Improving High Level Synthesis Optimization Opportunity through Polyhedral Transformations

Wei Zuo[1,2], Yun Liang[3], Kyle Rupnow[4], Peng Li[3], Deming Chen[1,4], Jason Cong[3,5]

[1]ECE, University of Illinois, Urbana-Champaign, USA

[2]School of Information and Electronics, Beijing Institute of technology, China

[3] School of EECS, Peking University, China

[4]Advanced Digital Science Center, Singapore

[5]Computer Science, University of California at Los Angeles, USA

Advantages and Issues of High Level Synthesis (HLS)

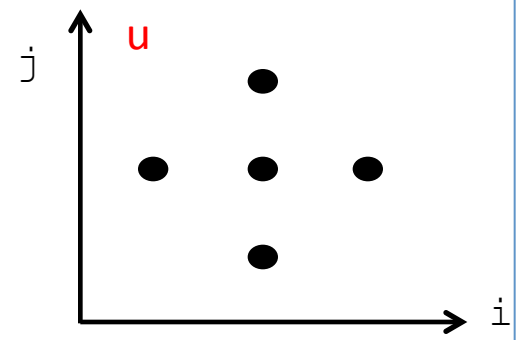
- Good results dealing with single block (a module or a loop kernel)
  - Higher productivity
  - Automated optimization for quality improvement

- A significant performance gap can exist between HLS and manual design for complex applications [Rupnow et al, FPT 2011]
  - For example, 40X difference for a stereo matching implementation
  - HLS facing challenges with multi-dependent blocks

An Illustrative Example: Denoise

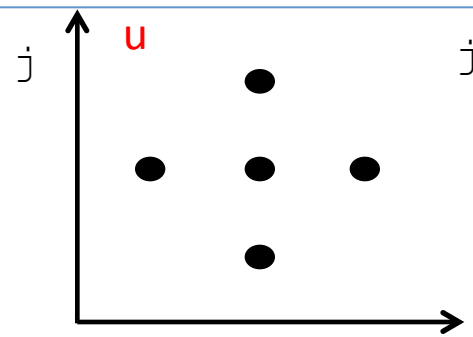
Block 1

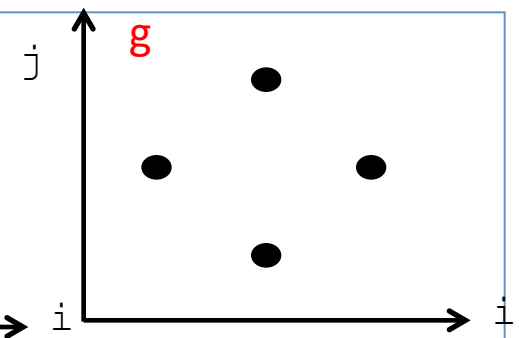
```
for (i = 1; i < 5; i++)
  for (j = 1; j < 5; j++)
{
        g[j][i] = f1(u[j][i], u[j-1][i], u[j+1][i], u[j][i-1], u[j][i+1]);
}
```



Block 2

```
for (i = 1; i < 5; i++)
  for (j = 1; j < 5; j++)
{
        u[j][i] = f2(u[j][i], u[j][i+1], u[j][i-1], u[j+1][i], u[j-1][i], g[j][i+1], g[j][i-1], g[j+1]
[i], g[j-1][i]);
}
```



