Realization of Sequential Circuit using Finite state Machine

Urvashi Kodwani
Department of Computer Science Engineering
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
kodwaniurvashi@gmail.com

Sonal Rajurkar
Department of Computer Science Engineering
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
sonurajurkar22@gmail.com

Prof. S. G. Mundada
Assistant Professor
Department of Computer Science Engineering
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
mundadasg@rknec.edu

***Abstract* - Automata Theory is a tool which is used in multidisciplinary computing and scientific research. It is the basis behind the traditional model of computation and is used for many purposes such as controller circuit design, sequential circuit design etc. A Sequence detector is a sequential state machine used to detect consecutive bits in a binary string..This technical paper examines various sequences and gives output as 1 if the sequence 1101 is detected. The types of the sequence examined are overlapping sequences. Algorithm designed for detecting the sequence 1101 uses flip-flops. The flip-flops help to detect the pattern in the given string.**

Keywords— VHDL , Sequence detector, Sequential circuits, Flip-flops, Mealy machine.

I. INTRODUCTION

The automata theory is the basis behind the traditional model of computation and is used for many purposes other than controller circuit design, including computer program compiler construction, proofs of algorithm complexity, and the specification and classification of computer programming languages [1]. Finite automata are the useful model for many software and hardware. They used in software for digital circuits, finding text pattern in web pages and verifying systems (Example Communication protocol) [1]. Many research papers [8, 9, 10, 11] and books [1, 6, 7] are published demonstrating applications of finite automata.

A sequence detector is a sequential state machine. A sequence detector is good example of sequential logic circuit also called FSM (Finite State Machine). Generally sequence detector detects consecutive bits in a string of binary bits. The objective of sequence detector is to introduce the use of sequential logic. In sequential logic the output depends on the current input values and also the previous inputs.

II. MEALY AND MOORE MACHINE

The two most popular state machines or finite automata with output are referred to as Moore machine and Mealy machines. These machines are named after two researchers who explored the structure of machines [2, 3]. The primary difference between these two state machines is that the output of a Moore machine depends only upon the state of the circuit whereas the output of a Mealy machine depends upon both the state and the inputs of the circuit. This has a practical effect in that the output signals of a Moore machine only change after output logic delays following a clock signal edge whereas the output signals of a Mealy machine may change at any time shortly after an input signal changes value [1].

In theoretical computer science, a Moore machine and a Mealy machine are considered to have similar efficiency because both can recognize “regular expressions”. There are three major differences between the Moore machine and Mealy machine.

- First, a Mealy machine requires fewer states to perform the same task because its output is a function of states and external inputs, and several possible output values can be specified in one state.
- Second, a Mealy machine can generate a faster response. Since a Mealy output is a function of input, it changes whenever the input meets the designated condition and a Moore machine reacts indirectly to input changes.
- The third difference involves the control of the width and timing of the output signal. In a Mealy machine, the width of an output signal varies with input and can be very narrow. A Mealy machine is susceptible to disturbances in the input signal and passes to the output. The output of a Moore machine is synchronized with the clock edge and its width is about the same as a clock period

III. TYPES OF SEQUENCE DETECTOR

There are basically two types of sequence detector depending on the type of sequence they identify, which are as follows:

- **Overlapping Sequence Detector:** In a sequence detector that allows overlap, the final bits of one sequence can be the start of another sequence. For example will be an 1101sequence detector. It raises an output of 1 when the last 4 binary bits received are 1101.
- **Non-Overlapping Sequence Detector:** The sequence detector with no overlap allowed resets itself to the start state when the sequence has been detected. After the initial sequence 1101 has been detected, the detector with no overlap resets and starts searching for the initial 1 of the next sequence.

