RICE UNIVERSITY

DELAY LINE SYNTHESIS OF SEQUENTIAL MACHINES

by

Shreehari Marathe

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Science

Thesis Director's Signature:

Houston, Texas

July, 1970
DELAY LINE SYNTHESIS OF SEQUENTIAL MACHINES

by

Shreehari Marathe

ABSTRACT

This thesis deals with the problem of realizing state sequential machines with delay line networks which are a special case of shift register networks. A formal definition of delay line networks is given with an algebraic model in terms of the following four parameters: the input set $I$, the number of delay lines $m$, the length of delay line $l$, and the logic function $f$. With the help of this model, the concept of a state being in a cycle of length $c$ is defined and it is shown that such a state is periodic with period $c$. The following result concerning the periodicity of doubly periodic finite sequence is proved: if a finite sequence is periodic with periods $u$ and $v$ and if the length of the sequence is greater than or equal to $u+v-g$ where $g$ is the greatest common division of $u$ and $v$, then the sequence is periodic with period $g$. These results are related by proving that if a state is in two cycles of length $u$ and $v$, then the length of the delay line must be less than $u+v-g$. 
A method is developed to derive bounds for the length and number of delay lines for a delay line network to realize a given machine. A constructive synthesis algorithm is given to find a delay line realisation of given machine in the above bounds if such an assignment exists. This algorithm utilizes a state assignment procedure to assign the states of a given machine to the states of a specific delay line network.
TO VAL
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................... Page 1

1. INTRODUCTION ........................................... 1

2. A DELAY LINE NETWORK MODEL ........................................... 5
   2.1 Formalization of the Model ........................................... 5
   2.2 Properties of Delay Line Networks ........................................... 8

3. SYNTHESIS ALGORITHM ........................................... 18
   3.1 Procedure for State Assignment ........................................... 18
   3.2 Analysis of the State Assignment Procedure ........................................... 24
   3.3 An Example ........................................... 27
   3.4 Bounds on the Length of Delay Lines ........................................... 29
   3.5 Bounds on the Number of Delay Lines ........................................... 30
   3.6 The Complete Synthesis Algorithm ........................................... 31

4. SUMMARY AND CONCLUSION ........................................... 33

APPENDIX: DETERMINATION OF CYCLE LENGTHS ........................................... 35

REFERENCES ........................................... 37
ACKNOWLEDGEMENTS

The author wishes to express his gratitude to his advisor Dr. J. R. Jump for his encouragement and numerous valuable suggestions.

The National Science Foundation and Rice University are to be thanked for supporting this research.
INTRODUCTION

In this thesis, we are concerned with the realization of sequential machines using binary shift registers. We will be dealing with a special class of binary shift registers. A binary shift register consists of a series of flip-flops having two states. The flip-flops are connected in such a manner that when a shift signal is received (sometimes called the clock signal), the \((m-1)\)-th flip-flop changes its state to that of the \(m\)-th flip-flop. The last flip-flop has an input line (sometimes two). The state of the input line, when the shift signal is received, determines the state of the last flip-flop. We say that a flip-flop has a tap line if there is an output from that flip-flop which tells its state at any instant of time. In a shift register, several (or all) flip-flops have tap lines. We shall deal with a class of shift registers which have a tap only on its first flip-flop. These are called delay lines. In this thesis, we are trying to realize sequential machines using networks of delay lines and logical gates called delay line networks.

A typical delay line network is shown in figure 1.1. There are \(m\) - delay lines each having \(1\) - flip-flops. The taps from the first flip-flops of these delay lines along with external input lines are connected to the combinational logic. The \(1\) - outputs of this combinational logic are used as inputs to the last flip-flops. The network is synchronized with a clock signal. Whenever a clock signal is received the \((m-1)\)-th flip-flop in each delay line assumes the state of the \(m\)-th flip-flop in that delay line. The states of the last flip-flops are determined by the outputs of the combinational logic which in turn are
Figure 1.1
determined by the states of the taps before the clock signal is received and the current states of the external input lines. It may be noted that the delay lines need not necessarily be made of flip-flops. Sonic delay lines also could be used in such a network.

Considerable amount of research has been done in realisation of sequential machines using shift registers that have taps on all the flip-flop elements. There are several reasons for this interest. Any sequential machines having less than $2^n$ states can be realized by using $n$ flip-flops and appropriate combinational logic. In general the outputs of all the flip-flops are needed to determine the next states of these flip-flops. Thus the combinational logic needs $n$ input lines and $n$ output lines. Thus the logic is quite complex. If the same machine could be realized using 1 or 2 shift registers, then the complexity of the combinational logic would considerably reduce since it would need to have only 1 or 2 outputs. The reduction of complexity of interconnection would increase the reliability and reduce the cost. Another factor which influences the research is the availability of cheap microminiaturised shift registers which can easily be used as memory elements. Davis [2] has given a procedure to realise machines using a single shift register. Johnson and O'Keefe [3,4] have used shift registers as a special case of their work involving realisation of machines using $n$-identical modules. Su and Yau [5] have considered the problem with multiple state assignments. Nichols [6] has dealt with the same problem in his minimum shift register realisation of sequential machines.
Let us consider the advantages of using delay lines over shift registers. If we consider medium scale integration, we would be constructing shift registers with integrated circuits. The complexity of integrated circuits is limited by the number of external pins that can be brought out in the construction. A typical flat-pack has 14 pins. Two of these are for power supply. If we use one for the shift signal, we are left with 11 pins. Hence we can have a shift register of at most 10 flip-flops all of which are tapped. On the other hand if we were to have delay lines in a similar flat-pack, we can have 5 delay lines with any number of flip-flops as each needs only 2 pins. One for the tap on the first flip-flop and one as an input to the last flip-flop. In a general realization of sequential machines using shift registers, the number of taps required is not fixed. On the other hand if we have n delay lines then we have only n taps and hence the number of inputs of the combinational logic is fixed. Also the number of outputs of the combinational logic is fixed. Hence we can realize various machines by using universal logic elements having fixed number of inputs and outputs. Thus a flat-pack having a fixed number of delay lines along with a flat-pack of universal logic can become standard building blocks for realization of sequential machines.

