**Name: Shishir Ghorashainee**

CSC 362 Homework #3

Due: Monday, September 16

**Word process all answers** (figures (1a&c, 4b, 6) may be hand drawn) and **show all work**. Make sure your name appears at the top of the assignment.

1. We know that 𝑋 ⨁ 𝑌is actually . Given F below, answer the questions that follow. Remember that ⨁ has the *lowest* operator precedence.
   1. Draw the circuit for F using an XOR gate for the ⨁ operation.

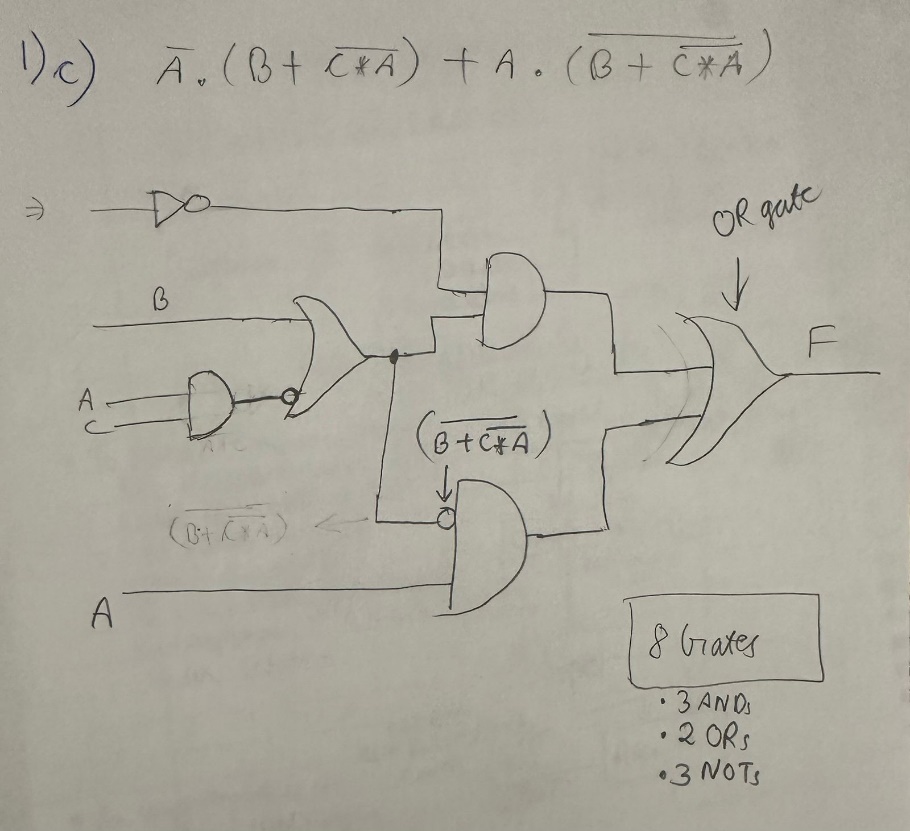
A white paper with black writing on it

Description automatically generated

* 1. Rewrite the expression, replacing the ⨁ with the equivalent expression using ANDs/ORs/NOTs. Do not simplify this expression in any way.

= A’ \* (B+ (C\*A)’) + A \* ((B+ (C\*A)’)’

* 1. Draw circuit for the revised expression in part b.



* 1. Count the total number of gates from part a and compare them to the total number of gates from part c.

= **There are a total of 4 gates in part a and 8 gates in part c.**

1. Instead of using a negator and adder to perform subtraction, we can build a subtractor circuit using the same type of logic that we used to build the adder. The subtractor circuit computes the difference and borrow out. For an n-bit subtractor, the leftmost subtraction can use a half subtractor while the rest of the subtractions will use a full subtractor. The half subtractor has two inputs (the bit from X and from Y) and two outputs, the difference and a borrow out. The full subtractor has three inputs (a borrow in) and the two outputs. See the truth tables below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Xi | Yi | Bi | Diffi | Bi+1 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Xi | Yi | Diffi | Bi+1 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

