*AMSE JOURNALS-AMSE IIETA publication-2017-Series: Advances B; Vol. 60; N°2; pp 313-318*
*Submitted Apr. 04, 2017; Revised Sept. 25, 2017; Accepted Sept. 28, 2017*

# DNA Sequence Detector Using Finite State Machine Methodology

*Jishan Mehedi, **Nilkantha Rooj, ***Snehanjali Majumder, ****Anirban Mukherjee, *****Niladri Hore

*Department of Electronics & Communication Engineering, Jalpaiguri Government Engineering College, Jalpaiguri, West Bengal, India (j.mehedi@gmail.com)
**Department of Electronics & Communication Engineering, Jalpaiguri Government Engineering College, Jalpaiguri, West Bengal, India (nilkantharooj48@gmail.com)
***Department of Electronics & Communication Engineering, Jalpaiguri Government Engineering College, Jalpaiguri, West Bengal, India (snehanjali.maj@gmail.com)
****Department of Electronics & Communication Engineering, Jalpaiguri Government Engineering College, Jalpaiguri, West Bengal, India (im.anirban94@gmail.com)
*****Department of Electronics & Communication Engineering, Jalpaiguri Government Engineering College, Jalpaiguri, West Bengal, India (theniladrihore@gmail.com)

## Abstract

In this work a technique has been proposed where a DNA sequence is obtained and matched with another sequence using Finite State Machine (FSM) methodology. Each of the bases of a DNA molecule strand, viz. Adenine, Thymine, Guanine or Cytosine is assigned a 2 bit binary code. By performing this, a binary sequence (a string of binary information) corresponding to a DNA molecule is obtained. We aim to match this sequence (target string) with another predetermined DNA sequence (source string). This in particular can have an extra edge in terms of precision and reduce the errors while matching the source and the target sequences clinically. To further reduce the time of operation and optimize the performance, techniques to identify the number of 1's in the binary sequence by using 8085 microprocessor have been applied. The proposed technique has been implemented in circuit and the result obtained is accurate. The idea is new in this field and has a potential to expand in domains other than DNA molecules.

## Key words

DNA sequence, FSM, Microprocessor programming.

## 1. Introduction

Deoxyribonucleic Acid (DNA) is the hereditary material of humans and almost all other organisms. The information in DNA is stored as a code made up of four chemical bases: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T).DNA bases pair up with each other, A with T and C with G, to form units called base pairs. As a consequence it is enough to identify the code of a single strand to find the whole DNA. For example, if one strand consist a sequence ATGCCA, the other strand will have TACGGT. DNA sequencing is the process of determining the precise order of the bases within a DNA molecule. It includes any method or technology that is used to determine the order of the four bases (adenine, guanine, cytosine, and thymine) in a strand of DNA. DNA sequencing may be used to determine the sequence of individual genes, larger genetic regions, full chromosomes or entire genomes, along with positioning of genes [1,2].

DNA Fingerprinting or DNA profiling helps in identifying crime suspects, diagnosing genetic disorders, establishing paternity or other family relationships etc. In current date DNA profiling is used vastly to identify individuals by characteristics of their DNA, which is extensively done by DNA sequencing.

## 2. Logic Representation

Designing a DNA sequence detector using FSM process can help to detect any type of DNA Sequence. For example, suppose we take a DNA Sequence as ATGCGA. This sequence can be detected in a serial fashion [4].

First of all sequence is coded in binary pattern by assigning A as "00", C as "01", G as "10" and T as "11". We already know that A is complementary to T, and C is complementary to G. So the binary pattern of the observed sequence is written as "001110011000". This is primarily divided into two subgroups (each group contain 6 states). If we take First Group "001110"then we can see that there are six states, namely P, Q, R, S, T, U. As in FSM process, if P is in '0'state, then the state will be succeeded to next state, if next state Q is '0', next state will be succeeded, else state will be back to previous state. This process will continue till last stage U as '0'. Then the process is further transferred to the next sub group, and each subgroup result is stored in memory [1].

## 3. Design Procedure and Implementation

### 3.1 Design Procedure

Each Group as mentioned in section 2 is stored in a Serial register, as data are formed in a serial fashion with clock pulse being given in synchronous manner. In each clock pulse, data is given as input to FSM for design [5, 8 and 9]. Detection process is divided into four sub process:

1. Counting the number of '1' s in each sequence through microprocessor [5].

2. After counting the number of ‘1’, we have to consider the Binary pattern of the sequence and subdivide the sequence in each subgroup.

3. Then each subgroup has to undergo the FSM process which can be implemented in FPGA circuit technology.

4. After that the result will be obtained as the output of the circuit. Counting the number of '1' in the each subgroup is done by microprocessor programming. Here we delete the sequences which do not match with the given sequence in number of '1's. As a result less number of sequences has to be processed in FSM process.

After counting number of '1's in each sequence we find that the sequences to be matched are minimized within the FSM process. Each sequence is given to the FPGA circuit as input in serial fashion by the 8255 Interface and it takes less time to be processed in FSM. First, we have to assign each state and state will proceed to next state via each bit process.

## 3.2 FSM Implementation

Suppose we have a sequence "001110", this sequence is detected by the following procedure as shown in table1.

Transition from 'U' state to 'Q' state obtained as output '1'. Let X is an input and Z as a output and Generate Input-Output relation of FSM is obtained as shown in table 2. We know that for Six states , three Flip-flops are required and each state can be assign as three bit binary such as 'P' as "000" , 'Q' as "001", 'R' as "010", 'S' as "011", 'T' as "100" and 'U' as "101" and generate present state and next state output table as shown in table 3.

