Search Space Properties for Pipelined FPGA Applications

University of Southern California
Information Sciences Institute

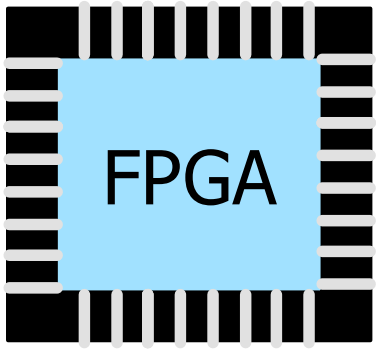
Heidi Ziegler, Mary Hall, Byoungro So

Oct 2, 2003

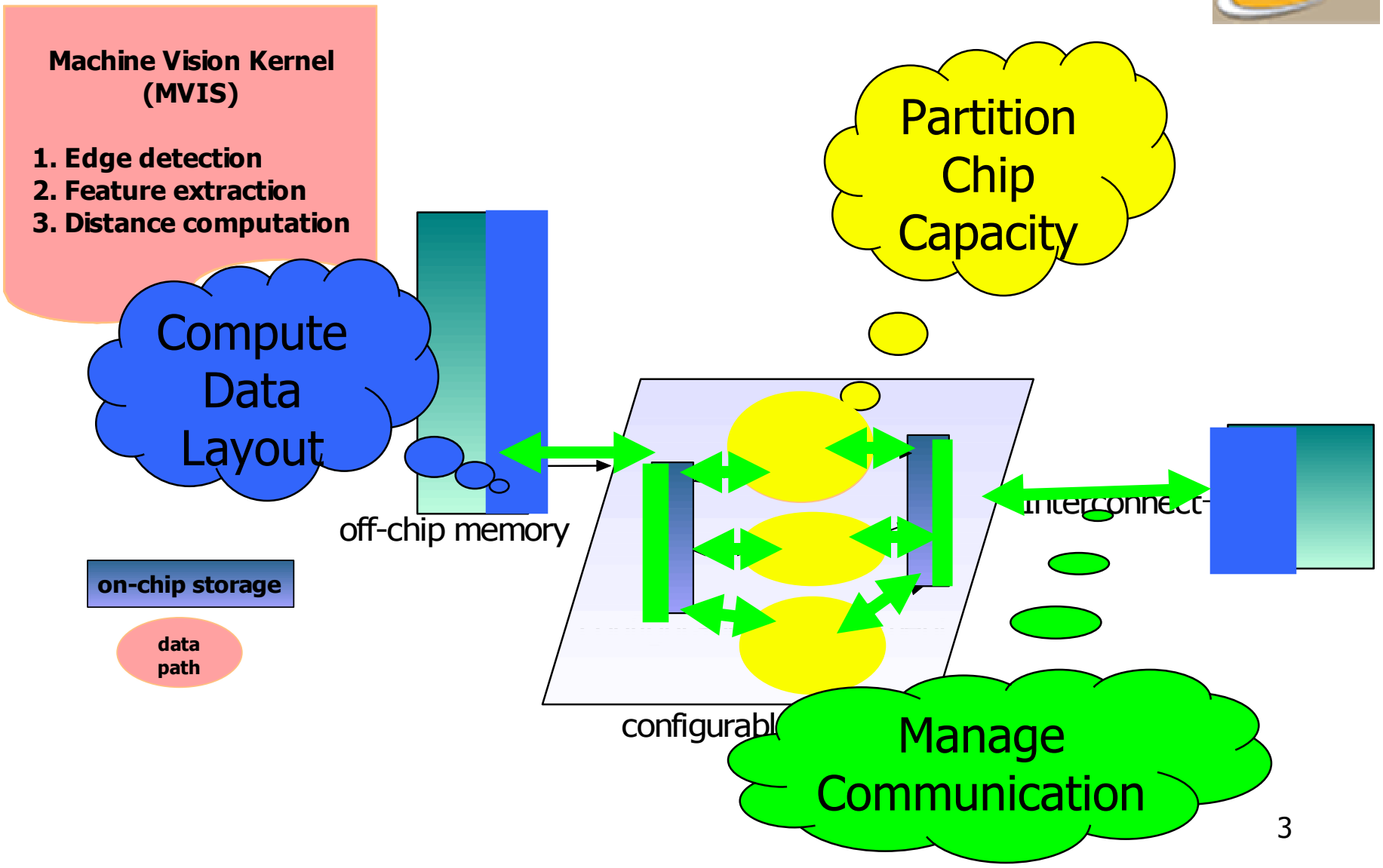
Mapping Assignment

Machine Vision Kernel (application requirements)

