B38DF **Lab 1 Report Template**

**PRINT NAME: ………Yupu Wang………**

**Matriculation No: ………H00365030(20012100012) ………**

Complete the following sections:

1. Section 7 Checking the Simulation Result

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Does the actual truth table found from simulation match the intended truth table on page 5?***    Through the above simulation, we can get the truth table:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | A | B | C | D | Z | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | 0 | | 0 | 1 | 0 | 0 | 0 | | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | 0 | | 1 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 0 | | 1 | 0 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 0 | | 1 | 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | 1 |   We can get that the actual truth table found from simulation matches the intend truth table on Page 5. |

1. Section 8 Modifying the Logic Function

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Your K-map and your derived logic expression derived should be shown below***  We know the truth table as follows:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | A | B | C | D | Z | | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 0 | 1 | | 0 | 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 0 | 1 | | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 0 | | 1 | 0 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 1 | | 1 | 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | 0 |   Hence, we can get the K-map and the logic expression: |

1. Modified Verilog Source file

|  |
| --- |
| ***(i)Give verilog code for the correct logic expression. (ii)Is this structural or behavioural representation?***  ***(iii)Give your modified source file.***   1. *Verilog code for the correct logic expression:*   Z=(~A) | (B & ~D) | (~B & C & D);   1. This is structural representation. Because it describes functions defined using basic components such as inverters, multiplexers, adders, decoders and basic logic gates. It’s different from the behavioural representation, which describes the function of a set of concurrent algorithms. 2. Modified source file   module lab1(A,B,C,D,Z); //declare module name as lab1 with inputs/outputs A,B,C,D and Z    input A,B,C,D;  output Z;    reg Z; //value of z is assigned in an 'always' block, so we declare it as data type ‘reg    always @ (A,B,C,D) //always block , generates signals based on (A,B,C,D)  begin //marks start of code in the always block.  Z=(~A) | (B & ~D) | (~B & C & D); // Z assigned value of logic function specified.  // ~ means 'NOT' or 'inverse', & means AND, | means OR  end //marks end of code in the always block. Equivalent to } in C programming    endmodule |

1. Simulator Waveforms for 3 above.

|  |
| --- |
|  |

1. Verilog code for Section 10.

|  |
| --- |
| module lab2 (A,B,W,X,Y,Z);  input A,B;  output W,X,Y,Z;    reg W,X,Y,Z; //declare W,X,Y,Z as data type ‘reg    always @ (A,B)  begin //start of always block  if((A==0)&(B==0))  begin //start of if block  W=0;X=0;Y=0;Z=0; //use the behavioral design  end //end of code in the if block    else if((A==0)&(B==1))  begin  W=0;X=0;Y=1;Z=1;  end    else if((A==1)&(B==0))  begin  W=1;X=1;Y=0;Z=0;  end    else if((A==1)&(B==1))  begin  W=1;X=0;Y=0;Z=1;  end  end //marks end of code in the always block.  endmodule |

1. Simulator Waveforms for 5 above.

|  |
| --- |
|  |