It should be noted that not all sequential machines can be realized using delay line networks. In this thesis we will derive an upper bound on the length of delay lines that can be used in realizing a given machine. The bound is based upon the cycle lengths of different states of the machine. In the appendix, a procedure is given to calculate the cycle lengths. It may be noted that the same bound applies to the
length of shift registers that can be used to realize the given machine. We will develop a procedure for assigning the states of the machine to the states of a delay line network. A synthesis algorithm is given in the third chapter to determine whether a given machine can be realized as a delay line network based on the bounds on the length of delay lines and the number of delay lines.
2. A DELAY LINE NETWORK MODEL

In this chapter we will define an abstract model for the class of delay line networks which was described in the introduction. This will allow us to define the 'behavior' of a delay line network and the concept of a delay line network 'realizing' a sequential machine. In addition, we will derive several salient properties of the model which will be useful in the development of a synthesis algorithm.

2.1 Formalization of the Model:

A delay line network is completely characterized by its input set 'I', the number of its delay lines 'm', the length of each delay line 'l' and a function 'f' which maps the output of the delay lines and the input set into the inputs to the delay lines. We assume that the elements of the delay lines assume values from the set \{0,1\}.

**Definition 2.1.1**

An abstract delay line network is a 4-tuple \(N = (I, m, l, f)\) where:

1. \(I\) is a finite set of input symbols,
2. \(m\) is the number of delay lines,
3. \(l\) is the length of (or the number of flip-flops in) the delay line, and
4. \(f\) is a function mapping \([0,1]^m\times I\) into \([0,1]^m\) called the logic function of \(N\).
We have not mentioned the set \( \{0,1\} \) in the 4-tuple as it is implicitly assumed that the elements of the delay line assume binary values.

Since the properties we are trying to study do not involve outputs, we will use the following model of a sequential machine.

**Definition 2.1.2**

A **state sequential machine** is a 3-tuple \( M = (I, Q, d) \) where:

1. \( I \) is the finite nonempty set of **input symbols**, 
2. \( Q \) is the finite nonempty set of **states**, and 
3. \( d \) is a function mapping \( Q \times I \) into \( Q \) called the **state transition function**.

With the following definition we can interpret a delay line network as a state sequential machine.

**Definition 2.1.3**

Let \( N = (I, m, 1, f) \) be a delay line network. Then the **machine realized by** \( N \) is the sequential machine \( M(N) = (I, Q', g) \) where:

1. \( I \) is the finite set of input symbols 
2. \( Q' \) is the set of states defined as a sequence of length 1 
   \[ Q' = \{ m_4 \ m_2 \ \ldots \ m_1 \mid m_1 \in \{0,1\} \} \]
3. \( g \) is the state transition function mapping \( Q' \times I \) into \( Q' \) and is defined as follows:
   \[ g (m_4 \ m_2 \ \ldots \ m_1) = (m_2 \ m_3 \ \ldots \ m_1 \ m_1+1) \]
   where \( m_{1+1} = f (m_1, 1) \)

The function \( 'g' \) is called the 'behavior' of the network \( N \).
At this point we can define precisely the concept of a delay line network realizing a sequential machine.

**Definition 2.1.5**

Given a sequential machine \( M = (I, Q, d) \), we say that \( M \) is realisable as a delay line network if there exists a delay line network \( N = (I, m, 1, f) \) and a one-to-one function \( h \) (called assignment function) mapping the state set \( Q \) into the state set \( Q' \) of \( M(N) \) such that

\[
h(d(q,i)) = g(h(q),i) \quad \text{for all } i \in I \text{ and all } q \in Q.
\]

Note that this definition is equivalent to saying that \( M \) is an isomorphic image of a submachine of \( M(N) \).

We shall illustrate the above definitions by means of an example.

**Example No. 1**

Consider a delay line network \( N = (I, m, 1, f) \) where \( I = \{0,1\} \), \( m = 1 \), \( 1 = 2 \), and \( f \) is given by

\[
\begin{array}{c|cc}
  f & i_1 = 0 & i_2 = 1 \\
  \hline
  0 & 1 & 0 \\
  1 & 1 & 0
\end{array}
\]

The sequential machine realized by \( N \) is \( M(N) = (I, Q', g) \) where \( I = \{0,1\} \), \( Q' = \{00, 01, 10, 11\} \) and \( g \) is given by

\[
\begin{array}{c|cc}
  g & i_1 = 0 & i_2 = 1 \\
  \hline
  00 & 00 & 01 \\
  01 & 10 & 11 \\
  10 & 01 & 00 \\
  11 & 11 & 10
\end{array}
\]
The physical model corresponding to this network is:

This network realizes the machine \( M = (I, Q, d) \) where \( I = \{0, 1\} \), \( Q = \{q_1, q_2, q_3, q_4\} \), and \( d \) is given by

<table>
<thead>
<tr>
<th>( d )</th>
<th>( i_1 = 0 )</th>
<th>( i_2 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>( q_1 )</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_3 )</td>
<td>( q_4 )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_2 )</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( q_4 )</td>
<td>( q_3 )</td>
</tr>
</tbody>
</table>

Where the assignment function \( h \) is given by

<table>
<thead>
<tr>
<th>( q )</th>
<th>( q_1 )</th>
<th>( q_2 )</th>
<th>( q_3 )</th>
<th>( q_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h(q) )</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

This can be checked by noting that for all \( q \in Q \) and all \( i \in I \)

\( h(d(q, i)) = g(h(q), i) \)

2.2 Properties of Delay Line Networks

In this section we will study the cycle structure of delay line networks. We begin by extending the next state function to input sequences in the usual way.

