Student Name: Ashna Mittal

Student Number: 251206758

CS 3350B: Assignment 2

Solution 1:

Original Expression: (A.B.C)' + (A.B)'.C + A'.B.C' + A.B'.C + A.(B.C)'

Using *De Morgan's law*: (X.Y)' = X' + Y'

= (A' + B' + C') + (A' + B').C + A'.B.C' + A.B'.C + A.(B' + C')

= A' + B' + C' + A'.C + B'.C + A'.B.C' + A.B'.C + A.B' + A.C'

Using *Absorption law*: X + X.Y = X

This eliminates A'.B.C' as A' absorbs it, A.B'.C as B'.C absorbs it, and A.C' as A absorbs it.

= A' + B' + C' + A'.C + B'.C + A.B'

Using *Absorption law* again, A'.C is absorbed by A', B'.C is absorbed by B' and A.B' is absorbed by A' or B'. Hence, we are left with:

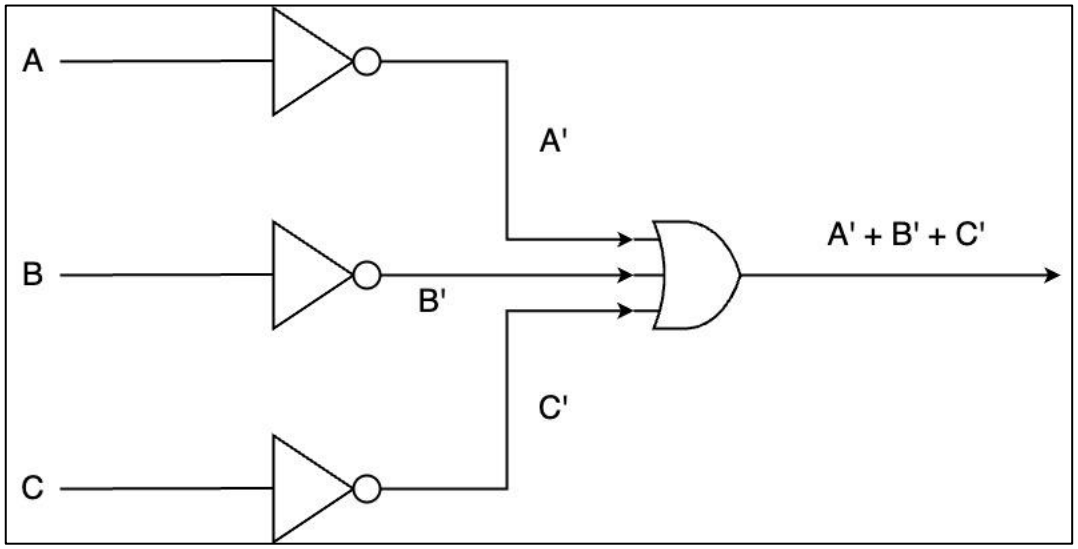
Final Expression = A' + B' + C'

Truth Table for the original expression and final expression:

A	B	C	A'	B'	C'	(ABC)'	(AB)'C	A'BC'	AB'C	A(BC)'	**OE**	**FE**
0	0	0	1	1	1	1	0	0	0	0	**1**	**1**
0	0	1	1	1	0	1	1	0	0	0	**1**	**1**
0	1	0	1	0	1	1	0	1	0	0	**1**	**1**
0	1	1	1	0	0	1	1	0	0	0	**1**	**1**
1	0	0	0	1	1	1	0	0	0	1	**1**	**1**
1	0	1	0	1	0	1	1	0	1	1	**1**	**1**
1	1	0	0	0	1	1	0	0	0	1	**1**	**1**
1	1	1	0	0	0	0	0	0	0	0	**0**	**0**

As evident from the truth table, the original expression column is same as the final expression column. Hence, the final expression is validated.

Logic circuit diagram for the final expression:



Solution 2:

The only two binary gates that are considered as functionally complete gates are NAND and NOR. These gates are functionally complete because by using any one of them alone, it is possible to implement any circuit. To prove that the gates are functionally complete, we can show that the operators of that set can mimic the functionality of the set {+ , . , ¬ }.

