Conflict-Free Access to Rectangular Subarrays in Parallel Memory Modules

by

Doreen Lynn Erickson

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 1993

Abstract

In a parallel computing environment, we consider conflict-free access to constant-perimeter rectangular subarrays, using a natural formulation in terms of latin squares. For parallel matrix computations, there are frequently portions of data, referred to as templates, that one desires to be stored and retrieved in such a way that one can operate on the data simultaneously with each processor working on one element of the template. If more than one processor attempts to retrieve data from a single memory module during the same memory cycle, there is a memory conflict. When the array is stored to allow the desired templates to be accessible from the memory modules without memory conflicts, it is conflict-free for that set of templates. In this thesis, we examine a set of structured templates where the number of template instances defined by template grows with respect to the maximum size of a template. In particular, we examine the set of constant-perimeter rectangular subarrays.

A square is *perimeter rectangular conflict-free* (p_{rcf}) if it is conflict-free for all rectangular subarrays whose perimeter is less than or equal to $2p$. If p is even, the problem is to provide conflict-free access to all rectangular subarrays of size $(\frac{p}{2} - i) \times (\frac{p}{2} + i)$ for $-\frac{p}{2} < i < \frac{p}{2}$. We show that the necessary number of memory modules is $\frac{p^2}{2} - p + 1$. Furthermore, $\frac{p^2}{2} - p + 1$ memory modules are sufficient, and there is a linear skewing scheme to realize this bound. If p is odd, the problem is to provide conflict-free access to all rectangular subarrays of size $(\lfloor\frac{p}{2}\rfloor - i) \times (\lceil\frac{p}{2}\rceil + i)$ for $-\lfloor\frac{p}{2}\rfloor < i < \lfloor\frac{p}{2}\rfloor$. We show that the necessary number of memory modules is $\lfloor\frac{p}{2}\rfloor^2$, and there is a linear skewing scheme that realizes this bound. We also provide bounds and constructions for subsets of constant-perimeter rectangular subarrays, in particular all $(x - i) \times (y + i)$ rectangular subarrays of an $n \times n$ array for all nonnegative i where $p = x + y$. The linear results for the constant-perimeter rectangular subarrays hold when the subarrays are stretched by a factor of v where v is relatively prime to the number of memory modules. A subarray is stretched by v if every vth element in a row and every vth row is selected. In this situation, the perimeter includes only those elements in the rectangular subarray and not those elements skipped because of the stretch. Thus, the perimeter of a given rectangular subarray is the same regardless of its stretch.

In addition, the perimeter results provide a lower bound for conflict-free access to constant-area rectangular subarrays. An upper bound is found using a technique of relating conflict-free access to the chromatic number of a graph. In addition to bounds for constant-area rectangular subarrays, some computational results are provided. Finally, we propose a new method for defining skewing schemes, in particular, skewing schemes defined by permutations.

Acknowledgements

First and foremost, I wish to extend my sincerest gratitude to my supervisor, Charlie Colbourn. Thank you for keeping me as motivated as I wanted to be, for your advice, your guidance, your endless assistance, for always finding time even when no time existed and for introducing me to a whole new world of research and academia. One could not ask for a better supervisor and I'm glad I have had the priviledge of working with you.

I would like to send my gratitude to my committee for their encouragement throughout the program and their helpful comments on my thesis. Specifically, I would like to thank Ron Mullin, Wei Pai Tang (and his 'warm body' representative, Peter Forsyth), and my external examiner, Frantisek Franek. I would also like extend a HUGE thank you to David Taylor who was an extremely helpful committee member and an office neighbour. Thank you for everything including your detailed and helpful comments on the thesis and preliminary papers and for putting up with my stereo on weekends.

I would like to acknowledge the University of Waterloo, in particular, the Math Faculty and the Computer Science Department, for providing me with the environment and means for me to learn all that I have during my stay as a graduate student. In particular I would like to thank the high school liason group for sharing with me, their love of teaching and working with students and Sue Guckenberger for her endless help.

I wish to acknowledge Mark Fishman, Ed Gallizzi and George Loftquist at Eckerd College. It was their encouragement in undergrad that brought me into computer science, it was their faith in me that brought me to graduate school and it was the knowledge and background I obtained from them that helped me in my research.

I would like to extend my appreciation to a great friend, Ron Castelletto. Thanks for being a friend through the good times and the stressed times. I appreciate all the times you would listen just to help me think. I appreciate all the distractions when I needed them and the encouragement when things were not going the best. I would also like to acknowledge Ron's family in particular his mother and the fantastic chicken soup that helped me through many cold winter days.

I would like to acknowledge those that have opened their homes to me during the defence/correction phase and have helped me through a very tough time in the program and in life. In particular, I would like to thank Mandy Skaff, Verna Friesen and her family.

Finally I would like to thank all my friends in the department, the Minotians, the SigSporters, my buddies at Abstract, my email buddies, new friends I have met and friends I have yet to meet and finally, my fellow Floridian, Drew. The tough times are always easier and the good times are better with friends like ya'll.

Thank you everyone.

Dedication

This thesis is dedicated to my parents who have always believed in me, who have always tried to understand and who have always encouraged me to chase my dreams even if it means living 1500 miles away.

This thesis is also dedicated to the memory of my grandma, Anna Bednarz. It was the strength I inherited from her through my mother that helped me through the many tough times that accompany a Ph.D. program.

Thank you!

Doreen

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In a parallel computing environment, consider the problem of storing an $n \times n$ array in n parallel memory modules. If the entire array is stored in the same memory module, then a memory 'bottle neck' occurs when multiple processors are simultaneously operating on the array. More formally, a *memory conflict* [1]* occurs when more than one request is given to the same memory module during a single memory cycle. Hence, one may wish to spread the information among the various memory modules. Ideally, one would optimize the distribution of information among the n memory modules. This may be accomplished by determining which parts of the array will be requested by the processors at the same time. Historically, to make this determination, one would assume that all the processors work under the same clock. This would imply the use of the SIMD (single instruction stream, multiple data stream) model for parallel computation (see Figure 1.1). One could instead assume that the synchronization is done at the software level, and thus generalize to any parallel computation model. Intuitively, a *template** specifies the shape of

[1]An asterisk is used in the text to denote an expression listed in Appendix A, Terms

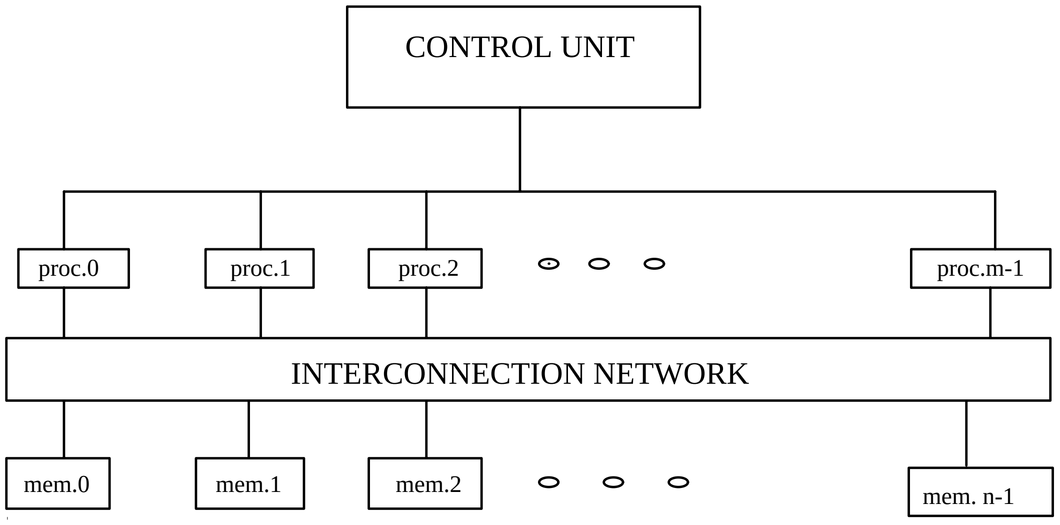

Figure 1.1: SIMD model of computation

a subarray, all instances of which, are desired to be conflict-free. More formally, a template consists of a distinguished position called the handle and possibly other array positions defined by their relative location with respect to this handle. If the actual position of the handle is not specified, it is assumed to be the leftmost element in the uppermost row. Common templates include the row template, the column template, and front and back diagonal templates. All *template instances** are translates of a template. Thus, a template instance is a specific instantiation of a template such as the first row. Only one template instance is accessed in a given memory cycle. An array element could refer to an elementary entity such as an integer or it could refer to a composite entity such as a portion of a larger array. Without loss of generality, we consider it to be an elementary entity. The size of a template is the number of elements contained in the template. The function that maps the array elements into the memory modules is a *skewing scheme**. A skewing scheme is *linear** if it maps the array elements $a_{i,j} \equiv q(i-\alpha) + r(j-\beta) \pmod{n}$ for some fixed integers q, r, α and β [Kuc68]. Otherwise the skewing scheme is *non-*

linear. A square defined by a linear skewing scheme is a cyclic square. A skewing scheme may provide conflict-free access to a single template such as row template or it may provide conflict-free access to a set of templates such as rows, columns and constant-perimeter rectangular subarrays.

In a more general framework, when one has a parallel algorithm, an efficient implementation is desired. One aspect of the efficient implementation entails the storage and retrieval of data. There are several factors or points which allow this action to occur efficiently. The templates are determined by the algorithm (and the synchronization). One could design algorithms to fit existing skewing schemes, or develop skewing schemes that fit the needs of algorithms. One could choose to use a skewing scheme that is conflict-free for a large set of templates. Preferably, one would employ a skewing scheme which realizes at least the templates dictated by the algorithm. This is best summarized by our first point.

POINT 1 The set of templates accessible without memory conflicts should reflect the needs of algorithms.

Once we have a skewing scheme that provides us with the appropriate set of templates, we want to ensure that the skewing scheme is efficient. Efficiency is dependent on the model of computation used. In this thesis, we use the uniform cost RAM model of computation. The function the skewing scheme performs is determining which memory module contains the desired data for a given processor, the second point.

POINT 2 The computation to generate the address of the appropriate memory module should require constant time.

Once the processor computes the memory module it needs to access, it uses an interconnection network (IN) to access the appropriate module. The skewing scheme and the IN need to be compatible. Thus, all the processors and memory modules need to be efficiently connected with respect to the skewing scheme. This idea is expanded upon in Section 1.2 and leads us to the third point.

POINT 3 There should exist an interconnection network that can efficiently realize the processor/memory connections needed by the skewing scheme.

Finally, we want to ensure efficiency in terms of the memory utilization. *Memory utilization** is the percentage of memory modules being accessed for an arbitrary template instance. Using n^2 memory modules would lead to a trivial solution for an $n \times n$ array but low memory utilization. This is the final point.

POINT 4 The memory utilization should be high.

These points are all significant if one is to run a parallel program efficiently. Let us further examine the first point. One could design algorithms to fit existing skewing schemes or develop skewing schemes that fit the needs of algorithms. Ideally, the natural set of templates that one desires for a particular algorithm are those provided by an existing skewing scheme. Suppose there are no skewing schemes that provide conflict-free access to the exact set of templates needed for a particular algorithm or the existing skewing schemes are unacceptable by any of the aforementioned points. In this situation, one would desire to find a new skewing scheme. Thus, the next step would involve determining what is theoretically possible. This would help narrow down the search for a skewing scheme. If no theoretical results are available, one would then desire to have some techniques to discover what is possible. Some theoretical results can be found in [Sha78, Wij89]. When developing skewing schemes, one must predetermine what set of templates are to be conflict-free. One reasonable approach in choosing a set of templates is to accommodate those that are frequently found in parallel computing applications. In this thesis, we focus on rectangular subarrays. These appear in algorithms for applications involving image processing [SMJ92, DH91], biological simulation [RM85] and scientific computing [Fei82, Chu88]. We present lower bounds that apply to both linear and non-linear skewing schemes. We also present constructions, most of which are linear skewing schemes.

The next section, Section 1.1, presents background information on skewing schemes. For the sake of completeness, this chapter includes a survey on interconnection networks in Section 1.2. Finally, an overview of the thesis is presented in Section 1.3.

1.1 Skewing Schemes

In 1968, Kuck [Kuc68] described the linear skewing scheme used for ILLIAC IV. The scheme provided conflict-free access to columns and rows. The skewing scheme is $a_{i,j} \equiv i + j \pmod{n}$. For example $a_{1,1} \equiv 1 + 1 \pmod{4} = 2$. Table 1.1 demonstrates where each of the elements of a 4×4 array are mapped. Each array location is mapped to the memory module identified by the integer in the center of the square. Lawrie and Vora [LV82] claimed to have improved the average case for address generation in their computation model when the number of memory modules is a prime and have their address generation accomplished in hardware. When one assumes that n is prime, the skewing scheme is referred to as a *prime memory system* [LV82]. In 1977, Batcher [Bat77] used a non-linear skewing scheme for STARAN that provided for the same set of templates as Kuck's [Kuc68]. This scheme is best known for its simplicity in terms of addressing hardware and in-

$a_{0,0}$ 0	$a_{0,1}$ 1	$a_{0,2}$ 2	$a_{0,3}$ 3
$a_{1,0}$ 1	$a_{1,1}$ 2	$a_{1,2}$ 3	$a_{1,3}$ 0
$a_{2,0}$ 2	$a_{2,1}$ 3	$a_{2,2}$ 0	$a_{2,3}$ 1
$a_{3,0}$ 3	$a_{3,1}$ 0	$a_{3,2}$ 1	$a_{3,3}$ 2

Table 1.1: Skewing scheme for ILLIAC IV

terconnection network. This scheme uses the xor function and maps $a_{i,j} \equiv i \oplus j \pmod{n}$ where i and j are in their binary form.

A larger set of templates is conflict-free in the linear skewing scheme proposed by Budnik and Kuck [BK71]. This scheme provided conflict-free access to rows, columns, diagonals and $\sqrt{n-1} \times \sqrt{n-1}$ subarrays. This scheme uses a prime, P, larger than the dimension of the array as the number of memory modules. In addition, a skewing factor, s, is used. Specifically, $a_{i,j} \equiv is+j \pmod{P}$. A similar set of templates is accessible by the scheme proposed by Lawrie [Law75]. The set of templates here include the row template, the column template, diagonals and $\sqrt{n} \times \sqrt{n}$ rectangular subarrays. This scheme is not conflict-free for all of these templates but guarantees that each template instance can be accessed in no more than two memory cycles.

1.1.1 Skewing Schemes and Latin Squares

In 1989, Kim and Kumar [KK89] expressed the problem of skewing schemes in terms of *latin squares**. A latin square of order n is composed of the symbols 0 to $(n-1)$ such that no symbol appears more than once in any row or column [DK74]. Table 1.2 shows a latin square of order 3.

0	1	2
1	2	0
2	0	1

Table 1.2: Latin square of order 3

A *diagonal latin square** (Table 1.3) is a latin square such that no symbol appears more than once in either one of its two main diagonals [DK74].

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

Table 1.3: A diagonal latin square

A new type of latin square, a *perfect latin square**, was proposed by Kim and Kumar [KK89, HKK92] in 1989 (see Table 1.4 for an example). A perfect latin square of order n is a diagonal latin square such that no symbol appears more than once in any *main subsquare**. The order, n, must be a perfect square. A subsquare of order $\sqrt{n}$, $S_{i,j}$, is a main subsquare if its top left cell is (i,j) where $i \equiv 0 \bmod \sqrt{n}$ and $j \equiv 0 \bmod \sqrt{n}$ [HKK92].

0	3	6	1	4	7	2	5	8
2	5	8	0	3	6	1	4	7
1	4	7	2	5	8	0	3	6
3	6	0	4	7	1	5	8	2
5	8	2	3	6	0	4	7	1
4	7	1	5	8	2	3	6	0
6	0	3	7	1	4	8	2	5
8	2	5	6	0	3	7	1	4
7	1	4	8	2	5	6	0	3

Table 1.4: Perfect latin square

They also show that all subsquares of size $\sqrt{n}$ are accessible in two memory cycles when using perfect latin squares. When the perfect latin squares are of order 2^{2k} for k an integer, the address generation may be accomplished in constant time. In addition, they present skewing schemes for $3D$ arrays $(n \times n \times n)$.

Colbourn and Heinrich [CH92] used latin squares to access an $x \times y$ rectangular subarray. Specifically, there is an (x, y) *conflict-free latin square** for all $n = xy + p$

for $1 \leq p \leq y$. An (x, y) conflict-free latin square is a latin square in which no element appears more than once in any $x \times y$ subarray as well as the rows and columns. Since p may be equal to one, only one spare memory is required for conflict-free storage. They have also shown that if $x > y + 1$, there is an $\{x, y\}$ *conflict-free latin square** of order n for $n \geq xy + y$ [CH92]. An $\{x, y\}$ conflict-free latin square is a latin square in which no element appears more than once in any $x \times y$ **or** $y \times x$ subarrays as well as the rows and columns. If $2 \leq y < x < 2y$ they have shown that a $\{x, y\}$ conflict-free latin square of order n exists if $n \geq xy + (2y - x)(x - y) + 1$. This implies that if $x > y$ then a $\{x, y\}$ conflict-free latin square of order n exists if $n \geq xy + y$ [CH92]. They improved the lower bounds for the situation in which $x \not\equiv 1 \pmod{y}$, $x \geq 2y - 1$, and $x > y > 3$. In this situation, a $\{x, y\}$ conflict-free latin square of order n exists if $n > xy + y$. If $x \equiv 1 \pmod{y}$, then a $\{x, y\}$ conflict-free latin square of order n exists if $n = xy + y$.

Colbourn and Heinrich have also presented some results for *d-Manhattan latin squares**. A d-Manhattan latin square does not have two entries containing the same element whose *Manhattan distance** is less than $2d + 1$. The Manhattan distance between the entries (i, j) and (k, l) is $|i - k| + |j - l|$ [CH91]. It is shown that there exists a d-Manhattan latin square of order n where $n = 2d^2 + 2d + 1$ for all $d \geq 0$.

1.1.2 Skewing Schemes and Tilings

This section contains various theoretical limitations known for skewing schemes.

Shapiro has shown that a given template of size n it can be stored conflict-free in n memory modules if no two template instances involve the same array elements [Sha78]. This is equivalent to tiling the plane with the template. If one can tile the plane with the template, one can provide conflict-free access in n memory modules.

Shapiro shows that for a set of R templates:

1. if all templates are of size n,

2. each template may be stored conflict-free, and

3. each template has the same set of *designated elements** for the set of template instances involved in the tiling,

then this set of R templates may be stored conflict-free in n memory modules [Sha78]. The designated element or handle of Shapiro's template instance is the first element of the template instance when it is represented as a list of its elements [Sha78]. Thus, each template instance for each template can be defined by one element, the handle and a relationship of the other elements in the template instance to this handle. Each handle is a translate of the handle which defines the template. Informally as explained by Shapiro [Sha78], place each set of template instances involved in the tiling on a clear plastic overhead. Mark each handle with an asterisk. There is a valid skewing scheme for any set of templates which allow the asterisks to be overlaid. Recall that a skewing scheme only allows one template instance from the set of templates to be accessed without conflicts in any given memory cycle. One example can be seen in Figure 1.2. The element denoted with an asterisk is the handle. The clear rows to the left are template instances of partial rows from a partial row template. The shaded regions denote partial columns from a partial column template. Once again, this is equivalent to tiling the plane with a template and if more than one template is desired to be conflict-free, then the handle must match up for each individual tiling. If we can tile the plane subject to these restrictions, we can provide conflict-free access.

Additional theoretical results can be found in the monograph by Wijshoff [Wij89]. One result of particular interest is that concerning *block templates**. A $(p_1, v_1) \times$

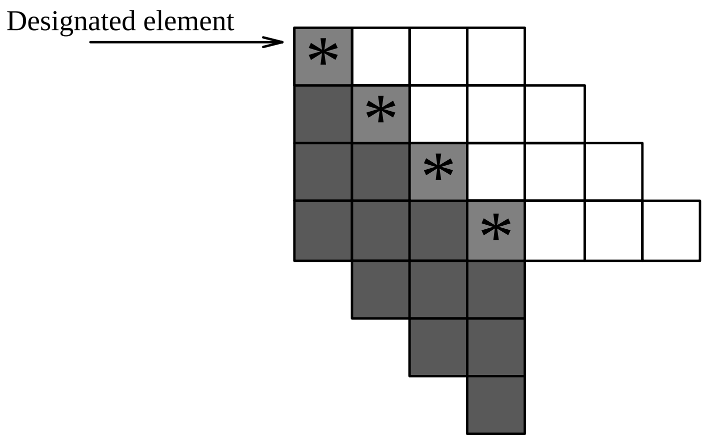

Figure 1.2: An example of Shapiro's handles

$(p_2, v_2) \times \ldots \times (p_d, v_d)$-block template B on a d-dimensional array is the set $\{(i_1v_1, i_2v_2, \ldots, i_dv_d) | 0 \le i_1 < p_1, 0 \le i_2 < p_2, \ldots, 0 \le i_d < p_d\}$ [Wij89]. Informally, the m^{th} tuple relates to the m^{th} dimension. Blocks in two dimensions are also referred to as rectangular subarrays. p_i indicates the number of elements chosen and v_i indicates one chooses every v_i^{th} element in i^{th} dimension for $1 \le i \le d$. See Figure 1.3 for an example. They have shown that one cannot store two different block templates which have equal stretch and equal size, n, in n memory modules [Wij89]. The stretch of a block defined above is $(v_1, v_2, \ldots v_d)$ [Wij89]. If the stretch is not specified it is assumed to be equal to one. When $v > 1$, the blocks are also referred to as stretch blocks.

1.2 Background on Interconnection Networks

The interconnection network, IN, allows the processors and the memory modules to communicate. These networks are also referred to as alignment and permutation networks [Sie85]. Section 1.2.1 reviews the design issues involved in an IN. Sec-

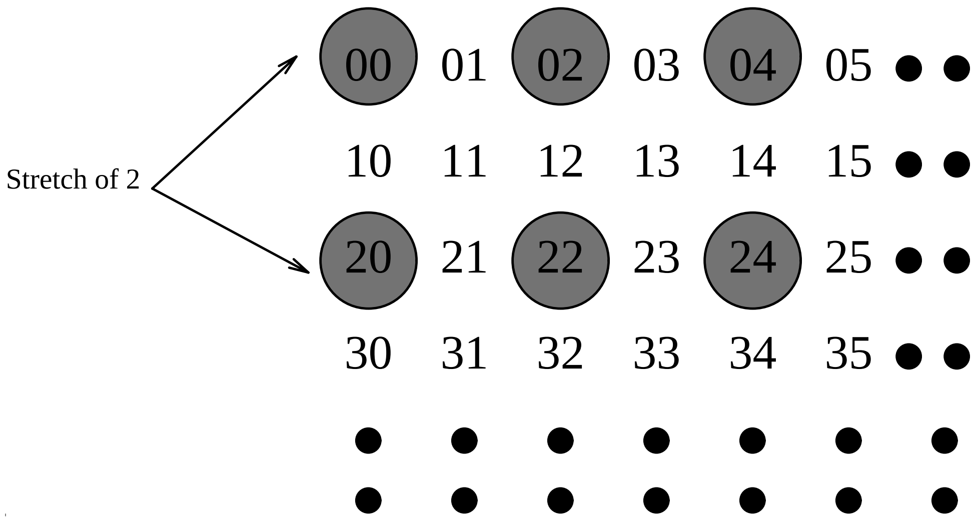

Figure 1.3: A $(2,2) \times (3,2)$ block

tion 1.2.2 presents various topologies and Section 1.2.3 presents various evaluation techniques for INs with respect to the needs of a skewing-scheme designer.

