# CSCI 355 - Lab 3 Report

**Student: Luka Karanovic**
**Student #: 665778833**
**Instructor: Ajay Shrestha**

# 1 - Pre-Lab Answers

## 4.1 Boolean Function

Note: I will be using the Boolean Algebra Theorems handout with the number/letter of the axiom I'm using.

### 4.1.1 Use algebraic manipulation to prove:

**i) x + yz = (x + y)(x + z)**

This is just axiom 12b: x + yz = (x + y)(x + z)

To prove in more steps:
12a: x + yz = xx + xz + yx + yz
7a: x + yz = x + xz + yx + yz
12a: x + yz = x(1 + z + y) + yz
5b: x + yz = x(1) + yz
6a: x + yz = x + yz

**ii) xy + yz + ~xz = xy + ~xz**

6a: xy + 1yz + ~xz = xy + ~xz
8b: xy + (x + ~x)yz + ~xz = xy + ~xz
12a: xy + xyz + ~xyz + ~xz = xy + ~xz
13a twice: xy + ~xz = xy + ~xz

OR

6a: xy + 1yz + ~xz = xy + ~xz
8b: xy + (x + ~x)yz + ~xz = xy + ~xz
12a: xy + xyz + ~xyz + ~xz = xy + ~xz
12a: xy(1 + z) + ~xz(1 + y) = xy + ~xz
5b: xy(1) + ~xz(1) = xy + ~xz
6a: xy + ~xz = xy + ~xz

**iii) ~( ((a + b) + ~c + ~d))(~(ab)~c~d) ) = ~(a + b)cd + ab + c + d**

15a: ~((a + b) + ~c + ~d) + ~(~(ab)~c~d) = ~(a + b)cd + ab + c + d
15b: ~(a + b)~(~c)~(~d) +  ~(~(ab)) + ~(~c)  + ~(~d) = ~(a + b)cd + ab + c + d
9: ~(a + b)cd + ab + c + d = ~(a + b)cd + ab + c + d

### 4.1.2 Use algebraic manipulation to find the minimum sum-of-products (SOP) expression for the function:

**i) f = x1~x2~x3 + x1x2x4 + x1~x2x3~x4**

12a: f = x1(~x2~x3 + x2x4 + ~x2x3~x4)
12a: f = x1(~x2(~x3 + x3~x4) + x2x4)
12b: f = x1(~x2((~x3 + x3)(~x3 + ~x4) + x2x4)
8b: f = x1(~x2((1)(~x3 + ~x4)) + x2x4)

6a: f = x1(~x2(~x3 + ~x4) + x2x4)
12a: f = x1(~x2~x3 + ~x2~x4 + x2x4)
12a: f =  x1~x2~x3 + x1~x2~x4 + x1x2x4

