On Shift Register Realization of Sequential Circuits II

Shunichi Toida

July 11, 1997

Abstract

This is the second of a series of intended reports on realization of sequential machines with shift registers and on properties of shift registers. Some of the pathological cases for the shift register realization algorithms **SRR/GSRR** reported in the previous report are investigated here.

One of the pathological cases turned out to be simple cycles, and based on the work of Yoeli, an algorithm has been developed for finding a shift register code for them. Another case is simple cycles with one chord. It is shown that they can be realized with a shift register if their length n is a power of 2, and the cycle determined by their chord is of length $n/4$ or less, or equal to $n/2$.

Also the realizability of sequential machines with bidirectional shift registers is studied. A necessary and sufficient condition has been found for the case when the number of states is a power of 2.

1 Introduction

With the increasing degrees of integration of logic circuits, the automatic generation of tests for detecting faults in logic circuits is becoming more and more time consuming. For sequential circuits, although there are a number of test generation methods such as (1) scan methods, (2) time domain expansion methods, (3) BIST, and (4) design for test (other than scan or BIST) [1], none of them seems to be completely satisfactory at the moment. For the scan methods, the performance degradation as well as the area overhead is generally considered unsatisfactory. The time domain expansion tends to take too much time, and BIST generally does not give good fault coverage. That is hardly unexpected because the test generation problem for sequential circuits is NP-hard [6]. The last possibility, the design for test, is currently under active investigation by a number of researchers. For combinational logic circuits, a number of good algorithms have been developed for detecting stuck-at faults [12, 13, 22] and they seem to be performing satisfactorily at present. But, since the test generation problem is computationally intractable, as the circuits become larger and more complex, new test generation methods are going to be required.

Considering the difficulty of the problem of test generation for sequential circuits, design for test seems to be one of the most promising approaches to test generation. Though there are some interesting works on design for test of sequential circuits by Devadas et al. [7], and Park and Menon [20] for example, it is one of the least investigated areas at the moment [17, 24].

It has long been noted, for example in [18], that the test generation for sequential circuits is easier if all the feedback loops are cut. Cheng and Agrawal [3, 4, 5] observed that for sequential circuits with many feedback loops between the memory elements, the test generation tends to take a lot of time. To alleviate this problem they have proposed a partial scan method in which a fraction of the memory elements are selected for scan so that all the feedback loops are broken. This method makes the test generation easier without incurring much area overhead. Along the same lines, they have also suggested a state assignment method which minimizes the number of feedbacks between the memory elements for finite memory machines [3]. Their works suggest (among others) that a realization of a sequential circuit with a shift register(s) makes test generation easier, because there are not many feedbacks between the memory elements, and all the feedbacks are to a single memory element in a feedback shift register. In addition, if a sequential circuit can be realized with shift registers, then very little extra hardware is necessary for scanning the memory elements. They also suggest embedding shift registers (more generally a testing machine) into a sequential circuit [2]. In this method, a subset of transitions of a given sequential circuit is realized with shift registers and the rest are added to the shift registers. When shift registers are embedded in a sequential circuit, a scan circuit is already built in, making test generation and testing easy. Their results on MCNC benchmark circuits show promise for this approach. In the light of these works the shift register realization of sequential circuits, which was studied by several researchers in the past [10][11] [16] [18] [19] ([18] also contains a list of other works), is very interesting and requires reexamination from the testing point of view.

In the previous report by this author [23] a necessary and sufficient condition for a sequential machine to be realizable with multiple binary shift registers is formulated and proved.

Based on this condition a randomized algorithm was developed which finds a smallest set of transitions to be removed from a given sequential machine to make it realizable with shift registers. Also a randomized algorithm based on P tree by Roome and Torng[21] was obtained which finds longest shift registers for a given sequential machine known to be realizable with shift registers. The results of computational experiments on MCNC benchmark circuits show that more than 50% of the transitions can be realized as those of a shift register. However, there is no known efficient algorithm to realize a sequential machine with a given state transition diagram even if it is known to be realizable with shift registers. In fact the complexity of the problem is not known at the moment. The algorithm GSRR in [23] finds shift registers if a given sequential machine is realizable with shift registers. However, in the worst case it takes an exponential amount of time. The worst case occurs when the blocks E^i_j of partition E^i can not be identified uniquely. In the case of realization with a single shift register, for example, this means that every state of G^i is mapped to a single state by the transition function and that every state must be paired with another state to find the right pairing to construct G^{i+1} from G^i.

In this report it is noted that some of the worst cases for Algorithm **GSRR** can be identified and methods are going to be given to find shift registers for them without going through the exhaustive search. In particular if every state of G^i is mapped to a single state by the transition function but if G^i is connected, then G^i is a simple cycle. In this case G^i is shown to be realizable with a shift register. Also if G^i is a simple cycle with a chord, then for some cases it can also be realized with a shift register.

In Chapter 2 following this some basic concepts used in this paper are defined. Then in Chapter 3 the relevant results from the previous report are summarized, in particular the algorithm for finding shift registers, algorithm SRR, is reviewed. One of the pathological cases, simple cycles, is discussed in Chapter 4, and an algorithm for finding a shift register coding for a simple cycle is developed. In Chapter 5 another of the worst cases for the algorithm SRR, simple cycles with a chord, is investigated. Then in Chapter 6 a realization with bidirectional shift registers is studied. Chapter 7 is the conclusion and discusses some of the unsolved problems as well.

2 Preliminaries

A **partition** P of a set S is a collection of disjoint subsets of S such that the union of the subsets is equal to S. The subsets are called **blocks** of partition P. A partition of a set S is denoted by $\underline{0}$ if all of its blocks are a singleton, and by $\underline{1}$ if it has exactly one block. Let P and Q be partitions of a set S. Then the **sum** of P and Q, denoted by $P+Q$, is defined as the partition such that every pair of elements x and y of S are in the same block of $P+Q$ iff there is a sequence $x_0 = x$, x_1, ..., $x_k = y$, of elements of S, where for all $i < k$, either x_i and x_{i+1} are in the same block of P or Q. The **product** of P and Q, denoted by $P * Q$, is defined as the partition such that every pair of elements x and y of S are in the same block of $P * Q$ iff the are in the same block in P and also in Q. A partition Q is a **refinement** of a partition P, denoted $P \geq Q$, if and only if each block of Q is a subset of a block of P. A sequence of arcs of a directed graph(digraph) with their endpoints is called a **walk** from the first vertex to the last vertex, if the head of an arc is the tail of the next arc in the

sequence for every arc. A walk is **closed** if the tail of its first arc is the same as the head of the last arc. If no arc appears more than once in a walk, the walk is called a **trail**. If no vertices appear more than once in a trail, the trail is called a **path**. A closed path is called a **cycle**. An arc from one vertex of a cycle to another is called a **chord**. A digraph is called an **Euler graph** if the in-degree is equal to the out-degree for each of its vertices. A digraph is **strongly connected**, if there is a path between any pair of vertices in both directions. A labeling of the vertices of a digraph G is called a **coding** for G, and a label for a vertex of G is called a **code word** for the vertex. A digraph G is said to be **realizable with m shift registers** if every vertex of G can be labeled with a concatenation of m strings of possibly different lengths of symbols 0 and 1, called **factors**, in the following way:
There is an arc from vertex u to vertex v of G if and only if the label of v is obtained by shifting each factor of the label of u to the left one position and inserting 0 or 1 into the rightmost position of each resulting factor. Such a labeling is called a **shift register coding** for the digraph. The factors in a code word are assumed to be ordered in some order. A graph with n vertices and with arcs in either direction between every pair of the vertices including self-loops is denoted by CD_n. In this paper unless otherwise specified a simple **graph** means a strongly connected graph. Hence it is assumed that a sequential circuit does not have transient states.

Definition 1 *Given a graph G, relations C and R are defined on the vertex set V of G as follows:*
For every u and $v \in V$, $(u, v) \in C$ iff there is a vertex $w \in V$ such that arcs (u, w) and (v, w) exist in G.
Let T_c denote the transitive closure of C.
Similarly for every u and $v \in V$, $(u, v) \in R$ iff there is a vertex $w \in V$ such that arcs (w, u) and (w, v) exist in G.
Let T_r denote the transitive closure of R. If the graph needs to be specified for T_r, $T_r(G)$ is used for T_r.

The following lemma can easily be seen from the definition of T_c and T_r.

Lemma 1 *T_c and T_r are an equivalence relation on V.*

Let us denote by P_c and P_r the partitions induced by T_c and T_r, respectively.

Let S be the state set of a sequential machine with the transition function δ. Let P and Q be partitions of S with m blocks. P is said to **co-map** into Q, written $[P, Q]$, iff the blocks of P and Q are ordered so that $\delta(p_i) \subseteq q_i$ and $\delta^{-1}(q_i) \subseteq p_i$ for all $i \leq m$. Here p_i and q_i denote a block of P and Q, respectively, and $\delta(p)$ for a block p of P or Q is the union of the images of p under δ over the set I of all input symbols, that is $\delta(p) = \{x : \exists s \in S \exists i \in I \delta(s, i) = x\}$.

3 Summary of Relevant Results of Previous Report

3.1 Realizability Condition

Not all sequential machines are realizable with shift registers except the trivial realization of using only shift registers of length 1. In this chapter first a necessary and sufficient condition is proved for a sequential machine to be realizable with binary shift registers at least one of which is of length 2 or larger. This condition is then expressed in terms of a bipartite graph, and a randomized algorithm is developed to obtain the smallest number of transitions to be removed to make a given sequential machine realizable with shift registers. The results of a computational experiment using this algorithm are presented.

It is assumed that a sequential machine does not have any transient states and that a transition graph is strongly connected.

Theorem 1 *If a digraph $G(V, A)$ is realizable with shift registers at least one of which is of length 2 or larger (simply realizable with shift registers hereafter), then $P_c * P_r = \underline{0}$.*

The converse of Theorem1 is trivially true, if a multivalued shift register is used. It can, however, also be proven true, when shift registers are restricted to binary ones.

Theorem 2 *If $P_c * P_r = \underline{0}$ for a graph $G(V, A)$, then G is realizable with binary shift registers.*

3.2 Algorithm for Finding Shift Registers

Here an algorithm for finding shift registers is presented when the given sequential machine is known to be realizable with shift registers. Since only realizations with one shift register are considered in htis report, only Algorithm **SRR** is given here.

The algorithm finds shift registers by producing a coding of states such that for any transition the code word of the destination is obtained by shifting that of the source to the left by one bit. Each bit of this coding is found by successively obtaining homomorphs of the state diagram of the given sequential machine. Let us start with the definition of this homomorphism.

Definition 2 *A* **k-th order homomorph** *G^k of a graph G is defined recursively as follows:*
(1) $G^0 = G$.
(2) For i = 1 to k, partition the vertex set V^{i-1} of G^{i-1}(denote the partition by E^i) so that

(a) for any two vertices u and $v \in V^{i-1}$, if $(u, v) \in T_r(G^{i-1})$, then u and v are in one block of the partition E^i,

(b) for each block E^i_j of partition E^i, if u and $v \in E^i_j$, then the vertices to which u and v are connected in G^{i-1} are in different blocks of E^i, and

(c) each block E^i_j contains at most 4 vertices of G^{i-1}.

If no such partition exists, then G^i does not exist.

(3) Construct G^i by using a vertex for each block of the partition E^i and by connecting vertices x and y of G^i by an arc from x to y iff (u, v) is an arc of G^{i-1}, where u and v are vertices in the blocks of E^{i-1} represented by x and y, respectively.

Obviously a homomorph is determined by a partition. Thus the homomorph G^i is called the **graph**(or **homomorph**) induced by the partition E^i.

