Instructions:
- Put your name on this page now and on all other pages after the exam begins.
- Do not cheat! Cheaters will be prosecuted.
- Show all work on the test paper. If you need more room, make a clearly indicated note on the front of the page, "MORE ON BACK", and use the back. The back of the page will not be graded without an indication on the front.
- You may not use any notes, homework, labs, other books, or calculators. You may use one 8.5x11 inch sheet.
- Read each question carefully and follow the instructions.
- Unless otherwise stated, assume all PALs/GALs are the type you use in the lab: PAL/GAL 22V10.
- The approximate point values are given to the left of each problem along with an estimated time for problem completion. I reserve the right to alter the points available for any given problems.

**PAL/GAL 22V10**

<table>
<thead>
<tr>
<th>Page</th>
<th>Approx. Available</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Total

---

Last Name:  
First Name:  

---

**CLK/I0**

GND

I/O/Q0

I/O/Q1

I/O/Q2

I/O/Q3

I/O/Q4

I/O/Q5

I/O/Q6

I/O/Q7

I/O/Q8

I/O/Q9

VCC
1. Below is a simplified block diagram of Val the Valet Robot. You saw a video of this robot in class on Friday. Val is trained to perform a task (such as fetching a beer) and then can repeat that task over and over again. Val can also recharge itself. You will design a small part of Val’s controller, the Fetch Controller. Notice that the Robot Controller below has three parts: a Train Controller, a Fetch Controller and the overall Robot Controller. The Train Controller writes data into the RAM from a training procedure and determines the address for the last byte of data, AddrEnd. (The first piece of data is saved to address 0.) A partial ASM flow chart is shown to the right.

Using VHDL, you are to design the part of the Fetch Controller (shown at right) enclosed in the dotted-box. You must get the data from the RAM (starting at address 0 and ending at address AddrEnd) and send it to the Robot Actuator. You must also control the active-high LED’s attached to the Fetch Controller. The ASM gives all the information you need.
(45%) 1. (Continued) Write the entire VHDL code for the design on the previous page. Don’t worry about pin placement.
[30%] 2. Design an 4-bit up/down counter triggered with a rising edge clock, with an active-low synchronous reset, active-low synchronous set, an active-high synchronous load, and active-high up and down controls. (Highest priority is reset, then set, then load, then up, then down.) Use the input names: Clk, R, S, Ld, Up, Dwn.

(20%) a) Write the **entire** VHDL code for the design described above. (Don’t worry about pin placement.)
(10%) 2. b) The 22V10 does not have a synchronous reset to control its flip-flops; the reset is asynchronous. How do you suppose VHDL will implement the synchronous reset? Explain with as much detail as you think necessary. Hint: How would you do it with discrete elements as you would have in EEL 3701?
[25%] 3. Design an 8-to-1 multiplexer. You must create a (partly) structural design and use the 4-to-1 MUX shown below with VHDL code shown to the right.

(10%) a) Draw a structural block diagram for the 8-to-1 multiplexer. You may assume structural components for any other simple device needed (like AND’s, OR’s, NOT’s, etc.). You may **not** use these added structural components for part b.

```
library ieee;
use ieee.std_logic_1164.all;
entity MUX4to1 is port(
    Input: in std_logic_vector(3 downto 0);
    Sel: in std_logic_vector(1 downto 0);
    Y: out std_logic);
end MUX4to1;
architecture behavior of MUX4to1 is
begin
    with Sel select
    Y <= Input(0) when "00",
    Input(1) when "01",
    Input(2) when "10",
    Input(3) when "11",
    Input(0) when others;
end behavior;
```

---

**Diagram:**
- **A0**, **A1**, **A2**, **A3**
- **S1**, **S0**
- **4-to-1 MUX**
- **X**
(15%) 3. b) Write the VHDL code to create an 8-to-1 multiplexer. Do not create any structures in addition to the 4-to-1 MUX. Hint: The design must be a mix of structural and behavioral VHDL.