Theory and Algorithm for Generalized Memory Partitioning in High-Level Synthesis

Yuxin W., Peng L., Jason C.

FPGA'14, Feb 26-28, 2014

March 14, 2022

Presented by: Akshay G

About the Authors

北京大学高能效计算与应用中心
Center for Energy-efficient Computing and Applications

Joint Research Institute
in Science and Engineering
by Peking University and UCLA

Outline

- Introduction.
- Background.
- Bank Partitioning.
- Intra-bank Offset.
- Results.
- Comments.

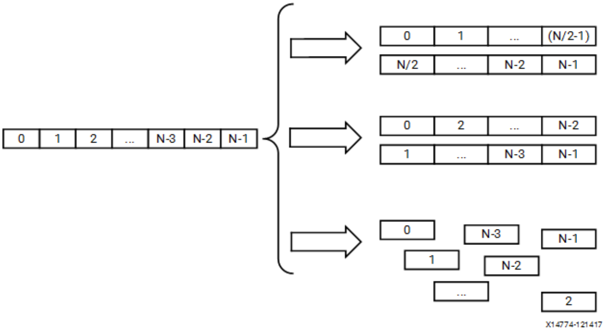
Introduction

- Memory Partitioning problem.
- Partitioning given multiple memory ports?
- Algorithm parametric to partition scheme?
- Modular to memory ports?

Memory Partitioning

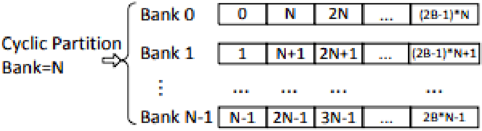
```
int A[w0][w1];
for (j=1; j<w0-1; j++)
 for (i=1; i<w1-1; i++)
  foo(A[j][i-1], A[j-1][i ], A[j+1][i], A[j][i+1]);
```

(a) Loop kernel

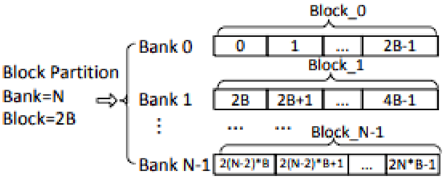


Partitioning Schemes

(a) Original data

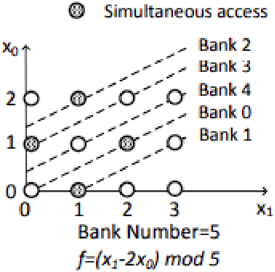


(b) Cyclic partitioning

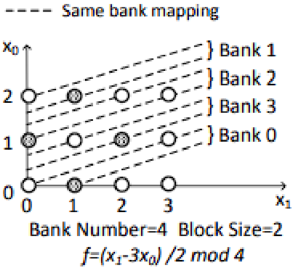


(c) Block partitioning

Efficient Mapping



(b) Cyclic partitioning



(c) Block-cyclic partitioning

Towards Theory: Symbols!!

Table 1: Symbol table

Variables	Meaning
N	Partition factor, representing the number of logic banks used after memory partitioning
B	Partition block size
P	Memory port number
l	Level of loop nest
d	Number of dimensions of the array
m	Number of array references in the inner loop
$\mathcal{D}$	Iteration domain
$\mathcal{M}$	Data domain
$\vec{i}$	Iteration vector
$\vec{x}$	Array index vector
$\vec{\alpha}$	Partition vector
q	Padding size
i, j, k, t	Temporal variables
$\mathbb{Z}$	Integer set
w_k	The k-th dimensional size of the array

Iteration Vector and Affine Reference


```
int A[w0][w1];
for (j=1; j<w0-1; j++)
 for (i=1; i<w1-1; i++)
  foo(A[j][i-1], A[j-1][i ], A[j+1][i], A[j][i+1]);
```

(a) Loop kernel

Example 1. An affine array reference $A[i_0][i_1+1]$ is represented as $\vec{x} = (i_0, i_1+1)^T$, where

$$\vec{x} = \begin{pmatrix} i_0 \\ i_1 + 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} i_0 \\ i_1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Framing the Partitioning Problem

Two parts:

- Bank Minimization.
- Storage Minimization.

$$Minimize : N = max_{\leq i < m}\{f(\vec{x}_i)\}$$
$$\exists \vec{i} \in \mathcal{D}, 0 \leq j < k < m, f(\vec{x}_j) \neq f(\vec{x}_k).$$

$$Minimize : \sum_{j=0}^{N-1} max_{\leq i < m, f(\vec{x}_i)=j}\{g(\vec{x}_i)\}$$
$$\forall \vec{x}_j, \vec{x}_k \in \mathcal{M}, (f(\vec{x_j}), g(\vec{x_j})), \neq (f(\vec{x_k}), g(\vec{x_k})).$$

Bank Mapping

$$\mathcal{P}_{conf}(\vec{x}_0, \vec{x}_1) = \{\vec{i} | \forall \vec{i} \in \mathcal{D}, f(\vec{x}_0) = f(\vec{x}_1)\}.$$

Obviously, if $\forall \vec{i} \in \mathcal{D}$, $f(\vec{x}_0) \neq f(\vec{x}_1)$, $\mathcal{P}_{conf}(\vec{x}_0, \vec{x}_1)$ is empty.

$$\mathcal{P}_{conf} : \begin{cases} \vec{\alpha} \cdot (A_0 - A_1) \cdot \vec{i} + \vec{\alpha} \cdot (C_0 - C_1) + Nk = 0 \\ \vec{i} \in \mathcal{D} \\ k \in \mathbb{Z} \end{cases}$$