- NAND is functionally complete.
  NAND ≡ |
  To prove functional completeness, we can show that the operators of the set can mimic the functionality of the set {+,·,¬}.

  ¬A ≡ A | A

A	A’	A . A	(A . A)’
0	1	0	1
1	0	1	0

  A . B ≡ (A | B)’ ≡ (A | B) | (A | B)

A	B	X = A | B	X | X
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

  A + B ≡ ((A + B)’)’ ≡ (A’ . B’)’ ≡ (A | A) | (B | B)

A	B	A’	B’	A’ | B’
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

- NOR is functionally complete.
  NOR ≡ |
  To prove functional completeness, we can show that the operators of the set can mimic the functionality of the set {+,·,¬}.

  ¬A ≡ A | A

A	A’	A + A	(A + A)’
0	1	0	1
1	0	1	0

  A + B ≡ (A | B)’ ≡ (A | B) | (A | B)

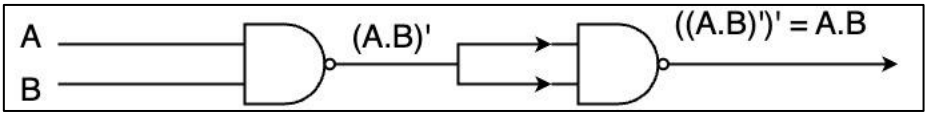
A	B	X = A | B	X | X

0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

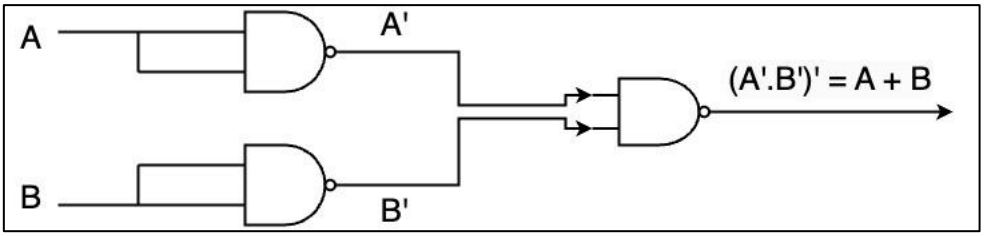
A . B ≡ ((A + B)')' ≡ (A' . B')' ≡ (A | A) | (B | B)

A	B	A'	B'	A' | B'
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

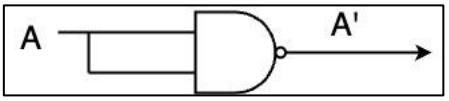
1. NAND gates only:
   a. AND: A . B



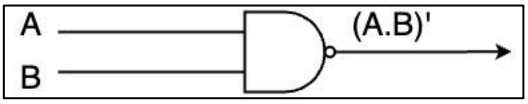
   b. OR: A + B



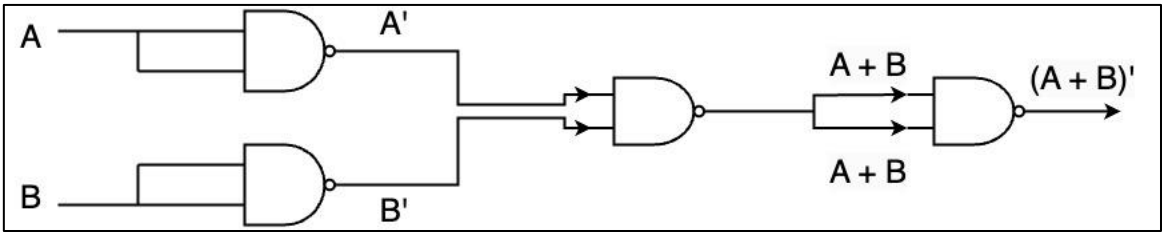
   c. NOT: A'



   d. NAND: (A . B)'