Note that G^k is homomorphic to G, and if G is strongly connected then so is G^k.

For example let G be given by the following table.

Table 1

Source	Destinations
0	0 , 1
1	2 , 3
2	4
3	6 , 7
4	1
5	2
6	4 , 5
7	7

Then relation R defined in Definition 1 is $\{(0,1),(2,3),(4,5),(6,7),(1,0),(3,2),(5,4),(7,6)\}$. Hence the partition E^1 based on T_r is $\{ \{0,1\}, \{2,3\}, \{4,5\}, \{6,7\}\}$.
Let $A_0 = \{0,1\}$, $A_1 = \{2,3\}$, $A_2 = \{4,5\}$, and $A_3 = \{6,7\}$. Then $V^1 = \{ A_0, A_1, A_2, A_3\}$, and the arc set of G^1 is $\{ (A_0, A_0), (A_0, A_1), (A_1, A_2), (A_1, A_3), (A_2, A_0), (A_2, A_1), (A_3, A_2), (A_3, A_3) \}$. Then the partition $E^2 = \{ \{A_0, A_1\}, \{A_2, A_3\} \}$. Let $B_0 = (A_0, A_1)$, and $B_1 = (A_0, A_1)$. Then $V^2 = \{B_0, B_1\}$, and the arc set of G^2 is $\{ (B_0, B_0), (B_0, B_1), (B_1, B_0), (B_1, B_1) \}$.

Using this homomorph, two shift registers which realize a given sequential circuit can be obtained by the following algorithm if it is realizable with two shift registers. If this algorithm fails to produce shift registers, more than two shift registers are necessary to realize the sequential circuit.

Algorithm SRR

/* Input: A state transition graph G of a sequential circuit */
/* Output: A pair of factors for each vertex of G such that the code word of the destination state of each transition is obtained by shifting the factors of the code word of the source state to the left one position. */

/* Main */

Step 1: Find the i-th order homomorph G^i, $i \geq 1$, of G successively and apply **procedure template generation** for each G^i, starting with G until one of the following two graphs is

reached:
(a) CD_4 or its subgraph
(b) G^i can not be obtained because (2)(b) of Definition 2 is not satisfied. All the other conditions of Definition 2 hold and the number of vertices connected to the same set of destinations (or the destination set of one is a subset of that of the other) in G^{i-1} is at most two. Find a homomorph of G^{i-1} following Definition 2 but ignoring the condition (2)(b) (Here $DM(k,1)$ or its subgraph has been reached for some k). Let this homomorph be denoted by G^K.

If (a) is the case, then go to Step 2, else if (b), then go to Step 3. If neither (a) nor (b), then G can not be realized with two shift registers.

Step 2: Assign $00, 01, 10$, and 11 arbitrarily to the four vertices of CD_4. These strings are interpreted as two 1-bit factors. Obtain two factors for each vertex of G by substituting each of these 1-bit factors into the templates for CD_4. The code word for a vertex of G is the concatenation of the se two factors.

Step 3: At this point i.e. at the end of 1(b), one of the two factors for each vertex of G can be found as follows:
To each pair of vertices of G^K having an identical template, i.e. the same set of destinations (or the destination set of one is a subset of the other), assign a 0 or 1 to distinguish them from each other. Assign all the other vertices a 0 or 1 arbitrarily. Then substitute it into the template of each vertex. The result is one of the two factors for the vertex.

To find the other factor for the vertices of G, go to 4.

Step 4: Merge each pair of parallel arcs, hence their destination vertices.
Then continue finding i-th order homomorph of G^K and templates but with at most two vertices in each E^i_j of Definition 2(c) until CD_2 or its subgraph is reached.

Step 5: Assign a 0 or 1 to each vertex of CD_2. Then substitute it into the template of each vertex of G. The result is the second factor for the vertex. The concatenation of this factor and that from Step 4 is the code word for the vertex.

/* End of Main */

procedure template generation

/ This is a procedure to generate templates for (the factors of the code words of) the v ertices of G in terms of those for the vertices of G^q for a given q, when G^q exists. */*

/* template generation body */

Step 1: Assign an arbitrary label as a template to each vertex of G^{q+1}. No two of them may be identical.

Step 2: Express the template for each vertex of G^q, $q \geq 0$, in terms of templates for G^{q+1} as follows:

Let E^{q+1} be a partition of the vertex set of G^q which satisfies the conditions of Definition 2. Then each vertex of G^{q+1} corresponds to a block of E^{q+1}. Let the template of a vertex of G^{q+1} corresponding to a block E_i^{q+1} be denoted by E_i^{q+1}. For each vertex u of G^q, if it is in block E_i^{q+1} and if it is connected to vertices of E_j^{q+1}, then the template for u in terms of vertices of G^{q+1} is $E_i^{q+1}E_j^{q+1}$.

Step 3: The templates for the vertices of G in terms of vertices of G^{q+1} are obtained as follows:

Let $E_{i_1}^q\ E_{i_2}^q\ ...\ E_{i_q}^q$ be the template of a vertex u of G in terms of templates for G^q. Also let $E_{i_j1}^{q+1}E_{i_j2}^{q+1}$ be the representation of $E_{i_j}^q$ in terms of templates for G^{q+1}.
Then simultaneously replace each $E_{i_j}^q E_{i_{j+1}}^q$ with $E_{i_j1}^{q+1}E_{i_j2}^{q+1}E_{i_{j+1}2}^{q+1}$
to obtain the template for u in terms of those for G^{q+1}.

/* End of template generation */

As an example to illustrate Algorithm SRR let us consider the graph G having 32 vertices labeled $0 \sim 31$ and the incidence relation given by the following table.

Table 2

Source	Destinations	Source	Destinations
0	$0,1,4,5$	16	$0,1,4,5$
1	$2,3,6,7$	17	$2,3,6,7$
2	$0,1,4,5$	18	$0,1,4,5$
3	$2,3,6,7$	19	$2,3,6,7$
4	$8,9,12,13$	20	$8,9,12,13$
5	$10,11,14,15$	21	$10,11,14,15$
6	$8,9,12,13$	22	$8,9,12,13$
7	$10,11,14,15$	23	$10,11,14,15$
8	$16,17,20,21$	24	$16,17,20,21$
9	$18,19,22,23$	25	$18,19,22,23$
10	$16,17,20,21$	26	$16,17,20,21$
11	$18,19,22,23$	27	$18,19,22,23$
12	$24,25,28,29$	28	$24,25,28,29$
13	$26,27,30,31$	29	$26,27,30,31$
14	$24,25,28,29$	30	$24,25,28,29$
15	$26,27,30,31$	31	$26,27,30,31$

Then V can be partitioned into the following subsets which satisfy the conditions of Definition 2:

$A_0 = \{0,1,4,5\}$, $A_1 = \{2,3,6,7\}$, $A_2 = \{8,9,12,13\}$, $A_3 = \{10,11,14,15\}$, $A_4 = \{16,17,20,21\}$, $A_5 = \{18,19,22,23\}$, $A_6 = \{24,25,28,29\}$, $A_7 = \{26,27,30,31\}$.
G^1 in this case is given by the following table.
Table 3

Source	Destinations	Source	Destinations
A_0	A_0, A_1, A_2, A_3	A_4	A_0, A_1, A_2, A_3
A_1	A_0, A_1, A_2, A_3	A_5	A_0, A_1, A_2, A_3
A_2	A_4, A_5, A_6, A_7	A_6	A_4, A_5, A_6, A_7
A_3	A_4, A_5, A_6, A_7	A_7	A_4, A_5, A_6, A_7

Then V^1 can be partitioned into the following subsets which satisfy the conditions of Step 1(b) of Algorithm SRR:
$B_0 = \{A_0, A_1, A_2, A_3\}$, $B_1 = \{A_4, A_5, A_6, A_7\}$.
G^2 in this case is given by the following table.

Table 4

Source	Destinations
B_0	B_0, B_0, B_1, B_1
B_1	B_0, B_0, B_1, B_1

Here all parallel arcs are listed. In G^2 these parallel arcs are coalesced into one arc. So G^2 is, in this case, CD_2.

As can be easily seen from above, the code word of a vertex of G has two factors. It can be obtained as follows.
First the templates for the vertices of G are given in terms of vertices of G^1 as given in the following table.

Table 5

Vertex	Code	Vertex	Code	Vertex	Code	Vertex	Code
0	A_0A_0	8	A_2A_4	16	A_4A_0	24	A_6A_4
1	A_0A_1	9	A_2A_5	17	A_4A_1	25	A_6A_5
2	A_1A_0	10	A_3A_4	18	A_5A_0	26	A_7A_4
3	A_1A_1	11	A_3A_5	19	A_5A_1	27	A_7A_5
4	A_0A_2	12	A_2A_6	20	A_4A_2	28	A_6A_6
5	A_0A_3	13	A_2A_7	21	A_4A_3	29	A_6A_7
6	A_1A_2	14	A_3A_6	22	A_5A_2	30	A_7A_6
7	A_1A_3	15	A_3A_7	23	A_5A_3	31	A_7A_7

Next the templates for the vertices of G^1 are represented by vertices of G^2 as given in the following table. Note that some vertices have identical template. Their code words must therefore be distinguished by introducing another factor.

Table 6

Vertex	Code	Vertex	Code
A_0	B_0B_0	A_4	B_1B_0
A_1	B_0B_0	A_5	B_1B_0
A_2	B_0B_1	A_6	B_1B_1
A_3	B_0B_1	A_7	B_1B_1

To find the code word for A_is, assign 0 to B_0 and 1 to B_1, and distinguish A_is with the same template by assigning a second factor of 0 or 1. Then we have for A_is

Table 7

Vertex	Code	Vertex	Code
A_0	(00)(0)	A_4	(10)(0)
A_1	(00)(1)	A_5	(10)(1)
A_2	(01)(0)	A_6	(11)(0)
A_3	(01)(1)	A_7	(11)(1)

Applying to these the templates for the vertices of G, the following code words are obtained for the vertices of G.

Table 8

Vertex	Code	Vertex	Code	Vertex	Code	Vertex	Code
0	(000)(00)	8	(010)(00)	16	(100)(00)	24	(110)(00)
1	(000)(01)	9	(010)(01)	17	(100)(01)	25	(110)(01)
2	(000)(10)	10	(010)(10)	18	(100)(10)	26	(110)(10)
3	(000)(11)	11	(010)(11)	19	(100)(11)	27	(110)(11)
4	(001)(00)	12	(011)(00)	20	(101)(00)	28	(111)(00)
5	(001)(01)	13	(011)(01)	21	(101)(01)	29	(111)(01)
6	(001)(10)	14	(011)(10)	22	(101)(10)	30	(111)(10)
7	(001)(11)	15	(011)(11)	23	(101)(11)	31	(111)(11)

Theorem 3 *A sequential circuit with a state transition graph G is realizable with two or less shift registers if Algorithm SRR produces a coding for its states.*

The converse of Theorem 3 also holds as shown below.

Theorem 4 *If G is realizable with two shift registers of length k_1+1 and k_2+1, $k_1 \geq k_2$, then*
(1) G^i exists for all i, $1 \leq i \leq k_2$,
(2) if $k_1 = k_2$, then G^{k_2} is CD_4,
(3) if $k_1 > k_2$, then
(a) for every vertex of G^{k_2} all of its outgoing arcs go to at most two different vertices(destinations),
(b) $(G^{k_2})^i$ exists for $1 \leq i \leq (k_1 - k_2)$, where each block contains at most two vertices of $(G^{k_2})^{i-1}$, and
(c) $(G^{k_2})^{k_1-k_2}$ is CD_2.
Hence if G is realizable with two shift registers, then Algorithm SRR applied on G produces CD_2 or CD_4.

