CSCI 355 - Lab 1 Report

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

1 - Pre-Lab Answers

4.1 - 7400 Series

Go to wikipedia.org and look up “7400 series”. Review the article and answer the following questions:

i. What is the difference between the 7400 series and the 5400 series?

They are both families of transistor-trainsistor logic (TTL) type ICs.

The difference between them is that the 7400 series is for commercial use and the 5400 series is military grade. The 7400 series’ temperature range is from 0 to 70 C, while the 5400 series’ temperature range is from -55 to 125 C.

ii. Integrated circuits of the “HC” device family are popular nowadays. What does “HC” stand for??

HC stands for high-speed CMOS. These devices use CMOS technology instead of TTL. CMOS is better than TTL because it has lower power consumption and is faster (most of the time).

iii. What is the maximum propagation delay (“TP (max)”) for the “HC” device family?

For the 74HC family, the maximum propagation delay is 8 ns.

4.2 - Dual In-Line Package

74XX devices are configured in a “dual in-line package” (DIP). Go to wikipedia.org and look up “dual inline package”. Review the article and answer the following questions:

i. When was the dual-inline format invented and by whom?

The dual-inline format was invented in 1964 by Don Fores, Rex Rice, and Bryant Rogers

ii. Which is PIN #1? How are the rest of the pins numbered?

Pin 1 is at the top left corner of the device. The pins are then numbered consecutively in a counter-clockwise order (or a U shape).

4.3 - Datasheets

A datasheet is a document that summarizes the performance and other technical characteristics of a device. Go to wikipedia.org and look up “list of 7400 series integrated circuits”. From the list, answer the following questions:

i. What is a 7474 IC?

A 7474 IC is a dual D positive edge triggered flip-flop, asynchronous clear & preset, Q & /Q outputs. It has 14 pins.

ii. What is the number for a quad 2-input XOR gate?

The number for a quad 2-input XOR gate is 74x86 74x136, or 74x386.

iii. Download and review the "datasheet" for the 7400 - quad 2-input NAND gate, and 7402 - quad 2-input NOR gate

I downloaded and reviewed them.

4.4 - Number Conversion

i. Show a complete process of converting a binary number (1011.101)2 to decimal.

Go from left to right, and add up the numbers based on the digit position (powers of 2).

1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 + 1 * 2^-1 + 0 * 2^-2 + 1 * 2^-3
= 8 + 0 + 2 + 1 + 1/2 + 0 + ⅛
= 11 + 0.5 + 0.125 = (11.625)10

ii. Show a complete process of converting gray code (1011101) to normal binary.

Go from left to right, copy the left-most bit first, and compare each bit of result to the next bit of the binary number (XOR operation).

Gray Code: 1011101

Copy first bit: 1
XOR of 1 (first of binary) and 0 (second of gray code) = 1
Binary: 11
XOR of 1 and 1 = 0
Binary: 110
XOR of 0 and 1 = 1
Binary: 1101
XOR of 1 and 1 = 0
Binary: 11010
XOR of 0 and 0 = 0
Binary: 110100
XOR of 0 and 1 = 1
Binary: 1101001

2 - Verilog Source Listings

File: and_from_nand.v

```
module and_from_nand(y, a, b);
   output y;
   input a, b;
   wire n;
```

```
    nand (n, a, b);
    nand (y, n, n);
endmodule
```

File: and_from_nor.v

```
module and_from_nor(y, a, b);
    output y;
    input a, b;
    wire na, nb;

    // ~a and ~b using NOR with itself
    nor (na, a, a);
    nor (nb, b, b);
    // y = ~(~a + ~b)
    nor (y, na, nb);;
endmodule
```

File: or_from_nand.v

```
module or_from_nand(y, a, b);
    output y;
    input a, b;
    wire na, nb;
    // ~a
    nand(na, a, a);
    // ~b
    nand(nb, b, b);

    // ~(~a * ~b)
    nand(y, na, nb);
endmodule
```

File: or_from_nor.v

```
module or_from_nor(y, a, b);
    output y;
    input a, b;
    wire n;
    // ~(a+b)
    nor(n, a, b);
    // ~(~a * ~b)
    nor(y, n, n);
endmodule
```

3 - XDC Excerpt

File: lab1_constraints.xdc

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

```
set_property IOSTANDARD LVCMOS33 [get_ports a]
## SWITCH (SW1) -> Port b
set_property PACKAGE_PIN V16 [get_ports b] ;# SW1
set_property IOSTANDARD LVCMOS33 [get_ports b]
## LED (LED0) -> Port y
set_property PACKAGE_PIN U16 [get_ports y] ;# LED0
set_property IOSTANDARD LVCMOS33 [get_ports y]
```

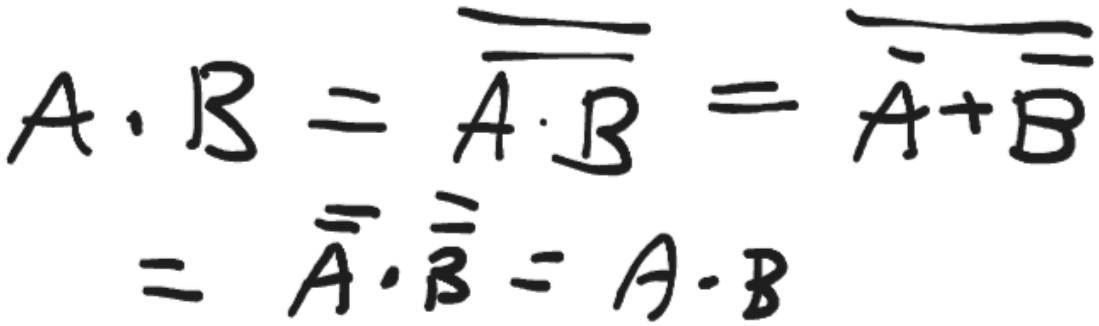
4 - Truth Tables

All truth tables were filled from hardware measurements.

6.1.1c AND Gate using NAND truth table:

a	b	n	y
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

6.1.2a AND Gate using NOR Boolean algebra



6.1.2c AND Gate using NOR truth table:

a	b	NOT a	NOT b	y
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

6.2.1b OR Gate using NOR truth table:

a	b	n	y
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

6.2.2a OR Gate using NAND Boolean Algebra

$$A + B = \overline{\overline{A+B}} = \overline{\bar{A} \cdot \bar{B}}$$

$$= \bar{\bar{A}} + \bar{\bar{B}} = A + B$$

6.2.2b OR Gate using NAND truth table:

a	b	NOT a	NOT b	y
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

5 - Short Discussion

Compare NAND-only vs NOR-only realizations for AND/OR.
Briefly relate to De Morgan's transformations.
Note any gotchas with constraints or port naming.

The NAND-only realization was more complicated for the OR gate than the AND gate. This is because applying De Morgan's transformations allows an AND gate to be expressed in terms of NAND operations with fewer gate levels. The NOR-only realization was more complicated for the AND gate than the OR gate. This is because applying De Morgan's transformations allows an OR gate to be expressed in terms of NOR operations with fewer

gate levels. Overall, NAND naturally complements AND better and NOR naturally complements OR better, as you'd expect.

For gotchas with constraints or port naming, most of my difficulty has been with the similarities between the names of the pins (U16 vs V16), it's easy to make a typo and accidentally assign ports to the wrong pins.