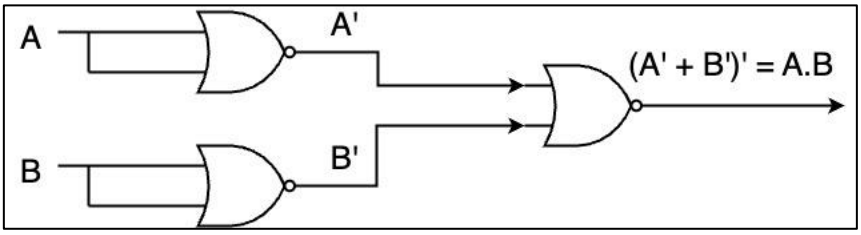


e. NOR: (A + B)’

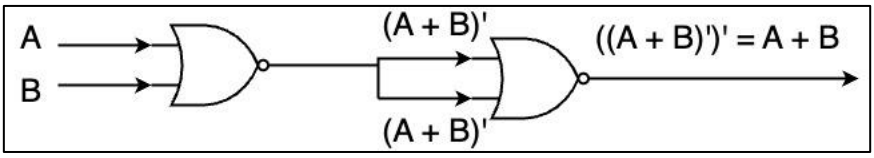


2. NOR gates only:

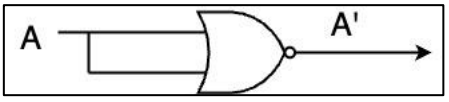
a. AND: A . B



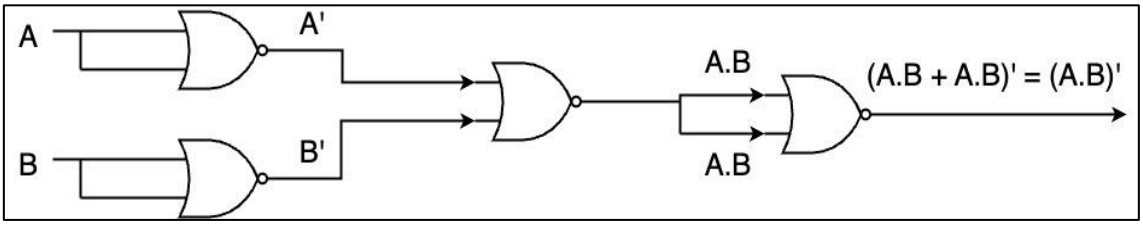
b. OR: A + B



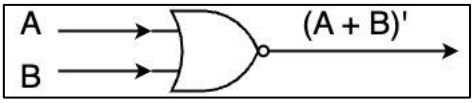
c. NOT: A’



d. NAND: (A.B)’



e. NOR: (A + B)’



Solution 3:

In Toggle flip-flop, if input is 1, the current state is toggled. We must use the current state to determine the next state. If T = 0, it means 'hold' and the next state is same as the current state. If T = 1, it means 'Toggle' and the next state is opposite of the current state.

Here, we are given the next state of the MJ flip-flop. Hence the characteristic truth table of MJ flip-flop using T flip-flop below shows the current state and the toggle at each level.

Qn	M	J	Qn+1	T
0	0	0	1	1
0	0	1	0	0
0	1	0	Qn' = 1	1
0	1	1	Qn = 0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	Qn' = 0	1
1	1	1	Qn = 1	0

We can now make the Excitation Table using the toggle values from the above truth table:

Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Making the Karnaugh map using the truth tables above:

T - Qn \ MJ	**00**	**01**	**11**	**10**
0	1	0	0	1
1	0	1	0	1

Original Canonical SOP Expression: Qn'M'J' + Qn'MJ' + QnM'J + QnMJ'

Taking *Qn*'J' common

= Qn'J' (M'+ M) + QnM'J + QnMJ'

Using *Complementation law*: X + X' = 1

= Qn'J' . (1) + QnM'J + QnMJ'

Using *Identity law*: X . 1 = X

= Qn'J' + QnM'J + QnMJ'