4 Coding of Cycle

As explained in Chapter 8 of Part 1, **procedure template generation**(see also algorithm SRR in the preceding chapter) is in general nondeterministic. That is, in Step 2 of the procedure the vertex set of G^q must be partitioned to find a template for the vertices. This partition, E^{q+1}, is in general not unique. Depending on how blocks are merged, different E^{q+1} hence different G^{q+1} is obtained.

Let us first review this nondeterminism. Suppose that P_r of G^q is always used as the partition E^{q+1} in Step 2 of **procedure template generation**. Let us call this version of Algorithm SRR Algorithm DSRR. Algorithm DSRR is deterministic. It terminates successfully as long as $E^q \neq E^{q+1}$ in Step 2 every time **procedure template generation** is called.

There are two cases for which Algorithm DSRR fails to produce shift registers. The first case is when $G^q = G^{q+1}$ for some q. The second case is when two or more states receive the same code word. These two are the only cases Algorithm DSRR fails to produce shift registers. For if case 1 does not occur, the size(no. of vertices) of G^q keeps decreasing as q increases, and eventually a subgraph of CD_2 or CD_4 is reached unless case 2 happens. If case 2 does not occur, no two vertices of G^q receive the same code word for any q. Hence the vertices of G receive different code words.

It can easily be seen that if $E^q = E^{q+1}$, then Algorithm DSRR fails to produce shift registers. This happens iff P_r of G^q is the set of singletons of the vertices of G^q. Since G hence G^q is assumed to be connected, this means that G^q is a cycle.

It is also possible that for some graph G, G^q becomes a cycle for some q. For example if G is a cycle plus a chord, then for some q, G^q is a cycle as can be verified easily.

In the remainder of this chapter a method of finding a coding for a cycle is discussed first. Then a coding for some special cases of a cycle plus chords is discussed.

4.1 Relevant Results on Cycles in $DM(k)$

Coding of the vertices of a cycle of length 2^k with a shift register coding has been the subject of considerable interest for coding theory researchers, [8, 9, 14, 15, 25, 26] to name a few. However, in this paper our interest is not just in cycles of length 2^k but also in cycles of an arbitrary length and the only previous work on that subject is by Yoeli[26].

The key result for our purpose in Yoeli's paper is that $DM(k)$ has a cycle of every length from 1 to 2^k. This result can be used to find a coding of a cycle of a given length. Some of the properties used in his paper can also be used to find a coding of a cycle with a chord. In this section results on cycles in de Bruijn graph (i.e. $DM(k)$) from his paper[26] are presented first. Some of them are generalized and proved here fro the first time. But even for the ones that are the same as Yoeli's, their proofs are repeated here, since they are either the key to new results or necessary to understand them.

First let us list some simple properties of $DM(k)$ without proofs.

Lemma 2 *$DM(k)$ is regular of in-degree 2 and out-degree 2.*

Lemma 3 *$DM(k)$ is strongly connected.*

Lemma 4 *$DM(k)$ has 2^{k+1} edges.*

Next let us list the results leading to one of the key properties of $DM(k)$: A closed trail of $DM(k-1)$ is a cycle of $DM(k)$ and vice versa. The following lemma is well known and it is illustrated in Fig. 7.

Lemma 5 *Let w, x, y, and z be vertices of $DM(k)$. If arcs (w, x), (w, y), and (z, x) exist in $DM(k)$, then arc (z, y) also exists in $DM(k)$.*

Let x and y be vertices of $DM(k)$ such that the arc (x, y) exists in $DM(k)$. Let $< x_1, x_2, ..., x_k >$ and $< y_1, y_2, ..., y_k >$ be the code word for vertices x and y, respectively, for a shift register code of $DM(k)$. Also let $< x, y >$ be defined as the vertex of $DM(k+1)$ with the code word $< x_1, x_2, x_3, ..., x_k, y_k >$ (Note that $< x_1, x_2, x_3, ..., x_k, y_k > = < x_1, y_1, y_2, ..., y_{k-1}, y_k >$). Obviously this code word can be assigned to the arc (x, y) of $DM(k)$. See for example Fig. 7 (a) and (b). Then the following lemma can easily be proven.

Lemma 6 *Each vertex of $DM(k+1)$ appears exactly once as the code word of an arc of $DM(k)$.*
Also let x, y, and z be vertices of $DM(k)$. If arcs (x, y) and (y, z) exist in $DM(k)$, then the arc $(< x, y >, < y, z >)$ exists in $DM(k+1)$.

The following lemma can be easily derived from Lemma 6, and it is one of the very useful results here.

Lemma 7 *A sequence of arcs $< e_1, ..., e_s >$ of $DM(k)$ is a closed trail if and only if the sequence of vertices $< \bar{e}_1, ..., \bar{e}_s >$ of $DM(k+1)$ corresponding to $< e_1, ..., e_s >$ forms a cycle in $DM(k+1)$.*

For example, the closed trail $< (0,0),(0,1),(1,0) >$ in $DM(1)$ in Fig. 7 (a) corresponds to the closed trail represented by the vertex sequence $< 00, 01, 10 >$ in Fig. 7 (b).

A **P-set of cycles** is a set of vertex disjoint cycles which includes all the vertices of the graph. The next lemma shows that $DM(k)$ can be partitioned into two P-sets of cycles. For example $Q_1 = \{< (0,0) >, < (1,1) >\}$ representing two loops, and $Q_2 = \{< (0,1),(1,0) >\}$ representing the cycle of length 2 are P-sets of cycles of $DM(1)$ of Fig. 7 (a).

Lemma 8 *[26] Let Q_1 be a P-set of cycles of $DM(k)$, and let Q_2 be the complement of Q_1 in $DM(k)$. Then Q_2 is also a P-set of cycles of $DM(k)$.*

Proof: Remove the arcs of Q_1 from $DM(k)$. Then since each vertex has exactly one incoming and one outgoing arc, Q_2 is the union of cycles and all the vertices are in exactly one of the cycles. Hence Q_2 is a P-set of cycles. ♣

For example Q_1 and Q_2 of the above example for Fig. 7 (a) are complement to each other in $DM(1)$.

The following lemma is a generalization of a similar result on one cycle by Yoeli[26].

Lemma 9 *Let X_1, ..., X_s be a set of vertex disjoint (hence also arc disjoint) cycles of $DM(k)$. Then there is a P-set of cycles of $DM(k)$ which contains X_1, ..., X_s.*

Proof: Let W_1, ..., W_s be the set of closed trail of $DM(k-1)$ corresponding to the cycles X_1, ..., X_s, respectively. Then W_1, ..., W_s do not share arcs between them. Therefore if W_1, ..., W_s are removed from $DM(k-1)$, then each vertex in the remainder has the same in- and out-degree. Hence the arcs in each connected component of the remainder can be ordered to form a closed trail. Each of these closed trails corresponds to a sequence of vertices of a cycle in $DM(k)$. Obviously they together with the vertices of X_1, ..., X_s cover all the vertices of $DM(k)$. Hence they form a P-set of cycles. ♣

From Lemmas 7 and 8 the following lemma can be easily derived. It is also a generalization of a similar result on one cycle by Yoeli[26].

Lemma 10 *Let X_1, ..., X_s be a set of vertex disjoint (hence also arc disjoint) cycles of $DM(k)$. Then there is a P-set of cycles of $DM(k)$ which* **does not** *contain any of X_1, ..., X_s.*

Proof: By Lemma 9 there is a P-set of cycles, call it Q_1, containing the cycles X_1, ..., X_s. By Lemma 8 the complement of Q_1 in $DM(k)$, call it Q_2, is also a P-set of cycles. Q_2 obviously does not contain any arcs of X_1, ..., X_s. ♣

The next lemma follows from Lemma 5 and it can be used to construct a larger cycle from two smaller cycles. It is illustrated in Fig. 7.

Lemma 11 *[26] Let X_1 and X_2 be cycles of $DM(k)$, and let e be an arc connected from a vertex of X_1 to a vertex of X_2. Then the vertices of X_1 and X_2 can be ordered to form a single cycle X.*

Proof: Let $e = (w, x)$. Hence w is in X_1 and x is in X_2. Let (w, y) be an arc in X_1 and let (z, x) be an arc in X_2. Then by Lemma 5 there is arc (z, y) in $DM(k)$. Hence by adding arcs (w, x) and (z, y) to $X_1 \cup X_2$ and by deleting arcs (w, y) and (z, x) from the resultant graph, a new cycle containing all the vertices of X_1 and X_2 is obtained. ♣

The arc $e = (w, x)$ from a vertex of X_1 to a vertex of X_2 in Lemma 11 is called a **bridging arc**, and arc (z, y) is called the **companion** of e.

The following theorem is the key to the recursion used in the main result given in Theorem 6, and its proof is utilized in proving the existence of a cycle of length 2^k with a chord.

Theorem 5 *[26] If there exists a cycle X of length ℓ in $DM(k)$, then there also exists in $DM(k)$ a closed trail of length $\ell + k$.*

Proof: By Lemma 10 there is a P-set of cycles Q_1 which does not contain any arcs of cycle X. Let H_1 be the subgraph of $DM(k)$ formed by the arcs of X and Q_1. (An example to illustrate this proof is given below). If H_1 is not connected, there is an arc e_1 connecting two components of H_1. This e_1 is a bridging arc from one cycle Y' to another cycle Y'' of Q_1, and it is not in X. Hence by Lemma 11 two cycles Y' and Y'' can be merged to form another cycle, say Y. Replace Y' and Y'' with Y in Q_1. Then the resultant graph is a P-set of cycles with one less cycle than Q_1 and it does not contain any of its arcs. Repeat this process until a P-set of cycles Q_m is obtained which together with X forms a connected graph. Obviously the arcs of $Q_m \cup X$ can be ordered to form a closed trail. ♣
For example in Fig. 7 (a) $Q_1 = \{Y_1, Y_2, Y_3\}$ for cycle X. H_1 in this case is not connected and consists of two components $X \cup Y_1 \cup Y_2$ and Y_3. e_1 in Fig. 7 is the bridging arc. Y_1 and Y_3 can be merged as shown by dashed lines in Fig. 7 (b). It can be easily seen that there is a closed trail traversing all the arcs of $X \cup Y_1' \cup Y_2$.

We now can prove the main theorem.

Theorem 6 *[26] $DM(k)$ contains a cycle of length ℓ, for all ℓ, $0 < \ell \leq k$.*

Proof: The following two cases can be considered depending on the value of ℓ: $\ell \leq 2^{k-1}$, and $\ell > 2^{k-1}$.

Case 1. $\ell \leq 2^{k-1}$: This case reduces to the question of existence of a cycle of length ℓ for $DM(k-1)$, and eventually it goes down to the existence of cycle of length 1 in $DM(1)$, which obviously exists.

Case 2. $\ell > 2^{k-1}$: Let $\ell' = \ell - 2^{k-1}$. Then by Case 1 we can assume that a cycle of length ℓ' exists in $DM(k-1)$. Hence by Lemma 5 there is a closed trail of length ℓ. Hence by Lemma 7 there is a cycle of length ℓ in $DM(k)$. ♣

4.2 Coding of Cycle

A cycle of length ℓ can always be realized by a shift register of length $\lceil \lg \ell \rceil$ as can be seen from the results by Yoeli[26].