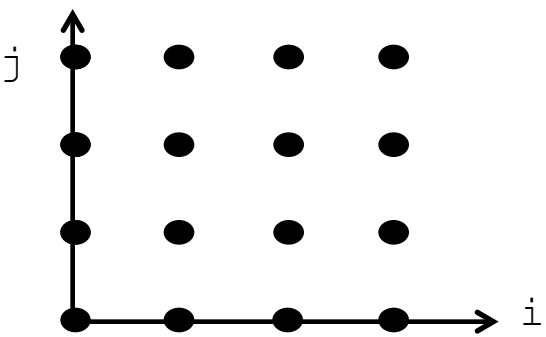


Inter-block Dependence G: RAW

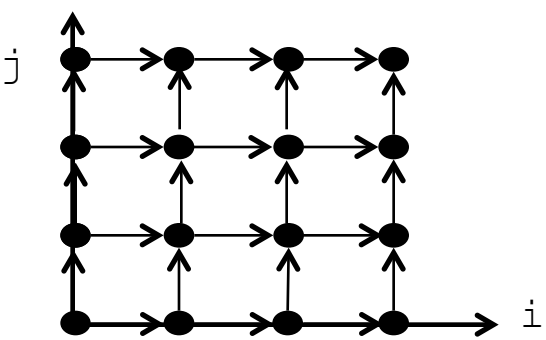
U: WAR

Dependence Analysis

- Intra-block



Block 1

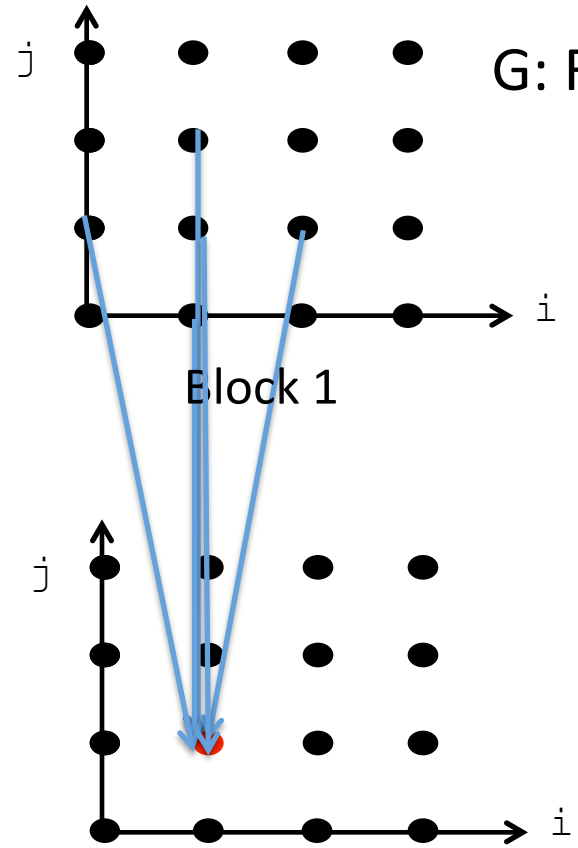


Block 2

- Inter-block

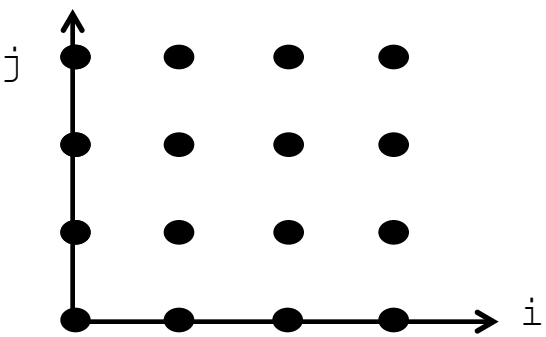
U: WAR

G: RAW

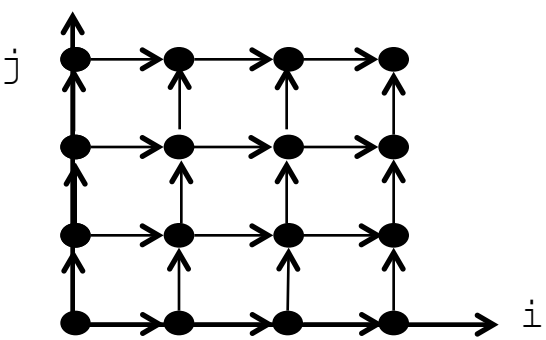


Dependence Analysis

- Intra-block



Block 1

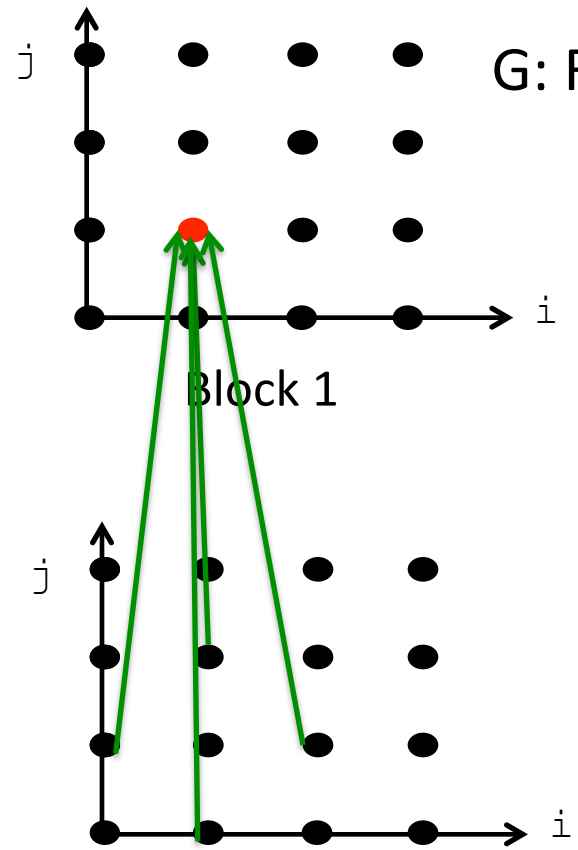


