Inter In-place Storage Size Requirement Estimation

Pål Rydland
NTNU, Trondheim, Norway

Martin Palkovic
IMEC, Leuven, Belgium

Per Gunnar Kjeldsberg,
NTNU, Trondheim, Norway

Erik Brockmeyer
IMEC, Leuven, Belgium

Francky Catthoor,
IMEC/KU Leuven, Belgium

Abstract

Data storage is an important contributor to power dissipation, particularly in multi-media embedded systems. To achieve low power implementation for these systems, the storage requirement can be reduced by exploiting limited data lifetimes and reuse of memory locations. This poster presents techniques for storage requirement estimation. It reuses established techniques for estimation of individual dependencies, but adds a global perspective via a grouping algorithm. A prototype CAD tool has been developed and used to test the technique on real-life multi-media kernel.

Motivation

- Estimate memory requirements early in the design flow.
- Embedded applications with deep loop-nests and multi-dimensional arrays.
- Data dominated multi-media and signal-processing applications.

Starting Point

- Static analysis of single assignment and manifest code.
- Single thread of control.

```
for (int j = 0;i<=9;i++)
  for (int i=0;j<=9;j++) {
    if ( 1 <= i <= 4 ) && ( 2 <= j <= 6 )
      A[i][j] = a( input );
    if ( 3 <= i <= 6 ) && ( 4 <= j <= 8 )
      B[i][j] = b( A[i-2][j-2] );
  }
```

Figure 1: Application program code

Polytope Model

- Iteration space with polytopes corresponding to production and consumption of data.

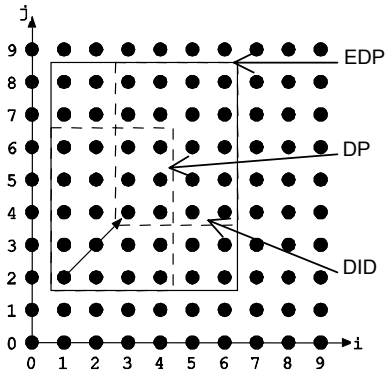

Figure 2: Data dependencies between executions of statements

Polytope Model Concepts:

- Dependency Part of Iteration Domain
- Extended Dependency Part
- Depending Iteration Domain
- Domain Dependency

Global Storage Requirements

- Generate common iteration space

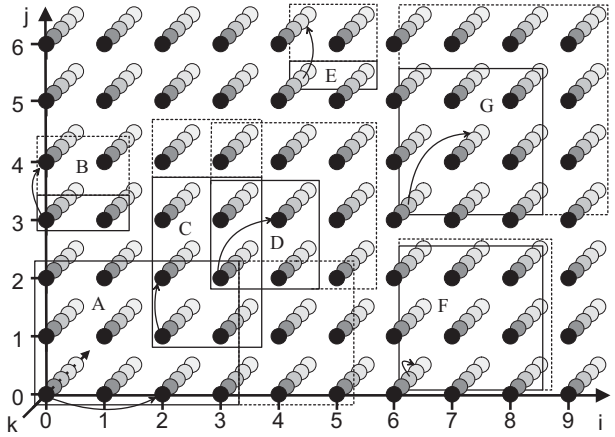

Figure 3: Common iteration space, containing all EDPs

- Group simultaneously alive EDPs.
- Storage requirements = maximum group size.

```
max_size = 0
groups = Group simultaneously alive domain dependencies
for each group in groups:
  size = 0
  for each domain dependency in group:
    size += Identify maximum domain dependency size
  max_size = max(max_size,size}
```

Figure 4: Estimation methodology

Grouping Algorithm:

- Inspects the iteration space recursively.
- Starts at the outermost dimension.
- Assume iteration order [j , i , k]:

Dim.	Overlapping EDPs	Identified groups
j	ACDF, BCDG,GE	CD{AF}, BCDG, GE
i	A,F	CDA, CDF, BCDG, GE

Storage Requirement of Individual EDPs

- Existing techniques used for Individual EDPs storage requirement estimation.
- Example in Figure 5 assumes iteration order [j , i , k]

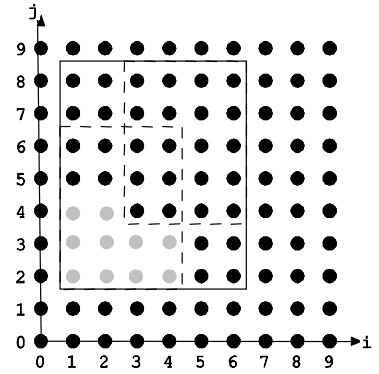

Figure 5: Storage requirement of EDP from Figure 2

- Estimated storage requirement = 10

Results

- Prototype tool implemented in Python.
- Tested on QSDPCM application.
- No serious regression discovered.
- Tested on iteration space in Figure 3, with Memory Compaction Tool (developed at IMEC) used as reference.
- Assume iteration order [j , i , k]:

Group	MC	PySTOREQ
CDA	32	29
CDF	29	26
CDBG	43	40
GE	14	13

Presented at IEEE NORCHIP, Riga, Latvia, November 2003

Imec, Leuven, Belgium

Norwegian University of Science and Technology, Trondheim, Norway