Algorithm CYCLE_CODING(ℓ)

/* Input: a positive integer ℓ */
/* Output: a sequence of binary code words for vertices of a cycle of length ℓ */

/* Main */

$k := \lceil \lg \ell \rceil$;
if $(\ell = 1)\{$
  then return $< 0 >$;
  else $\{$
    if $(\ell = 2)\{$
      then return $< 0, 1 >$;
      else $\{$
        $n := 2^{\lceil \lg \ell \rceil}$;   /* Note that $\ell > n/2$. */
        Find a shift register coding with $(k-1)$ bits for a cycle, call it C_1, of length $(\ell - n/2)$ by recursively applying Algorithm CYCLE_CODING;
        If the obtained coding is of length less than $k-1$, then convert it to a $k-1$ bit coding by applying the procedure given below;
        Find a P-set of cycles Q of $DM(k-1)$ which does not contain the arcs of C_1 as given below;
        Construct a closed trail W in $DM(k-1)$ from cycles of Q and C_1 following the procedure of Theorem 5;
        return the cycle in $DM(k)$ which corresponds to W.
      $\}$
    $\}$
  $\}$
$\}$

A shift register coding of a cycle in $DM(k)$ can be found from that for a closed trail in $DM(k-1)$ as follows:

Let $u_1, ..., u_\ell$ be ℓ $(k-1)$-bit code words representing the vertices of a closed trail in $DM(k-1)$ for some positive integer k;
For each i, $1 \leq i \leq \ell$, let $v_i := u_i$ with the last bit of $u_{i+1}(\text{mod } \ell)$ appended at the end;
Then $v_1, ..., v_\ell$ are code words representing the vertices of a cycle of length ℓ in $DM(k)$ corresponding to the closed trail.

A P-set of cycles Q of $DM(k-1)$ which does not contain the arcs of a given cycle C can be obtained as follows:

Find a closed trail W in $DM(k-2)$ corresponding to C;
Remove the arcs of W from $DM(k-2)$. Denote the resultant graph by $DM(k-2)-W$;
For each component of $DM(k-2)-W$, find a closed trail;

The cycles of $DM(k-1)$ corresponding to the closed trails together with C form a P-set of cycles, call it Q', of $DM(k-1)$ containing C;
The complement of Q' in $DM(k-1)$ is a P-set of cycles of $DM(k-1)$ which does not contain Q.

In **Algorithm CYCLE_CODING** to find a P-set of cycles, closed trails for Euler graphs must be found. This can be done in $O(e)$, for example, by first removing an arbitrary arc from a given Euler graph and then tracing its arcs starting at the tail of the removed arc until all the arcs are traversed.

As an example for CYCLE_CODING let us find a shift register coding for a cycle of length 11.
Since $\ell = 11$, $k = 4$ and $n = 16$. Hence $\ell - n/2 = 3$. Hence CYCLE_CODING(3) is entered.
In CYCLE_CODING(3), $\ell = 3$. Hence $k = 2$ and $n = 4$. Hence $\ell - n/2 = 1$ and CYCLE_CODING(1) is entered.
CYCLE_CODING(1) returns $< 0 >$.

Back in CYCLE_CODING(3), since $k = 2$, a P-set of cycles Q not containing cycle $< 0 >$, that is the loop at vertex 0, is sought in $DM(1)$. Obviously $Q = \{< 0, 1 >\}$ and the closed trail W is $< 0, 0, 1 >$. See Fig. 7 (a) and (b). Hence CYCLE_CODING(3) returns $< 00, 01, 10 >$.

Back in CYCLE_CODING(11), since $k = 4$, a P-set of cycles Q not containing cycle $< 001, 010, 100 >$ (converted into 3 bit code from $< 00, 01, 10 >$), call it C_1, is sought in $DM(3)$. For Q for example $\{< 000, 001, 011, 111, 110, 100 >, < 010, 101 >\}$ can be used, as shown in Fig. 7 (c). Then the closed trail W combining C_1 and Q is $< 000, 001, 011, 111, 110, 100, 001, 010, 101, 010, 100 >$. Hence the cycle $< 0001, 0011, 0111, 1110, 1100, 1001, 0010, 0101, 1010, 0100, 1000 >$ is obtained.

5 Coding of Cycle with Chord

Along with cycles, cycles with one chord have the least clue as to which P_r blocks to merge to obtain partitions that lead to shift registers. In this section it is going to be proven that a cycle with a chord can be realized with a shift register if the length of the cycle determined by the chord is equal to $n/2$ or not larger than $n/4$, where n is the number of vertices of the cycle and it is a power of 2. It is conjectured that a cycle with a chord can be coded with the shortest shift register code if it is not a cycle of length 8 with a chord which forms a cycle of length 3. Note, however, that if H is a cycle of length 8 with a chord forming a cycle of length 3 or 5, then it is a subgraph of $DM(4)$.

Theorem 7 *Let H be a cycle of length $n = 2^k$ with a chord, and let ℓ be the length of the subcycle of H determined by the chord. If $1 \leq \ell \leq n/4$, then H is a subgraph of $DM(k)$.*

Proof: For $k = 1$ obviously no chord can exists. For $k = 2$, there is only one possible chord, and the theorem obviously holds for that. So let us assume that $k \geq 3$.

Let H be a cycle of length n with a chord e_1. If the chord e_1 forms a cycle C_1 of length ℓ in H, then by Lemma 5, there is a unique chord e_2 which is the companion of e_1. When added to H, it forms a cycle of length $n - \ell$ disjoint from C_1 in H. Thus what we need to do is to find a splitting of a cycle of length 2^k into two disjoint cycles. Obviously these cycles are of length 2 or larger.

According to Lemmas 6 and 7, there is a one-to-one correspondence between the arcs of $DM(k-1)$ and the vertices of $DM(k)$, Hence we now need to find a closed trail of a given length in $DM(k-1)$ which leaves a connected graph when removed from $DM(k-1)$.

By symmetry we can assume that $\ell \leq 2^{k-1}$.

First the following claim is going to be proven:

Claim: If a cycle C is removed from $DM(k-2)$, then the cycle C' in $DM(k-1)$ corresponding to C is removed from $DM(k-1)$ and the resultant subgraph of $DM(k-1)$ is connected as a non-directed graph.

With this claim if the arcs of C' is removed from $DM(k-1)$, the resultant graph, denoted by $DM(k-1) - C'$, is a connected Euler graph. Hence there is a closed trail that covers $DM(k-1) - C'$, and the corresponding subgraph of $DM(k)$ is a cycle. This cycle together with the cycle in $DM(k)$ corresponding to C' form the desired pair of cycles.
Proof of Claim: Suppose that $DM(k-1) - C'$ is not connected. Let x_1 and x_2 be vertices that are in different connected components of $DM(k-1) - C'$, and that are the endpoints of an arc e_1 of C', say $e_1 = (x_1, x_2)$. Then x_1 and x_2 are connected in $DM(k-1) - C'$ by a (undirected) path. For, corresponding to vertices x_1 and x_2, there are arcs x_1 and x_2 in $DM(k-2)$, and they share a vertex, say v. See Fig. 7. In $DM(k-2)$ there are two more arcs incident to v. Call the incoming arc x_3 and the outgoing arc x_4. Then in $DM(k-1)$ there are arcs $(x_1, x_4), (x_3, x_2)$, and (x_3, x_4). Hence there is a path between x_1 and x_2 other than the arc (x_1, x_2) in $DM(k-1)$. Hence x_1 and x_2 are connected in $DM(k-1) - C'$ provided that $(x_1, x_4), (x_3, x_2)$, and (x_3, x_4) are not removed, that is, provided that they are not in C'. Now C in $DM(k-2)$ is a cycle. Thus every vertex in C is visited once when C is traversed. Hence when C is removed from $DM(k-2)$, exactly one of the arcs $(x_1, x_2), (x_1, x_4), (x_3, x_2)$ and (x_3, x_4) is removed. Thus x_1 and x_2 are connected in $DM(k-1) - C'$. This contradicts the assumption that x_1 and x_2 are in different connected components of $DM(k-1) - C'$.

Since $DM(k-1)$ is Euler, $DM(k-1) - C'$ is obviously also Euler. Corresponding to these, there are cycles in $DM(k)$. They are of lengths ℓ and $n - \ell$, and vertex disjoint. Thus they are the desired pair of cycles. QED for the Theorem ♣

To illustrate the proof let us consider a cycle of length 16 with a chord forming a cycle of length 3. This can be embedded in a $DM(4)$ as follows.

In $DM(2)$, $00-01-10-00$ is a cycle of length 3. call it C. Corresponding to C, there is a cycle of length 3 $001-010-100-001$ in $DM(3)$. This cycle is C'. Remove the arcs of C' from $DM(3)$. Then $DM(3) - C'$ is a closed trail. For example $000-000-001-011-111-111-110-101-010-101-011-110-100-000$ is a closed trail that covers all the arcs of $DM(3)$. From this trail and C' two cycles of $DM(4)$ can be obtained which can be merged into a Hamiltonian cycle of $DM(4)$. That is, cycles $0010-0100-1001-0010$, and $0000-0001-0011-0111-1111-1110-1101-1010-0101-1011-0110-1100-1000-0000$. They can be merged into the Hamiltonian cycle of $0000-0001-0011-0111-1111-1110-1101-1010-0100-1001-0010-0101-1011-0110-1100-1000-0000$ as shown in Fig. 7.

Theorem 8 *Let H be a cycle of length $n = 2^k$ with a chord, and let ℓ be the length of the subcycle of H determined by the chord. If $\ell = n/2$, then H is a subgraph of $DM(k)$.*

Proof Outline: In this proof two cycles of length $n/2$ are going to be found exploiting the symmetry of $DM(k)$.

First draw a graph of $DM(k)$ by placing its vertices clockwise along a circle in the increasing order of their label. It is assumed that the vertices of $DM(k)$ are labeled with the integers from 0 to $n-1$ so that an arc (i, j) exists from vertex i to vertex j iff $j = 2i$ or $j = 2i + 1$. It can easily be seen that $DM(k)$ is symmetric about a straight line passing between vertices 0 and 1, and between vertices $n/2$ and $n/2 - 1$ in the drawing. Let us denote this straight line by L. If $DM(k)$ is folded along the straight line L merging the corresponding vertices and removing parallel arcs, the resultant graph is $DM(k-1)$ as shown below (See Fig. 7 for example). Two cycles of length $n/2$ of $DM(k)$ can be obtained from a Hamiltonian cycle in this $DM(k-1)$.

Proof: This is going to be proven by a series of lemmas. First the graph obtained by folding $DM(k+1)$ along L is studied.

Let f be the function from the set of integers $\{0, 1, ..., 2^{k+1} - 1\}$ to the set of integers $\{0, 1, ..., 2^k - 1\}$ defined as follows:

For each integer i, $0 \le i \le 2^k - 1$, let $f(i) = i$,
and for each integer i, $2^k \le i \le 2^{k+1} - 1$, let $f(i) = 2^{k+1} - 1 - i$.

Construct a graph with 2^k vertices as follows: Label the vertices with integers $0, 1, ..., 2^k - 1$. Connect vertices u and v iff there are vertices i and j in $DM(k+1)$ such that (i, j) is an arc of $DM(k+1)$, $u = f(i)$, and $v = f(j)$. It can easily be seen that the graph thus obtained is the result of folding $DM(k+1)$ along L. Let us denote the graph thus obtained by H_k.

Let n be the number of vertices of $DM(k+1)$. Hence $n = 2^{k+1}$. Let us examine what happens to an arc when the function f is applied to $DM(k+1)$ (See also Fig. 7). Depending on the position of the tail of an arc the following four cases are considered:

Let (i, j) be an arbitrary arc of $DM(k+1)$.

Case 1: $0 \le i < n/4$

In this case, since $i, j \le n/2$, $f(i) = i$, and $f(j) = j$. Hence when f is applied, the arc (i, j) becomes the arc (i, j) in $DM(k)$.

Case 2: $n/4 \le i < n/2$

In this case $f(i) = i$. But since $n/2 \le j \le n-1$, $f(j) = n - 1 - j$, and $0 \le f(j) < n/2$.

