EE 260 — Final Exam

Fall 2001

Name: ________________________________

login id: ________________________________

Team Name: ________________________________

December 2, 2002

This is an CLOSED book, CLOSED notes, closed neighbor, closed teammate exam. You may have ONE 8.5x11 sheet of crib notes during the exam. Partial credit is available, so where ever possible, Show Your Work. This is particularly important as some of the problems on this exam require multiple steps to arrive at a solution. You should show all of the steps – answers alone without showing how they were derived will not be sufficient.

The exam consists of two Parts:

- **Part A**: consists of the first three problems. Show your work and solutions on these pages. You will have 90 minutes to complete this part doing your OWN work. You must turn in Part A to begin Part B. Part A consists of 9 pages including this cover sheet.

- **Part B**: consists of problem 4 which you will do as a team. You will have 30 minutes to complete Part B. Part B consists of 3 pages including a cover sheet and two answer sheets.

<table>
<thead>
<tr>
<th></th>
<th>Points</th>
<th>Your Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part A</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>Part B Team</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Part B Individual</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

1
1. (25 Points). For the Midterm 2 Makeup problem, you designed a controller for a system that counted how many people were in a secure area by detecting when someone entered or exited the area through the single door. In that problem, the light barrier sensors produced a one clock cycle true pulse when the light beam was broken, no matter how long someone was blocking the sensor. Unfortunately, the parts that did that are no longer available. Instead, you will have to build the system with light barrier sensors that produce a true signal as long as the light source is blocked from the detector. Your job here is to redesign the controller for these new sensors. To refresh your memory, below is the spec for the system. **The change to the spec for these new sensors is shown in BOLD.**

You would like to build a system that keeps track of access to a secure area that has only one door. The system consists of a controller, a counter, and some sensors. Someone else will design the counter and sensor modules; you will design the controller. Your circuit will produce two signals: UP, which will be true for at most one clock cycle when someone enters the area; and DWN, which will true for at most one clock cycle when someone exits the area.

The inputs to the controller are two light barrier sensors, INNER and OUTER. A light barrier sensor consists of a light source shining across the doorway opening to a detector on the other side. If no one is in the doorway, the light shines on the detector, and the detector output is false. **When someone enters the doorway, they break the light beam and the output of the detector will be true until they pass through the beam and the light shines again on the detector.** The sensors produce only positive logic signals. There are two light barrier sensors: one on the inside of the doorway (INNER) and one on the outside of the doorway (OUTER). The sensors are 2 inches apart. **The clock is running at 10 K Hz.**

Your job is to design this new controller. (Hint: it can still be done with 3 states; think about how the sensor values change as someone passes through the door).

Implement the state register with JK FF’s. Use a minimum number of 2-input device chips (‘00 or ‘02) to implement the next state and output logic. **Show your work for all of the steps in this design.**
2. In class we derived iterative circuits for comparing two n-bit patterns for equality. In some applications, it is ok to have two n-bit patterns that may differ by at most one bit. (Such applications may use error detection/correction schemes that can correct single bit errors). So we would like to design an iterative circuit which has two n-bit inputs, A and B, and one output, Z, which is true iff the 2 n-bit patterns are identical or differ by at most one bit.

(a) (10 Points) Derive a minimum unit cell for an iterative in space approach which can be used to solve the above spec. Implement this cell using any gates you need. Also show any “end cap” cell needed to produce the output, Z.
(b) (10 Points) Derive the cells needed to solve this task using a hierarchical approach.
Assume we will implement the above unit cell using a technology with the following properties:

- For a 2-input device, $\tau_{PLH}$ is 3ns
- For a 2-input device, $\tau_{PHL}$ is 1ns
- For each additional input to a device, the propagation delays above increase by 2 ns.

(c) (6 Points) For the iterative in space approach:

i. What is the propagation delay (in ns) for your cell from A or B to the output?

ii. What is the propagation delay for your cell from the stage inputs to the output?

iii. In general, how long will it take to determine Z for 2 n-bit patterns (as a function of n)?

(d) (4 Points) How long will this hierarchical approach take for two 8 bit patterns?
3. (25 Points). To try to earn more money for your Christmas shopping, you and your team decide to design and build a Christmas Tree ornament. (It is so cool, EVERYONE will want one, and you think you’ll make big bucks :-)) The ornament is a box with LED’s on the front, and when you plug it in, it displays a Christmas tree pattern on the LEDs.

