Search Space Properties for Pipelined FPGA Applications

University of Southern California
Information Sciences Institute

Heidi Ziegler, Mary Hall, Byoungro So
Oct 2, 2003
Mapping Assignment

Machine Vision Kernel (application requirements)

1. Edge detection
2. Feature extraction
3. Distance computation

Map

FPGA
Mapping an Application to Hardware

Machine Vision Kernel (MVIS)
1. Edge detection
2. Feature extraction
3. Distance computation

Compute Data Layout

Partition Chip Capacity

Manage Communication

on-chip storage

data path

configurable
interconnect
Build on Prior Work in DEFACTO

- Automatic design space exploration for individual loop nests (DAC03, PLDI02)
- Analyses and transformations to exploit ILP (PLDI02) and maximize memory bandwidth (LPCP02)
- Communication and pipeline analysis to exploit data and task parallelism (FCCM02, DAC03)
This Research

- Integrates communication and pipelining analysis with the single loop design space exploration

- Defines and illustrates search space properties for the global optimization problem

- Describes a search algorithm and presents a case study
Sequential MVIS Kernel

Execution Order

- Pipeline Stage S1
- Pipeline Stage S2
- Pipeline Stage S3

Write

- B
- D
- F

Read

- A
- B
- D

Time

Data dependence

Access order row-wise

2-D array
Reaching Definition Data Access Descriptor

\[ RDAD_{r, w}, s(A) = \langle \alpha|\tau|\delta|\omega|\gamma \rangle \]

Set describes **basic data access information**

- \( s \) program point
- \( r, w \) read or write array access
- \( \alpha \) accessed array section, integer linear inequalities
- \( \tau \) traversal order, vector of dims., slowest to fastest
- \( \delta \) vector of dominant induction variables for ea. dim
- \( \omega \) set of statements this tuple describes (def or use)
- \( \gamma \) set of reaching definitions
Communication Requirements

Solve directly for data, granularity, placement

\[ \text{Write (3)} \quad \text{Read (4)} \]

Stage S1

\[ f \left( RDAD_{w,s1}(B), \quad RDAD_{r,s2}(B) \right) \]

\[ RDAD_{w,s1}(B) = \left\langle 0 \leq d1 \leq 29, 0 \leq d2 \leq 29 \left| \langle 1,2 \rangle \langle x, y \rangle \{3\} \mid \emptyset \right. \right\rangle \]

\[ RDAD_{r,s2}(B) = \left\langle 0 \leq d1 \leq 29, 0 \leq d2 \leq 29 \left| \langle 1,2 \rangle \langle x, y \rangle \{4\} \mid \{3\} \right. \right\rangle \]
Task Graph

- Nodes are pipeline stages
- Communication edge descriptors (CEDs) computed from RDADs

\[ \text{CED}_{s_i \rightarrow s_j}(A) = \langle \alpha | \lambda | \rho \rangle \]

- \( \alpha \) array section, per communication instance
- \( \lambda \) send point
- \( \rho \) receive point
Global Optimization Strategy

- 2 Criteria
  - Design’s execution time should be minimized
  - Design’s space utilization, for a given level of performance, should be minimized

- Estimates
  - Behavioral synthesis area (all loops)
  - Behavioral synthesis timing (all loops)
  - Communication rates
Transformations

- Local
  - Unroll and jam
  - Scalar replacement
  - Custom data layout

- Global
  - Communication granularity and placement
  - Producer-Consumer Rate Matching
  - Data reorganization on-chip
High-Level Design Flow

Basic Compiler Optimizations
Scalar Replacement
Communication and Pipeline Analysis
Custom Data Layout
SUIF to VHDL
Behavioral Synthesis and Estimation

Good Design?
Yes
No
Unroll and Jam
Producer-Consumer Rate Matching
Communication Granularity Analysis

Logic Synthesis / Place & Route
Configuration Bit Stream
Observation 1: Non-increasing Memory Accesses

- Choose to place communication on-chip
Observation 2: Non-increasing Unroll Factor

- Local solution assumed to be best-case performance, worst-case space estimate

Single Loop Solution

Global Solution
Observation 3: Matching Rates without Affecting Performance

- Avoid creating longer critical paths

If \( \text{rate}_{\text{prod}}(d) < \text{rate}_{\text{cons}}(d) \), we can safely reduce the unroll factor for S3 until the rates match.
Optimization Algorithm: Step 1

Apply Pipeline and Communication Analysis

```c
for (x=0; x<image-2; x++) {
    for (y=0; y<image-2; y++) {
        uh1 = -3*u[x][y] - 3*u[x+1][y] ......;
        uh2 = -3*u[x][y] + 3*u[x+1][y] ......;
        peak[x][y] = uh1 + uh2;
    }
}

for (x=0; x<image-2; x++) {
    for (y=0; y<image-2; y++) {
        if (peak[x][y] > threshold)
            feature_x[x][y] = x;
        else feature_x[x][y] = 0;
    }
}

for (x=0; x<image-2; x++) {
    for (y=0; y<image-2; y++) {
        if (feature_x[x][y] != 0)
            ssd[x][y] = (u[x][y] - v[x][y+1])^2 ..........
    }
}
```
Optimization Algorithm: Step 2

Find Single Loop Solutions in Isolation

```c
for (x=0; x<image-2; x++) {
    for (y=0; y<image-2; y++) {
        uh1 = -3*u[x][y] - 3*u[x+1][y]......;
        uh2 = -3*u[x][y] + 3*u[x+1][y] ......
        peak[x][y] = uh1 + uh2;
    }
}
for (x=0; x<image-2; x++) {
    for (y=0; y<image-2; y++) {
        if (peak[x][y] > threshold)
            feature_x[x][y] = x;
        else feature_x[x][y] = 0;
    }
}
for (x=0; x<image-2; x++) {
    for (y=0; y<image-2; y++) {
        if (feature_x[x][y] != 0)
            ssd[x][y] = (u[x][y]-v[x][y+1])^2 ...........
    }
}
Optimization Algorithm: Step 3

Match Producer and Consumer Rates

rate_{prod}(peak) = rate_{cons}(peak)
rate_{prod}(feature_x) = rate_{cons}(feature_x)
Optimization Algorithm: Step 4

Apply Greedy Strategy to Meet Chip Constraint

\[ \text{capacity} \geq \sum_{1}^{n} \text{area}_i \]

If not, apply greedy strategy and then repeat steps 3 and 4.

Final Solution
Related Work

- Synthesizing high-level constructs
  - Handel-C, RaPiD, PipeRench, Babb et al.

- Design space exploration
  - Derrien/Rajopadhye, Cameron, PICO

- Program analysis on arrays
  - Hall et al, Amarasinghe, Balasundaram & Kennedy

- Pipeline analysis
  - Splash 2, Weinhardt & Luk, Du et al, Goldstein et al.
Conclusion

- System-level compiler automatically derives a pipelined implementation with explicit communication, while partitioning the chip capacity among pipeline stages

- Global optimization strategy
  - Built upon local solution with communication

- Constrain the search space
  - Non-increasing memory accesses
  - Non-increasing unroll factors
Contact Information

Project Web Site

www.isi.edu/asd/defacto

Authors’ email addresses

ziegler, mhall, bso@isi.edu