EECS 303: Advanced Digital Logic Design
Final Exam

Robert Dick
6 December 2006

Show your work. Derivations are required for credit; end results are insufficient. Read each sentence.
In some cases I indicate something that can save you a lot of work.
1 (10 pts.) VHDL

Consider the following VHDL code.

```vhdl
entity RECOG is
    port (clk, a, reset: in bit;
          h: out bit);
end RECOG;

architecture STATE_MACHINE of RECOG is
    type state_type is (s0, s1, s2, s3);
    signal ps, ns : state_type;
begin
    STATE: process (reset, clk)
    begin
        if (reset = '1') then
            ps <= s0;
        elsif (clk'event and clk = '1') then
            ps <= ns;
        end if;
    end process STATE;

    NEW_STATE: process (ps, a)
    begin
        case ps is
        when s0 => when s1 =>
            case a is case a is
            when '0' => ns <= s2; when '0' => ns <= s0;
            when '1' => ns <= s1; when '1' => ns <= s1;
            end case; end case;
        when s2 =>
            case a is
            when '0' => ns <= s0;
            end case;
        when s3 =>
            case a is
            when '0' => ns <= s3;
            end case;
        end case;
    end process NEW_STATE;

    OUTPUT: process (ps)
    begin
        case ps is
        when s0 => h <= '0';
        when s1 => h <= '0';
        when s2 => h <= '0';
        when s3 => h <= '0';
        end case;
    end process OUTPUT;
end STATE_MACHINE;
```

1. Derive the corresponding state table.
2. Derive the corresponding state diagram.
3. Write the regular expression for the accepted sequences.

2 (10 pts.) State minimization

Minimize the following finite state machine. Show each step of minimization.

<table>
<thead>
<tr>
<th>Current State</th>
<th>Next State (input) 0</th>
<th>Next State (input) 1</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>A</td>
<td>X</td>
</tr>
</tbody>
</table>
3 (10 pts.) Asynchronous finite state machine design

Codes will be supplied to your machine via two binary inputs, \( \alpha \) and \( \beta \). The inputs will never change simultaneously. The combined pair of inputs permit four symbols to be entered, \( f = \alpha\beta \), \( g = \overline{\alpha}\beta \), \( h = \alpha\overline{\beta} \), and \( i = \alpha\beta \). The input is initially \( f \).

1. Design an asynchronous finite state machine that will accept the following sequences:

\[
f^+ g^+ h^+ i(f + g + h + i)^*\]

You need only carry out those stages of the design process up to and including state assignment and derivation of a fully-specified state table.

2. In three or fewer sentences, explain why there were so many plusses in the regular expression, e.g., \( f^+ \).

3. In three or fewer sentences, explain why it would or would not be reasonable to use kernel extraction to minimize the state variable functions of this machine.

4 (10 pts.) Arithmetic

Show the circuit-level diagram of an adder that accepts two two-bit unsigned integers and produces a three-bit unsigned integer. There must be at most two levels of logic between the inputs and outputs.

5 (10 pts.) Hazards

Consider the following function:

\[
f(a, b, c) = \sum (0, 1, 2)\]  

Assume access to complemented and uncomplemented input literals.

1. Show the schematic for a hazard-free two-level CMOS implementation with a minimal number of transistors.

2. How many transistors does your implementation have?

6 (10 pts.) State machines

Either prove that the following two machines are equivalent or prove that they are different. For each machine, A is the start state. Two machines are equivalent if there exists no single sequence of inputs that, if applied to both machines, will produce a different sequence of outputs.

<table>
<thead>
<tr>
<th>Current State</th>
<th>Next State (input)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>E</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Current State</th>
<th>Next State (input)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>C</td>
</tr>
</tbody>
</table>
7 (0 pts.) Research opportunity

1. How many unique and minimal deterministic finite state machines of \( l \) input symbols, \( m \) output symbols, and \( n \) states exist?

2. Explain how to design an asynchronous self-correcting finite state machine, i.e., a finite state machine that will have some specified behavior even if the indicated transition will be followed with probability \( \alpha \) where \( \alpha > 0.5 \).

3. Describe the simplest metric capable of accurately quantifying the exploitable memory access patterns in a heterogeneous NUMA processor.