1.2.1 Design Issues of Interconnection Networks

There are various design issues concerning an IN [Roo91] [RP91] [Dan91]. Feng [Fen81] mentions the following four issues.

1. **Mode of Operation.** The type of communication may be synchronous and controlled by a system-wide clock or it may be asynchronous. In this thesis we are concerned with synchronous systems.

2. **Control Strategy.** The control for the routing may be distributed or centralized. If the routing is centralized it is dependent on a critical element, the controller. This may introduce delays and thus, distributed routing is generally used.

3. **Switching Methodology.** The two primary switching methodologies are virtual circuit and packet switching. In a virtual circuit the entire path between the source and destination is reserved. A reservation means that all links on this path are locked during communication and no one else may use them. In packet switching, a *packet* or specified fixed amount of information is sent on a free line in the direction of the destination. No specific route is reserved or specified. If a link is busy, the packet is either blocked or sent along another link. There are also variations and combinations of these two methodologies.

4. **Network Topology.** A network topology may be static or dynamic. A static topology cannot be reconfigured while dynamic topologies may. Dynamic INs are reconfigured during operation by their switches which allow various connections. Static links are predetermined at the time the network is constructed and will always connect the same links in the same manner. Dynamic INs are also referred to as reconfigureable switching networks [Dan91].

1.2.2 Topologies of Interconnection Networks

Since we are operating in a synchronous environment, we assume our INs operate in a synchronous manner. As previously stated, centralized routing is slow. It follows that most of the INs we present in this review use distributed routing. We now present a brief overview of various topologies. Many topologies can be shown to be equivalent. For more information see Siegel [Sie85].

Crossbar Interconnection Network

One of the simplest dynamic INs is the crossbar network (see Figure 1.4). A crossbar network of size $n \times n$ (connecting n memory modules with n processors) requires $O(n^2)$ switching elements. Therefore this network is quite expensive. The advantages of this network include 100% *Permutation capability** and a delay independent of the size of the network [Dan91].

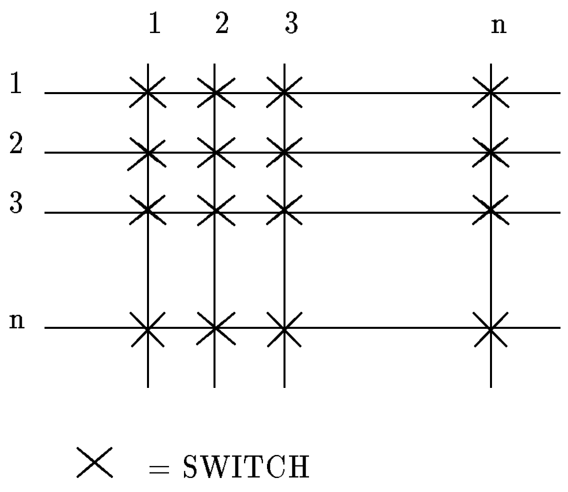

Figure 1.4: Crossbar interconnection network

Hypercube

A well known static IN is the hypercube. The hypercube is defined in terms of its dimension. A k-cube is a graph with 2^k vertices. These are also referred to as binary hypercubes [Dan91]. The vertices are numbered from 0 to $2^k - 1$ (see Figure 1.5). Two vertices in a binary hypercube are connected if and only if the binary representations of the vertices differ in exactly one bit [Roo91]. For example, vertices 0100 and 0101 are connected but 0100 and 0111 are not. The benefits of this topology include the fact that the distance between the source and the destination is equivalent to the number of bits in which they differ. This number is also known as the Hamming distance. The number of routing choices at any internal vertex along the route is equal to the Hamming distance between the current vertex and the destination vertex [Roo91].

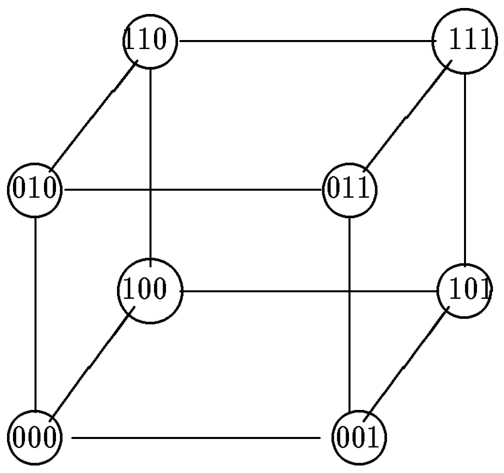

Figure 1.5: An order 3 hypercube network

Star Graphs

The star graph was proposed to help alleviate the problems associated with the high degree of the vertices that result in binary hypercubes [Roo91]. A k - star graph connects $K = k!$ vertices. Each vertex is labeled with a permutation on k elements. A vertex, p, is connected to another vertex, q, if and only if q's permutation is equivalent to p's with p's first element exchanged with any other single element. For example, vertex 1234 and 2134 are connected but 1234 and 1324 are not (see

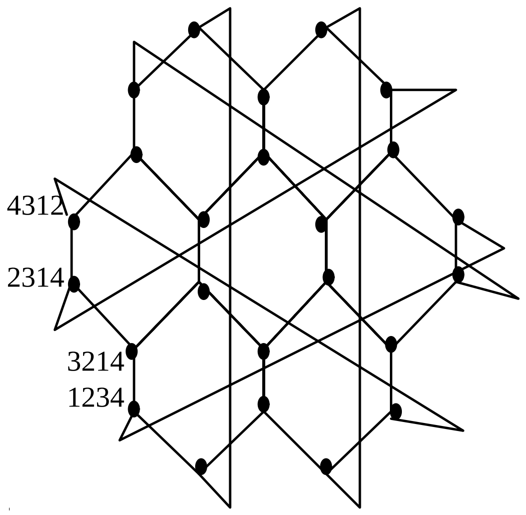

Figure 1.6: A 4-star graph

Figure 1.6 for a 4-star graph with some labelled vertices from [Roo91]).

Shuffle-Exchange Network

One of the most dynamic networks is the shuffle-exchange network. The switches in this type of network allow dynamic reconfiguration of the network [Dan91]. A simple example of a 2×2 switch can be seen in Figure 1.7. A 2×2 switch allows the inputs to proceed straight through, to be exchanged, or either of the inputs to be broadcast. The behaviour of the box is determined by the control bits. The network shuffles the data between the levels of switch boxes much the way one shuffles a deck of cards. A perfect shuffle is achieved if the 'higher' destinations are perfectly interleaved with the 'lower' destinations (see Figure 1.8). The other function, the exchange, allows communication with a destination whose address only differs in the lowest bit [Sie85]. Figure 1.9 is an example of a multi-stage shuffle-exchange network with eight inputs and outputs [Dan91]. An interesting feature of this network is that any source can find the destination by its address.

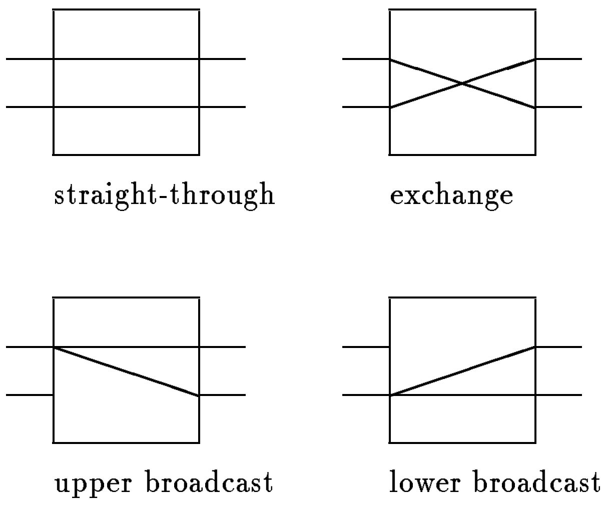

Figure 1.7: A 2×2 switch box

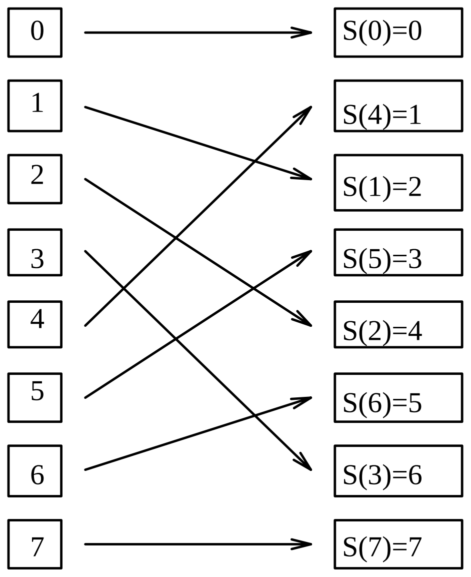

Figure 1.8: A perfect shuffle for eight elements

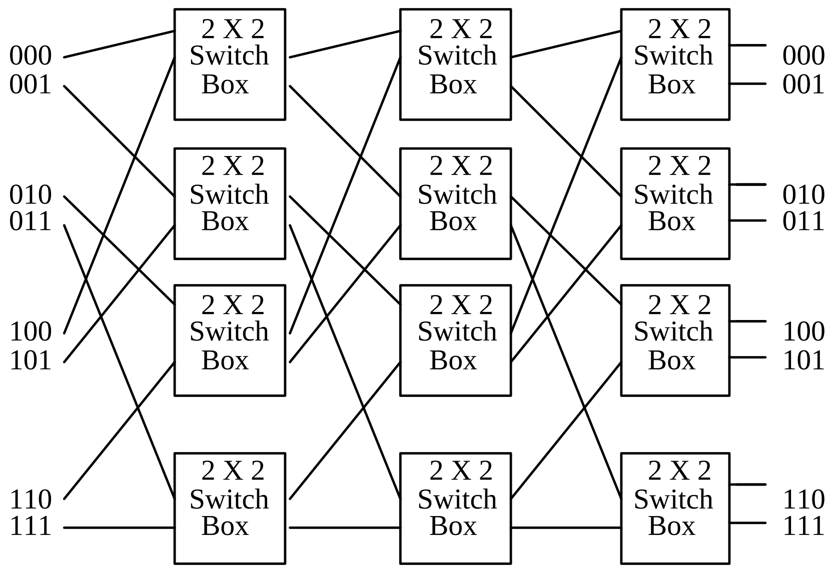

Figure 1.9: A multi-stage shuffle-exchange network

The highest bit determines the action to be taken at the first stage, the second-highest bit at the second stage, etc. A 1-bit indicates the bottom output and a 0-bit indicates the top output should be taken. Thus, the destination address can be used to set switch settings to route a message [Dan91]. Various other networks are based on the shuffle-exchange network. They vary in terms of the type of control strategy and the functions and size of the switching boxes [Dan91]. Examples include Benes networks [Ben65] [HKK92], the Omega networks [Law75] [Bro91], the Banyan Network [CM85] and the Theta Network [Lee88].

1.2.3 Evaluation of Interconnection Networks

There are various methods one may use to evaluate INs. Criteria may be deterministic and evaluate such characteristics as the number of paths in the network [MM80]. Alternatively, the criteria may be probabilistic. In this situation, one would be concerned with something such as the probability of blocking when a particular source and destination wish to communicate [MM80]. In this thesis, the evaluation criteria we are concerned with must be helpful to designers with a skewing scheme searching for an interconnection network. The designers will then use various criteria to find an IN to suit their needs.

The following set of desirable features for an IN were presented by Kim and Kumar [HKK92].

- The IN should be able to realize the required permutations between the processing elements and the memory modules (determined by the skewing scheme).
- The IN should provide fast routing. This includes network set-up time and network delay time. The maximum delay time should be minimized since the length of time for a communication cycle must accommodate the worst-case delay time.
- The cost, in terms of the number of logic gates or switches, and in terms of the total VLSI chip area, should be low.

We expand on the first point since it is the most important criterion to a designer with a skewing scheme. The required connections between the memory modules and the processors or the memory module/processor permutations are determined by the skewing scheme employed. Rooks [Roo91] [RP91] suggests a related criterion referred to as the permutation capability of the networks. More specifically, a network is viewed in terms of the percentage of all possible permutations that it can satisfy. It is desirable to have a network capable of all $n!$ permutations. A network is *rearrangeable** if it can realize all these permutations. Evaluating the satisfiability of all $n!$ permutations is not tractable on an n-node network [Roo91]. Thus, the permutation capability is frequently estimated. Szymanski [Szy89] observes a relationship between the blocking probability of a circuit and the permutation capability [Roo91]. While rearrangeable INs will always satisfy our permutation needs, what can be said about INs that are not rearrangeable?

There are various methods for increasing the permutation capability of an IN. Some of the methods *recirculate** data through a network, some use a *multi-stage** IN, some *dilate** the network, and some combine the aforementioned methods. Enhancing the permutation capability seems very beneficial. If the capability is increased to 100% and the network becomes rearrangeable, then we know we can realize all desired permutations. Suppose we have a high permutation capability but not 100%. What do we then know about the ability of the IN? How do we know that the permutations we need are represented in the permutation capability? Using only this measure, we do not know if this IN will satisfy our needs. This calls for additional measures in terms of the permutations the network can realize. Some classifications for sets of permutations can be found in [BR88, YL81, NS81, Len78]

Thus, the challenge to designers who do not wish to use a rearrangeable network is to determine what set of processor/memory module permutations they need. Ideally, this set can be realized by an existing IN. Additional consideration should be given to the class of permutations involving a larger number of memory modules than processors. That is, when $n > m$ (see Figure 1.1). It is possible that the templates may demand more than n memory modules in order to provide conflict-free access. This would involve permutation problems for an IN which connects n processors and $n + k$ memory modules for some positive integer k.

1.3 Overview of the Thesis

In the literature, the set of templates that are accommodated by existing skewing schemes contain a fixed number of templates. In contrast, the number of templates in the set of templates accommodated by skewing schemes in this thesis grow with respect to the maximum size of a template. Specifically, we examine the templates that provides conflict-free access to constant-perimeter rectangular subarrays. The number of templates is a function of the perimeter. For example, if the desired perimeter is 14, there are six templates to accommodate. If the desired perimeter is 32, there are 15 templates which need to be conflict-free. In general, for the perimeter, $\gamma = 2p$, there are $p - 1$ templates for a given constant perimeter. Furthermore, if you allow the rectangular subarrays to be stretched, each stretch produces a unique template, and the number of templates in the set is once again increased.

Thus, the most significant results can be found in Chapters 2 and 3. Chapter 2 presents bounds and constructions for all $(x-i, v) \times (y+i, v)$ rectangular subarrays of an $n \times n$ array for all nonnegative i, where v is the stretch of the subarray. These subarrays are a subset of those rectangular subarrays that have constant perimeter. The set of templates contains x distinct rectangular subarrays plus a template for each stretch allowed. The results that have implications for the complete set of constant-perimeter rectangular subarrays, and those that are stretched by v, where v is relatively prime to n, are presented in Chapter 3. The partial and complete constant-perimeter results use a mega-template which encapsulates the interactions of the set of constant-perimeter templates.

Chapter 4 examines a problem in which we want conflict-free access to all constant-area rectangular subarrays. In this chapter, we use the constant-perimeter results to obtain a lower bound on the number of memory modules needed for conflict-free storage. Furthermore, we relate the problem to that of the chromatic number of a graph and obtain an upper bound. An additional approach is proposed which involves the use of permutations for skewing schemes. Finally, computational results for constant-area rectangular subarrays are presented. Chapter 5 contains concluding remarks and areas for future research.

Chapter 2

Some Constant-Perimeter Subarrays

Squares and rectangular subarrays are natural templates to examine [KK89, HKK92, CH92]. In this chapter, we consider cases involving a subset of constant-perimeter rectangular subarrays, which are a type of block template. Let x and y be integers. We examine conflict-free access to all $(x - i, v) \times (y + i, v)$ rectangular subarrays of an $n \times n$ array for all nonnegative i. Let the perimeter of the rectangular subarray desired to be conflict-free be defined as $\gamma = 2(x + y)$. Thus, these templates represent a subset of all constant-perimeter rectangular subarrays. Squares that are conflict-free for all such subarrays are denoted by $(x, y)_{\mathrm{rcf},v}$*. If v is not specified it is assumed to be equal to one. Any $n \times n$ square on n symbols that is conflict-free for rows and columns is a well-studied combinatorial structure, a *latin square.* If the $(x, y)_{\mathrm{rcf},v}$ square is also latin, it is a *latin rectangular conflict-free square with stretch* v^* $(x, y)_{\mathrm{lrcf},v}$. As the perimeter desired increases, the skewing scheme needs to be conflict-free for a larger set of templates, all of varying shapes. There are two main reasons for interest in this problem. First, it is of theoretical interest as a prototypical problem in which the number of templates and their shape is a function of the desired perimeter. Secondly, it is of practical concern in certain navigational and layout problems [SD90], for example when accessing a portion of a two-dimensional map or layout corresponding to elements at a fixed horizontal and vertical rectilinear distance from a fixed point. Thus, these templates reflect the needs of algorithms and satisfy **Point 1**.

2.1 Preliminary Definitions and Theorems

To prove our bounds, we use a *mega-template** which encapsulates the essence of the interaction of the set of templates we want to be conflict-free. A mega-template is the set of elements which are forced to be distinct by the templates we wish to be conflict-free. In other words, every element in a mega-template is in a template instance with each other element. Furthermore, no element outside of the mega-template is in a template instance with every element of the mega-template.

The following definitions and lemmas are needed for the proofs of $(x,y)_{\mathrm{rcf},v}$ squares. The rectangular subarray $S_v(a,b:c,d)$, refers to the subarray of S with the rows a to $v(b-a)+a$ inclusive and the columns c to $v(d-c)+c$ inclusive, a $(b-a+1,v)\times(d-c+1,v)$ rectangular subarray. Furthermore, $S_v(a:c,d)$ is a partial row of row a from column c to $v(d-c)+c$ and $S_v(a,b:c)$ is the partial column of column c from row a to $v(b-a)+a$. A similar notation is used in [CH92]. The following definitions are needed to define a diamond which we use as a mega-template. In addition, we proceed with some fundamental theorems regarding the mega-template needed for the proofs of $(x,y)_{\mathrm{rcf},v}$ and $(x,y)_{\mathrm{lrcf},v}$ squares.

- The *base* is $S_v(0, x-1 : 0, y-1)$.
- A *stair* at level L on the *West* side is $S_v(L, x-1-L : -L) : 0 < L \leq \lfloor \frac{x-1}{2} \rfloor$.
- A *stair* at level L on the *East* side is $S_v(L, x-1-L : y-1+L) : 0 < L \leq \lfloor \frac{x-1}{2} \rfloor$.
- An $(x, y)_v$*-cut diamond* is a template that is the union of the base, the West, and East stairs (see Figure 2.1) or any translate in an $n \times n$ square with wrapping.

Recall that a square is *cyclic* if and only if it can be defined by a linear skewing scheme.

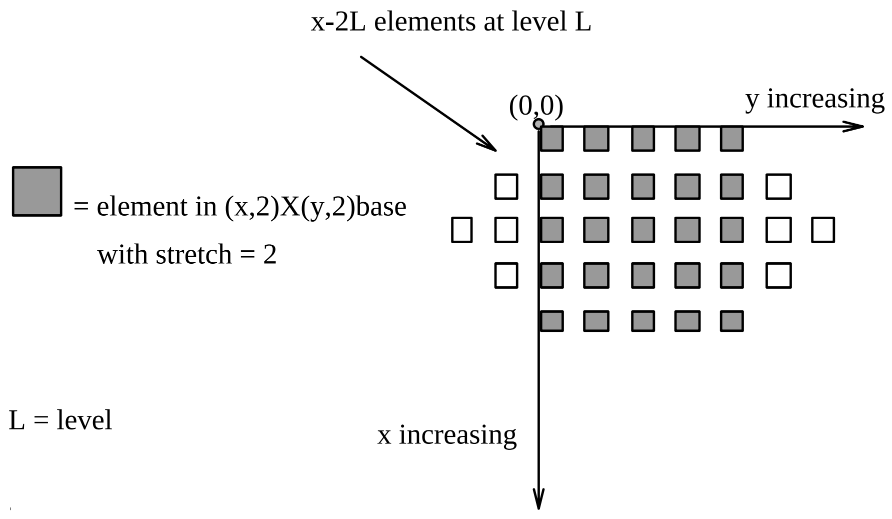

Figure 2.1: An $(x, y)_2$-cut diamond

Theorem 2.1.1 *Let M be a square. The following two conditions are equivalent.*

1. *Each $(x, y)_v$-cut diamond is conflict-free in M and if x is even, each $(1, v) \times (x + y - 1, v)$ subarray is also conflict-free in M.*

2. *For each* $0 \le i < x$, *each* $(x-i,v)\times(y+i,v)$ *template instance is conflict-free in* M.

Proof: (1) $\Rightarrow$ (2): Assume that (2) is false, so that some $(x-i,v)\times(y+i,v)$ subarray does not contain distinct symbols. The definition of the $(x,y)_v$-cut diamond ensures that every $(x-i,v)\times(y+i,v)$ subarray is contained in some $(x,y)_v$-cut diamond or is a $(1,v)\times(x+y-1,v)$ subarray. Thus, (1) does not hold if (2) does not and hence (1) $\Rightarrow$ (2).

(2) $\Rightarrow$ (1): Assume first that some $(x,y)_v$-cut diamond does not contain distinct elements. Let the two equal entities be at horizontal distance vh and vertical distance vk from one another. Then $vk \le v(x-1)$ since the cut diamond has height vx, and $v(k+h) \le v(x+y-2)$ for any two entries of the cut diamond. Now since $v(k+h) \le v(x+y-2)$ and $vk \le v(x-1)$, and some $(k+1,v)\times(h-1,v)$ rectangular subarray is not conflict-free, and thus, some $(x-i,v)\times(y+i,v)$ template instance is not conflict-free and (2) does not hold. Similarly if some $(1,v)\times(x+y-1,v)$ subarray contains a conflict, (2) does not hold. Thus (2) $\Rightarrow$ (1). □

Corollary 2.1.2 *Let* M *be a cyclic square. The following two conditions are equivalent.*

1. *Some* $(x,y)_v$*-cut diamond is conflict-free in* M *and if* x *is even, some* $(1,v)\times(x+y-1,v)$ *subarray is also conflict-free in* M.

2. *For each* $0 \le i < x$, *some* $(x-i,v)\times(y+i,v)$ *template instance is conflict-free in* M.

