Workshop on Essential Abstractions in GCC

Introduction to Parallelization and Vectorization

GCC Resource Center
(www.cse.iitb.ac.in/grc)

Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay

July 2010

Notes

Outline

- Transformation for parallel and vector execution
- Data dependence

Notes

July 2010 Par-Vect Intro: Outline 1/1

Outline

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

The Scope of this Tutorial

- What this tutorial does not address
  - Algorithms used for parallelization and vectoriation
  - Machine level issues related to parallelization and vectoriation
- What this tutorial addresses

Basics of Discovering Parallelism using GCC

Notes

July 2010 Par-Vect Intro: Outline 2/1

The Scope of this Tutorial

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Part 1

Transformations for Parallel and Vector Execution

Notes

A Taxonomy of Parallel Computation

	Single Program	Multiple Programs
Single Data	SPSD	MPSD
Multiple Data	SPMD	MPMD

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 3/1

A Taxonomy of Parallel Computation

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 3/1

A Taxonomy of Parallel Computation

	Single Program		Multiple Programs
	Single Instruction	Multiple Instructions	
Single Data	SISD	MISD	MPSD
Multiple Data	SIMD	MIMD	MPMD

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 3/1

A Taxonomy of Parallel Computation

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

A Taxonomy of Parallel Computation

	Single Program		Multiple Programs
	Single Instruction	Multiple Instructions	
Single Data	SISD	?	?
Multiple Data	SIMD	MIMD	MPMD

Redundant computation for validation of intermediate steps

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 3/1

A Taxonomy of Parallel Computation

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 3/1

A Taxonomy of Parallel Computation

	Single Program		Multiple Programs
	Single Instruction	Multiple Instructions	
Single Data	SISD	MISD	MPSD
Multiple Data	SIMD	MIMD	MPMD

Transformations performed by a compiler

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 3/1

A Taxonomy of Parallel Computation

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Vectorization: SISD ⇒ SIMD

- Parallelism in executing operation on shorter operands (8-bit, 16-bit, 32-bit operands)
- Existing 32 or 64-bit arithmetic units used to perform multiple operations in parallel
  A 64 bit word ≡ a vector of 2×(32 bits), 4×(16 bits), or 8×(8 bits)

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 4/1

Vectorization: SISD ⇒ SIMD

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

Example 1

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

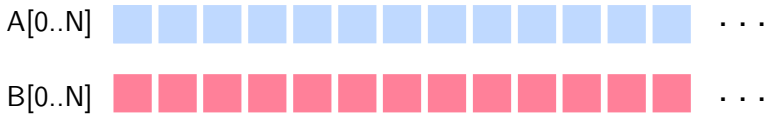
Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Observe reads and writes into a given location



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

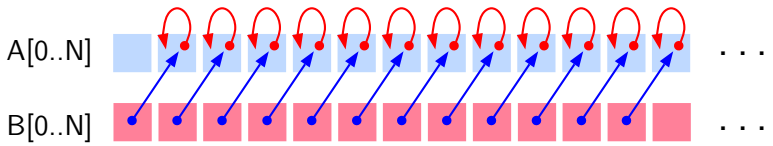
Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Observe reads and writes into a given location



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

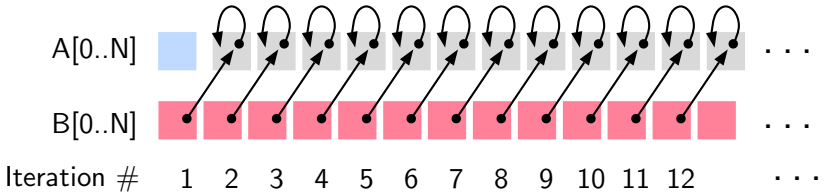
Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Observe reads and writes into a given location



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

Example 1

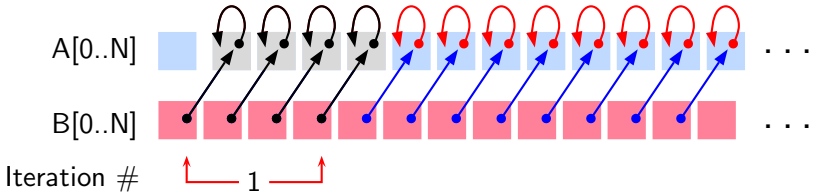
Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Vectorization Factor

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

```
int A[N], B[N], i;
for (i=1; i<N; i=i+4)
  A[i:i+3] = A[i:i+3] + B[i-1:i+2];
```



Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

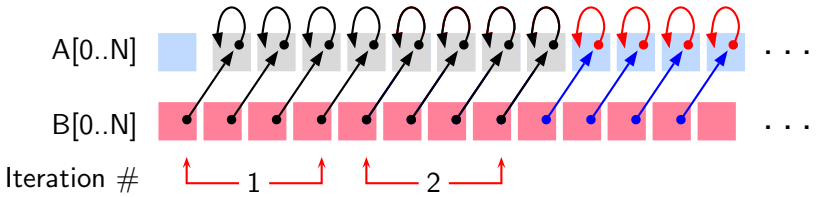
Vectorization Factor

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Vectorized Code

```
int A[N], B[N], i;
for (i=1; i<N; i=i+4)
  A[i:i+3] = A[i:i+3] + B[i-1:i+2];
```



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 5/1

Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

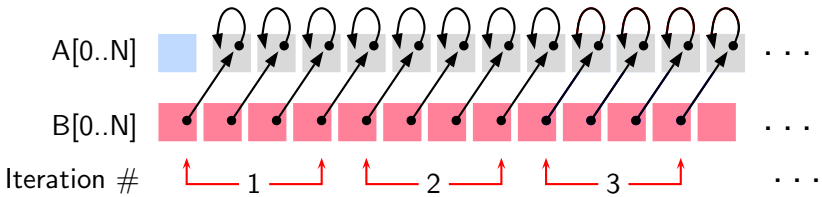
Vectorization Factor

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Vectorized Code

```
int A[N], B[N], i;
for (i=1; i<N; i=i+4)
  A[i:i+3] = A[i:i+3] + B[i-1:i+2];
```



Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Observe reads and writes into a given location

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 6/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 6/1

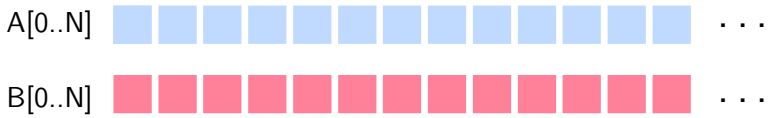
Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Observe reads and writes into a given location



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 6/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

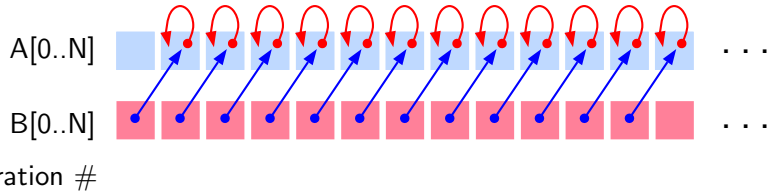
Example 1

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Observe reads and writes into a given location



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 6/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 6/1

Example 1

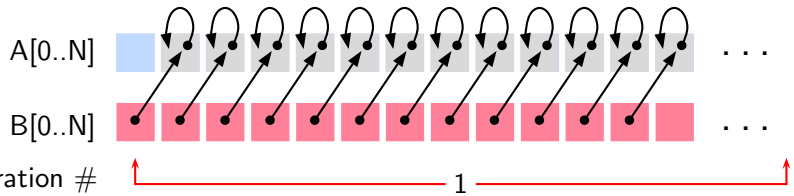
Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

Original Code

```
int A[N], B[N], i;
for (i=1; i<N; i++)
  A[i] = A[i] + B[i-1];
```

Parallelized Code

```
int A[N], B[N], i;
foreach (i=1; i<N; )
  A[i] = A[i] + B[i-1];
```



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 6/1

Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

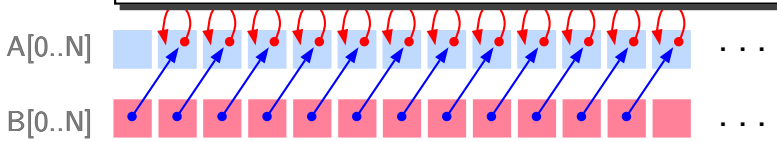
Example 1: The Moral of the Story

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : Yes

When the same location is accessed across different iterations, the order of reads and writes must be preserved

```
int A[
for (i
  A[i]
```

Nature of accesses in our example		
Iteration i	Iteration $i+k$	Observation
Read	Write	No
Write	Read	No
Write	Write	No
Read	Read	Does not matter



