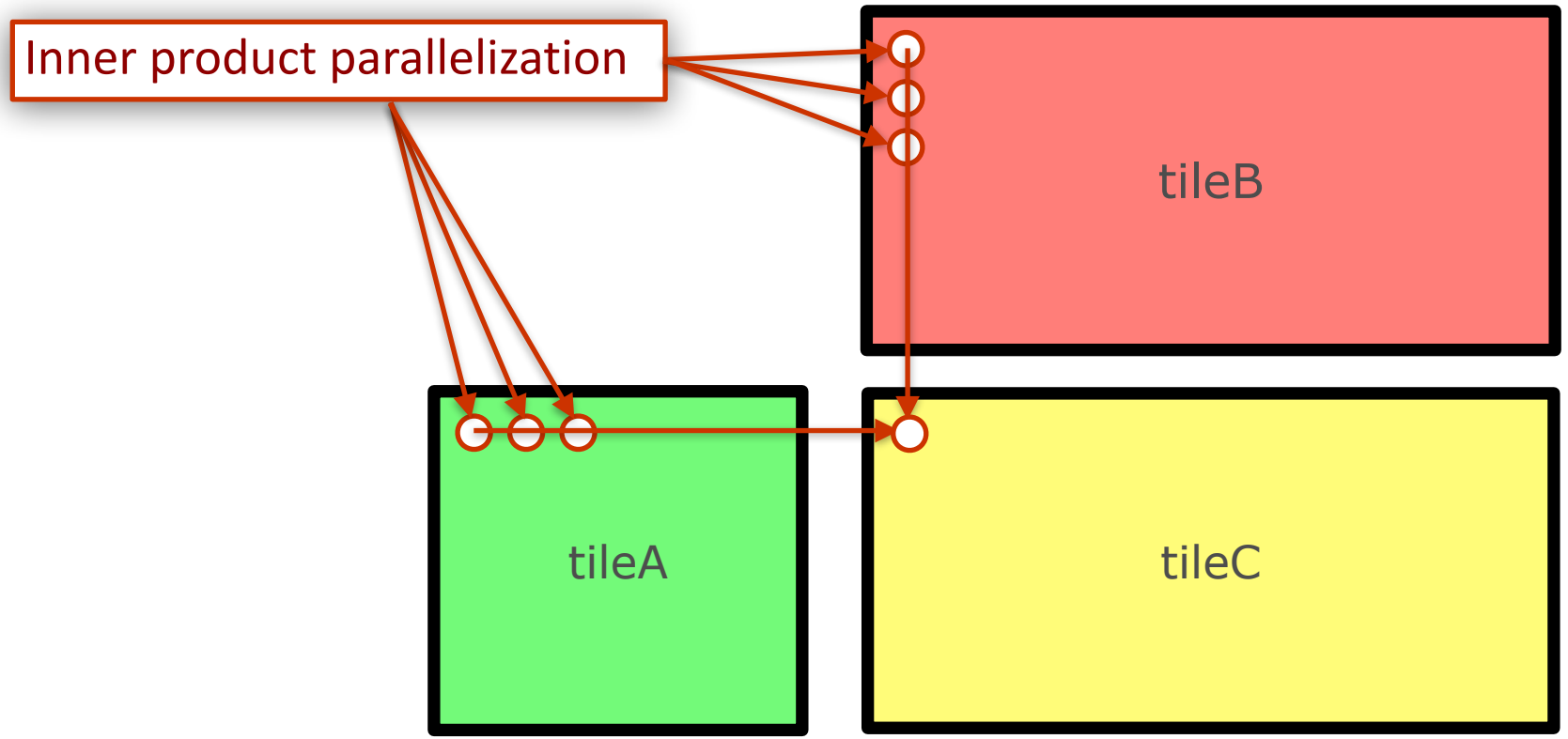
GeMM in Hardware

- There are a few obvious design choices…



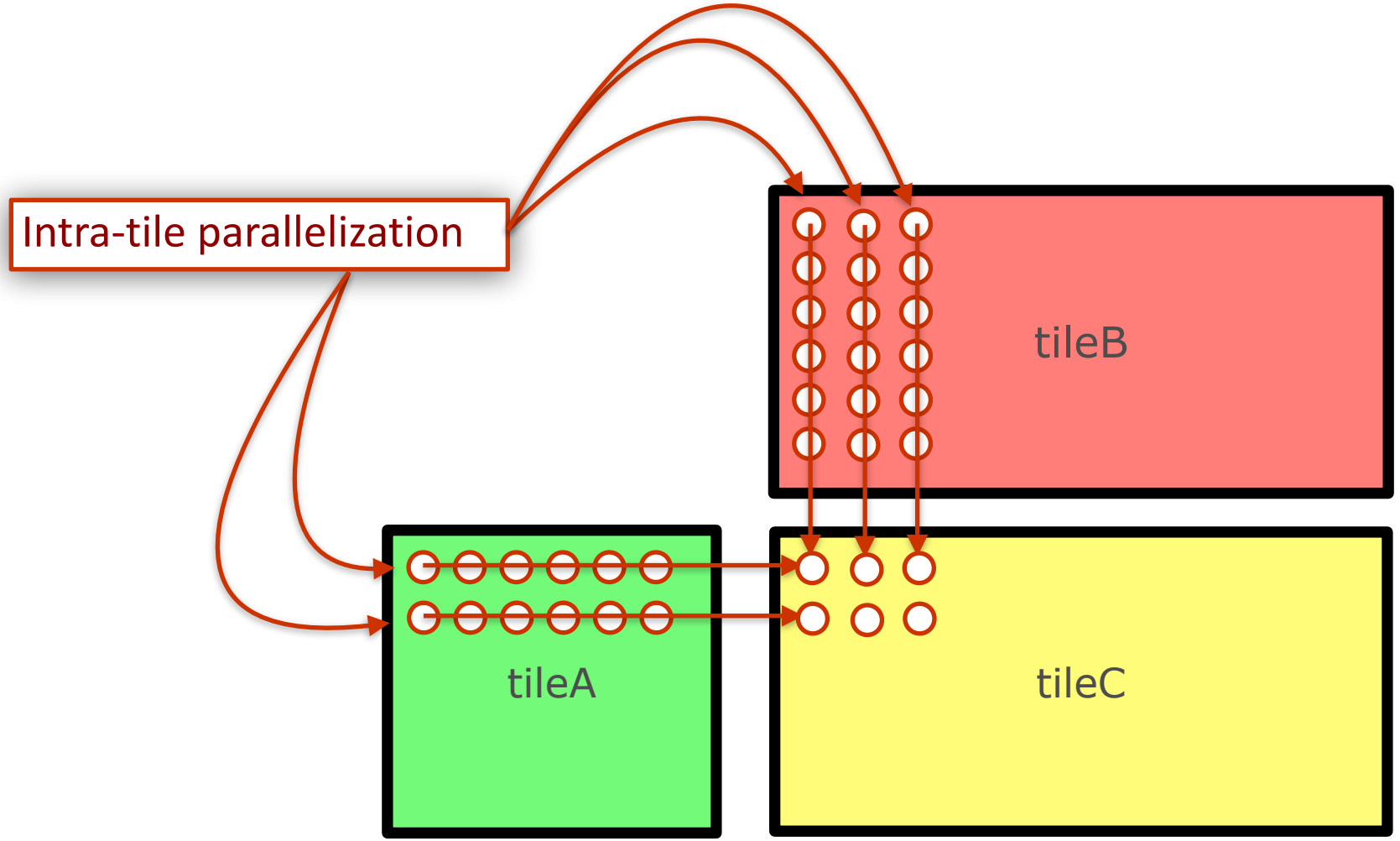
GeMM in Hardware

- There are a few obvious design choices…



GeMM in Hardware

- There are a few obvious design choices…



GeMM in Hardware

- There are a few obvious design choices…
  - Tile sizes
  - Tile-level parallelization
  - Intra-tile parallelization
  - Inner product parallelization