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

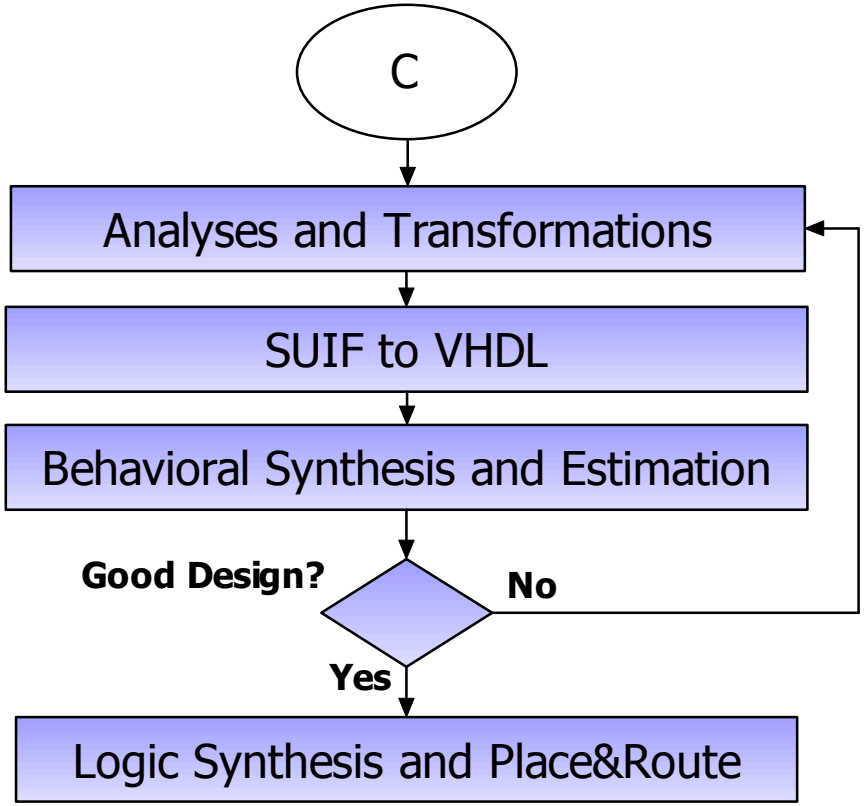


Mapping an Application to Hardware



Build on Prior Work in DEFACTO

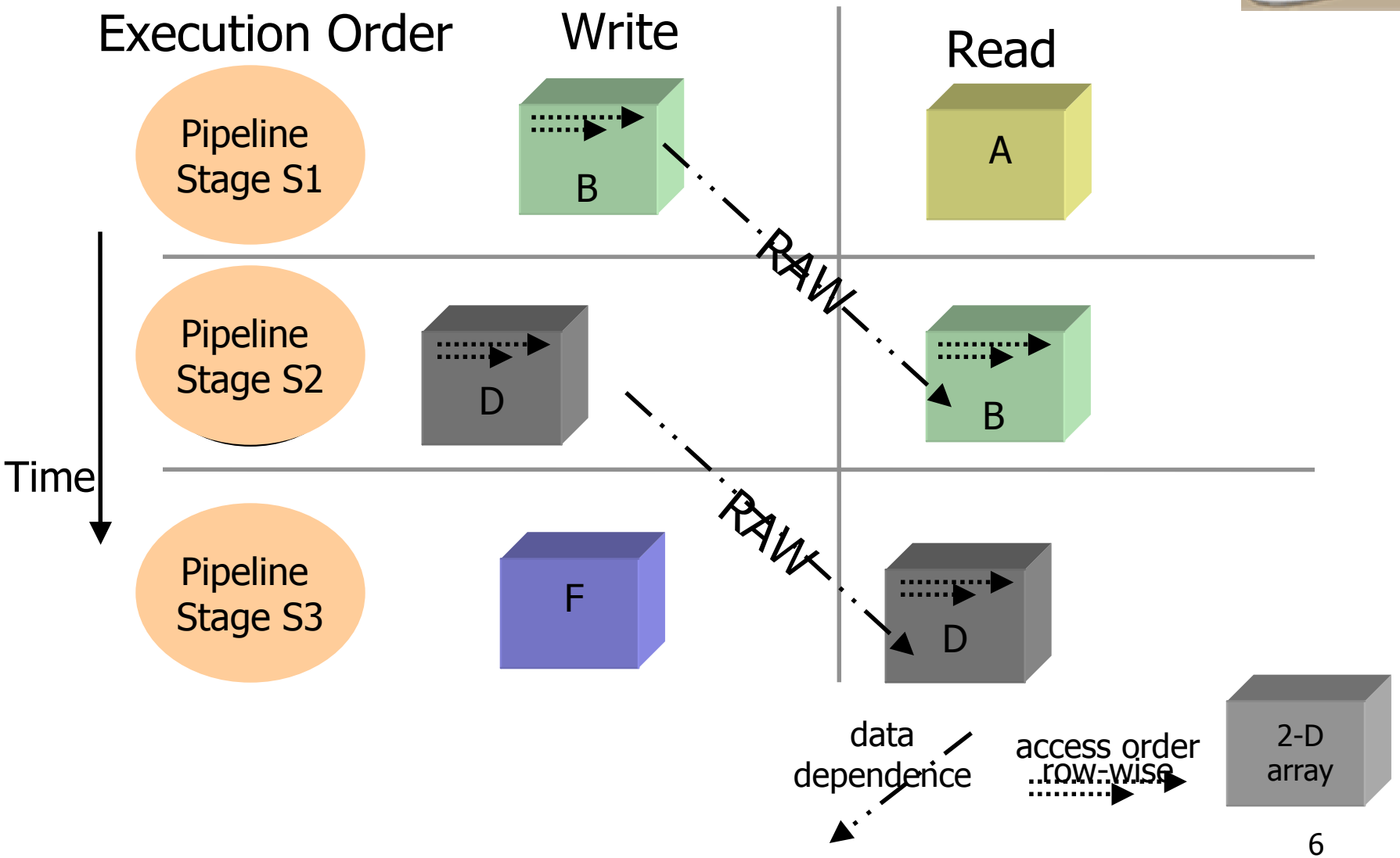
- Automatic design space exploration for individual loop nests (DAC03, PLDI02)
- Analyses and transformations to exploit ILP (PLDI02) and maximize memory bandwidth (LPCP02)
- Communication and pipeline analysis to exploit data and task parallelism (FCCM02, DAC03)