**Definition 2.2.1**

Let \( M = (I, Q, d) \) be a state sequential machine and let

\( J = i_1 i_2 \ldots i_k \) be a sequence of \( k \) input symbols, then
the terminal state function \( d^* \) is defined inductively as follows:

1. If \( k = 1 \) then \( d^*(q,J) = d^*(q, 1_1) = d(q, 1_1) \)
2. If \( k > 1 \) then \( d^*(q,J) = d^*(q, 1_1 1_2 \ldots 1_k) \)
   \[ = d(d^*(q, 1_1 1_2 \ldots 1_{k-1}), 1_k) \]

**Definition 2.2.2**

Let \( M = (I, Q, d) \) be a state sequential machine. We say that a state \( q \) is in a cycle of length \( C (C \geq 1) \) if and only if there exists an input sequence \( J = 1_{1} 1_{2} \ldots 1_{C} \) such that \( d^*(q,J) = q \) and for all input subsequences \( J_d = 1_{d} 1_{d+1} \ldots 1_{t} \) of length less than \( C, d^*(q,J_d) \neq q \).

With these definitions, we are ready to derive some properties possessed by the states of a delay line network.

**Lemma 2.2.1**

Let \( N = (I, m, f) \) be a delay line network and let \( M(N) = (I, Q', g) \) be the sequential machine realized by \( N \). Let \( q = m_1 \ldots m_L \) be a state of \( M(N) \). Let \( J = 1_{1} 1_{2} \ldots 1_{C} \) be an input sequence where \( C \leq 1 \). Then \( g^*(q,J) = q \) (or state \( q \) is in a cycle under \( J \)) if and only if \( n_{i+c} = n_i \) for all \( 0 < i \leq 1-c \) and \( n_{i+j} = f(m_j, 1_{i}) = n_{1-c+j}, \forall j, 0 < j \leq c \).

**Proof:**

**Necessary Condition:** The proof can be seen by considering the sequence of states through which the machine goes as the input sequence \( J \) is applied.
Input Sequence  

<table>
<thead>
<tr>
<th>Sequence</th>
<th>State Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>$m_1 m_2 \cdots m_i = q$</td>
</tr>
<tr>
<td>$i_1$</td>
<td>$m_2 m_3 \cdots m_i m_{i+1} = g(q, i_1)$</td>
</tr>
<tr>
<td>$i_1 i_2$</td>
<td>$m_3 m_4 \cdots m_{i+1}m_{i+2} = g^*(q, i_1 i_2)$</td>
</tr>
</tbody>
</table>

But since $g^*(q, J) = q$, we get $m_{c+1} \cdots m_{i+1} m_{i+2} = g^*(q, i_1 i_2)$

and $m_{c+1} = m_1$

$m_{c+2} = m_2$

$\vdots$

$m_1 = m_{1-c}$

So that $m_{1+c} = m_1$ for all $i$ such that $0 < i \leq 1-c$; and

$m_{i+1} = f(m_i, i_1) = m_{1-c+i}$

$m_{i+2} = f(m_i, i_2) = m_{1-c+i+2}$

$\vdots$

$m_{1+c} = f(m_{c}, i_1) = m_1$

So that $m_{i+j} = f(m_i, i_j) = m_{1-c+j}$ for all $j$ such that $0 < j \leq c$

Sufficient condition: If we go through the above steps in exactly the reverse direction, it would be obvious that if the sequence $J$ is applied the machine would go back to the original state 'q'. QED.

We can observe from the above lemma that a state which is in a cycle of length 'c' has a periodic property in 'm'. Since $m_{1+c} = m_1$, we see that the state has at most $c$ distinct $m$ elements.
Lemma 2.2.2

If a state 'q' of M(N) is in a cycle of length c (c ≤ 1) then there is no integer d such that d divides c and \( m_{1+d} = m_1 \) for all i such that \( 0 < i \leq 1-d \).

Proof:

Since the state 'q' is in a cycle of length c, there exists an input sequence \( J = i_1 i_2 \ldots i_c \) such that \( g^*(q, J) = q \). We will prove the result by contradiction. Suppose there is a 'd' such that \( c = kd \) and \( m_{1+d} = m_1 \) for all \( i \) such that \( 0 < i \leq 1-d \). Let us see what happens to the state of the machine when part of the input sequence \( J = i_1 i_2 \ldots i_d \) is applied.

<table>
<thead>
<tr>
<th>Input Sequence</th>
<th>Machine Goes To State</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>( m_1 m_2 \ldots m_1 = q )</td>
</tr>
<tr>
<td>( i_1 )</td>
<td>( m_2 m_3 \ldots m_1 = g(q, i_1) )</td>
</tr>
<tr>
<td>( i_1 i_2 \ldots i_d )</td>
<td>( m_{d+1} m_{d+2} \ldots m_1 \ldots m_{1+d} = g^*(q, J_1) ).</td>
</tr>
</tbody>
</table>

But \( m_{1+d} = m_1 \) for \( 0 < i \leq 1-d \), so that \( g^*(q, J_1) = m_1 \ldots m_{1-d} m_{1+1} \ldots m_{1+d} \).

Now from Lemma 2.2.1 we have \( m_1 + j = m_1 - c + j = m_1 - kd + j = m_1 - d + j \) for all \( J \) such that \( 0 < j \leq c \).