Example 1: The Moral of the Story

Notes

Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

Iteration #

Example 2

Notes

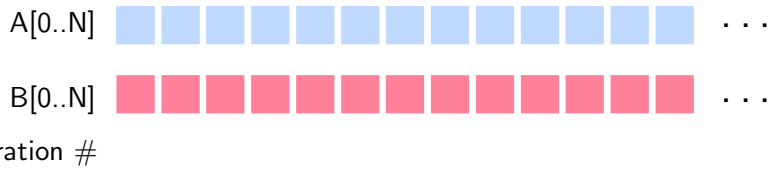
Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

Observe reads and writes into a given location



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 8/1

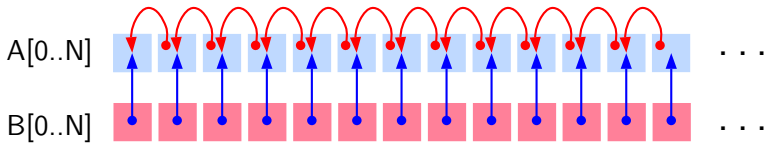
Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

Observe reads and writes into a given location



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

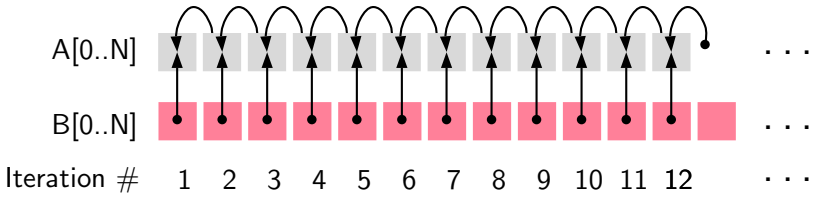
Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

Observe reads and writes into a given location



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 8/1

Example 2

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 8/1

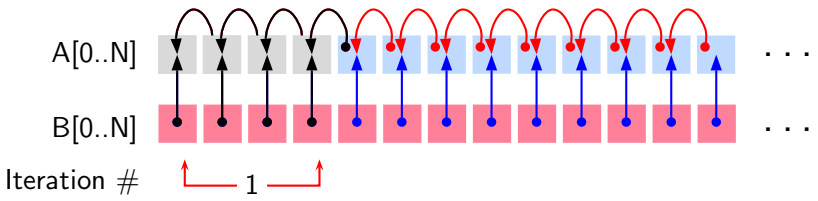
Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

- Vector instruction is synchronized: All reads before writes in a given instruction



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 8/1

Example 2

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

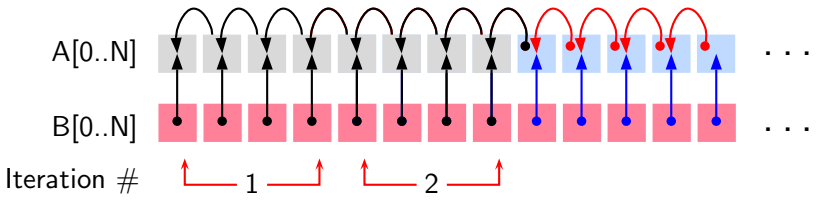
Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

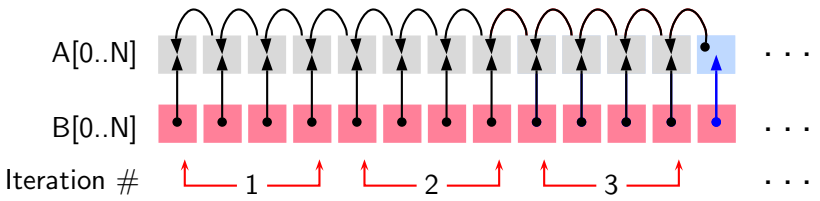
```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

- Vector instruction is synchronized: All reads before writes in a given instruction



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 8/1
Example 2
Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No
Original Code
int A[N], B[N], i;
for (i=0; i<N; i++)
A[i] = A[i+1] + B[i];
• Vector instruction is synchronized: All reads before writes in a given instruction



Essential Abstractions in GCC GCC Resource Center, IIT Bombay
Notes

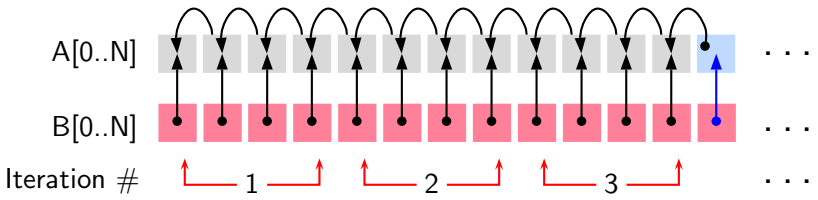
Example 2

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : No

Original Code

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i] = A[i+1] + B[i];
```