Block 2

- Inter-block

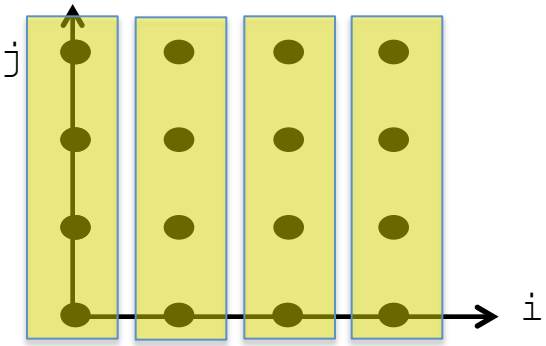
U: WAR

G: RAW

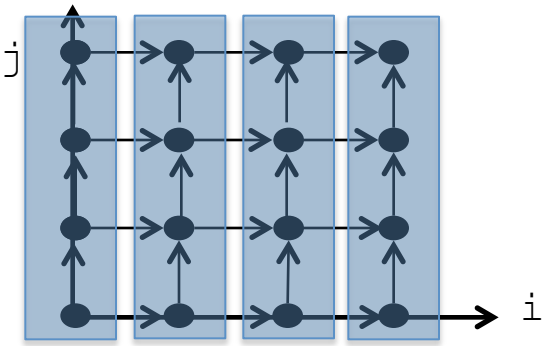


Dependence Analysis

- Intra-block

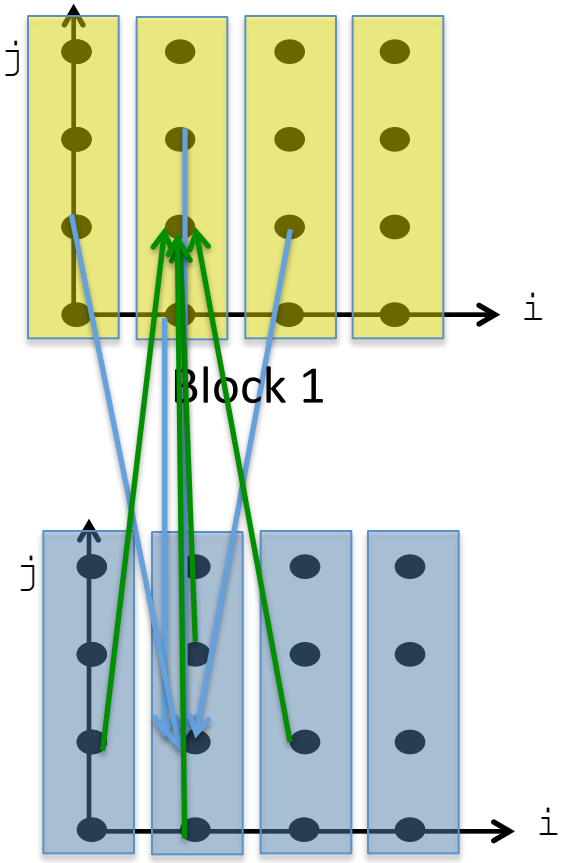


Block 1



Block 2

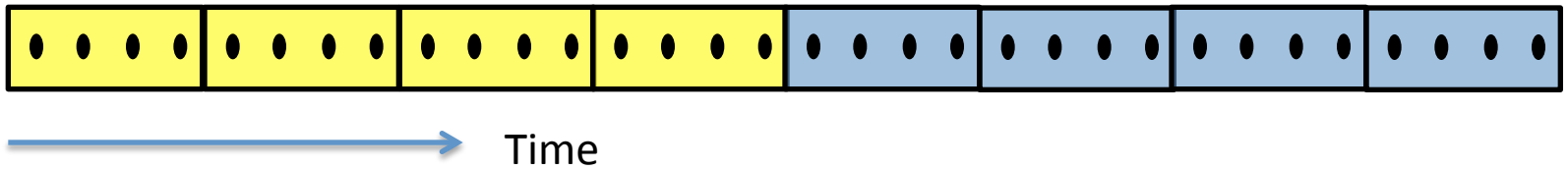
- Inter-block



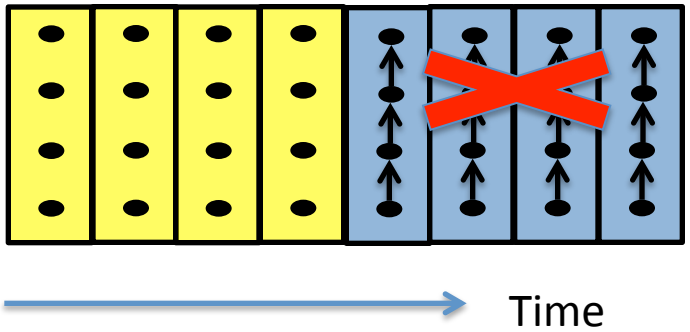
Intra-block Parallelization

	Cycle	Frequency (MHz)
Baseline	5408	160

- The baseline:



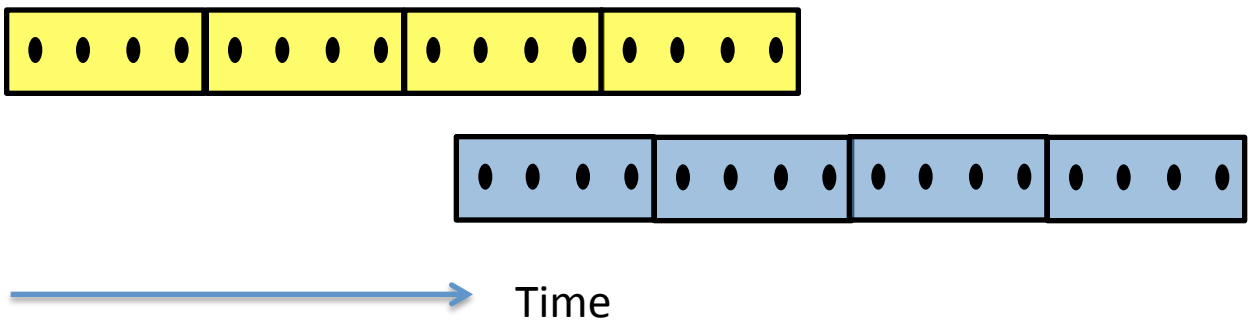
- After parallelization:



Inter-block Data Streaming

- With optimization:
  - Can be streamed
  - The initiation interval is long

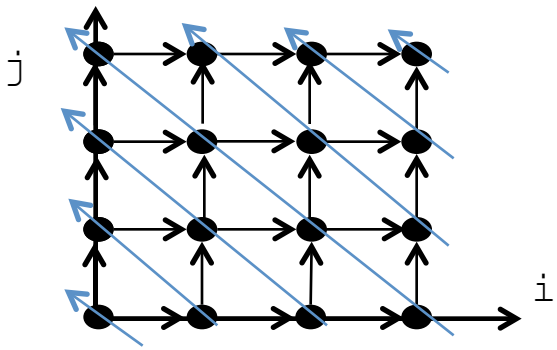
	Cycle	Frequency(MHz)	Speedup
Baseline	5408	160	1
Opt1	**1808**	**182**	**3.40**



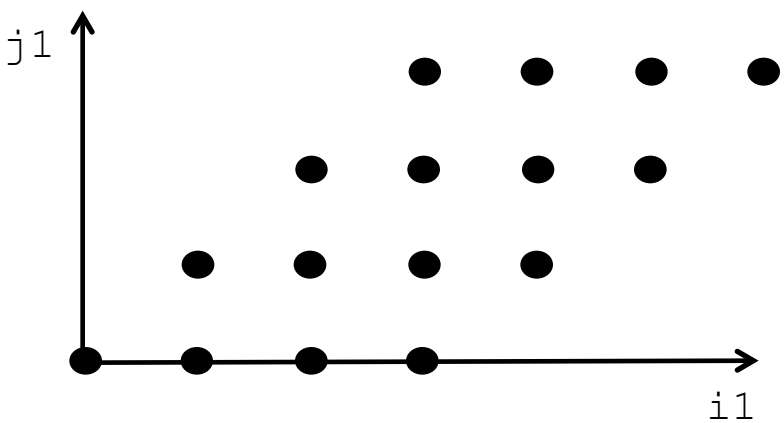
- Transformation is needed for more efficient data streaming