Proof: One instance of a template is conflict-free if and only if all instances in M are conflict-free. This follows immediately by the cyclic construction. Thus, the proof is complete by Theorem 2.1.1. □

We now proceed to count the elements contained in the $(x,y)_v$-cut diamond. We use a convenient notation suggested by Knuth [Knu92], denoting by $[x \text{ odd}]$, the function whose value is one if x is odd, and zero otherwise. The stairs can be examined in two separate cases. Let z be the length of the side of the base upon which the stairs are built. Let $n(z)$ be the number of elements in the set of stairs on side z.

Lemma 2.1.3 *$n(z) = \frac{1}{4}z^2 - \frac{1}{2}z + \frac{1}{4}$ if z is odd.*

Proof: For odd z, the number of elements in the set of stairs is:

$$\begin{aligned} n(z) &= \sum_{i=1}^{\frac{z-1}{2}} (z-2i) \\ &= \frac{z-1}{2} + 2\sum_{i=0}^{\frac{z-3}{2}} i \\ &= \frac{(z-1)^2}{4} \\ &= \frac{1}{4}z^2 - \frac{1}{2}z + \frac{1}{4}. \end{aligned}$$

□

Lemma 2.1.4 *$n(z) = \frac{1}{4}z^2 - \frac{1}{2}z$ if z is even.*

Proof: If z is even, then the number of elements in the set of stairs is:

$$\begin{aligned} n(z) = \sum_{i=1}^{\frac{z}{2}}(z-2i) \\ &= 2\sum_{i=0}^{\frac{z-2}{2}}(i) \\ &= (\frac{z-2}{2})(\frac{z}{2}) \\ &= \frac{1}{4}z^2 - \frac{1}{2}z. \end{aligned}$$

□

Lemma 2.1.5 *The $(x,y)_v$-cut diamond contains $xy + \frac{1}{2}x^2 - x + \frac{1}{2}[x \text{ odd}]$ elements.*

Proof: The number of elements in an $(x,y)_v$-cut diamond is the sum of the base and the two sets of stairs. By Lemma 2.1.3, $n(z) = \frac{1}{4}z^2 - \frac{1}{2}z + \frac{1}{4}$ if z is odd. By Lemma 2.1.4, $n(z) = \frac{1}{4}z^2 - \frac{1}{2}z$ if z is even. Thus, the number of elements in the $(x,y)_v$-cut diamond is $xy + 2n(x) = xy + \frac{1}{2}x^2 - x + \frac{1}{2}[x \text{ odd}]$. □

Define a (t,s,r,v) *shift-box** by $S_v(1,t:1,s) \cup S_v(t+1:1,r)$.

Theorem 2.1.6 *If W is a (t,s,r,v) shift-box then W contains distinct elements in c where c is an $n \times n$ cyclic square defined by $(a_{i,j}), a_{i,j} \equiv (i-1)s + j \pmod{n}$, $st + r = n$ and v is relatively prime to n.*

Proof: Recall the skewing function for a cyclic square. Successive array locations in a row are mapped to successive integers modulo n. If v equals one, the first row in W is mapped to the first s integers. The first element in the second row is mapped to the integer $s+1$. Thus, the first t rows are mapped to the first st integers. Thus, there are r integers remaining which are mapped into row $t+1$. Since $n = st + r$ all the array locations in the shift-box are mapped to distinct integers which represent the memory modules. If v is greater than one but relatively prime to n, the array locations in the first row of the (t, s, r, v) shift-box are mapped to $1, v+1, 2v+1, \ldots, s(v-1)+1$. The second row is mapped to $v(s)+1$. If v is relatively prime to n, the mapping will cycle through all the integers since the mapping is modulo n. Thus all the elements in the shift-box are mapped to distinct integers. □

Lemma 2.1.7 *If $a_{i,j} \equiv a_{i+b,j+c} \pmod{n}$, then $a_{i,j} \equiv a_{i+vb,j+vc} \pmod{n}$ in a cyclic square.*

Proof: If $a_{i,j} \equiv a_{i+b,j+c} \pmod{n}$, then according to the cyclic construction, $bq + cr \equiv 0 \pmod{n}$. Thus, $v(bq+cr) \equiv 0 \pmod{n} \equiv vbq + vcr \pmod{n}$. This implies that $a_{i,j} \equiv a_{i+vb,j+vc} \pmod{n}$. □

Observation 2.1.8 *If a skewing scheme is conflict-free for a template, then the transpose of that skewing scheme is conflict-free for the transpose of that template. Furthermore, observe that the transpose of a cyclic skewing scheme is cyclic.*

2.2 Bounds and Constructions

We now proceed with the proofs for $(x, y)_{\mathrm{rcf},v}$ squares. The general strategy is to exhibit a mega-template that is easily seen to be conflict-free, and for which the absence of conflicts implies the whole square is conflict-free for the set of templates. Rather than directly show that the $(x,y)_v$-cut diamond is conflict-free within the square, we insert an intermediate step involving a box, usually a shift-box. The primary remaining step is to show the relationship between the shift-box and the $(x,y)_v$-cut diamond.

Theorem 2.2.1 *If an $(x,y)_{rcf,v}$ or an $(x,y)_{lrcf,v}$ square of order n exists, $n \geq xy + \frac{1}{2}x^2 - x + \frac{1}{2}[x \text{ odd}]$.*

Proof: By Theorem 2.1.1 all of the elements in the $(x,y)_v$-cut diamond must be distinct. By Lemma 2.1.5, $n \geq xy + \frac{1}{2}x^2 - x + \frac{1}{2}[x \text{ odd}]$. □

Let $n = xy + \frac{1}{2}x^2 - x + \frac{1}{2}$, and define an $n \times n$ array $C = (a_{i,j})$, $a_{i,j} \equiv (i-1)(2y+x-2)+j \pmod{n}$.

Theorem 2.2.2 *For all odd x and v relatively prime to n, C is an $(x,y)_{rcf,v}$ and an $(x,y)_{lrcf,v}$ square.*

The proof is based on the following lemma.

Lemma 2.2.3 *If a $(\frac{1}{2}x - \frac{1}{2}, 2y+x-2, y+\frac{1}{2}x-\frac{1}{2}, v)$ shift-box contains distinct elements in C, then the $(x,y)_v$-cut diamond contains distinct elements for odd x and v relatively prime to n.*

Proof: First we examine the $(x,y)_1$-cut diamond. The upper left-hand corner of the shift-box is placed over the North-West corner of the base of the $(x,y)_1$-cut diamond as displayed in Figure 2.2. Now, $a_{i,j} \equiv a_{i-\frac{1}{2}x-\frac{1}{2},\, j+y+\frac{1}{2}x-\frac{3}{2}} \pmod{n}$ since

$(-\frac{1}{2}x - \frac{1}{2})(2y + x - 2) + y + \frac{1}{2}x - \frac{3}{2} \equiv -xy - \frac{1}{2}x^2 + x - \frac{1}{2} \equiv 0 \pmod{n}$ and the element is the same. Thus, rows $(\frac{1}{2}x + \frac{3}{2})$ through x are shifted up $(\frac{1}{2}x + \frac{1}{2})$ rows and over $(y + \frac{1}{2}x - \frac{3}{2})$ columns to the left as displayed in Figure 2.3. Now, $a_{i,j} \equiv a_{i-1,j+2y+x-2} \pmod{n}$ since $-1(2y + x - 2) + 2y + x - 2 \equiv 0 \pmod{n}$ and the element is the same. Thus, the final shift involves all the elements from rows 2 through $(\frac{1}{2}x + \frac{1}{2})$ not contained in the shift-box which are shifted up one row and over $(2y + x - 2)$ columns. This shift is displayed in Figure 2.4. Finally, by

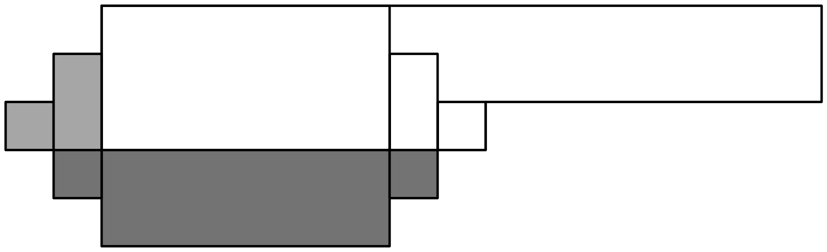

Figure 2.2: Shift-box placed over $(x, y)_1$-cut diamond

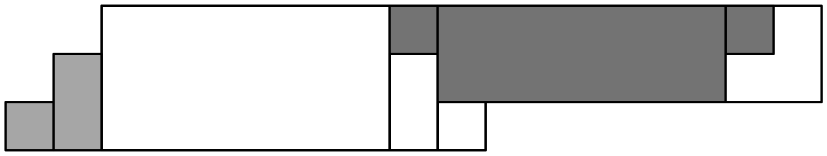

Figure 2.3: Shifting rows $\frac{1}{2}x + \frac{3}{2}$ through x to shift-box

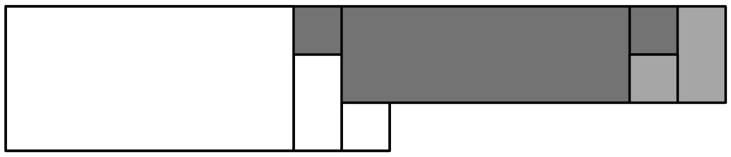

Figure 2.4: Shifting rows 2 through $\frac{1}{2}x + \frac{1}{2}$ to shift-box

Lemma 2.1.7, the above shifts are valid for $(x, y)_v$-cut diamonds. □

Proof of Theorem 2.2.2: By Corollary 2.1.2, if C is conflict-free for some $(x,y)_v$-cut diamond, then it is a $(x,y)_{\mathrm{rcf},v}$ square. By Lemma 2.2.3, an $(x,y)_v$-cut diamond contains the same elements as the shift-box. Finally, by Lemma 2.1.6 the shift-box contains distinct elements and C is an $(x,y)_{\mathrm{rcf},v}$ square for odd x and v relatively prime to n.

Finally, we need to show the shift is relatively prime to n when x is odd.

$$\begin{aligned}
&gcd(xy + \frac{1}{2}x^2 - x + \frac{1}{2}, 2y + x - 2)\\
&= gcd((xy + \frac{1}{2}x^2 - x + \frac{1}{2}) - (\frac{x}{2} - \frac{1}{2})(2y + x - 2), 2y + x - 2)\\
&= gcd(2y + x - 2, y + \frac{1}{2}x - \frac{1}{2})\\
&= 1
\end{aligned}$$

Thus, C is an $(x,y)_{\mathrm{lrcf},v}$ square. □

Now we treat the case when x is even, and y is any integer. Let $n = xy + \frac{1}{2}x^2 - x$. For the construction to be presented, for positive integers a and b, we let $Q(a,b)$ and $R(a,b)$ be the quotient and remainder, respectively, when a is divided by b. Now we write $e = \frac{x}{2}$, and define an $n \times n$ square $D = (d_{i,j})$ as follows: $d_{i,j} = R(i,e)(y+e-1) + R(j, y+e-1) + e(y+e-1)[(Q(i,e) + Q(j, y+e-1))\ \mathrm{odd}]$ An example is given in Figure 2.5. Note the repeated pattern of $(e) \times (y+e-1)$ rectangular subarrays.

Theorem 2.2.4 *D is an $(x,y)_{rcf,1}$ square for even x and $n = xy + \frac{1}{2}x^2 - x$.*

The proof is based on the following lemmas.

0	1	2	3	4	5	12	13	14	15	16	17	0	1	2	3	4	5
6	7	8	9	10	11	18	19	20	21	22	23	6	7	8	9	10	11
12	13	14	15	16	17	0	1	2	3	4	5	12	13	14	15	16	17
18	19	20	21	22	23	6	7	8	9	10	11	18	19	20	21	22	23
0	1	2	3	4	5	12	13	14	15	16	17	0	1	2	3	4	5
6	7	8	9	10	11	18	19	20	21	22	23	6	7	8	9	10	11
12	13	14	15	16	17	0	1	2	3	4	5	12	13	14	15	16	17
18	19	20	21	22	23	6	7	8	9	10	11	18	19	20	21	22	23
0	1	2	3	4	5	12	13	14	15	16	17	0	1	2	3	4	5
6	7	8	9	10	11	18	19	20	21	22	23	6	7	8	9	10	11
12	13	14	15	16	17	0	1	2	3	4	5	12	13	14	15	16	17
18	19	20	21	22	23	6	7	8	9	10	11	18	19	20	21	22	23
0	1	2	3	4	5	12	13	14	15	16	17	0	1	2	3	4	5
6	7	8	9	10	11	18	19	20	21	22	23	6	7	8	9	10	11
12	13	14	15	16	17	0	1	2	3	4	5	12	13	14	15	16	17
18	19	20	21	22	23	6	7	8	9	10	11	18	19	20	21	22	23
0	1	2	3	4	5	12	13	14	15	16	17	0	1	2	3	4	5
6	7	8	9	10	11	18	19	20	21	22	23	6	7	8	9	10	11

Figure 2.5: Example of D square with $x = 4, y = 5, n = 18$

Lemma 2.2.5 *If a $(2e, y+e-1, 0, 1)$ shift-box contains distinct elements in D, then the $(x, y)_1$-cut diamond contains distinct elements for even x, where e is an integer.*

Observe that this shift-box is just a $(2e, 1) \times (y+e-1, 1)$ rectangular subarray. *Proof*: First, the upper left-hand corner of the shift-box is placed over the North-West corner of the base in the $(x, y)_1$-cut diamond as displayed in Figure 2.6. Next, rows 2 through e not contained in the shift-box are shifted over $(y+e-1)$ columns and down e rows as displayed in Figure 2.7. The remainders remain unchanged, and the quotients each increase by 1. Thus the evaluation of $[(Q(i,e)+Q(j,y+e-1))$ odd$]$ remains unchanged and we have the same element. Finally, rows $(e+1)$ through $(2e-1)$ not contained in the shift-box are shifted up e rows and over $(y+e-1)$ columns. Once again, the remainders remain unchanged and the value of $Q(i,e)$ decreases by one and the value of $Q(j, y+e-1)$ increases by one: thus, the evaluation of $[(Q(i,e)+Q(j,y+e-1))$ odd$]$ remains unchanged and we have the same element. □

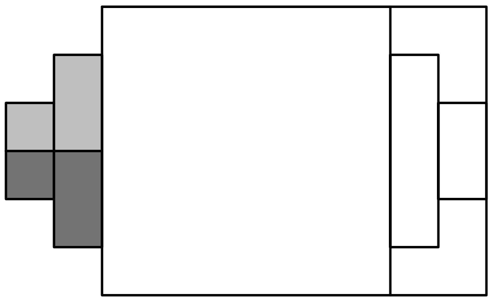
Figure 2.6: Shift-box placed over $(x, y)_1$-cut diamond

Lemma 2.2.6 *Each $(2e, y+e-1, 0, 1)$ shift-box contains distinct elements in D where e is an integer.*

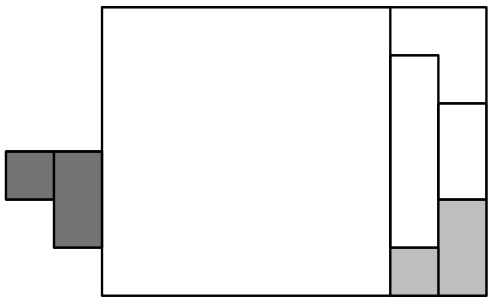

Figure 2.7: Shifting rows 2 through e to shift-box

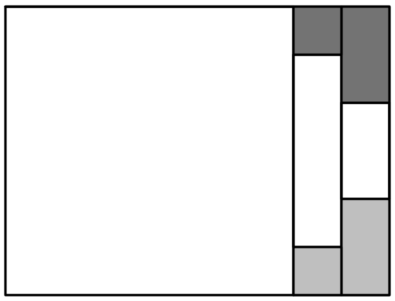

Figure 2.8: Shifting rows $e+1$ through $2e$ to shift-box

Proof: Suppose that two entries of D in some $(2e) \times (y+e-1)$ rectangular subarrays are the same. Let (s_1, t_1) and (s_2, t_2) be the indices of two such entries. By the definition of D, $R(s_1, e)(y+e-1) + R(t_1, y+e-1) + e(y+e-1)[(Q(s_1, e) + Q(t_1, y+e-1)) \text{ odd}] = R(s_2, e)(y+e-1) + R(t_2, y+e-1) + e(y+e-1)[(Q(s_2, e) + Q(t_2, y+e-1)) \text{ odd}]$. Considering this equation modulo $y+e-1$, we find that $R(t_1, y+e-1) = R(t_2, y+e-1)$. Thus, t_1 and t_2 differ by a multiple of $y+e-1$. However, since the two entries cannot be $y+e-1$ or more columns apart, we conclude that $t_1 = t_2$. Thus, $R(s_1, e) + e([(Q(s_1, e) + Q(t_1, y+e-1) \text{ odd}]) = R(s_2, e) + e([(Q(s_2, e) + Q(t_1, y+e-1)) \text{ odd}])$. That implies that $R(s_1, e) = R(s_2, e)$ and $Q(s_1, e) \equiv Q(s_2, e) \pmod 2$. If follows that $s_1 \equiv s_2 \bmod 2e$. Since the two entries appear fewer than $2e$ rows apart, it follows that $s_1 = s_2$.

Thus, any two equal entries in a rectangular subarray in fact occur at the same position, so each rectangle is conflict-free. $\Box$

Proof of Theorem 2.2.4: By Theorem 2.1.1, D contains distinct elements if each $(x,y)_1$-cut diamond contains distinct elements. By Lemma 2.2.5, a $(x,y)_1$-cut diamond contains distinct elements if each shift-box contains distinct elements. Finally, by Lemma 2.2.6 each shift-box contains distinct elements and D is an $(x,y)_{\mathrm{rcf}}$ square. $\Box$

Theorem 2.2.7 *If an $(x,y)_{lrcf,v}$ square of order n exists for even x, $n \geq xy + \frac{1}{2}x^2 - x + 1$.*

Proof: By Theorem 2.2.1, $n \geq xy + \frac{1}{2}x^2 - x$ for even x. Thus, we only need to show that $n \neq xy + \frac{1}{2}x^2 - x$. Shapiro shows in general that a template tiles a plane if and only if a skewing scheme exists [Sha78]. Thus, there must be a tiling of the plane with an $(x,y)_v$-cut diamond for even p that is conflict-free for rows and columns if there is to be an $(x,y)_{\mathrm{lrcf},v}$ square. By Theorem 2.2.4, we know there is a tiling that is conflict-free for all $(x-i,v) \times (y+i,v)$ rectangular subarrays. One can examine each possible tiling by picking a single external element of the $(x,y)_v$-cut diamond and attempting to place it next to the external elements of another copy of an $(x,y)_v$-cut diamond. This case analysis shows that this is the only tiling. Since this tiling is not conflict-free for rows and columns $n \geq xy + \frac{1}{2}x^2 - x + 1$. $\Box$

Let $n = 24e^2 + 8ed + 2$ and $x = 4e$, $y = 4e + 2d + 1$. Define an $n \times n$ array $E = (a_{i,j})$, $a_{i,j} \equiv (i-1)(12e^2 + 4ed - 6e - 2d + 1) + j \pmod{n}$ where e and d are integers.

Theorem 2.2.8 *E is an $(x,y)_{lrcf,v}$ square for $x = 4e$, $y = 4e + 2d + 1$ and v relatively prime to n, where e and d are integers.*

The proof is based on the following lemma.

An *$(x,y)_v$-cut zircon* is the $(x,y)_v$-cut diamond for $x = 4e, y = 4e + 2d + 1$, together with $S_v(1 : y+1) \cup S_v(\frac{x}{2} + 1 : \frac{x}{2} + y)$ which is one element added onto the North side of the first level of East stairs and another element added onto the far-East set of stairs as depicted in Figure 2.9.

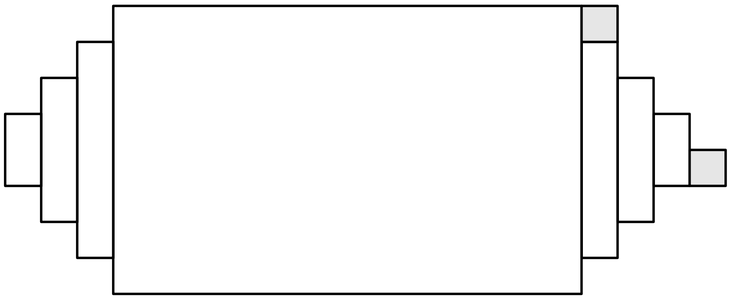

Figure 2.9: The $(x,y)_v$-cut zircon

Define an xy-box to be $S_v(1, \frac{x}{2}+1 : 1, \frac{x}{2}+y) \cup S_v(\frac{x}{2}+2, x : 1, \frac{x}{2}+y-2)$.