Taking J' common

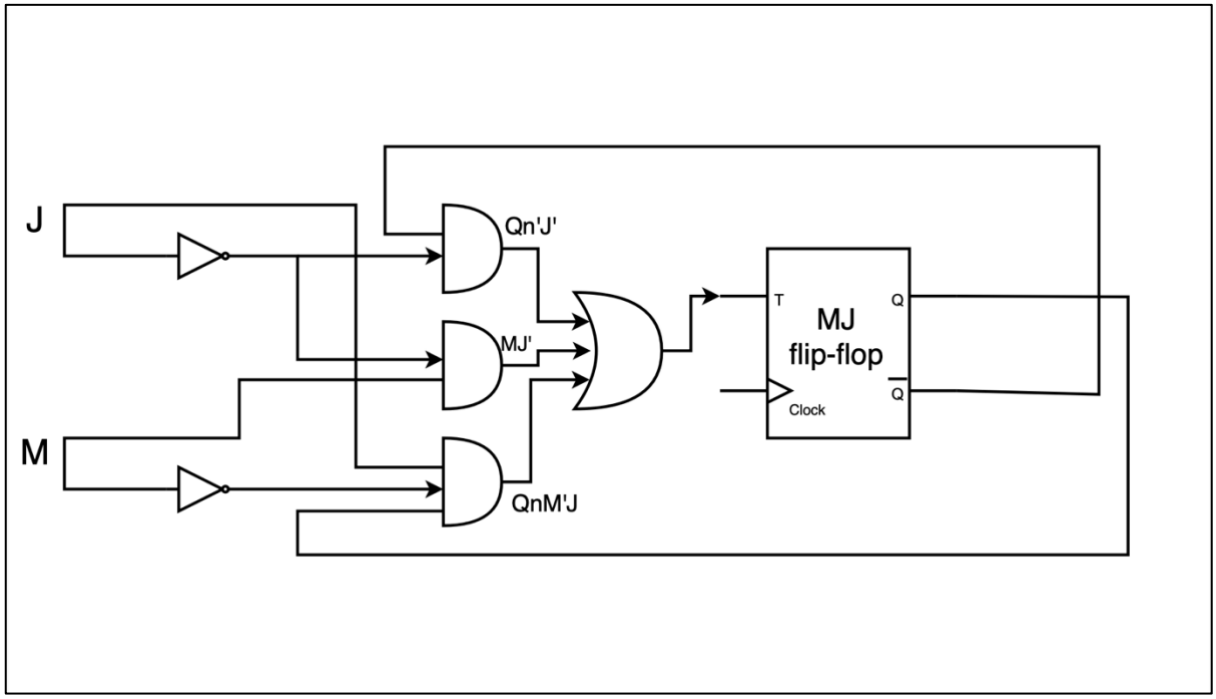
= J’ (Qn’+ QnM) + QnM’J

Using A + A’B ≡ A(1+B) + A’B ≡ A + AB + A’B ≡ A + (A + A’)B ≡ A + B

= J’(Qn’+ M) + QnM’J

Final Expression = J’Qn’+ J’M + QnM’J

The circuit diagram for MJ flip-flop using T Flip-Flop is:



Solution 4:

Synchronous counters are the ones in which all the flipflops are connected to one clock. In Toggle flip-flop, if input is 1, the current state is toggled. We must use the current state to determine the next state. If T = 0, it means 'hold' and the next state is same as the current state. If T = 1, it means 'Toggle' and the next state is opposite of the current state.

Here, we are given the counter outputs and the order. Hence, the characteristic truth table of Counters using T flip-flop below shows the current state, next state and the toggle at each level.

Qn (Current State)		Qn+1 (Next State)		T (Toggle)	
A	B	An+1	Bn+1	TA	TB
0	0	0	1	0	1
0	1	1	1	1	0
1	0	-	-	-	-
1	1	0	0	1	1

Making the Karnaugh maps for TA and TB using the truth tables above:

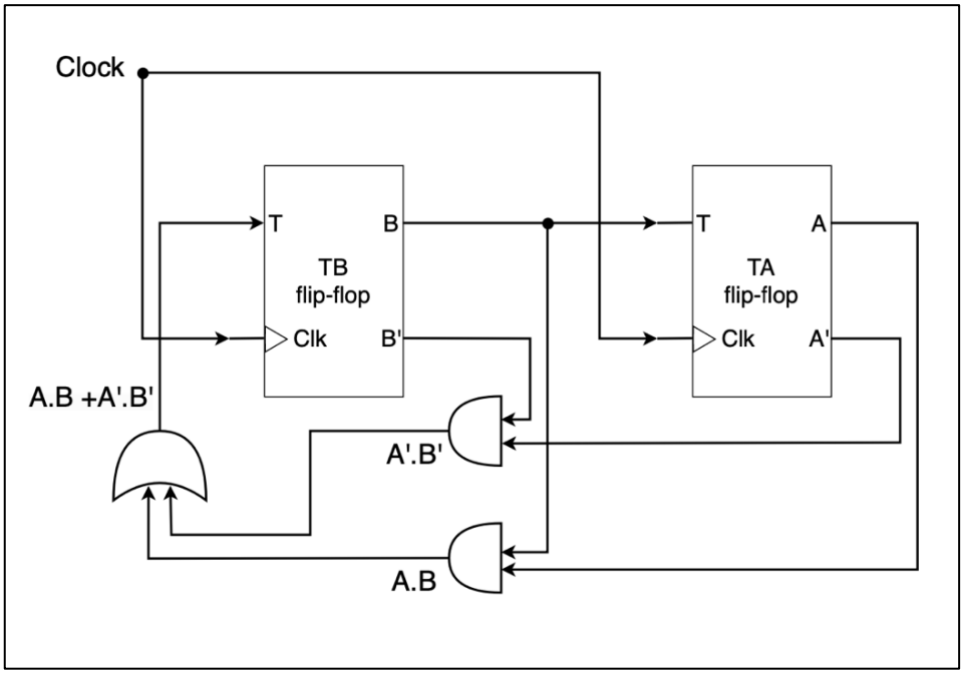
TA - A \ B	**0**	**1**
0	0	1
1	-	1

TA Canonical SOP expression :
= A' . B + A . B
= B . (A' + A)
= B . (1) ≡ B

TB - A \ B	**0**	**1**
0	1	0
1	-	1

TB Canonical SOP expression :
= A'. B' + A . B

The circuit diagram for synchronous counter using T Flip-Flop is:



Solution 5:

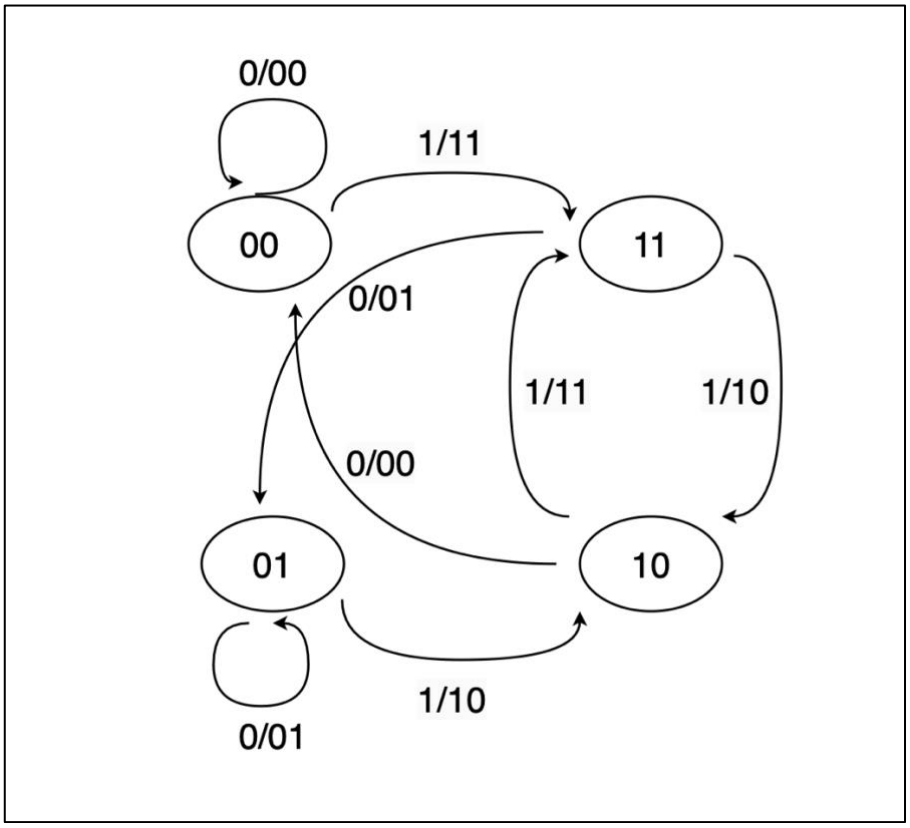
To design the Finite State Machine (FSM) for the coffee grinder control circuit, we will be considering the inputs, outputs, and the states of the system. The FSM will have 4 states(0,1,2,3), each representing a combination of the G (grinder) and F (fine/coarse setting) outputs. The input I from the heating element circuit will trigger the transitions. The states, inputs, and outputs would be:

States:
State 00: Grinder Inactive, Coarse Setting (G=0, F=0)
State 01: Grinder Inactive, Fine Setting (G=0, F=1)
State 10: Grinder Active, Coarse Setting (G=1, F=0)
State 11: Grinder Active, Fine Setting (G=1, F=1)

Inputs:
I: Input from the heating element circuit (Coffee Maker Turned On)

Outputs:
G: Grinder Activation/Inactivation Control
F: Course/Fine Grinding Setting

(a) Here we have labelled state 0 as 00, 1 as 01, 2 as 10 and 3 as 11. The Finite State machine diagram describing the state circuit is give below:



(b) The truth table describing the inputs, outputs, and transitions of the Finite State Machine is shown below:

Present State		Input	Next State		Output	
PSG	PSF	I	NSG	NSF	G	F
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	1	1	1	1	1
1	1	0	0	1	0	1
1	1	1	1	0	1	0

The first two columns represent the present state (PSG and PSF), the next column represents the input (I), the next two columns represent the next state (NSG and NSF), and the last two columns represent the output for each state(G and F).

- When the present state is 00 (Grinder Inactive, Coarse Setting) and the input I is 0 (Coffee machine off), the next state remains 00 (G=0, F=0). When the input I is 1 (Coffee Maker on), the next state becomes 11 (G=1, F=1).
- When the present state is 01 (Grinder Inactive, Fine Setting) and the input I is 0, the next state remains 01 (G=0, F=1). When the input I is 1, the next state becomes 10 (G=1, F=0).
- When the present state is 10 (Grinder Active, Coarse Setting) and the input I is 0, the next state becomes 00 (G=0, F=0). When the input I is 1, the next state becomes 11 (G=1, F=1).
- When the present state is 11 (Grinder Active, Fine Setting) and the input I is 0, the next state becomes 01 (G=0, F=1). When the input I is 1, the next state becomes 10 (G=1, F=0).

This truth table describes the combinational logic of the FSM, showing how the next state and outputs G and F are determined based on the current state and the input I. Each row corresponds to a specific combination of inputs and current states, resulting in the corresponding next state and outputs.

(c) Making the Karnaugh map for NSF using the truth tables above:

Input (I) \ Output(PS)	**00**	**01**	**11**	**10**
0	0	1	1	0
1	1	0	0	1

Original DNF Expression for NSF: IPSG’PSF’ + I’PSG’PSF + I’PSGPSF + IPSGPSF’

Taking I.PSF’ and I’.PSF common,

= I.PSF'. (PSG' + PSG) + I'.PSF. (PSG' + PSG)

= I.PSF'. (1) + I'.PSF. (1)

Final expression = I.PSF' + I'.PSF

Making the Karnaugh map for NSG using the truth tables above:

Input (I) \ Output(PS)	**00**	**01**	**11**	**10**
0	0	0	0	0
1	1	1	1	1

Original DNF Expression for NSG: IPSG'PSF' + IPSG'PSF + IPSGPSF + IPSGPSF'

Taking I common,

= I. (PSG'PSF' + PSG'PSF + PSGPSF + PSGPSF')

= I. (PSG'(PSF' + PSF) + PSG(PSF + PSF'))

= I. (PSG' . (1) + PSG . (1))

