Lab 1 Report

George P. Burdell
ECE 2031 L02

01 January 1970

Contents

1 Unformatted Results

Lab Report Grading Categories

Technical Accuracy (80 points)

Each result in the lab report, including its caption, is worth a number of points for its technical information. The "Lab Report Content Requirements" document includes the point values associated with each result.

Points for technical accuracy are based on how much of the result is present and correct. For example, a simulation that only covers half of the required test cases will only earn half of that result's possible value, and a caption with technical errors will be penalized according to the severity of the errors.

For labs where later results are built upon earlier results, if technical errors are propagated, point deductions will only apply to the first result, not on each result where the error is present. For example, if a K-map is solved incorrectly, any resulting Boolean expression and gate-level schematic will also be incorrect, but points will only be deducted the first time the error occurs as long as each step was done correctly based on the error.

Check-offs with no required associated result are graded based solely on the presence of the check-off.

Information Effectiveness (50 points)

Technical communication has a strong emphasis on the clear presentation of information. For ECE 2031 lab reports, this category requires the following:

- **Clear and concise captions**. Captions should explain the content and context of the figure; see later sections of this document for additional guidelines. Ambiguity, rambling, colloquial or extravagant language, confusing sentence structure, poor word-choice, spelling and grammar errors, and irrelevant information all reduce the effectiveness of your caption and will result in penalties.
- **Clear and concise images**. This the visual extension of the previous point, and requires (among other things) clear and logical layout of schematics and diagrams, appropriate image sizes and aspect ratios, and removal of unnecessary image elements and areas. Some common errors are described later.

Because each lab report contains a different number of results, the points in this category are distributed evenly between each formatted result. Each ineffective aspect of a result reduces its points by 1/3 (33%).

For grading simplicity, the score for this category will be calculated by summing the number of ineffective elements (at most three per result) and calculating the overall result, rounded to the nearest integer.

For example: if a lab report consists of six results and has four effectiveness errors, it would earn

$$50 * \left[1 - \frac{4}{6 * 3}\right] = 39\ points$$

Document Formatting (20 points)

All technical documents have some amount of formatting specification. The formatting specification for ECE 2031 lab reports follows a subset of IEEE formatting guidelines. All formatting requirements for lab reports are detailed later in this document.

Formatting errors count off 5 points each, up to the maximum 20 points of this grading category.

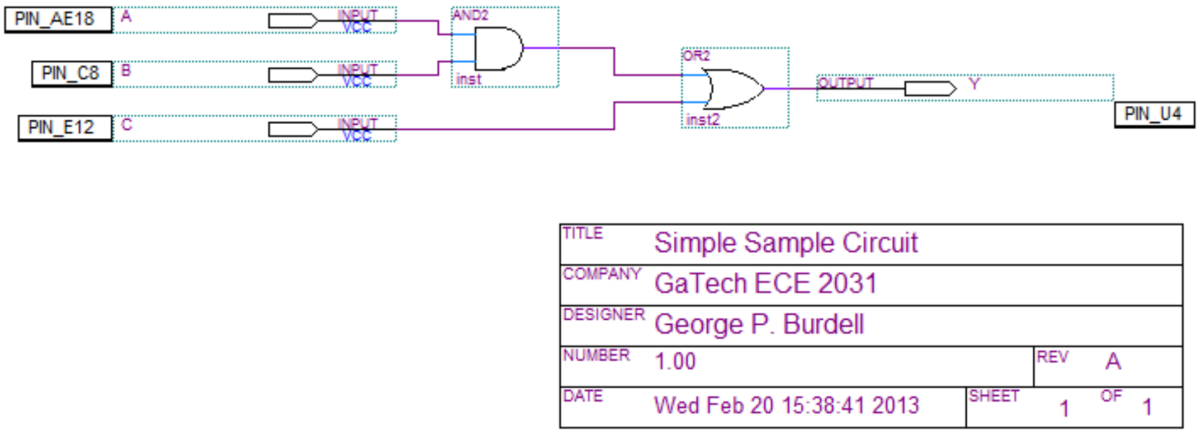

Figure 1. Schematic with FPGA pin assignments of circuit implementing $Y = AB + C$.