1. Explain the logic of the two truth tables (that is, explain why the outputs are as shown).

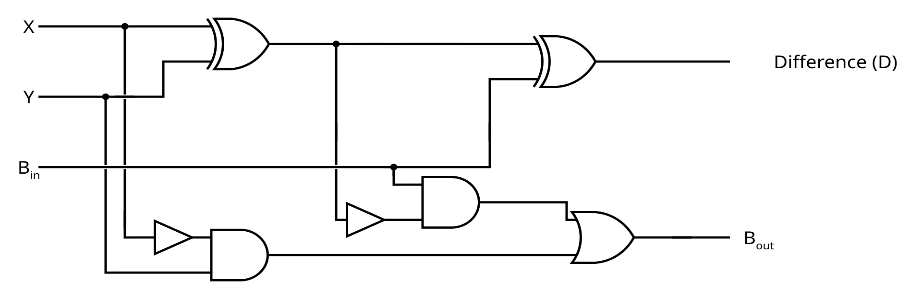
**Table (I):**

In the given first Half Subtracter (two bits) table, we need to take the difference of two bits, Xi and Yi. If Xi is smaller than Yi, then we need to take a borrow. For example, in the 3rd row, we have (0 - 1), where Xi is 0 and Yi is 1, which cannot be done without taking a borrow. We take a borrow from Xi, making 0 become 2, and then the difference is taken (2 - 1 = 1). We write 1 in the Diff and 1 in the Borrow Out bit. However, when we do the difference operation (1 - 0), where Xi is greater than Yi, we don’t need to take a borrow. The difference is 1, and the borrow remains 0.

**Table (II):**

In the given full subtractor (3 bits) table, we take the difference of two bits, Xi and Yi, along with an additional input Bi from the previous result. If Xi is smaller than the sum of Yi and Bi, a borrow is needed. For example, in the 4th row, we have Xi = 0, Yi = 1, and Bi = 1, giving us (0 - 1 - 1). Since we cannot subtract 2 from 0 without borrowing, we take a borrow, and the difference becomes 0, and the borrow out becomes 1.

However, if Xi is greater than or equal to the sum of Yi and Bi, no borrow is needed in cases like this. For instance, in the 5th row, Xi = 1, Yi = 0, and Bi = 0, giving us (1 - 0 - 0), the difference is 1, and so borrow out remains 0.

1. The circuit below is the implementation of a full subtractor. Like the full adder, it consists of two half subtractors plus an OR gate. Explain how it works.

= The given circuit has two half subtractors with an OR gate. When the inputs are given, X XOR Y is calculated and sent to another XOR gate with the Borrow-in input and after passing through that 2nd XOR gate, we get the result for the Difference(D).

Also, another input from X is sent through a NOT gate and is ANDed with Y. This AND circuit are ORed with another circuit from the 1st XOR (negated) and ANDed with Borrow-in input. After passing through the OR gate, we get the result for the Borrow Out.

1. Explain the changes you would make to adapt the unsigned multiplication circuit (slide 19 from the chapter 3 power point notes) into an unsigned division circuit.

**We use this Algorithm for Unsigned Division :**

A = All 0s

Q = Numerator

M = Denominator

Repeat n times:

1. Left Shift A, Q
2. If A >= M, then A=A-M, Q0 = 1

Else

Q0 = 0

( Where, A = Remainder and Q = Quotient )

For building the **unsigned division circuit**, we add a new circuit called Comparator circuit to that pulls out 3 bits (positive, negative and Zero flags respectively), and a subtractor circuit to perfor (A-M). But, instead of right shift, we do a left shift. Also, we don’t need a carry bit.

A paper with text on it

Description automatically generated

1. A jukebox is controlled by a circuit to determine if it should play a selected song. The circuit outputs a 1 under the following conditions, all of which must be true.

* The jukebox is in active operating mode.
* A legitimate song must have been selected. Songs are numbered 0-31 but the input is a number from 0-99, so numbers 32-99 are not legal.
* The user must have inserted payment.
* The record containing the song and the turntable must be free of obstruction.

The circuit receives input as follow, outputting a 1 if the above conditions are true.

