**Specification:**

We need to design a three-speed fan circuit to show speed “0 1 2 3” on the 7 segment display. For this portion we use 2 bit binary input **A**, **B** and the outputs are **a**, **b**, **c**, **d**, **e**, **f**, **g** which will show on 7 segment display. And in the Sequential part we need a counter which count binary 0 to 3 for direction 1 and remain same for direction 0. In present state we have two bits, and for next state we have also 2 bits and that is, and we design it using **JK Flip-Flop**.

**Using Basic Gate’s**

**List of Equipment’s:**

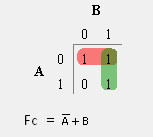
1. IC 7432 - 2 input OR gate
2. IC 7408 - 2 input AND gate
3. IC 7473 - Dual JK Flip-Flop
4. Breadboard
5. 7 Segment Display (Common Cathode)
6. 1.5 V AA Battery (4)
7. Battery Case
8. Switch
9. LED

**Combinational Part**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | **Output** | | | | | | | | | | |
| **A** | **B** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **Speed-1** | **Speed-2** | **Speed-3** |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Table-1: Truth Table

**K-MAP:**



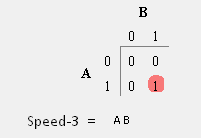
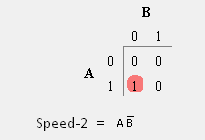
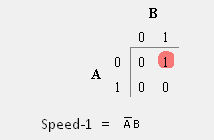
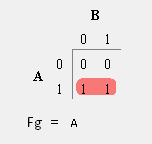
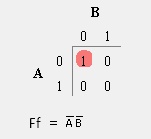
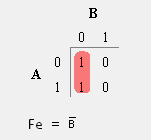
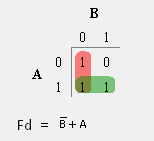
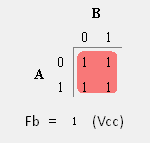
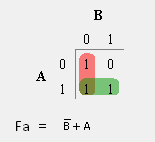


Fig.-1: Combinational K-Map

**Combinational Circuit Diagram:**

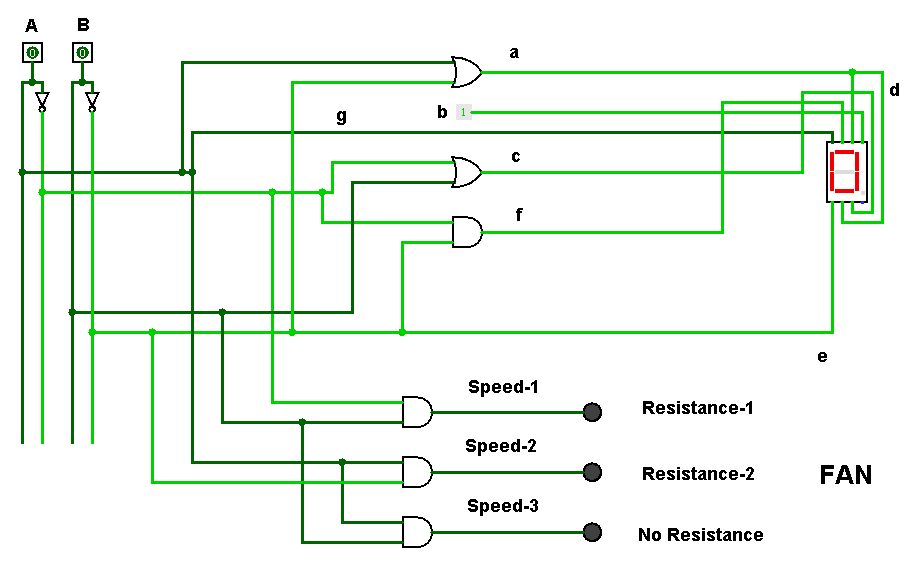


Fig.-2: Combinational Output for ‘00’