= I. (PSG' + PSG)

= I. (1)

Final expression = I

Making the Karnaugh map for G using the truth tables above:

Input (I) \ Output(PS)	**00**	**01**	**11**	**10**
0	0	0	0	0
1	1	1	1	1

Original DNF Expression for G: IPSG'PSF' + IPSG'PSF + IPSGPSF + IPSGPSF'

Taking I common,

= I. (PSG'PSF' + PSG'PSF + PSGPSF + PSGPSF')

= I. (PSG'(PSF' + PSF) + PSG(PSF + PSF'))

= I. (PSG' . (1) + PSG . (1))

= I. (PSG' + PSG)

= I. (1)

Final expression = I

Making the Karnaugh map for F using the truth tables above:

Input (I) \ Output(PS)	00	01	11	10
0	0	1	1	0
1	1	0	0	1

Original DNF Expression for F: IPSG'PSF' + I'PSG'PSF + I'PSGPSF + IPSGPSF'

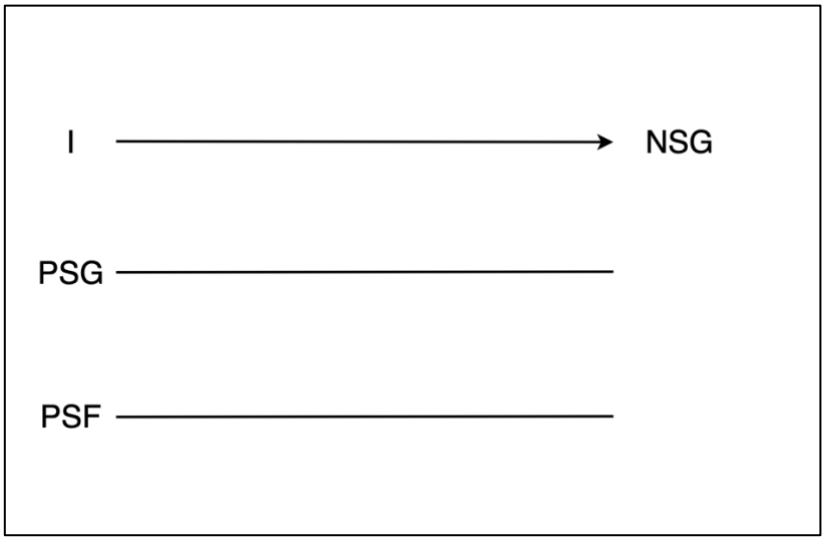
Taking I.PSF' and I'.PSF common,

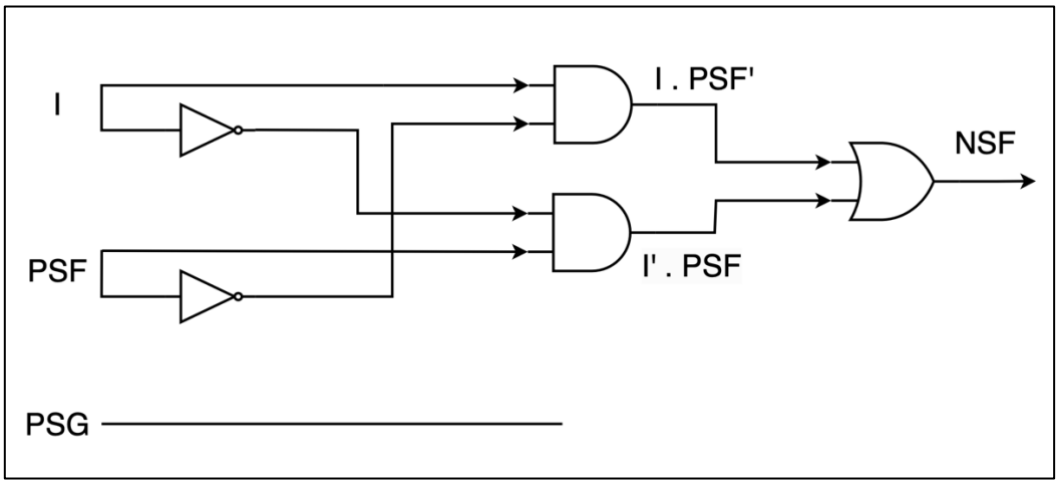
= I.PSF'. (PSG' + PSG) + I'.PSF. (PSG' + PSG)

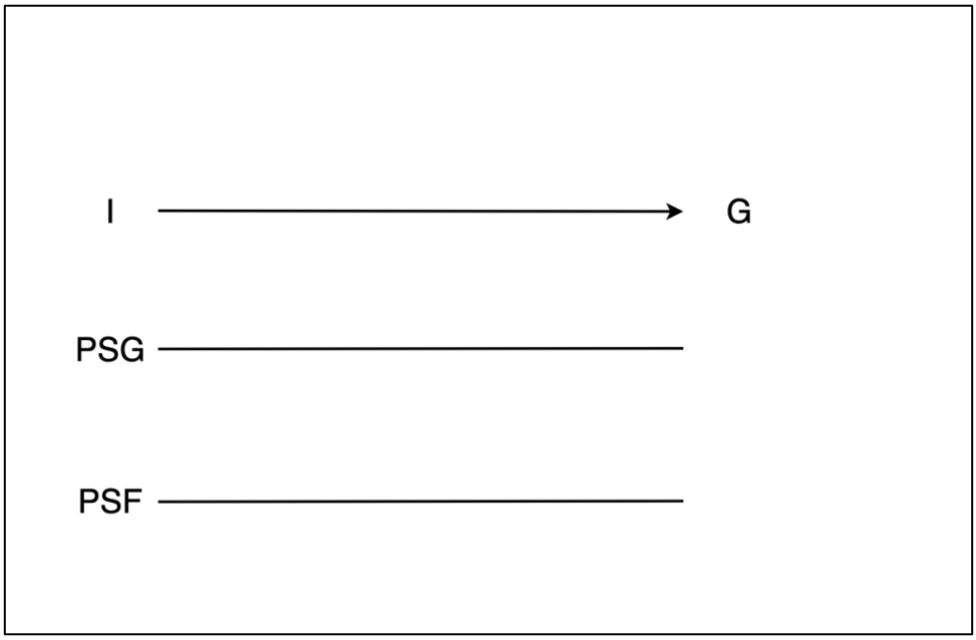
= I.PSF'. (1) + I'.PSF. (1)

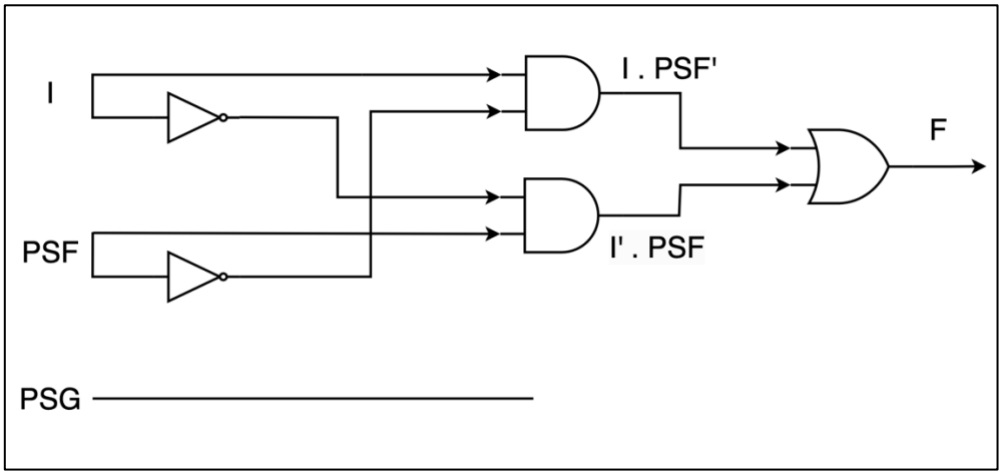
Final expression = I.PSF' + I'.PSF

(d) The circuit diagrams for the simplified expressions are as follows:






I
G
PSG
PSF


I
PSF
PSG
I . PSF'
I' . PSF
F