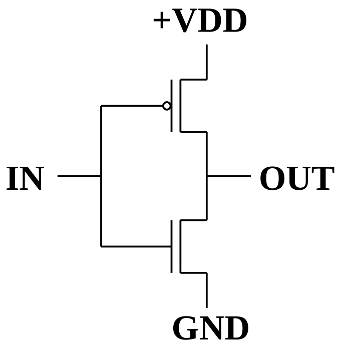
The University of Alabama in Huntsville
ECE Department
CPE 426 01
Final Exam Solution
Spring 2014

1. (6 points) Draw the transistor-level diagram of a CMOS inverter.



2. (5 points) If the NRE costs for CBIC and ASIC circuits are $400,000 and $2,500,000, respectively, and the cost of individual parts for CBIC and ASIC circuits are $35 and $11, respectively, what is the break-even manufacturing volume for these two types of circuits?

$400,000 + $35x = $2,500,000 + $11x
$24x = $2,100,000
87,500

3. (1 point) The starting point from which a verification plan can be created is a _**specification**_.

4 (1 point) List one type of coverage considered during the verification process. _**code** _

5. (8 points) For the process given below, A, B, C, and D are all integers that have a value of 0 at time = 10 ns. If E changes from '0' to '1' at time 20 ns, specify all resulting changes. Indicate the time at which each change will occur, the signal/variable affected, and the value to which it will change.

```
process
  variable F : integer := 1; variable A : integer := 0;
begin
  wait on E;
  A := 1;
  F := A + 5;
  B <= F + 1 after 5 ns;
  C <= B + 2 after 10 ns;
  D <= C + 5 after 15 ns;
  A := A + 5;
end process;
```

Scheduling Rules	Transport	Inertial
New before existing	Overwrite existing	Overwrite existing
New after existing	Append new	If $v_{new} = v_{existing}$, append new Elsif t_{new}-$t_{existing}$ > reject append new Else overwrite existing

Time	Signal/Variable	Scheduled
20 ns	A, F	B, 7, 25 ns
		C, 2, 30 ns
		D, 5, 35 ns
25 ns	B	
30 ns	C	
35 ns	D	

Time	A	B	C	D	E	F
10 ns	0	0	0	0	0	1
20 ns	6	0	0	0	1	6
25 ns	6	7	0	0	1	6
30 ns	6	7	2	0	1	6
35 ns	6	7	2	5	1	6

6. (2 points) List two layers that may be present in a layered testbench:
**scenario**
**transaction**

7. (1 point) A(n) _**interface**_ is a construct in System Verilog that represents a bundle of wires and has intelligence.

8. (10 points) Write a VHDL function that converts a 5-bit std_logic_vector to an integer. If any of the values are not '0' are '1', report an error. Note that the integer value of the binary number $a_4a_3a_2a_1a_0$ can be computed as $((((0 + a_4)*2 + a_3)*2 + a_2)*2 + a_1)*2 + a_0$

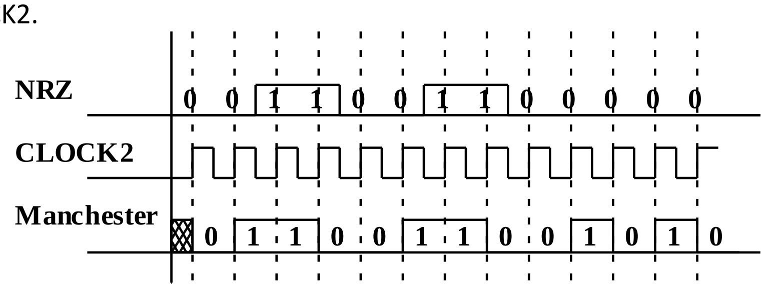
```
library ieee;
use ieee.std_logic_1164.all;

package MINE is
  function CONVERT (INPUT : std_logic_vector (4 downto 0)) return
integer;
end MINE;
package body MINE is
  function CONVERT (INPUT : std_logic_vector (4 downto 0)) return
integer is
    variable NUMB : integer;
  begin
    NUMB := 0;
    for I in 4 downto 0 loop
      if (INPUT(I) /= '0' and INPUT(I) /= '1') then
        report "Invalid bit provided in input.";
        return NUMB;
      else
        if INPUT(I) = '1' then
          NUMB := NUMB + 1;
        end if;
        NUMB := NUMB * 2;
      end if;
    end loop;
  end CONVERT;
end MINE;
```

9. (1 point) A(n) _**ASIC**_ is an integrated circuit produced for a specific application and produced in relatively small volumes.