GeMM in Hardware

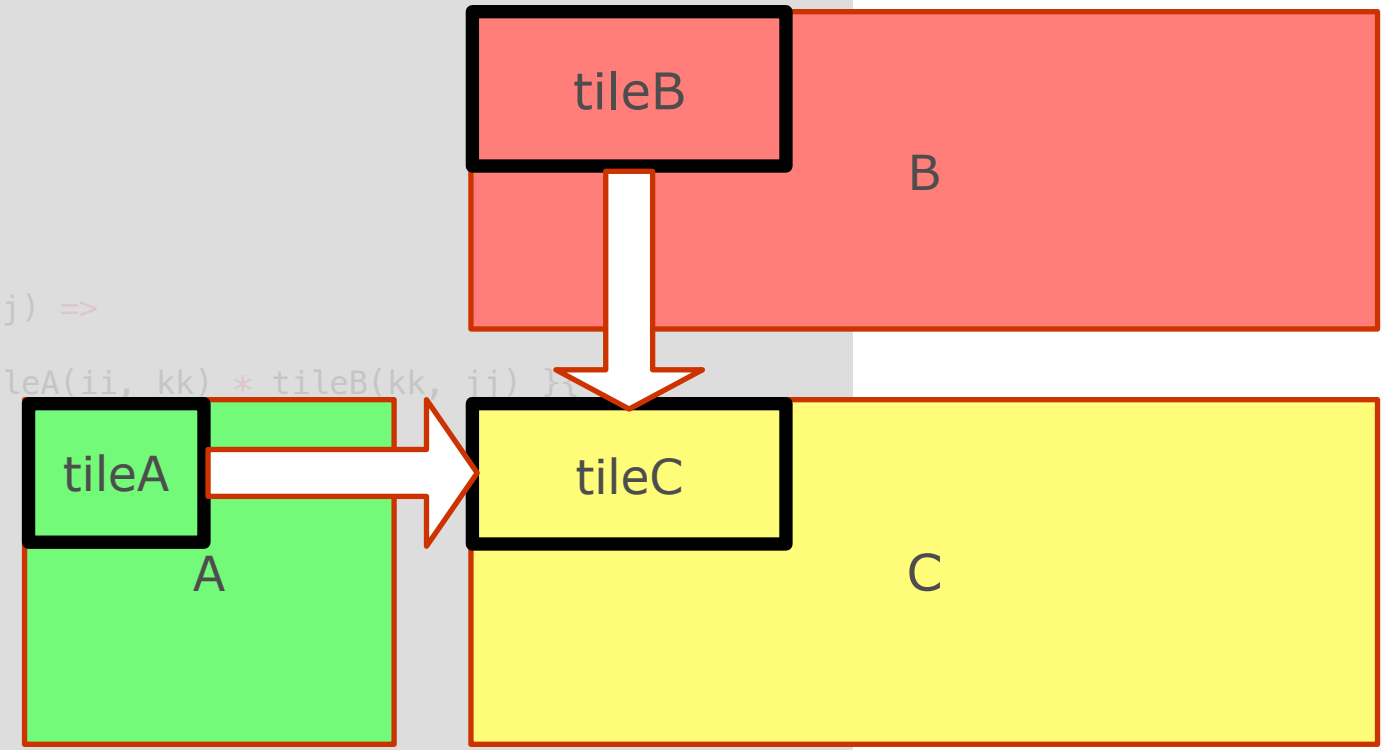
- There are a few obvious design choices…
  - Tile sizes
  - Tile-level parallelization
  - Intra-tile parallelization
  - Inner product parallelization
- … And some less obvious choices
  - Outer controller scheduling (Sequential or Pipeline)
  - DRAM transfer parallelization

GeMM in Hardware - Spatial App

```
Foreach(M by bm par tileM_par, P by bp par tileP_par){(i,j) =>
 val tileC = SRAM[T](bm, bp)
 Foreach(N by bn){k =>
   val tileA = SRAM[T](bm, bn)
   val tileB = SRAM[T](bn, bp)
   tileA load A(i::i+bm, k::k+bn par load_par)
   tileB load B(k::k+bn, j::j+bp par load_par)
   Foreach(bm by 1 par M_par, bp by 1 par P_par){ (ii,jj) =>
     val prod = Reduce(Reg[T])(bn by 1 par ip){kk => tileA(ii, kk) * tileB(kk, jj) }{_+_}
     val prev = mux(k == 0, 0.to[T], tileC(ii,jj))
     tileC(ii,jj) = prev + prod.value
   }
 }
 C(i::i+bm, j::j+bp par store_par) store tileC
```