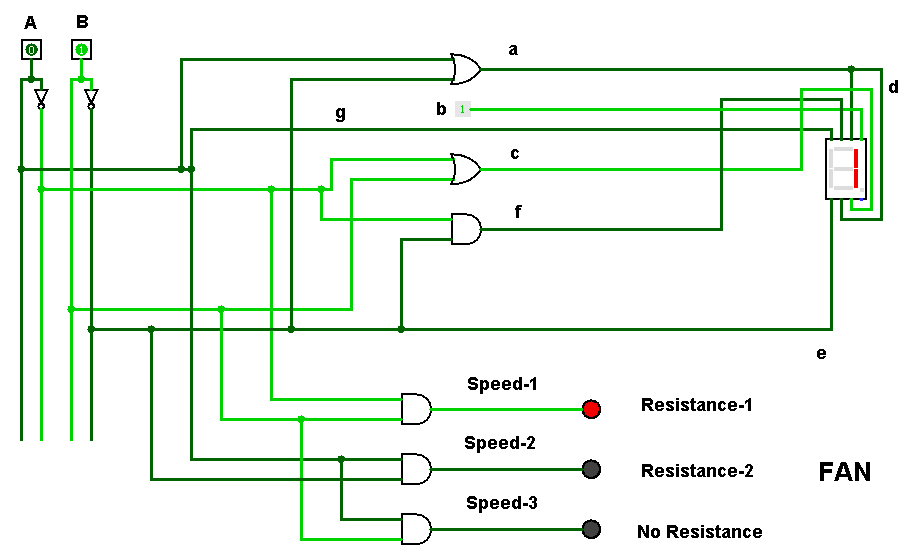


Fig.-3: Combinational Output for ‘01’

**Sequential Part**

**State Diagram:**

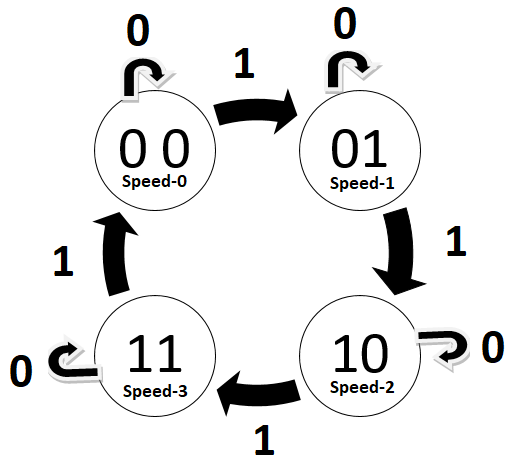


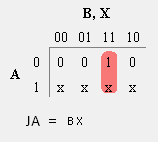
Fig.-4: State Diagram

**State Table:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Index** | **Present State** | | **Direction** | **Next State** | | **Flip-Flop Input** | | | |
|  |  |  | **X** |  |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | x |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | x | 1 | x |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 | x | x | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 1 | x | x | 1 |
| 4 | 1 | 0 | 0 | 1 | 0 | x | 0 | 0 | x |
| 5 | 1 | 0 | 1 | 1 | 1 | x | 0 | 1 | x |
| 6 | 1 | 1 | 0 | 1 | 1 | x | 0 | 0 | x |
| 7 | 1 | 1 | 1 | 0 | 0 | x | 1 | x | 1 |

Table-2: State Table

**Equation from K-MAP:**



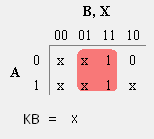
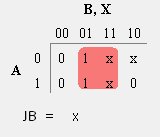
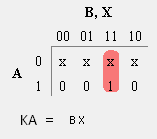


