筆記課題　　解答及び解説

筆記課題　解答及び解説

「ＨＤＬによるディジタル回路設計」

|  |
| --- |
| １　作業時間  90分（休憩を除く）  ２　配付資料  問題用紙、解答用紙  ３　課題作成、提出方法  解答用紙のみを回収する |





1. ＡＮＤ素子の図記号（MIL記号）、真理値表、論理式を書いて下さい。 （各2点：計6点）
2. 次の論理式をブール代数を用いて簡単化しなさい。 （5点）
3. 次の論理式をそのまま論理回路図で描きなさい。 （5点）  
   
4. 次の仕様のようなオーディションの合否回路について答えなさい。 （計20点）  
   　 仕様   
   ・審査員はＡ，Ｂ，Ｃ，Ｄの４名とし、それぞれが１個ずつの投票ボタンを持つ。  
   ・Ａを委員長とする。  
   ・多数決で合否を決めるが、２－２の場合は委員長の判断が優先されることとする。  
   ・合格の場合は出力Ｙ（ＬＥＤ）がＯＮとなる。  
   ・入出力ともに正論理で表現する。  
   ・保持機能やリセット機能は不要とする。よって、４入力１出力の回路とする。
   1. この回路の真理値表を作りなさい。 （3点）
   2. 真理値表を元にカルノー図を作りなさい。 （3点）
   3. この回路を論理式で表しなさい。 （3点）
   4. 論理回路図を描きなさい。（多入力論理素子の使用可） （3点）
   5. 空欄を埋めてこの回路のＨＤＬ記述を完成させなさい。（ライブラリの宣言は省略してあります）

　 （各2点：計8点）

entity CONTEST is

port ( BTN : in std\_logic\_vector(3 downto 0); -- 投票ボタン

LED : out std\_logic); -- 出力LED

end CONTEST;

architecture RTL of CONTEST is

begin

process( ア BTN )

begin

イcase BTN is

when"0000" => LED <= '0';

when"0001" => LED <= '0';

when"0010" => LED <= '0';

when"0011" => LED <= '0';

when"0100" => LED <= '0';

when"0101" => LED <= '0';

when"0110" => LED <= '0';

when"0111" => LED <= '1';

when"1000" => LED <= ウ'0';

when"1001" => LED <= '1';

when"1010" => LED <= '1';

when"1011" => LED <= '1';

when"1100" => LED <= '1';

when"1101" => LED <= '1';

when"1110" => LED <= '1';

when"1111" => LED <= '1';

when others　エ => LED <= 'X';

end 　　イ case;

end process;

end RTL;

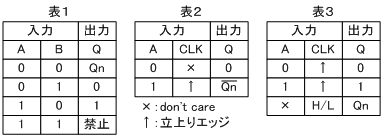
５．下表の（ア）～(エ)の空欄に入る語句をａ～ｇのうちから選び、表を完成させなさい。  
　　また、（オ）～（ク）に入る数値を記入しなさい。 （各2点：計16点）



a：２進数　b：5進数　c：6進数　d：8進数　ｅ：16進数　f：ＢＣＤコード　g：グレイコード

６．下記文中の括弧に入る語句をａ～より選び、記号で答えなさい。 （各2点：計20点）

論理回路は、（①）と（②）に大別することができる。（①）は、その時の入力の状態のみで出力が決まる回路で、以前の回路状態に依存しない。（②）はその時の入力の状態とそれ以前の状態で出力が決まる回路で、内部に記憶回路を有している。例えばカウンタ回路は（②）、（③）は（①）に分類される。

　（②）が有する記憶回路のことをフリップフロップ回路という。フリップフロップ回路にはいくつかの種類があり、それぞれの動作によって名称が異なる。下図の真理値表のような動作をするフリップフロップ回路を、それぞれ表１（④）、表２（⑤）、表３（⑥）と呼ぶ。

フリップフロップ回路は、その動作によって（⑦）と（⑧）に分類される。（⑦）のフリップフロップ回路は、入力される基準パルス（一般的にクロックと呼ぶ）が変化するタイミング（エッジ）で出力が変化し、（⑧）のフリップフロップ回路は、基準パルスのタイミングとは無関係に、入力信号が変化するタイミングで出力が変化する。表１は（⑧）のフリップフロップ回路、表２と表３は（⑦）のフリップフロップ回路である。

