DAYANANDASAGAR COLLEGE OF ARTS SCIENCE AND COMMERCE

Shavige Malleshwara Hills,1stStage,Kumaraswamy Layout,Bengaluru, Karnataka.

Department of Computer Application-MCA(BU)

A project report on

The Application of TOC on Digital Circuits

Submitted To:
Ms.Akshatha
Assistant Professor
DSCASC

Submitted By:
Prathusha(P03CJ24S126073)
Yashaswini(P03CJ24S126120)
Usha B A(P03CJ24S126112)
Sneha N S(PO3CJ24S126104)

CONTENTS

1.Problem Statement

The goal of this project is to explore the application of Theory of Computation (TOC) principles in digital circuit design, specifically using Finite State Machines (FSMs). The task involves designing and simulating basic digital circuits that model the behavior of three digital components:

- Shift Registers
- Synchronous Counters
- Multiplexers

Each component should exhibit the appropriate FSM transitions based on specific inputs.

2. INTRODUCTION

The Theory of Computation (TOC) is a fundamental area of computer science that focuses on the mathematical and logical foundations of computation. It is concerned with the study of abstract computational models, formal languages, and the inherent capabilities and limitations of computers in solving problems. The central objective of TOC is to determine what can be computed, how it can be computed, and how efficiently it can be done.

Applications of TOC in Digital Circuits:

1. **Finite State Machines (FSMs):**

   - FSMs are widely used in digital circuit design (e.g., control units in CPUs, protocol handle.

2. **Regular Expressions & Pattern Recognition**

- Used in digital signal processing (DSP) and circuits that recognize specific bit patterns.

3. **Lexical Analysis in Compilers (Hardware Realization):**

- FPGA implementations of compilers or interpreters use these principles.

4. **Sequential Logic Design:**

- Mapping input-output behavior in sequential circuits (e.g., counters, registers).

5. **Model Checking & Verification:**

- Verifying circuit behavior using model checkers, which rely on TOC foundations.

To design and simulate basic circuits using **Finite State Machines (FSMs)** for **Shift Registers**, **Synchronous Counters**, and **Multiplexers**, we'll follow these steps:

1. Shift Register (3-bit Serial-In Serial-Out - SISO) FSM

Modeling:

- **States**: Each state represents the 3-bit content of the register.
- **Inputs**: `In` (1-bit serial input), `Clk` (trigger).
- **Transitions**: On each clock pulse, the content shifts right, and `In` is inserted into the leftmost bit.

Example States (partial for 3-bit):

```
Vb net
CopyEdit
State: 000 → In=1 → Next: 100
State: 100 → In=0 → Next: 010
State: 010 → In=1 → Next: 101
```

Simulation Logic (pseudocode/VHDL-style):

```
vhdl CopyEdit
process(Clk)
begin
  if rising_edge(Clk) then
    Register(2) <= In;
    Register(1) <= Register(2);
Register(0) <= Register(1);   end
if;
end process;
```

2. 3-bit Synchronous Counter (Up Counter) FSM

Modeling:

- **States**: `000` to `111` (0 to 7)
- **Inputs**: `Enable`, `Clk`
- **Transitions**: On every clock edge (when `Enable=1`), the counter increases by 1.

State Transitions:

```
CopyEdit
000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000
...
```

Simulation Logic:

```
vhdl CopyEdit
process(Clk)
begin
```

```
  if rising_edge(Clk) then
if Enable = '1' then
Count <= Count + 1;
end if;   end if;
end process;
```

3. 4-to-1 Multiplexer (MUX) FSM

Modeling:

- **States**: MUX does not change states per se, but you can model selection behavior using FSM.
- **Inputs**: `Sel` (2-bit), `I0`, `I1`, `I2`, `I3`
- **Outputs**: `Y = Ix` based on `Sel` **FSM Behavior**:

```
ini CopyEdit
Sel = "00" → Output = I0
Sel = "01" → Output = I1
Sel = "10" → Output = I2   Sel
= "11" → Output = I3
```

Simulation Logic:

```
vhdl CopyEdit
process(Sel, I0, I1, I2, I3)
begin   case Sel is     when
"00" => Y <= I0;     when
"01" => Y <= I1;     when
"10" => Y <= I2;     when
"11" => Y <= I3;   end case;
end process;
```

Simulation Approach:

You can simulate these using:

- **VHDL/Verilog** in **ModelSim**, **Vivado**, or **Quartus**
- **FSM simulation tools** (like Logisim or digital logic simulators)
- **Python FSM frameworks** (for logic-level simulation, e.g., `transitions` library)

Summary Table

Component	FSM States	Input(s)	Output	FSM Transition Trigger
Shift Register	8 (3-bit register states)	In, Clk	Serial Out	Clock Edge
Synchronous Counter	8 (000 to 111)	Enable, Clk	Count (3-bit)	Clock Edge if Enabled
Multiplexer	4 (based on Sel)	I0–I3, Sel	Y	Change in `Sel`

Conclusion:

This report demonstrates the application of TOC through FSMs in modeling and simulating fundamental digital circuits. Shift registers, counters, and multiplexers were effectively described using state-based models, reflecting the power of theoretical computational models in real-world digital design.