One of your teammates is building the box and other circuits to make the display work. You decide to use your EE 260 Computer development kit to implement the logic to drive the display. When your development kit arrived the other day, you noticed they sent you the EE 260 II Development Kit, which includes the next generation CPU. The Instruction Set for this improved CPU is given as the last page of this exam. Looking it over, you notice 2 new instructions, LDX (Load Indexed) and STX (Store Indexed). These instructions add a new addressing mode to the architecture called Memory Indexed, which computes the memory address of the operand by adding the contents of the B register to the address specified in the instruction. For example, if the B register has the value 2 in it, then the instruction:

```
LDX 0x80
```

will load the contents of memory address 0x82 (0x80 + GPR(B)) into the A register. This new addressing mode allows array indexing in C! (the B register contains the array index, and the addr in the instruction is the address of the beginning of the array in memory).

To make your job easier, another of your teammates has already written the code for the EE 260 II in C as shown on the next page. Your job is to compile this code into EE 260 II assembly code.
#define OUT *(0x40)
#define MAX 6

main()
{   int i;
    int temp;
    int lights[MAX] = { 0x40, 0x1C, 0x3E,
                      0x7F, 0x1C, 0x1C };

    while(1)
    {   temp = lights[0];
        OUT = temp;
        for( i=0; i != MAX-1; i++ )
            lights[i] = lights[i+1];
        lights[i] = temp;
    }
}
ISET
PART B
Directions

4. On the last problem of Part A, you might have realized that the program there was just shifting the values in the array one position per iteration of the outer loop. (Actually, it was rotating the values, but rotation is just a special case of shifting). Someone on your team had the idea that maybe using the EE 260 II kit was overkill for your ornament. Maybe the same thing can be done using shift registers. As a team, you decide to investigate this possible solution.

However, to allow for future alternate versions of your ornament, you decide that whatever shift registers you use, they MUST be both Universal and Bi-directional. Recall, the function table for a ’194 (4-bit Universal Bi-directional Shift Register):

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Shift Left</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Shift Right</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Load</td>
</tr>
</tbody>
</table>

Besides the mode control inputs, and the 4 outputs ($Q_D, Q_C, Q_B, Q_A$), these devices have 4 parallel load inputs (D, C, B, and A), and Shift Left Serial input (SL), and Shift Right Serial input (SR). These Serial inputs are the values that fill in the vacated position when the other values shift.

Sometimes, as in your ornament, we don’t need a 4-bit shift register; we might need a 3-bit Universal Bi-directional Shift Register. With a little discussion and thought, you realize that a ’194, by itself, cannot be used as a 3-bit Universal Bi-directional Shift Register without some additional logic. Then the thought occurs to the team that maybe other kinds of registered devices like ’161 4-bit Binary Counter, or ’195 Quad D-FF or even ’107 Dual JK-FF’s can be used with additional logic to build 3-bit Universal Bi-directional Shift Registers.

You will investigate these possibilities as a team for this problem. Each team will turn in ONE copy of the next page with your team answers, and each INDIVIDUAL will turn in a copy of the last page with your individual contribution to the problem solution.
(a) (5 Points) Once we have built such a 3-bit device (show it as a box with the appropriate inputs and outputs), you can use them to implement larger shift registers. For example, for the ornament, you really need seven 6-bit Universal Bi-directional Shift Registers (did you realize only 7 bits of the output mattered in the program?). Show how to connect two 3-bit Universal Bi-directional Shift Registers to make a 6-bit Universal Bi-directional Shift Register.

(b) (5 Points) After completing your individual designs, do you, as a team, think using this solution for your ornament is better than using the EE 260 II kit in your ornament? Justify your answer.
PART B
Individual Page

(c) (10 Points) Each individual should choose A DIFFERENT ONE of the registered devices
- '194 (4-bit Universal Bi-directional Shift Register) or
- '161 (4-bit Binary Counter) or
- '195 (Quad D-FF) or
- '107 (Dual JK-FF).

Show how to implement a 3-bit Universal Shift Register using one or more of your registered device, as many '153 (Quad 4-1 MUX with Common Selects) and any additional gates you need.