10. (1 point) _**Pragmas**_ are inserted into VHDL models to give instructions to synthesis or other tools.

11. (15 points) Design a Moore state machine in VHDL which converts NRZ (non-return-to-zero) coding to Manchester coding. In NRZ coding, each bit is transmitted for one bit time without any change. For the Manchester code, a 0 is transmitted as 0 for the first half of the bit time and 1 for the second half, but a 1 is transmitted as 1 for the first half and 0 for the second half. In order to do this conversion, use a clock(CLOCK2) that is twice the frequency of the basic clock. Note that if the NRZ bit is 0, it will be 0 for two CLOCK2 periods. Similarly, if the NRZ bit is 1, it will be 1 for two CLOCK2 periods. Your design should have an active low synchronous reset and work with CLOCK2.




```
entity NRZ2MAN is
  port (NRZ, CLOCK2, RESET : in BIT;
                      MAN : out BIT);
end NRZ2MAN;

architecture BEHAVE of NRZ2MAN is
  type STATE_TYPE is (S0, S1, S2, S3);
  signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
begin
  process (CURRENT_STATE)
  begin
    case CURRENT_STATE is
      when S0 => if (NRZ = '0') then
                   NEXT_STATE <= S1;
                 else
                   NEXT_STATE <= S3;
                 end if;
      when S1 => NEXT_STATE <= S2;
      when S2 => if NRZ = '0' then
                   NEXT_STATE <= S1;
                 else
                   NEXT_STATE <= S3;
                 end if;
      when S3 => NEXT_STATE <= S0;
      when OTHERS => NEXT_STATE <= S0;
    end case;
  end process;
```

```
  process (RESET, CLOCK2)
  begin
    if (RESET = '0') then
      CURRENT_STATE <= S0;
    elsif (CLOCK2'EVENT and CLOCK2 = '1') then
      CURRENT_STATE <= NEXT_STATE;
    end if;
  end process;

  process (CURRENT_STATE)
  begin
    case CURRENT_STATE is
      when S0 | S1 => MAN <= '0';
      when S2 | S3 => MAN <= '1';
      when OTHERS  => MAN <= '0';
    end case;
  end process;

end BEHAVE;
```

12. (6 points) In the following VHDL, `state` and `nextstate` are integers with a range of 0 to 2 **(should be 0 to 3)**.

```
process (state, X)
begin
  case state is
    when 0 => if X = '1' then nextstate <= 1;
    when 1 => if X = '0' then nextstate <= 2;
    when 3 => If X = '1' then nextstate <= 0;
  end case
end process;
```

a. Explain why a latch would be created when the code is synthesized.

A latch is created because no action is defined for the case of state = 2.

b. Make changes in the code that would eliminate the latch.

```
process (state, X)
begin
  case state is
    when 0=> if X = '1' then nextstate <= 1;
    when 1 => if X = '0' then nextstate <= 2;
    when 2 => nextstate <= 0;
    when 3 => If X = '1' then nextstate <= 0;
  end case
end process;
```

13. (12 points) (a) Write a VHDL entity and architecture for a circuit that has four inputs and three outputs. The 3-bit output should be a binary number equal to the number of 1's found in the inputs. (b) Write a VHDL entity and architecture for a circuit that counts the number of 1's in a 12-bit number. Use three of the modules from (a) along with overloaded addition operators.