Transformation

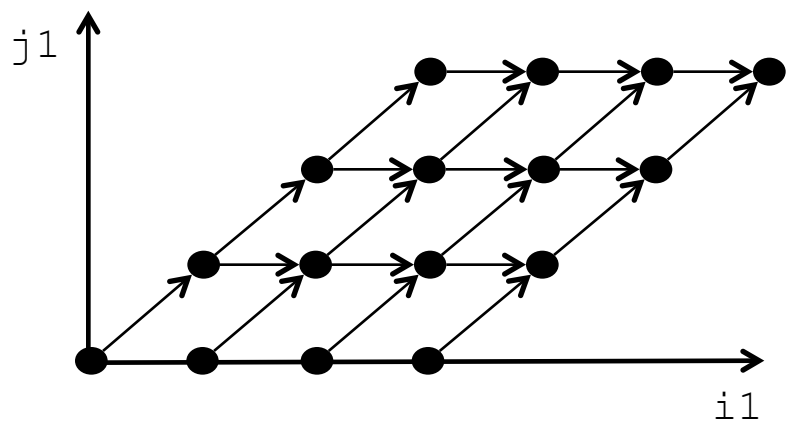
Diagonal array access order



Loop Transformation



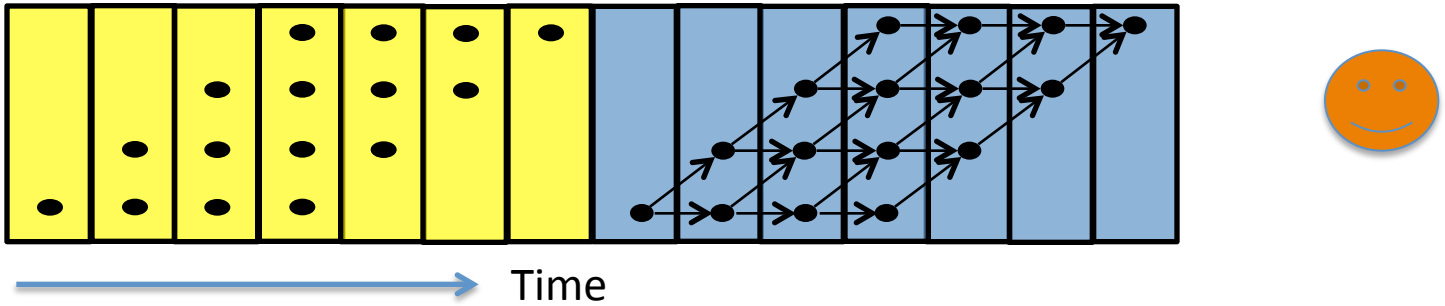
Block 1



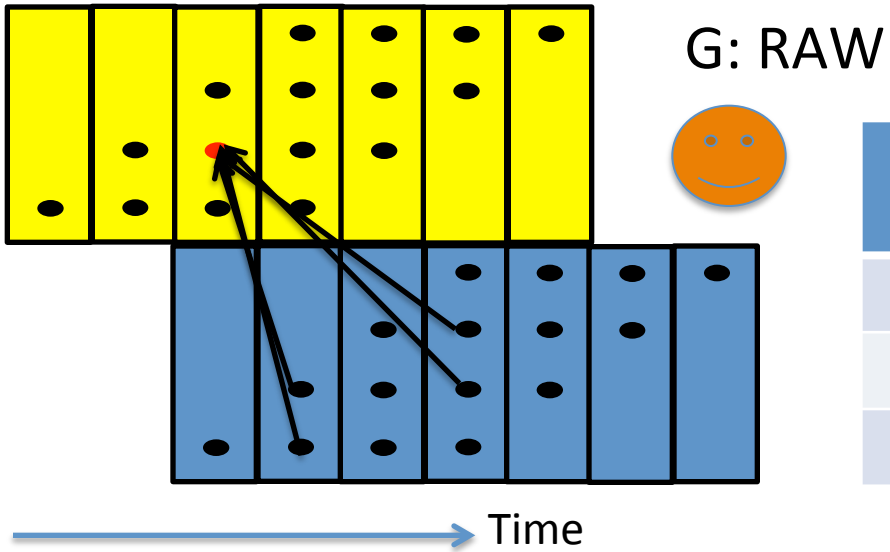
Block 2

HLS Result after Transformation

- Optimization 1: Intra-block parallelization



- Optimization2: Inter-block data streaming

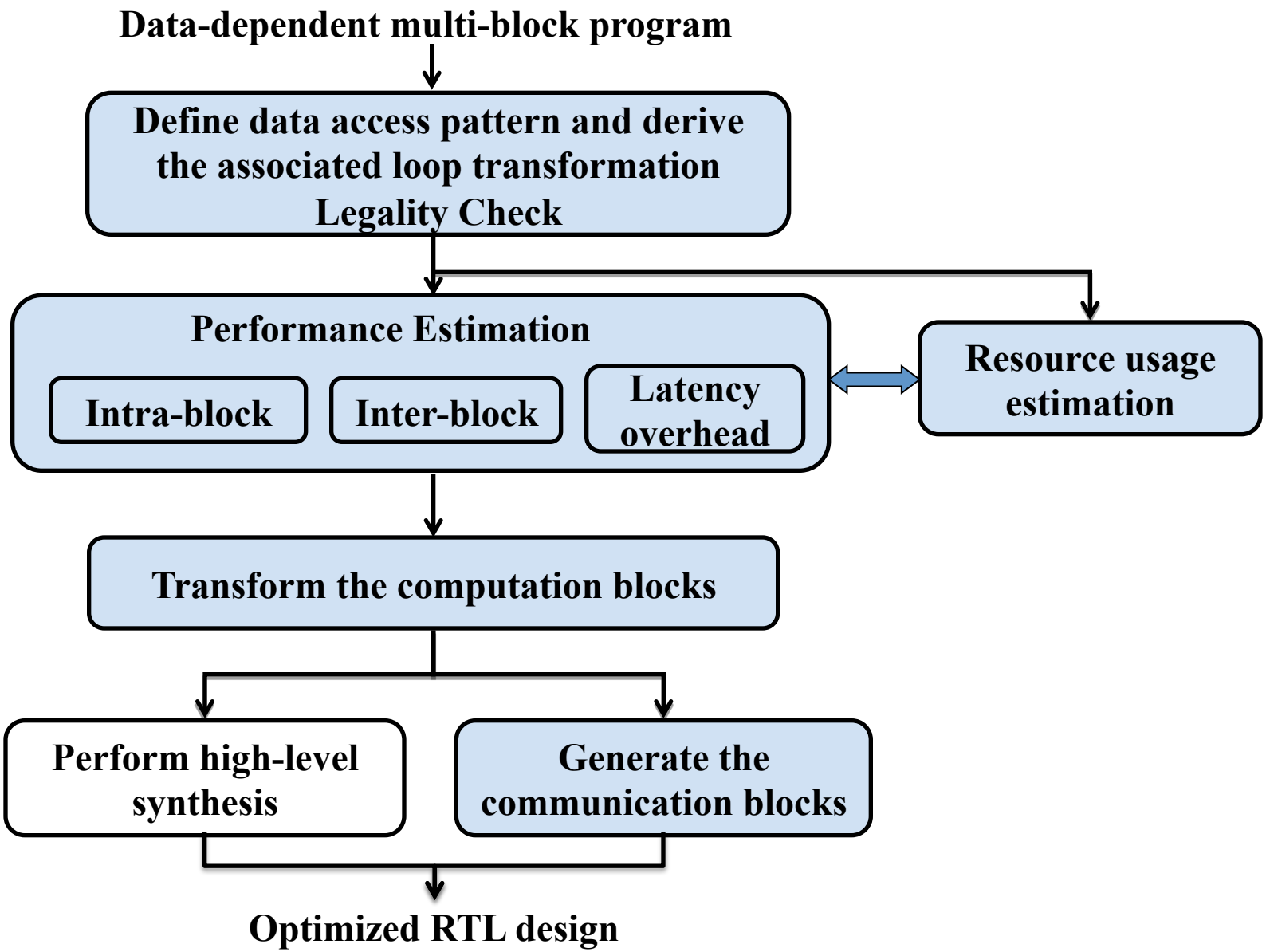


	Cycle	Frequency (MHz)	Speedup
Baseline	5408	160	1
Opt1	1808	182	3.40
Opt1&2	**250**	**230**	**31.09**

Our Goals and Contributions

- Improve optimization opportunities for HLS using polyhedral model:
  - Integration of intra-block parallelization and inter-block data streaming
  - Memory access pattern analysis and loop transformation
  - Performance optimization with efficient hardware implementation
  - Fully automated polyhedral model-based framework

Optimization Flow

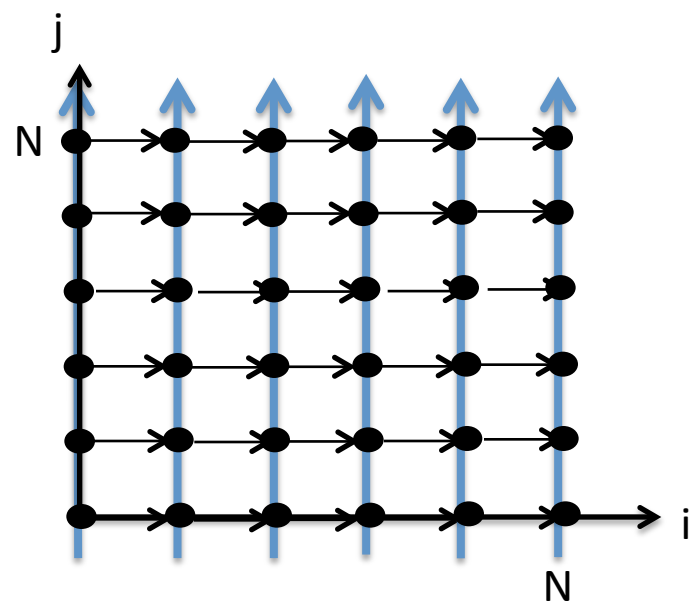


Polyhedral Model

- A powerful mathematic model for representing loop nest and performing loop transformation
  - Loop iteration domain
  - Dependence information
  - Scheduling function

Source code

```
for(i=0; i<=N; i++)
  for(j=0; j<=N; j++)
      S1: B[i+1][j] = B[i][j] + A[i][j];
```