Lemma 2.2.9 *If the xy-box contains distinct elements in E, then the $(x,y)_v$-cut zircon contains distinct elements for $x = 4e$, $y = 4e + 2d + 1$, where e and d are integers.*

Proof: First we examine the $(x,y)_1$-cut zircon. Place the xy-box over the North-West corner of the base of the $(x,y)_1$-cut zircon as depicted in Figure 2.10. Now, $a_{i,j} \equiv a_{i+2e+1,j+6e+2d-1} \pmod{n}$ since $(2e+1)(12e^2+4ed-6e-2d+1)+6e+2d-1 \equiv 24e^3 + 8e^2d + 2e \equiv 0 \pmod{n}$ and the two elements are the same. Thus, those elements in rows 2 through $(2e-1)$ not contained in the xy-box are shifted down

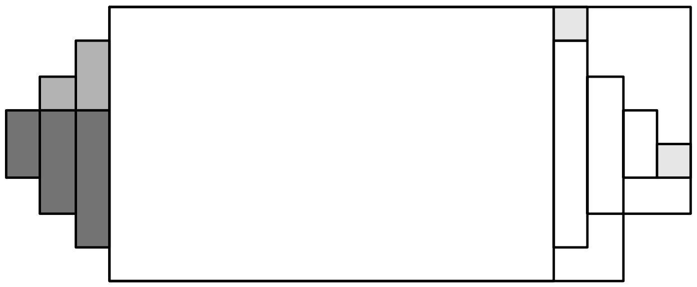

Figure 2.10: xy-box placed over the $(x, y)_1$-cut zircon

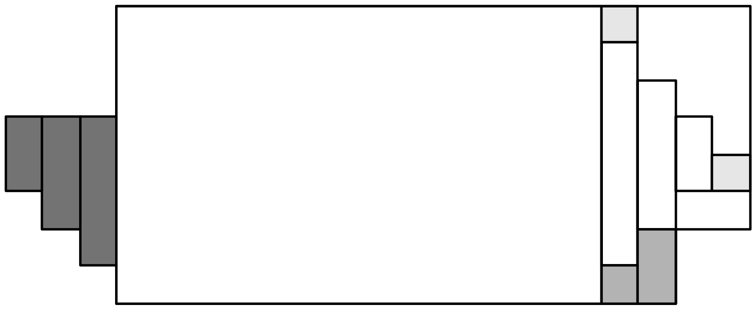

Figure 2.11: Shifting rows 2 through $2e - 1$ to xy-box

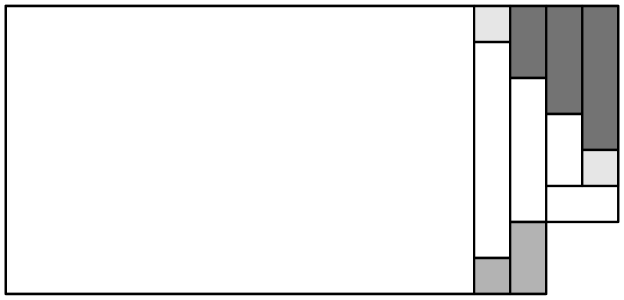

Figure 2.12: Shifting rows $2e$ through $4e - 1$ to xy-box

$(2e+1)$ and over $(6e+2d-1)$ columns as depicted in Figure 2.11. Next, $a_{i,j} \equiv a_{i-2e+1,j+6e+2d+1} \pmod{n}$ since $(-2e+1)(12e^2+4ed-6e-2d+1)+6e+2d+1 \equiv -24e^3-8e^2d-2e+24e^2+8ed+2 \equiv 0 \pmod{n}$. Thus, the elements in rows $(2e)$ through $(4e-1)$ not contained in the xy-box are shifted up $(2e-1)$ rows and over $(6e+2d+1)$ columns. This is depicted in Figure 2.12. Finally, by Lemma 2.1.7, the above shifts are valid for the $(x,y)_v$-cut zircon. □

Lemma 2.2.10 *The xy-box in E contains distinct elements for $x = 4e, y = 4e + 2d + 1$, where e and d are integers.*

Proof: According to the construction, each array element is mapped to an integer which represents the memory module in which it is stored. The sequential mapping of elements from array locations in row b are followed by the mapping of those elements in row $b + 2e - 1 \pmod{4e}$. If $b \leq 2e+1$, the last integer mapped into row b is $(b-1)s + 6e + 2d + 1 \pmod{n}$ where $s = 12e^2 + 4ed - 6e - 2d + 1$ and $n = 24e^2 + 8ed + 2$. This is equivalent to $bs - 12e^2 - 4ed + 4d + 12e \pmod{n}$. The integer mapped to the location preceding the first element in row $(b + 2e - 1)$ is $(b + 2e - 2)s \pmod{n}$.

$$\begin{aligned}
&\equiv bs + 24e^3 + 8e^2d - 4ed - 12e^2 + 2e - 2s \pmod{n} \\
&\equiv bs - 4ed - 12e^2 - 24e^2 - 8ed + 4d + 12e - 2 \pmod{n} \\
&\equiv bs - 4ed - 12e^2 + 4d + 12e \pmod{n}.
\end{aligned}$$

If $b = 2e + 1 + a, a > 0$, the value of the last element's mapping in the row is $(2e+a)s+6e+2d-1 \pmod{n} \equiv as-4ed-12e^2+6e+2d-1 \pmod{n}$. The next row according to the mapping defined above gets mapped to $2e+1+a+2e-1 \equiv a \pmod{4e}$. Thus, the integer that the element preceding the first element in the next sequential mapping row is $(a-1)s \pmod{n} \equiv as - 12e^2 - 4ed + 2d + 6e - 1 \pmod{n}$. Finally, since $2e-1$ is relatively prime to $4e$, all the rows are included in this cyclic mapping of the elements. $\square$

Proof of Theorem 2.2.8: By Corollary 2.1.2 if E is conflict-free for some $(x,y)_v$-cut diamond and for some $(1,v)\times(x+y-1,v)$ subarray, then it is a $(x,y)_{\mathrm{rcf},v}$ square. The $(x,y)_v$-cut zircon contains the $(x,y)_v$-cut diamond and thus the statement holds. Since

$$\begin{aligned} &gcd(24e^2+8ed+2, 12e^2+4ed-6e-2d+1) \\ &\quad= gcd(12e^2+4ed+1, 12e^2+4ed-6e-2d+1) \\ &\quad= gcd(6e+2d, 12e^2+4ed+1) \\ &\quad= 1, \end{aligned}$$

the square is latin and thus conflict-free for all $(1,v)\times(x+y-1,v)$ subarrays. By Lemma 2.2.9 an $(x,y)_v$-cut zircon is conflict-free if the shift-box is conflict-free. Finally, by Lemma 2.2.10 all of the elements in the shift box are distinct and E is a $(x,y)_{\mathrm{lrcf},v}$ square. $\square$

Let $n = 24e^2+8ed-2e+1$ and $x = 4e$, $y = 4e+2d$. Define an $n\times n$ array $F \equiv (a_{i,j}), a_{i,j} \equiv (j-1)(4e)+(i) \pmod{n}$ where e and d are integers.

Theorem 2.2.11 *F is an $(x,y)_{lrcf,v}$ square for $x = 4e$, $y = 4e+2d$ and v relatively prime to n, where e and d are integers.*

The proof is based on the following lemma.

The $(x,y)_v$-*cut sapphire* is the $(x,y)_v$-cut diamond for $x = 4e, y = 4e + 2d$, together with $S_v(1 : y+1) \cup S_v(\frac{x}{2}+1 : \frac{x}{2}+y) \cup S_v(x-L+1 : y+L)$ for $1 \le L \le \frac{x}{2}-1$ (each level of stairs on the East side of the diamond). In other words, there is one element added onto the North side of the first level of East stairs and one element added onto the South-East side of the East set of stairs. The remaining $\frac{x}{2}-1$ elements are added onto the South side of each East stair as depicted in Figure 2.13.

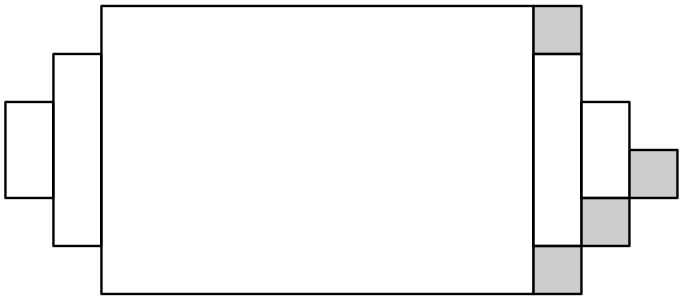

Figure 2.13: The $(x,y)_v$-cut sapphire

Here we use the transpose of a $(6e+2d-1, 4e, 2e+1, v)$ shift-box where e and d are integers.

Lemma 2.2.12 *If the transposed* $(6e+2d-1, 4e, 2e+1, v)$ *shift-box contains distinct elements F, then the* $(x,y)_v$*-cut sapphire contains distinct elements for* $x = 4e$, $y = 4e+2d$ *and* v *relatively prime to* n*, where* e *and* d *are integers.*

Proof: First we examine the $(x,y)_1$-cut sapphire. Place the transposed shift-box over the North-West corner of the base of the $(x,y)_1$-cut sapphire as depicted in Figure 2.14. Now, $a_{i,j} \equiv a_{i+2e+1,j+6e+2d-1} \pmod n$ since $(6e+2d-1)(4e) + 2e+1 \equiv 24e^2 + 8ed - 2e + 1 \equiv 0 \pmod n$ so the symbol is the same. Thus,

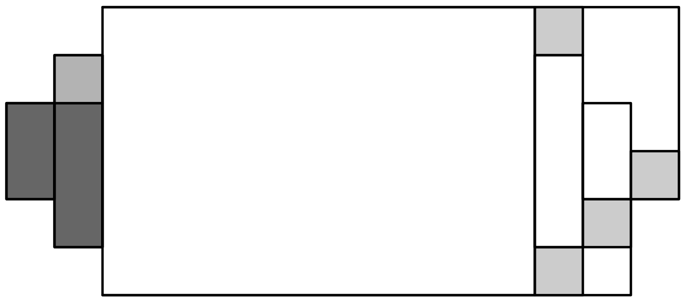

Figure 2.14: Transposed shift-box placed over the $(x, y)_1$-cut sapphire

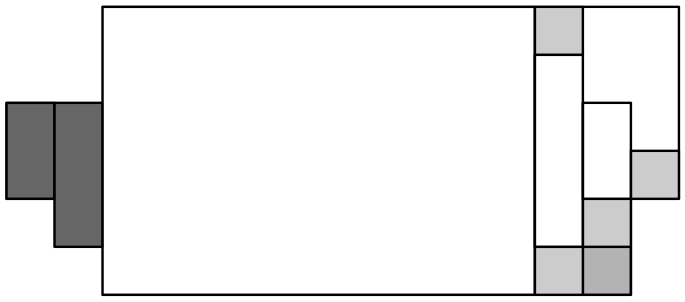

Figure 2.15: Shifting rows 2 through $2e - 1$ to transposed shift-box

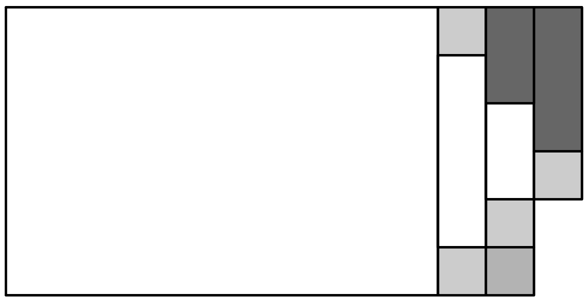

Figure 2.16: Shifting rows $2e$ through $4e - 1$ to transposed shift-box

those elements in rows 2 through $(2e-1)$ not contained in the transposed shift-box are shifted down $(2e+1)$ to rows $(2e+3)$ through $4e$ and over $(6e+2d-1)$ columns as depicted in Figure 2.15. Finally, $a_{i,j} \equiv a_{i-2e+1,j+6e+2d} \pmod{n}$ since $(6e+2d)(4e)+(-2e+1) \equiv 24e^2+8ed-2e+1 \equiv 0 \pmod{n}$ so the symbol is the same. Thus, the elements in rows $(2e)$ through $(4e-1)$ not contained in the transposed shift-box are shifted up $(2e-1)$ rows to rows 1 through $2e$ and over $(6e+2d)$ columns. This is depicted in Figure 2.16. Finally, the above shifts are valid for the $(x,y)_v$-cut sapphire by Lemma 2.1.7. $\square$

Proof of Theorem 2.2.11: By Corollary 2.1.2 if F is conflict-free for some $(x,y)_v$-cut diamond and for some $(1,v)\times(x+y-1,v)$ subarray, then it is an $(x,y)_{\mathrm{rcf},v}$ square. The $(x,y)_v$-cut sapphire contains the $(x,y)_v$-cut diamond and thus the statement holds. Since

$$\begin{aligned} &gcd(24e^2+8ed-2e+1, 6e+2d-1)\\ &= gcd(6e+2d-1, 2e+1)\\ &= 1 \end{aligned}$$

the square is latin and thus conflict-free for all $(1,v)\times(x+y-1,v)$ subarrays. By Lemma 2.2.12 an $(x,y)_v$-cut sapphire is conflict-free if the shift-box is conflict-free. Finally, by Theorem 2.1.6 applied to the transpose by Observation 2.1.8 all of the elements in the shift box are distinct and F is an $(x,y)_{\mathrm{lrcf},v}$ square. $\square$

Let $n = 4ey+3y+8e^2+4e-1$ and $x = 4e+2$. Define an $n\times n$ array $G=(a_{i,j}), a_{i,j} \equiv (i-1)(4e+2y-1)+j \pmod{n}$, where e and d are integers.

Theorem 2.2.13 *G is an $(x,y)_{lrcf,v}$ square for $x = 4e+2$ and v relatively prime to n, where e and d are integers.*

The proof is based on the following lemma.

The $(x,y)_v$*-cut rhinestone* is the $(x,y)_v$-cut diamond for $x = 4e+2$, together with $S_v(\frac{x}{2} : y + \frac{x}{2}, 2y + \frac{x}{2} - 2)$. In other words, $y-1$ elements are added onto the North-East side of the East stairs as depicted in Figure 2.17.

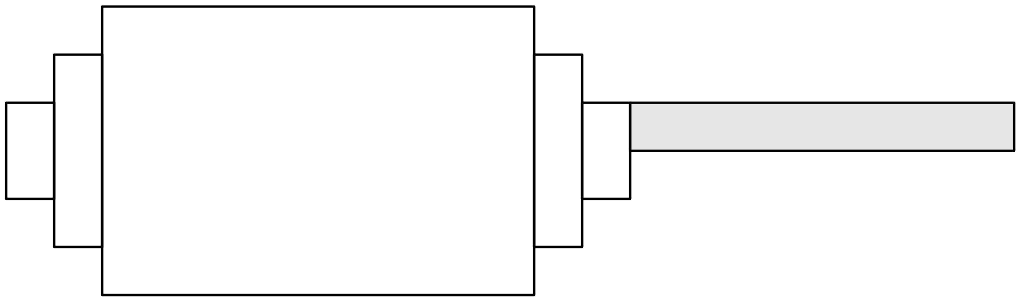

Figure 2.17: The $(x,y)_v$-cut rhinestone

Lemma 2.2.14 *If a $(2e+1, 4e+2y-1, y+2e, v)$ shift-box contains distinct elements G, then the $(x,y)_v$-cut rhinestone contains distinct elements for $x = 4e+2$ for v relatively prime to n, where e and d are integers.*

Proof: We first examine the $(x,y)_1$-cut rhinestone. Place the shift-box over the North-West corner of the base of the $(x,y)_1$-cut rhinestone as depicted in Figure 2.18. Now $a_{i,j} \equiv a_{i-1,j+4e+2y-1} \pmod n$ since $-1(4e+2y-1)+4e+2y-1 \equiv 0 \pmod n$ and the elements are the same. Thus, those elements in rows 2 through $2e+2$ not contained in the shift-box are shifted up one row and over $4e+2y-1$ columns. Next, $a_{i,j} \equiv a_{i-2e-2,j+y+2e-1} \pmod n$ since $(-2e-2)(4e+2y-1)+y+2e-1 \equiv -8e^2 - 4ey - 3y - 4e + 1 \equiv 0 \pmod n$ and the elements are the same.

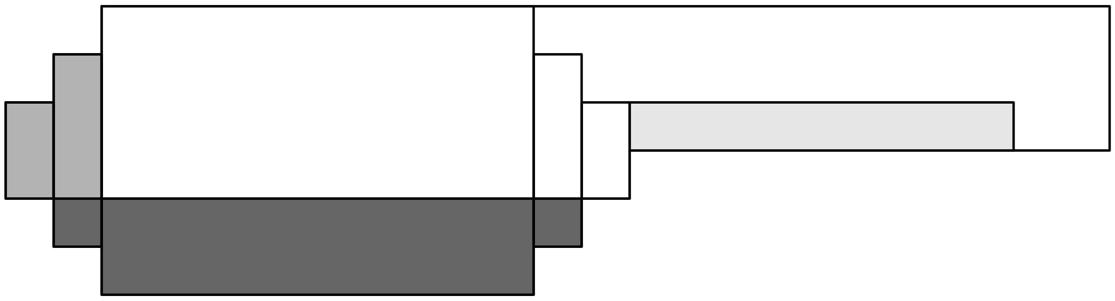

Figure 2.18: Shift-box placed over the $(x, y)_1$-cut rhinestone

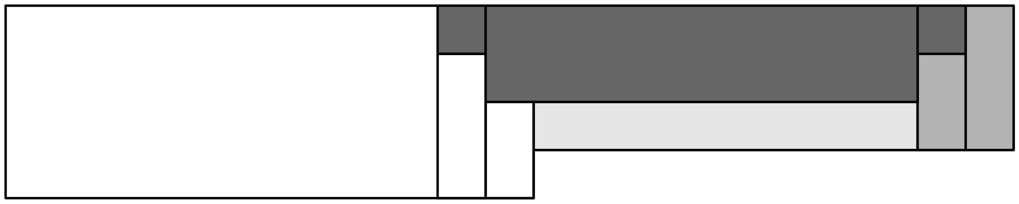

Figure 2.19: Shifting rows $2e + 2$ through $4e + 2$ to shift-box

Thus, the elements in rows $(2e + 3)$ through $(4e + 2)$ are shifted up $(2e + 2)$ rows and over $(y + 2e - 1)$ columns. This is depicted in Figure 2.19. Finally, the above shifts hold for the $(x, y)_v$-cut rhinestone by Lemma 2.1.7. □

Proof of Theorem 2.2.13: By Corollary 2.1.2 if G is conflict-free for some $(x, y)_v$-cut diamond and for some $(1, v) \times (x + y - 1, v)$ subarray, then it is an $(x, y)_{\text{rcf},v}$ square. The $(x, y)_v$-cut rhinestone contains the $(x, y)_v$-cut diamond and thus the statement holds. Since

$$\begin{aligned}
&gcd(4ey + 3y + 8e^2 + 4e - 1, 4e + 2y - 1) \\
&= gcd(4e + 2y - 1, 2e + y) \\
&= 1,
\end{aligned}$$

the square is latin and thus conflict-free for all $(1, v) \times (x + y - 1, v)$ subarrays. By Lemma 2.2.14 an $(x, y)_v$-cut rhinestone is conflict-free if the shift-box is conflict-free. Finally, by Lemma 2.1.6 all of the elements in the shift-box are distinct and G is an $(x, y)_{\mathrm{lrcf},v}$ square. $\square$

Chapter 3

All Constant-Perimeter Subarrays

In this chapter, we examine the templates that represents all constant-perimeter rectangular subarrays. Once again, as the perimeter desired increases, the skewing scheme has a larger set of templates for which to provide conflict-free access.

Let the perimeter of the rectangular subarray desired to be conflict-free be defined as $\gamma = 2p$ where $p = x + y$, $x = \lfloor \frac{p}{2} \rfloor, y = \lceil \frac{p}{2} \rceil$. A square is said to be a *perimeter rectangular conflict-free square with stretch* v * ($p_{\mathrm{rcf},v}$) if it is conflict-free for all rectangular subarrays whose perimeter is less than or equal to γ. If the $p_{\mathrm{rcf},v}$ square with stretch v is also latin, it is a *perimeter latin rectangular conflict-free square with stretch* v * ($p_{\mathrm{lrcf},v}$).

To prove our bounds, we once again use a mega-template which encapsulates the essence of the interaction of the set of templates we want to be conflict-free.

3.1 Preliminary Definitions and Theorems

The following definitions are needed to define a p_v-diamond which we use as a mega-template. In addition, we proceed with some fundamental theorems needed for the proofs of $p_{\mathrm{rcf},v}$ and $p_{\mathrm{lrcf},v}$ squares.

- The *base* is $S_v(0, x-1 : 0, y-1)$.
- A *stair* at level L on the *North* side is $S_v(-L : L, y-1-L) : 0 < L \leq \lfloor \frac{y-1}{2} \rfloor$.
- A *stair* at level L on the *South* side is $S_v(x-1+L : L, y-1-L) : 0 < L \leq \lfloor \frac{y-1}{2} \rfloor$.
- A *stair* at level L on the *West* side is $S_v(L, x-1-L : -L) : 0 < L \leq \lfloor \frac{x-1}{2} \rfloor$.
- A *stair* at level L on the *East* side is $S_v(L, x-1-L : y-1+L) : 0 < L \leq \lfloor \frac{x-1}{2} \rfloor$.
- A *p_v-diamond* is a template that is the union of the base, the North, South, West and East stairs (see Figure 3.1) or any translate in an $n \times n$ square with wrapping.

Corollary 3.1.1 *Let M be a latin square. The following two conditions are equivalent.*

1. *Each p_v-diamond is conflict-free in M.*
2. *For each i satisfying $0 < i < p$, each $(p-i, v) \times (i, v)$ template instance is conflict-free in M.*

Proof: Since M is latin all $(p-1, v) \times (1, v)$ and $(1, v) \times (p-1, v)$ rectangular subarrays are conflict-free. Observe that if $p \equiv 1, 2 \pmod 4$, the p_v-diamond is

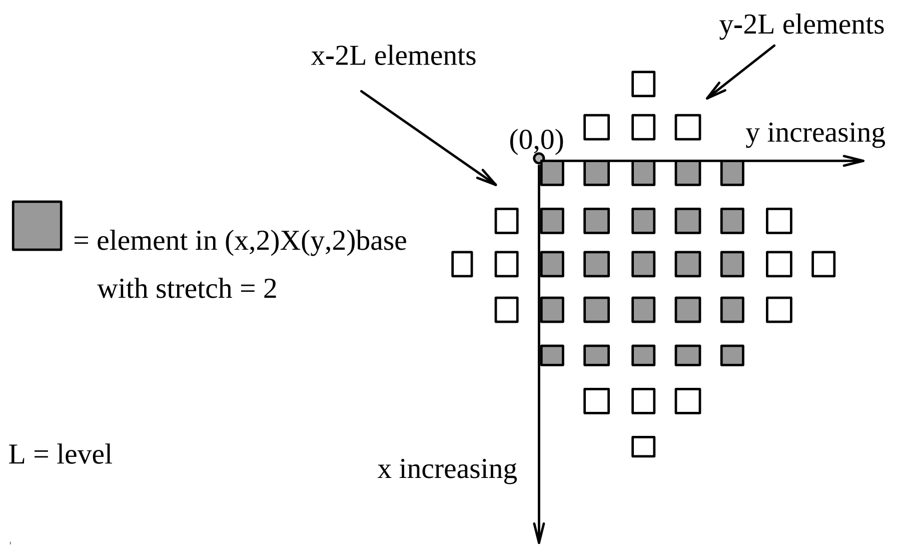

Figure 3.1: The p_2-diamond

isomorphic to the $(p-1,1)_v$-cut diamond. Apply Theorem 2.1.1 for $(x,y)_v$-cut diamonds to obtain the result.

Similarly, if $p \equiv 0,3 \pmod 4$, the p_v-diamond is isomorphic to the $(p-2,2)_v$-cut diamond. Again, apply Theorem 2.1.1 for $(x,y)_v$-cut diamonds to obtain the result.

□

Corollary 3.1.2 *Let M be a cyclic latin square. The following two conditions are equivalent.*

1. *Some p_v-diamond is conflict-free in M.*

2. *For each i satisfying $0 < i < p$, some $(p-i,v) \times (i,v)$ template instance is conflict-free in M.*

Proof: One instance of a template is conflict-free if and only if all instances in M are conflict-free. This follows immediately by the cyclic construction. Thus, the proof is complete by Corollary 3.1.1. $\square$

We now proceed to count the elements contained in p-diamond. Recall that $[x \text{ odd}]$ denotes the function whose value is one if x is odd, and zero otherwise.

Lemma 3.1.3 *The p_v-diamond contains $xy+\frac{1}{2}x^2+\frac{1}{2}y^2-x-y+\frac{1}{2}[x\ odd]+\frac{1}{2}[y\ odd]$ elements.*

Proof: The number of elements in a p_v-diamond is the sum of the base and the four sets of stairs. Once again, let z be the length of the side of the base upon which the stairs are built. Let $n(z)$ be the number of elements in the set of stairs on side z. By Lemma 2.1.3, $n(z) = \frac{1}{4}z^2 - \frac{1}{2}z + \frac{1}{4}$ if z is odd. By Lemma 2.1.4, $n(z) = \frac{1}{4}z^2 - \frac{1}{2}z$ if z is even. Thus, the number of elements in the p_v-diamond is $xy + 2n(x) + 2n(y) = xy + \frac{1}{2}x^2 + \frac{1}{2}y^2 - x - y + \frac{1}{2}[x \text{ odd}] + \frac{1}{2}[y \text{ odd}]$.