- Vector instruction is synchronized: All reads before writes in a given instruction
- Read-writes across multiple instructions executing in parallel may not be synchronized



July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 8/1

Example 2

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

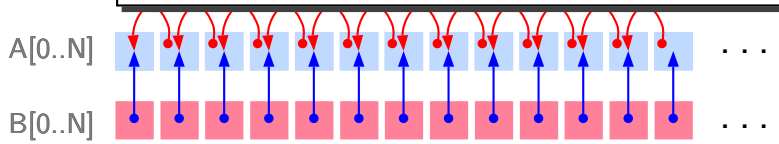
July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 9/1

Example 2: The Moral of the Story

Vectorization (SISD ⇒ SIMD) : Yes
Parallelization (SISD ⇒ MIMD) : **No**

When the same location is accessed across different iterations, the order of reads and writes must be preserved

Nature of accesses in our example		
Iteration i	Iteration $i+k$	Observation
Read	Write	Yes
Write	Read	No
Write	Write	No
Read	Read	Does not matter



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 9/1

Example 2: The Moral of the Story

Notes

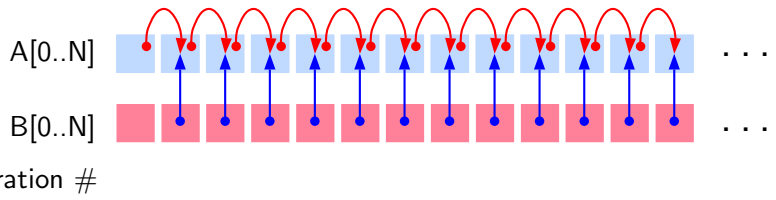
Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 3

Vectorization (SISD ⇒ SIMD) : No
Parallelization (SISD ⇒ MIMD) : No

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i+1] = A[i] + B[i+1];
```

Observe reads and writes into a given location



Notes

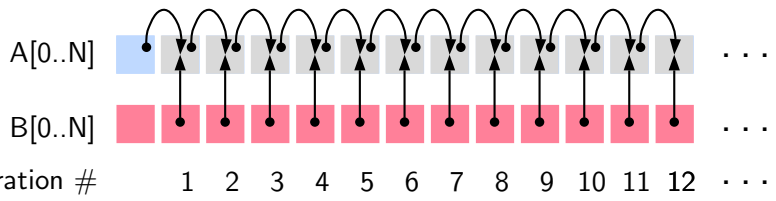
July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 10/1

Example 3

Vectorization (SISD ⇒ SIMD) : No
Parallelization (SISD ⇒ MIMD) : No

```
int A[N], B[N], i;
for (i=0; i<N; i++)
  A[i+1] = A[i] + B[i+1];
```

Observe reads and writes into a given location



Essential Abstractions in GCC GCC Resource Center, IIT Bombay

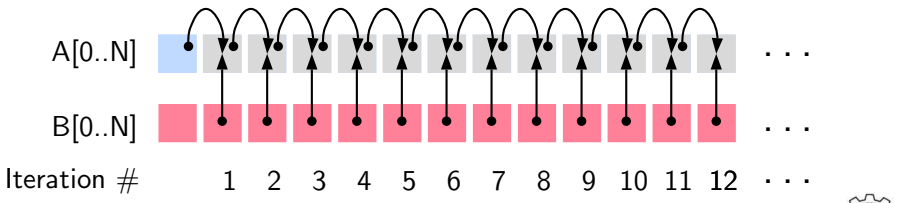
Notes

Example 3

Vectorization (SISD ⇒ SIMD) : **No**
Parallelization (SISD ⇒ MIMD) : **No**

