1. The assignment Aim

The assignment aims to design an electronic voting machine to create an efficient, and reliable system that can be used to conduct elections electronically.

The steps of creating the electronic voting system:

- **Choosing which type of flip-flops to use to construct the counters.**
- **Defining the state table of the behavior of the counter I want.**
  Which is an asynchronous BCD up counter that counts from 0 to 9 then reset back to 0 and keep counting and so on.
- **Analyzing how to connect 3 counters with each other to create a whole counter of 3 digits so it counts to 999.**
- **Enhance the design by adding a button to vote and another button to reset.**
- **Analyzing which other components needed for the design which are 7-segment decoders and 7-segment displays to display the counting.**
- **Simulating the design on proteus to make sure it works fine.**
- **Implementing the design using hardware description language (Verilog hdl).**
- **Simulating the code to make sure it works fine.**

2. The Problem Solution

2. 1. Inputs and Outputs

General design:
Inputs:
CLK input which comes from the switch
RESET input which comes from a switch

Outputs:
7-Segment Decoder for ones (a,b,c,d,e,f,g)
7-Segment Decoder for tens (a2,b2,c2,d2,e2,f2,g2)
7-Segment Decoder for hundreds (a2,b2,c2,d2,e2,f2,g2)

Ones Counter:
A Flip Flop:
Inputs:
Da and CLK

Outputs:
A and A’

B Flip Flop:
Inputs:
Db and CLK

Outputs:
B and B’

C Flip Flop:
Inputs:
Dc and CLK

Outputs:
C and C’

D Flip Flop:
Inputs:
Dd and CLK

Outputs:
D and D’

Tens Counter:

A2 Flip Flop:
Inputs:
Da2 and CLK2

Outputs:
A2 and A2’

B2 Flip Flop:
Inputs:

Db2 and CLK2

Outputs:
B2 and B2’

C2 Flip Flop:
Inputs:
Dc2 and CLK2

Outputs:
C2 and C2’

D2 Flip Flop:
Inputs:
Dd2 and CLK2

Outputs:
D2 and D2’

Hundreds Counter:

A3 Flip Flop:
Inputs:
Da3 and CLK3

Outputs:
A3 and A3’

B3 Flip Flop:
Inputs:
Db3 and CLK3

Outputs:
B3 and B3’

C3 Flip Flop:
Inputs:
Dc3 and CLK3

Outputs:
C3 and C3’

D3 Flip Flop:
Inputs:
Dd3 and CLK3

Outputs:
D3 and D3’

CLK2 Circuit:
Flip Flop:
Inputs:
Dff=A.D

clk=A.D

Outputs:
CLK2=Dff.(A+B+C+D)

CLK3 Circuit:
Flip Flop:
Inputs:
Dff2=A2.D2
clk2=A2.D2

Outputs:
CLK3=Dff2.(A2+B2+C2+D2)

First 7-Segment Decoder:
Inputs:
$A_{decoder}=D_{counter}$
$B_{decoder}=C_{decoder}$
$C_{decoder}=B_{decoder}$
$D_{decoder}=A_{decoder}$

Outputs:
a,b,c,d,e,f,g

Second 7-Segment Decoder:
Inputs:
$A2_{decoder}=D2_{counter}$
$B2_{decoder}=C2_{decoder}$
$C2_{decoder}=B2_{decoder}$
$D2_{decoder}=A2_{decoder}$

Outputs:
a2,b2,c2,d2,e2,f2,g2

Third 7-Segment Decoder:
Inputs:
$A3_{decoder}=D3_{counter}$
$B3_{decoder}=C3_{decoder}$
$C3_{decoder}=B3_{decoder}$
$D3_{decoder}=A3_{decoder}$

Outputs:
a3,b3,c3,d3,e3,f3,g3

2. 2. The Truth Tables

2. 2.1 BCD Counter State Table:

	Present State				Next State				Flip-Flops Inputs			
	A	**B**	**C**	**D**	**A+**	**B+**	**C+**	**D+**	**D_A**	**D_B**	**D_C**	**D_D**
0	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	1	0
2	0	0	1	0	0	0	1	1	0	0	1	1
3	0	0	1	1	0	1	0	0	0	1	0	0
4	0	1	0	0	0	1	0	1	0	1	0	1
5	0	1	0	1	0	1	1	0	0	1	1	0
6	0	1	1	0	0	1	1	1	0	1	1	1
7	0	1	1	1	1	0	0	0	1	0	0	0
8	1	0	0	0	1	0	0	1	1	0	0	1
9	1	0	0	1	0	0	0	0	0	0	0	0
10	1	0	1	0	d	d	d	d	d	d	d	d
11	1	0	1	1	d	d	d	d	d	d	d	d
12	1	1	0	0	d	d	d	d	d	d	d	d
13	1	1	0	1	d	d	d	d	d	d	d	d
14	1	1	1	0	d	d	d	d	d	d	d	d
15	1	1	1	1	d	d	d	d	d	d	d	d

2. 2.1 D Flip Flop:

CIK	D	Q	Q’	State
1	0	Q	Q’	No Change
1	0	0	1	Resets Q to 0
1	1	1	0	Sets Q to 1
0	x	Q	Q’	No Change