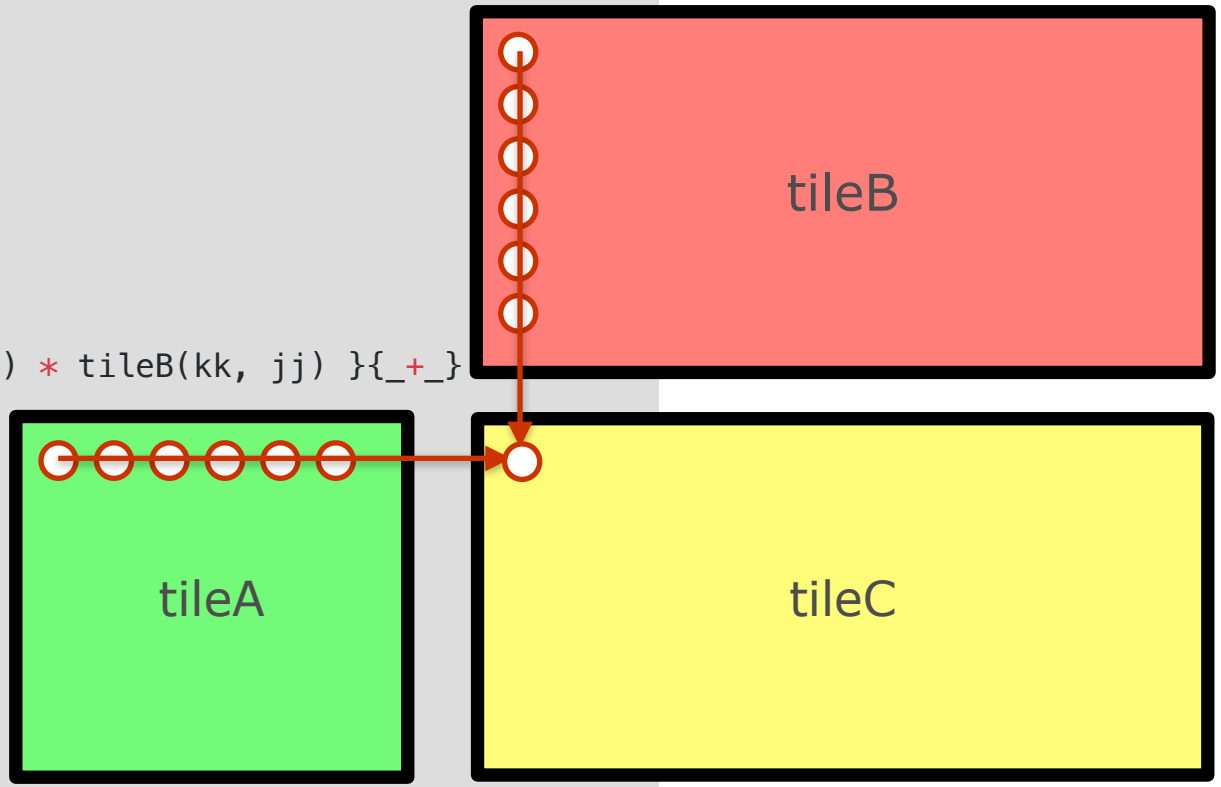
GeMM in Hardware - Spatial App

```
Foreach(M by bm par tileM_par, P by bp par tileP_par){(i,j) =>
 val tileC = SRAM[T](bm, bp)
 Foreach(N by bn){k =>
   val tileA = SRAM[T](bm, bn)
   val tileB = SRAM[T](bn, bp)
   tileA load A(i::i+bm, k::k+bn par load_par)
   tileB load B(k::k+bn, j::j+bp par load_par)
   Foreach(bm by 1 par M_par, bp by 1 par P_par){ (ii,jj) =>
     val prod = Reduce(Reg[T])(bn by 1 par ip){kk => tileA(ii, kk) * tileB(kk, jj) }
     val prev = mux(k == 0, 0.to[T], tileC(ii,jj))
     tileC(ii,jj) = prev + prod.value
   }
 }
 C(i::i+bm, j::j+bp par store_par) store tileC
```