Thus we get \( m_{1+d} = m_{1-d+4} \)

\( m_{1+2} = m_{1-d+2} \)

\( m_{1+d} = m_1 \).

So that \( g^*(q, J_1) = m_1 m_2 \ldots m_{1-d} m_{1-d+1} \ldots m_1 = q \).

But this is a contradiction to definition of cycle. QED.
In the following lemmas and theorems we will establish a relation between the cycle lengths of a state of \( M(N) \) and the length of delay line of \( N \). This will be done by deriving a property of finite periodic sequences (Theorem 2.2.5). This theorem in conjunction with Lemmas 2.2.1 and 2.2.2 will be used to prove the relation between cycle lengths and length of delay line (Theorem 2.2.6).

**Lemma 2.2.3**

Let \( S = m_1^n \ldots m_k \) be a finite sequence which is periodic with period \( t \). Thus it satisfies \( n_i+t = m_i, i+t \leq 1 \).

Then \( m_{1+kt} = m_1 \) for \( 1+kt \leq 1 \) where \( k \) is a positive integer.

**Proof:**

We have \( m_{1+kt} = m_{1+(k-1)t} \) for \( 1+kt \leq 1 \). But \( 1 + kt \leq 1 \) implies \( 1 + (k-1)t \leq 1 \). Thus \( m_{1+(k-1)t} = m_{1+(k-2)t} \). Proceeding in the same manner we get \( m_{1+kt} = m_1 \) for \( 1+kt \leq 1 \). QED.

**Lemma 2.2.4**

Let \( S = m_1^n \ldots m_k \) be a finite sequence with period \( u \).

Thus \( m_{1+u} = m_1 \) for \( 1+u \leq 1 \). Let the subsequence \( S_1 = m_1^n \ldots m_k \) be periodic with period \( v \) and let \( v \) divide \( u \).

Thus \( m_{1+v} = m_1 \) for \( 1+v \leq u \). Then \( S \) is periodic with period \( v \), that is \( m_{1+v} = m_1 \) for \( 1+v \leq 1 \).

**Proof:**

Consider \( m_{1+v} \) for all \( i \) such that \( 1+v \leq 1 \). Since \( u \) is smaller than \( 1 \), there exists a positive integer \( k \) such that \( 1+v-ku \leq u \). Now \( m_{1+ku} = m_{1+v} - ku \) since \( 1+v-ku \leq u \). But by Lemma 2.2.3 \( m_{1-ku} = m_1 \) and \( m_{1+v-ku} = m_{1+v} \). Hence we get \( m_{1+v} = m_1 \) for \( 1+v \leq 1 \). QED
Theorem 2.2.5

Let \( n_1, n_2, \ldots, n_i \) be a finite sequence such that
\[ m_{i+u} = n_1 \text{ for } i + u \leq 1, \]
\[ m_{i+v} = n_1 \text{ for } i + v \leq 1, \]
and
\[ 1 \geq u + v - g \] where \( g \) is the greatest common divisor of \( u \) and \( v \). Then
\[ m_{i+g} = n_1 \text{ for } i + g \leq 1. \]

Proof:

The proof involves the step-by-step carrying out of the procedure to determine the greatest common divisor by means of the Euclidean Algorithm.

1. Let \( u = k_1 u + u_1 \). We have \( m_{i+u} = n_1 \) for \( i + u \leq 1 \), thus we get
\[ m_{i+k_1 u+u_1} = n_1 \text{ for } i + k_1 u + u_1 \leq 1. \]
But \( m_{i+u} = n_1 \) for \( i + u \leq 1 \).

Hence from lemma 2.2.3, we get \( m_{i+u_1} = n_1 \) for \( i + u_1 \leq 1 - k_1 u \).

2. Let \( u = k_2 u_1 + u_2 \). We have \( m_{i+u} = n_1 \) for \( i + u \leq 1 \), have
\[ m_{i+k_2 u_1+u_2} = n_1 \text{ for } i + k_2 u_1 + u_2 \leq 1. \]
But \( m_{i+u_1} = n_1 \) for \( i + u_1 \leq 1 - k_1 u \).

Hence from lemma 2.2.3, we get
\[ m_{i+u_2} = n_1 \text{ for } i + u_2 \leq 1 - k_1 u - k_2 u_1 \]

3. Let \( u_1 = k_3 u_2 + u_3 \). We have \( m_{i+u} = n_1 \) for \( i + u \leq 1 - k_1 u \),

Hence \( m_{i+k_3 u_2+u_3} = n_1 \text{ for } i + k_3 u_2 + u_3 \leq 1 - k_1 u \).

But \( m_{i+u_2} = n_1 \) for \( i + u_2 \leq 1 - k_1 u - k_2 u \).

Hence from lemma 2.2.3, we get
\[ m_{i+u_3} = n_1 \text{ for } i + u_3 \leq 1 - k_1 u - k_2 u_1 - k_3 u_2 \]
We will proceed in this manner until we come to a value of $i$ such that $u_{i+1} = k_{i+3} u_{i+2}$ where $u_{i+2}$ is the greatest common divisor of $u_i$.

Let us write the last two steps.

1. let $u_{i-1} = k_{i+1} u_i + u_{i+1}$.

   By the same argument

   $$m_1 u_{i+1} = m_1$$

   for $i + u_{i+1} = 1 - k_1 u - k_2 u_1 - \ldots - k_{i+1} u_i$

1. let $u_1 = k_{i+2} u_{i+1} + u_{i+2}$.

   By the same argument

   $$m_1 u_{i+2} = m_1$$

   for $i + u_{i+2} = 1 - k_1 u - k_2 u_1 - \ldots - k_{i+2} u_{i+1}$

But $u_{i+2}$ is the gcd of $u, v$.