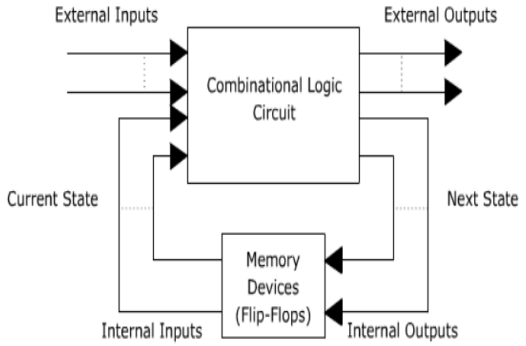


FIG 1: SEQUENCE DETECTOR

IV. VHDL

Most hardware designers use hardware description languages (HDLs) to describe designs at various levels of abstraction. A hardware description language is a high level programming language, with programming constructs such as assignments, conditions, iterations and extensions for timing specification, concurrency and data structure proper for modeling different aspects of hardware. The most popular hardware description languages are VHDL [4] and Verilog [5]. VHDL (VHSIC (*Very High Speed Integrated Circuits*) *Hardware Description Language*) [4] is an IEEE Standard since 1987 while Verilog was standardized in 1995.

VHDL is a programming language used to model a digital system by dataflow, behavioral and structural style of modeling. This language was first introduced in 1981 for the department of Defense (DoD) under the VHSIC program. It was originally developed under contract F33615-83-C-1003 from the United States Air Force awarded in 1983 to a team with Intermetrics, Inc. as language experts and prime contractor, with Texas instruments as chip design experts and IBM as computer system design experts. The language has undergone numerous revisions and has a variety of sub-standards associated with it that extend it in important ways.

V. FLIP-FLOPS

A flip-flop or latch is a circuit that has two stable states and can be used to store state information. A flip-flop is a bistable multivibrator. The circuit can be made to change state by signals applied to one or more Control inputs and will have one or two outputs, one for the normal value and one for the complement value of the stored bit. Memory elements in any sequential circuit are usually flip-flops.

The flip flops can be divided into 4 types:

- Set-Reset(SR) Flip-Flop
- Delay(D) Flip-Flop
- Toggle(T) Flip-Flop
- JK Flip-Flop

Flip-Flop Name	Flip-Flop Symbol	Characteristic Table	Characteristic Equation	Excitation Table
SR	S Q, Clk, R Q'	S R Q(next): 0 0 Q; 0 1 0; 1 0 1; 1 1 ?	Q(next) = S + R'Q SR = 0	Q Q(next) S R: 0 0 0 X; 0 1 1 0; 1 0 0 1; 1 1 X 0
JK	J Q, Clk, K Q'	J K Q(next): 0 0 Q; 0 1 0; 1 0 1; 1 1 Q'	Q(next) = JQ' + K'Q	Q Q(next) J K: 0 0 0 X; 0 1 1 X; 1 0 X 1; 1 1 X 0
D	D Q, Clk, Q'	D Q(next): 0 0; 1 1	Q(next) = D	Q Q(next) D: 0 0 0; 0 1 1; 1 0 0; 1 1 1
T	T Q, Clk, Q'	T Q(next): 0 Q; 1 Q'	Q(next) = TQ' + T'Q	Q Q(next) T: 0 0 0; 0 1 1; 1 0 1; 1 1 0

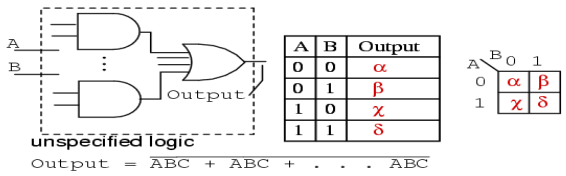
Table 1: Characteristics of Flip-Flops

Uncertainty in the state of SR flip flop when S=R=1 can be eliminated by converting it into JK flip flop. The data inputs are J and K which are ANDED with Q and Q respectively to obtain S and R .The characteristic equation of the JK flip-flop is: S=JQ & R=KQ

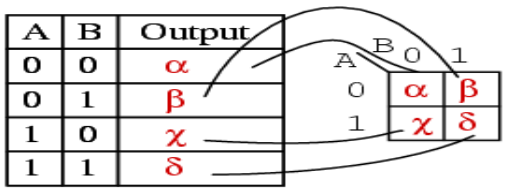
VI. KARNAUGH MAP (K-MAP)