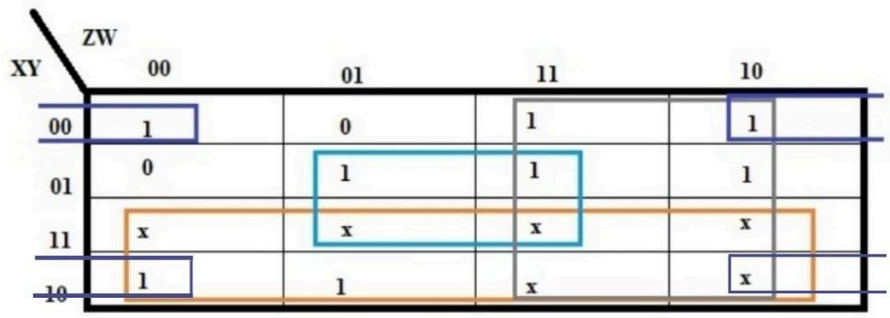
2. 2.3 BCD to 7 segment Decoder:

	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

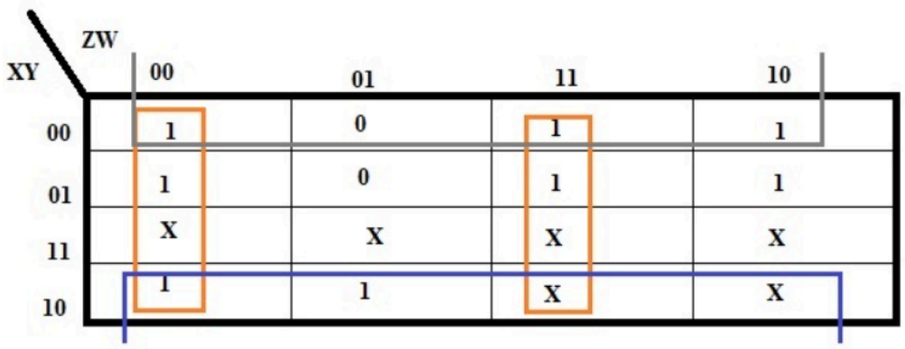
2. 3. K-Maps

2. 3.1 BCD to 7 segment Decoder:

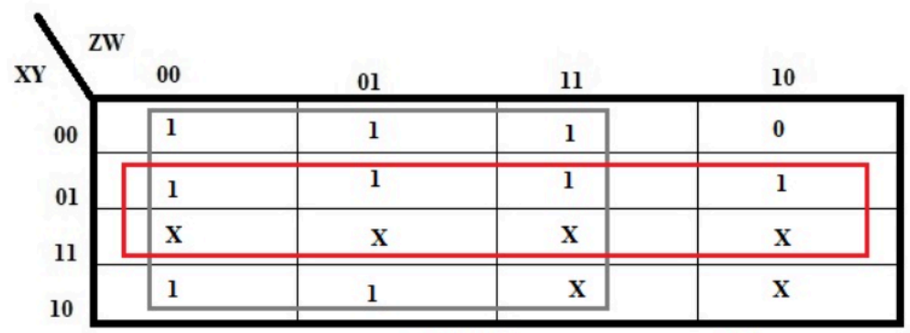
a = F1 (X, Y, Z, W) = ∑m (0, 2, 3, 5, 7, 8, 9)



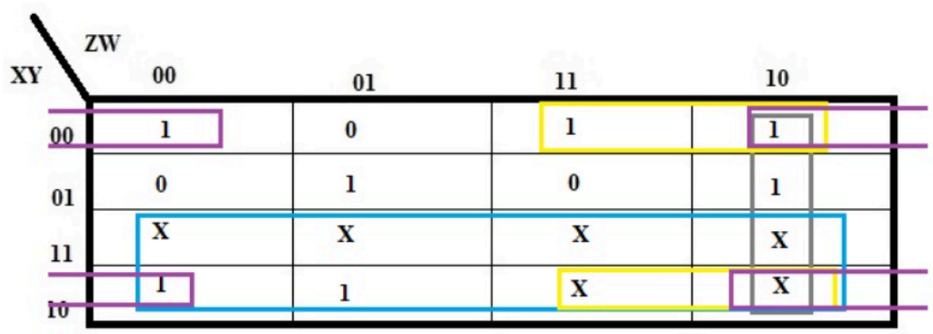
b = F2 (X, Y, Z, W) = ∑m (0, 1, 2, 3, 4, 7, 8, 9)



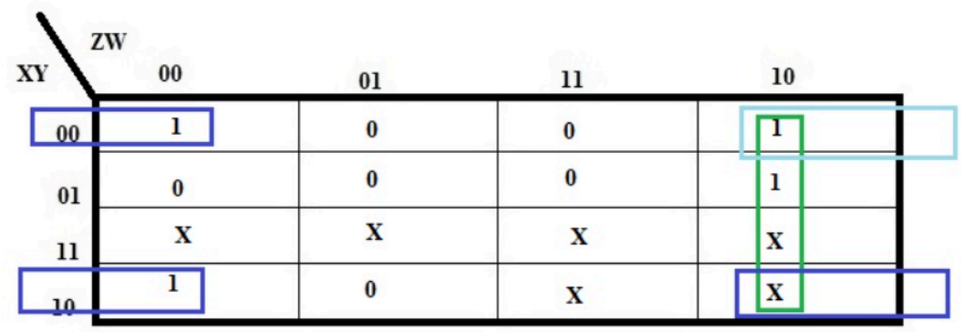
c = F3 (X, Y, Z, W) = ∑m (0, 1, 3, 4, 5, 6, 7, 8, 9)



d = F4 (X, Y, Z, W) = ∑m (0, 2, 3, 5, 6, 8)