Hence $m_{i+1} = m_1$ for $i + g = 1 - k_1 u - k_2 u_1 - \ldots - k_{i+2} u_{i+1}$

Let us evaluate the quantities

$$1 - k_1 u - k_2 u_1 - \ldots - k_{j+1} u_j$$

for $j \leq i + 1$

We have the following set of equations

\[
\begin{align*}
v &= k_1 u + u_1, \\
u &= k_2 u_1 + u_2, \\
\vdots & \quad \vdots \\
u_{j-1} &= k_{j+1} u_j + u_{j+1}, \\
\vdots & \quad \vdots \\
u_{i-1} &= k_{i+1} u_i + u_{i+1}, \\
u_1 &= k_{i+2} u_{i+1} + u_2.
\end{align*}
\]

Summing the left hand and right hand sides of equations through $u_{j-1}$ we have

\[
\begin{align*}
v + u + \ldots + u_{j-1} &= k_1 u + k_2 u_1 + \ldots + k_{j+1} u_j + u_1 + u_2 + \ldots + u_{j-1} + u_j + u_{j+1} \\
\text{Hence } k_1 u + k_2 u_1 + \ldots + k_{j+1} u_j &= v + u - u_j - u_{j+1} \\
\text{So that } 1 - (k_1 u + k_2 u_1 + \ldots + k_{j+1} u_j) &= 1 - (v + u - u_{j+1}) + u_j.
\end{align*}
\]
Thus we get the following relations:

\[ n_{i+1} = n_i \quad \text{for } i + v \leq 1, \]

\[ n_{i+u} = n_i \quad \text{for } i + u \leq 1 - (v + u - u) + v, \]

\[ n_{i+u_1} = n_i \quad \text{for } i + u_1 \leq 1 - (v + u) - u_1 + u, \]

\[ n_{i+u_2} = n_i \quad \text{for } i + u_2 \leq 1 - (v + u - u_2) + u_1, \]

\[ n_{i+u_1} = n_i \quad \text{for } i + u_1 \leq 1 - (v + u - u_1) + u_1 - u \]

\[ n_{i+u_1+1} = n_i \quad \text{for } i + u_1+1 \leq 1 - (v + u - u_1+1) + u_1 \]

\[ n_{i+u_1+2} = n_i \quad \text{for } i + u_1+2 \leq 1 - (v + u - u_1+2) + u_1+1 \]

that is \( n_{i+u} = n_i \) for \( i + u \leq 1 - (v + u - g) + u_1 + 1 \)

Since \( 1 \geq v + u - g \) or \( 1 - (v + u - g) \geq 0 \)

and since \( g \) is less than \( u_1+2, u_1+1, \ldots, u_1 \) we have

\[ 1 \geq v + u - u_1+2 \text{ or } 1 - (v + u - u_1+2) \geq 0, \]

\[ 1 \geq v + u - u_1 \text{ or } 1 - (v + u - u_1) \geq 0. \]

Hence \( n_{i+g} = n_i \) for \( i + g \leq u_1+1 \) \( \Rightarrow \) \( \{1\} \)

\[ n_{i+u_1+1} = n_i \quad \text{for } i + u_1+1 \leq u_1 \]

\( \vdots \)

\[ n_{i+u_1} = n_i \quad \text{for } i + u_1 \leq u \]

\( \vdots \)

\[ n_{i+u} = n_i \quad \text{for } i + u \leq 1 \]

By using lemma 2.2.4 and \( \{1\} \) and \( \{2\} \) we get

\[ n_{i+g} = n_i \quad i + g \leq u_1 \]

Using lemma 2.2.4 repeatedly down this set of relations we get the desired result.

\[ n_{i+g} = n_i \quad \text{for } i + g \leq 1. \quad \text{QED.} \]
Now we shall give an example in which $1 < u + v - g$ where we cannot come to the above conclusion. Consider a sequence $S = 1010110101$ with length $l = 10$. For this sequence we can see that $n_{i+5} = n_i$ for $i + 5 \leq 10$ and $n_{i+7} = n_i$ for $i + u \leq 10$. Now $g = \text{gcd}(5, 7) = 1$ and hence $1 < 5 + 7 - 1 = 11$. Also it is obvious that $n_{i+1} \neq n_i$ for $i + 1 < 11$.

We shall derive in the next theorem, a relation between cycle lengths and the length of the delay line.

**Theorem 2.2.6**

Let $N^* = (I, m, 1, f)$ be a delay line network. Let $M(N)$ be the machine realized by $N$. Let 'q' be a state of $M(N)$ which is in two cycles of lengths $u$ and $v$. Then $1 < u + v - g$ where $g$ is the greatest common divisor of $u$ and $v$.

**Proof:**

Proof is given by contradiction. Suppose that $1 \geq u + v - g$. Now from lemma 2.2.1, we have

$n_{i+u} = n_i$ for $i + u \leq 1$,

and $n_{i+v} = n_i$ for $i + v \leq 1$.

Hence from theorem 2.2.5, we get

$n_{i+g} = n_i$ for $i + g \leq 1$.

But since $g$ divides $u$ and $v$, this contradicts lemma 2.2.2. Therefore $1 < u + v - g$.

QED.

The significance of this theorem is to put an upper bound on the length of a delay line that can be used to realize a given machine. It may be noted that this bound is also valid for shift register realization of sequential machines.

In the next chapter, we shall derive an algorithm for the realization of sequential machines as delay line networks.
3. SYNTHESIS ALGORITHM

In this chapter, we will develop an algorithm for determining whether a given sequential machine can be realized as a delay line network. The complete algorithm involves the determination of the length of the delay line, the number of delay lines and the logic function of the delay line network; and an assignment function mapping the states of the given machine into the states of the delay line network. We shall determine bounds on the length of the delay line and the number of delay lines. We shall also develop a state assignment procedure for determining whether the given machine can be realized as a delay line network with a specific length and number of delay lines. The complete synthesis algorithm in the last section consists of applying the state assignment procedure for all the lengths and number of delay lines in their respective bounds until an assignment is found if one exists.