3.2 Bounds and Constructions

We now proceed with the proofs for $p_{\text{rcf},v}$ and $p_{\text{lrcf},v}$ squares. We first treat the case when p is odd. Let $x = \lfloor\frac{p}{2}\rfloor, y = \lceil\frac{p}{2}\rceil$ (recall by Observation 2.1.8 that the results apply if $x = \lceil\frac{p}{2}\rceil, y = \lfloor\frac{p}{2}\rfloor$). We want conflict-free access to all $(x-i,v)\times(y+i,v)$ rectangular subarrays of an $n \times n$ array, where $-x < i < x$. This is equivalent to requiring that all rectangular subarrays of perimeter γ be conflict-free where $\gamma = 2p$.

Theorem 3.2.1 *If an $n \times n$ $p_{rcf,v}$ square exists for odd p, then $n \geq 2x^2$.*

Proof: By Corollary 3.1.1 all the elements in the p_v-diamond must be distinct. It follows by Lemma 3.1.3 that $n \geq xy + \frac{1}{2}x^2 + \frac{1}{2}y^2 - x - y + \frac{1}{2}[x \text{ odd}] + \frac{1}{2}[y \text{ odd}]$ where $y = x + 1$.
$n \geq x(x+1) + \frac{1}{2}x^2 + \frac{1}{2}(x+1)^2 - x - (x+1) + \frac{1}{2}[x \text{ odd}] + \frac{1}{2}[(x+1) \text{ odd}] =$

$$\begin{aligned} & x(x+1) + \frac{1}{2}x^2 - x + \frac{1}{2} + \frac{1}{2}(x+1)^2 - (x+1) \\ = \; & x^2 + x + \frac{1}{2}x^2 - x + \frac{1}{2} + \frac{1}{2}(x^2 + 2x + 1) - (x+1) \\ = \; & \frac{3}{2}x^2 + \frac{1}{2} + \frac{1}{2}x^2 + x + \frac{1}{2} - x - 1 \\ = \; & 2x^2. \end{aligned}$$

□

Let $n = 2x^2$ and define an $n \times n$ square $A = (a_{i,j}), a_{i,j} \equiv (i-1)(2x-1) + j \pmod{n}$.

Theorem 3.2.2 *A is a $p_{rcf,v}$ and a $p_{lrcf,v}$ square for odd p when v is relatively prime to n.*

The general strategy is to exhibit a mega-template that is easily seen to be conflict-free, and for which the absence of conflicts implies the whole square is conflict-free for the set of templates. Rather than directly show that the p_v-diamond is conflict-free within the square, we frequently insert an intermediate step involving a box, usually a shift-box. The primary remaining step is to show the relationship between the shift-box and the p_v-diamond. Thus, the proof of Theorem 3.2.2 is based on the following lemma.

Lemma 3.2.3 *If an $(x-1, 2x-1, 3x-1, v)$ shift-box contains distinct elements in A then the p_v-diamond contains distinct elements for all x, odd p and v relatively prime to n.*

Proof: First we examine the p_1-diamond. The upper left-hand corner of the shift-box is placed over the North-West corner of the North-most stair of the p_1-diamond as displayed in Figure 3.2. $a_{i,j} \equiv a_{i-1,j+2x-1} \pmod{n}$ since $(-1)(2x-1)+(2x-$

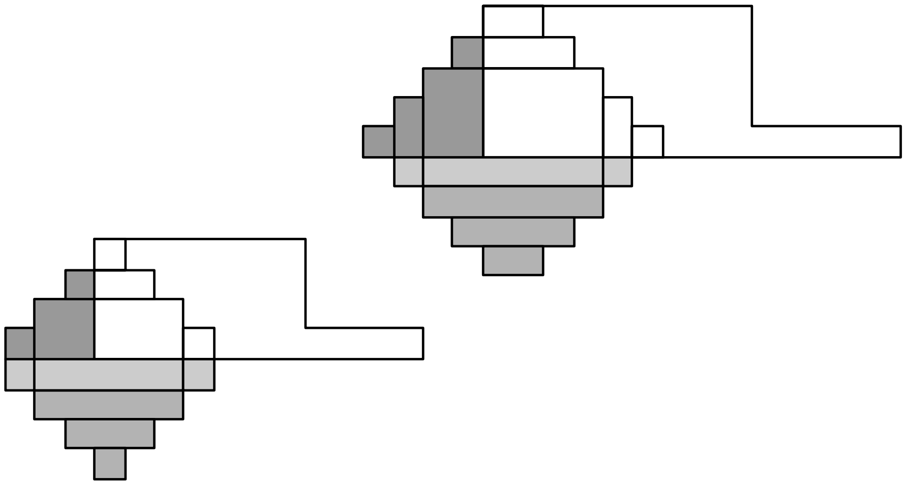

Figure 3.2: Shift-box placed over the p_1-diamond

$1) \equiv 0 \pmod{n}$, and the symbol is the same. Thus, row $(x+1)$ is shifted up one row and over $(2x-1)$ columns as displayed in Figure 3.3. Now, $a_{i,j} \equiv a_{i-x-1,j+x-1} \pmod{n}$ since $(-x-1)(2x-1)+(x-1) \equiv -2x^2 \equiv 0 \pmod{n}$ and the symbol is the same. Thus, rows $(x+2)$ through $(2x-1[x \text{ odd}])$ are shifted up $(x+1)$ rows and over $(x-1)$ columns as displayed in Figure 3.4. Finally, $a_{i,j} \equiv a_{i-1,j+2x-1} \pmod{n}$ since $(-1)(2x-1)+(2x-1) \equiv 0 \pmod{n}$ and the symbol is the same. Thus, all the elements in rows 2 through x not contained in the shift-box are shifted up 1 row and over $(2x-1)$ columns. This final shift is displayed in Figure 3.5. Finally, the above shifts are valid for the p_v-diamond by Lemma 2.1.7. □

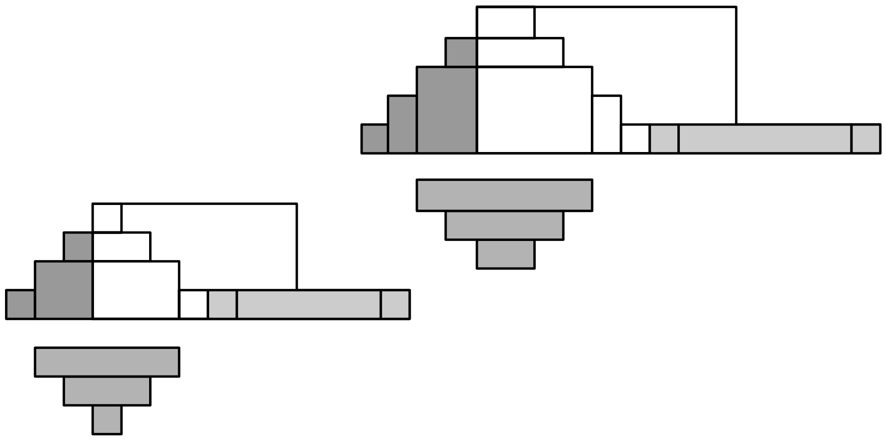

Figure 3.3: Shifting row $x + 1$ to shift-box

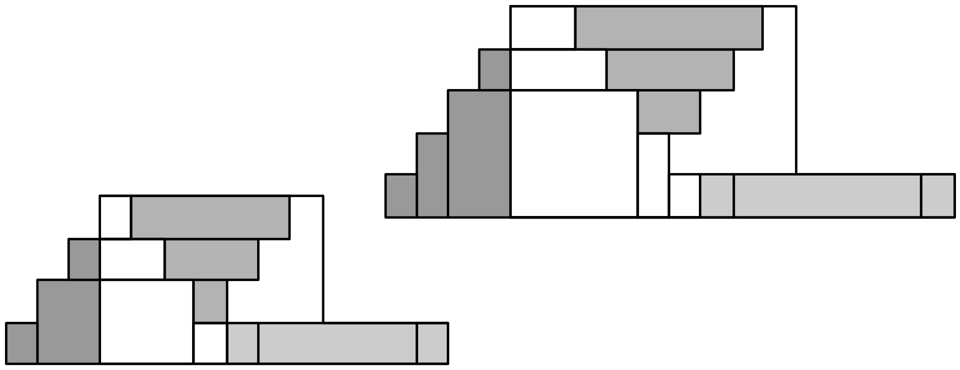

Figure 3.4: Shifting rows $x + 2$ through $2x - 1[x \text{ odd}]$ to shift-box

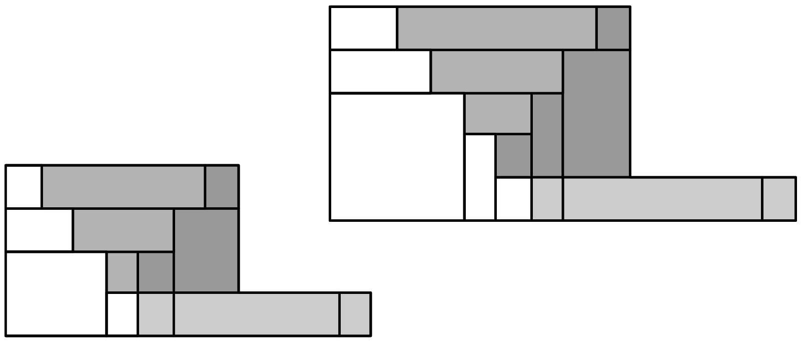

Figure 3.5: Shifting rows 1 through x to shift-box

Proof of Theorem 3.2.2: Since $n = 2x^2$ is relatively prime to $s = 2x - 1$, the square is latin. Thus, by Corollary 3.1.2 if A is conflict-free for some p_v-diamond then it is a $p_{\mathrm{rcf},v}$ square. By Lemma 3.2.3, a p_v-diamond is conflict-free if the shift-box is conflict-free. Finally, by Theorem 2.1.6 all of the elements in the shift box are distinct and A is a $p_{\mathrm{lrcf},v}$ square. $\square$

Let σ be the maximum size of any template in the set of constant-perimeter templates. For odd p, $\sigma = x^2 + x$ and for even p, $\sigma = x^2$.

Corollary 3.2.4 *The $p_{lrcf,v}$ square for odd p can be accessed in two memory cycles when stored in σ memory modules if v is relatively prime to n.*

Proof: Since $n = 2x^2 \leq 2x(x+1) = 2\sigma$ for all $x \geq 0$, any memory modules from σ to $2\sigma - 1$ can be mapped onto the memory modules 0 to $\sigma - 1$ and can be accessed in the second memory cycle. $\square$

Thus, if one wishes to increase memory utilization to satisfy **Point 3**, one can store the templates in σ memory modules.

Now, we turn to the second main case, when the perimeter $\gamma = 2p$ and p is even. We write $p = 2x$, and thus $y = x = \frac{p}{2}$. We would like conflict-free access to all $(x - i, v) \times (x + i, v)$ rectangular subarrays of an $n \times n$ array for $-x < i < x$.

Theorem 3.2.5 *If a $p_{rcf,v}$ square of order n exists for even p, $n \geq 2x^2 - 2x + 1$.*

Proof: By Corollary 3.1.1 all of the elements in the p_v-diamond must be distinct. It follows by Lemma 3.1.3 that $n \geq 2x^2 - 2x + [x \text{ odd}]$. Thus, when x is odd, we achieve the bound in the statement of the theorem. When x is even, we obtain $n \geq 2x^2 - 2x$ by Lemma 3.1.3. Thus, we only need to show that $n \neq 2x^2 - 2x$. Shapiro shows in general that a template tiles a plane if and only if a skewing scheme exists [Sha78]. Thus, any $p_{\mathrm{rcf},v}$ square of order $2x^2 - 2x$ must correspond to a tiling of the plane with a p_v-diamond. Thus, one can choose an arbitrary external element of the p_v-diamond and attempt to place it next to every possible external element in a replica copy of a p_v-diamond. This type of case analysis reveals that there are only two possible tilings which are represented in Figure 3.6 and Figure 3.7.

In the first tiling, a $(1, v) \times (x + x - 1, v)$ template is not conflict-free. Thus,

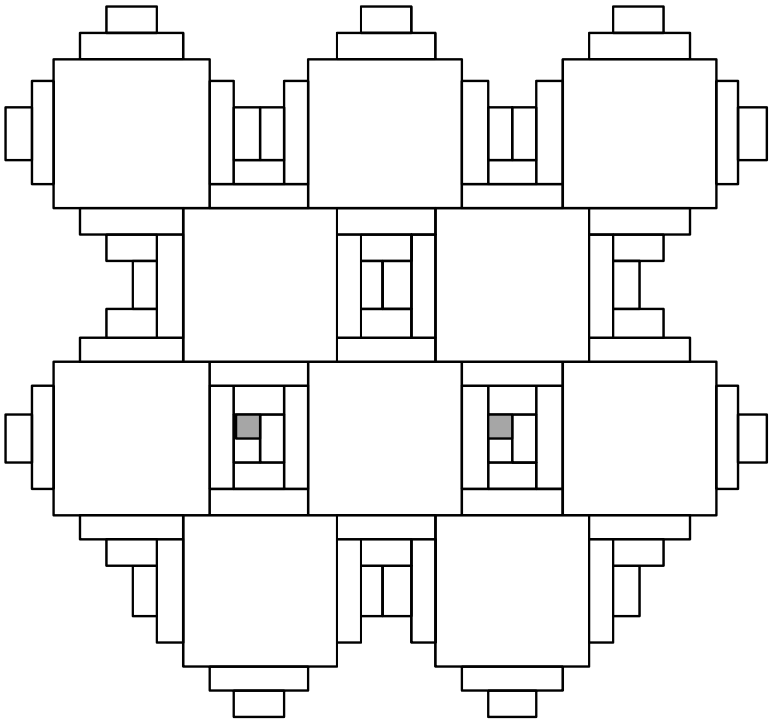

Figure 3.6: First possible tiling of the p_v-diamond for even p and even x

while the p_v-diamond tiles the plane, the tiling does not provides a $p_{\mathrm{rcf},v}$ square. In the second tiling, an $(x + x - 1, v) \times (1, v)$ template is not conflict-free. Thus,

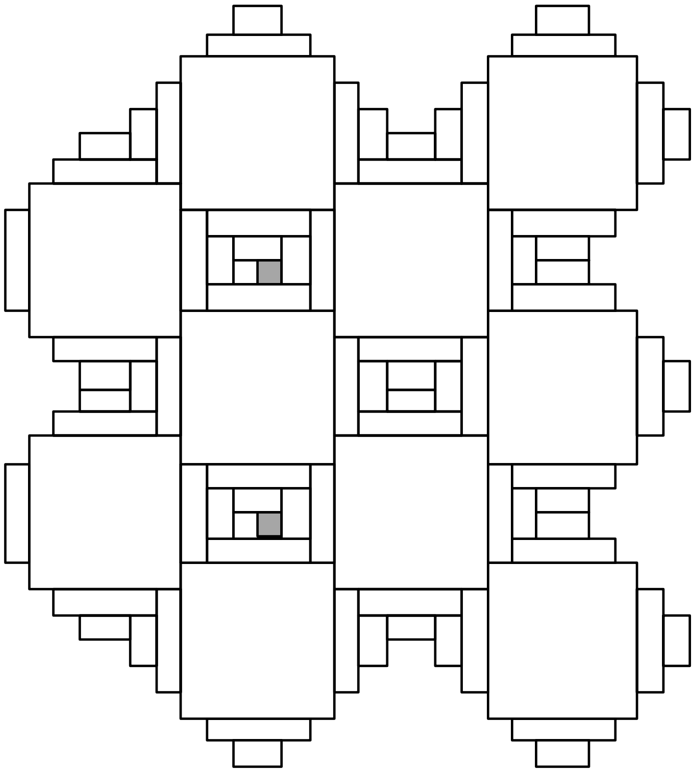
Figure 3.7: Second possible tiling of the p_v-diamond for even p and even x

there is no possible tiling of the plane with the p_v-diamond and $n \geq 2x^2-2x+1$. $\square$

Let $n = 2x^2-2x+1$, and define an $n \times n$ square $B = (a_{i,j})$, $a_{i,j} \equiv (i-1)(2x-1)+j \pmod{n}$.

A *p_v-zircon* is the p_v-diamond for even p and even x together with the cell $S_v(\frac{x}{2}-1 : x+\frac{x}{2}-1)$, which is an element added onto the North-East side of the East stairs as depicted in Figure 3.8.

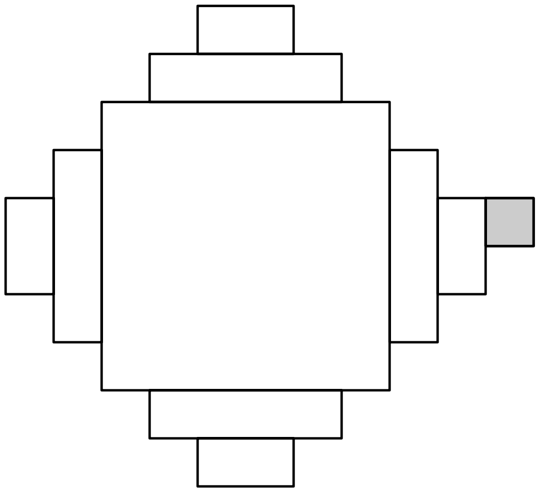

Figure 3.8: The p_v-zircon

Theorem 3.2.6 *B is a $p_{rcf,v}$ and a $p_{lrcf,v}$ square for even p when v is relatively prime to n.*

The proof is based on the following lemma.

Lemma 3.2.7 *If an $(x-1, 2x-1, x, v)$ shift-box contains distinct elements in B, then the p_v-diamond for odd x and the p_v-zircon for even x contain distinct elements for even p when v is relatively prime to n.*

Proof: First we examine the p_1-diamond and the p_1-zircon. The upper left-hand corner of the shift-box is placed over the North-West corner of the North-most stair of the p_1-diamond or the p_1-zircon as displayed in Figure 3.9. Next, $a_{i,j} \equiv$

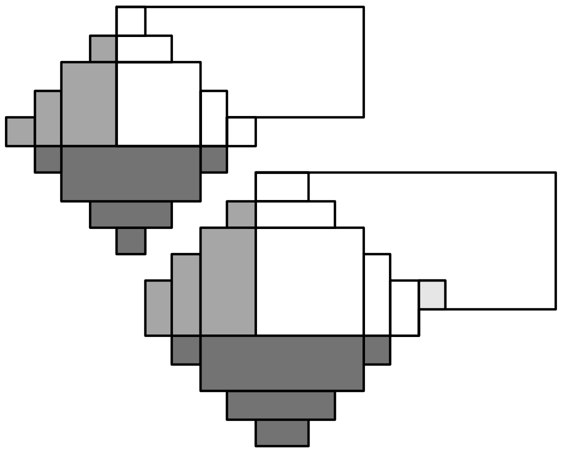
Figure 3.9: Shift-box placed over the p_1-diamond and p_1-zircon

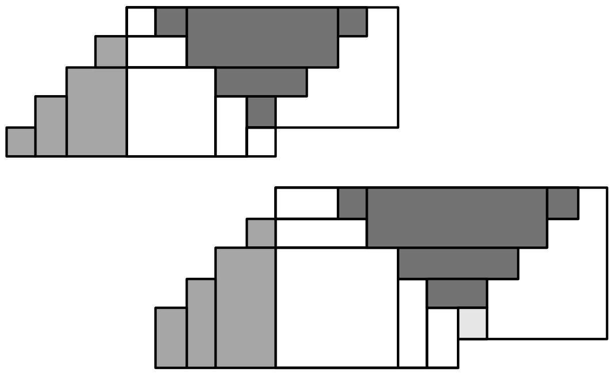
Figure 3.10: Shifting rows $x+1$ through $2x-1$ to shift-box

$a_{i-x,j+x-1} \pmod{n}$ since $(-x)(2x-1)+(x-1) \equiv 0 \pmod{n}$. Thus, according to the cyclic construction, this element is the same element and rows $(x+1)$ through $(2x-1)$ are shifted up x rows and over $(x-1)$ columns as displayed in Figure 3.10. Now, $a_{i,j} \equiv a_{i-1,j+2x-1} \pmod{n}$ since $(-1)(2x-1)+2x-1 \equiv 0 \pmod{n}$, this element is the same. Thus, all of the elements in rows 2 through x not contained in the shift-box are shifted up one row and over $(2x-1)$ columns. This shift is displayed in Figure 3.11. Finally, by Lemma 2.1.7, the above shifts are valid for p_v-diamonds and p_v-zircons. $\square$

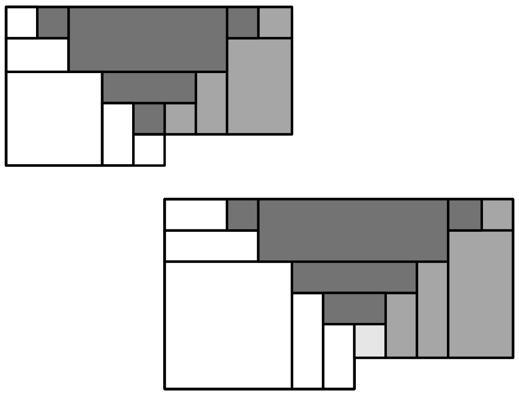

Figure 3.11: Shifting rows 1 through x to shift-box

Proof of Theorem 3.2.6. Since $(2x-1)$ is relatively prime to $(2x^2-2x+1) = (2x-1)(x-1)+x$, B is latin. Thus, by Corollary 3.1.2 if B is conflict-free for some p_v-diamond then it is a $p_{\mathrm{rcf},v}$ square. Since the p_v-zircon contains the p_v-diamond, this also holds for the p_v-zircon. By Lemma 3.2.7, the p_v-diamond and the p_v-zircon are conflict-free if the shift-box is conflict-free. Finally, by Theorem 2.1.6 all of the elements in the shift box are distinct and B is a $p_{\mathrm{lrcf},v}$ square. $\square$

Recall that σ is the maximum size of any template in the set of constant-perimeter templates.

Corollary 3.2.8 *The $p_{lrcf,v}$ square for even p can be accessed in two memory cycles when stored in $\sigma = x^2$ memory modules if v is relatively prime to n.*

Proof: Since $n = 2x^2 - 2x + 1 < 2x^2 = 2\sigma$ for all $x > 0$, any memory modules from σ to $2\sigma - 1$ can be mapped onto the memory modules 0 to $(\sigma - 1)$ and be accessed in the second memory cycle. $\square$

Once again, if one wishes to increase memory utilization, one can store the templates in σ memory modules. This would lead to a higher memory utilization and accommodate **Point 3**. Observe that if $vu = n$, that is to say, v is not relatively prime to n, there are v symbols in the p-diamond which are repeated u times. Each set of v symbols requires two memory cycles. Thus, for $vu = n$, $2u$ memory cycles are required when stored in σ memory modules.