GeMM in Hardware - Spatial App

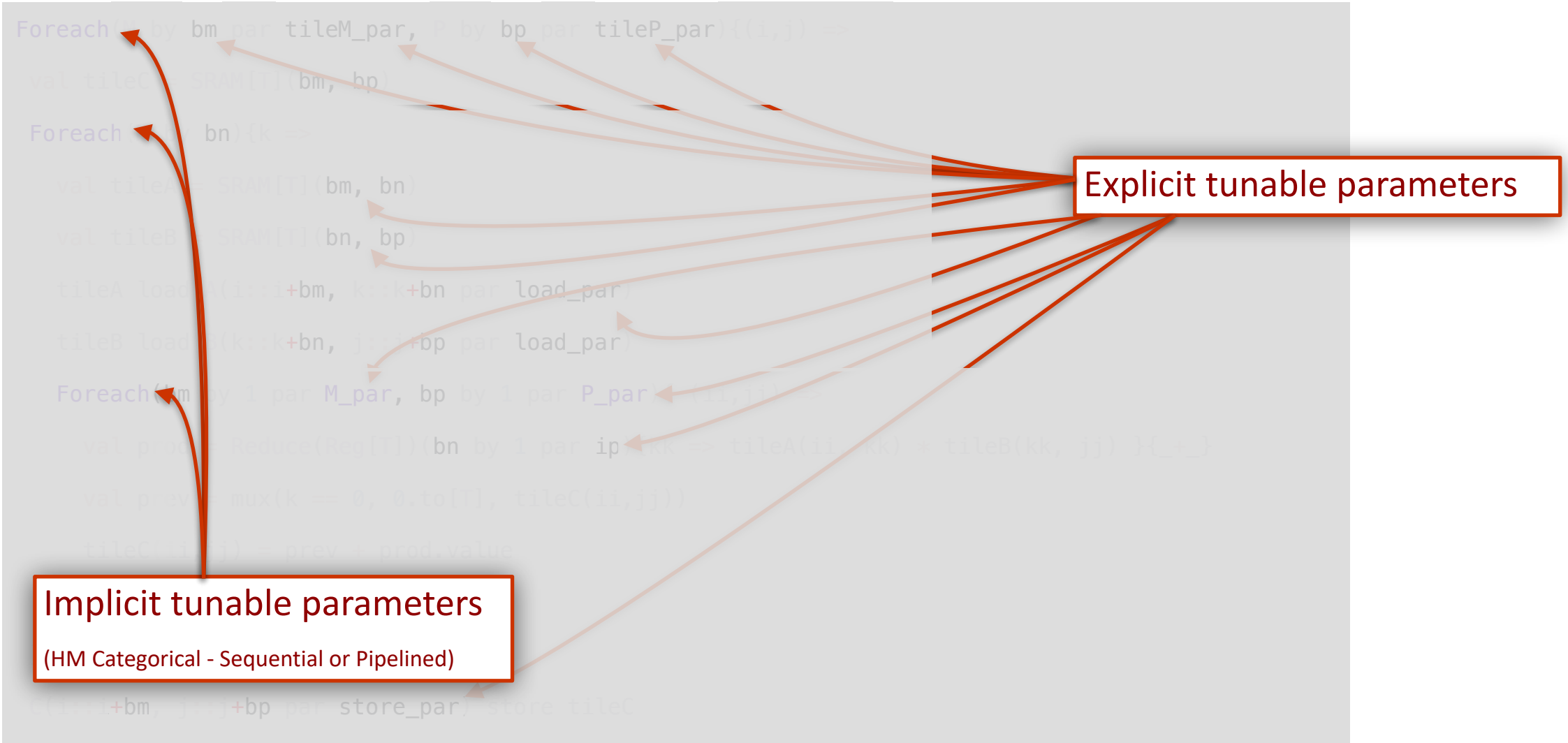
```
Foreach(M by bm par tileM_par, P by bp par tileP_par){(i,j) =>
 val tileC = SRAM[T](bm, bp)
 Foreach(N by bn){k =>
   val tileA = SRAM[T](bm, bn)
   val tileB = SRAM[T](bn, bp)
   tileA load A(i::i+bm, k::k+bn par load_par)
   tileB load B(k::k+bn, j::j+bp par load_par)
   Foreach(bm by 1 par M_par, bp by 1 par P_par){ (ii,jj) =>
     val prod = Reduce(Reg[T])(bn by 1 par ip){kk => tileA(ii, kk) * tileB(kk, jj) }{_+_}
     val prev = mux(k == 0, 0.to[T], tileC(ii,jj))
     tileC(ii,jj) = prev + prod.value
   }
 }
 C(i::i+bm, j::j+bp par store_par) store tileC
```



GeMM in Hardware - Spatial App

```
Foreach(M by bm par tileM_par, P by bp par tileP_par){(i,j) =>
 val tileC = SRAM[T](bm, bp)
 Foreach(N by bn){k =>
   val tileA = SRAM[T](bm, bn)
   val tileB = SRAM[T](bn, bp)
   tileA load A(i::i+bm, k::k+bn par load_par)
   tileB load B(k::k+bn, j::j+bp par load_par)
   Foreach(bm by 1 par M_par, bp by 1 par P_par){ (ii,jj) =>
     val prod = Reduce(Reg[T])(bn by 1 par ip){kk => tileA(ii, kk) * tileB(kk, jj) }{_+_}
     val prev = mux(k == 0, 0.to[T], tileC(ii,jj))
     tileC(ii,jj) = prev + prod.value
   }
 }
 C(i::i+bm, j::j+bp par store_par) store tileC
```