**ii) f(x1, x2, x3) = Σm(1, 3, 4, 6, 7)**
f = ~x1~x2x3 + ~x1x2x3 + x1~x2~x3  + x1x2~x3 + x1x2x3

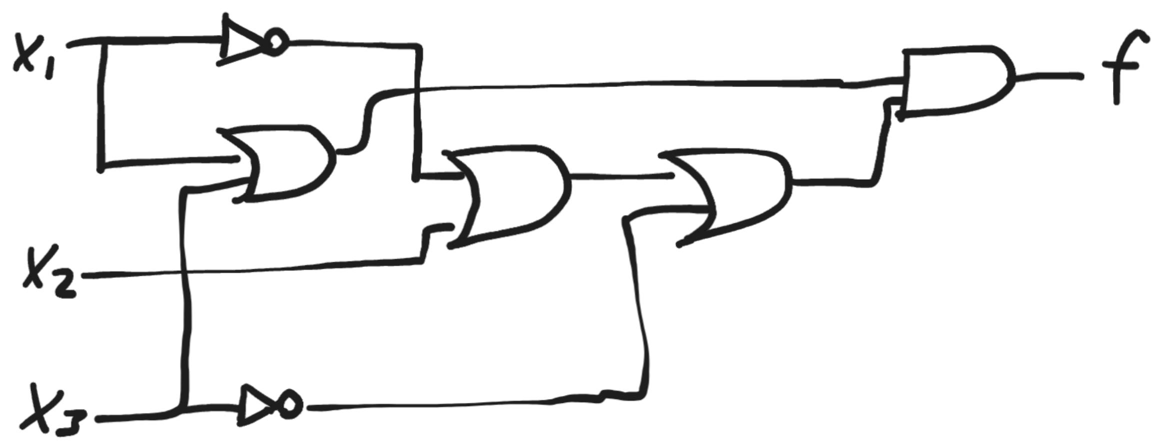
12a: ~x1x3(~x2 + x2) + x1~x3(~x2 + x2) + x1x2x3
8b: ~x1x3(1) + x1~x3(1) + x1x2x3
6a: ~x1x3 + x1~x3 + x1x2x3
12a: ~x1x3 + x1(~x3 + x2x3)
12b: ~x1x3 + x1((~x3 + x2)(~x3 + x3))
8b: ~x1x3 + x1((~x3 + x2)(1))
6a: ~x1x3 + x1(~x3 + x2)
12a: ~x1x3 + x1~x3 + x1x2

## 4.1.3 Use algebraic manipulation to find the simplest products-of-sums (POS) circuit that implements the function: f(x1, x2, x3) = ΠM(0, 2, 5)
f = (x1 + x2 + x3)(x1 + ~x2 + x3)(~x1 + x2 + ~x3)

12a: (x1x1 + x1~x2 + x1x3 + x2x1 + x2~x2 + x2x3 + x3x1 + x3~x2 + x3x3)(~x1 + x2 + ~x3)
7a: (x1 + x1~x2 + x1x3 + x2x1 + x2~x2 + x2x3 + x3x1 + x3~x2 + x3)(~x1 + x2 + ~x3)
8a: (x1 + x1~x2 + x1x3 + x2x1 + 0 + x2x3 + x3x1 + x3~x2 + x3)(~x1 + x2 + ~x3)
6b: (x1 + ~x2x1 + x3 + ~x2x3 + x1x3 + x1x3 + x2x1 + x2x3)(~x1 + x2 + ~x3)
- Removed 0 and reordered some terms

7b: (x1 + ~x2x1 + x3 + ~x2x3 + x1x3 + x2x1 + x2x3)(~x1 + x2 + ~x3)
12a: (x1(1 + ~x2) + x3(1 + ~x2 + x1 + x2 + x2))(~x1 + x2 + ~x3)
5b: (x1(1) + x3(1))(~x1 + x2 + ~x3)
6a: (x1 + x3)(~x1 + x2 + ~x3)

The circuit:



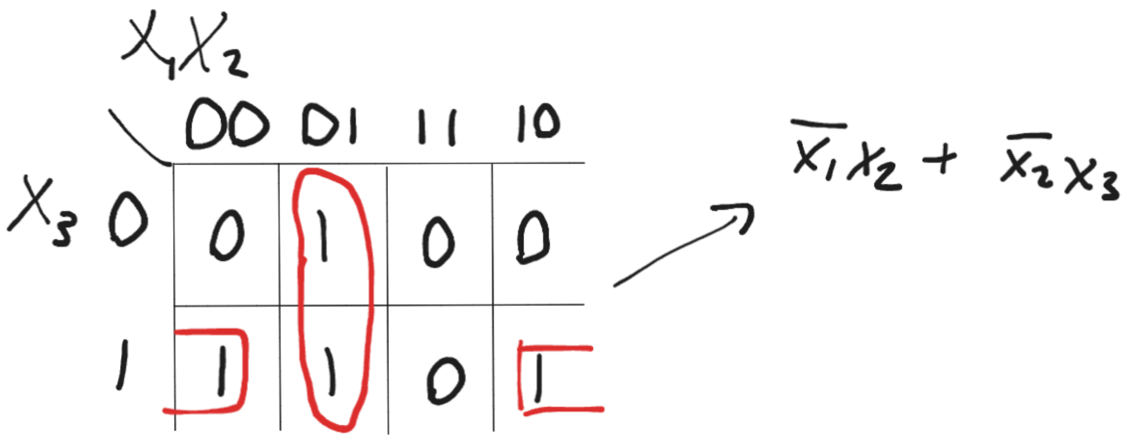
**4.1.4 Find the minimum-cost SOP and POS forms for the function f(x1, x2, x3) = Σm(1, 2, 3, 5)**