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

entity COUNT_ONES is
  port (INPUT : in std_logic_vector (3 downto 0);
        OUTPUT : out std_logic_vector (2 downto 0));
end COUNT_ONES;

architecture BEHAV of COUNT_ONES is
begin
  process (INPUT)
    variable COUNT : std_logic_vector (2 downto 0);
  begin
    COUNT := "000";
    for I in 3 downto 0 loop
      if (INPUT(I) = '1') then
        COUNT := COUNT + "1";
      end if;
    end loop;
    OUTPUT <= COUNT;
  end process;
end BEHAV;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use work.all;

entity HIER_COUNT_ONES is
  port (INPUT : in std_logic_vector (11 downto 0);
        OUTPUT : out std_logic_vector (3 downto 0));
end HIER_COUNT_ONES;

architecture STRUCT of HIER_COUNT_ONES is
  signal COUNT : std_logic_vector (8 downto 0);
begin
  U1 : entity COUNT_ONES(BEHAV)
       port map (INPUT(11 downto 8), COUNT(8 downto 6));
  U2 : entity COUNT_ONES(BEHAV)
       port map (INPUT(7 downto 4), COUNT(5 downto 3));
  U3 : entity COUNT_ONES(BEHAV)
       port map (INPUT(3 downto 0), COUNT(2 downto 0));
  OUTPUT <= "0000" + COUNT(8 downto 6) + COUNT(5 downto 3)
          + COUNT(2 downto 0);
end STRUCT;
```

14. (1 point) The @ construct in System Verilog is similar to the _**wait**_ construct in VHDL.

15. (1 point) A _**constrained random**_ test environment allows you to run hundreds of tests without having to hand check the results.

16. (12 points) Create a class `Test_problem` containing two random variables, 8-bit `data` and 4-bit `address`. Create a constraint block so that:
a. `data` is always equal to 5
b. The probability of `address == 0` is 10%
c. The probability of `address` being between [1:14] is 80%
d. The probability of `address == 15` is 10%

In an initial block, construct a `Test_problem` object and randomize it.

```
package RVars;
  class Test_problem;
    rand bit[7:0] data;
    rand bit[3:0] address;
    constraint C1
    {
       data inside {5};
       address dist {0 :/10, [1:14] :/80, 15 :/ 10};
    }
  endclass : Test_problem
endpackage : RVars

import RVars::*;
module test();
  initial begin
    Test_problem Object;
    Object = new();
    Object.randomize();
  end
endmodule
```

17. (1 point) _**std_logic_vector**_ is an example of an unconstrained array.

18. (1 point) _**VHDL**_ is an annoyingly strongly typed language.

19. (12 points) Design an interface and testbench for the ARM Advanced High-performance Bus (AHB). You are provided a bus master as verification IP that can initiate AHB transactions. You are testing a slave design. The testbench instantiates the interface, slave, and master. Your interface will display an error if the transaction type is not IDLE or NONSEQ on the negative edge of HCLK. The AHB signals are described below.

Signal	Width	Direction	Description
HCLK	1	Output	Clock
HADDR	21	Output	Address
HWRITE	1	Output	Write flag: 1=write, 0=read
HTRANS	2	Output	Transaction type: 2'b00=IDLE, 2'b10=NONSEQ
HWDATA	8	Output	Write data
HRDATA	8	Input	Read data

```
interface AHB_IF (input bit CLK);
  logic HWRITE;
  logic [1:0] HTRANS;
  logic [7:0] HWDATA, HRDATA;
  logic [20:0] HADDR;
  clocking CB @(negedge CLK);
    output HRDATA;
    input HADDR, HWRITE, HTRANS, HWDATA;
  endclocking;
  modport TEST(clocking CB);
endinterface

module TOP();
  bit CLK;
  always #5 CLK = ~CLK;
  AHB_IF AHBIF(CLK);
  MASTER U1 (.DATA (AHBIF.CB.HRDATA),
             .CLK (CLK));
  SLAVE U2 (.HCLK (CLK),
            .HADDR (AHBIF.CB.HADDR),
            .HWRITE (AHBIF.CB.HWRITE),
            .HTRANS (AHBIF.CB.HTRANS),
            .HWDATA (AHBIF.CB.HWDATA),
            .HRDATA (AHBIF.CB.HRDATA));
  TEST U3 (AHBIF.CB);
endmodule

module TEST(AHB_IF AHBIF);
  initial begin
    repeat (100)
    begin
      @(negedge AHBIF.CB);
      if (AHBIF.HTRANS != 2'b00  && AHBIF.HTRANS != 2'b10)
        $display ("Error : Wrong transaction type");
    end
  end
endmodule
```

20. (3 points) The three primary types of design units are _**entity**_, _**configuration**_, and _**package**_