This test is open book and open notes. You have 2 hours. In the problems below, assume we are using a 0.5µm process (λ=0.25 µm) with V_{DD}=3.3V unless otherwise specified. Use Table 2-4 as needed when working out these problems.

Prefix reminder: nano = 10^{-9}, pico=10^{-12}, femto=10^{-15}

In problems 1-3 we will consider the design of an output pad and driver that connects a chip output to the outside world. The basic pad is a piece of metal2, sized 60µm X 80µm.

1. **Parasitic Calculations**  
   10 Points

   Calculate the parasitic capacitance of the output pad shown in the above diagram. Note that the dimensions are given in µm, not lambda!

2. **Delay Calculation / Transistor Sizing**  
   20 Points

   (a) Assume that the output buffer shown above is a static CMOS inverter with minimum size transistors. Assuming no external load, calculate the rise and fall times of this output buffer using the parasitic values calculated in problem 1.
(b) Now assume that we connect the chip to an external load capacitance of 3pF. Calculate the rise and fall times of the output buffer under these conditions.

(c) Suppose that we want the rise and fall time of our output buffer to be $t_r = t_f = 3\text{ns}$ with an external load of 3pF. Assuming that we use minimum length ($L = 2\lambda$) transistors, what values of width should we use for the n-transistor and p-transistor of the buffer?

3. **Power Estimation**

Assume that the output signal connected to this pad changes at a rate of 1MHz. How much power will be consumed just by the output buffer?
4. Latch Operation 15 Points

Given the schematic diagram of the dynamic latch shown below, fill in the timing diagram to show how nodes $S$ and $Q'$ change in response to changes on the $\phi$, $\phi'$ and $D$ inputs. Assume that $Q'$ is initially equal to 1.
Problems 4-8 deal with the VHDL code that follows, which describes a controller and datapath that performs a calculation on a stream of 8-bit data values that arrive at the input to this circuit once every clock cycle. When the result is calculated, the circuit signals this by asserting the output signal “result_avail” true for one clock cycle while making the calculated data available on the “result” output signal.

**Structural Description: file sys.vhd:**

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity sys is
  port( clk, reset : in std_logic;
       idata : in std_logic_vector(7 downto 0);
       result: out std_logic_vector(7 downto 0);
       result_avail : out std_logic);
end sys;

architecture structure of sys is

  component summer is
    port ( clk, reset, start_sum : in std_logic;
           idata : in std_logic_vector(7 downto 0);
           result : out std_logic_vector(7 downto 0) );
  end component;

  component fsm is
    port ( clk, reset : in std_logic;
           start_sum, result_avail : out std_logic );
  end component;

  for all : summer use entity work.summer (behavior);
  for all : fsm use entity work.fsm (behavior);

  signal start_sum: std_logic;

  begin

    S1: summer port map(clk=>clk, reset=>reset,
                        start_sum=>start_sum,
                        idata=>idata, result=>result);

    F1: fsm port map(clk=>clk, reset=>reset,
                     start_sum=>start_sum,
                     result_avail=>result_avail);

  end structure;
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity fsm is
  port ( clk, reset : in std_logic;
         start_sum, result_avail : out std_logic );
end fsm;

architecture behavior of fsm is
  type state_type is (S0, S1, S2, S3);
  signal cs, ns : state_type := S0;

begin
  comb: process (reset, cs)
  begin
    start_sum <= '0';
    if (reset = '1') then ns <= S0;
    else
      case (cs) is
      when S0 =>
        ns <= S1;
        result_avail <= '1';
        start_sum <= '1';
      when S1 =>
        ns <= S2;
      when S2 =>
        ns <= S3;
      when S3 =>
        ns <= S0;
      end case;
    end if;
  end process comb;

  state: process
  begin
    wait until (clk'event and clk='1');
    cs <= ns;
  end process state;
end behavior;
Data Unit Entity: file summer.vhd:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity summer is
  port ( clk, reset, start_sum : in std_logic;
         idata : in std_logic_vector(7 downto 0);
         result : out std_logic_vector(7 downto 0) );
end summer;

architecture behavior of summer is
  signal csum, nsum : std_logic_vector(9 downto 0);
begin
  comb: process (reset, start_sum, csum)
  begin
    result <= csum(9 downto 2);
    if (reset = '1') then nsum <= "0000000000";
    elsif (start_sum = '1') then nsum <= "00" & idata;
    else nsum <= unsigned("00" & idata) + unsigned(csum);
    end if;
  end process comb;
  state: process
  begin
    wait until (clk'event and clk='1');
    csum <= nsum;
  end process state;
end behavior;
```

5. VHDL Structural Description 15 Points

Draw a block diagram in the space provided below that shows input and output ports of the structural description, the component instances inside the structural description, and the connections between these ports and components.
6. **VHDL State Machine Description**  
**15 Points**

Draw a state diagram of the fsm entity in the space provided below.

---

7. **VHDL Datapath Component Description**  
**15 Points**

The summer entity mixes combinational and sequential logic functions to calculate and store a data value. Briefly describe what function this circuit performs

(a) when the “start_sum” input is 0 and the “reset” input is 0.

(b) When the “start_sum” input is 1 and the “reset” input is 0.

(c) When the “reset” input is 1.