Maurice Karnaugh, a telecommunications engineer, developed the Karnaugh map at Bell Labs in 1953 while designing digital logic based telephone switching circuits

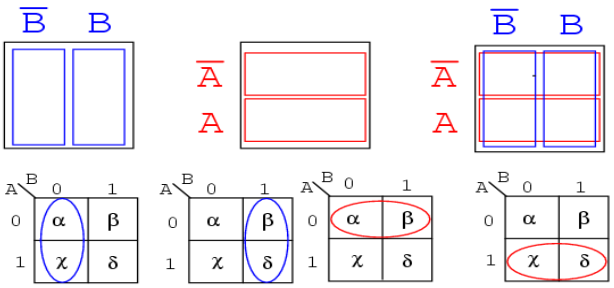
A Karnaugh map (K-map) is a pictorial method used to minimize Boolean expressions without having to use Boolean algebra theorems and equation manipulations. A K-map can be thought of as a special version of a truth table.



Four individual items as shown above, are just different ways of representing the same thing: an arbitrary 2-input digital logic function. First is logic gates, a truth table, a Karnaugh map, and a Boolean equation. The point is that any of these are equivalent. Two inputs **A** and **B** can take on values of either **0** or **1**, high or low, open or closed, True or False, as the case may be. There are $2^2 = 4$ combinations of inputs producing an output. This is applicable to all five examples.
These four outputs may be observed on a lamp in the relay ladder logic, on a logic probe on the gate diagram. These outputs may be recorded in the truth table, or in the Karnaugh map. Look at the Karnaugh map as being a rearranged truth table. The Output of the Boolean equation may be computed by the laws of Boolean algebra and transfered to the truth table or Karnaugh map.



The outputs of a truth table correspond on a one-to-one basis to Karnaugh map entries. Starting at the top of the truth table, the A=0, B=0 inputs produce an output α. Note that this same output α is found in the Karnaugh map at the A=0, B=0 cell address, upper left corner of K-map where the A=0 row and B=0 column intersect. The other truth table outputs β, χ, δ from inputs AB=01, 10, 11 are found at corresponding K-map locations.
Below, we show the adjacent 2-cell regions in the 2-variable K-map with the aid of previous rectangular Venn diagram like Boolean regions.



Cells α and χ are adjacent in the K-map as ellipses in the left most K-map below. Referring to the previous truth table, this is not the case. There is another truth table entry (β) between them. Which brings us to the whole point of the organizing the K-map into a square array, cells with any Boolean variables in common need to be close to one another so as to present a pattern that jumps out at us. For cells α and χ they have the Boolean variable **B'** in common. We know this because **B=0** (same as **B'**) for the column above cells α and χ. Compare this to the square Venn diagram above the K-map.
A similar line of reasoning shows that β and δ have Boolean **B** (B=1) in common. Then, α and β have Boolean **A'** (A=0) in common. Finally, χ and δ have Boolean **A** (A=1) in common.

VII. Sequence Dectector Design

Sequence detector for sequence **1101** with the help of Mealy machine is as given below

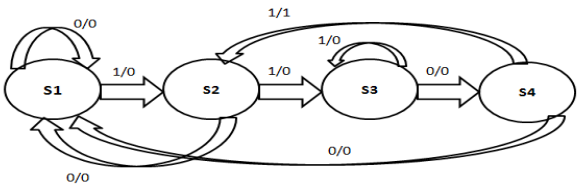


Fig 2: Sequence Detector using Mealy Machine

State table for the sequence detector is as follows:

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	0
S1	S0	S2	0	0
S2	S3	S2	0	0
S3	S0	S1	0	1

Table 2: State table

The binary representation for above state table is as follows:

Present State	Next State				Output	
	X=0		X=1		X=0	X=1
	A1	B1	A2	B2		
0 0	0	0	0	1	0	0
0 1	0	0	1	0	0	0
1 0	1	1	1	0	0	0
1 1	0	0	0	1	0	1

Table 3: Binary representation for above state table

Now to implement sequence detector we require flip-flops. Let the two required flip-flops be JK flip-flops. The above table along with JK flip-flop can be written as:

Present State		Input (x)	Next State		Input to Flip-flop				Output
A	B		A	B	Ja	Ka	Jb	Kb	
0	0	0	0	0	0	X	0	x	0
0	0	1	0	1	0	X	1	X	0
0	1	0	0	0	0	X	X	1	0
0	1	1	1	0	1	X	X	1	0
1	0	0	1	1	X	0	1	X	0
1	0	1	1	0	X	0	0	X	0
1	1	0	0	0	X	1	X	1	0
1	1	1	0	1	X	1	X	0	1

Table 4: State table with flip-flops

The equations for flip flops are determined using K-Maps. K-Map for the above flip flop:

X \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$	0	2	6 X	4 X
X	1	3 1	7 X	5 X

$J_A = BX$

X \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$	0 X	2 X	6 1	4
X	1 X	3 X	7 1	5

$K_A = B$

X \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$	0	2	6 X	4 1
X	1 1	3 X	7	5

$J_B = A\bar{X} + \bar{A}X$

X \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$	0 X	2 1	6 1	4 X
X	1 X	3 1	7	5 X

$K_B = \bar{A} + \bar{X}$

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{X}$	0	2	6	4
X	1	3	7 1	5

$Y = ABX$

On the basis of equations obtained through K-Map, we design a circuit diagram as given below. It contains two JK flip flops. The demonstration of equations are represented using different gates used like AND, OR, XOR, NOT gates. We have also used clock (clk). When clk is 1 then flip flop is in active state and at other times it is inactive. X ,Y are used to denote input and output respectively. Output Y is obtained with the help of AND gate by using equations.

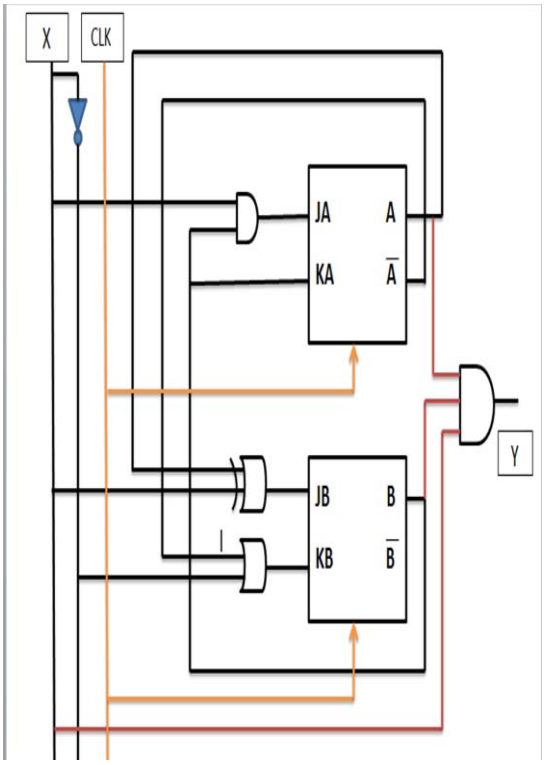


Fig3: Circuit Diagram.

VIII. IMPLEMENTATION

```
entity seq is
port (clk,a,b,x: in bit; ja,ka,jb,kb,y: out bit);
end seq;
architecture dataflow of seq  is
begin
ja<=b and x;
ka<=b;
jb<=a xor x;
kb<=(not a) or (not x);
y<=a and b and x;
end dataflow;
```

Table 5: VHDL design for sequence detector circuit

IX. TESTING

Sample Input I:

Consider the sequence as 0111101 as shown in figure 4

Refer state diagram:

Input: 0 1 1 1 1 0 1

Output: 0 0 0 0 0 0 1

The above input string contains instance of 1101. Hence, the output string contains '1' at position 7. It means that the sequence 1101 is detected in the given sample input.