Fig.-5: Sequential K-Map

**Sequential Circuit Diagram:**

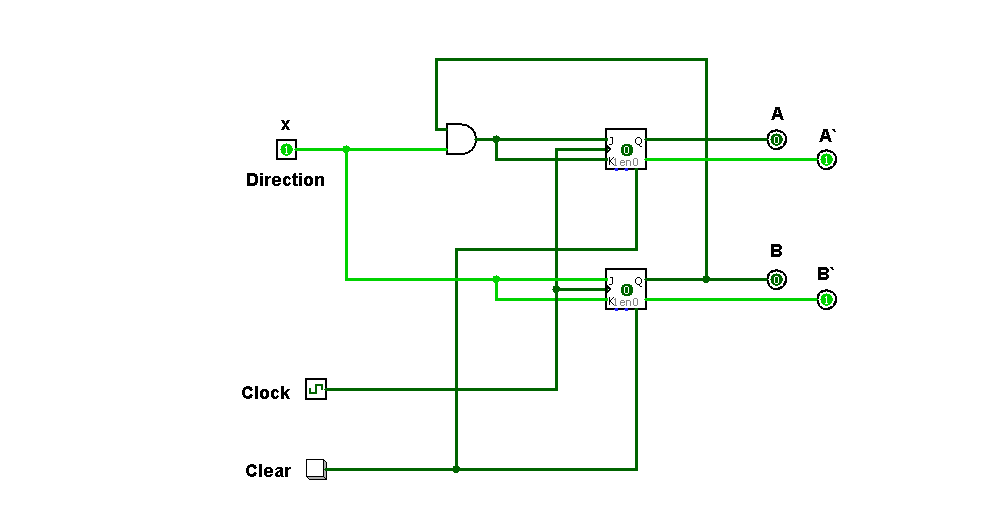


Fig.-6: Sequential Circuit

**Combined Circuit Diagram:**

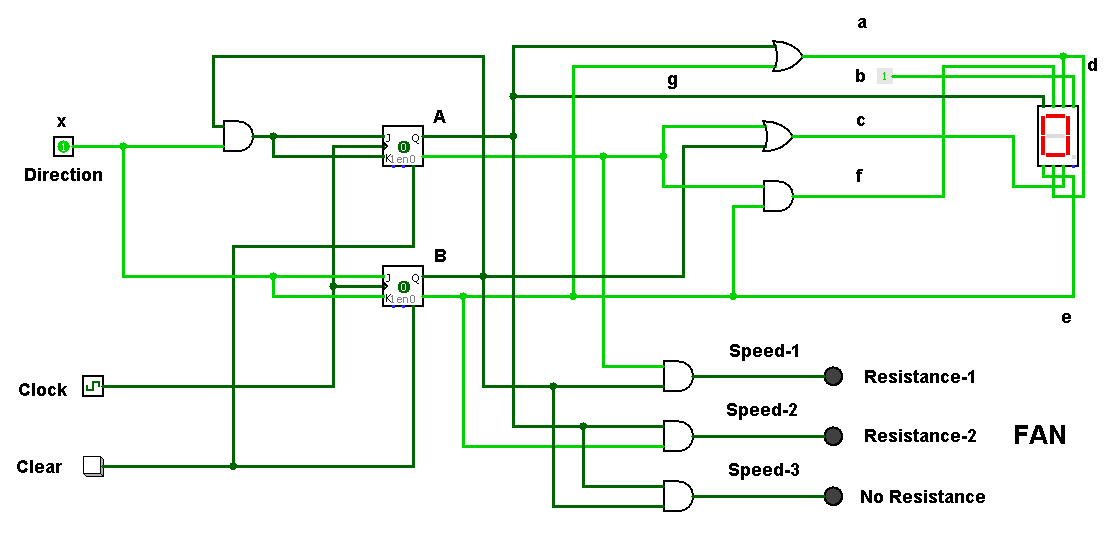


Fig.-7: Output For ‘00’ is 0

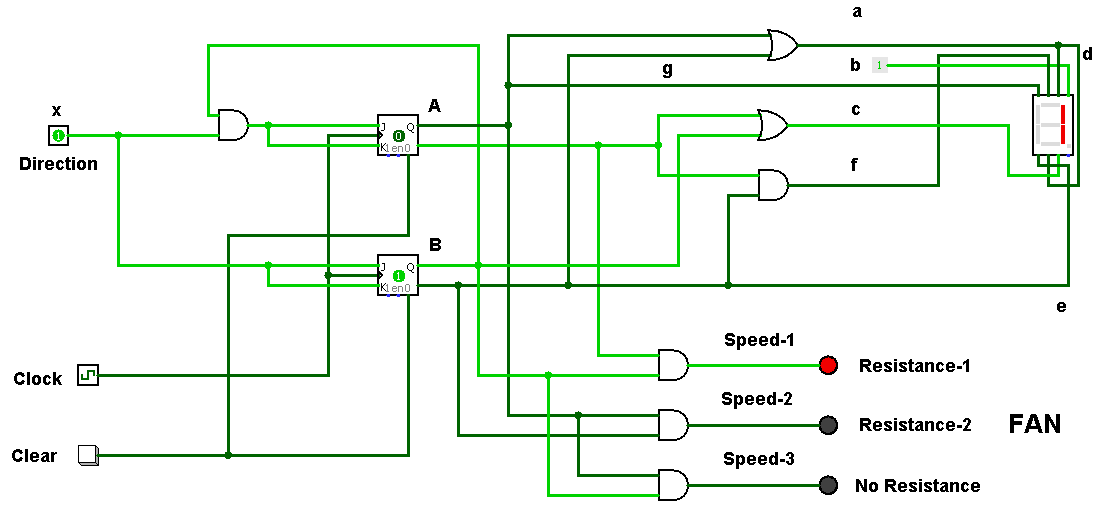


Fig.-8: Output For ‘01’ is 1

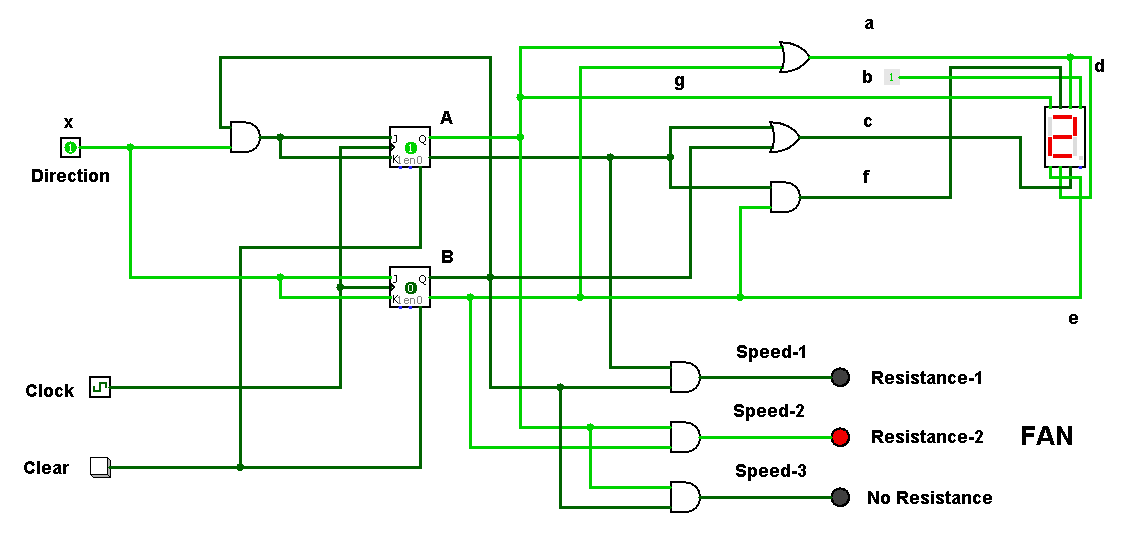