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



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

α accessed array section, integer linear inequalities

τ traversal order, vector of dims., slowest to fastest

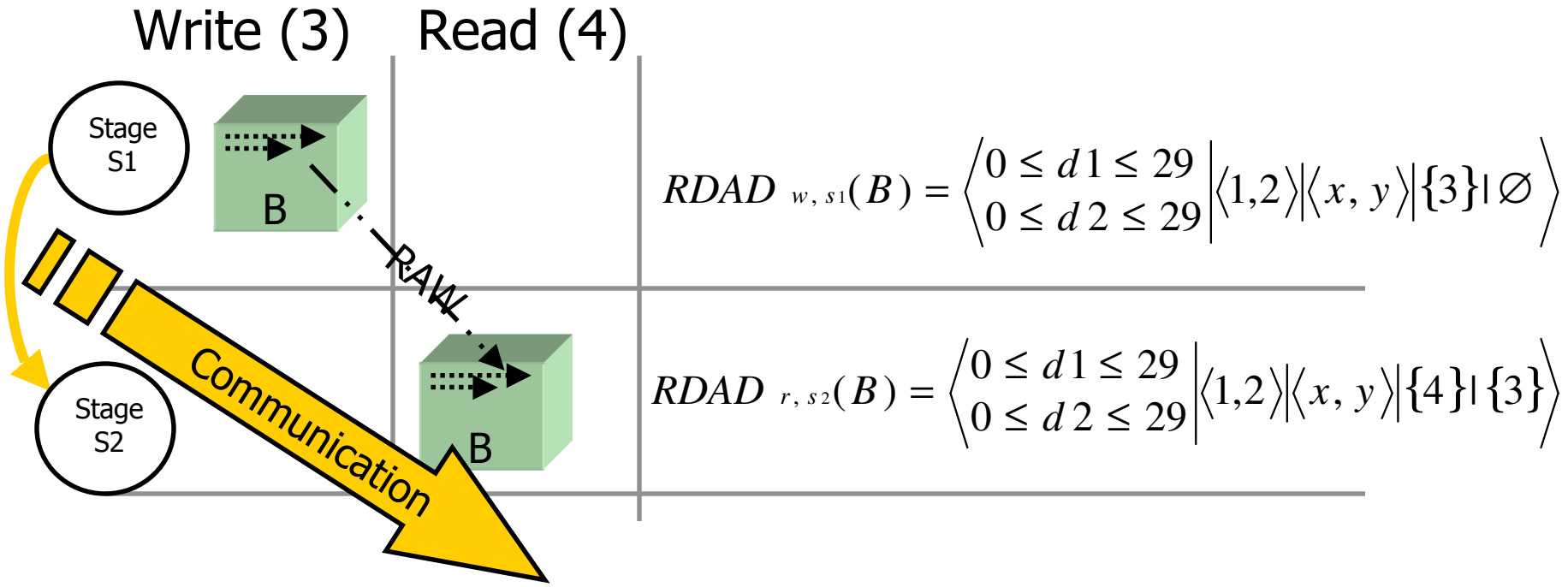
δ vector of dominant induction variables for ea. dim

ω set of statements this tuple describes (def or use)