GeMM in Hardware - Spatial App



GeMM in Hardware - Spatial App

- Parameter spaces can be specified:

```
// Expose tile sizes as DSE parameters
val bm = 16 (1 -> 128)
val bn = 16 (16 -> 16 -> 1028)
val bp = 16 (16 -> 16 -> 1028)

// Expose parallelization factors as DSE parameters
val tileM_par = 2   (1 -> 6)
val tileP_par = 2   (1 -> 6)
val M_par = 2 (1 -> 8)
val P_par = 2 (1 -> 8)
val ip = 2  (1 -> 64)
val load_par = 4   (1,2,4,8,16)
val store_par = 4   (1,2,4,8,16)
```

GeMM in Hardware - Spatial App

- Parameter spaces can be specified:

```
// Expose tile sizes as DSE parameters
val bm = 16 (1 -> 128)
val bn = 16 (16 -> 16 -> 1028)
val bp = 16 (16 -> 16 -> 1028)

// Expose parallelization factors as DSE parameters
val tileM_par = 2   (1 -> 6)
val tileP_par = 2   (1 -> 6)
val M_par = 2 (1 -> 8)
val P_par = 2 (1 -> 8)
val ip = 2  (1 -> 64)
val load_par = 4    (1,2,4,8,16)
val store_par = 4   (1,2,4,8,16)
```

Start -> Stop range (HM Integer)

Start -> Step -> Stop range (HM Ordinal)

Explicit possibilities (HM Ordinal)