Fig.-9: Output For ‘10’ is 2

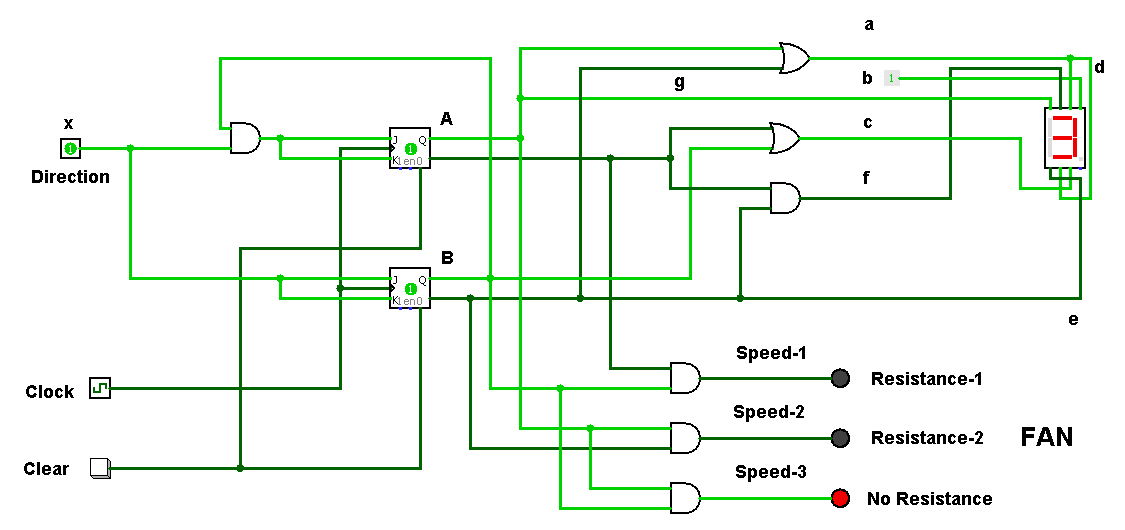


Fig.-10: Output For ‘11’ is 3

**Alternate Methods:**

**Using Decoder:**

[Using Table-1, 2 & K-Map’s]

**a** = A + B` = A (B+B`) + B` (A+A`) = AB + AB` + AB` + A`B`

= A`B` + AB` + AB

**b** = 1 (**Vcc**)

**c** = A` (B+B`) + B (A+A`) = A`B + A`B` + AB + A`B

= A`B` + A`B + AB