3.1 Procedure for State Assignment

We are given the state sequential machine \( M = (I, Q, d) \) which we shall assume to be strongly connected. By strongly connected we mean that for any two states \( q_i \) and \( q_j \), there exists an input sequence \( J \) such that \( d^*(q_i, J) = q_j \) where \( d^* \) is the terminal state function.

Now for the delay line network \( N = (I, m, 1, f) \), we assume that the input set \( I \), the delay length \( l \) and the number of delay lines are fixed. To realize \( M \) as a submachine of \( M(N) \), the machine realized by \( N \), we have
to determine the logic function \( f: \{0, 1\}^n \times I \rightarrow \{0, 1\}^n \) and an assignment function \( h: Q \rightarrow Q' \) which maps the states of \( M \) into the states of \( M(N) \).

We will limit ourselves to one-to-one assignments. These are assignments in which one state of \( Q \) will be assigned to one and only one state of \( M(N) \). Thus we will realize \( M \) as an isomorphic image of a submachine of \( M(N) \). Without loss of generality, we will assume that the symbols in the input set \( I \) are integers \( 1, 2, \ldots \). We will also assume that the states in the state set \( Q \) of \( M \) are relabelled with integers \( 1, 2, \ldots \). We shall further assume that the relabelling is done in such a manner that for any entry state \( t(t \geq 2) \) in table 1 for the next state function \( d \) is such that there exist a state \( u \) in \( Q \) such that \( d(u, i) = t \) for some \( i \) and \( u \preceq t \). We can always do this since the machine is strongly connected.

\[
\begin{array}{c|cc|cca}
Q & 1 & 2 & 1 & |I| \\
\hline
1 & d(1,1) & d(1,2) & & \\
2 & d(2,1) & d(2,2) & & \\
\hline
l & & & d(l,i) & \\
|Q| & & & & \\
\hline
\end{array}
\]

Table 1

Thus we can denote the state to which the machine goes under an input \( i \) when in state \( t \) as \( d(t,i) \).
We have to determine the logic function \( f \) for the delay line network. We will relabel the \( m \)-tuples in the set \( \{0, 1\}^m \) with integers \( 0, 1, \ldots, 2^m - 1 \) in the obvious way by considering the \( m \)-tuple as a \( m \)-bit binary number. Let \( K = \{0, 1, 2, \ldots, 2^m - 1\} \). Thus the logic function \( f \) maps \( K \times I \) into \( K \) as shown in table 2.

| \( n \) | 1 | 2 | 1 | \( |1| \) |
|---|---|---|---|---|
| 1 |   |   |   |   |
| 2 |   |   |   |   |
| \( n \) |   |   |   | \( f(n, 1) \) |
| \( 2^m \, -1 \) |   |   |   |   |

Table 2
Thus we have to determine the various values of $f(m, i)$ as $m$ runs through $1, 2, \ldots, 2^n - 1$, and as $i$ runs through $1, 2, \ldots, |M|$.  

With the notation adapted for the set $\{0, 1\}^m$, the state set $Q'$ of $M(N)$ consists of sequences of length $1$ of symbols from $K$. Thus $Q' = \{m_1 m_2 \ldots m_i \mid m_i \in K\}$. We have to determine the assignment function $h$ as

<table>
<thead>
<tr>
<th></th>
<th>$Q$</th>
<th>$Q'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>$h(1)$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$h(2)$</td>
</tr>
<tr>
<td>$i$</td>
<td></td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$</td>
<td>Q</td>
<td></td>
</tr>
</tbody>
</table>

Table 3

shown in table 3. We have to determine the entries $h(1), h(2)$ \ldots $h(|Q|)$ from the set $Q'$. If a typical entry is $h(t) = m_1 m_2 \ldots m_i$, then we will use the notation $h^u(t)$ to denote the $u$-th symbol in the sequence. Thus $h^1(t) = m_1$, $h^2(t) = m_2$, \ldots, $h^{|Q|}(t) = m_i$.

Explanation of the state assignment procedure:

The state assignment procedure is flow-charted in figure 3.1.

We assign to state 1 of $M$ various sequences from the set $Q'$ starting with 000 \ldots 0. Then we go through the table 1 for the next state function row by row, carrying out assignments which are consistent with the 'behavior' of the machine $M(N)$. We, at the same time, construct the logic function $f$.

1. We initialize by setting the entries in table 2 and 3 to a special symbol 'b' called the blank symbol. 'k' denotes the current assignment of state 1 which is set to 000 \ldots 0.
Figure 3.1
Figure 3.1 Continued
Figure 3.1 Continued
2. If \( k = 2^m \), then we have tried all the sequences in \( Q' \) as assignments for state 1 and have failed in successfully carrying out the procedure.

3. Assign to state 1, the current value of \( k \).

4. \( N_a \) denotes the quantity \( h^1(a) \) for various values of \( a \). \( N_a \) takes the values 0, 1, \ldots, \( 2^m - 1 \) successively. In this step we initialize by setting it to zero.

5. The symbol \( t \) denotes the current row in which we are assigning the states the sequences from \( Q' \). The symbol 'j' the current input symbol we are considering.

7. Are the first 1-1 positions of the assignment \( d(f,j) \) consistent with the behavior of the machine \( H(N) \)?

8,9. If it is not, go back to row \( d(t,j) \).

11. If the logic function \( f(h^1(t),j) \) is not equal to the last symbol of \( h(d(t,j)) \) then go back to row \( d(t,j) \).