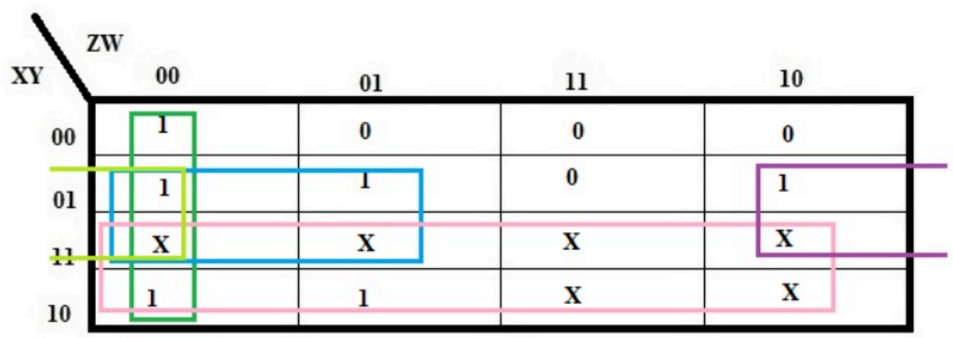


e = F5 (X, Y, Z, W) = ∑m (0, 2, 6, 8)



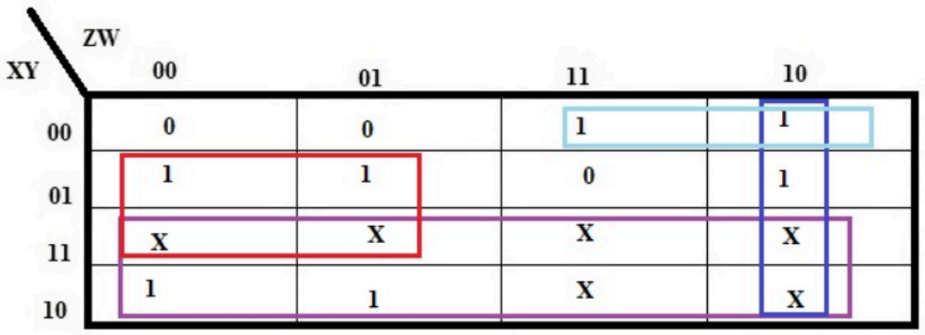
XY \ ZW	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

f = F6 (X, Y, Z, W) = ∑m (0, 4, 5, 6, 8, 9)



XY \ ZW	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

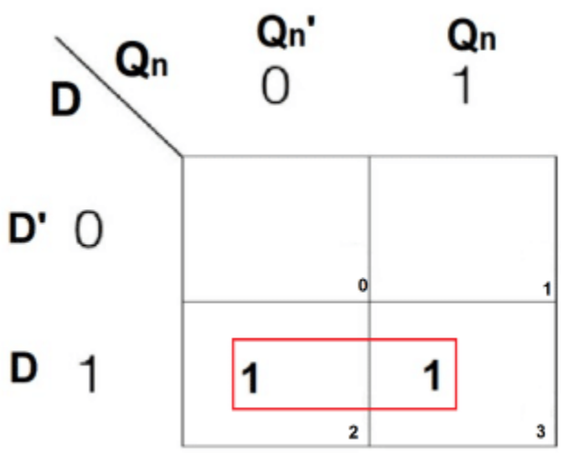
g = F7 (X, Y, Z, W) = ∑m (2, 3, 4, 5, 6, 8, 9)



XY \ ZW	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

2. 3.2 D Flip Flop:

Q+ = D



2. 3.3 BCD counter:

D_A:

		AB			
		00	01	11	10
CD	00	0	0	X	1
	01	0	0	X	0
	11	0	1	X	X
	10	0	0	X	X

$D_A = A\bar{D} + BCD$

D_B:

		AB			
		00	01	11	10
CD	00	0	1	X	0
	01	0	1	X	0
	11	1	0	X	X
	10	0	1	X	X

$D_B = B\bar{C} + B\bar{D} + \bar{B}CD$

D_C:

		AB			
		00	01	11	10
CD	00	0	0	X	0
	01	1	1	X	0
	11	0	0	X	X
	10	1	1	X	X

$D_C = C\bar{D} + \bar{A}\bar{C}D$

D_D:

		AB			
		00	01	11	10
CD	00	1	1	X	1
	01	0	0	X	0
	11	0	0	X	X
	10	1	1	X	X

$D_D = \bar{D}$

2. 4. Boolean Functions:

2. 4.1 BCD Counter:

$D_A = AD' + BCD$

$D_B = BC' + BD' + B'CD$

$D_C = CD' + A'C'D$

$D_D = D'$

2. 4.2 BCD to 7 segment Decoder:

$a = X + Z + YW + Y'W'$

$b = Y' + Z'W' + ZW$

$C = Y + Z' + W$

$d = Y'W' + ZW' + YZ'W + Y'Z + X$

$e = Y'W' + ZW'$

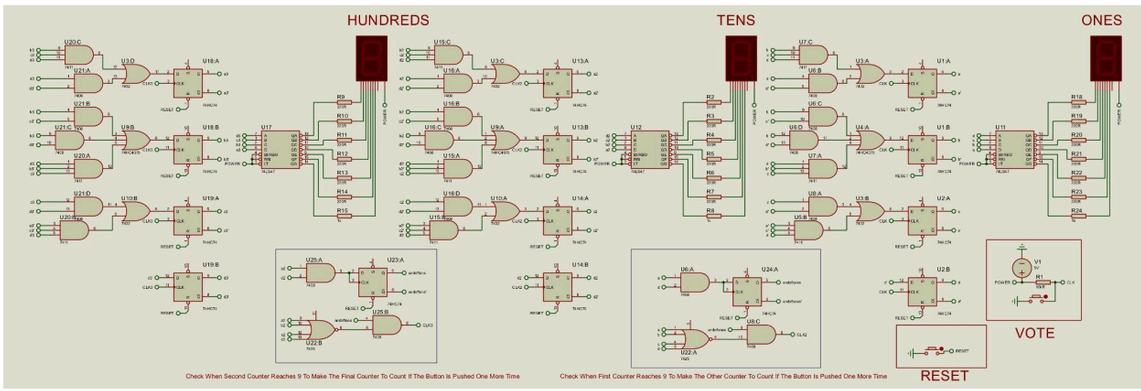
$f = X + Z'W' + YZ' + YW'$

$g = X + YZ' + Y'Z + ZW'$

2. 4.3 D Flip Flop

$Q_{n+1} = D$

2. 5. The Logic Circuits Design



3. Implementation

3.1 Implementation the system using Verilog HDL

- **7-Segment Decoder Verilog code**

```
module decoder_7seg(A, B, C, D, a, b, c, d, e, f, g);
input A, B, C, D;
output a, b, c, d, e, f, g;

assign a =      A | C | B&D | ~B&~D;
assign b =      ~B | ~C&~D | C&D;
assign c =      B | ~C | D;
assign d =      ~B&~D | C&~D | B&~C&D | ~B&C | A;
assign e =      ~B&~D | C&~D;
assign f =      A | ~C&~D | B&~C | B&~D;
assign g =      A | B&~C | ~B&C | C&~D;

endmodule
```

- **Counter Verilog code**

```
//to code any counter we just describe the behavior of it using verilog
module bcd_counter(clk,reset,count);
input clk,reset;

//the reason for creating a counter that counts in 10 bits
//is because we want it to count to 999
//Log2(999)= 9.9643408677924
//so we need at least 10 bits to represent the number 999
output reg [9:0]count=0;
//creating a vector to represent the count
//it is from a register type because of the always block we created below

//always block for repeating the counting operation
always @(posedge clk or posedge reset)
if(reset)
//we use <= instead of just = because we are dealing with sequential logic
count<=0;
else if(count==999)
count<=0;
else
count <= count + 1;

endmodule

module bcd_conter_dut();
wire clk;
reg reset;
wire [9:0]count;
initial
begin
reset=1; #50;
reset=0;
end
bcd_counter bcd_cnt(clk,reset,count);
```

```
endmodule
```

- **Connecting the Counter to three 7-Segment Decoders Verilog code**

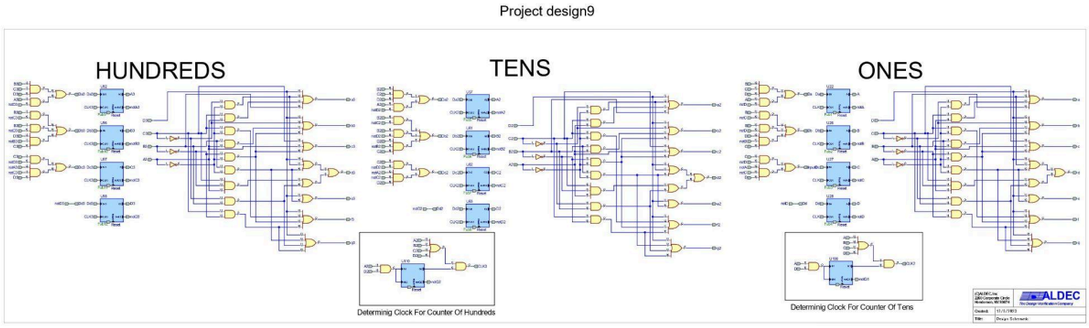
```
module counter_to_decoder(clk,reset,led,led2,led3);

input clk,reset;

//the final outputs which represent the leds of the 7-segment display
output [6:0]led,led2,led3;
wire [9:0]count;

bcd_counter bcd_cnt(clk,reset,count);

//each digit starts from 0 to 9 so we need 4 bits for each digit that's why we created a vector
wire [3:0]digit1 =count%10; //gets the first right number of the count and then we connect it to the first 7-segment decoder
wire [3:0]digit2 = (count/10)-(count/100)*10; //gets the number in the middle and then we connect it to the second 7-segment decoder
wire [3:0]digit3 =count/100; //gets the first left number of the count
//and then we connect it to the third 7-segment decoder



//the connections to all the three 7-segment decoders to create a 3-digit 7-segment display
decoder_7seg d7seg(digit1[3],digit1[2],digit1[1],digit1[0],
led[6],led[5],led[4],led[3],led[2],led[1],led[0]);

decoder_7seg d7seg2(digit2[3],digit2[2],digit2[1],digit2[0],
led2[6],led2[5],led2[4],led2[3],led2[2],led2[1],led2[0]);

decoder_7seg d7seg3(digit3[3],digit3[2],digit3[1],digit3[0],
led3[6],led3[5],led3[4],led3[3],led3[2],led3[1],led3[0]);


endmodule
```

- **Creating a clock generator using Verilog code and connecting the final code to it**