**d** = A`B` + AB` + AB (Same as **a**)

**e** = B` = B` (A+A`) = A`B` + AB`

**f** = A`B`

**g** = A = A (B+B`) = AB`+ AB

**Speed-1** = A`B

**Speed-2** = AB`

**Speed-3** = AB

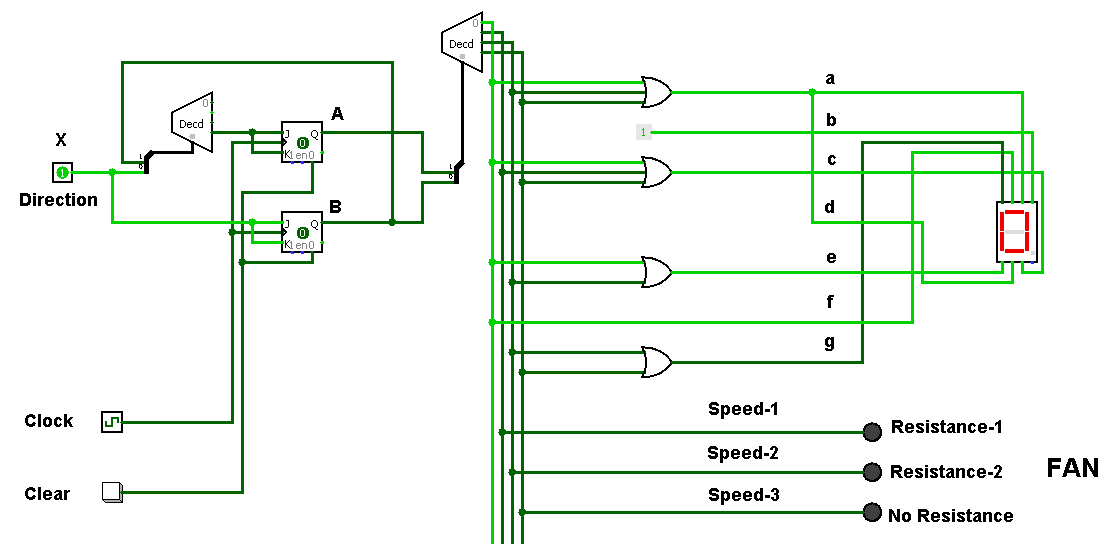
****

Fig.-11: Full Implementation Using Decoder

**Using Universal Gate:**

[Using Table-1, 2 & K-Map’s]