```
int A[N], B
for (i=0; i
  A[i+1] =
```

Nature of accesses in our example		
Iteration i	Iteration $i+k$	Observation
Read	Write	No
Write	Read	Yes
Write	Write	No
Read	Read	Does not matter



July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 10/1

Example 3

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 11/1

Example 4

Vectorization (SISD ⇒ SIMD) : No
Parallelization (SISD ⇒ MIMD) : Yes

- This case is not possible
- Vectorization is a limited granularity parallelization
- If parallelization is possible then vectorization is trivially possible
- Blank line to increase height
- Blank line to increase height

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 11/1

Example 4

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Data Dependence

Let statements S_i and S_j access memory location m at time instants t and $t + k$

Access in S_i	Access in S_j	Dependence	Notation
Read m	Write m	Anti (or Pseudo)	$S_i \bar{\delta} S_j$
Write m	Read m	Flow (or True)	$S_i \delta S_j$
Write m	Write m	Output (or Pseudo)	$S_i \delta^O S_j$
Read m	Read m	Does not matter	

- Pseudo dependences may be eliminated by some transformations
- True dependence prohibits parallel execution of S_i and S_j

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 12/1

Data Dependence

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Loop Carried and Loop Independent Dependences

Consider dependence between statements S_i and S_j in a loop

- Loop independent dependence. t and $t + k$ occur in the same iteration of a loop
  - S_i and S_j must be executed sequentially
  - Different iterations of the loop can be parallelized
- Loop carried dependence. t and $t + k$ occur in the different iterations of a loop
  - Within an iteration, S_i and S_j can be executed in parallel
  - Different iterations of the loop must be executed sequentially
- S_i and S_j may have both loop carried and loop independent dependences

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 13/1

Loop Carried and Loop Independent Dependences

Notes

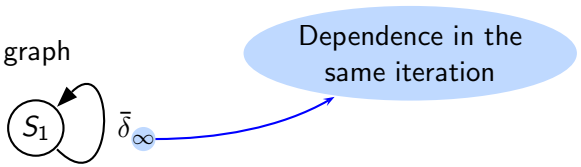
Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Dependence in Example 1

- Program

```
int A[N], B[N], i;
for (i=1; i<N; i++)
    A[i] = A[i] + B[i-1];    /* S1 */
```

- Dependence graph



- No loop carried dependence
  Both vectorization and parallelization are possible

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 14/1

Dependence in Example 1

Notes

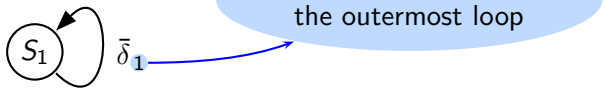
Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Dependence in Example 2

- Program

```
int A[N], B[N], i;
  for (i=0; i<N; i++)
     A[i] = A[i+1] + B[i];  /* S1 */
```

- Dependence graph



- Loop carried anti-dependence
  Parallelization is not possible
  Vectorization is possible since all reads are done before all writes

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 15/1

Dependence in Example 2

Notes

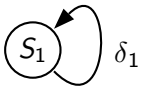
Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Dependence in Example 3

- Program

```
int A[N], B[N], i;
  for (i=0; i<N; i++)
     A[i+1] = A[i] + B[i+1];  /* S1 */
```

- Dependence graph



- Loop carried flow-dependence
  Neither parallelization not vectorization is possible

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 16/1

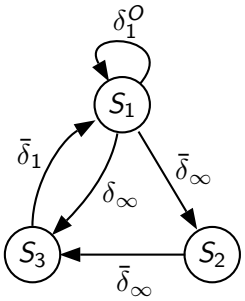
Dependence in Example 3

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

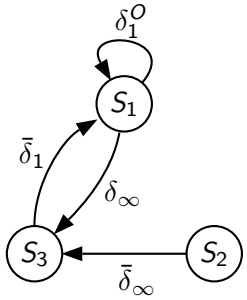
Example 4: Dependence

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Loop independent anti dependence due to `A[i]`

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

Notes

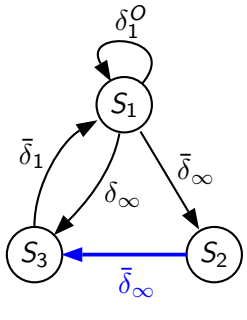
Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Loop independent anti dependence due to `B[i]`

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

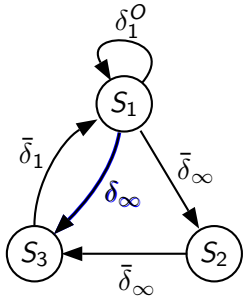
Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Loop independent flow dependence due to T

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

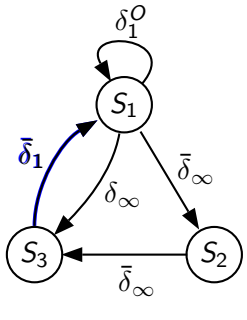
Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Loop carried anti dependence due to T

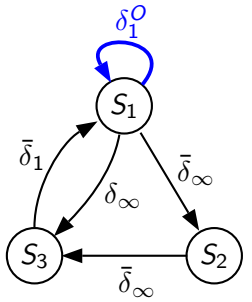
Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Notes

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Loop carried output dependence due to T

Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

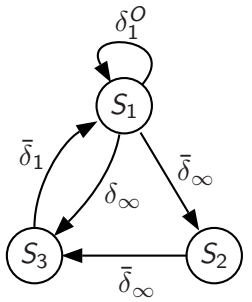
Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

Program to swap arrays	Dependence Graph