Case 3: $n/2 \le i < 3n/4$

In this case $f(i) = n - 1 - i$. Hence $n/4 \le f(i) < n/2$. Also since $j = 2i, 2i+1 \mod n$, $f(j) = j$, and $0 \le f(j) < n/2$.

Case 4: $3n/4 \le i < n$

In this case $f(i) = n - 1 - i$. Hence $0 \le f(i) < n/4$. Also since $n/2 \le j \le n-1$, $f(j) = n - 1 - j$. Hence $0 \le f(j) < n/2 - 1$.

It is easy to see that the arcs of Case 3 and Case 4 are duplicates of those of Case 1 and Case 2.

Next it is going to be shown that the graph H_k thus obtained is a $DM(k)$ by labeling the vertices of H_k with a certain sequence called a "**folded graph sequence**", which is defined below, and then by showing that the arcs (i, j) of H_k satisfy $j = 2i$ or $j = 2i + 1 \mod 2^k$.

Definition 3 *For a natural number k, a* **folded graph sequence** $S_k =< s_0, s_1, ..., s_{2^{k+1}-1} >$ *is a sequence of natural numbers defined inductively as follows:*
Basis Step: $S_0 =< 0, 1 >$ *is a folded graph sequence.*
Inductive Step: Let $S_k =< s_0, s_1, ..., s_{2^{k+1}-1} >$ *be a folded graph sequence. Then* $S_{k+1} =< s_0, s_1, ..., s_{2^{k+1}-1}, s_{2^{k+1}}, ..., s_{2^{k+2}-1} >$ *is a folded graph sequence if*
(1) for every i, $0 \leq i \leq 2^{k+1} - 1$, s_i is the same as those in S_k, and (2) for every i, $2^{k+1} \leq i \leq 2^{k+2} - 1$, $s_i = s_{2^{k+2}-1-i} + 2^{k+1}$.

Example: A folded graph sequence $S_1 =< s_0, s_1, s_2, s_3 >$ is obtained as follows: Since $S_0 =< 0, 1 >$, for S_1, $s_0 = 0$ and $s_1 = 1$. $s_2 = s_1 + 2^1 = 3$, and $s_3 = s_0 + 2^1 = 2$. Thus $S_1 =< 0, 1, 3, 2 >$. Similarly $S_2 =< 0, 1, 3, 2, 6, 7, 5, 4 >$.

Now it is going to be shown that the graph H_k is in fact a $DM(k)$.

Lemma 12 *H_k is a $DM(k)$.*

Proof: Given H_k, relabel the vertices with the folded graph sequence S_k, that is, relabel vertex i of H_k with s_i of S_k. Let G_k be the relabeled graph (See Fig. **??**). It is going to be shown that there is an arc (i, j) in H_k iff $s_j = 2s_i$ or $s_j = 2s_i + 1 \bmod 2^k$ in G_k.

Let $n = 2^k$, the number of vertices in G_k. Depending on the value of the tail vertex i of an arc (i, j) of H_k the proof is divided into the following two cases: $0 \leq i \leq n/2 - 1$, and $n/2 \leq i \leq n - 1$.

Case 1: $0 \leq i \leq n/2 - 1$. The arcs going out of i are $(i, 2i)$ and $(i, 2i + 1)$ in H_k. Hence in G_k the corresponding arcs are (s_i, s_{2i}) and (s_i, s_{2i+1}).

Claim $\{s_{2i}, s_{2i+1}\} = \{2s_i, 2s_i + 1\}$.
Proof of Claim: By induction on k of H_k.
Basis Case: For $k = 0$, since $S_0 = \{0, 1\}$, it obviously holds true.
Inductive Step: Assume true for k, and prove for $k + 1$.
If $0 \leq i < n/4$, where $n = 2^{k+1}$, then since $2i, 2i + 1 \leq n/2(= 2^k)$ and $\{s_0, ..., s_{2^k-1}\} = S_k$, by the induction hypothesis the claim holds. So let us assume $n/4 \leq i < n/2$.
By the way s_i are constructed,

$$s_i = s_{n/2-i-1} + n/4.$$

Hence $2s_i = 2s_{n/2-i-1} + n/2$.
First let us consider $2s_{n/2-i-1}$. Since $0 \leq n/2 - i - 1 < n/4$, by the induction hypothesis the vertex $2(n/2 - i - 1)$ or $2(n/2 - i - 1) + 1$ of H_{k+1} is labeled with $2s_{n/2-i-1}$ in G_{k+1}.
Also by the way s_i are constructed,

$$s_j = s_i + n/2 \text{ iff } j + i = n - 1.$$

Hence the label in H_{k+1} of the vertex corresponding to $2s_i$ is $n - 1$ minus that of the vertex corresponding to $2s_{n/2-i-1}$. Hence $2s_i = 2s_{n/2-i-1} + n/2$ is at $(n - 1) - (n - 2i - 2)$ or $(n - 1) - (n - 2i - 1)$. That is $2s_i$ is at $2i$ or $2i + 1$. Hence $\{2s_i, 2s_i + 1\} = \{s_{2i}, s_{2i+1}\}$.

QED for Claim hence for Case 1.

Case 2: $n/2 \leq i < n$. In this case, by the way s_i are constructed $s_i = s_{(n-1)-i} + n/2$ holds.
Hence $2s_i = 2s_{(n-1)-i} \bmod n$

$$= s_{2(n-1)-2i} \text{ or } s_{2(n-1)-2i+1}$$

by the induction hypothesis since $i \geq n/2$.

Hence $\{2s_i, 2s_i+1\} = \{s_{2n-2-2i}, s_{2n-2-2i+1}\}$.

QED for Case 2 hence for Lemma.

Thus it has been shown that H_k obtained from $DM(k+1)$ is $DM(k)$.

Next let us obtain two cycles of $DM(k+1)$ from H_k. Note that the vertices of H_k are assumed to be labeled as mentioned above. First find a Hamiltonian cycle of H_k, and denote it by C. Let n be the number of vertices of $DM(k+1)$. From C construct two cycles (denote them as C_1 and C_2) of length $n/2$ in $DM(k+1)$ as follows:

procedure to obtain cycle C_1

/* Let i represent a vertex of $DM(k+1)$. */

Initially let $i = 0$ and $C_1 = \emptyset$.
for each vertex i {
    if $0 \leq i < n/4$, and (i, j) is an arc of C, **then** add the arc (i, j) to C_1.
        else if $n/4 \leq i < n/2$, and (i, j) is an arc of C, **then** add the arc $(i, n-1-j)$ to C_1.
            else if $n/2 \leq i < 3n/4$, and $(n-1-i, j)$ is an arc of C, **then** add the arc (i, j) to C_1.
                else if $3n/4 \leq i < n$, and $(n-1-i, j)$ is an arc of C, **then** add the arc $(i, n-1-j)$ to C_1.
    if arc (p, q) has been added to C_1, **then** update i by $i := q$.
    if $i = 0$, **then halt**.
}
end of procedure

procedure to obtain cycle C_2
Initially $C_2 = \emptyset$.
For each vertex i of C_1, add the vertex $n-1-i$ to C_2.

As an example let us construct two cycles of length 8 from $DM(4)$. Hence $n = 16$. First obtain H_3 from $DM(4)$ (see Fig. 7). Then find a Hamiltonian cycle C in H_3, say $0-1-3-6-2-5-4-7-0$. Now we are going to apply the procedure to obtain C_1 from this C.
Initially $C_1 = \emptyset$ and $i = 0$.
The arc going out of vertex 0 in C is $(0, 1)$. Since $0 \leq i < n/4$, $(0, 1)$ is added to C_1.
i is now updated to $i = 1$ since $(0, 1)$ has been added to C_1.
The arc going out of vertex 1 in C is $(1, 3)$. Since $0 \leq i < n/4$, $(1, 3)$ is added to C_1.
i is now updated to $i = 3$ since $(1, 3)$ has been added to C_1.
The arc going out of vertex 3 in C is $(3, 6)$. Since $0 \leq i < n/4$, $(3, 6)$ is added to C_1.
i is now updated to $i = 6$ since $(3, 6)$ has been added to C_1.
The arc going out of vertex 6 in C is $(6, 2)$. Since $n/4 \leq i < n/2$, $(6, 13)$ is added to C_1.
i is now updated to $i = 13$ since $(6, 13)$ has been added to C_1.

Since $i > n/2$, the vertex $n - 1 - i = 2$ in C is considered. The arc going out of vertex 2 in C is $(2, 5)$. Since $n/2 \leq i < 3n/4$, $(13, 10)$ is added to C_1.
i is now updated to $i = 10$ since $(13, 10)$ has been added to C_1.
Since $i > n/2$, the vertex $n - 1 - i = 5$ in C is considered. The arc going out of vertex 5 in C is $(5, 4)$. Since $n/2 \leq i < 3n/4$, $(10, 4)$ is added to C_1.
i is now updated to $i = 4$ since $(10, 4)$ has been added to C_1.
The arc going out of vertex 4 in C is $(4, 7)$. Since $n/4 \leq i < n/2$, $(4, 8)$ is added to C_1.
i is now updated to $i = 8$ since $(4, 8)$ has been added to C_1.
Since $i \geq n/2$, the vertex $n - 1 - i = 7$ in C is considered. The arc going out of vertex 7 in C is $(7, 0)$. Since $n/2 \leq i < 3n/4$, $(8, 0)$ is added to C_1.
i is now updated to $i = 0$ since $(8, 0)$ has been added to C_1.
Since $i = 0$, the procedure stops.
Thus the cycle $C_1 = 0 - 1 - 3 - 6 - 13 - 10 - 4 - 8 - 0$ has been obtained.
From this C_1, C_2 is obtained as $C_2 = 15 - 14 - 12 - 9 - 2 - 5 - 11 - 7 - 15$.
It can easily be seen that by replacing arcs $(5, 11)$ and $(13, 10)$ with arcs $(5, 10)$ and $(13, 11)$, a Hamiltonian cycle of $DM(4)$ is obtained.

Lemma 13 *C_1 and C_2 obtained by the procedures given above are disjoint cycles of $DM(k+1)$ with length $n/2$.*

Proof: From the way H_k is constructed from $DM(k + 1)$, the arcs of C_1 and C_2 clearly exist in $DM(k + 1)$.

Since each vertex i of C represents two vertices i and $n - 1 - i$ of $DM(k + 1)$, and one of them is selected for C_1 and the other is used in C_2, C_1 and C_2 are disjoint.

Since no vertices are repeated in constructing C_1 and C_2, the vertices in C_1 are distinct from each other. For each arc of C_1 is added into C_1 as an arc of C is traversed and no arcs of C are traversed more than once. Thus there is an obvious isomorphism between C and C_1. Hence no vertices are duplicated in C_1. Similarly for C_2.

Finally both C_1 and C_2 are closed, that is, a cycle. For let us call an arc (i, j) of C_1 a **cross arc** iff either $0 \leq i < n/2$ (i is on the right side) and $n/2 \leq j < n$(j is on the left side), or i is on the left side and j is on the right side. Then there are an even number of cross arcs in C_1. For all arcs leaving the vertices in the upper half of H_k are cross arcs and the rest are not. Hence if C_1 is traversed starting at a vertex, say p, on the right side, then it must come back to the right side. Corresponding to this traversal of C_1, the arcs of C are traversed, and the starting vertex, say p', corresponding to p, is reached after the traversal. Since each vertex of C represents two vertices of $DM(k + 1)$, one on either side, when p' is reached, that p' also corresponds to p in C_1. Thus C_1 is a cycle. Similarly for C_2.