**i) Minimum-cost SOP**

**f(x1, x2, x3) = Σm(1, 2, 3, 5)**

f = ~x1~x2x3 + ~x1x2~x3 + ~x1x2x3 + x1~x2x3

Using a K-Map:



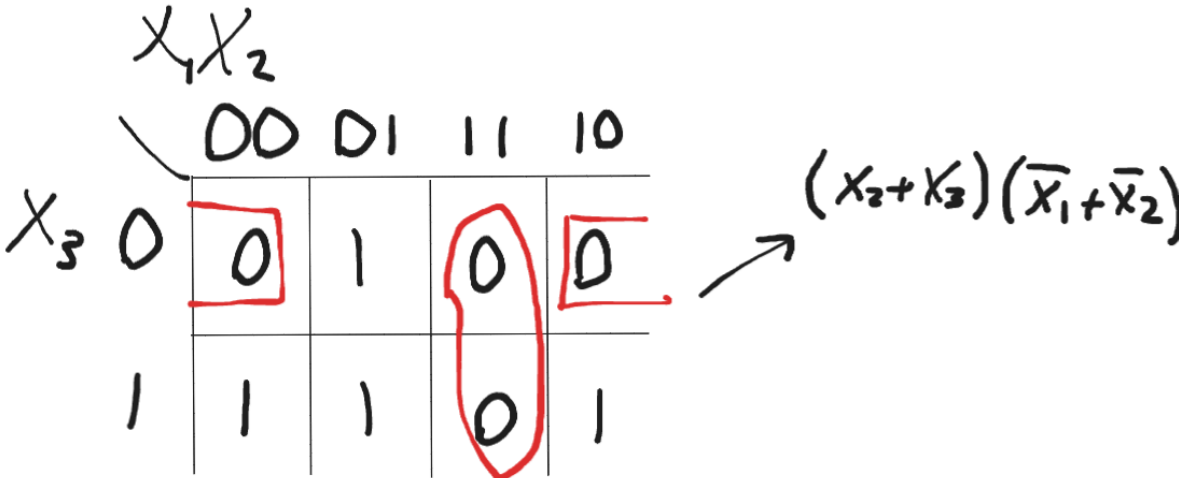
Our minimum-cost SOP is ~x1x2 + ~x2x3

**ii) Minimum-cost POS**

**f(x1, x2, x3) = Σm(1, 2, 3, 5) = ΠM(0, 4, 6, 7)**

f = (~x1 + ~x2 + ~x3)(x1 + ~x2 + ~x3)(x1 + x2 + ~x3)(x1 + x2 + x3)

Using a K-Map:



Our minimum-cost POS is (x2 + x3)(~x1 + ~x2)

## 4.2 Full Adder Implementation

Keep the sum part the same, as it is in 2 XORs:

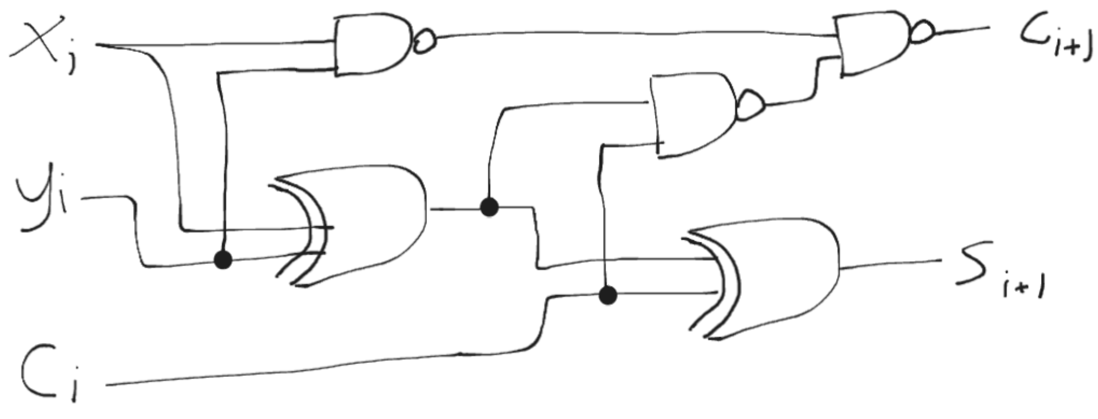
$$((x \oplus y) \oplus c) = s$$

Use DeMorgan’s Theorem for the carry out part:

$$c = xy + (x \oplus y)c$$

$$= \overline{\overline{xy + (x \oplus y)c}} = \overline{\overline{xy}\ \overline{((x \oplus y)c)}}$$

The NAND/XOR only circuit:



## 4.3 Parity Generator

**i) Is the circuit shown in section 7.2 generating an “even” or “odd” parity bit for the input 5-bit binary number?**

In this case, x = 1 if there is an odd amount of input 1s. According to the definition of an even or odd parity bit, this circuit is generating an even parity bit for the input 5-bit binary number.

**ii) What is the purpose of a parity generator? Consider that the parity bit is added to the input number to form a 6-bit binary number while sending from the transmitter to the receiver.**

<u>How it works:</u>

A 5-bit even parity generator will add that bit to the 5-bit binary number to form a 6-bit binary number. Doing this will ensure that the 6-bit binary number has an even number of 1s.
This is because if there is an odd amount of 1s, it will output 1 and add it to the number, making the number of 1s even. If there is an even amount of 1s, it will output 0 and add it to the number, keeping the number of 1s even.

Now for its purpose:
Parity generators are used for error detection. In the case we use an even parity generator, all 5-bit transmissions will be turned into 6-bit numbers (adding the parity bit) and transmitted to the receiver. If there isn't an even amount of 1s in any of these 6-bit numbers, then we know that we lost some data as some bits were flipped during transmission. It only helps for possible error detection and not error correction.

- Note it won't detect all errors as there is the case where it flips 2 bits for example.

# 2 - Lab Procedures

## 7.1 Full Adder

**i) Implement a full-adder circuit similar to Figure 3-2 but using only NAND and XOR gates**

Filename: full_adder_nand_xor.v

```
module full_adder_nand_xor(Co, S, xi, yi, ci);
   output Co, S;
   input xi, yi, ci;
   wire a, b, c;

   nand(a, xi, yi);

   xor(b, yi, xi);

   xor(S, b, ci);

   nand(c, b, ci);

   nand(Co, a, c);
endmodule
```

**ii) Provide your XDC file**

Filename: fulladderconstraints.xdc

```
## SWITCH (SW0) -> Port xi
set_property PACKAGE_PIN V17 [get_ports xi] ;
set_property IOSTANDARD LVCMOS33 [get_ports xi]
## SWITCH (SW1) -> Port yi
set_property PACKAGE_PIN V16 [get_ports yi] ;
set_property IOSTANDARD LVCMOS33 [get_ports yi]
## SWITCH (SW2) -> Port ci
set_property PACKAGE_PIN W16 [get_ports ci] ;
set_property IOSTANDARD LVCMOS33 [get_ports ci]
## LED (LED0) -> Port S
```

```
set_property PACKAGE_PIN U16 [get_ports S] ;
set_property IOSTANDARD LVCMOS33 [get_ports S]
## LED (LED1) -> Port Co
set_property PACKAGE_PIN E19 [get_ports Co] ;
set_property IOSTANDARD LVCMOS33 [get_ports Co]
```