γ set of reaching definitions

Communication Requirements

Solve directly for data, granularity, placement



$$RDAD_{w,s1}(B) = \left\langle \begin{matrix} 0 \le d1 \le 29 \\ 0 \le d2 \le 29 \end{matrix} \middle| \langle 1,2 \rangle \middle| \langle x, y \rangle \middle| \{3\} \middle| \varnothing \right\rangle$$

$$RDAD_{r,s2}(B) = \left\langle \begin{matrix} 0 \le d1 \le 29 \\ 0 \le d2 \le 29 \end{matrix} \middle| \langle 1,2 \rangle \middle| \langle x, y \rangle \middle| \{4\} \middle| \{3\} \right\rangle$$

$$f\left(RDAD_{w,si}(B), \quad RDAD_{r,sj}(B) \right)$$

Task Graph

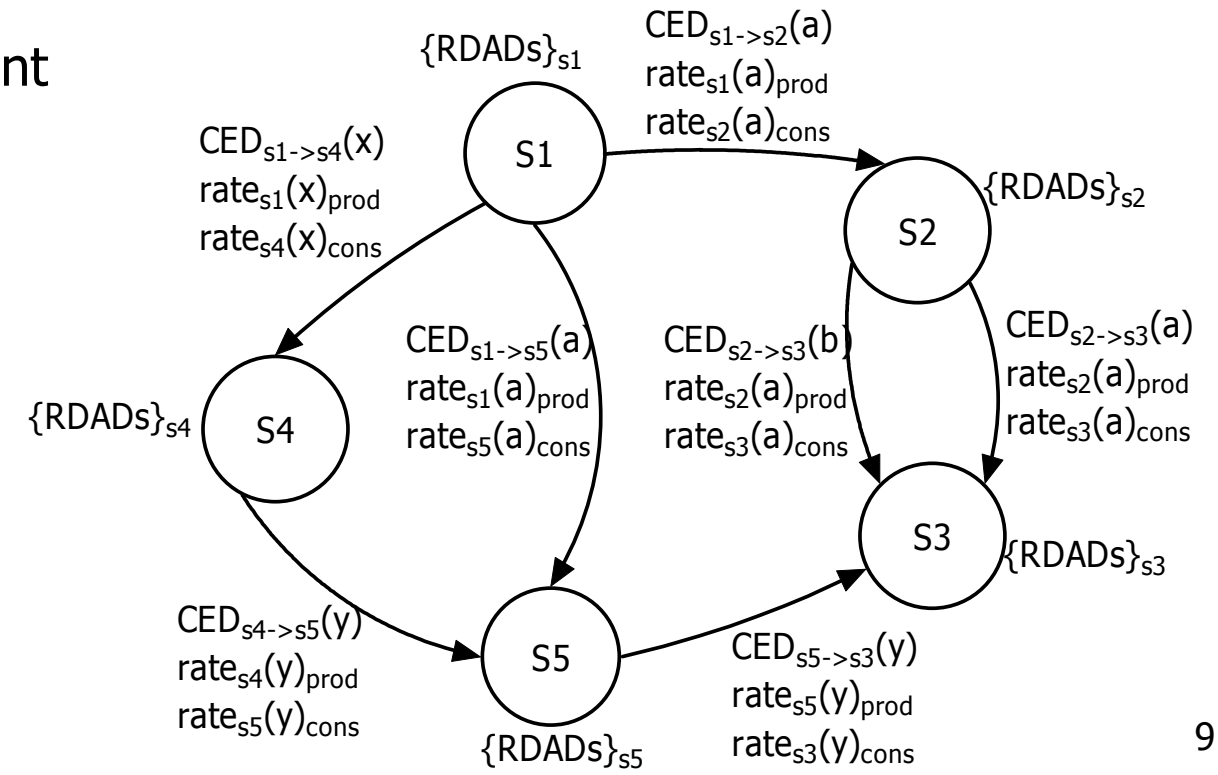
- Nodes are pipeline stages
- Communication edge descriptors (CEDs) computed from RDADs