QED

Thus two disjoint cycles of $DM(k + 1)$ of length $n/2$ have been obtained. These cycles halve a Hamiltonian cycle of $DM(k+1)$. For they are a cycle in $DM(k+1)$ and all the vertices are present in them. Thus they are connected in $DM(k + 1)$ by at least one arc. Hence by

Lemma 5 there are two arcs connecting C_1 and C_2. Hence by Lemma 11 a Hamiltonian cycle can be constructed from them in $DM(k+1)$.

6 Bidirectional Shift Registers

In this chapter properties of the state transition graphs(STG) of sequential machines realizable with bidirectional binary shift registers (abbreviated as BDBSR) are discussed.

Obviously more transitions of a sequential machine can be accommodated by BDBSR than by a binary shift register. At the moment, however, no easily testable conditions are known for testing whether or not a given STG is realizable with BDBSRs, including the case when transitions are completely specified. In this chapter STGs of completely specified sequential machines which are realizable with BDBSR are going to be studied.

First let us denote by $UDM(k_1, ..., k_m)$ the undirected graph obtained by removing the direction from each arc of $DM(k_1, ..., k_m)$ (the resulting parallel edges are not removed), i.e. in $UDM(k_1, ..., k_m)$ an edge exists between vertices u and v iff there is an arc from u to v or from v to u in $DM(k_1, ..., k_m)$. Given a state transition graph of a sequential machine, construct an undirected graph from the STG by removing the direction from each arc. Obviously a given STG is realizable with BDBSR if and only if the resulting graph is a subgraph of a $UDM(k_1, ..., k_m)$.

Here an algorithm is going to be developed for testing whether or not a given undirected graph is isomorphic to $UDM(k_1, ..., k_m)$. The basic idea of the algorithm is to construct $DM(k_1, ..., k_m)$ from $UDM(k_1, ..., k_m)$ by giving the right direction to the edges of $UDM(k_1, ..., k_m)$. Let us first note that there are following two types of edges among the edges incident to a vertex, say v, in $UDM(k_1, ..., k_m)$: The ones corresponding to the arcs coming into v in $DM(k_1, ..., k_m)$ and the ones corresponding to the arcs going out of v in $DM(k_1, ..., k_m)$. Let us call the former **in-edges** and the latter **out-edges**. Since these edges can be uniquely identified by their endpoint opposite to v, edges and their endpoint are used interchangeably in this chapter. Once the in- and out-edges are identified for each vertex, testing for $UDM(k_1, ..., k_m)$ is straightforward. That is, remove the in-edges from the adjacency lists of the given undirected graph and test whether or not $P_c * P_r = \underline{0}$ is satisfied by the resultant adjacency lists considered as those of a directed graph. To identify the in- and out-edges, let us note the following simple facts about $UDM(k_1, ..., k_m)$.

Remark 1 *There are 2^m in-edges and 2^m out-edges for each vertex of $UDM(k_1, ..., k_m)$.*

Since in $DM(k_1, ..., k_m)$ each vertex has 2^m outgoing arcs, and 2^m incoming arcs, this remark is obvious.
If the arrows of the arcs of $DM(k_1, ..., k_m)$ are reversed, a $DM(k_1, ..., k_m)$ is obtained. Hence the following remark is also immediate.

Remark 2 *For a set of in- and out-edges of $UDM(k_1, ..., k_m)$, there is a $DM(k_1, ..., k_m)$ which has the in-edges as outgoing arcs and the out-edges as incoming arcs for each vertex.*

Let us denote by $\hat{k_i}$ a k_i bit code word consisting of only 0s or only 1s. Then a vertex coded with $\hat{k_1}\hat{k_2}...\hat{k_m}$ in a shift register code has a transition to itself, that is, it has a loop in STG. By the same argument as for Remark 1 the next remark is obvious.

Remark 3 *If a vertex v of $DM(k_1, ..., k_m)$ is coded with a code word $\hat{k_1}\hat{k_2}...\hat{k_m}$ (hence it has a loop in STG), then there is a shift register coding which assigns $\hat{0}$ to the vertex v.*

Remark 4 *If a vertex v of $UDM(k_1, ..., k_m)$ has a loop, then v appears twice in an adjacency list of itself. One of the two v's corresponds to the incoming arc and the other to the outgoing arc of the corresponding $DM(k_1, ..., k_m)$.*

Lemma 14 *Let u and v be two different in-edges at a vertex t of $UDM(k_1, ..., k_m)$, where $k_i \geq 2$ for all $i, 1 \leq i \leq m$. Let $A(u)$ and $A(v)$ be an adjacency list of vertices u and v in $UDM(k_1, ..., k_m)$, respectively. Then $A(u) \cap A(v)$ is the set of out-edges of u and v. Hence $A(u) - A(u) \cap A(v)$ is the set of in-edges of u.*

Proof: The set of destinations of u must be the same as that for v if they share a vertex. In fact all the in-edges of t have the same set of out-edges. This is because u and v differ only in the most significant bit of one or more of the m factors of their code word. Also u and v can not share any in-edge unless $k_i = 1$ for some i. For if they do, then their code words differ only in the least significant bit of one or more factors. This means that their code words are identical, which can not be the case. *QED*

As an example to illustrate Lemma 14 let us consider $UDM(2, 2)$ given by the following table of adjacency lists of its vertices.

Vertex	Out-edges				In-edges			
0	0	1	4	5	0	2	8	10
1	2	3	6	7	0	2	8	10
2	0	1	4	5	1	3	9	11
3	2	3	6	7	1	3	9	11
4	8	9	12	13	0	2	8	10
5	10	11	14	15	0	2	8	10
6	8	9	12	13	1	3	9	11
7	10	11	14	15	1	3	9	11
8	0	1	4	5	4	6	12	14
9	2	3	6	7	4	6	12	14
10	0	1	4	5	5	7	13	15
11	2	3	6	7	5	7	13	15
12	8	9	12	13	4	6	12	14
13	10	11	14	15	4	6	12	14
14	8	9	12	13	5	7	13	15
15	10	11	14	15	5	7	13	15

As a t take vertex 1 and as in-edges u and v take vertices 2 and 8. Then $A(2) = \{0, 1, 4, 5, 1, 3, 9, 11\}$ and $A(8) = \{0, 1, 4, 5, 4, 6, 12, 14\}$. Hence $A(2) \cap A(8) = \{0, 1, 4, 5\}$, and these are the out-edges of vertices 2 and 8. The in-edges of 2 are $\{1, 3, 9, 11\}$ and those of 8 are $\{4, 6, 12, 14\}$.

If some k_i is equal to 1, then there are $n/2$ pairs of vertices which have the same adjacency list. Thus vertices distinguished by a shift register of length 1 are identified. They can be coalesced before proceeding to the following algorithm for testing for the realizability with bidirectional shift registers.

Algorithm BDBSR_TEST

Input: a digraph G

It is assumed that the vertices with the same adjacency list are coalesced in G.

Output: "Yes" if G can be realized with BDBSR's, "No" otherwise.

Construct an undirected graph from G by removing the direction from each arc of G, call the resultant graph also G;
Find a vertex with a loop in G, and denote it by u;
Select an arbitrary vertex from the adjacency list $A(u)$ of u, and denote it by v;
$S := \emptyset;\qquad P_r := \emptyset;$
$P_r := P_r \cup \{A(u) \cap A(v)\};$
$S := S \cup \{A(u) - (A(u) \cap A(v))\} \cup \{A(v) - (A(u) \cap A(v))\};$
$A(u) := A(u) \cap A(v);$
$A(v) := A(u) \cap A(v);$
while $S \neq \emptyset$ {
  Select an arbitrary element X from S;
  Delete X from S;
  Pair the vertices of X;
  for each *unexamined* pair x and y of X do{
    $S := S \cup \{A(x) - (A(x) \cap A(y))\} \cup \{A(y) - (A(x) \cap A(y))\};$
/* If $A(i) - (A(i) \cap A(j))$ is already in S or it was added to S before and subsequently deleted for $i, j = x, y$, then it is not added to S again here. */
    $P_r := P_r \cup \{A(x) \cap A(y)\};$
    $A(x) := A(x) \cap A(y);$
    $A(y) := A(x) \cap A(y);$
    The pair x and y are now *examined.*
  }
}
while there is an *unexamined* vertex do{
  Let s be an *unexamined* vertex.
  Let w be an *examined* vertex such that $w \in A(s)$ and let t be an *examined* vertex such that $w \in A(t)$. $S := S \cup \{A(s) - (A(s) \cap A(t))\};$
  $A(s) := A(s) \cap A(t);$
  s is now *examined.*
}

For the resultant adjacency lists, if $P_C * P_R = \underline{0}$ holds then return Yes, else return No.
end of BDBSR_TEST

As an example let us consider $UDM(2,2)$ as given above.
Each row of the table is an adjacency list of a vertex in the leftmost column. Since 0 appears twice in the adjacency list of 0, 0 is selected as u (3, 12, or 15 can also be selected as the initial u). Let us next select 2 as v (any other vertex in $A(0)$ also works).
Hence $P_r = \{(0,1,4,5)\}$ (here (0,1,4,5) is used to denote a set for the sake of clarity), and $S = \{(0,2,8,10),(1,3,9,11)\}$.
$A(0) = A(2) = \{0,1,4,5\}$.
Select (0,2,8,10) as X. Since 0 and 2 were paired initially as u and v (or x and y), here 8 and 10 are paired. Thus there is no change in P_r, and $S = \{(1,3,9,11),(4,6,12,14),(5,7,13,15)\}$.
Now $A(8) = A(10) = \{0,1,4,5\}$.
Proceeding in this manner, the $DM(2,2)$ which has the left half of the table of the adjacency lists of $UDM(2,2)$ given above as its adjacency lists is obtained. Different selections for u and/or v produce different shift register codings of $DM(2,2)$.

Theorem 9 *G is a $DM(k_1,...,k_m)$ iff the algorithm BDBSR_TEST when applied to it returns Yes.*

Proof As in algorithm BDBSR_TEST let G be the undirected graph obtained from the given digraph G and let $A(v)$ denote the adjacency list of vertex v of G. Select a vertex u such that u appears twice in $A(u)$ and label it with $\hat{0}$. By Remark 3 there must be a shift register coding with $\hat{0}$ for u.
Let v be an arbitrary vertex in $A(0)$. v can be considered an in-edge without loss of generality. For if the arrows are reversed in a DM, DM is still a DM isomorphic to the original DM.

Then by Lemma 14 $A(0) \cap A(v)$ is the set of out-edges of 0 and v, $A(0) - A(0) \cap A(v)$ is the set of in-edges of 0, and $A(v) - A(0) \cap A(v)$ is the set of in-edges of v.
Similarly the set of out-edges and the set of in-edges are identified for all the vertices of G. Hence the adjacency list of every vertex of G is obtained.
If the given G is a DM, then these adjacency lists obtained from the undirected G are those of a DM and they clearly satisfy $P_C * P_R = \underline{0}$. Thus BDBSR_TEST returns Yes.
Conversely if BDBSR_TEST returns Yes, then the undirected graph G is UDM, hence no matter what direction is given to its edges, the resultant digraph is an STG of a bidirectional shift register.

QED

7 Conclusion

No efficient algorithm is known for finding shift registers to realize a given sequential circuit even if it is known to be realizable with shift registers. Algorithm SRR and its generalization GSRR reported in the report by this author [23] also are not efficient for certain types of circuits. This inefficiency is due to the fact that in general some of the states of G^i are mapped to only one state by the transition function. If that happens, the partition of the state set of G^i is not unique and in the worst case all possible partitions must be examined before the right partition is found. In this paper two of the worst cases have been examined. They are simple cycles and simple cycles with a chord. Incidentally these are also two of the worst cases for the method of Roome and Torng [21]. It has been found that simple cycles of any length n can be shift register coded with a code of length $\lg n$. For simple cycles with one chord, that is exactly one state is mapped to two states, and the rest are mapped to one state, the problem has been solved only partially. That is, if the cycle determined by the chord is of length less than or equal to $n/4$, or equal to $n/2$, and $n = 2^k$ for some integer k, then it can be shift register coded. For the rest of the cases the problem has not been solved. It is conjectured that any simple cycles with a chord can be shift register coded.