Since the skewing scheme should reflect the needs of algorithms (**Point 1**) and maximize memory utilization (**Point 3**), one may desire more flexibility for the situation when v is not relatively prime to n. Thus, we offer another set of skewing schemes. If the number of memory modules used in the first scheme is relatively prime to the number of memory modules used in the additional scheme, then the only stretches which cannot be accommodated are those which have a common prime factor with each scheme.

Once again, let the perimeter of the rectangular subarray be $\gamma = 2p$. We treat the case when p is even, the perimeter $\gamma = 2p$ and thus $y = x$. In this case we directly show that the p-diamond is conflict-free within the square. This is accomplished by breaking the problem into three cases, when $x \equiv 0, 1, 2 \pmod 3$. We would like to be able to have conflict-free access to all $(x-i) \times (x+i)$ contiguous rectangular subarrays of an $n \times n$ array where $-x < i < x$.

Let $n = 2x^2 - 2x + 3 = 18e^2 - 18e + 7$, and define an $n \times n$ square $A = (a_{i,j})$, $a_{i,j} \equiv (i-1)(6e^2 - 8e + 4) + j \pmod n$ where $x = 3e - 1$ and e is an integer.

Let the p_v-rhinestone be the p_v-diamond for $p \equiv 0 \pmod 4$ together with $S_v(\frac{x}{2} : x + \frac{x}{2}) \cup S_v(\frac{x}{2} + 1 : x + \frac{x}{2} - 1) \cup S_v(\frac{x}{2} + 2 : x + \frac{x}{2} - 2)$. In other words, there is one element added onto the South-East end of the three East-most stairs (see Figure 3.12).

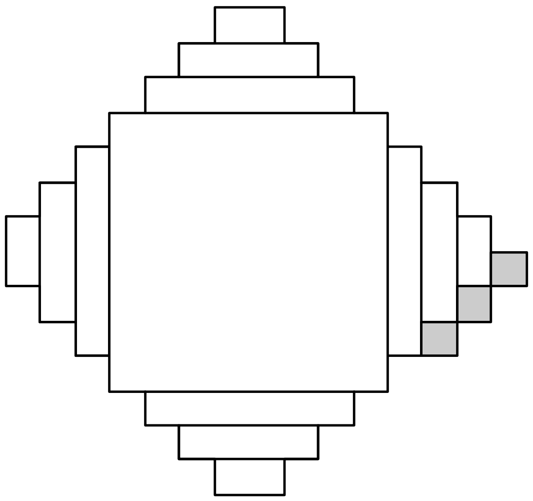

Figure 3.12: The p_v-rhinestone

Let the p_v-sapphire be the p_v-diamond for $p \equiv 2 \pmod 4$ together with $S_v(\lceil\frac{x}{2}\rceil : x + \lfloor\frac{x}{2}\rfloor - 1) \cup S_v(\lceil\frac{x}{2}\rceil + 1 : x + \lfloor\frac{x}{2}\rfloor)$. In other words, there is one element added onto the South side of the two East-most stairs (see Figure 3.13).

Theorem 3.2.9 *A is a $p_{rcf,v}$ and a $p_{lrcf,v}$ square for even p and $x = 3e - 1$ where e is an integer.*

The proof is based on the following lemmas.

Lemma 3.2.10 *The p_v-sapphire contains distinct elements in A for $x = 3e - 1$ where e is an even integer.*

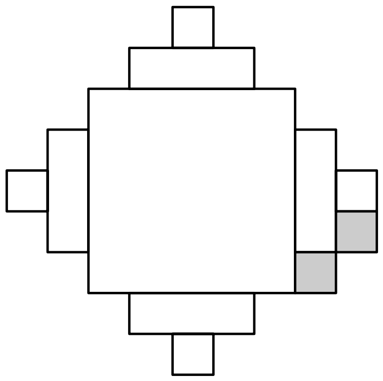

Figure 3.13: The p_v-sapphire

Proof: For even e, there are $6e-3$ rows in the p_v-sapphire. The first $3e-1$ rows continue counting $3e-2$ rows down since $a_{i,j} \equiv a_{i+3e-2,j-3e+1} \pmod{n}$. Thus, rows $3e-3$ through $6e-3$ are a continuation of these rows (see Figure 3.14). Rows $3e$ and $3e+1$ continue at rows $3e-3$ and $3e-2$ as they are shifted up three rows since $a_{i,j} \equiv a_{i-3,j-6e+5} \pmod{n}$. The remaining rows, row $3e+2$ through $6e-3$ continue counting $3e+1$ rows up since $a_{i,j} \equiv a_{i-3e-1,j-3e+4} \pmod{n}$. Thus, all the rows and all the elements are uniquely accounted for. $\square$

Lemma 3.2.11 *The p_v-rhinestone contains distinct elements in A for $x = 3e-1$ where e is an odd integer.*

Proof: For odd e, $e > 1$, there are $6e-4$ rows in the p_v-rhinestone. The first $3e-2$ rows continue counting $3e-2$ rows down since $a_{i,j} \equiv a_{i+3e-2,j-3e+1} \pmod{n}$. Thus, rows $3e-3$ through $6e-4$ are a continuation of these rows. Rows $3e-1$ through $3e+1$ continue at rows $3e-4$ through $3e-2$ as they are shifted up three rows since $a_{i,j} \equiv a_{i-3,j-6e+5} \pmod{n}$. The remaining rows, row $3e+2$ through $6e-4$ continue counting $3e+1$ rows up since $a_{i,j} \equiv a_{i-3e-1,j-3e+4} \pmod{n}$. Thus, all the rows and all the elements are uniquely accounted for.

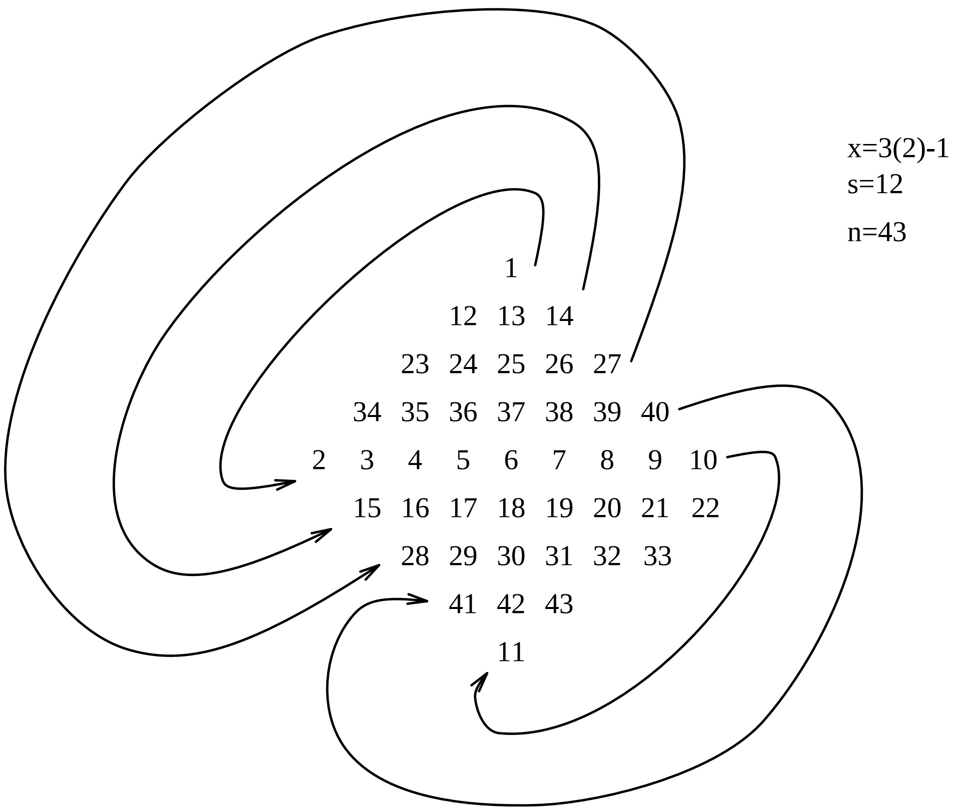

Figure 3.14: First $3e-1$ rows continue $3e-2$ rows down

For the case when $e = 1$, the 4_v-diamond is conflict-free as exhibited in the square in Figure 3.1. □

Proof of Theorem 3.2.9: Since,

$$\begin{aligned} & gcd(18e^2 - 18e + 7, 6e^2 - 8e + 4) \\ = \;& gcd(18e^2 - 18e + 7, 3e^2 - 4e + 2) \\ = \;& gcd(3e^2 - 4e + 2, 6e - 5) \\ = \;& gcd(6e^2 - 8e + 4, 6e - 5) \end{aligned}$$

0	1	2	3	4	5	6
2	3	4	5	6	0	1
4	5	6	0	1	2	3
6	0	1	2	3	4	5
1	2	3	4	5	6	0
3	4	5	6	0	1	2
5	6	0	1	2	3	4

Table 3.1: The A square for $e = 1$

$$\begin{aligned} &= gcd(6e-5, -3e+4) \\ &= gcd(-3e+4, 3) \\ &= 1. \end{aligned}$$

the shift is relatively prime to n and A is a latin square. By Corollary 3.1.2, A is a $p_{\mathrm{lrcf},v}$-square if the p_v-diamond contains distinct elements. Since p_v-sapphire and the p_v-rhinestone contain the p_v-diamond, this also holds for the p_v-sapphire and the p_v-rhinestone. By Lemmas 3.2.10 and 3.2.11, the p_v-sapphire and the p_v-rhinestone respectively contain distinct elements. □

Let $n = 2x^2 - 2x + 5 = 18e^2 - 6e + 5$, and define an $n \times n$ square $B = (a_{i,j})$, $a_{i,j} \equiv (i-1)(6e^2 - 4e + 2) + j \pmod{n}$ where $x = 3e$ and where e is an integer.

Let the p_v-ruby be the p_v-diamond for $p \equiv 0 \pmod 4$, together with the elements $S_v(\frac{-x}{2}+1 : \frac{x}{2}+1) \cup S_v(\frac{x}{2} : x + \frac{x}{2} - 1) \cup S_v(\frac{x}{2}+1 : x + \frac{x}{2} - 2, x + \frac{x}{2})$. In other words, there is one element added onto the East side of the North-most stair, one element added onto the South-East set of stairs, and three elements added onto the

East side of the stairs at level $\frac{x}{2} - 2$ (see Figure 3.15).

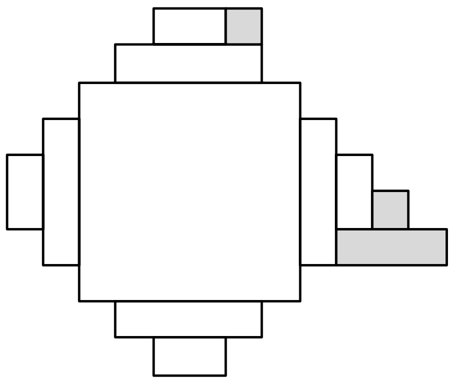

Figure 3.15: The p_v-ruby

Let the p_v-topaz be the p_v-diamond for $p \equiv 2 \pmod 4$, together with the elements $S_v(\lfloor\frac{-x}{2}\rfloor : \lceil\frac{x}{2}\rceil, \lceil\frac{x}{2}\rceil + 1) \cup S_v(\lceil\frac{x}{2}\rceil : x + \lfloor\frac{x}{2}\rfloor - 1, x + \lfloor\frac{x}{2}\rfloor)$. In other words, there are two elements added onto the North-East side of the North stairs, and two elements added onto the East side of the East stairs at level $\lfloor\frac{x}{2}\rfloor - 1$ (see Figure 3.16).

Theorem 3.2.12 *B is a $p_{rcf,v}$ and a $p_{lrcf,v}$ square for even p and $x = 3e$.*

The proof is based on the following lemmas.

Lemma 3.2.13 *The p_v-ruby contains distinct elements in B for even $x = 3e$, where e and p are even.*

Proof: For even e, there are $6e-2$ rows in the p_v-ruby. The first $3e$ rows continue counting $3e-2$ rows down since $a_{i,j} \equiv a_{i+3e-2,j-3e-1} \pmod n$. Thus, rows $3e-1$ through $6e-2$ are a continuation of these rows. Row $3e+1$ continues at row $3e-2$ since $a_{i,j} \equiv a_{i-3,j-6e+1} \pmod n$. The remaining rows, row $3e+2$ through $6e-2$

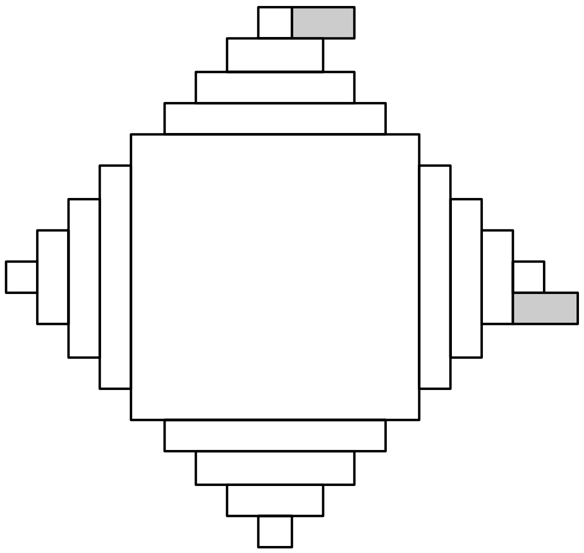

Figure 3.16: The p_v-topaz

continue counting $3e+1$ rows up since $a_{i,j} \equiv a_{i-3e-1,j-3e+2} \pmod{n}$. Thus, all the rows and all the elements are uniquely accounted for. $\Box$

Lemma 3.2.14 *The p_v-topaz contains distinct elements in B for even $x = 3e$, where e is odd and p is even.*

Proof: For odd e, there are $6e-1$ rows in the p_v-topaz. The first $3e+1$ rows continue counting $3e-2$ rows down since $a_{i,j} \equiv a_{i+3e-2,j-3e-1} \pmod{n}$. Thus, rows $3e-1$ through $6e-1$ are a continuation of these rows. The remaining rows, row $3e+2$ through $6e-1$ continue counting $3e+1$ rows up since $a_{i,j} \equiv a_{i-3e-1,j-3e+2}$ $\pmod{n}$. Thus, all the rows and all the elements are uniquely accounted for. $\Box$

Proof of Theorem 3.2.12: Since,

$$gcd(18e^2 - 6e + 5, 6e^2 - 4e + 2)$$

$$\begin{aligned}
&= gcd(18e^2 - 6e + 5, 3e^2 - 2e + 1)\\
&= gcd(3e^2 - 2e + 1, 6e - 1)\\
&= gcd(6e^2 - 4e + 2, 6e - 1)\\
&= gcd(6e - 1, -3e + 2)\\
&= gcd(-3e + 2, 3)\\
&= 1.
\end{aligned}$$

the shift is relatively prime to n and B is a latin square. By Corollary 3.1.2, B is a $p_{\mathrm{lrcf},v}$-square if the p_v-diamond contains distinct elements. Since p_v-ruby and the p_v-topaz contain the p_v-diamond, this also holds for the p_v-ruby and the p_v-topaz. By Lemmas 3.2.13 and 3.2.14, the p_v-ruby and the p_v-topaz respectively contain distinct elements. □

Let $n = 2x^2 - 2x + 5 = 18e^2 + 6e + 5$, and define an $n \times n$ square $C = (a_{i,j}), a_{i,j} \equiv (i-1)(6e^2 + 4e + 2) + j \pmod n$ and where $x = 3e + 1$, where e is an integer.

Let the p_v-emerald be the p_v-diamond for $p \equiv 0 \pmod 4$ together with the elements $S_v(\frac{x}{2} - 2 : x + \frac{x}{2} - 2, x + \frac{x}{2}) \cup S_v(\frac{x}{2} - 1 : x + \frac{x}{2} - 1) \cup S_v(x + \frac{x}{2} - 2 : \frac{x}{2} + 1)$. In other words, there are three elements added onto the East side of the East stairs at level $\frac{x}{2} - 2$, one element added onto the East side of the stairs at level $\frac{x}{2} - 1$ and one element added onto the East side of the South stair at level $\frac{x}{2} - 1$ (see Figure 3.17).

Let the p_v-opal be the p_v-diamond for $p \equiv 2 \pmod 4$ together with the elements $S_v(\lfloor\frac{x}{2}\rfloor - 1 : x + \lfloor\frac{x}{2}\rfloor - 2, x + \lfloor\frac{x}{2}\rfloor - 1) \cup S_v(x + \lfloor\frac{x}{2}\rfloor - 1 : \lfloor\frac{x}{2}\rfloor + 1, \lfloor\frac{x}{2}\rfloor + 2)$. In other words, there are two elements added onto the East side of the East stairs at level $\lfloor\frac{x}{2}\rfloor - 1$ and two elements added onto the East side of the South stairs at

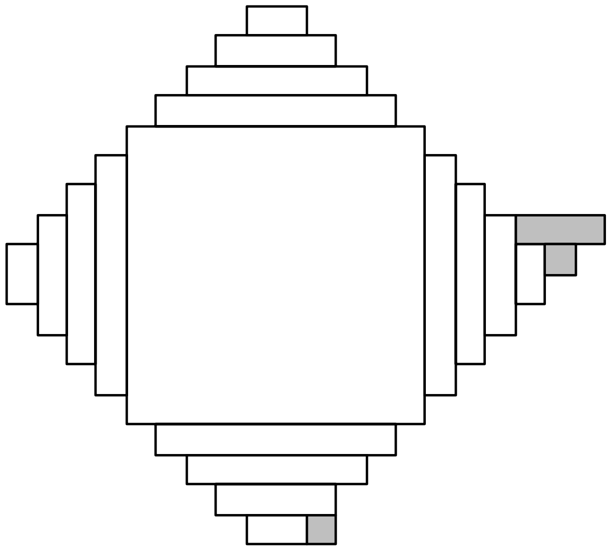

Figure 3.17: The p_v-emerald

level $x + \lfloor \frac{x}{2} \rfloor + 1$ (see Figure 3.18).

Theorem 3.2.15 *C is a p_{rcf_v} and a p_{lrcf} square for even p and $x = 3e + 1$ where e is an integer.*

The proof is based on the following lemmas.

Lemma 3.2.16 *The p_v-opal contains distinct elements in C for $x = 3e$ where e and p are even.*

Proof: For even e, there are $6e + 1$ rows in the p_v-opal. The first $3e - 1$ rows continue counting $3e + 2$ rows down since $a_{i,j} \equiv a_{i+3e+2,j-3e+1} \pmod{n}$. Thus, rows $3e + 3$ through $6e + 1$ are a continuation of these rows. The remaining rows, row $3e$ through $6e + 1$ continue counting $3e - 1$ rows up since $a_{i,j} \equiv a_{i-3e+1,j-3e-2}$ $\pmod{n}$. Thus, all the rows and all the elements are uniquely accounted for. $\square$

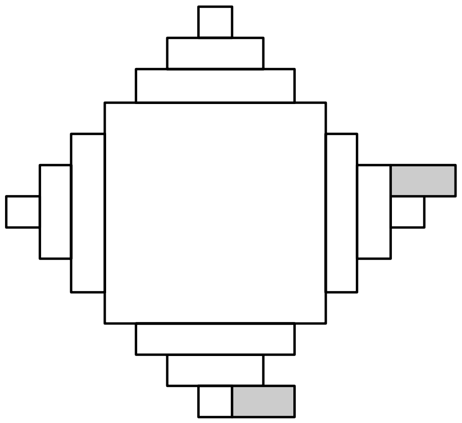

Figure 3.18: The p_v-opal

Lemma 3.2.17 *The p_v-emerald contains distinct elements in C for $x = 3e$ where e is odd and p is even.*

For odd e, there are $6e$ rows in the p_v-emerald. The first $3e - 2$ rows continue counting $3e + 2$ rows down since $a_{i,j} \equiv a_{i+3e+2,j-3e+1} \pmod{n}$. Thus, rows $3e + 3$ through $6e$ are a continuation of these rows. Row $3e - 1$ continues at row $3e + 2$ since $a_{i,j} \equiv a_{i+3,j-6e-1} \pmod{n}$. The remaining rows, row $3e$ through $6e$ continue counting $3e - 1$ rows up since $a_{i,j} \equiv a_{i-3e+1,j-3e-2} \pmod{n}$. Thus, all the rows and all the elements are uniquely accounted for. □

Proof of Theorem 3.2.15 Since,

$$\begin{aligned} & gcd(18e^2 + 6e + 5, 6e^2 + 4e + 2) \\ = \;& gcd(6e^2 + 4e + 2, -6e - 1) \\ = \;& gcd(-6e - 1, 3e + 2) \\ = \;& gcd(3e + 2, 3) \\ = \;& 1. \end{aligned}$$

the shift is relatively prime to n and C is a latin square. By Corollary 3.1.2, A is a $p_{\mathrm{lrcf},v}$-square if the p_v-diamond contains distinct elements. Since p_v-opal and the p_v-emerald contain the p_v-diamond, this also holds for the p_v-opal and the p_v-emerald. By Lemmas 3.2.16 and 3.2.17, the p_v-opal and the p_v-emerald respectively contain distinct elements. □

Tables 3.2 through 3.4 contain the listings of the prime factors of the n value for our minimum linear skewing scheme, referred to in the tables as first, and the aforementioned schemes referred to in the tables as second. If one or the other scheme yields a prime number of memory modules, for a particular x value, then that x value is not listed since all possible stretch values are possible in those cases. A stretch value which will not be conflict-free must have a prime factor in each of the two categories, otherwise it is conflict-free in the scheme with which it does not share a prime factor.

