ECE 238L
Digital Computers and Number Systems
Lecture lab 9

November 20, 2006

Lab 9 – FSM control for the LC-3

- Implement at least AND, ADD, NOT, JSR, BR, LD, ST and JMP/RET instructions
- ADD, AND, NOT should not require multiple states
- Don't forget the to have a decode state – use a decoder and not if/then within a process.

Problem – Programming VHDL can be tedious and error prone

- There other behavioral and structural circuits.
- VHDL has some structures that are very close to structural, but more convenient.
- I prefer using the when/else construct and will provide examples.
- Continue to use the provided flipflops, but these other structures will provide us some easier middle ground. (Don't go and write funky processes. I won't help you and you will loose points on your code.)

Example – Finite State Machines (FSMs)

```
library ieee ;
use ieee.std_logic_1164.all;

-----------------------------------------------

entity seq_design is
port( a: in std_logic;
clock: in std_logic;
reset: in std_logic;
x: out std_logic
);
end seq_design;

-----------------------------------------------
```

Example – Finite State Machines (FSMs) - Interface

```
architecture FSM of seq_design is

    -- define the states of FSM model

    type state_type is (S0, S1, S2, S3);
    signal next_state, current_state: state_type;
```

Example – FSMs - State FF

```
begin
    -- concurrent process 1: state registers
    state_reg: process(clock, reset)
    begin
if (reset='1') then
            current_state <= S0;
elsif (clock'event and clock='1') then
    current_state <= next_state;
end if;
    end process;
```

Example – FSMs - Next State/Output

```
   -- concurrent process 2: combinational logic
   -- sensitivity list: current_state, all inputs (a, ...)
    comb_logic: process(current_state, a)
    begin
case current_state is
    when S0 =>
                        x <= '0'; -- output
                        if a='0' then
                          next_state <= S0; -- calc next state
                        elsif a ='1' then
                          next_state <= S1;
                        end if;
```

Example – FSMs - Next State/Output

```
when S1 =>
          x <= '0';
          if a='0' then
            next_state <= S1;
          elsif a='1' then
            next_state <= S2;
          end if;
```

Example – FSMs - Next State/Output

```
when S2 =>
     x <= '0';
     if a='0' then
       next_state <= S2;
     elsif a='1' then
       next_state <= S3;
     end if;
when S3 =>
      x <= '1';
      if a='0' then
        next_state <= S3;
      elsif a='1' then
        next_state <= S0;
      end if;
```

Example – FSMs - Closing

```
    when others =>
      x <= '0';
      next_state <= S0;
  end case;
end process;
```