The two cases discussed in this paper are relatively simpler cases because it is the given transition graph that is a cycle. However, any of G^i's can become a cycle even if the original graph is not a cycle. These obviously include much more graphs. Identifying them and developing an efficient method for finding a shift register coding for them are open problems.

If bidirectional shift registers are used, obviously more transitions can be accommodated. One important question here is how to test if a given sequential machine can be realized by bidirectional shift registers. In this paper a necessary and sufficient condition for a sequential machine to be realizable with one bidirectional shift register has been established for sequential machines whose transition graph becomes a undirected de Bruijn graph if the direction of the transitions are ignored. Again the general case is an open problem.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design.* Computer Science Press, 41 Madison Avenue, New York, NY, 1990.

[2] V. D. Agrawal and K.-T. Cheng. Test function specification in synthesis. In *Proc. 27th ACM/IEEE Design Automation Conference*, pages 235–240, 1990.

[3] K. T. Cheng and V. D. Agrawal. Design of sequential machines for efficient test generation. *ICCAD*, pages 358–361, 1989.

[4] K. T. Cheng and V. D. Agrawal. A partial scan method for sequential circuits with feedback. *IEEE Trans. Comput.*, pages 544–548, April 1990.

[5] K. T. Cheng and V. D. Agrawal. An economical scan design for sequential logic test generation. *19th Int'l Symp. on Fault Tolerant Comp.*, pages 28–35, June 1989.

[6] S. Devadas and K. Keutzer. A unified approach to the synthesis of fully testable sequential machines. *IEEE Trans. Computer-Aided Design*, pages 39–50, January 1991.

[7] S. Devadas, H-K Ma, A.R. Newton, and A. Sangiovanni-Vincentelli. A synthesis and optimization procedure for fully and easily testable sequential machines. *IEEE Trans. Computer-Aided Design*, pages 1100–1107, October 1989.

[8] T. Etzion and A. Lempel. Algorithms for the generation of full length shift register sequences. *IEEE Trans. Inform. Theory*, IT-30, No. 3, 1984.

[9] H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *SIAM Rev.*, 24:195–220, April 1982.

[10] H. Fujiwara and K. Kinoshita. Design of diagnosable sequential machines utilizing extra outputs. *IEEE Trans. Comput.*, pages 138–145, February 1974.

[11] H. Fujiwara, Y. Nagao, T. Sasao, and K. Kinoshita. Easily testable sequential machines with extra inputs. *IEEE Trans. Comput.*, pages 821–826, August 1975.

[12] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Trans. Comput.*, pages 1137–1144, December 1983.

[13] P. Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Trans. Comput.*, pages 215–222, March 1981.

[14] S. W. Golomb. *Shift Register Sequences.* Aegean Park Press, Laguna Hills, CA 92653, revised ed. edition, 1982.

[15] Y. Huang. A new algorithm for the generation of binary de bruijn sequences. *J. of Algorithms*, 11:44–51, 1990.

[16] C. L. Liu. K-th order finite automaton. *IEEE Trans. Comput.*, pages 470–475, October 1963.

[17] R. Z. Makki, S. Bou-Ghazale, and C. Tianshang. Automatic test pattern generation with branch testing. *IEEE Trans. Comput.*, pages 785–791, June 1991.

[18] R. L. Martin. *Studies in feedback-shift-register synthesis of sequential machines.* MIT Press, Cambridge, MA, 1969.

[19] A. J. Nichols. Minimal shift-register realizations of sequential machines. *IEEE Trans. Elect. Comput.*, pages 688–700, October 1965.

[20] B.-H. Park and P. R. Menon. Design of scan-testable CMOS sequential circuits. In *Proc. International Test Conference*, pages 369–376, 1990.

[21] W. D. Roome and H. C. Torng. Algorithms for multiple shift register realizations of sequential machines. *IEEE Trans. Comput.*, pages 933–943, October 1973.

[22] M. Schulz and E. Auth. Advanced automatic test pattern generation and redundancy identification techniques. In *Proc. of Fault Tolerant Computing Symp.*, pages 30–35, 1988.

[23] S. Toida. On shift register realization of sequential circuits. Technical Report TR-96-21, Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162, September 1996.

[24] T. W. Williams and K. P. Parker. Design for testability - a survey. *IEEE Trans. Comput.*, pages 2–15, January 1982.

[25] S. Xie. Note on de bruijn sequences. *Discrete Applied Math.*, 16:157–177, 1987.

[26] M. Yoeli. Binary ring sequences. *Amer. Math. Monthly*, 69:852–855, 1962.

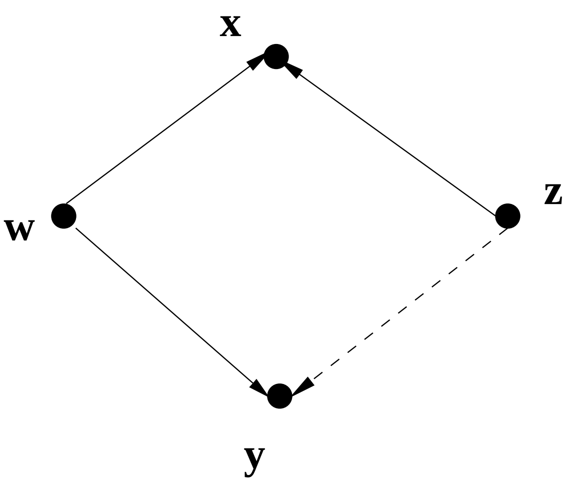
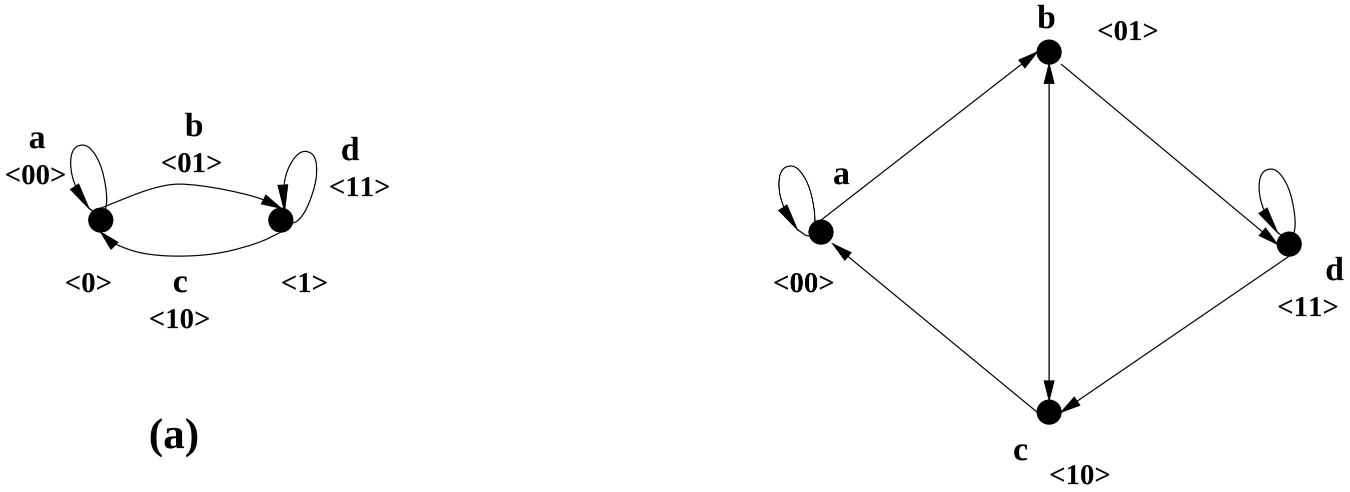


Fig. 1



(a)

(b)

Fig. 2

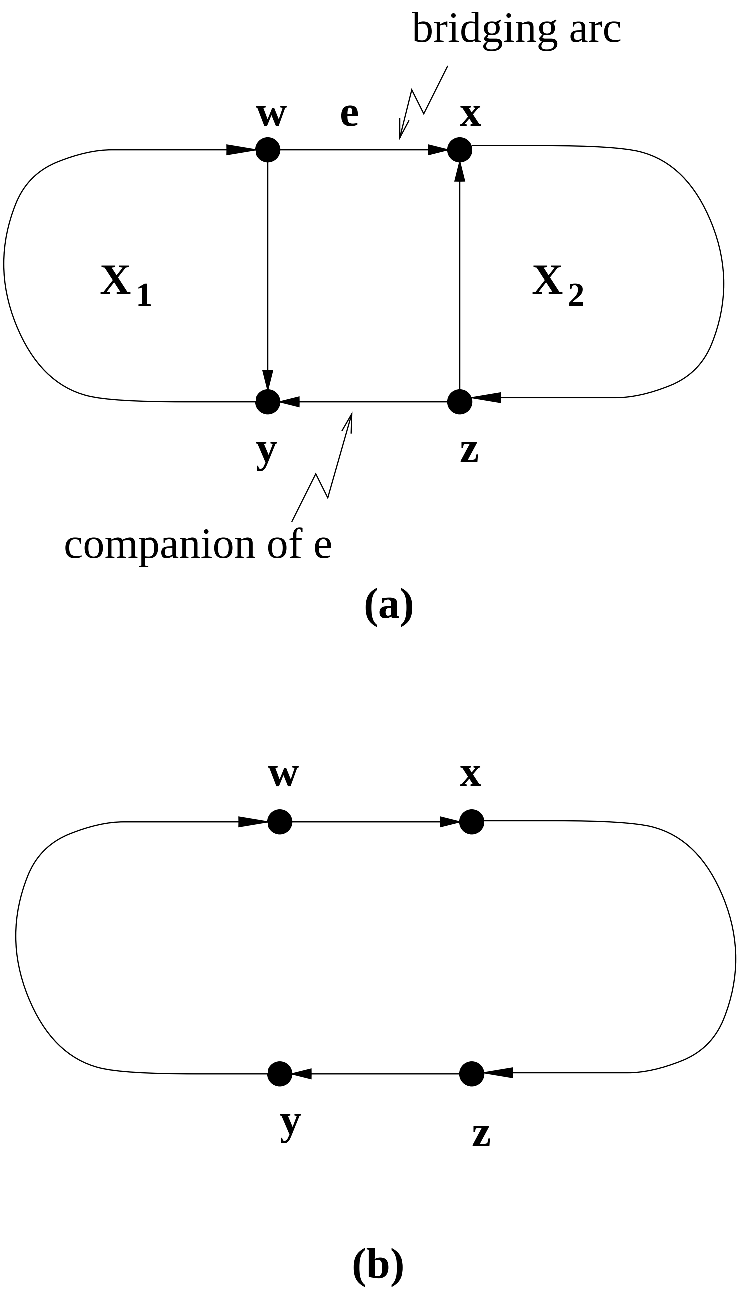
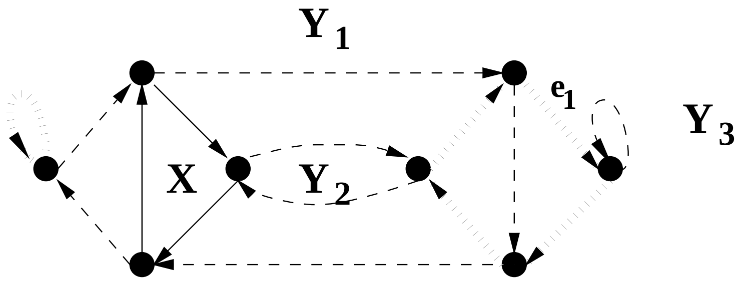
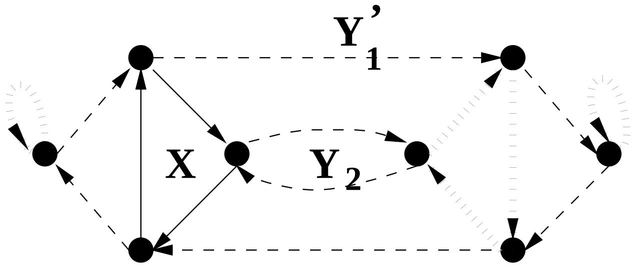