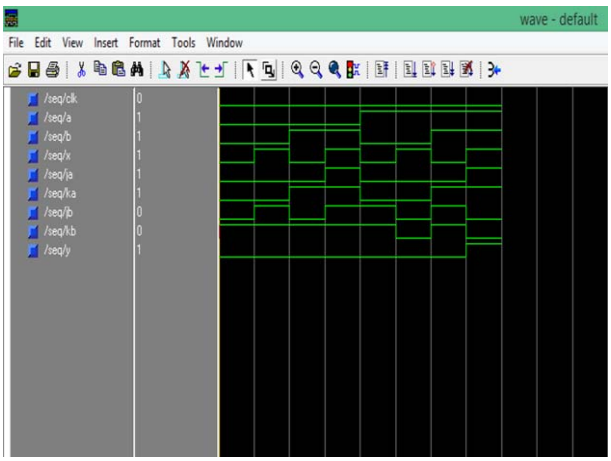


Fig 4.Simulation Result for Sample Input I

Sample Input II:

Consider the sequence as 0001 as shown in figure 5

Refer state diagram:

Input: 0 0 0 1

Output: 0 0 0 0

The above input string does not contain instance of 1101. Hence, the output string does not contain '1'. It means that the sequence 1101 is not present in the given sample input.

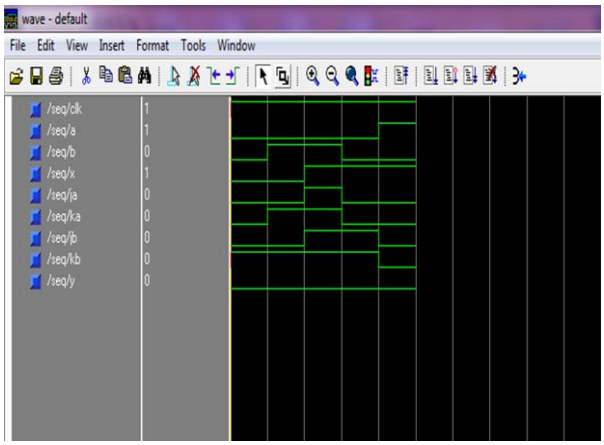


Fig 5.Simulation Result for Sample Input II

X. CONCLUSION

This paper shows application of finite state machines that is finite automata with output for realization of sequential circuits using VHDL. Sequence detector has been implemented by applying the concept of theory of automata. Sequence detector has wide variety of applications such as design of Ring counter, Serial Adder, Serial Data, Schmitt trigger.

References

[1] Hopcroft, J. E., Ullman, J. D."Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, 1979.

[2] Mealy, G. H., "A method for synthesizing sequential circuits," Bell System Tech. J., Vol. 34, No. 5, pp. 1045–1079, 1955

[3] Moore, E. F., "Gedanken experiments on sequential machines," Automata Studies. Princeton, NJ: Princeton University Press, pp. 129-53,1956

[4] IEEE Standard 1076-1993, IEEE Standard Description Language based on the VHDL Hardware Description Language, 1993.

[5] IEEE Standard 1364-2001, IEEE Standard Description Language based on the Verilog Hardware Description Language, 2001.

[6] D. Perrin, "Finite Automata", Handbook of Theoretical Computer Science. Elsevier Science Published 1990.

[7] H. R. Lewis, C. H. Papadimitriou.,"Elements of the Theory of Computation", Prentice Hall 1981.

[8] K. Culik II and J. Kari,." Image Compression Using Weighted Finite Automata, in Fractal Image Compression. In Theory a Techniques", Springer-Verlag, pp 243-258, 1994.

[9] Mindek, M., "Finite State Automata and Image Recognition" DATESO 2004, pp 132-143 (2004), ISBN: 80-248-0457-3

[10] G. Navarro, R. Baeza-Yates, "Improving an Algorithm for Approximate String Matching.",Algorithmica, 30(4) 2001

[11] S.V.Ramasubramanian and Kamala Krithivasan,"Finite Automata and Digital Images", IJPRAI, Vol. 14, No. 4, pp. 501-524, 2000.

[12] John C.Martin , "Introduction to Languages and Theory of Computation",Third edition, Tata McGraw Hill,2009