Tab.1. State Transition of FSM

| State | Has | Awaiting |
|---|---|---|
| P | - - - | ''001110'' |
| Q | ' 0 ' | ''01110'' |
| R | ' ' 0 0 ' ' | ''1110'' |
| S | ' ' 0 0 1 ' ' | ' ' 1 1 0 ' ' |
| T | ' ' 0 0 1 1 ' ' | ' ' 1 0 ' ' |

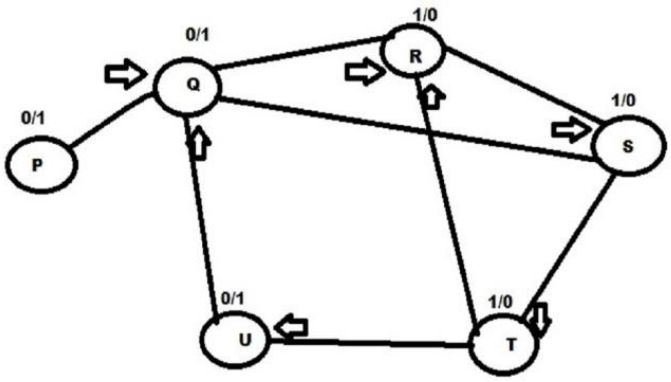


Fig.1. FSM

Tab.2. Present State Next State Table

| Present State | Next State | |
|---|---|---|
| | X = '0' | X = '1' |
| P | Q / 0 | P / 0 |
| Q | R / 0 | P / 0 |
| R | R / 0 | S / 0 |
| S | P / 0 | T / 0 |
| T | P / 0 | U / 0 |
| U | P / 0 | Q / 1 |

By this process we can detect the whole sequence and circuit of the FSM as shown in figure1 give the result in binary '1' form. It can be easily implemented by J-K or D flip-flop where inputs are taken as serial input and output is taken from the last flipflop output [3] and the hardware model has been developed [8]. We have implemented the FSM also using 8085 microprocessor. The program for 8085 microprocessor is shown in table 4.

Tab.3. Present State Next State with Output

| Present state | | Next state /output | |
|---|---|---|---|
| | Y2 Y1 Y0 | Y2 Y1 Y0 (X='0') | Y2 Y1 Y0 (X='1') |
| P | 0 0 0 | 0 0 1/0 | 0 0 0 / 0 |
| Q | 0 0 1 | 0 1 0/0 | 0 0 0 / 0 |
| R | 0 1 0 | 0 1 0/0 | 0 1 1 / 0 |
| S | 0 1 1 | 0 0 0/0 | 1 0 0 / 0 |
| T | 1 0 0 | 0 0 0/0 | 1 0 1 / 0 |
| U | 1 0 1 | 0 0 0/0 | 0 0 1 / 1 |

Tab.4. Program to Implement in 8085 Microprocessor

| Address | Level | Mnemonics |
|---|---|---|
| FF00 | START | MVI B,00H |
| FF02 | | MVI C,06H |
| FF04 | | LXI H, FFF0 |
| FF07 | | MOV A,M |
| FF08 | BACK | RAR |
| FF09 | | JNC SKIP |
| FF0C | | INR B |
| FF0D | SKIP | DCR C |
| FF0E | | JNZ BACK |
| FF0F | | INX H |
| FF10 | | HLT |

## Conclusion

An effective design for matching of binary strings in form of encoded DNA sequences has been shown in this paper. Finite Sate Machine has been used to generate the desired sequence to match the random sequences. Another advantage of using a Finite State Machine is that it leads to less errors and debugging is easier too. The concept is innovative and one of a kind in its field and is expected to give faster results too.

DNA sequencing paves way for further research in the field of cryptography and this project aims to simplify the process of sequencing and detecting any known sequence within the 55 million publicly available DNA sequence. Sequencing Technologies [7 and 9], it is used in Plant Biotechnology and Breeding, Crop Protection, Improvement of Farm Animal Breeding , Animal Systematic. Pooled DNA Sequencing is used in Disease Association Study too. The molecular data generated by DNA sequencing has played an important role in animal systematic over the last decades indicating the importance of this kind of information in evolutionary biology as a whole.

## References


1. K.M. Chao, Basic Concepts of DNA, Proteins, Genes and Genomes. Graduate Institute of Biomedical Electronics and Bioinformatics (Department of Computer Science and Information Engineering) Graduate Institute of Networking and Multimedia National Taiwan University, Taiwan.
2. R.E. Franklin, G. Dickerson, The DNA helix and how it is read,1983.

3. S. Salivahanan, S. Arivazhagan, Digital Circuit and Design.
4. S. Faro, T. Lecroq, An efficient matching algorithm for encoded DNA sequences and binary strings, Dipartimento di Matematica e Informatica, Universit`a di Catania, Italy. University of Rouen, LITIS EA 4108, 76821 Mont-Saint-Aignan Cedex, France.
5. Gaonkar, Microprocessor Architecture, Programming and Applications with the 8085.
6. B.J. Yoon, Signal Processing Methods for Genomic Sequence Analysis, California Institute of Technology.
7. A. Adnan, DNA Sequencing: Method, Benefits and Applications.
8. M. Perkowski, Digital Design Automation Finite State Machine, Department of Electrical Engineering Portland State University.
9. M. Lueders, S. Schauer, Finite State Machines for MSP430, Texas Instruments.