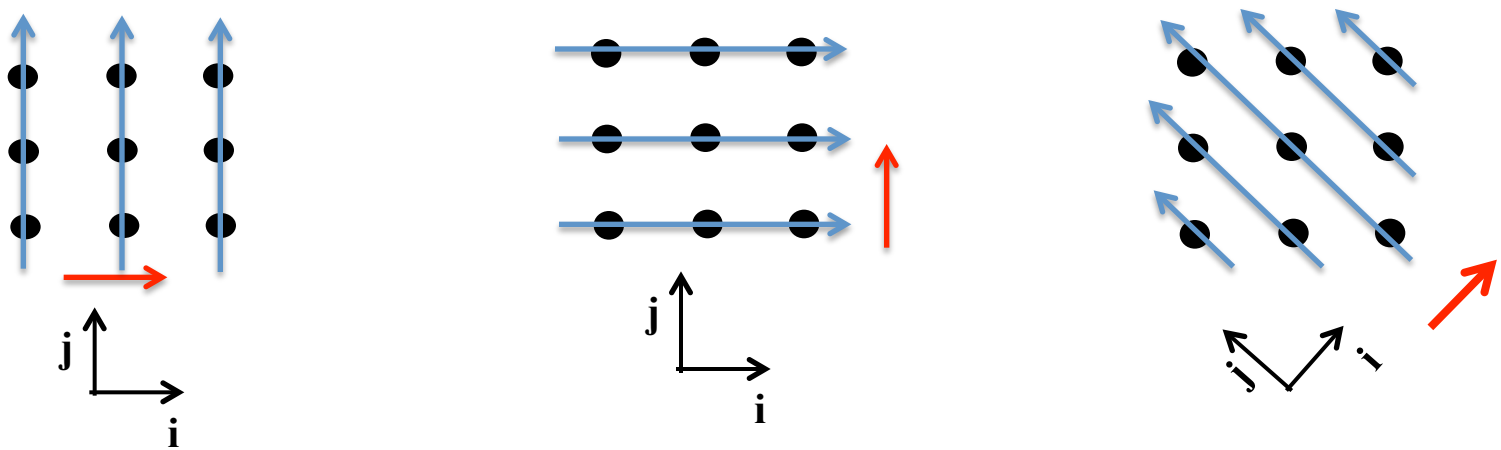


Array Access Pattern (M)

- Define array access patterns using polyhedral model
- For 2-D array:

$$M = \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}$$

Different values of *M* lead to different patterns.



Loop Transformation

- Find the loop transformation according to different array access pattern

Theorem (In paper)

The transformation function T required for the desired data Access pattern M_{des} is

$$T = M_{des}^{-1} M_{ori} F_{ori}^{-1}$$

- M_{ori} : Data access pattern without transformation
- F_{ori}: Schedule function of without transformation

Performance Metric

- Objective: Maximize the speedup
- A model that combines intra- and inter-block speedup and the implementation latency overhead

Performance estimation in clock cycle

For each pattern *P*, the overall latency lat_P is:

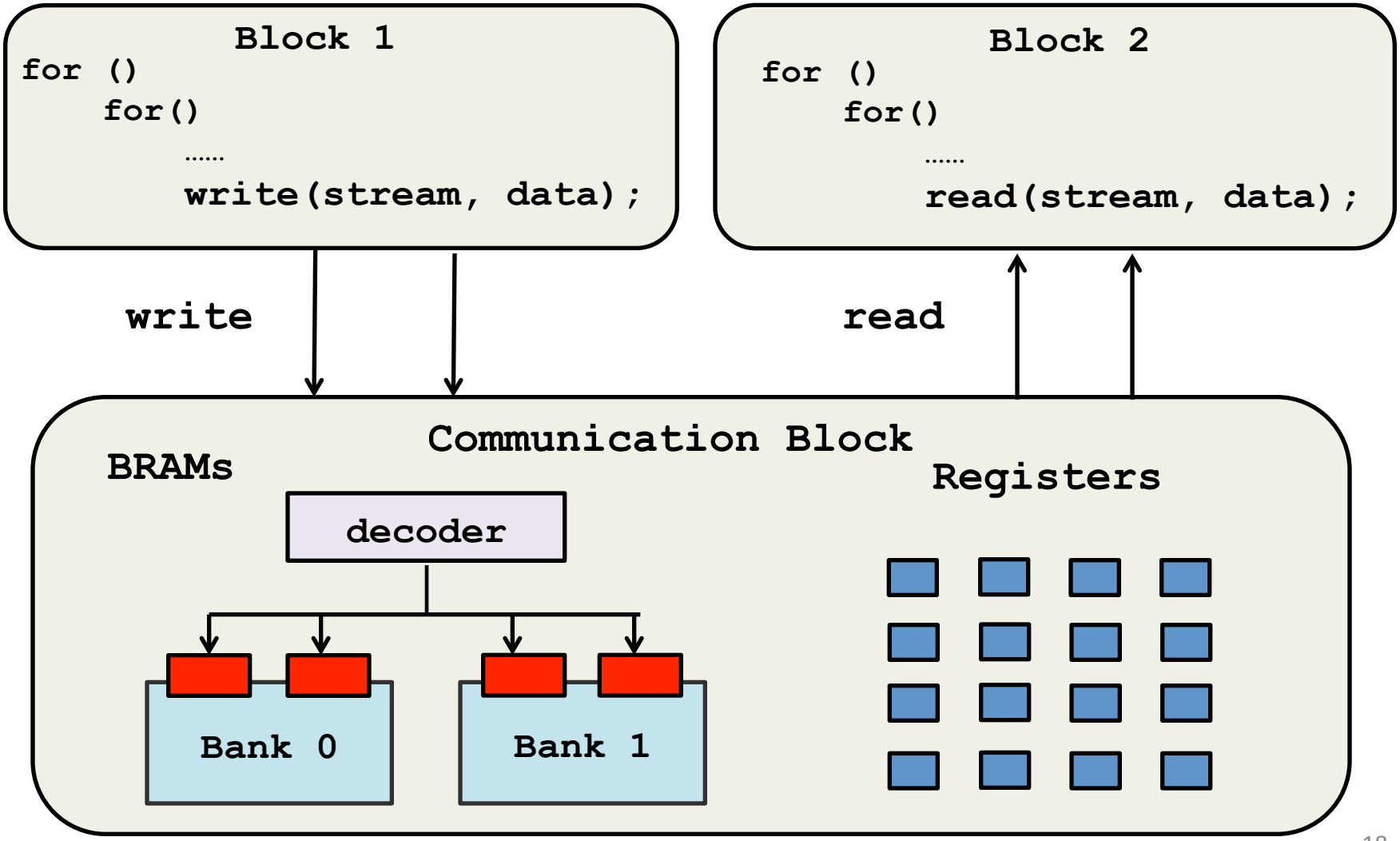
$$lat_P = \frac{lat_{base}}{S_P^{intra} \times S_P^{inter}} + cost_P$$