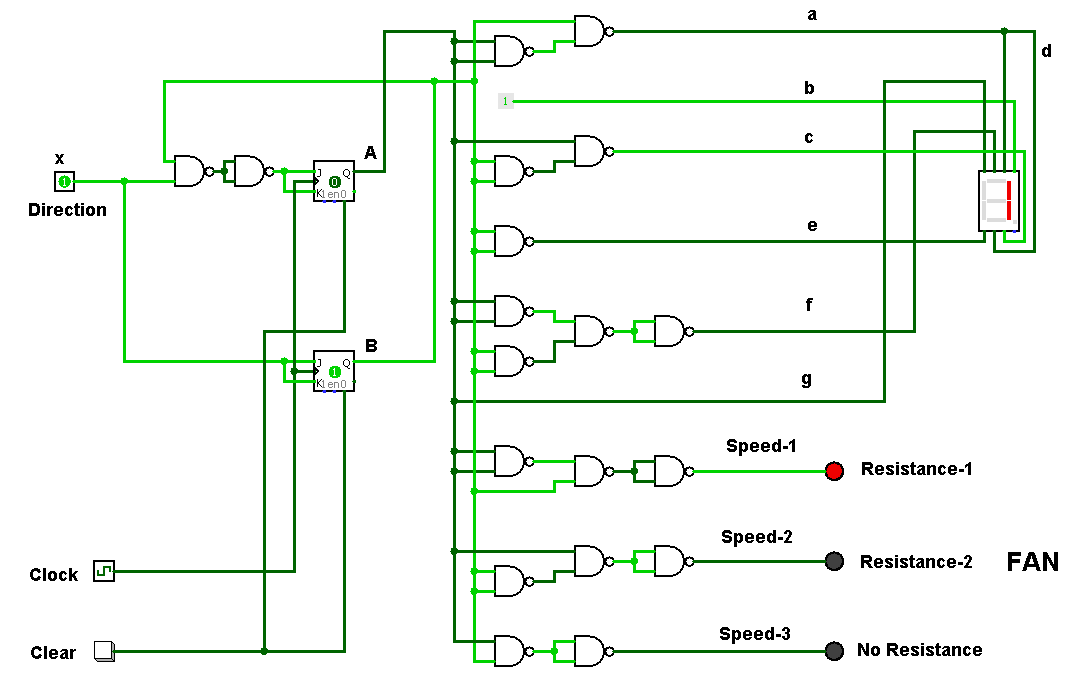
****

Fig.-12: Full Implementation Using NAND

**Using Multiplexer:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Selector | | Input( 0 = **GND** , 1 = **Vcc** ) | | | | | | | | | |
| **A** | **B** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **Speed-1** | **Speed-2** | **Speed-3** |
| 0 | 0 | **I0 =**1 | **I0 =**1 | **I0 =**1 | **I0 =**1 | **I0 =**1 | **I0 =**1 | **I0 =**0 | **I0 =**0 | **I0 =**0 | **I0 =**0 |
| 0 | 1 | **I1 =**0 | **I1 =**1 | **I1 =**1 | **I1 =**0 | **I1 =**0 | **I1 =**0 | **I1 =**0 | **I1 =**1 | **I1 =**0 | **I1 =**0 |
| 1 | 0 | **I2 =**1 | **I2 =**1 | **I2 =**0 | **I2 =**1 | **I2 =**1 | **I2 =**0 | **I2 =**1 | **I2 =**0 | **I2 =**1 | **I2 =**0 |
| 1 | 1 | **I3 =**1 | **I3 =**1 | **I3 =**1 | **I3 =**1 | **I3 =**0 | **I3 =**0 | **I3 =**1 | **I3 =**0 | **I3 =**0 | **I3 =**1 |