Fig. 3

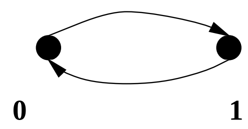


(a)

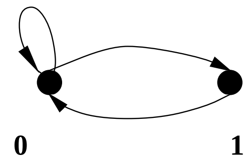


(b)

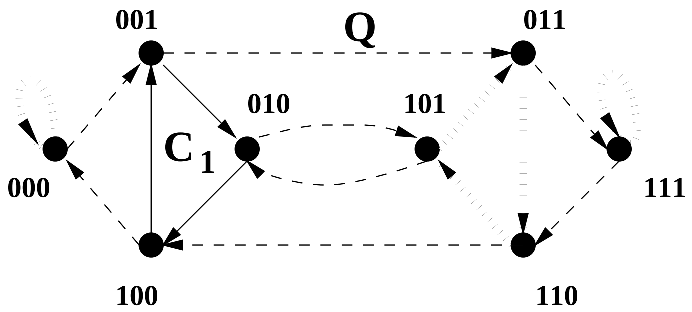
Fig. 4



(a) Q in DM(1)



(b) Closed trail W



(c) C_1 and Q in DM(3)

Fig. 5

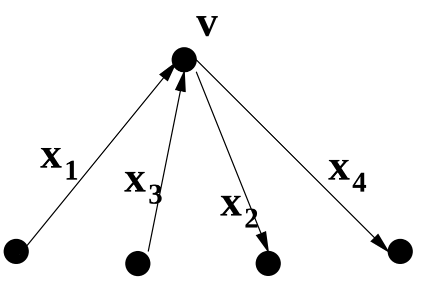


Fig. 6 (a) Vertex v in DM(k-2)

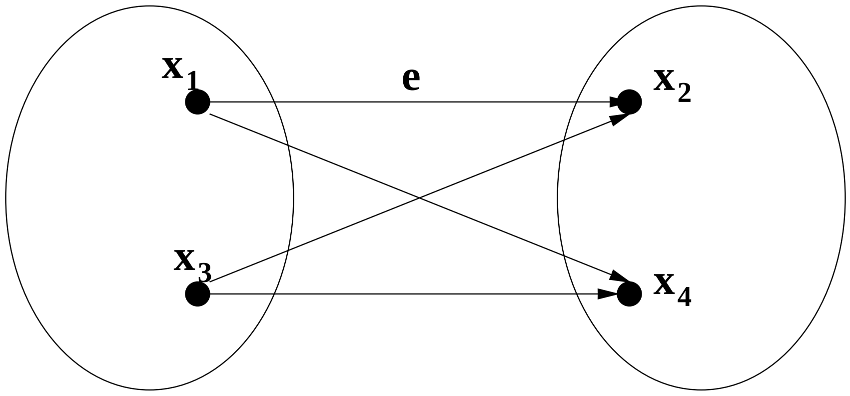


Fig. 6 (b) Vertex v in DM(k-1)-C'

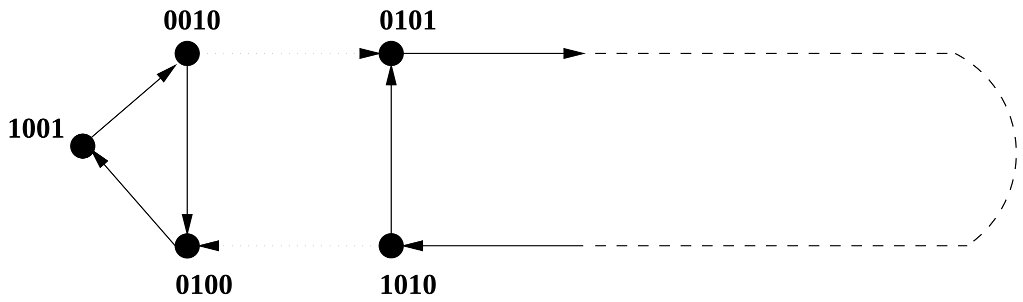


Fig. 7 (a) Two Cycles in DM(4)

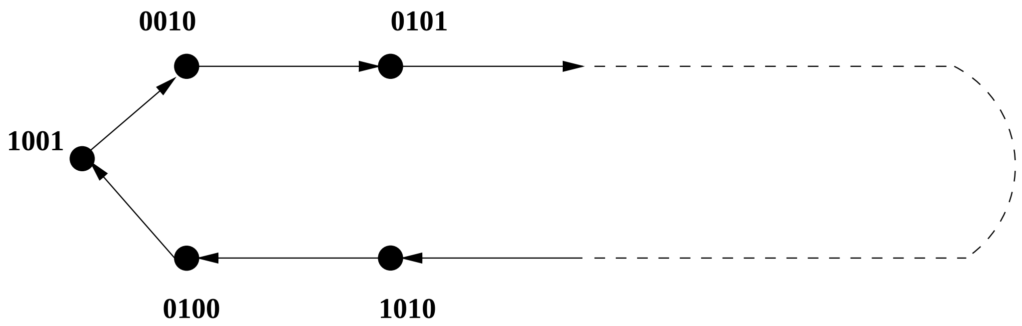


Fig. 7 (b) Cycle of Length 16 in DM(4)

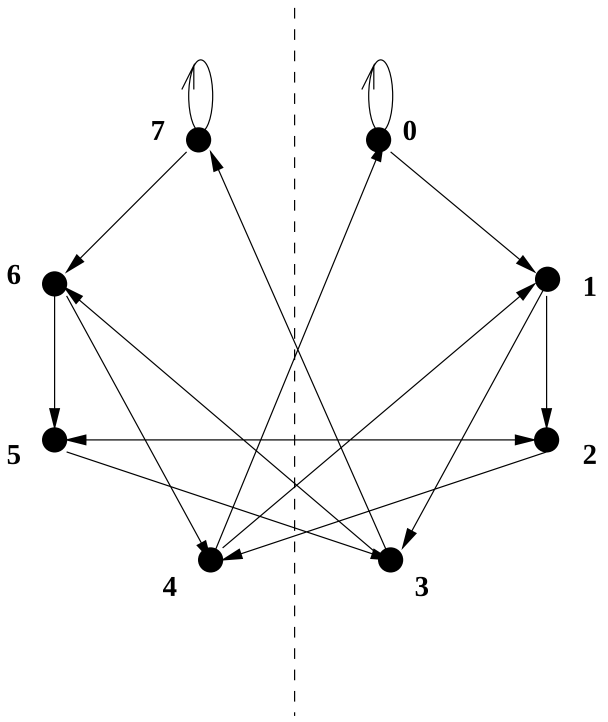


Fig. 8 (a) Symmetry in DM(3)

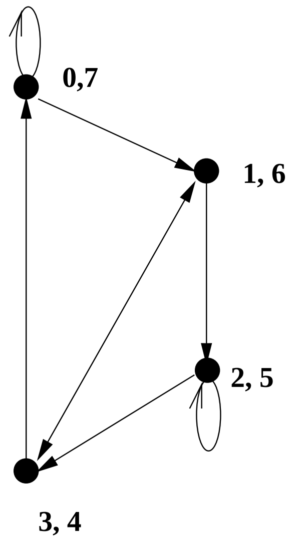


Fig. 8 (b) DM(4) folded along L

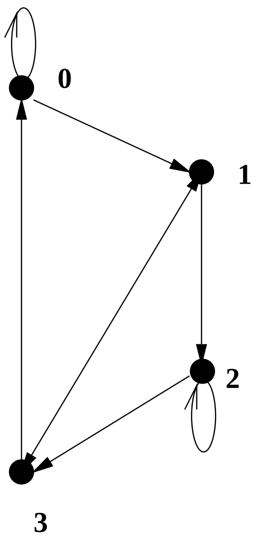


Fig. 9 (a) H_2

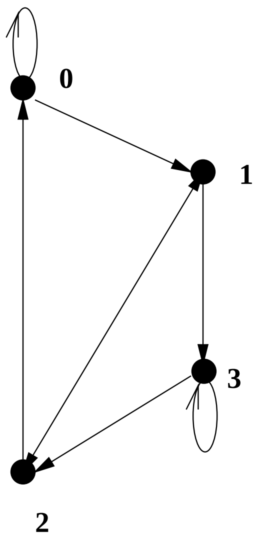


Fig. 9 (b) G_2 : Relabeled H_2

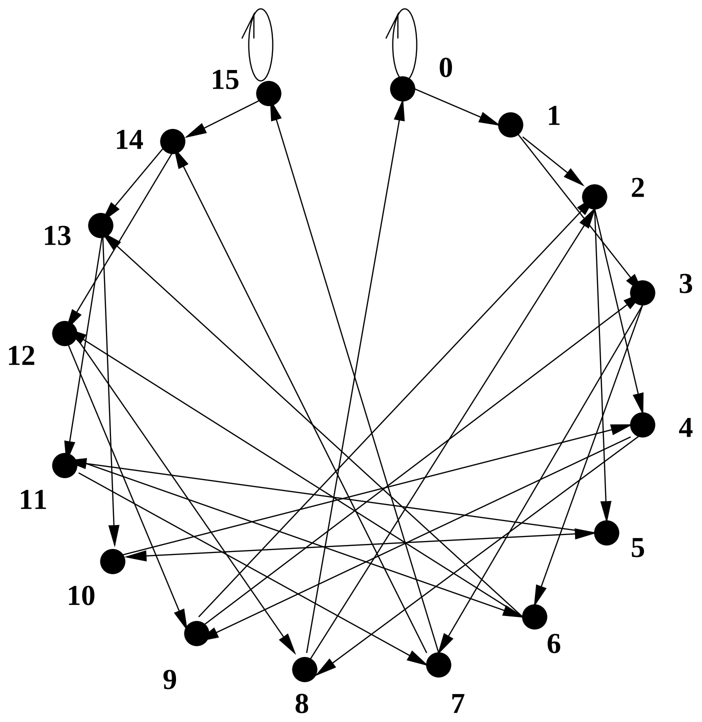


Fig. 10 (a) DM(4)

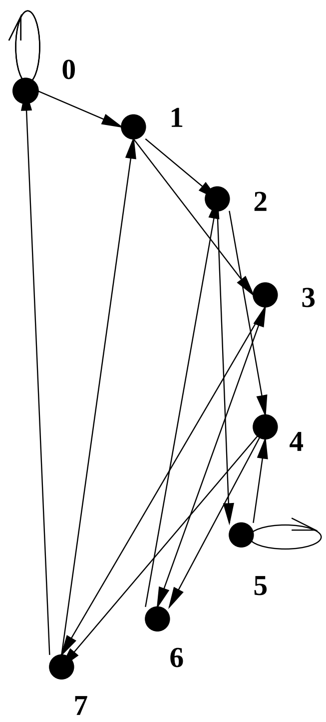


Fig. 10 (b) H_3