```
for (i=0; i<N; i++)
{   T = A[i];        /* S1 */
    A[i] = B[i];     /* S2 */
    B[i] = T;        /* S3 */
}
```



Notes

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 17/1

Example 4: Dependence

Notes

Tutorial Problem for Discovering Dependence

Draw the dependence graph for the following program
(Earlier program modified to swap 2-dimensional arrays)

```
for (i=0; i<N; i++)
{
    for (j=0; j<N; j++)
    {   T = A[i][j];        /* S1 */
        A[i][j] = B[i][j]; /* S2 */
        B[i][j] = T;        /* S3 */
    }
}
```

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 18/1

Tutorial Problem for Discovering Dependence

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Data Dependence in Loops

- Analysis in loop is tricky, as
  - Loops may be nested
  - Different loop iterations may access same memory location
  - Arrays occur frequently
  - Far too many array locations to be treated as independent scalars
- Consider array location `A[4][9]` in the following program

```
for(i = 0; i <= 5; i ++)
  for(j = 0; j <= 4; j ++)
  {
    A[i+1][3*j] = ... ;          /* S1 */
          ...   = A[i+3][2*j+1]; /* S2 */
  }
```

S2 accesses in iteration (1,4), S1 accesses in iteration (3,3)

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 19/1

Data Dependence in Loops

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Iteration Vectors and Index Vectors: Example 1

```
for (i=0, i<4; i++)
  for (j=0; j<4; j++)
  {
     a[i+1][j] = a[i][j] + 2;
  }
```

Loop carried dependence exists if

- there are two distinct iteration vectors such that
- the index vectors of LHS and RHS are identical

Conclusion: Dependence exists

Iteration Vector	Index Vector	
	LHS	RHS
0,0	1,0	0,0
0,1	1,1	0,1
0,2	1,2	0,2
0,3	1,3	0,3
1,0	2,0	1,0
1,1	2,1	1,1
1,2	2,2	1,2
1,3	2,3	1,3
2,0	3,0	2,0
2,1	3,1	2,1
2,2	3,2	2,2
2,3	3,3	2,3
3,0	4,0	3,0
3,1	4,1	3,1
3,2	4,2	3,2
3,3	4,3	3,3

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 20/1

Iteration Vectors and Index Vectors: Example 1

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Iteration Vectors and Index Vectors: Example 2

```
for (i=0, i<4; i++)
  for (j=0; j<4; j++)
  {
     a[i][j] = a[i][j] + 2;
  }
```

Loop carried dependence exists if

- there are two distinct iteration vectors such that
- the index vectors of LHS and RHS are identical

Conclusion: No dependence

Iteration Vector	Index Vector	
	LHS	RHS
0,0	0,0	0,0
0,1	0,1	0,1
0,2	0,2	0,2
0,3	0,3	0,3
1,0	1,0	1,0
1,1	1,1	1,1
1,2	1,2	1,2
1,3	1,3	1,3
2,0	2,0	2,0
2,1	2,1	2,1
2,2	2,2	2,2
2,3	2,3	2,3
3,0	3,0	3,0
3,1	3,1	3,1
3,2	3,2	3,2
3,3	3,3	3,3

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 21/1

Iteration Vectors and Index Vectors: Example 2

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay

Data Dependence Theorem

There exists a dependence from statement S_1 to statement S_2 in common nest of loops if and only if there exist two iteration vectors **i** and **j** for the nest, such that

1. $\mathbf{i} < \mathbf{j}$ or $\mathbf{i} = \mathbf{j}$ and there exists a path from S_1 to S_2 in the body of the loop,
2. statement S_1 accesses memory location M on iteration **i** and statement S_2 accesses location M on iteration **j**, and
3. one of these accesses is a write access.

July 2010 Par-Vect Intro: Transformations for Parallel and Vector Execution 22/1

Data Dependence Theorem

Notes

Essential Abstractions in GCC GCC Resource Center, IIT Bombay