$$CED_{si \rightarrow sj}(A) = \langle \alpha | \lambda | \rho \rangle$$

α array section, per communication instance

λ send point

ρ receive point



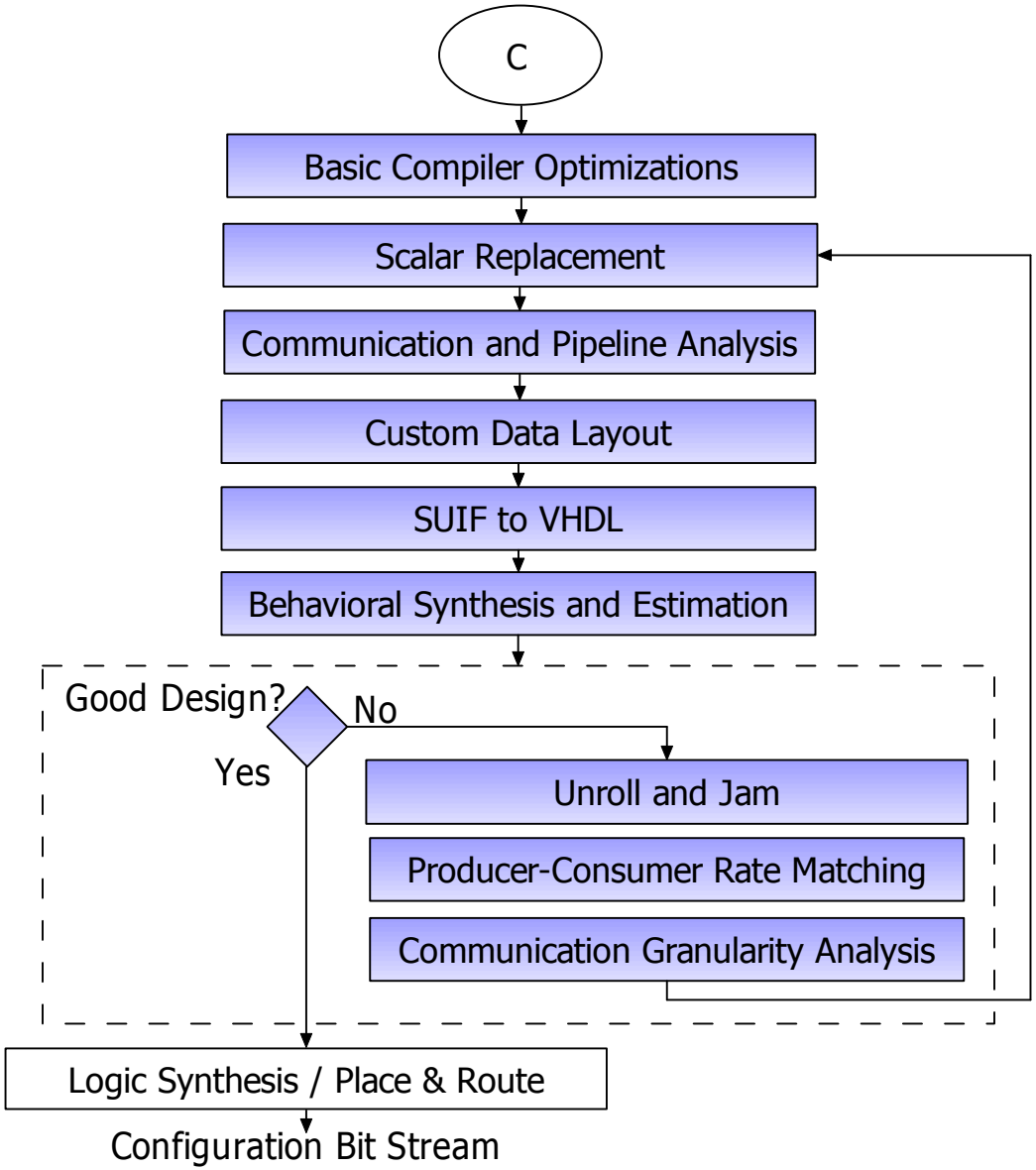
Global Optimization Strategy

- 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

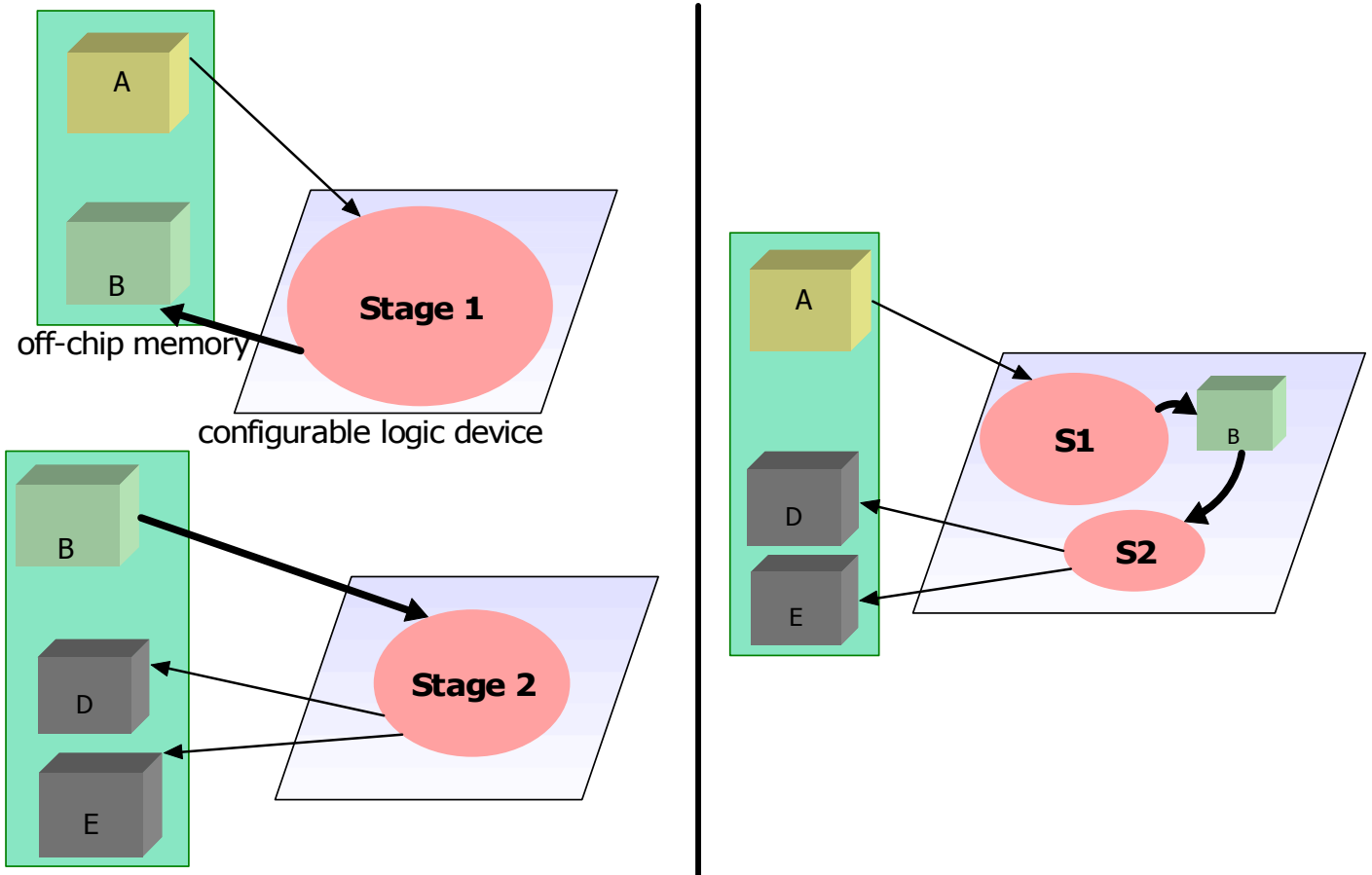
- Local
  - Unroll and jam
  - Scalar replacement
  - Custom data layout
- Global
  - Communication granularity and placement
  - Producer-Consumer Rate Matching
  - Data reorganization on-chip