Table-3: Multiplexer Selector Table

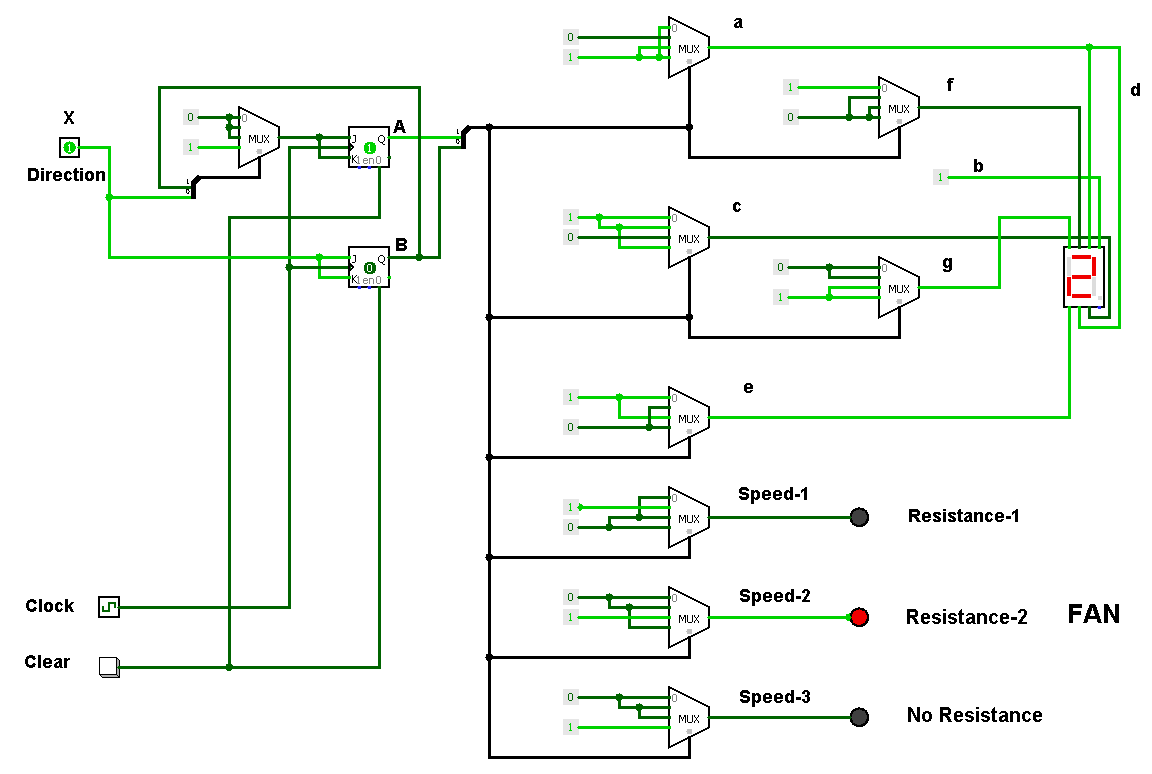


Fig.-13: Full Implementation Using Multiplexer

**Why we choose Basic Gates to implement the circuit:**

**COST:**

|  |  |  |
| --- | --- | --- |
| **Using Method** | **IC** | **IC COST \*** |
| **Universal (NAND)** | **( 2 input NAND ) IC 7400 - 5**  **( Dual JK FF ) IC 7473 - 1** | **5 \* 23 = 115**  **1 \* 35 = 35**  **Total = 150** |
| **Decoder** | **( 2 to 4 Decoder ) IC 74139 - 2**  **( 2 input OR ) IC 7432 - 1**  **( 3 input OR ) IC 4075 - 1**  **( Dual JK FF ) IC 7473 - 1** | **2 \* 30 = 60**  **1 \* 26 = 26**  **1 \* 23 = 23**  **1 \* 35 = 35**  **Total = 144** |
| **Multiplexer** | **( 4 to 1 Multiplexer ) IC 74153 - 9**  **( Dual JK FF ) IC 7473 - 1** | **9 \* 34 = 306**  **1 \* 35 = 35**  **Total = 341** |
| **Basic Gates** | **( 2 input OR ) IC 7432 - 1**  **( 2 input AND ) IC 7408 – 2**  **( Dual JK FF ) IC 7473 - 1** | **1 \* 26 = 26**  **2 \* 24 = 48**  **1 \* 35 = 35**  **Total=109** |

Table-4: IC Cost Table

**Complexity:**

**If we implement using NAND / Decoder / Multiplexer method then it is more complex than Basic Gates. Because in NAND / Decoder / Multiplexer take more IC than Basic Gates and for connecting an IC to another IC with wire it turn into a complex path. That’s why we choose Basic Gates to implement the circuit.**

**Conclusion:** At the end of our work the display was working as we planned. In this project we touched the exciting part of digital design every issue was exciting challenge. The most important part of this project that we covered what we have studied experimentally and practice a new experience; also we had a good review for the final Exam.

\* According to RobodocBD.com