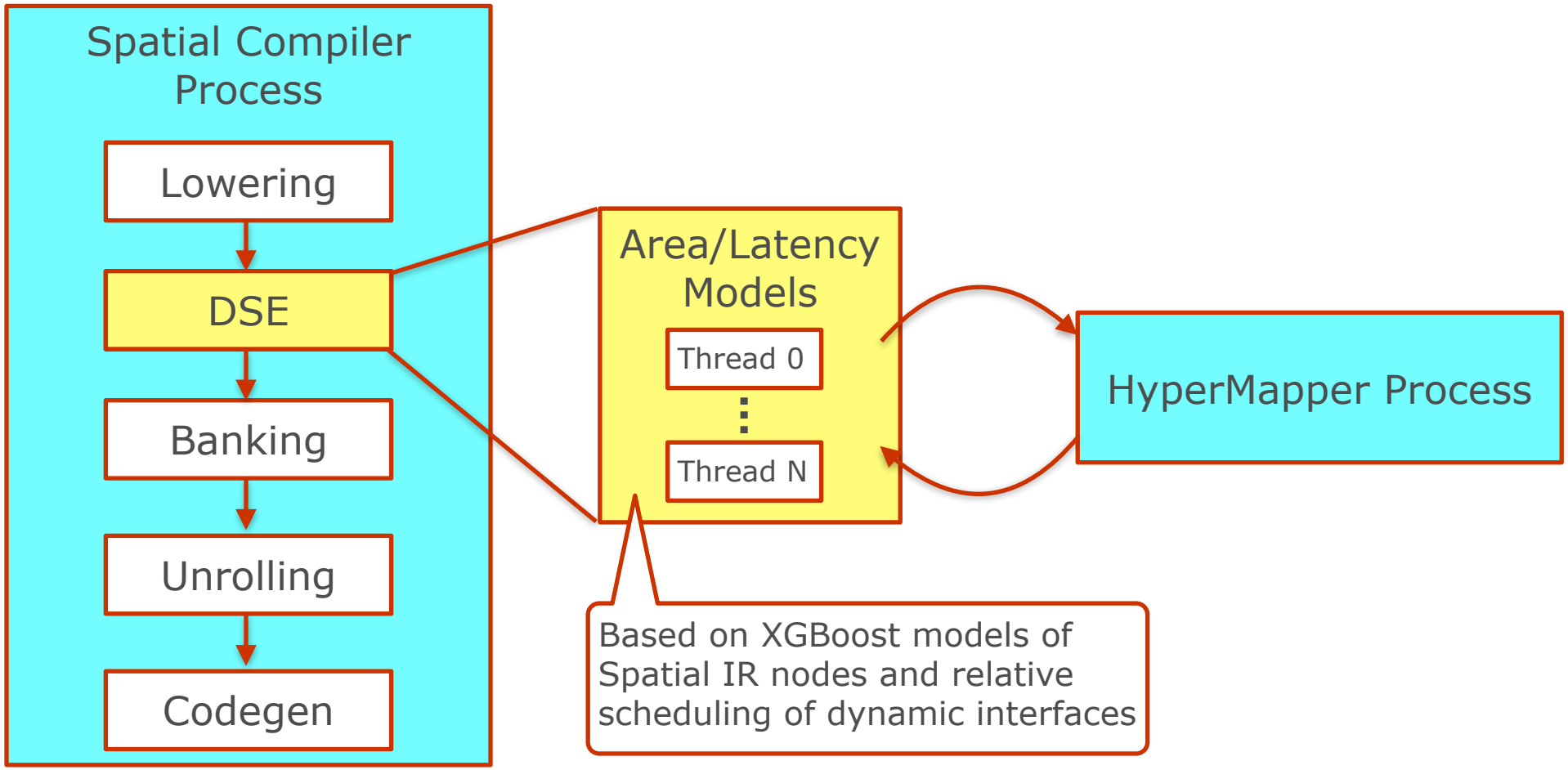
GeMM in Hardware - Spatial App

- Default values for dynamic variables can be set:

```
// Create input arguments for sizing DRAM regions (matrices)

val m = args(0).to[Int]
val n = args(1).to[Int]
val p = args(2).to[Int]

// Set default values for runtime args

val bound(m) = 512
val bound(n) = 256
val bound(p) = 512
```