High-Level Design Flow



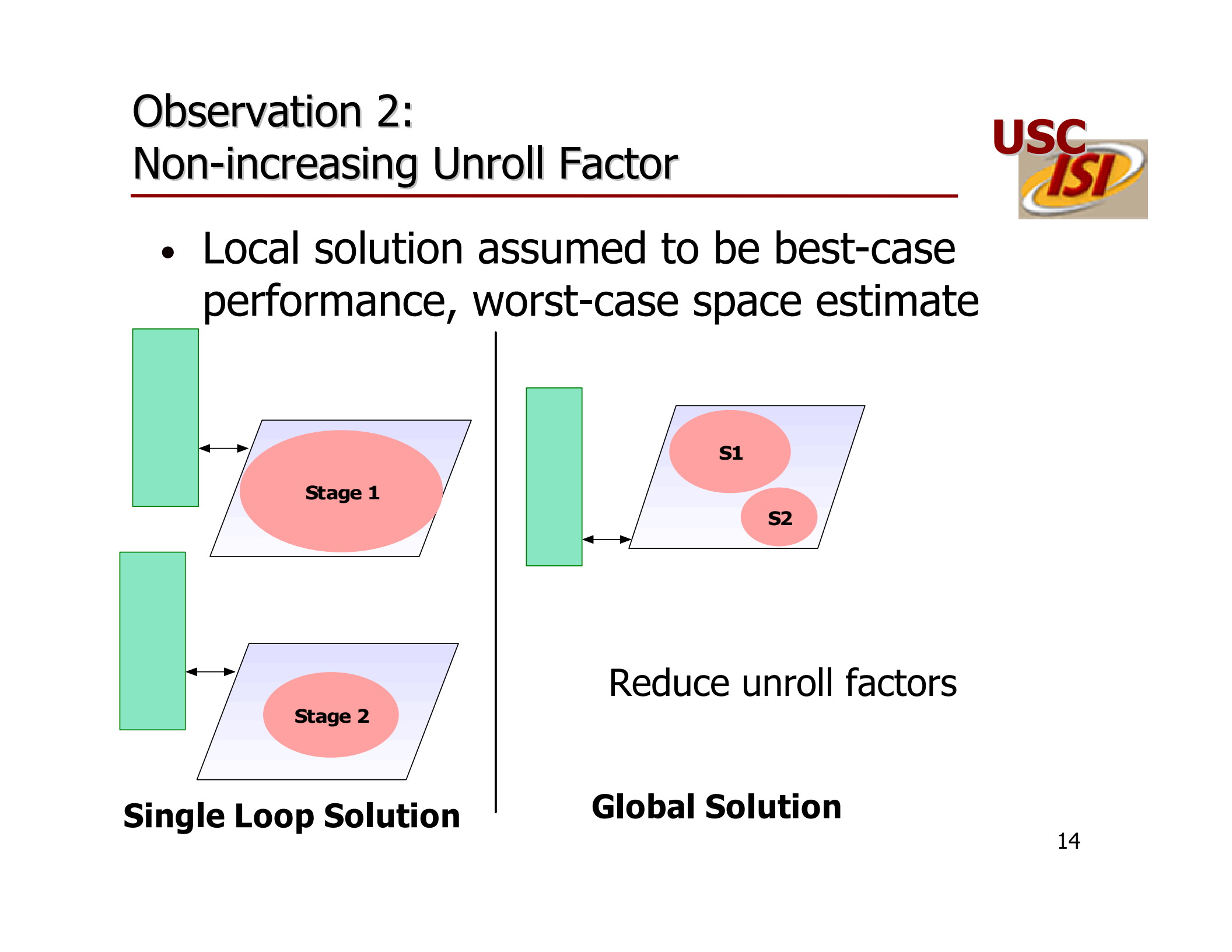
Observation 1: Non-increasing Memory Accesses