Table 3.5 indicates computational results for additional linear skewing schemes for $p_{\mathrm{lrcf},v}$ squares for even p. They are displayed by their x value and by the number of memory modules utilized above the minimum linear skewing scheme. The bold-face numbers represent the families of skewing schemes proved in this thesis. Table 3.6 indicates computational results for additional linear skewing schemes for $p_{\mathrm{lrcf},v}$ squares for odd p. They are displayed by their x value and by the number of memory modules utilized above the minimum linear skewing scheme. The bold-face numbers indicate those schemes which are equivalent to using the skewing scheme for $p + 1$.

x	first	second	x	first	second
16	13, 37	5, 97	75	17, 653	5, 2221
19	5, 137	13, 53	76	13, 877	5, 2281
28	17, 89	37, 41	77	2341, 5	23, 509
34	5, 449	13, 173	78	41, 293	61, 197
37	5, 13, 41	17, 157	81	13, 997	5, 2593
38	29, 97	5, 563	82	5, 2657	97, 137
41	17, 193	7, 67	84	5, 2789	13, 29, 37
44	5, 757	7, 541	90	37, 433	5, 641
45	17, 233	5, 13, 61	96	17, 29, 37	5, 41, 89
46	41, 101	5, 829	97	5, 149	13, 1433
49	5, 941	17, 277	99	5, 3881	13, 1493
53	37, 149	5, 1103	102	5, 13, 317	37, 557
54	5, 229	17, 337	104	5, 857	7, 3061
55	13, 457	5, 29, 41	106	113,197	5, 61,73
58	17, 389	13, 509	107	5,13,349	7, 463
59	5, 37	41, 167	112	5, 4973	13,1913
60	73, 97	5, 13, 109	113	17,1489	5,61,83
62	5, 17, 89	7, 23, 47	114	5,5153	73,353
65	53, 157	7, 29, 41	115	13,2017	5,1049
68	13, 701	5, 1823	117	5,61,89	17,1597
69	5, 1877	41, 229	121	113,257	5,37,157
72	5, 409	53, 193	125	29,1069	7,43,103
74	5, 2161	101, 107			

Table 3.2: List of prime factors of n values $1 \leq x \leq 125$

x	first	second	x	first	second
126	17, 109	5, 6301	171	53, 1097	5, 29, 401
128	13, 41, 61	5, 7, 929	172	5, 13, 181	89, 661
130	17, 1973	5, 6709	177	5, 17, 733	13, 4793
134	5, 7129	43, 829	178	61, 1033	29, 41, 53
135	97, 373	5, 7237	180	13, 4957	5, 12889
137	5, 29, 257	83, 449	181	17, 3833	5, 13033
139	5, 7673	17, 37, 61	182	5, 13177	41, 1607
141	13, 3037	5, 53, 149	185	13, 5237	103, 661
142	5, 8009	29, 1381	187	5, 13913	73, 953
143	17, 2389	5, 8123	191	181, 401	7, 10369
146	13, 3257	7, 23, 263	192	5, 14669	41, 1789
149	5, 8821	7, 6301	193	13, 5701	137, 541
151	89, 509	5, 13, 17, 41	195	29, 2609	5, 37, 409
152	5, 9181	29, 1583	197	5, 3089	29, 2663
153	241, 193	181, 257	201	37, 41, 53	5, 13, 1237
156	137, 353	5, 17, 569	202	5, 109, 149	17, 281
159	5, 13, 773	109, 461	204	5, 3313	113, 733
160	17, 41, 73	5, 10177	207	5, 37, 461	17, 29, 173
162	5, 10433	13, 4013	209	5, 17389	7, 12421
164	5, 17, 37	127, 421	210	41, 2141	5, 97, 181
166	29, 1889	5, 10957	211	13, 17, 401	5, 709
167	853, 5, 13	7, 89	212	5, 29, 617	7, 12781
169	5, 41, 277	109, 521	214	5, 18233	13, 7013
170	37, 1553	7, 8209	215	17, 5413	23, 4001

Table 3.3: List of prime factors of n values $126 \leq x \leq 215$

x	first	second	x	first	second
216	317, 293	5, 13, 1429	256	137, 953	5, 26113
217	5, 18749	241, 389	257	5, 26317	181, 727
219	5, 13, 113	17, 41, 137	258	13, 101	17, 29, 269
220	557, 173	5, 19273	263	13, 10601	5, 43, 641
228	17, 6089	61, 1697	269	5, 28837	23, 6269
229	5, 4177	13, 29, 277	270	29, 5009	5, 17, 1709
232	5, 13, 17, 97	37, 2897	271	13, 11257	5, 29269
233	73, 1481	5, 7, 3089	272	5, 5897	7, 21061
235	109, 1009	5, 21997	274	5, 29921	41, 89
238	37, 3049	101, 1117	275	37, 4073	7, 21529
239	5, 61, 373	29, 3923	276	13, 11677	5, 97, 313
240	89, 1289	5, 13, 353	278	233, 661	5, 30803
241	29, 3989	5, 17, 1361	279	5, 17, 73	13, 11933
242	5, 41, 569	67, 1741	284	5, 13, 2473	23, 29, 241
246	809, 149	5, 24109	287	5, 32833	181, 907
247	5, 4861	53, 2293	289	5, 13, 197	61, 2729
248	101, 1213	5, 107, 229	292	5, 41, 829	13, 17, 769
249	5, 17, 1453	113, 1093	293	137, 1249	5, 7, 4889
250	13, 61, 157	5, 37, 673	294	5, 34457	13, 29, 457
251	41, 3061	7, 17929	295	89, 1949	5, 34693
252	5, 25301	73, 1733	296	17, 10273	7, 61, 409
253	29, 4397	13, 17, 577	300	17, 61, 173	5, 53, 677
254	5, 53, 97	7, 43, 61			
255	281, 461	5, 13, 1993			

Table 3.4: List of prime factors of n values $216 \leq x \leq 300$

In general, to find a family of skewing schemes, one attempts to find a pattern in the number of memory modules. If a pattern is located, one then examines all the related skew factors and tries to find a pattern. If a pattern is found, one then attempts to prove that this pattern of skews represents a family of skewing schemes by proving that the elements in the p-diamond are distinct. It would be of substantial interest to find a mechanical technique for finding such patterns. By examining Tables 3.5 and 3.6, one realizes that one cannot arbitrarily choose a value for n above the minimum when implementing a skewing scheme. All n values, even if larger than the upper bound, do not realize a construction because of constraints on the relationship of the skew value, the n value and the set of templates. Furthermore, the aforementioned techniques do not seem to be helpful in identifying additional linear skewing schemes for $p_{\mathrm{lrcf},v}$ squares.

3.3 Additional Implications

In addition to the above results, there are additional implications when one is dealing with $(x, v) \times (y, v)$ rectangular subarrays because of the theorem regarding shift-boxes which allows them to be stretched, in particular Theorem 2.1.6 and because of the lemma which allows the shifts to hold, in particular, Lemma 2.1.7. Recall from Wijshoff that two blocks of equal size n and equal stretch can not be stored in n memory modules [Wij89].

Define a $(x, y)_v$ *conflict-free latin square** to be a (x, y) conflict-free latin square with stretch v.

Lemma 3.3.1 *There is a $(x, y)_v$ conflict-free latin square of order n with $n = xy+p$ for $1 \leq p \leq y$.*

x	Difference above minimum n																
2	**2**	3	4		6	7	8	9		11	12	13	14	15	16	17	18
3	2		**4**	5	6		8	9	10	11	12	13	14	15	16	17	18
4			**4**		6	7	8	9			12	13	14	15	16		18
5	**2**		4		6		8	9	10		12		14		16	17	18
6			**4**		6		8		10	11	12	13	14	15	16	17	18
7			**4**		6				10		12	13	14		16		18
8	**2**		4				8		10		12		14	15	16	17	18
9			**4**		6		8				12		14		16	17	18
10			**4**		6				10		12				16		18
11	**2**				6		8		10		12				16		18
12			**4**		6						12		14		16		18
13			**4**		6				10		12				16		18
14	**2**		4				8				12		14		16		18
15			**4**						10		12		14		16		18
16			**4**		6				10		12				16		18
17	**2**		4		6		8				12		14		16		18
18			**4**		6				10		12				16		18
19			**4**		6						12				16		18
20	**2**				6		8		10		12				16		
21			**4**						10		12				16		18
22			**4**								12				16		18
23	**2**		4		6		8		10		12		14		16		18
24			**4**		6						12				16		18
25			**4**		6				10		12				16		18

Table 3.5: Additional skewing schemes found for even p

x	Difference above minimum n																
x	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	**3**		5		7		9		11		13		15		17		19
3			**5**		7				11		13				17		19
4			5		**7**		9		11		13		15		17		19
5							**9**		11						17		19
6					7				**11**		13				17		19
7							9		11		**13**		15		17		19
8							9		11				**15**		17		
9									11						**17**		19
10									11		13				17		**19**
11													15		17		

Table 3.6: Additional skewing schemes found for odd p

Proof: By Theorem 2.1.6 their shift-box equivalent may be stretched. By Lemma 2.1.7 their shifts are valid if the blocks are stretched. The non-stretch proofs are presented in [CH92]. □

Define a $\{x,y\}_v$ *conflict-free latin square** to be a latin square in which no element appears more than once in any $(x,v)\times(y,v)$ **or** $(y,v)\times(x,v)$ subarray as well as the rows or columns. Table 3.7 displays extensions to the results from [CH92] which follow immediately from Theorem 2.1.6 and Lemma 2.1.7 for v relatively prime to n.

relations for x and y	n value
$x > y+1$	$n \geq xy+y$
$2 \leq y < x < 2y$	$n \geq xy+(2y-x)(x-y)+1$
$x > y$	$n \geq xy+y$
$x > y \geq 2$ and $x \equiv 1 \pmod{y}$	$n = xy+y$
$x > y \geq 2$ and $gcd(x,y)=1$	$n = y(x+y-1)$

Table 3.7: Results for blocks of equal size and stretch

Lemma 3.3.2 *When $x > 4$ is even, there is no $\{x,2\}_v$ conflict-free latin square of order $2x+2$ or $2x+3$.*

Proof: As before, by Theorem 2.1.7, Lemma 2.1.6 and by [CH92]. □

Lemma 3.3.3 *If $x \not\equiv 1 \pmod{y}, x \geq 2y-1, x > y \geq 3$, there is no $\{x,y\}_v$ conflict-free latin square of order $xy+y$.*

Proof: As before, by Theorem 2.1.7, Lemma 2.1.6 and by [CH92]. □

Chapter 4

Constant-Area Subarrays

The previous chapters all concern constant-perimeter rectangular subarrays. These results can be used for lower bounds for conflict-free access to constant-area rectangular subarrays. Consider the problem of conflict-free storage of all $\alpha \times \beta \leq z$ contiguous rectangular subarrays of an $n \times n$ array. Such a square is said to be *area conflict-free* $(z_{acf})^*$. If the square is also latin, the square is said to be area latin conflict-free $(z_{\text{alcf}})^*$. This problem has a set of templates whose size is a function of the area.

Consider all rectangular subarrays of fixed area z. Now choose a perimeter p satisfying $p^2 - 1 \leq 4z < (p+1)^2$. Every rectangular subarray of perimeter p has area at most z. Thus, we can apply our results to obtain a lower bound, using a $\frac{p}{2}{\text{lrcf}}$ square when p is odd. Figure 4.1 presents a comparative graph. The previous lower bound results from work by Colbourn and Heinrich [CH92]. The linear results for constant-area rectangular subarrays are found by brute force and represent the minimal linear skewing schemes for constant-area rectangular subarrays plotted at every change in skew factor and number of memory modules. The new lower bound represents the results for constant-perimeter rectangular subarrays plotted at their z values. The actual computation values for minimal linear skewing schemes are located in Tables 4.1 and 4.2. This table presents the minimum and maximum z value that works for a given shift value, s, and a given n.

4.1 New Upper Bound

We wish to find an upper bound on n, the number of memory modules needed to store z_{acf} squares. In order to obtain a rough upper bound, we consider a single element, the *star-element.* Next we consider every other element that may be possibly involved in a template instance with our chosen element. These elements together with the star element form the *star-diamond.* An example of a star-diamond is shown in Figure 4.2.

Lemma 4.1.1 *There are at most* $4z\ln z + O(z)$ *elements in the star-diamond.*

Proof: Partition the star-diamond into three pieces, the center row, the portion above the center and the portion below the center. The center row has $2z - 1$ elements. The portion below and the portion above have the same number of elements. The number of elements in the portion above is less then $\sum_{i=1}^{z-1}(2i-1)(\frac{z}{i} - \frac{z}{i+1})$ where $2i - 1$ defines the width and $(\frac{z}{i} - \frac{z}{i+1})$ defines the height for a given segment of the star-diamond. Thus, counting the elements in the star-diamond, we see that

$$n \leq 2z - 1 + 2\sum_{i=1}^{z-1}(2i-1)(\frac{z}{i} - \frac{z}{i+1})$$

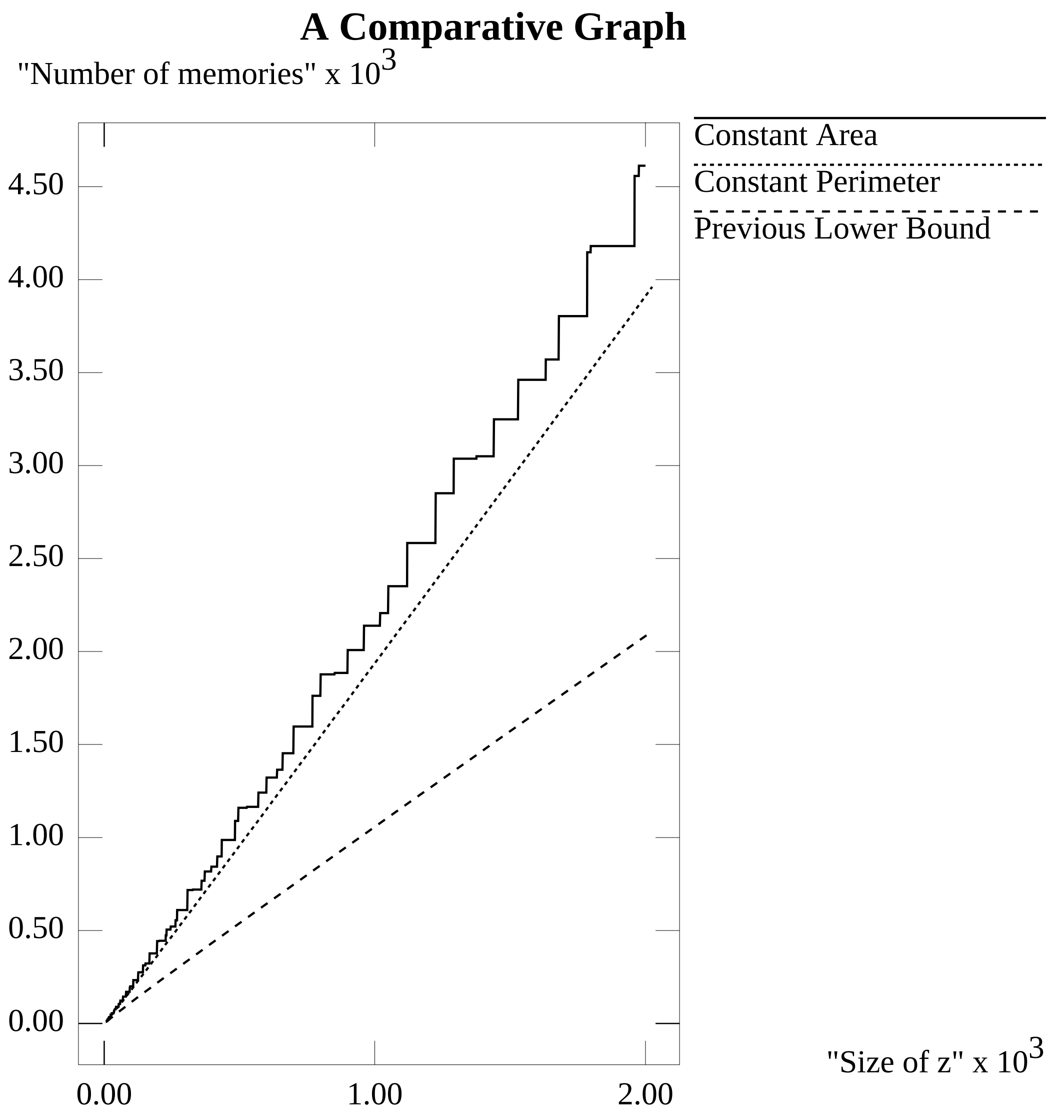

Figure 4.1: Constant-perimeter vs. constant-area

z	s	n	n/z	z	s	n	n/z	z	s	n	n/z
6	3	8	1.25	54	29	105	1.94	195	144	377	1.93
7	3	8	1.14	59	29	105	1.22	196	131	443	2.26
8	5	12	1.5	60	34	123	2.05	203	131	443	2.17
9	5	13	1.44	69	34	123	1.78	204	123	445	2.18
11	5	13	1.18	70	55	144	2.06	227	123	445	1.96
12	8	21	1.75	80	55	144	1.82	228	131	474	2.07
15	8	21	1.41	81	47	170	2.09	230	131	474	2.06
16	8	29	1.81	94	47	170	1.81	231	212	505	2.19
17	8	29	1.71	95	81	193	2.03	245	212	505	2.06
18	13	34	1.88	96	55	199	2.07	246	144	521	2.12
23	13	34	1.42	107	55	199	1.86	263	144	521	1.98
24	13	47	1.96	108	89	233	2.16	264	212	555	2.10
27	13	47	1.52	125	89	233	1.86	269	212	555	2.06
28	21	55	1.96	126	81	274	2.17	270	233	610	2.25
35	21	55	1.24	127	81	274	2.15	307	233	610	1.98
36	18	65	1.81	128	76	275	2.15	308	212	717	2.33
37	18	65	1.75	143	76	275	1.92	327	212	717	2.19
38	31	74	1.95	144	131	312	2.17	328	199	720	2.20
39	21	76	1.95	152	131	312	2.05	359	199	720	2.01
43	21	76	1.76	153	89	322	2.10	360	212	767	2.13
44	34	89	2.02	167	89	322	1.93	371	212	767	2.06
53	34	89	1.67	168	144	377	2.24	372	343	817	2.19

Table 4.1: Constant area constructions for $6 \leq z \leq 372$

z	s	n	n/z	z	s	n	n/z	z	s	n	n/z
395	343	817	2.06	699	555	1453	2.70	1291	788	2851	2.21
396	233	843	2.13	700	610	1597	2.20	1292	898	3037	2.21
417	233	843	2.02	769	610	1597	2.08	1375	898	3037	2.21
418	343	898	2.15	770	487	1762	2.28	1376	843	3050	2.22
434	343	898	2.06	799	487	1762	2.20	1439	843	3050	2.12
435	377	987	2.27	800	555	1877	2.35	1440	898	3249	2.26
483	377	987	2.04	851	555	1877	2.21	1529	898	3249	2.12
484	301	1089	2.25	852	521	1885	2.21	1530	1453	3461	2.26
495	301	1089	2.20	899	521	1885	2.10	1631	1453	3461	2.12
496	343	1160	2.34	900	555	2008	2.23	1632	987	3571	2.19
527	343	1160	2.20	959	555	2008	2.09	1679	987	3571	2.13
528	322	1165	2.21	960	898	2139	2.23	1680	1453	3804	2.26
569	322	1165	2.05	1019	898	2139	2.10	1784	1453	3804	2.13
570	343	1241	2.18	1020	610	2207	2.16	1785	898	4147	2.32
599	343	1241	2.07	1049	610	2207	2.10	1797	898	4147	2.31
600	555	1322	2.20	1050	898	2351	2.24	1798	1597	4181	2.31
638	555	1322	2.07	1119	898	2351	2.10	1959	1597	4181	2.13
639	377	1364	2.13	1120	987	2584	2.31	1960	987	4558	2.33
659	377	1364	2.07	1224	987	2584	2.12	1975	987	4558	2.31
660	555	1453	2.13	1225	788	2851	2.33	1976	1275	4613	2.33
								2000	1275	4613	2.31

Table 4.2: Constant area constructions for $372 \leq z \leq 2000$

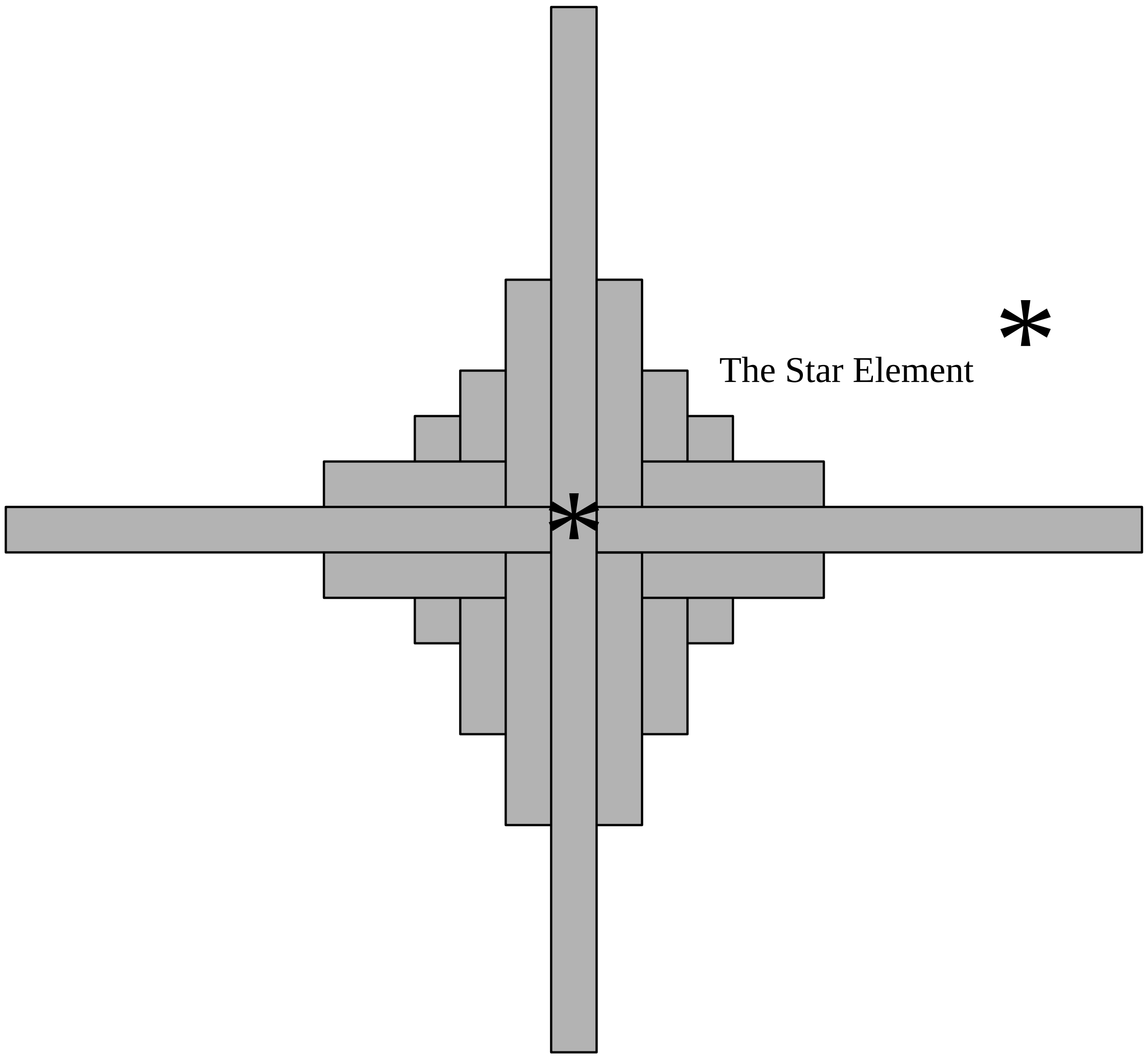

Figure 4.2: The star-diamond

$$\begin{aligned}
&= 2z - 1 + 2z\sum_{i=1}^{z-1}\frac{2i-1}{i(i+1)} \\
&= 2z - 1 + 2z[\sum_{i=1}^{z}\frac{2}{i+1} - \sum_{i=1}^{z}\frac{1}{i(i+1)}] \\
&\leq 2z - 1 + 2z[\sum_{i=1}^{z}\frac{2}{i+1} \\
&\leq 2z - 1 + 4z\ H_z \\
&\leq 2z - 1 + 4z\ln z + O(1) \\
&\leq 4z\ln z + O(z)
\end{aligned}$$

H_z is defined to be the z^{th} harmonic. See [GKP89] for more details. Thus, if we count these elements, we get an upper bound on the number of elements of $4z\log z + O(z)$. □

Theorem 4.1.2 *There exists a constant c and an n_0 such that if $n \geq n_0$ and $n \geq cz\log z$ one can always provide a z_{acf} square (that is, n is $\Omega(z\log z)$).*

Proof: Relating this problem to graph-colouring, we let every element in the square be a vertex. We place an edge between two vertices if and only if the two elements are involved in a template instance. Thus, Lemma 4.1.1 provides an upper bound on the maximum degree of a vertex in the graph. The upper bound on the degree provides an upper bound on the chromatic number of the graph. Since the chromatic number is equivalent to the minimum number of memory modules for conflict-free storage, we have an upper bound on n by Brooks's Theorem [CL86]. □

This is a large improvement over the previous results of $O(z^2)$ which can be found using d-Manhattan results by Colbourn and Heinrich [CH91].

