**Case Study 3: Boolean Circuit Equivalence**

**Step 1: Deriving Boolean Expressions**

We start by analyzing the given Boolean circuits. Each gate is carefully traced to obtain the final expression.

**Circuit (a):**

a) Input A passes through a NOT gate → **A'**  
b) Input C passes through a NOT gate → **C'**  
c) These are combined using AND → **(A'·C')**  
d) This output passes through a NOT gate → **(A'·C')'**  
e) By applying De Morgan’s Law: **(A'·C')' = A + C**  
f) Input B passes through a NOT gate → **B'**  
g) A and B' are combined using AND → **A·B'**  
h) A, B, and C are combined using AND → **A·B·C**  
i) OR operation between step (g) and step (h): **(A·B') + (A·B·C)**  
j) Simplify: **A·B' + A·C**

**Final Expression for Circuit (a):**  
**X = A·B' + A·C**

**Circuit (b):**

a) Input B passes through a NOT gate → **B'**  
b) OR gate with C → **B' + C**  
c) Input A AND with (B' + C) → **A·(B' + C)**  
d) Apply distributive law: **A·B' + A·C**

**Final Expression for Circuit (b):**  
**Y = A·B' + A·C**

**Step 2: Python Code Implementation**

The Boolean circuits can be simulated using Python. The code accepts inputs A, B, and C, then prints outputs X (Circuit A) and Y (Circuit B):

# Case Study 3: Boolean Circuit Equivalence

# This program simulates two Boolean circuits and checks their outputs.

# Function for Circuit A

def circuit\_a(A, B, C):

a = not A # Step a: NOT A

c = not C # Step b: NOT C

temp1 = not (a and c) # Step c-d: (A'·C')'

b\_not = not B # Step f: NOT B

e = A and b\_not # Step g: A·B'

f = A and B and C # Step h: A·B·C

result = e or f # Step i: (A·B') + (A·B·C)

return result

# Function for Circuit B

def circuit\_b(A, B, C):

b\_not = not B # Step a: NOT B

temp1 = b\_not or C # Step b: B' + C

result = A and temp1 # Step c: A·(B' + C)

return result

# Main program: User input

A = bool(int(input("Enter A (0 or 1): ")))

B = bool(int(input("Enter B (0 or 1): ")))

C = bool(int(input("Enter C (0 or 1): ")))

# Outputs

X = circuit\_a(A, B, C)

Y = circuit\_b(A, B, C)

# Print results

print("X (Circuit A) =", int(X))

print("Y (Circuit B) =", int(Y))

**Step 3: Truth Tables and Testing**

To confirm correctness, we generate truth tables for both circuits.

**Truth Table for Circuit A (X):**

The circuits can be represented in Python using Boolean logic operators.

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **X (Circuit A)** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Truth Table for Circuit B (Y):**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **Y (Circuit B)** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Testing with Console Outputs:**

Testing is performed to confirm that both circuits consistently yield the same outputs. Each test case includes input, working explanation, expected output, and observed result.

**Test Case 1: A=1, B=0, C=0**

* **Input:** A=1, B=0, C=0
* **Working:** Substituting values into both circuits yields X=1 and Y=1.
* **Output:** X=1, Y=1
* **Result:** Pass

**Console Output:**

A screenshot of a computer program

Description automatically generated

**Test Case 2: A=1, B=0, C=1**

* **Input:** A=1, B=0, C=1
* **Working:** Both expressions evaluate to 1.
* **Output:** X=1, Y=1
* **Result:** Pass

**Console Output:**

A screenshot of a computer code

Description automatically generated

**Test Case 3: A=1, B=1, C=0**

* **Input:** A=1, B=1, C=0
* **Working:** Both circuits evaluate to 0.
* **Output:** X=0, Y=0
* **Result:** Pass

**Console Output:**

A screenshot of a computer program

Description automatically generated

**Test Case 4: A=0, B=0, C=1**

* **Input:** A=0, B=0, C=1
* **Working:** Both circuits evaluate to 0.
* **Output:** X=0, Y=0
* **Result:** Pass

**Console Output:**

A screenshot of a computer program

Description automatically generated

**Step 4: Equivalence Verification**

By comparing truth tables and test outputs, we observe:

* For **all possible input values** of A, B, and C, the outputs X and Y are identical.
* Both circuits simplify to the **same Boolean expression: A·B' + A·C**.
* This proves that **Circuit A and Circuit B are equivalent**.