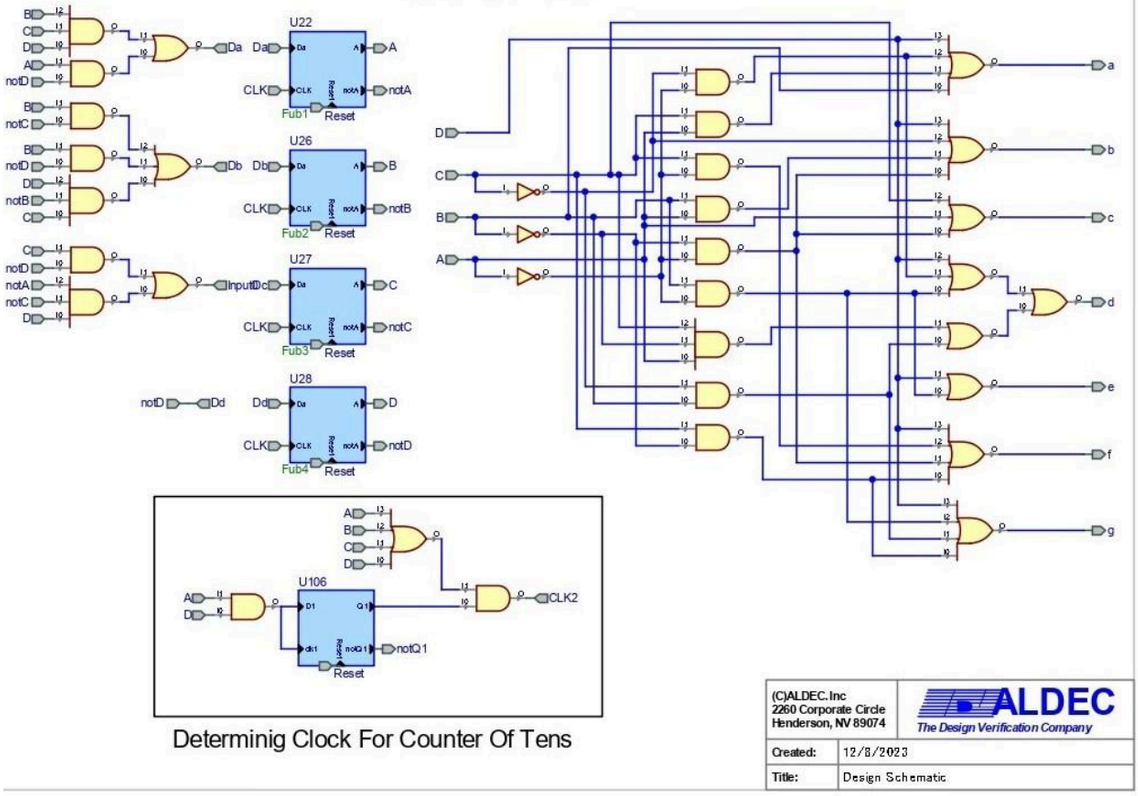
```
module ClockGen(CLK);
output CLK;

//we must define the clock as a register
//because we initialized, assigned and added it to always block
//so we can't treat it as a normal wire
reg CLK;

//defining the inputs and outputs of our final code
//to create an instance out of it and send it the generated clock signal we made
wire reset;
wire [6:0]led,led2,led3;
wire [9:0]count;

initial CLK=0;

//we can't use just assign statement or initial block because we need to repeat it
always
```

```
//we must add a delay time to see the generated wave clearly
//we set the delay time to 50 ps
#50 CLK = ~CLK;

//creating an instance of our final code
counter_to_decoder countertodecoder(CLK,reset,led,led2,led3);
endmodule


//creating the testbench
//it doesn't take any inputs or outputs
module ClockGen_DUT();
//we want to create an instance of the clockGen module
//so we must first define its output clk

//we defined it as a wire
//because we didn't use it inside of an always or initial block
wire CLK;
reg reset;
wire [9:0]count;
initial
begin
reset=1; #50;
reset=0;
end

//creating the instance
ClockGen CG1(CLK);

endmodule
```