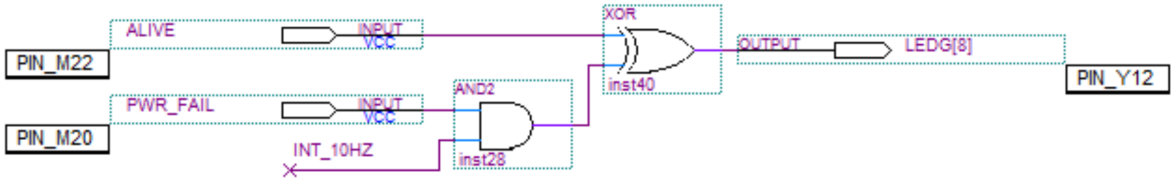

Figure 2. Circuit to light LED G8 when robot motors are enabled, and flash it at 10Hz when robot battery voltage is below minimum threshold.

2 Results

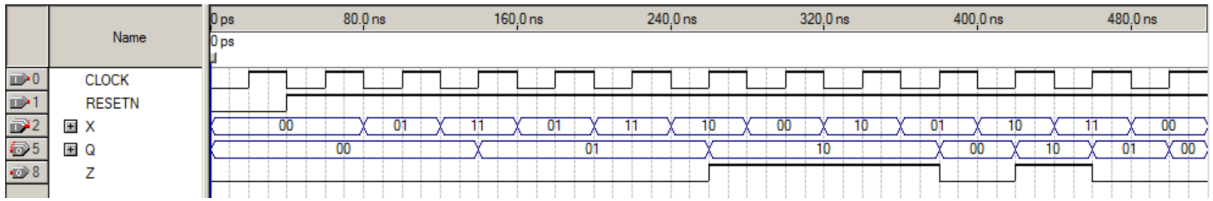

Figure 3. Functional simulation waveform of state machine with three states, 2-bit input X, and 1-bit output Z. The input vector provides 100% coverage of state transitions to prove correct behavior of the state machine, including assertion of Z in state '10 This figure should be properly rotated for both digital viewing and printing.

```
-- ORGATE.VHD (VHDL)
-- This code produces a negative-logic OR circuit
-- George P. Burdell
-- ECE2031 L01
-- 01/31/2009

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;


ENTITY orgate IS
PORT(
        PB1, PB2 : IN STD_LOGIC;
        LED : OUT STD_LOGIC);
END orgate;

ARCHITECTURE a OF orgate IS
BEGIN
        LED <= NOT(NOT PB1 OR NOT PB2);
END a;
```

Figure 4. VHDL code representing a three state, two input FSM satisfying the assigned state transitions, which outputs high only in state C.

