

Q How to Map a matrix onto multiple memory modules such that we can access various vectors in optimal time.

Mapping function is called skewing scheme.

eg. Consider a simple mapping, ϕ which maps a 4×4 matrix A onto 4 memory modules (M_0, M_1, M_2, M_3):
 $\phi: A[i, j] \rightarrow \{M_0, M_1, M_2, M_3\}$
 such that $\phi(i, j) = j \quad 0 \leq j \leq 3 \quad 0 \leq i \leq 3$

row # col # index of memory module

M-Matrix (mapping onto the memory module)

| M_0 | M_1 | M_2 | M_3 |
|----------|----------|----------|----------|
| A_{00} | A_{01} | A_{02} | A_{03} |
| A_{10} | A_{11} | A_{12} | A_{13} |
| A_{20} | A_{21} | A_{22} | A_{23} |
| A_{30} | A_{31} | A_{32} | A_{33} |

\leftrightarrow

Q-Matrix

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 0 | 1 | 2 | 3 |
| 0 | 1 | 2 | 3 |
| 0 | 1 | 2 | 3 |

Diagonal mapping

Row-wise mapping

Module #

Row Access Any row can be accessed in 1 cycle

Diagonal Access 1 Cycle

Column Access Due to conflict, it takes at most 4 cycles.

E.g. Consider

$$\phi(i, j) = (i + j + 1) \bmod 4$$

(We are mapping a 4×4 matrix onto 4 memory modules)

In general, $N \times N$ matrix mapped onto N modules.

| | | | |
|-------------|-------------|-------------|-------------|
| 1 | 2 | 3 | \emptyset |
| 2 | 3 | \emptyset | 1 |
| 3 | \emptyset | 1 | 2 |
| \emptyset | 1 | 2 | 3 |

Rows: Conflict free

ie. In each can be accessed in 1 cycle

Columns: Conflict free

Diagonals: Not conflict free, takes 2 cycles.

Observations

- ① There does not exist any linear ϕ function such that an $N \times N$ matrix can be mapped onto N memory modules such that all rows,

columns and diagonals can be accessed conflict free.

② Non-linear functions are no more powerful than linear functions.

Ex: (Prime memory module)

• Uses 5 memory modules for a 4x4 matrix.

$$\phi(i,j) = (i+j) \bmod 5 + 1 \quad 0 \leq i, j \leq 4$$

| M_0 | M_1 | M_2 | M_3 | M_4 | $i \downarrow j \rightarrow$ | ϕ -matrix | | | |
|----------|----------|----------|----------|----------|------------------------------|----------------|---|---|---|
| | A_{00} | A_{01} | A_{02} | A_{03} | 0 | 1 | 2 | 3 | 4 |
| A_{13} | | A_{10} | A_{11} | A_{12} | 1 | 2 | 3 | 4 | 0 |
| A_{22} | A_{23} | | A_{20} | A_{21} | 2 | 3 | 4 | 0 | 1 |
| A_{31} | A_{32} | A_{33} | | A_{30} | 3 | 4 | 0 | 1 | 2 |

Rows: Can be accessed conflict free.

Columns: Conflict free

Diagonal: Conflict free


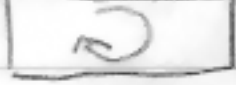
Drawbacks

① Modulo prime is very expensive whether at the compile stage or hardware stage.

② It has "holes", and hence less than 100% memory utilization.

③ In most machines the number of modules is a power of 2.

Multiskewing Scheme

Φ_1 :  $N \times N$ matrix onto N modules.
 Φ_2 : 

$\Phi_1: (i,j) = (j - 2i) \bmod N+1$ if $i \leq N/2$
 $\Phi_2: (i,j) = (2i - j - \frac{N}{2} - 1) \bmod N+1$, $i > N/2$

eg. Mapping an 8×8 matrix to 8 memory modules

Φ matrix

| | | | | | | | | |
|----------|---|---|---|----|----|----|----|----|
| | | → | | | | | | |
| | 8 | ① | 2 | 3 | 4 | 5 | 6 | 7 |
| Φ_1 | 6 | 7 | 8 | ①→ | 2 | 3 | 4 | 5 |
| | 4 | 5 | 6 | 7 | 8 | ①→ | 2 | 3 |
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ①→ |
| | 5 | 4 | 3 | 2 | ←① | 8 | 7 | 6 |
| Φ_2 | 7 | 6 | 5 | 4 | 3 | 2 | ←① | 8 |
| | 1 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| | 3 | 2 | ① | 8 | 7 | 6 | 5 | 4 |

Rows: Conflict free

Columns: Conflict free

Diagonal: Conflict free

- No prime arithmetic
- No of memory modules power of 2
- No holes