12. Since the logic function \( f(h^1(t),j) \) is blank, set it to the last symbol of \( h(d(t,j)) \).

13. Are all input symbols in this row considered?

14. Are all input symbols in this row considered? If all rows are considered then we are finished in successfully carrying out the procedure.

15. Is logic function \( f(h^1(t),j) \) blank?

16. Set the positions 1 through 1-1 of the assignment of \( d(t,j) \) consistent with the behavior of \( M(N) \).

17. Have all symbols of \( k \) been tried for \( h^1(d(t,j)) \)?

18. If yes, then go back to step 2 after incrementing \( k \) by 1.

21. Is this assignment for \( d(t,j) \) already in the table for some other state? If it is then we have to try another assignment for \( d(t,j) \) as the assignment function is a one to one function.
3.2 Analysis of the State Assignment Procedure:

We shall show that this procedure realizes $M$ (as defined in chapter 2) as a delay line network if the procedure is successfully completed.

Let $h(t) = m_1 m_2 \ldots m_k$ and let

$v = d(t,j)$. Then steps 7 and 11 ensure that $h(d(t,j)) = m_2 \ldots m_k f(m_1,j)$

$= g(h(t),j)$.

As these steps are carried out for all the entries in table 1, the above relation is valid for all states in $Q$ and all input symbols in $I$.

Step 21 ensures that $H$ is a one to one function and thus if the procedure is successfully completed, we have a delay line network

$N = (I, m, 1, f)$ with a function $h$, which realizes the given machine $M$.

Let us consider the number of distinct assignments that will be tried in this procedure. For a particular assignment of state 1 from the set $Q'$, the assignments of states 2, 3 \ldots $Q$ can take almost $2^n$ values (until $N_a = 2^n - 1$ in step 17) and hence the number of distinct assignments for states 2, 3 \ldots $Q$ for a specific assignment of state 1 is $2^n \times 2^n \ldots (|Q| - 1)$ times. But the state 1 can be assigned at most $2^{n_1}$ values (step 2) and hence the total number of distinct state assignments is given by

$2^{n_1} \times (2^n \times 2^n \ldots (|Q| - 1) \text{ times}) = 2^{n +(|Q| - 1)}$

If we had tried a procedure to assign the states in an arbitrary manner, then first state could have been assigned $2^{n_1}$ values, the second $2^{n_1} - 1$ and so on for the $|Q|$ th state $2^{n_1} - (|Q| + 1)$ values. Hence the total number of distinct assignments would be

$2^{n_1} \times (2^{n_1} - 1) \ldots (2^{n_1} - |Q| + 1)$
which is much larger than the number of distinct assignments which the given procedure tries.

3.3 Example:

We shall illustrate the procedure with an example. We are given the sequential machine \( M \) as

1. \( I = \{ 1, 2 \} \)
2. \( Q = \{ j \mid j = 1, \ldots, 9 \} \)
3. The state transition function \( d \) is given as

\[
\begin{array}{c|cc}
Q & 1 & 2 \\
\hline
1 & 1 & 2 \\
2 & 3 & 4 \\
3 & 5 & 5 \\
4 & 6 & 6 \\
5 & 7 & 8 \\
6 & 9 & 9 \\
7 & 1 & 1 \\
8 & 3 & 3 \\
9 & 7 & 7 \\
\end{array}
\]

Table 1

For the delay line network \( N \) we are given that \( I = \{ 1, 2 \}, n = 2, 1 = 2 \). Hence the set \( \{ 0, 1 \}^n = \{ (0,0), (0,1), (1,0), (1,1) \} \) is relabelled as \( K = \{ 0, 1, 2, 3 \} \).

We have to determine the function \( f = K \times I \rightarrow K \) and the state assignment function \( h: Q \rightarrow Q' \) where the set \( Q' \) is

\( Q' = \{ 00, 01, 02, 03, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 32, 33 \} \)
1. h(U) = b, u = 1, ... 9
   f(u,v) = b, u = 0, 1, ... 3; v = 1, 2

   N_a = 0, a = 2, 3, ... , 9
   k = 00, t = 1, j = 1

   h(1) = 00

   d(1,1) = 1 which is already in table 3.

   Hence f(0,1) = 0

2. t = 1, j = 2

   d(1,2) = 2 which is not yet assigned.

   f(0,2) is not in table

   f(0,2) = N_2 = 0

   Hence h(2) = 00 which is already in table 3.

   Hence N_2 = 0 + 1 = 1 hence

   f(0,2) = N_2 = 1

   Hence h(2) = 01

3. t = 2, j = 1

   d(2,1) = 3 which is not yet assigned.

   f(0,1) = 0

   Hence h(3) = 10

4. t = 2, j = 2

   q(2,2) = 4 which is not yet assigned

   f(0,2) = 1 from table 2.

   h(4) = 11

   Proceeding in the same manner, we arrive at the following assignment function
and the logic function $f$ as

$$
\begin{array}{c|cc}
R & 1 & 2 \\
\hline
0 & 0 & 1 \\
1 & 2 & 2 \\
2 & 0 & 0 \\
3 & - & - \\
\end{array}
$$

3.4 Bounds on the Length of the Delay Line:

We can always realize a given sequential machine using delay lines of length 1 \([1]\). As we are interested in realizing machines with delay lines of length greater than one, the lower bound on the length of delay line is 2.