ＨＤＬを用いたデジタル回路設計においては、単一のクロックのタイミングで全ての回路が動作する（⑨）として設計する。ＰＬＤの内部で複数のクロックが用いられると、回路の動作タイミングの検証が困難となり、後段の回路において（⑩）が発生する原因になる。

ａ：単相同期回路 ｂ：非同期型 ｃ： クロストーク

ｄ：Dフリップフロップ ｅ：組み合わせ回路 ｆ： シフトレジスタ

ｇ：シーケンサ ｈ：Tフリップフロップ ｉ：スキュー

ｊ：同期型 ｊ：ジッタ ｋ：RSフリップフロップ

ｌ：デコーダ ｍ：順序回路 ｎ：ラッチ

※　使用しないものもあります

７．下図はPLDと７セグメントLEDの接続図です。このハードウェアを用いて、HDL（VHDL）による１桁のカウンタ回路を設計しました。以下の問いに答えなさい。

（HDLによる記述では、ライブラリの宣言は所略してあります） （計20点）

このカウンタ回路は、電源が入ると１秒間隔でカウントアップし、その数値を７セグメントＬＥＤに表示します。９までカウントしたら０に戻り、そのままカウントを継続します。

リセットボタンが押されると、カウント値が０になります。なお、システムクロックは50MHzとします。

1. Aの部分は何をしているのでしょうか。  
    点灯させる７セグメントＬＥＤを選択している。（６点）
2. Bの部分は何をしているのでしょうか。  
   　 １秒間隔のパルス信号を生成している。（６点）
3. Cの部分は何をしているのでしょうか。  
     
   　バイナリのカウント値を、７セグメントLEDに数値を表示するための信号に変換している。  
   （デコーダ回路） 　　（６点）
4. このカウンタ回路のリセット入力は同期リセットでしょうか。非同期リセットでしょうか。また、リセットボタンを押してから、どのタイミングでリセット機能が有効になるかを答えなさい。  
     
   プロセス文のセンシティビティリストにRESETがないので、クロック入力（CLK）に同期して機能する同期リセット入力である。50MHzのシステムクロックに同期した上で、B部で生成した１秒間隔のパルスに合わせてリセットが有効になる。 　　（10点）

entity COUNT is

port(CLK,RESET,UD : in std\_logic; -- UDは③の課題用｡それ以外では使用しない｡

SEG7\_LED : out std\_logic\_vector(7 downto 0);

SEG7\_SELECT : out std\_logic\_vector (3 downto 0));

end COUNT;

architecture RTL of COUNT is

signal COUNT: integer range 0 to 49999999;

signal DO : std\_logic;

signal WORK : std\_logic\_vector(3 downto 0);

begin

SEG7\_SELECT <= "1110";　　　・・・・・・・・・・・・・・・・・・・・・Ａ

process(CLK)

begin

if(CLK'event and CLK = '1') then

COUNT <= COUNT + 1;

end if;

end process;

process(COUNT) 　　 　　　　　　　　　　・・・・・・・・・・・・・・Ｂ

begin

if (COUNT = 49999999 ) then

DO <= '1';

else

DO <= '0';

end if;

end process;

process(CLK)

begin

if(CLK'event and CLK = '1')then

if(DO = '1') then

if ( RESET = '1' ) then

WORK <= "0000";

else

WORK <= WORK + '1';

end if;

end if;

end　if;

end process;

process(WORK)

begin

case WORK is

when "0000" => SEG7\_LED <= "00000011"; -- 0

when "0001" => SEG7\_LED <= "10011111"; -- 1

when "0010" => SEG7\_LED <= "00100101"; -- 2

when "0011" => SEG7\_LED <= "00001101"; -- 3

when "0100" => SEG7\_LED <= "10011001"; -- 4

when "0101" => SEG7\_LED <= "01001001"; -- 5 　　・・・Ｃ

when "0110" => SEG7\_LED <= "01000001"; -- 6

when "0111" => SEG7\_LED <= "00011011"; -- 7

when "1000" => SEG7\_LED <= "00000001"; -- 8

when "1001" => SEG7\_LED <= "00001001"; -- 9

when others => SEG7\_LED <= "XXXXXXXX";

end case;

end process;

end RTL;