**iii) Fill in the appropriate Truth Table and comment if the circuit is working as expected.**

| **xi** | **yi** | **ci** | **S** | **Co** |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

## 7.2 Parity Generator

**i) Realize the five-bit "parity" generator circuit shown in Figure 7-1 (below) on your Basys 3 Board.**

Filename: parity5_xor.v

```
// 5-bit even-parity generator f = x1 ^ x2 ^ x3 ^ x4 ^ x5
module parity5_xor(f, x1, x2, x3, x4, x5);
   output f;
   input x1, x2, x3, x4, x5;
   wire a, b, c;

   xor(a, x1, x2);
   xor(b, a, x3);
   xor(c, b, x4);
   xor(f, c, x5);
endmodule
```

**ii) Provide your XDC file**

Filename: paritygenconstraints.xdc

```
## SWITCH (SW0) -> Port x1
set_property PACKAGE_PIN V17 [get_ports x1] ;
set_property IOSTANDARD LVCMOS33 [get_ports x1]
```

```
## SWITCH (SW1) -> Port x2
set_property PACKAGE_PIN V16 [get_ports x2] ;
set_property IOSTANDARD LVCMOS33 [get_ports x2]
## SWITCH (SW2) -> Port x3
set_property PACKAGE_PIN W16 [get_ports x3] ;
set_property IOSTANDARD LVCMOS33 [get_ports x3]
## SWITCH (SW3) -> Port x4
set_property PACKAGE_PIN W17 [get_ports x4] ;
set_property IOSTANDARD LVCMOS33 [get_ports x4]
## SWITCH (SW4) -> Port x5
set_property PACKAGE_PIN W15 [get_ports x5] ;
set_property IOSTANDARD LVCMOS33 [get_ports x5]
## LED (LED0) -> Port f
set_property PACKAGE_PIN U16 [get_ports f] ;
set_property IOSTANDARD LVCMOS33 [get_ports f]
```

**iii) Fill in the appropriate Truth Table and comment if the circuit is working as expected.**

| **x1** | **x2** | **x3** | **x4** | **x5** | **f** |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

**iv) For the given parity generator, there is always a parity checker at the receiver end to detect errors in the received codes. Now, what will be the expression for the accompanying parity checker circuit? Explain it considering the output that the checker yields for a different combination of inputs.**
The expression for the accompanying parity checker circuit is f = x1 ⊕ x2 ⊕ x3 ⊕ x4 ⊕ x5.

- The output that the checker will yield is 1 if the amount of input 1s is odd, 0 if the amount of input 1s is even (explained above in pre-lab tasks)
- It is an even parity checker.

## 7.3 - NAND Equivalent Circuit:

Function: f(x1, x2, x3) = Σm(1, 3, 4, 6, 7)

**ii) Write the minimum-cost SOP expression for the function:**
The minimum-cost SOP expression for this function (from 4.1.2) f = ~x1x3 + x1~x3 + x1x2