- Choose to place communication on-chip



Single Loop Solution **Global Solution**

Observation 2:
Non-increasing Unroll Factor



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

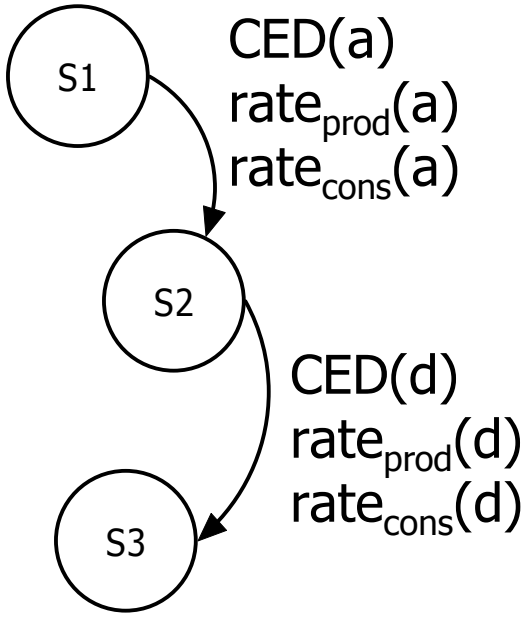
Single Loop Solution

Reduce unroll factors

Global Solution

Observation 3:
Matching Rates without Affecting Performance

- Avoid creating longer critical paths



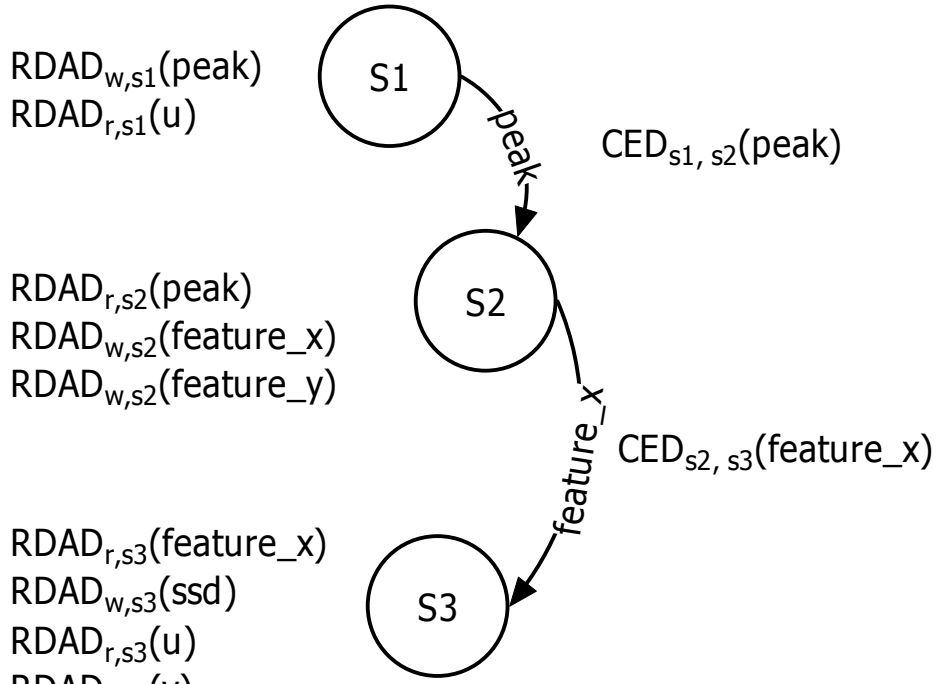
If rate$_{prod}$(d) < rate$_{cons}$(d),
we can safely reduce the unroll factor for S3
until the rates match

Optimization Algorithm: Step 1

Apply Pipeline and Communication Analysis

```
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 …….
       }
     }
```