Bank Mapping: Multi Port

$$f(\vec{x}_0) = f(\vec{x}_1) \text{ and } f(\vec{x}_1) = f(\vec{x}_2).$$

The conflict polytope is constructed as

$$\mathcal{P}_{conf} : \begin{cases} \vec{\alpha} \cdot (A_0 - A_1) \cdot \vec{i} + \vec{\alpha} \cdot (C_0 - C_1) + Nk_0 = 0 \\ \vec{\alpha} \cdot (A_1 - A_2) \cdot \vec{i} + \vec{\alpha} \cdot (C_1 - C_2) + Nk_1 = 0 \\ \vec{i} \in \mathcal{D} \\ k_0, k_1 \in \mathbb{Z} \end{cases}$$

Intra-bank Offset Mapping

Definition 7. (Bank Polytope [16]) Given a d-dimensional array reference $\vec{x}$, $\mathcal{P}_{bank}(\vec{x})$ is a bank polytope of $\vec{x}$ in the data domain $\mathcal{M}$ defined as

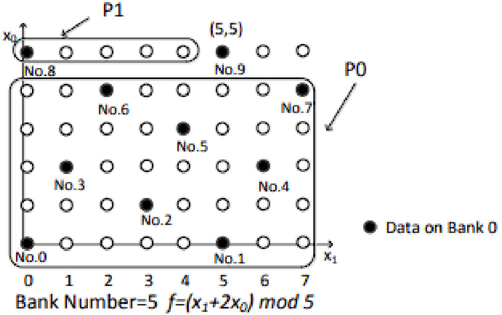
$$\mathcal{P}_{bank}(\vec{x}) = \{\vec{y} | \forall \vec{y} \in \mathcal{M}, f(\vec{x}) = f(\vec{y})\}.$$

Definition 8. (Lexicographic Order) A lexicographic order $\prec_{lex}$ on a d-dimensional set $\mathcal{M}$ is a relation, where for $\forall \vec{x}, \vec{y} \in \mathcal{M}$, $\vec{x} = (x_0, x_1, ..., x_{d-1})$ and $\vec{y} = (y_0, y_1, ..., y_{d-1})$,

$$\vec{y} \prec_{lex} \vec{x}$$

$$\Leftrightarrow \exists 1 < t < d, \forall 0 \leq i < t, (x_i = y_i) \wedge (y_t < x_t).$$

Offset Mapping Intuition



$$g(\vec{x}) = \sum C(\mathcal{P}_t(\vec{x}))$$

Algorithm Flow

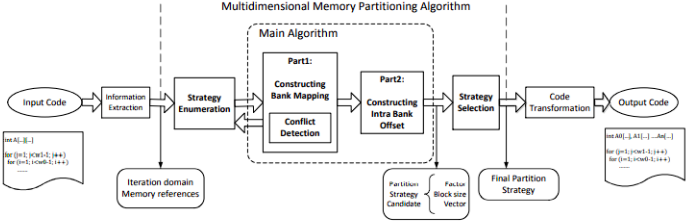


Figure 5: The design flow

Results (compared to previous work LTB)

Table 5: Experimental results

Benchmark	Access #	Bit width	Method	BRAM	Slice	DSP	CP (ns)	Dynamic Power(mw)
DENOISE	4	32	LTB [21]	5	520	4	3.729	26
			GMP (P=1), B=2	4	303	0	3.395	16
			GMP vs LTB	-20.00%	-41.73%	-100.00%	-8.96%	-38.46%
DECONV	5	32	LTB [21]	5	597	5	4.538	27
			GMP (P=1), B=1	5	597	5	4.538	27
			GMP vs LTB	0.00%	0.00%	0.00%	0.00%	0.00%
DENOISE-UR	8	32	LTB [21]	8	794	0	3.738	31
			GMP (P=1), B=1	8	794	0	3.738	31
			GMP vs LTB	0.00%	0.00%	0.00%	0.00%	0.00%
BICUBIC	4	32	LTB [21]	5	483	4	4.364	24
			GMP (P=1), B=2	4	238	0	3.169	15
			GMP vs LTB	-20.00%	-50.72%	-100.00%	-27.38%	-37.50%
SOBEL	9	32	LTB [21]	9	1523	9	4.468	53
			GMP (P=1), B=1	9	1523	9	4.468	53
			GMP vs LTB	0.00%	0.00%	0.00%	0.00%	0.00%
MOTION-LV	6	8	LTB [21]	6	538	6	3.682	25
			GMP (P=2), B=1	4	425	0	3.169	25
			GMP vs LTB	-33.33%	-21.00%	-100.00%	-13.93%	0.00%
MOTION-LH	6	8	LTB [21]	6	536	6	3.946	21
			GMP (P=2), B=1	4	334	0	3.169	23
			GMP vs LTB	-33.33%	-37.69%	-100.00%	-19.69%	9.52%
MOTION-C	4	8	LTB [21]	4	174	0	3.405	14
			GMP (P=2), B=1	2	155	0	3.169	12
			GMP vs LTB	-50.00%	-10.92%	0.00%	-6.93%	-14.29%
Average			**GMP vs LTB**	-19.58%	-20.26%	-50.00%	-9.61%	-10.09%

Limitations?

- Experiments on Partitioning algorithm performance.
- Lack of some important mathematical details (some function definitions missing).

Thank you

Questions?