**iii) Show schematic (can use 3-input NAND gate)**
Boolean algebra to convert it to NAND only:

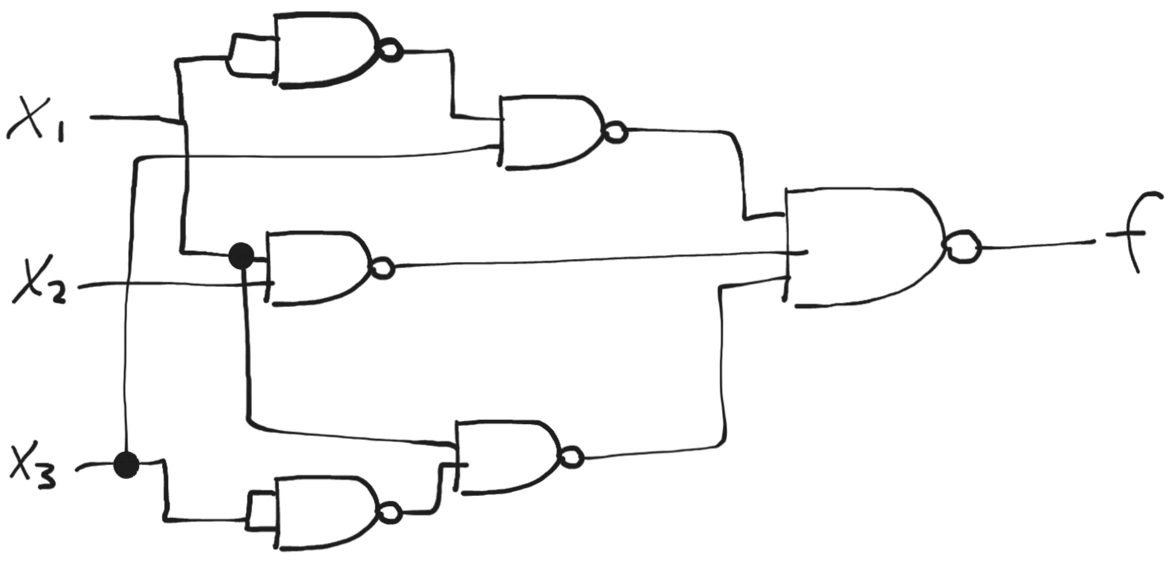
From 4.1.2 part ii)

$$f = \overline{x_1}x_3 + x_1\overline{x_3} + x_1x_2$$

$$= \overline{\overline{\overline{x_1}x_3 + x_1\overline{x_3} + x_1x_2}}$$

$$= \overline{(\overline{\overline{x_1}x_3})(\overline{x_1\overline{x_3}})(\overline{x_1x_2})}$$

We get this schematic:



**iv) Realize the circuit on your Basys 3 Board.**
Filename: f_nand_only.v

```
module f_nand_only(f, x1, x2, x3);
    output f;
    input x1, x2, x3;

    wire a, b, c, d, e;
    // a = ~x
    nand(a, x1, x1);
    // b = ~x3
    nand(b, x3, x3);
    // c = ~(x1x2)
    nand(c, x1, x2);
    // d = ~(~x1x3)
    nand(d, a, x3);
    // e = ~(x1~x3)
    nand(e, x1, b);

    nand(f, c, d, e);
endmodule
```

**v) Provide your XDC file**

Filename: f_nand_constraints.xdc

```
## SWITCH (SW0) -> Port x1
set_property PACKAGE_PIN V17 [get_ports x1] ;
set_property IOSTANDARD LVCMOS33 [get_ports x1]
## SWITCH (SW1) -> Port x2
set_property PACKAGE_PIN V16 [get_ports x2] ;
set_property IOSTANDARD LVCMOS33 [get_ports x2]
## SWITCH (SW2) -> Port x3
set_property PACKAGE_PIN W16 [get_ports x3] ;
set_property IOSTANDARD LVCMOS33 [get_ports x3]
## LED (LED0) -> Port f
set_property PACKAGE_PIN U16 [get_ports f] ;
set_property IOSTANDARD LVCMOS33 [get_ports f]
```

## 7.4 - NOR Equivalent Circuit:

Function: f(x1, x2, x3) = Σm(1, 3, 4, 6, 7)

**ii) Write the minimum-cost POS expression for the function:**

We know that f(x1, x2, x3) = Σm(1, 3, 4, 6, 7), which is equal to f = ΠM(0, 2, 5). This is what we solved in exercise 4.1.3!

From 4.1.3: The minimum-cost POS expression for the function is (x1 + x3)(~x1 + x2 + ~x3).