* Operating mode is denoted by variables m1m0 and is 00 (off mode), 01 (maintenance mode), 10 (active mode). 11 is not used.
* The song selection is converted from a 2-digit input into 7-bit binary number denoted by variables i6i5i4i3i2i1i0 where 0-31 is a legal number.
* Payment is one of credit card, coins or tokens. These are indicated through three variables: CC, CS, TS. If any of these are 1, the payment condition is fulfilled.
* Four sensors detect obstructions: R (record rack), A (moving arm), T (turntable) and N (turntable needle). The condition requires that there are no obstructions.
  1. Derive the Boolean expression for the circuit. Remember that the Boolean expression consists of Boolean variables (m1, m0, i6, ..., i0, CC, CS, TS, R, A, T, N) and Boolean operators (AND, OR, NOT, XOR), not notation like = 1 or <= 31. Provide the *simplified* version of the derived expression.
* The jukebox is in active operating mode.

**= It’s in active operating mode if: m1 \* ~m2**

* A legitimate song must have been selected. Songs are numbered 0-31 but the input is a number from 0-99, so numbers 32-99 are not legal.

**= Given the range for the songs that can be played legally are from 0-31, or in 7-bits binary, (000000 – 0011111). So, if we have i6=0 and i5= 0, then it won’t be legal and gets rejected as the number should not be above 31.**

**So, the condition is expressed as: ~i6 \* ~i5**

* The user must have inserted payment.

**= For the inserted payment, it must be taken through Credit Cards (CC), Coins (Coins), or (Tokens). This is expressed as: CC + CS + TS**

* The record containing the song, and the turntable must be free of obstruction.

**= ~ R \* ~A \* ~T \* ~N**

ANDing all the given conditions to produce an output equal to 1, we get this expression:

(**m1 \* ~m2 ) \* ( ~i6 \* ~i5 ) \* ( CC + CS + TS ) \* ( ~ R \* ~A \* ~T \* ~N )**

* 1. Draw the circuit from part a.

A drawing on a white board

Description automatically generated

1. Answer the following questions about the MUX.
   1. Part of the MUX is a decoder so that each AND gate operates as a “number recognizer”. What is a number recognizer and how does the AND gate recognize a number.

= A number recognizer is an AND gate circuit that outputs a 1 from the binary bit inputs. If we expect any inputs to be zero, then we negate those inputs. For instance, if the input is 11001, then we negate I1 and I2 and if other inputs are expected to be 1. Then after it passes through the AND gate circuit and if it’s true, we get 1.

* 1. Why does a MUX have an OR gate? Why does it have only a single OR gate?

= A multiplexer (MUX) has a single OR gate, which accumulates the results from the multiple AND gates, each corresponding to different input lines. The OR gate ensures that only the active input is passed as the final output. Since only one input is chosen at a time, a single OR gate is enough to combine and output the selected signal.

* 1. Explain in detail, but do not draw, a MUX that has 3 selector bits and 8 inputs. You can refer to these as s2s1s0 and i7,i6,i5,i4,i3,i2,i1,i0 if needed.

= In the given MUX with 3 selector bits (s2, s1, s0) and 8 inputs (i7, i6, i5, i4, i3, i2, i1, i0), each of the 8 AND gates correspond to one input and is activated by a specific combination of the selector bits. For instance, if the selector bit is 011, it means that input i3 is selected. Then, the AND gates are passed through a single OR gate, which produces the final output.

1. Draw the following circuits using S-R flip-flops. You do not need to draw either enable gates or the internal parts of the flip-flops (just the S, R, Q, ~Q labels in the flip-flops).
   1. 5-bit left rotate

A paper with a diagram

Description automatically generated

* 1. 5-bit arithmetic right shift

A drawing of a diagram

Description automatically generated

1. We want to expand the 2-bit ALU from the textbook (and slide 27 of the chapter 3 power point notes) to be a 16-bit ALU but still with the same four functions. Answer the following.
   1. How many columns of circuitry will the ALU have; of these, how many will contain a full adder and how many will contain a half adder?

= The ALU has 16 columns of circuitry. In this, 1 column contains half adder, and the remaining 15 columns contain full adders.

* 1. How many total OR gates will the ALU have?

= **total number of OR gates = 16 + 16 + 15**

**= 47**

* 1. How many selector bits will the decoder portion of the ALU have?

= 2 Selector bits in the decoder portion of the ALU.