Computation Block after Transformation

Block1

```
AP_STREAM(float, inU_0)
...
float seidel_moduleA_transformed(){
  float data_inU_0;
  ...
  AP_STREAM_READ(inU_0; data_inU_0);
  ...
 for (c2=1;c2<2*N-1;c2++)
   for (c4=1;c4<N-1;c4++) {
      if (c4>=max(1,c2-N+2) && c4 < min(c2+1, N-1)){
              result = f1(data_inU_0, data_inU_1, data_inU_2,  data_inU_3, data_inU_4);
       AP_STREAM_WRITE(outB_0, result);
                    ...
       }
       ...
}
```

Communication Block



Experiment: Optimization Details

Benchmark	Optimizations		Selected Data Access pattern
	w/o trans	w/trans	
Deconv	Inter-block pipeline	Inter-block pipeline & intra-block parallel	Diagonal(slope = 1)
Denoise	Inter-block pipeline	Inter-block pipeline & intra-block parallel	Diagonal(slope = 1)
Seg	none	Inter-block pipeline & intra-block parallel	Diagonal(slope = 1)
Seidel	Inter-block pipeline	Inter-block pipeline & intra-block parallel	Diagonal(slope = 2)
Jacobi	Intra-block parallel	Inter-block pipeline & intra-block parallel	Row

HLS tool: AutoPilot 2011.3
FPGA Device: Xilinx-Virtex-6 LX75T
Physical Implementation tool: Xilinx ISE13.1

Performance Speedup

Benchmark	Implementation	Cycles	Frequency (MHz)	Speedup
Deconv	w/o trans, w/o opt	5408	151	1
	w/o trans, w/ opt	1809	182	3.59
	w/ trans, w/ opt	257	192	25.25
Denoise	w/o trans, w/o opt	5408	160	1
	w/o trans, w/ opt	1809	182	3.40
	w/ trans, w/ opt	250	230	31.09
Seg	w/o trans, w/o opt	9864	117	1
	w/o trans, w/ opt	9864	117	1
	w/ trans, w/ opt	500	156	26.90
Seidel	w/o trans, w/o opt	64803	103	1
	w/o trans, w/ opt	1818	134	46.34
	w/ trans, w/ opt	1130	134	74.55
Jacobi	w/o trans, w/o opt	5373	101	1
	w/o trans, w/ opt	1439	134	4.85
	w/ trans, w/ opt	482	133	14.49

Resource Usage

Benchmark	Implementation	LUT	FF	DSP	BRAM
Deconv	w/o trans, w/o opt	3234	948	24	48
	w/o trans, w/ opt	6433	2650	24	5
	w/ trans, w/ opt	13819	13826	108	17
Denoise	w/o trans, w/o opt	3266	948	24	5
	w/o trans, w/ opt	6503	2672	24	5
	w/ trans, w/ opt	13817	13824	108	17
Seg	w/o trans, w/o opt	3735	1202	30	24
	w/o trans, w/ opt	3735	1202	30	24
	w/ trans, w/ opt	13824	9560	216	34
Seidel	w/o trans, w/o opt	1400	891	2	2
	w/o trans, w/ opt	13375	6626	32	6
	w/ trans, w/ opt	47402	20040	96	14
Jacobi	w/o trans, w/o opt	5563	1890	3	16
	w/o trans, w/ opt	39430	18832	64	10
	w/ trans, w/ opt	38877	18664	64	10

Conclusions and Future Work

- An integrated technique using polyhedral model
- Both intra- and inter-block optimization
  - 29.6X speedup over the un-optimized source code
  - 6X speedup over code that is optimized but not transformed
- Substantially improving HLS optimization opportunities

- Future work
  - Multi-dimension arrays and multiple (>2) kernels
  - Different access patterns across the set of different kernels
  - Multiple kernels pipelining and average initiation interval optimization

Thanks!