**iii) Show schematic (can use 3-input NOR gate)**

Boolean algebra to convert it to NOR only:

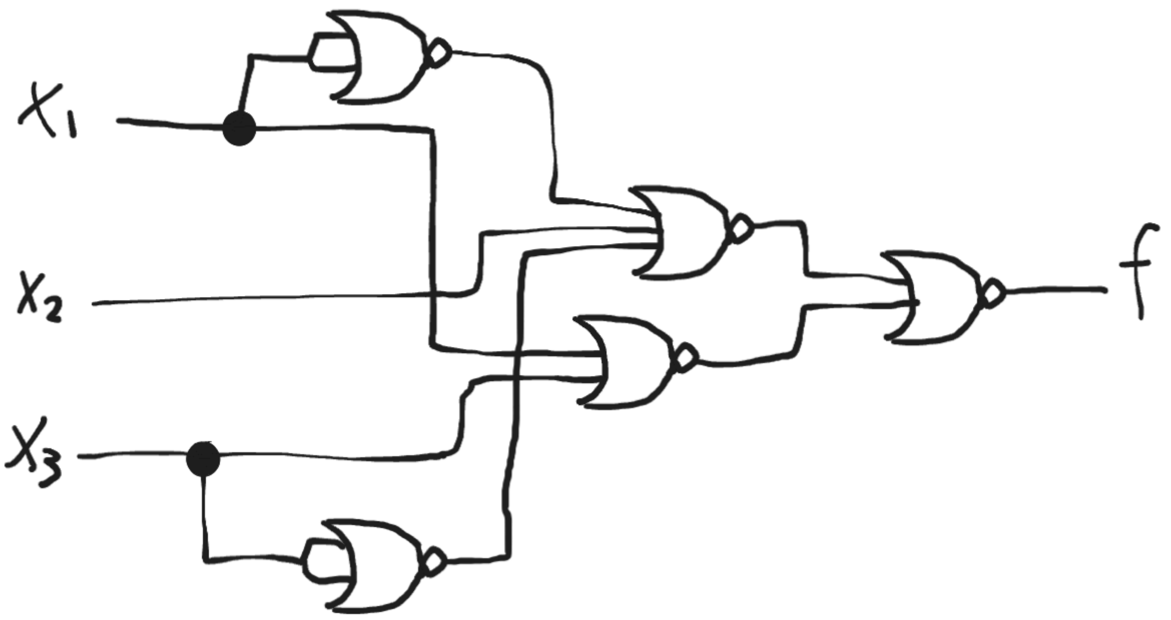
From 4.1.3:

$$f = (x_1 + x_3)(\overline{x_1} + x_2 + \overline{x_3})$$

$$= \overline{\overline{(x_1 + x_3)(\overline{x_1} + x_2 + \overline{x_3})}}$$

$$= \overline{\overline{(x_1 + x_3)} + \overline{(\overline{x_1} + x_2 + \overline{x_3})}}$$

We get this schematic:



**iv) Realise the circuit on your Basys 3 Board.**

<u>Filename: f_nor_only.v</u>

```
module f_nor_only(f, x1, x2, x3);
   output f;
   input x1, x2, x3;
   wire a, b, c, d;

   // a = ~x1
   nor(a, x1, x1);
```

```
    // b = ~x3
    nor(b, x3, x3);
    // c = ~(x1 + x3)
    nor(c, x1, x3);
    // d = ~(~x1 + x2 + ~x3)
    nor(d, a, x2, b);

    nor(f, c, d);
endmodule
```

**v) Provide your XDC file**

<u>Filename: f_nor_constraints.xdc</u>

```
## SWITCH (SW0) -> Port x1
set_property PACKAGE_PIN V17 [get_ports x1] ;
set_property IOSTANDARD LVCMOS33 [get_ports x1]
## SWITCH (SW1) -> Port x2
set_property PACKAGE_PIN V16 [get_ports x2] ;
set_property IOSTANDARD LVCMOS33 [get_ports x2]
## SWITCH (SW2) -> Port x3
set_property PACKAGE_PIN W16 [get_ports x3] ;
set_property IOSTANDARD LVCMOS33 [get_ports x3]
## LED (LED0) -> Port f
set_property PACKAGE_PIN U16 [get_ports f] ;
set_property IOSTANDARD LVCMOS33 [get_ports f]
```