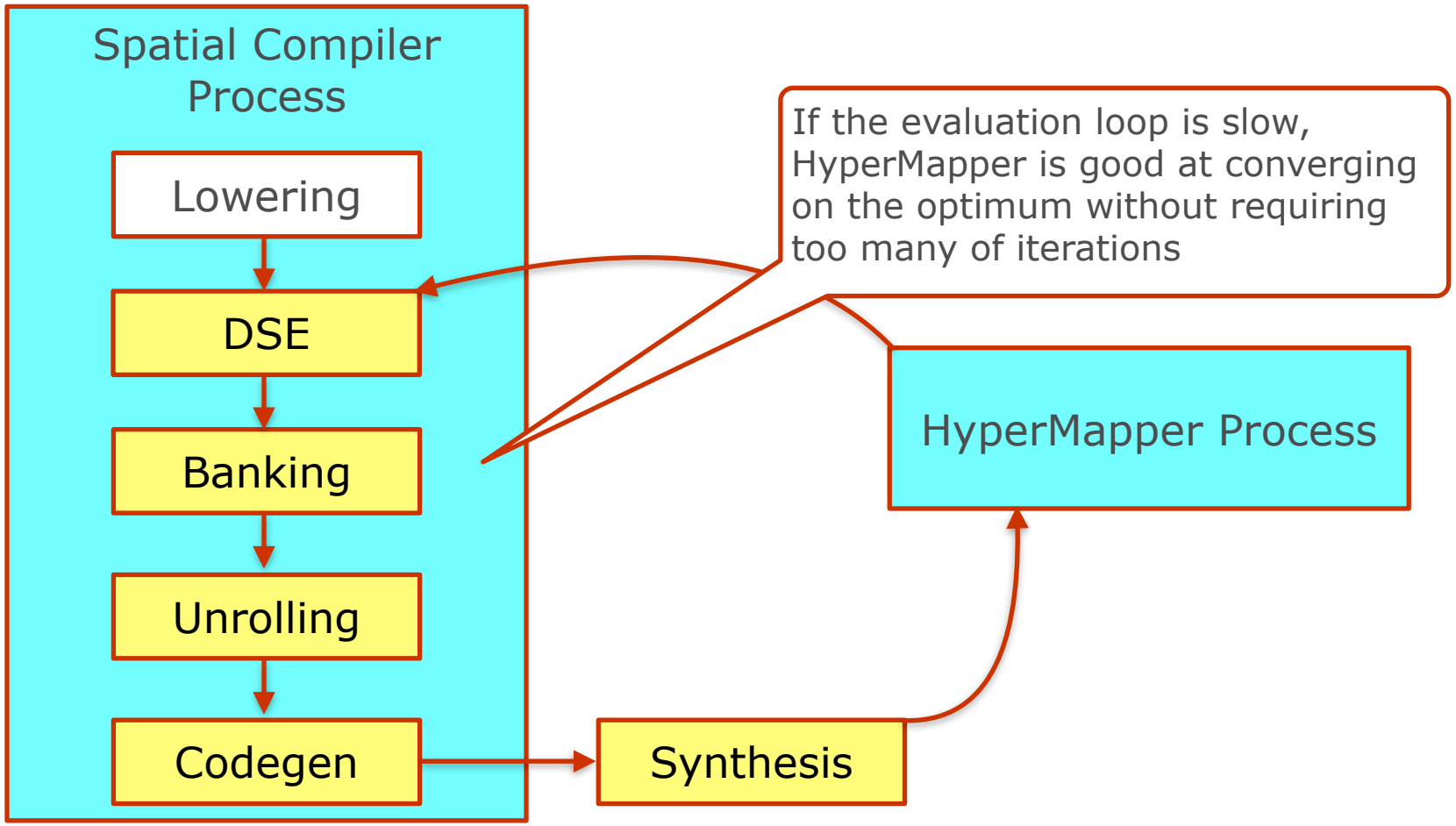
Spatial - Hypermapper Communication

- The design space can become huge!
- HyperMapper allows us to explore it intelligently.

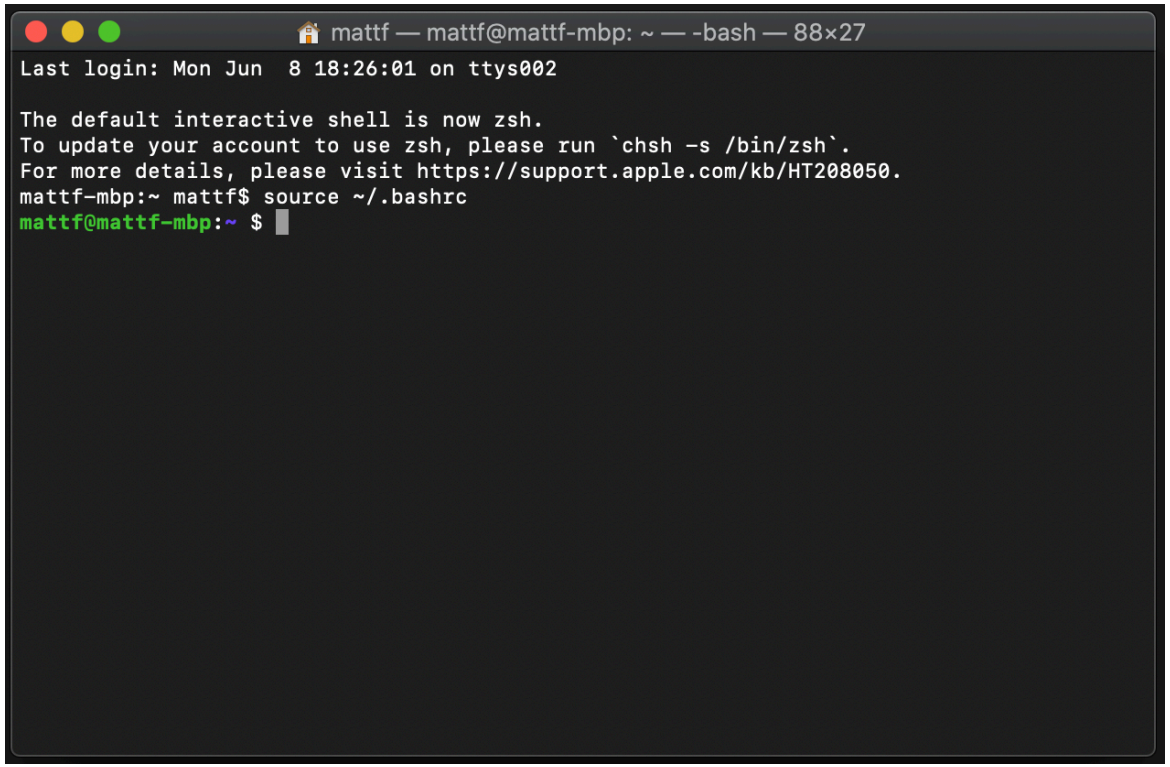


Spatial - Hypermapper Communication

- The design space can become huge!
- HyperMapper allows us to explore it intelligently.



Live Demo


mattf — mattf@mattf-mbp: ~ — -bash — 88×27
Last login: Mon Jun 8 18:26:01 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
mattf-mbp:~ mattf$ source ~/.bashrc
mattf@mattf-mbp:~ $