4.1.1 Graph-Colouring Method

In general, we can always relate the conflict-free problem to graph-colouring to attempt to obtain an upper and lower bound for a given set of templates. First, we let every element in the square be a vertex. We place an edge between two vertices if and only if the two elements are involved in a template instance. Thus, every template instance forms a clique. Two vertices have an edge if and only if they are in a template instance together. By definition of the vertex-colouring problem, two vertices are not coloured the same colour if they are connected by an edge. Likewise, two elements should not be placed in the same memory module if they are contained in a common template instance. Thus, the chromatic number of the graph is equivalent to the minimum number of memory modules necessary to store the set of templates conflict-free. If one can find the chromatic number of the graph in general, one has found the minimum number of memory modules necessary. If not, one needs to look for bounds. In finding a pessimistic upper bound on the chromatic number of a graph, one often uses Brooks's Theorem. This implies bounding the maximum degree of the graph. The upper bound on the maximum degree then, through Brooks's Theorem, one obtains an upper bound. There are several techniques one could use to attempt to lower this bound such as classifying the graph, but this may be difficult. One technique that is frequently used involves dividing the graph into subgraphs. One can then independently colour the subgraphs, and sum the number of colours used on the subgraphs. Since the edges between the subgraphs are connected to the different colour groups used by the subgraphs, they can be ignored.

4.1.2 Skewing Using Permutations

Consider a permutation $p_1, p_2, \ldots, p_n$. Let p_a represent where a 1 is placed in row a of column p_a. Thus, one could use the permutation to define a skewing scheme. The rest of the elements of the row continue sequentially from the first, modulo n. One could then place restrictions on the permutation to represent the set of templates one desires to be conflict-free. For example, a permutation is considered to be conflict-free for all rectangular subarrays of area z with horizontal wrapping where $z =$ minimum over all α and β of $(\min(|p_\alpha - p_\beta|, n - |p_\alpha - p_\beta|))(|\alpha - \beta|)$. If one also desires vertical wrapping, then a permutation is considered to be conflict-free for all rectangular subarrays of area z with horizontal wrapping where $z =$ minimum over all α and β of $(\min(|p_\alpha - p_\beta|, n - |p_\alpha - p_\beta|))(\min(|\alpha - \beta|, n - |\alpha - \beta|))$

By the permutation scheme, the element in row i of the array and column $p_i + j$ is put into memory module $j + i \pmod{n}$. Thus, this is a compact storage scheme in which one only needs to store n elements for address generation, specifically the permutation.

Chapter 5

Conclusions

In order to provide conflict-free access to all constant-perimeter rectangular subarrays, one has to accommodate the entire set of templates. The size of this set is a function of the perimeter. In addition, the templates are all of different shapes. The set of templates further expands when one allows the rectangular subarrays to be stretched. In this thesis, we employ the p-diamond and the (x, y)-cut diamond as mega-templates which encapsulate the interactions of the templates contained in the set of constant-perimeter templates and thus, the essence of the constant-perimeter problem. This mega-template is employed not only to find lower bounds which apply to any type of skewing scheme, but is also helpful in finding a construction. This set of templates appears to be useful for various applications, satisfying **Point 2**, which requires that the templates accessible without memory conflicts reflect the needs of algorithms. In addition, the linear skewing schemes are efficient, which satisfies **Point 1**.

In 1989, Kim and Kumar [KK89, HKK92] showed that two memory cycles are necessary for the retrieval of $\sqrt{n} \times \sqrt{n}$ subsquares of an $n \times n$ square when stored in n

memory modules. Thus, one of the more surprising aspects of the results presented in this thesis is that all constant-perimeter rectangular subarrays can be accessed in two memory cycles when stored in σ memory modules where σ is the maximum size of a template in the set of constant-perimeter templates. This satisfies **Point 3**, which requires high memory utilization. Table 5.1 presents a summary of all the upper and lower bounds found for constant-perimeter rectangular subarrays with stretch v when v is relatively prime to n. While experimental data reveals that in most cases the upper bound for $(x,y)_{\mathrm{lrcf},v}$ squares is minimal for a linear skewing scheme, there are instances which can beat this upper bound. For example, the upper bound and the related construction provided for a $(2,3)_{\mathrm{lrcf},v}$ square uses 8 memory modules when one could store it in 7 using a skew factor of 3. Table 5.2 gives the constructions for the linear skewing schemes presented in this thesis. The order of the square is the upper bound given in Table 5.1. The research presented in Chapter 2 and Chapter 3 is summarized in [EC92b, EC93].

	p	x cond.	y cond.	lower bound	upper bound
$p_{\mathrm{lrcf},v}$	even		$y=x$	$2x^2-2x+1$	$2x^2-2x+1$
	odd		$y=x+1$	$2x^2$	$2x^2$
$(x,y)_{\mathrm{rcf}}$		odd		$xy+\frac{1}{2}x^2-x+\frac{1}{2}$	$xy+\frac{1}{2}x^2-x+\frac{1}{2}$
		even		$xy+\frac{1}{2}x^2-x$	$xy+\frac{1}{2}x^2-x$
$(x,y)_{\mathrm{lrcf},v}$		odd		$xy+\frac{1}{2}x^2-x+\frac{1}{2}$	$xy+\frac{1}{2}x^2-x+\frac{1}{2}$
		$4e$	$4e+2d+1$	$xy+\frac{1}{2}x^2-x+1$	$xy+\frac{1}{2}x^2-x+2$
		$4e$	$4e+2d$	$xy+\frac{1}{2}x^2-x+1$	$xy+\frac{1}{2}x^2-x+2e+1$
		$4e+2$		$xy+\frac{1}{2}x^2-x+1$	$xy+\frac{1}{2}x^2-x+y-1$

Table 5.1: Table of upper and lower bounds

	p	x cond.	y cond.	skew factor
$p_{\mathrm{lrcf},v}$	even		$y = x$	$2x - 1$
	odd		$y = x + 1$	$2x - 1$
$(x, y)_{\mathrm{rcf}}$		odd		$2y + x - 2$
		even		non-linear
$(x, y)_{\mathrm{lrcf},v}$		odd		$2y + x - 2$
		$4e$	$4e + 2d + 1$	$12e^2 + 4ed - 2d - 6e + 1$
		$4e$	$4e + 2d$	$4e$ (column)
		$4e + 2$		$4e + 2y - 1$

Table 5.2: Table of constructions

The results for constant-perimeter also imply a lower bound for constant-area rectangular subarrays as discussed in Chapter 4 (see also [EC92a]). An upper bound for the constant-area is also provided by a relationship of conflict-free access to the problem of determining the chromatic number of a graph.

Thus, we have presented techniques to find bounds for sets of templates. In the problems we examined, the number of templates is a function of the maximum size of a template. For $p_{\mathrm{lrcf},v}$ squares, these techniques have found tight bounds. The lower bounds apply to any type of implementation one may choose to employ. In addition, we have found linear skewing schemes which are efficient and satisfy **Point 2**. When linear skewing schemes are deemed to be acceptable, our constructions can be applied to solve the conflict-free access problem. Future work includes determining which interconnection networks can efficiently realize the skewing schemes presented and thus, how these skewing schemes satisfy **Point 4**.

5.1 Future Directions

In this thesis, we examined a set of template where the number of templates is not fixed. This problem is more complicated than examining a set of templates containing a fixed number. While the templates are related since they represent constant-perimeter rectangular subarrays, this is a positive step into developing bounds and constructions for arbitrary templates. We have used such tools as mega-templates and graph-colouring as techniques for finding skewing schemes as well as bounds on the number of memory modules necessary for conflict-free access. We have also proposed the use of permutations to define skewing schemes. As new research on the problem of the chromatic number of a graph is conducted, the results may have implications for skewing schemes. Future directions include finding additional tools for those desiring new skewing schemes. These tools need to help identify theoretical bounds for the desired sets of templates as well as constructions. The hope is to someday have a general framework that one could employ for arbitrary sets of templates of arbitrary types.

The problem of conflict-free access is based on shared-memory modules for computation. Future directions include a related problem for localized memory, in other words, when each processor has its own memory. In this situation we want to minimize communications costs. Let a template represent the portion of data we wish to process locally. While each processor may have more than one local memory module, we assume that all local memory operations have the same communication cost. Thus, the memory module number and the processor number are assumed to be equivalent. For example, if one only wants localized memory access to rows, then one stores each row in a different memory module/processor location. For this template and storage scheme, there are no communication costs. If a skewing-scheme for conflict-free access requires t memory cycles for a maximum template instance size of u, then $u - t$ communications are necessary for the communication problem in the second model. As with the conflict-free problem, one would desire to have high memory utilization as well as processor utilization. Storing the entire array in one memory module would solve the problem but lead to poor processor utilization. Thus, determining the optimal communication cost with respect to processor utilization is a problem for future research which may be aided by the unification of the problem with research regarding the conflict-free access problem. Following are two possible models for formulating the communication problem.

In the first model, one assumes that communication cost is determined by the number of different processors involved in communication. This model assumes that the communication to request and receive a singular element or several elements has the same cost. In this situation, we want to minimize the number of different symbols in a template instance. Thus, if a single template instance contains two distinct symbols which represent their location, how many times each symbol appears is irrelevant but the number of different symbols that appear is very relevant. In conclusion, this model seeks to minimize the number of different symbols appearing in a template instance.

In the second model, one assumes that the communication cost is a function of every element or piece of data that must be retrieved from non-local memory. The symbol appearing most frequently should represent which processor functions on that template instance. In this situation, we want to minimize the number of different symbols and the frequency with which they appear in a template instance. Thus, by putting an upper bound on the number of elements different from the dominating element appearing in a template instance, we have a bound on the communication cost.

The problem of conflict-free access and the aforementioned communication problem have structural similarities but opposite goals. Conflict-free templates require that the elements in an instance be as different as possible while communication templates want the elements in an instance to be as similar as possible. The connection of the conflict-free access problem to that of the chromatic number of a graph is one particular technique presented in this thesis which may be adaptable. In particular, form a graph for the communication templates in the same manner as we did for conflict-free templates. The complement of the graph has an edge between two vertices if and only if those two vertices are not involved in a template instance. Thus, those vertices may be stored in separate memory modules. Observe that these two vertices do not have to be stored in separate memory modules and thus, the communication problem is not exactly solved by colouring the complement of the graph. One could also observe that both models ignore the topology of the processing array. Future directions include expanding the two models to accommodate the topologies, specifically, different locations having different communication costs.

Finally, one could have an algorithm that desires to employ conflict-free templates and their communication counterparts. Thus, we need a model that works with communication templates as well as conflict-free templates. The two sets of templates have very different objectives but both must be accommodated simultaneously. The skewing scheme would need to minimize memory conflicts on one set of templates and minimize communication cost for another set of templates. This might be accomplished by finding the complement of the communication graph, and then adding in the restrictions of the conflict-free graph.

Appendix A

Terms

- *area conflict-free (z_{acf})*: a square that is conflict-free for all $\alpha \times \beta \leq z$ contiguous rectangular subarrays of an $n \times n$ array.
- *area latin conflict-free (z_{alcf})*: a square that is conflict-free for all $\alpha \times \beta \leq z$ contiguous rectangular subarrays of an $n \times n$ array as well as all rows and columns.
- *block template*: a $(p_1, v_1) \times (p_2, v_2) \times \ldots \times (p_d, v_d)$-block template B on a d-dimensional array is the set $\{(i_1v_1, i_2v_2, \ldots, i_dv_d) | 0 \leq i_1 < p_1, 0 \leq i_2 < p_2, \ldots, 0 \leq i_d < p_d\}$ [Wij89].
- *designated element* : the first element of the template instance when the template is represented as a list [Sha78]. Also referred to as the handle.
- *diagonal latin square*: consists of the symbols 0 to $(n-1)$. The same symbol never appears more than once in the same row, column or any of its two main diagonals [DK74].
- *dilating*: accomplished by replacing each line of an interconnection network with l lines. Such a network would have a dilating factor of l [Roo91].

- *handle* : the first array element in a template definition whose location is specified. All other array element's locations are specified with respect to this handle. If the handle is not specified it is assumed to the leftmost element in the uppermost row.

- *interconnection network*: the network that connects the independent memory modules and processors.

- *latin square of order n*: an $n \times n$ array which consists of the symbols 0 to $(n-1)$. The same symbol never appears more than once in the same row or column [DK74].

- *linear skewing scheme*: a skewing scheme that maps the array elements $a_{i,j} \equiv q(i-\alpha) + r(j-\beta) \pmod{n}$ for some fixed integers p and q [Kuc68].

- *main subsquare*: an $\sqrt{n} \times \sqrt{n}$ subarray $S_{i,j}$, such that the top left cell is (i,j) where $i \equiv 0 \bmod \sqrt{n}$ and $j \equiv 0 \bmod \sqrt{n}$ in an $n \times n$ array [HKK92].

- *Manhattan distance*: the entries (i,j) and (k,l) have a Manhattan distance defined as $|i-k| + |j-l|$ [CH92].

- *d-Manhattan latin square* : a square that does not contain two identical entries whose Manhattan distance is less than $2d+1$ [CH92].

- *mega-template*: a set of elements which are forced to be distinct by the set of templates we wish to be conflict-free. It is used to encapsulate the interactions between a set of templates.

- *memory conflict*: when more than one processor requests information from a single memory module during a single memory cycle.

- *memory utilization*: the percentage of memory modules being accessed for the retrieval of an arbitrary template instance.

- *multi-stage interconnection network*: an interconnection network which results from joining two or more copies of a given network. The networks are joined by having the output lines of one connected to the input lines of another network. Each copy is referred to as a stage.

- *perfect latin square of order n^2*: a diagonal latin square such that no symbol appears more than once in any main subsquare [HKK92].

- *perimeter rectangular conflict-free* (p_{rcf}): a square that is conflict-free for all rectangular subarrays whose perimeter is less than or equal to $\gamma = 2p$.

- *perimeter rectangular conflict-free square with stretch v* ($p_{\mathrm{rcf},v}$): a square that is conflict-free for all rectangular subarrays with perimeter less than or equal to $\gamma = 2p$ and a stretch of v. The perimeter does not include elements skipped because of the stretch.

- *perimeter latin rectangular conflict-free* (p_{lrcf}): a square that is conflict-free for all rectangular subarrays whose perimeter is less than or equal to $\gamma = 2p$ and all rows and columns.

- *perimeter latin rectangular conflict-free square with stretch v* ($p_{\mathrm{lrcf},v}$): a square that is conflict-free for all rectangular subarrays with perimeter less than or equal to $\gamma = 2p$ and a stretch of v. In addition, it must be conflict-free for all rows and columns.

- *permutation capability*: the percentage of all possible n permutations of the inputs to the outputs that an interconnection network can realize.

- (t,s,r,v) *shift-box* is $S_v(1,t:1,s) \cup S_v(t+1:1,r)$.

- *stretch*: the stretch of a block template as defined above is $(v_1, v_2, \dots v_d)$ [Wij89]. For a two dimensional block, one takes every v_1th element from every v_2th row. If subscripts are not specified, they are assumed to be equal. If no stretch is specified, it is assumed to be equal to 1.

- *rearrangeable*: an interconnection network than can realize all possible permutations.

- *recirculate*: a permutation-increasing operation which involves connecting the output lines of an interconnection network to the input lines and sending the data though the network multiple times.

- *skewing scheme*: the function that maps the array elements into the memory modules. A skewing scheme may provide conflict-free access to a template or a set of templates.

- *stretch blocks*: blocks whose stretch is greater than one.

- *template*: consists of a distinguished position called the handle and possibly other array positions defined by their relative location with respect to the handle. If the actual position of the handle is not specified it is assumed to be the leftmost element of the uppermost row.

- *template instance*: a specific translate of a template such as row one.

- (x,y) *conflict-free latin square*: a latin square in which no element appears more than once in any $x \times y$ subarray as well as the rows or columns [CH92].

- $(x, y)_v$ *conflict-free latin square*: a latin square in which no element appears more than once in any $(x, v) \times (y, v)$ subarray as well as the rows or columns [CH92].
- $\{x, y\}$ *conflict-free latin square*: a latin square in which no element appears more than once in any $x \times y$ **or** $y \times x$ subarray as well as the rows or columns.
- $\{x, y\}_v$ *conflict-free latin square*: a latin square in which no element appears more than once in any $x \times y$ **or** $(y, v) \times (x, v)$ subarray as well as the rows or columns.

Bibliography

[Bat77] K.E. Batcher. The multidimensional access memory in STARAN. *IEEE Trans. Computers*, C-26:174–177, 1977.

[Ben65] V.E. Benes. *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, New York, 1965.

[BK71] P. Budnik and D. J. Kuck. The organization and use of parallel memories. *IEEE Trans. Computers*, C-20:1566–1569, 1971.

[BR88] R. Boppana and C. S. Raghavendra. On self routing in Benes and shuffle exchange networks. In *Proc. International Conference in Parallel Processing*, volume 1, pages 196–200, 1988.

[Bro91] A. Brodnik. Theoretical limits on interconnection networks. private communication, 1991.

[CH91] C. J. Colbourn and K. Heinrich. Conflict free access to parallel memories. Research Report, Center for Systems Sciences, Simon Fraser University, January 1991.

[CH92] C. J. Colbourn and K. Heinrich. Conflict free access to parallel memories. *Journal of Parallel and Distributed Computing*, 14:193–200, 1992.

[Chu88] E. Chu. *Orthogonal Decomposition of Dense and Sparse Matrices on Multiprocessors.* PhD thesis, University of Waterloo, 1988. Department of Computer Science.

[CL86] Gary Chartrand and Linda Lesniak. *Graphs and Digraphs.* Wadsworth, Inc., 1986.

[CM85] V. Cherkassky and M. Malek. On permuting properties of regular rectangular sw-banyans. *IEEE Trans. Computers*, C-34(6):542–546, 1985.

[Dan91] S. P. Dandamudi. *Hierarchical Hypercube Multicomputer Interconnection Networks.* Ellis Horwood, 1991.

[DH91] F. Dehne and S.E. Hambrusch. Parallel algorithms for determining k - width connectivity in binary images. *Journal of Parallel and Distributed Computing*, 12:12–23, 1991.

[DK74] J. Dénes and D. Keedwell. *Latin Squares and Their Applications.* Academic Press, 1974.

[EC92a] D. L. Erickson and C. J. Colbourn. Conflict-free access to rectangular subarrays. *Congressus Numerantium*, 1992. To Appear.

[EC92b] D. L. Erickson and C. J. Colbourn. Conflict-free access to rectangular subarrays with constant perimeter. Preprint, 1992.

[EC93] D. L. Erickson and C.J. Colbourn. Conflict-free access for collections of templates. In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, 1993. To Appear.

[Fei82] M. Feilmeier. Parallel numerical algorithms. In D. J. Evans, editor, *Parallel Processing Systems*, pages 285–338. Cambridge University Press, 1982.

[Fen81] T. Y. Feng. A survey of interconnection networks. *Computer*, 14(12):12–27, 1981.

[GKP89] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1989.

[HKK92] K. Heinrich, K. Kim, and V. K. Kumar. Perfect latin squares. *Discrete Applied Mathematics*, 37/38:281–286, 1992.

[KK89] K. Kim and V. K. P. Kumar. Perfect latin squares and parallel memory access. In *Proc. Sixteenth Int. Symp. Computer Architecture*, pages 372–379, 1989.

[Knu92] D.E. Knuth. Two notes on notations. *The American Mathematical Monthly*, 99(5):403–422, May 1992.

[Kuc68] D. J. Kuck. ILLIAC IV software and application programming. *IEEE Trans. Computers*, C-17:758–770, 1968.

[Law75] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Trans. Computers*, C-24:1145–1155, 1975.

[Lee88] D. Lee. Scrambled storage for parallel memory systems. In *Int. Symp. Computer Architecture*, pages 232–239, 1988.

[Len78] J. Lenfant. Parallel permutations of data: A Benes network control algorithm for frequently used permutations. *IEEE Trans. Computers*, C-27(7):637–647, 1978.

[LV82] K.H. Lawrie and C.R. Vora. The prime memory system for array access. *IEEE Trans. Computers*, C-31:435–442, 1982.

[MM80] M. Malek and W. W. Myre. Figures of merit for interconnection networks. In *Proc. Workshop on Interconnection Networks for Parallel and Distributed Processing*, pages 74–83, 1980.

[NS81] D. Nassimi and S. Sahni. A self routing Benes network and parallel permutation algorithms. *IEEE Trans. Computers*, C-30(5):332–340, 1981.

[RM85] R. Ransom and R.J. Matela. *Computers in Biology: An Introduction.* Open University Press, 1985.

[Roo91] A. Rooks. Routing and cost effectiveness of directly-connected interconnection networks. Master's thesis, University of Waterloo, 1991. Department of Electrical and Computer Engineering.

[RP91] A. Rooks and B. Preiss. A unifying framework for distributed routing algorithms. Preprint, 1991.

[SD90] M. Shiva and D.v. DeYong. Enhancement of single pixel targets using two-dimensional digital filters. In *IEEE PLANS 90: Position Location and Navigation Symposium Record*, pages 231–239, 1990.

[Sha78] H. D. Shapiro. Theoretical limitations on the effective use of parallel memories. *IEEE Trans. Computers*, C-27:421–428, 1978.

[Sie85] H. J. Siegel. *Interconnection Networks for Large-Scale Parallel Processing.* Lexington Books, 1985.

[SMJ92] W. Sung, S. K. Mitra, and B. Jeren. Multiprocessor implementation of digital filtering algorithms using a parallel block processing model. *IEEE Trans. Parallel and Distributed Systems*, 3(1):110–120, 1992.

[Szy89] T. Szymanski. On the permutation capability of a circuit-switched hypercube. In *Proceedings of the International Conference on Parallel Processing*, pages 1103–1110, 1989.

[Wij89] H. A. G. Wijshoff. *Data Organization in Parallel Computers.* Kluwer Academic Publishers, 1989.

[YL81] P.C. Yew and D. Lawrie. An easily controlled network for frequently used permutations. *IEEE Trans. Computers*, C-30(4):296–298, 1981.