3.2 Give the Schematic diagram using Active HDL



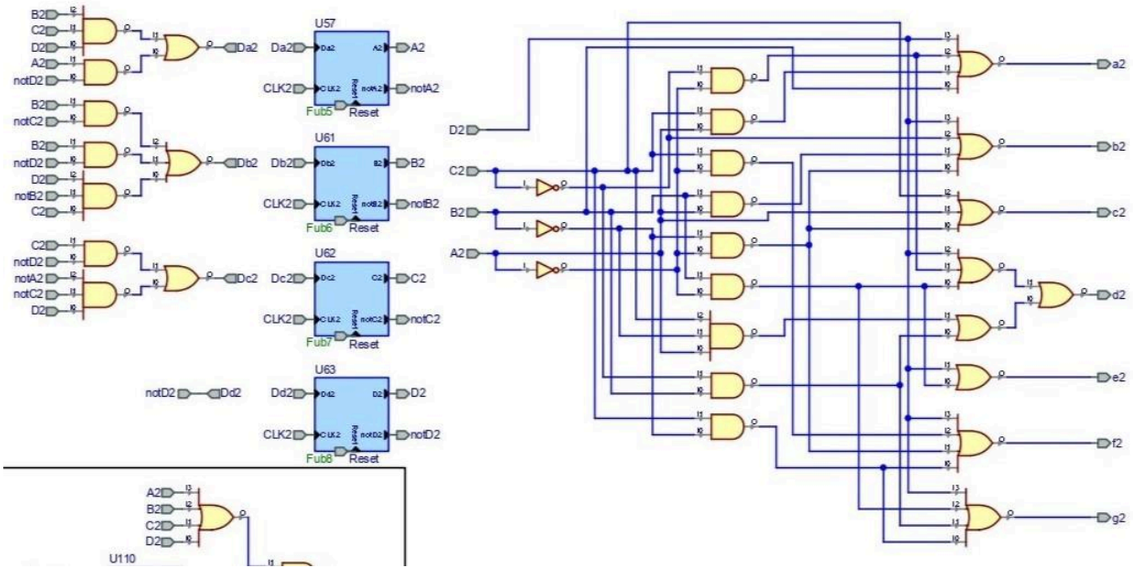
ONES



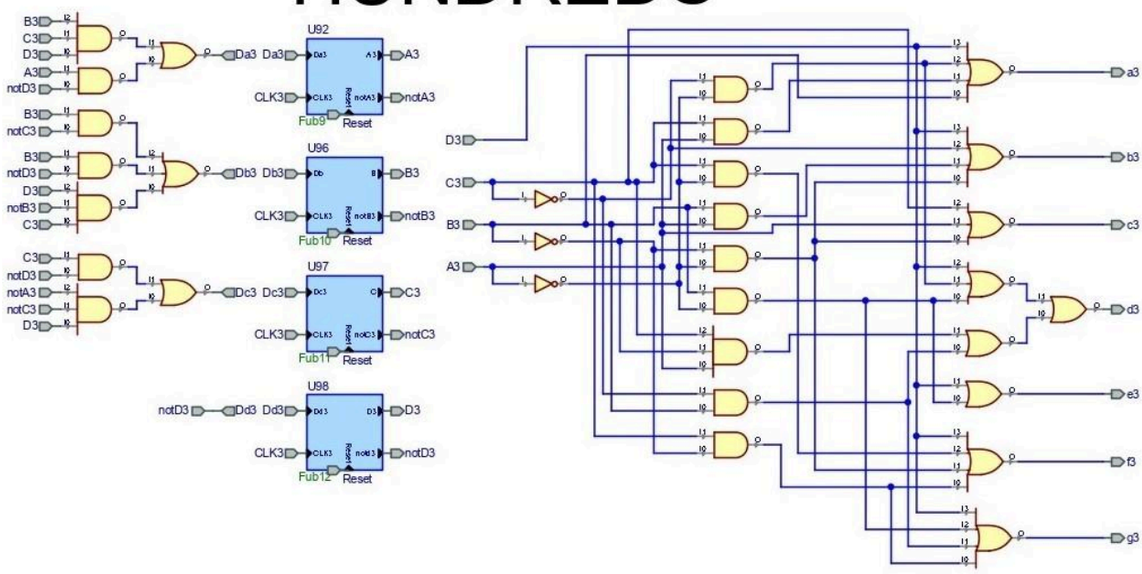
Determinig Clock For Counter Of Tens

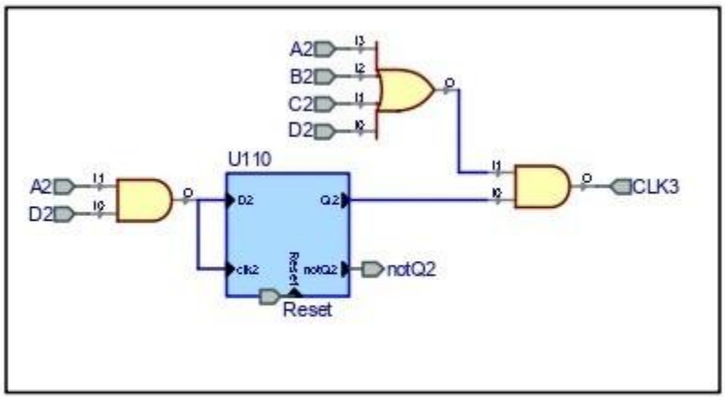
(C)ALDEC.Inc 2260 Corporate Circle Henderson, NV 89074	ALDEC The Design Verification Company
Created:	12/8/2023
Title:	Design Schematic

TENS

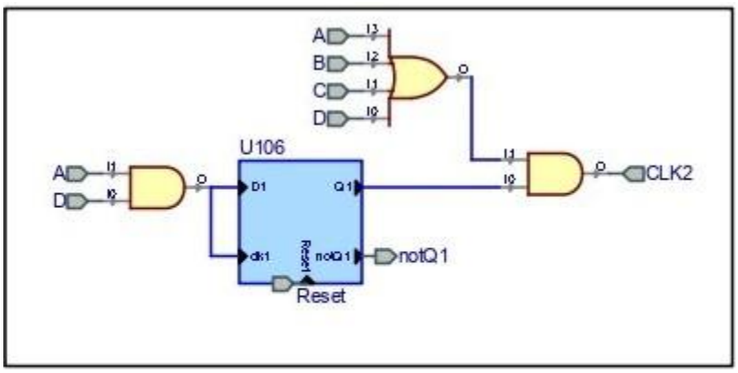


HUNDREDS





Determinig Clock For Counter Of Hundreds



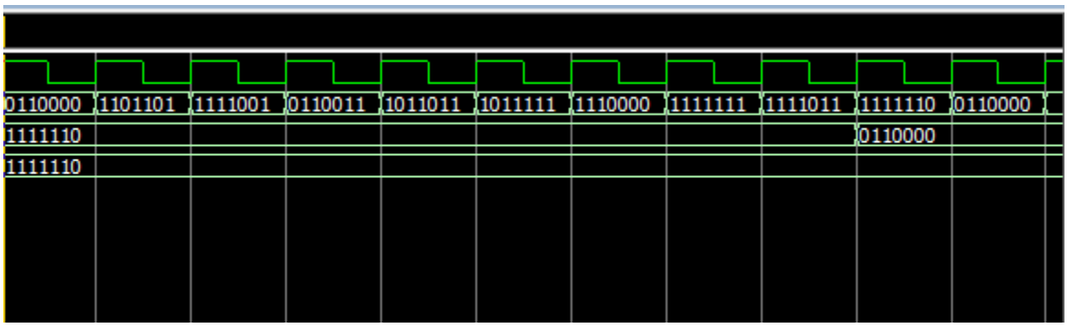
Determinig Clock For Counter Of Tens

3.3 Simulate the design

The code simulation:

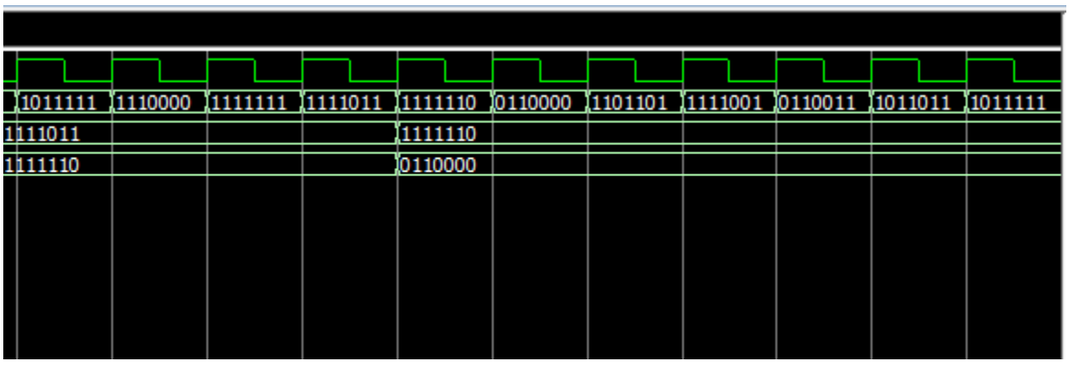
Beginning:

The first BCD counter is counting and the other two are stable
When the first counter reaches 10 it will reset to zero and the second one will start counting as tens

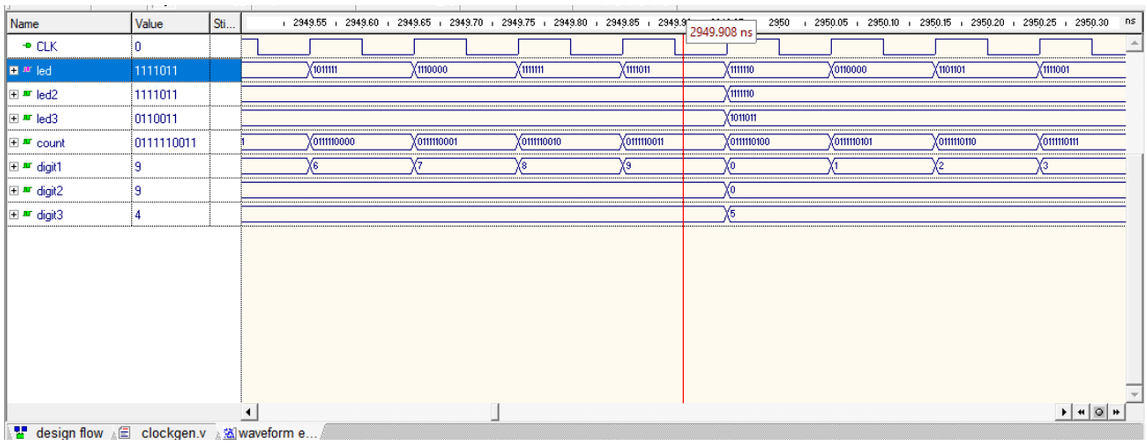


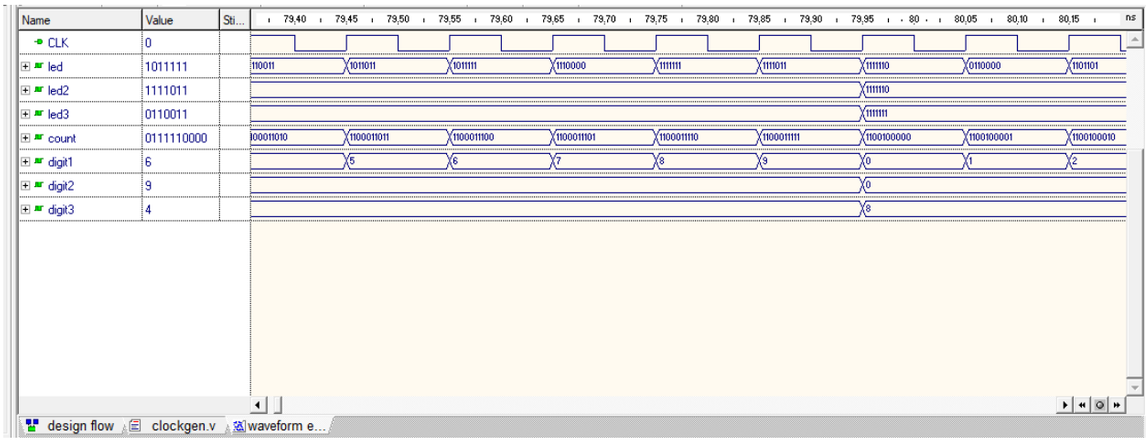
Along the way:

