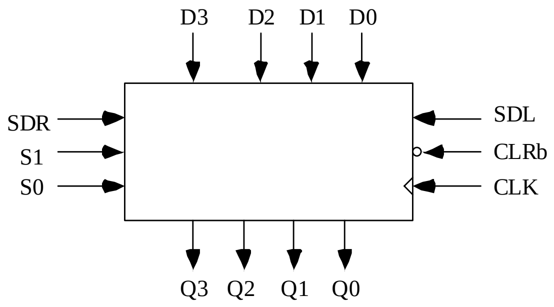
The University of Alabama in Huntsville
ECE Department
CPE 526 01
Midterm Exam Solution
Spring 2005

1. (10 points) A description of a 74194 4-bit bi-directional shift register follows: The CLRb input is asynchronous and active low and overrides all the other control inputs. All other state changes occur following the rising edge of the clock. If the control inputs S1 = S0 = 1, the register is loaded in parallel. If S1 = 1 and S0 = 0, the register is shifted right and SDR (serial data right) is shifted into Q3. IF S1 = 0 and S0 = 1, the register is shifted left and SDL is shifted into Q0. If S1 = S0 = 0, no action occurs.



(a) (7 points) Write a VHDL description of an 8-bit bi-directional shift register that uses two 74194s as components. The parallel inputs and outputs to the 8-bit register should be X and Y. The serial inputs should be RSD and LSD. (b) (2 points) Write an entity for the 74194. (c) (6 points) Write an architecture for the 74194

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

entity S74194 is
  port (CLRB : in std_logic; CLK : in std_logic;
        S1, S0 : in std_logic; SDR, SDL : in std_logic;
        D : in std_logic_vector(3 downto 0);
        Q : out std_logic_vector(3 downto 0));
end S74194;

architecture BEHAV of S74194 is
  signal TEMP : std_logic_vector(3 downto 0);
begin
  process(CLRB, CLK)
  begin
    if (CLRB = '0') then
      TEMP <= "0000";
    elsif (CLK'event and CLK = '1') then
      if (S1 = '1' and S0 = '1') then
        TEMP <= D;
      elsif (S1 = '1' and S0 = '0') then
        TEMP <= SDR & TEMP(3 downto 1);
      elsif (S1 = '0' and S1 = '1') then
        TEMP <= TEMP(2 downto 0) & SDL;
      end if;
    end if;
  end process;
end BEHAV;

library ieee;
use ieee.std_logic_1164.all;

entity SHIFTER8 is
  port (CLRB : in std_logic; CLK : in std_logic;
        S1, S0 : in std_logic; RSD, LSD : in std_logic;
        X : in std_logic_vector(7 downto 0);
        Y : out std_logic_vector(7 downto 0));
end SHIFTER8;

architecture STRUCTURE of SHIFTER8 is
  component S74194 is
    port (CLRB : in std_logic; CLK : in std_logic;
          S1, S0 : in std_logic; SDR, SDL : in std_logic;
          D : in std_logic_vector(3 downto 0);
          Q : out std_logic_vector(3 downto 0));
  end component;
  signal TEMP0, TEMP1 : std_logic_vector(3 downto 0);
begin
  U1 : S74194 port map(CLRB => CLRB, CLK => CLK, S1 => S1, S0 => S0,
                       SDR => RSD , SDL => TEMP0(3),
                       D => X(7 downto 4), Q => TEMP1);
  U2 : S74194 port map(CLRB => CLRB, CLK => CLK, S1 => S1, S0 => S0,
                       SDR => TEMP1(0), SDL => LSD,
                       D => X(3 downto 0),Q => TEMP0);
  Y(7 downto 4) <= TEMP1;
  Y(3 downto 0) <= TEMP0;
end STRUCTURE;
```

2. (1 point) VHDL is a strongly typed language. (True/False) _***TRUE***_

3. (15 points). Write a VHDL function that accepts a std_logic_vector of arbitrary length and an integer that specifies the number of bits the std_logic_vector is to be rotated to the left and returns the rotated std_logic_vector. Issue an error message if the integer is greater than the length of the input. For example:

Input:	0101111, 2
Output:	0111101

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

package NEED_IT_TO_COMPILE is
  function ROTATE_LEFT(VECTOR : in std_logic_vector;
                       N : in integer) return std_logic_vector;
end package;
```

```
package body NEED_IT_TO_COMPILE is
  function ROTATE_LEFT(VECTOR : in std_logic_vector;
                       N : in integer) return std_logic_vector is
    variable NEW_VECTOR : std_logic_vector(VECTOR'range);
    alias TEMP_NEW : std_logic_vector(VECTOR'length-1 downto 0) is
NEW_VECTOR;
    variable TEMP_BIT : std_logic;
  begin
    TEMP_NEW := VECTOR;
    for I in 0 to N-1 loop
      TEMP_BIT := TEMP_NEW(VECTOR'length-1);
      for J in VECTOR'length-1 downto 1 loop
        TEMP_NEW(J) := TEMP_NEW(J-1);
      end loop;
      TEMP_NEW(0) := TEMP_BIT;
    end loop;
    return TEMP_NEW;
  end ROTATE_LEFT;
end package body;
```

4. (1 point) A(n) ___event_ occurs when a signal changes value.

5. (1 point) All statements inside of a process are ___sequential_.

6. (3 points) For the following function call, which function will be called? _a_

```
VARIABLE a, b : INTEGER;
b := decrement (a);

(a)         FUNCTION decrement (x : INTEGER) RETURN INTEGER;
(b)         FUNCTION decrement (x : REAL) RETURN REAL;
```

7. (1 point) A __configuration__ binds an instantiated component to a library model

8. (15 points) Design an address decoder. One input to the address decoder is an 8-bit address, which can have any range with a length of 8, for example: std_logic_vector addr(8 to 15). The second input is check : std_logic_vector(5 down to 0). The address decoder will output Sel = ‘1’ if the upper 6 bits of the 8-bit address match the 6-bit check vector. For example, if addr = “10001010” and check = “1000--“ then Sel = ‘1’. Only the 6 leftmost bits of addr will be compared; the remaining bits are ignored. An ‘-‘ in the check vector is a don’t care.(a) (4 points) Write an entity for the device. (b) (11 points)Write an architecture for the device.

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

entity address_decoder is
  port (ADDRESS : in std_logic_vector;
        CHECK : in std_logic_vector(5 downto 0);
        SEL : out std_logic);
end ADDRESS_DECODER;
```

```
architecture BEHAV of ADDRESS_DECODER is
  alias ADDR : std_logic_vector(ADDRESS'length-1 downto 0) is ADDRESS;
begin
  process(ADDRESS, CHECK)
   variable MATCH : boolean;
  begin
    MATCH := TRUE;
    for I in ADDRESS'length-1 downto ADDRESS'length-1-5 loop
      if ((ADDR(I) /= CHECK(I-2)) and (CHECK(I-2) /= '-')) then
        MATCH := FALSE;
      end if;
    end loop;
    if (MATCH) then
      SEL <= '1';
    else
      SEL <= '0';
    end if;
  end process;
end BEHAV;
```

9. (6 points) Design a 2-to-1 multiplexer. (a) (2 points) Write a VHDL entity. (b) (4 points) Use concurrent signal assignments to implement the architecture.

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

entity MUX2_1 is
  port (I1, I0, SEL : in std_logic;
        F : out std_logic);
end MUX2_1;

architecture CONCURRENT of MUX2_1 is
begin
  F <= I1 when SEL = '1' else
       I0 when SEL = '0';
end CONCURRENT;
```

10. (10 points)A DD flip-flop is similar to a D flip-flop, except that the flip-flop can change state (Q+ = D) on both the rising edge and falling edge of the clock input. The flip-flop has a direct reset input R, and R = 0 resets the flip-flop to Q = 0 independent of the clock. (a) (2 points) Write a VHDL entity for the DD flip-flop. (b) (6 points) Write a VHDL architecture for a DD flip-flop. (b) (2 points) Is this description synthesizable? Why or why not? As it turns out, Synopsys refuses to synthesize it and Leonardo synthesizes it to a latch, which does not work the same as the behavioral model. The problem is having a change occur on both edges of the clock.

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

entity DD_FF is
  port (D, R, CLK : in std_logic;
        Q, QB : out std_logic);
end DD_FF;
```

```
architecture BEHAV of DD_FF is
  signal QTEMP : std_logic;
begin
  process (R, CLK)
  begin
    if (R = '0') then
      QTEMP <= '0';
    elsif (CLK'event) then
      QTEMP <= D;
    end if;
  end process;
  Q <= QTEMP;
  QB <= not QTEMP;
end BEHAV;
```

11. (15 points) For the following VHDL, assume that A changes to ‘1’ at 5 ns. Give the values of A, B, C, D, E, and F each time a change occurs. Carry this out until no further change occurs. I

```
entity prob is
  port (D : inout bit);
end prob;

architecture PROB of PROB is
  signal A, B, C, E, F : bit;
begin
  P1: process (A, C)
  begin
    B <= A after 3 ns;
    E <= C after 5 ns;
  end process P1;
  C1: C <= A after 10 ns;
  P2: process (C, E)
  begin
    F <= C and E after 4 ns;
  end process P2;
  C2: D <= A or B or C or F after 1 ns;
end PROB;
```

Time	A	B	C	D	E	F
0 ns	0	0	0	0	0	0
5 ns	1	0	0	0	0	0

Time	Event	Process Triggered	Scheduled Transaction	Event?
5 ns	A → ‘1’	P1	B ‘1’ 8 ns	Yes
		P1	E ‘0’ 10 ns	No
		C1	C ‘1’ 15 ns	Yes
		C2	D ‘1’ 6 ns	Yes
6 ns	D → ‘1’	none		
8 ns	B → ‘1’	C2	D ‘1’ 9 ns	No
15 ns	C → ‘1’	P1	B ‘1’ 18 ns	No
		P1	E ‘1’ 20 ns	Yes
		P2	F ‘0’ 19 ns	No
		C2	D ‘1’ 16 ns	No
20 ns	E → ‘1’	P2	F ‘1’ 24 ns	Yes
24 ns	F → ‘1’	C2	D ‘1’ 25 ns	No

12. (21 points) Specify type declarations for the following data types.

a. (4 points) A four valued logic system, MVL4, with values '0', '1', 'X' and 'Z'. Values '0' and '1' have the usual logic meaning and 'X' means unknown. Any uninitialized data item of this type should have value 'X'.

```
type MVL4 is ('X', '0', '1', 'Z');
```

b. (4 points) A GRADE_LEVEL enumeration data type.

```
type GRADE_LEVEL is (FIRST, SECOND, THIRD, FOURTH,
                       FIFTH, SIXTH, SEVENTH, EIGHTH,
                       NINTH, TENTH, ELEVENTH, TWELFTH);
```

c. (2 points) A data type AGE that can have integer values in the range from 1 to 120.

```
type AGE is range 1 to 120; or
subtype AGE is integer range 1 to 120;
```

d. (2 points) A data type COST that can have real values between $0.00 and $1,405.00.

```
type cost is range 0.0 to 1405.00; or
subtype SUB_COST is real range 0.0 to 1405.0;
```

e. (2 points) A descending range data type HIGH_WORD with integer values from 63 to 32.

```
type HIGH_WORD is range 63 downto 32; or
subtype SUB_HIGH_WORD is integer range 63 downto 32;
```

f. (3 points) A 64-bit ascending-index register composite data type, REG_64_HIGH, with index valued from the type HIGH_WORD declared above, and component values of type MVL4.

```
type REG_64_HIGH is array (HIGH_WORD) of MVL4; or
type SUB_REG_64_HIGH is array (SUB_HIGH_WORD) of MVL4;
```

g. (4 points) A four-dimensional table, TABLE_4D, with index values and table entries all of type std_logic (which has been declared elsewhere and is visible).

```
type TABLE_4D is array (std_logic, std_logic, std_logic,
                          std_logic) of std_logic;
```

13. (1 point) __Communication_ is the hardest problem.