We shall determine the upper bound on the length of delay line using theorem 2.2.6. From this theorem we know that if any state 'q' of the delay line network is in two cycles of length $u$ and $v$ then $1 \leq u + v - \text{gcd}(u,v)$. For each state of $M$, we shall determine the cycle lengths of
all the cycles that contain it by using the method given in the appendix.
From the cycle lengths, we can find
\[ L = \min \{ v(q) + u(q) - \gcd(v(q), u(q)) \} \]
over all \( q \in Q \) and all cycles so that any delay line network realizing this machine must have delay line length \( 1 < L \). Thus \( \text{lmax} = L - 1 \), and the delay line length \( 1 \) must be such that
\[ 2 \leq 1 \leq \text{lmax} \]

3.5 Bounds on the Number of Delay Lines:

For a specific length '1' of the delay line, we want to determine bounds on the number of delay lines that could be used to realize the given machine.

The lower bound is easily derived from consideration of the number of states. For a delay line network the number of possible states is \( 2^m \) which must be greater than or equal to the number of states of \( M \).

Hence
\[ 2^m \geq |Q| \quad \text{and} \quad m \geq [\log_2 |Q|/1] \]

The upper bound on the number of delay lines can be determined from cost considerations. We shall consider cost in the following manner.

The cost of realization can be split up into the cost of delay lines and the cost of logic to realize the function \( f \). Let us suppose that the cost of logic is proportional to the number of inputs.

Let \( n = \log_2 |Q| \) be the minimum number of delay lines of length 1 required to realize \( M \)

\[ c_1 = \text{cost of delay lines of length 1} \]
\[ c_1 = \text{cost of delay lines of length 1}. \]

Hence the cost to realize \( M \) with delay lines of length 1 is
\[ C = c_1 \cdot n + w (n + |Q|) \text{ where } w \text{ is the proportionality constant for the cost of logic.} \]
Figure 3.2
The cost to realize $M$ with $M$ delay lines of length 1 is

$$C_m = c_1 n + v (m + |I|).$$

Hence the maximum number of delay lines is given by $C_m \leq C_1$, that is

$$c_1 n_{\text{max}} + v (n_{\text{max}} + |I|) < c_1 n + v (n + |I|)$$

Thus the maximum number of delay elements $n_{\text{max}}$ can be determined.

This is one method of considering cost. Any other appropriate method would yield a suitable value of $n_{\text{max}}$.

3.6 The Complete Synthesis Algorithm:

The complete synthesis algorithm is given in figure 3.2. It uses the results developed in previous sections. We calculate bounds for length of delay line as in 3.3. We try lengths in the range $2 \leq l \leq l_{\text{max}}$ to realize the machine. For each length $l$, we calculate bounds on the number of delay lines. Then we systematically try the state assignment procedure for the values of $l, n$ in the above ranges. At any stage when the procedure is successful, we have found the desired delay line realization. If the procedure is not successful for any of the values of $l, n$ then we come to the conclusion that the given sequential machine cannot be realized as a delay line network.
4. SUMMARY AND CONCLUSION

In this thesis we have been concerned with the problem of realization of state sequential machines with delay line networks. We have considered the delay line networks as a special case of shift register networks. We have discussed the advantages of realizing sequential machines with delay line networks.

We have formally defined a delay line network by means of an algebraic model having four parameters. These are in the input set I, the number of delay lines, m, the length of delay line l, and the logic function f. We have also defined the machine realized by a delay line network and the concept of its behavior. After formally defining a state sequential machine, we have considered the concept of a delay line network realizing a given sequential machine.

With the help of the model, we have defined the concept of a state being in a cycle of same length c. We have shown that if a state is in a cycle of length c, then it is periodic with period c. We developed a result concerning the periodicaly of a doubly periodic finite sequence. We have shown that if a finite sequence is periodic with periods u, v and if the length of sequence is greater than or equal to $u + v - g$ where $g$ is the greatest common divisor of u, v then the sequence is periodic with period g. We have related these results by proving that if a state c is in two cycles of length u, v, then the length of the delay line must be less than $u + v - g$.

We have developed a state assignment procedure to assign states of a given sequential network to states of a specific delay line network.
We have shown how to derive bounds for the length and number of delay lines for a delay line network to realize a given machine. We have given a constructive synthesis algorithm to decide whether a given sequential machine could be realized as a delay line network. The algorithm can be easily implemented on a digital computer.

In future work, it would be interesting to develop theoretical bounds on the number of delay lines depending on the structure of the given machine. Another fruitful way of looking at this problem would be from the point of view of partition theory. A great deal of research has been done in the application of partition theory to shift register synthesis of sequential machine. A suitable modification pertinent to delay line synthesis might lead to some interesting insights.
APPENDIX

Determination of Cycle Lengths

Given a state sequential machine \( M = (I, Q, d) \), we will give a method to determine the lengths of cycles in which the states of \( Q \) are.

The method is based upon the construction of a response tree with various states as roots of the tree.

To find the cycle lengths of state \( q \), we construct a response tree with \( q \) as the root. The response tree is terminated at a particular node if that node has appeared before and if there is a path connecting the two appearances of the node. Thus the response tree will ultimately terminate as there are only a finite number of states of machine \( M \).

Now to find the cycle lengths of state \( q \), we examine the various heights of the tree. The state \( q \) is in a cycle of length \( C \), if node \( q \) is present at height \( C \) in the terminated response tree. Thus we can determine the various cycle lengths of state \( q \). This procedure would be repeated for all states to find cycle lengths of all states. We would illustrate the procedure by means of an example.

**Example:**

Let \( M = (I, Q, d) \) be given by \( I = \{1, 2\} \), \( Q = \{1, 2, 3\} \) and \( d \) is given by

<table>
<thead>
<tr>
<th>( d )</th>
<th>( I = 1 )</th>
<th>( I = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
We construct the terminated response tree for state 1, 2, 3 as follows:

Thus the cycle lengths of various states are given by

<table>
<thead>
<tr>
<th>State</th>
<th>Cycle Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>2</td>
<td>2, 3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>