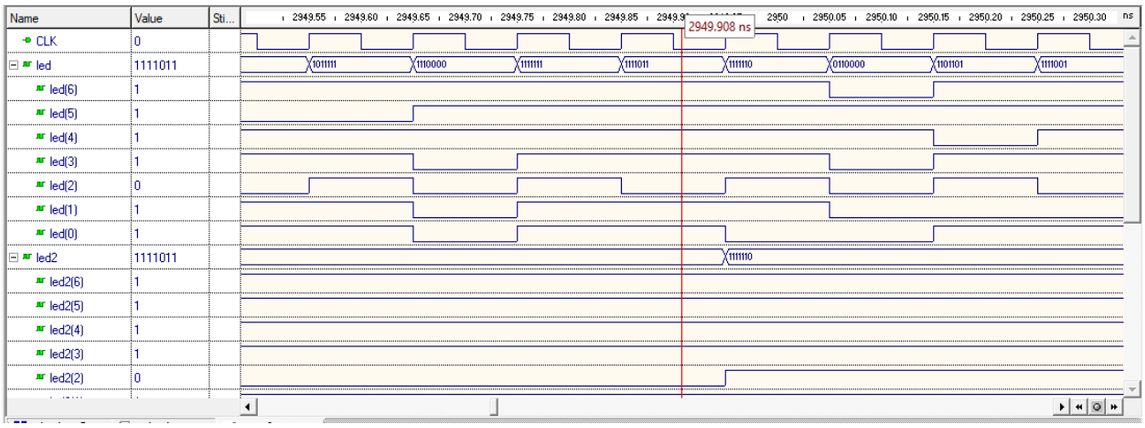
The same happens for the rest of BCD counters

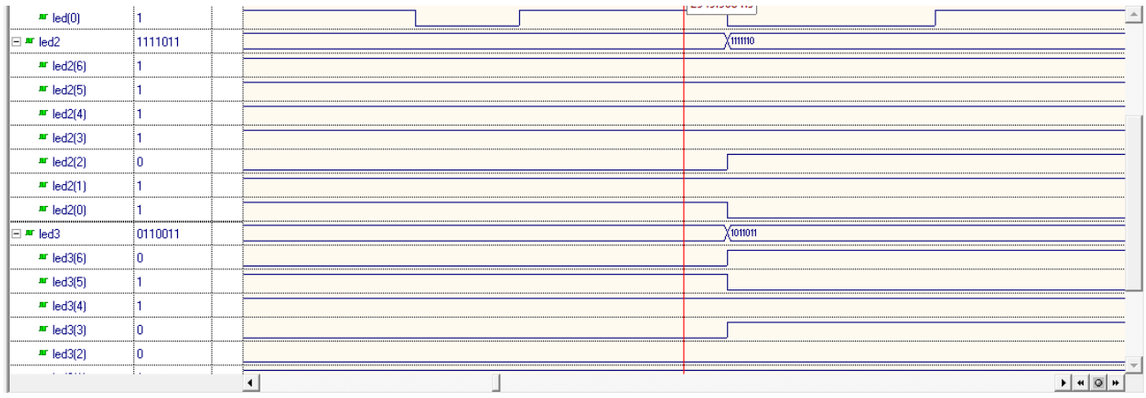


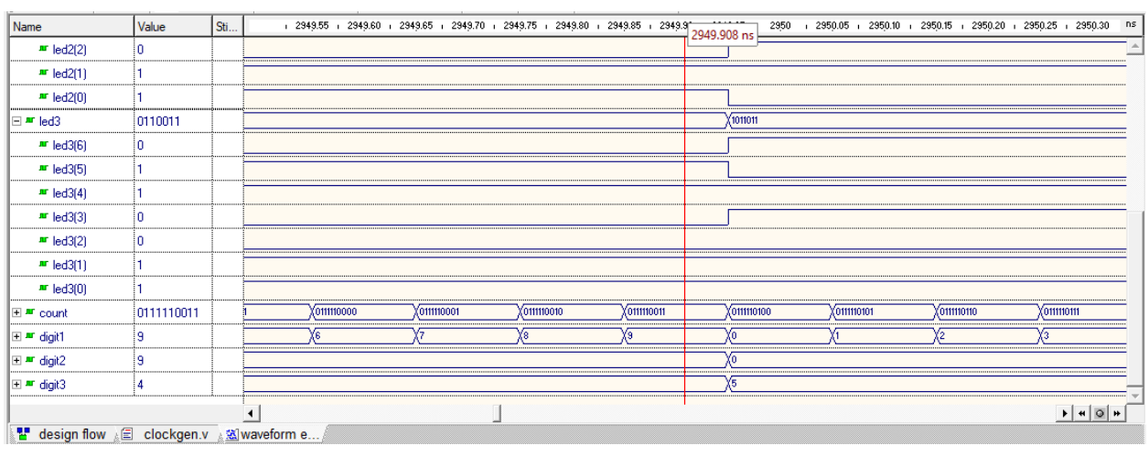
More Detailed Simulation:








led(0) 1
led2 1111011
led2(6) 1
led2(5) 1
led2(4) 1
led2(3) 1
led2(2) 0
led2(1) 1
led2(0) 1
led3 0110011
led3(6) 0
led3(5) 1
led3(4) 1
led3(3) 0
led3(2) 0
1111110
1011011


Name
Value
Sti...
2949.908 ns
ns
led2(2) 0
led2(1) 1
led2(0) 1
led3 0110011
led3(6) 0
led3(5) 1
led3(4) 1
led3(3) 0
led3(2) 0
led3(1) 1
led3(0) 1
count 0111110011
digit1 9
digit2 9
digit3 4
1011011
0
5
design flow
clockgen.v
waveform e...

The circuit simulation: