CIRCLE YOUR SELECTED ANSWERS OR FILL IN AS NEEDED

1. a b c d e (5 pts)
2. a b c d (5 pts)
3. a b c d (5 pts)
4. a b c d e (5 pts)
5. a b c d e (5 pts)
6. a b c d (5 pts)
7. a b c d e (5 pts)
8. a b c d e (5 pts)
9. a b c d e (10 pts)
10. a b c d (5 pts)
11. a b c d (5 pts)
12. a b c d e (10 pts)
13. a b c d (5 pts)
14. a b c d (5 pts)
15. Fill in at right (10 pts)
16. a b (5 pts)
17. a b c d e (5 pts)

Fill in the Karnaugh map with ones and zeroes. It is not necessary to circle prime implicants or determine the corresponding equation (but you may do so if it helps with problem 16).
1. (5 pts) What is the period of the signal shown below? The arrow to the left of waveform indicates the ground level, and the scales are displayed (1.00 V/division, 800 ns/division).

a) 1.25 GS/s  
b) 1.8 μs  
c) 800 ns  
d) 3.8 μs  
e) 4.7 ns

2. (5 pts) In Lab 8, you added an output called FETCH_OUT. Which of the following is true regarding this output? (Select one.)

a) It is asserted during EX_FETCH, the execute state of FETCH.

b) It was added so that the logic analyzer can distinguish between an instruction fetch and the retrieval of an operand.

c) It is a vector of multiple bits.

d) It acts as a “Memory Read” signal, telling the RAM when a read occurs, as opposed to MW, which indicates when a write occurs.

3. (5 pts) Which type of Quartus simulation would you use to better understand the effects of propagation delay in your design? (Select one.)

a) Timing  
b) Block  
c) Functional  
d) Fitting
A standard integrated circuit called the 74HCT147 has an equivalent part in the Quartus library. It has nine inputs and four outputs, as shown below. Its operation is described by a truth table below.

**Functional Diagram**

![Diagram of the 74HCT147](image)

**Truth Table**

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>i1</td>
<td>i2</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
</tbody>
</table>

H = High Logic Level, L = Low Logic Level, X = Don't Care


4. (5 pts) Referring to the truth table above, what are the outputs when the inputs are the following?

\[ \overline{i2} = L, \overline{i7} = L, \overline{i8} = H, \overline{i9} = H \] (other inputs unknown)

a) \( \overline{Y3} = H, \overline{Y2} = L, \overline{Y1} = L, \overline{Y0} = L \)

b) \( \overline{Y3} = H, \overline{Y2} = H, \overline{Y1} = L, \overline{Y0} = H \)

c) \( \overline{Y3} = L, \overline{Y2} = H, \overline{Y1} = H, \overline{Y0} = H \)

d) \( \overline{Y3} = L, \overline{Y2} = H, \overline{Y1} = H, \overline{Y0} = L \)

e) Impossible to determine without additional information
5. (5 pts) Given the circuit diagram below and the table of relevant timing parameters for various parts, find the worst-case timing path for the state machine. Do not attempt to minimize it or change it in any way. Then compute the minimum clock period (the time associated with that worst-case path).

<table>
<thead>
<tr>
<th>Device</th>
<th>( t_p ) (max propagation delay)</th>
<th>( t_{su} ) (setup time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D Flip Flop</td>
<td>8 ns</td>
<td>2 ns</td>
</tr>
<tr>
<td>OR Gate</td>
<td>11 ns</td>
<td>N/A</td>
</tr>
<tr>
<td>AND Gate</td>
<td>12 ns</td>
<td>N/A</td>
</tr>
<tr>
<td>Inverter</td>
<td>8 ns</td>
<td>N/A</td>
</tr>
</tbody>
</table>

a) 33 ns  
b) 44 ns  
c) 50 ns  
d) 52 ns  
e) None of the above

Questions 6 and 7 refer to the following VHDL source code, extracted from the IO_DECODER_0.VHD file provided as part of the project.

```vhdl
  IO_INT <= TO_INTEGER(UNSIGNED(IO_CYCLE & IO_ADDR));
  SWITCH_EN <= '1' when IO_INT = 16#100# else '0';
  LED_EN <= '1' when IO_INT = 16#101# else '0';
  TIMER_EN <= '1' when IO_INT = 16#102# else '0';
  DIG_IN_EN <= '1' when IO_INT = 16#103# else '0';
  HEX_EN <= '1' when IO_INT = 16#104# else '0';
  LCD_EN <= '1' when IO_INT = 16#106# else '0';
  ... (similar lines omitted)
  SONAR_EN <= '1' when ((IO_INT >= 16#1A0#) AND (IO_INT < 16#1A8#)) else '0';
```
6. (5 pts) Referring to the code above, why is there an operation of “IO_CYCLE & IO_ADDR” in the process of producing an INTEGER called IO_INT? (Select one.)
   a) So that the various output signals (chip selects or chip enables) can decode both the correct I/O address and the occurrence of an OUT operation
   b) Because “&” means logical AND, so this means that there IS an I/O cycle, and there IS an IO address.
   c) So that the various output signals (chip selects or chip enables) can decode both the correct I/O address and the occurrence of an IN or OUT operation
   d) So that the various output signals (chip selects or chip enables) can decode both the correct I/O address and the occurrence of a valid memory read or write

7. (5 pts) Still referring to the source code above, which of the output signals is asserted (showing a value of ‘1’) for at least one SCOMP cycle when an OUT to an IO_ADDR of 10100001 (in binary) occurs?
   a) SWITCH_EN
   b) LED_EN
   c) TIMER_EN
   d) SONAR_EN
   e) None of the above

8. (5 pts) In what part of the VHDL source for SCOMP would you expect to find the following?
   FOR i IN 0 TO 6 LOOP
   PC_STACK(i + 1) <= PC_STACK(i);
   END LOOP;
   PC_STACK(0) <= PC;
   PC          <= IR(9 DOWNTO 0);
   STATE       <= FETCH;
   a) Implementation of RAM
   b) Implementation of execute state for CALL
   c) The initialization related to PC_RESET
   d) Implementation of execute state for RETURN
   e) None of the above

9. (10 pts) Which of the following are differences between a “classic” state diagram and a UML statechart? (Mark all that apply.)
   a) One works only for Mealy machines, the other only for Moore machines
   b) UML statecharts include a means for calculating propagation delay
   c) UML statecharts explicitly show the name of any output variables, while classic diagrams may require some sort of legend
   d) Classic diagrams always have more transition arcs
   e) UML statecharts cannot show the state names, while classic diagrams put each state name in the corresponding state symbol
The following code is the complete implementation of SONAR.VHD that was provided as a starting point for the project, with comments removed and LIBRARY/USE statements omitted.

ENTITY SONAR IS
PORT(CLOCK,
    RESETN,
    CS,
    IO_WRITE,
    ECHO : IN    STD_LOGIC;
    ADDR : IN    STD_LOGIC_VECTOR(2 DOWNTO 0);
    INIT : OUT   STD_LOGIC;
    LISTEN : OUT   STD_LOGIC;
    SONAR_NUM : OUT   STD_LOGIC_VECTOR(2 DOWNTO 0);
    IO_DATA   : INOUT STD_LOGIC_VECTOR(15 DOWNTO 0) );
END SONAR;
ARCHITECTURE behavior OF SONAR IS

TYPE SONAR_DATA IS ARRAY (7 DOWNTO 0) OF STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL SONAR_RESULT : SONAR_DATA;
SIGNAL SELECTED_SONAR : STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL PING_TIME : STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL SONAR_CONTROL : STD_LOGIC_VECTOR(15 DOWNTO 0);
CONSTANT MAX_DIST  : STD_LOGIC_VECTOR(15 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(400, 16);
CONSTANT OFF_TIME  : STD_LOGIC_VECTOR(15 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(500, 16);
CONSTANT BLANK_TIME  : STD_LOGIC_VECTOR(15 DOWNTO 0) := CONV_STD_LOGIC_VECTOR(11, 16);
CONSTANT NO_ECHO  : STD_LOGIC_VECTOR(15 DOWNTO 0) := x"FFFF";
SIGNAL IO_IN : STD_LOGIC;
SIGNAL LATCH  : STD_LOGIC;
SIGNAL PING_STARTED : STD_LOGIC;
SIGNAL PING_DONE : STD_LOGIC;
SIGNAL I : INTEGER;
BEGIN

IO_BUS: lpm_bustri
GENERIC MAP ( lpm_width => 16 )
PORT MAP (data => SONAR_RESULT( CONV_INTEGER(ADDR)),
enabledt => IO_IN,
tridata => IO_DATA);
IO_IN <= (CS AND NOT(IO_WRITE));
LATCH <= CS AND IO_WRITE;
SONAR_NUM <= "000";
PINGER: PROCESS (CLOCK, RESETN)
BEGIN
IF (RESETN = '0') THEN
    PING_TIME <= x"0000";
    LISTEN <= '0';
    INIT <= '0';
    SELECTED_SONAR <= "000";
    PING_STARTED <= '0';
    PING_DONE <= '0';
    FOR I IN 0 to 7 LOOP
        SONAR_RESULT( I ) <= NO_ECHO;
    END LOOP;
ELSIF (RISING_EDGE(CLOCK)) THEN
    IF (PING_STARTED = '0') THEN
        PING_STARTED <= '1';
        PING_DONE <= '0';
        PING_TIME <= x"0000";
        LISTEN <= '0';
        INIT <= '1';
    ELSIF (PING_STARTED = '1') THEN
        PING_TIME <= PING_TIME + 1;
        IF ((ECHO = '1') AND (PING_DONE = '0')) THEN
            PING_DONE <= '1';
            SONAR_RESULT( 0 ) <= PING_TIME;
        END IF;
        IF (PING_TIME = BLANK_TIME) THEN
            LISTEN <= '1';
        ELSIF (PING_TIME = MAX_DIST) THEN
            INIT <= '0';
            LISTEN <= '0';
    END IF;
END IF;
END PINGER;
IF (PING_DONE = '0') THEN
    SONAR_RESULT( Conv_INTEGER(SELECTED_SONAR) ) <= NO_ECHO;
END IF;
ELSIF (PING_TIME = OFF_TIME) THEN
    PING_STARTED <= '0';
END IF;
END IF;
END IF;
END PROCESS;

INPUT_HANDLER: PROCESS (RESETN, LATCH)
BEGIN
    IF (RESETN = '0') THEN
        SONAR_CONTROL <= x"0000";
    ELSIF (RISING_EDGE(LATCH)) THEN
        SONAR_CONTROL <= IO_DATA(15 DOWNTO 0);
    END IF;
END PROCESS;
END behavior;

10. (5 pts) Which of the following describes the highest-level parallel actions that are occurring within the SONAR device above? (Select one.)
   a) An ARCHITECTURE, an ENTITY, and three assignment statements
   b) The operations resulting from IF RESETN = ‘0’, those resulting from ELSIF (RISING_EDGE(CLOCK), and those resulting from ELSIF (RISING_EDGE(LATCH)
   c) Two PROCESS statements, an instance of LPM_BUSTRI, and three assignment statements
   d) A PORT, an ARCHITECTURE, and two PROCESS statements

11. (5 pts) Still referring to the VHDL code for SONAR.VHD above, what is the apparent purpose of the signal called SONAR_CONTROL? (Select one.)
   a) It is directly connected to the INIT output, and thus controls the pinging of the sonar transducer
   b) It is a register that SCOMP can write to by performing an OUT instruction
   c) It is a register than SCOMP can read by performing an IN instruction
   d) It is the count of the number of clock cycles that have occurred since a ping started

12. (10 pts) Turn your attention specifically to the IF/THEN/ELSIF/ELSIF construct that begins with IF (PING_TIME = BLANK_TIME) THEN and continues to the matching ENDIF (highlighted in bold face). Which of the following are true statements regarding this block? You may have to consider some code outside the highlighted area, but the questions are specifically about the assignment of signal values inside the highlighted area. (Select all that apply – or none.)
   a) If RESETN has a value of ‘0’ then this block of code does not determine the value of any signals in the SONAR device
   b) When synthesized within the FPGA, this would be implemented as a CPU that sequentially compares PING_TIME to different values in successive clock cycles, then makes appropriate signal assignments before exiting
   c) Once the assignment of NO_ECHO is made to an element of SONAR_RESULT, the ping stops by the next rising edge of CLOCK (lowering INIT and LISTEN)
   d) This the only region of code that can determine the value for PING_STARTED
   e) LISTEN can simultaneously have the values of 0 and 1
Consider the following ASM code, which is identical to that provided as a starting point for the project (without the comments).

```
ORG &H000
Start: NOP
LOAD Zero
OUT LCD
IN SONAR0
OUT SEVENSEG
CALL Wait1
JUMP Start

Wait1: OUT TIMER
Wloop: IN TIMER
SUB ONESEC
JNEG Wloop
RETURN

ONESEC: DW 10
Zero: DW 0

; IO address space map
SWITCHES: EQU &H00
LEDS: EQU &H01
TIMER: EQU &H02
XIO: EQU &H03
SEVENSEG: EQU &H04
LCD: EQU &H06
SONAR0: EQU &HA0
ANOTHER_SONAR_REGISTER: EQU &HA1
```

13. (5 pts) At what memory address does the LOAD Zero instruction reside?  (Select one answer.)
   a) 0x0
   b) 0x1
   c) 0x2
   d) None of the above

14. (5 pts) If you replaced the JNEG Wloop instruction with JZERO Wloop (and reassembled, etc. as above), what would you expect to happen. You may assume that the IN Sonar0 has been restored to its original state, cancelling the effect of the previous question.
   a) The timer subroutine would never exit
   b) The timer subroutine would exit sooner
   c) The timer subroutine would exit later
   d) There would be no effect
15. (10 pts) Draw the Karnaugh Map for the following circuit, circling the implicants that are represented by the two NAND gates, and placing a “1” or “0” in EVERY cell. (Fill in the Karnaugh Map on the answer sheet.)

16. (5 pts) Is the circuit shown in the previous problem a minimal sum of products implementation?
   a) YES
   b) NO

17. (5 pts) The PC (Program Counter) of the simple computer was declared to be of what type?
   a) STD_LOGIC
   b) STD_LOGIC_VECTOR(9 DOWNTO 0)
   c) BIT
   d) STD_LOGIC_VECTOR(15 DOWNTO 0)
   e) None of the above