Table 1
Truth Table for $Y = \overline{ABC}$ With Highlights

ABC	Y
000	1
001	1
010	1
011	1
100	1
101	1
110	1
111	0

Table 2
Truth Table for $Y = \overline{ABCD}$ From CSV

$ABCD$	Y
0000	1
0001	1
0010	1
0011	1
0100	1
0101	1
0110	1
0111	1
1000	1
1001	1
1010	1
1011	1
1100	1
1101	1
1110	1
1111	0

Appendix A
The first one

```
--PS2KEY.VHD
--Basic PS/2 keyboard interface with clock filtering
--George P. Burdell
--ECE 2031 L01
--01/31/2009

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY keyboard IS
        PORT(
                keyboard_clk, keyboard_data,
                        clock_25Mhz, reset, read : IN STD_LOGIC;
                scan_code : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                scan_ready : OUT STD_LOGIC);
END keyboard;


ARCHITECTURE a OF keyboard IS
        SIGNAL INCNT : std_logic_vector(3 downto 0);
        SIGNAL SHIFTIN : std_logic_vector(8 downto 0);
        SIGNAL READ_CHAR : std_logic;
        SIGNAL INFLAG, ready_set : std_logic;
        SIGNAL keyboard_clk_filtered : std_logic;
        SIGNAL filter : std_logic_vector(7 downto 0);
BEGIN


PROCESS (read, ready_set)
BEGIN
        IF read = '1' THEN scan_ready <= '0';
        ELSIF ready_set'EVENT and ready_set = '1' THEN
        scan_ready <= '1';
        END IF;
END PROCESS;


--This process filters the raw clock signal coming from the
--keyboard using a shift register and two AND gates
Clock_filter: PROCESS
BEGIN
        WAIT UNTIL clock_25Mhz'EVENT AND clock_25Mhz= '1';
        filter (6 DOWNTO 0) <= filter(7 DOWNTO 1) ;
        filter(7) <= keyboard_clk;
        IF filter = "11111111" THEN keyboard_clk_filtered <= '1';
        ELSIF filter= "00000000" THEN keyboard_clk_filtered <= '0';
        END IF;
END PROCESS Clock_filter;

--This process reads in serial data coming from the terminal
PROCESS
BEGIN
WAIT UNTIL (KEYBOARD_CLK_filtered'EVENT AND KEYBOARD_CLK_filtered='1');
IF RESET='1' THEN
        INCNT <= "0000";
        READ_CHAR <= '0';
ELSE
```

```
        IF KEYBOARD_DATA='0' AND READ_CHAR='0' THEN
                READ_CHAR<= '1';
                ready_set<= '0';
        ELSE
                -- Shift in next 8 data bits to assemble a scan code
                IF READ_CHAR = '1' THEN
                        IF INCNT < "1001" THEN
                                INCNT <= INCNT + 1;
                                SHIFTIN(7 DOWNTO 0) <= SHIFTIN(8 DOWNTO 1);
                                SHIFTIN(8) <= KEYBOARD_DATA;
                                ready_set <= '0';
                                -- End of scan code character,
                                -- so set flags and exit loop
                        ELSE
                                scan_code <= SHIFTIN(7 DOWNTO 0);
                                READ_CHAR <='0';
                                ready_set <= '1';
                                INCNT <= "0000";
                        END IF;
                END IF;
        END IF;
END IF;
END PROCESS;
END a;
```

Appendix B
The second one

```
--PS2KEY.VHD
--Basic PS/2 keyboard interface with clock filtering
--George P. Burdell
--ECE 2031 L01
--01/31/2009

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY keyboard IS
        PORT(
                keyboard_clk, keyboard_data,
                        clock_25Mhz, reset, read : IN STD_LOGIC;
                scan_code : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
                scan_ready : OUT STD_LOGIC);
END keyboard;


ARCHITECTURE a OF keyboard IS
        SIGNAL INCNT : std_logic_vector(3 downto 0);
        SIGNAL SHIFTIN : std_logic_vector(8 downto 0);
        SIGNAL READ_CHAR : std_logic;
        SIGNAL INFLAG, ready_set : std_logic;
        SIGNAL keyboard_clk_filtered : std_logic;
        SIGNAL filter : std_logic_vector(7 downto 0);
BEGIN


PROCESS (read, ready_set)
BEGIN
        IF read = '1' THEN scan_ready <= '0';
        ELSIF ready_set'EVENT and ready_set = '1' THEN
        scan_ready <= '1';
        END IF;
END PROCESS;


--This process filters the raw clock signal coming from the
--keyboard using a shift register and two AND gates
Clock_filter: PROCESS
BEGIN
        WAIT UNTIL clock_25Mhz'EVENT AND clock_25Mhz= '1';
        filter (6 DOWNTO 0) <= filter(7 DOWNTO 1) ;
        filter(7) <= keyboard_clk;
        IF filter = "11111111" THEN keyboard_clk_filtered <= '1';
        ELSIF filter= "00000000" THEN keyboard_clk_filtered <= '0';
        END IF;
END PROCESS Clock_filter;

--This process reads in serial data coming from the terminal
PROCESS
BEGIN
WAIT UNTIL (KEYBOARD_CLK_filtered'EVENT AND KEYBOARD_CLK_filtered='1');
IF RESET='1' THEN
        INCNT <= "0000";
        READ_CHAR <= '0';
ELSE
```

```
        IF KEYBOARD_DATA='0' AND READ_CHAR='0' THEN
                READ_CHAR<= '1';
                ready_set<= '0';
        ELSE
                -- Shift in next 8 data bits to assemble a scan code
                IF READ_CHAR = '1' THEN
                        IF INCNT < "1001" THEN
                                INCNT <= INCNT + 1;
                                SHIFTIN(7 DOWNTO 0) <= SHIFTIN(8 DOWNTO 1);
                                SHIFTIN(8) <= KEYBOARD_DATA;
                                ready_set <= '0';
                                -- End of scan code character,
                                -- so set flags and exit loop
                        ELSE
                                scan_code <= SHIFTIN(7 DOWNTO 0);
                                READ_CHAR <='0';
                                ready_set <= '1';
                                INCNT <= "0000";
                        END IF;
                END IF;
        END IF;
END IF;
END PROCESS;
END a;
```