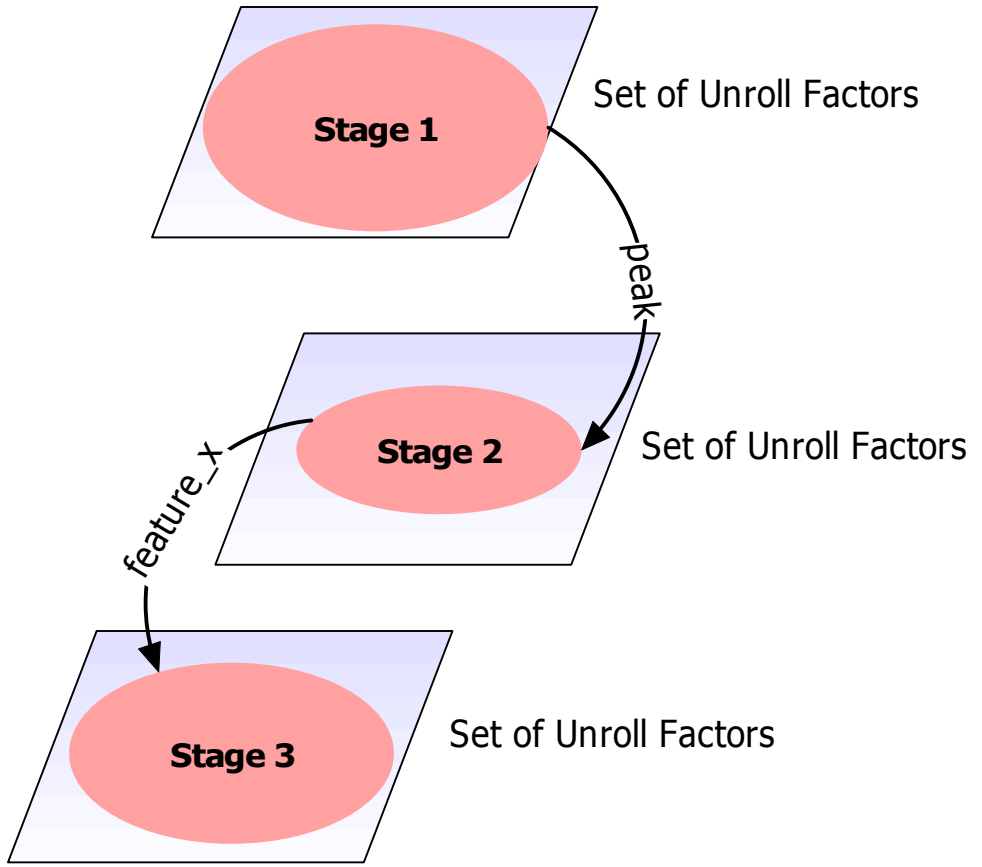
Optimization Algorithm: Step 2

Find Single Loop Solutions in Isolation

```
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;
  }
}
```

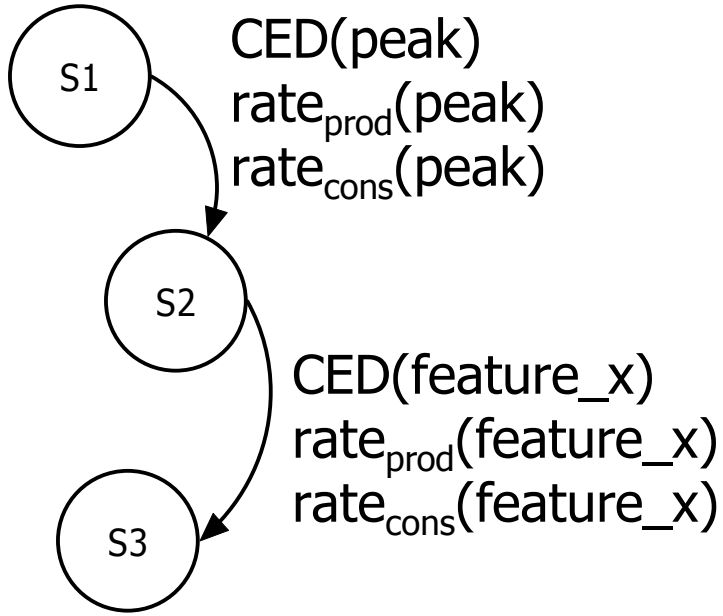
```
for (x=0;x<image-2;x++) {
  for (y=0;y<image-2;y++) {
       if (peak[x][y] > threshold)
          feature_x[x][y] = x;
       else feature_x[x][y] = 0;
  }
}
```

```
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



$rate_{prod}(peak) = rate_{cons}(peak)$

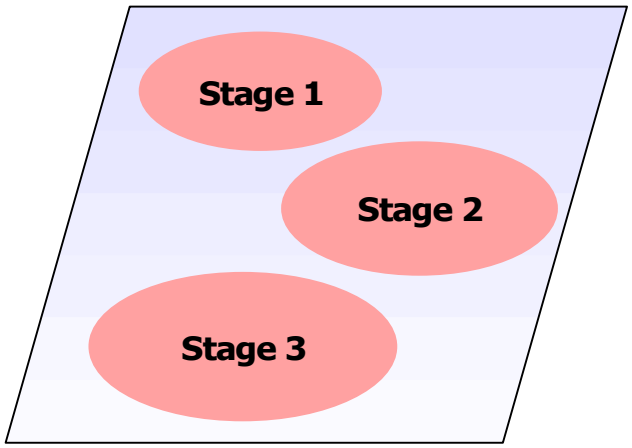
$rate_{prod}(feature\_x) = rate_{cons}(feature\_x)$

Optimization Algorithm: Step 4

Apply Greedy Strategy to Meet Chip Constraint

$$capacity \geq \sum_{1}^{n} area_i$$

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