Prova d'esame di
RETI LOGICHE (Prof. E. Torti)
Lauree in Ing. Elettronica e Informatica DM 270
27 giugno 2022

CANDIDATO: n. matricola:

1) Si semplifichi la seguente funzione booleana indicando chiaramente teoremi, proprietà e/o identità utilizzati. Implementare poi la funzione semplificata utilizzando un opportuno multiplexer.

$$F = \overline{\overline{(A+B)} + \overline{(A+C)} + \overline{(A\bar{B}C)}} + \bar{A}\bar{B}D + \bar{A}\bar{C}D + BD$$

2) Si progetti un full-adder, ricavando la tabella di verità, le mappe di Karnaugh e le espressioni algebriche. Utilizzare poi il full-adder come blocco elementare per progettare un sottrattore ad n bit con riporto. Applicare poi la tecnica della contrazione per progettare un circuito "decrementa di 8", utilizzando il numero minimo di bit, senza riporto in uscita.

3) Si descriva la struttura di un dispositivo FPGA. Si descrivano e confrontino poi le varie tecniche utilizzabili per produrre i bit di configurazione del dispositivo.

4) Si consideri un registro a caricamento parallelo. Si discuta la tecnica di gating del clock, descrivendo come viene implementata e quali problematiche comporta. Si descriva poi nel dettaglio una soluzione alternativa a tale tecnica.

5) Si progetti un circuito in grado di riconoscere la sequenza 1111 all'interno di una sequenza arbitraria di bit. Il circuito dovrà essere dotato di un ingresso X e di un'uscita Z. L'uscita Z dovrà assumere il valore 1 tutte le volte che all'ingresso X sono già stati applicati i valori 111 e l'ingresso corrente vale 1. In tutti gli altri casi l'uscita Z deve valere 0. Si preveda anche la presenza di un segnale di Reset asincrono che riporti il circuito nello stato iniziale. Si utilizzino sia la codifica Gray che la codifica one-hot per assegnare codici agli stati e si utilizzino solo flip-flop di tipo D e si disegni il dettaglio circuitale di come andrebbe collegato il segnale di reset asincrono nei due differenti casi. Quanto vale il costo degli ingressi del circuito, per ciascuna delle due codifiche, considerando il costo di ogni singolo flip-flop pari a 14 (ignorare il costo delle porte NOT)?
   Esempio di sequenza da riconoscere: X: 0 0 0 1 1 1 1 1 1 0 1 0
   uscita Z: 0 0 0 0 0 0 1 1 1 0 0 0

6) Si progetti un contatore in grado di generare la sequenza 0, 1, 3, 2, 4, 6 utilizzando solo flip-flop JK. Si considerino gli stati non utilizzati come condizioni di non specificazione. Si modifichino le equazioni per includere nel sistema un segnale di ENABLE ti tipo attivo basso. Si disegni il diagramma di stato del contatore, andando ad indicare cosa succederebbe qualora si presentassero in uscita combinazioni considerate come condizioni di non specificazione. Si calcoli il costo del circuito assumendo che ogni flip-flop abbia un costo pari a 14. Come andrebbero gestite le combinazioni considerate come condizioni di non specificazione qualora ci si trovasse di fronte ad una applicazione reale?

7) Si consideri il seguente codice VHDL che descrive una ALU. Commentare sinteticamente gli statement del codice (riportare sul foglio il numero della riga seguito dal relativo commento). Spiegare poi come andrebbe aggiunta una porta di ENABLE attivo basso. Modificare il codice in modo da introdurre una porta OVF che indichi il verificarsi della situazione di overflow.

Per le operazioni in cui non esiste overflow, la porta dovrà porre in uscita il valore ‘X’. Si aggiunga infine una porta di SET attivo basso con meno priorità rispetto al segnale di RESET e che ponga tutti i bit di uscita pari ad 1.

```
1:  library ieee;
2:  use ieee.std_logic_1164.all;
3:  use ieee.numeric_std.all;

4:  entity ALU is
5:          port(
6:                  A,B : in std_logic_vector(3 downto 0);
7:                  op : in std_logic_vector(1 downto 0);
8:                  clk, reset : in std_logic;
9:                  R   : out std_logic_vector(3 downto 0)
10:                 );

11: end ALU;

12: architecture arch of ALU is
13:         signal res : signed(3 downto 0);
14:         begin
15:                 process(clk,reset)
16:                 begin
17:                         if(reset='1') then
18:                                 res <= “0000”;
19:                         elsif(rising_edge(clk)) then
20:                                 if(op=”00”) then
21:                                         res <= signed(A) + signed(B);
22:                                 elsif(op=”01”) then
23:                                         res <= signed(A) + signed(not(B)) + 1;
24:                                 elsif(op=”10”) then
25:                                         res <= A and B;
26:                                 elsif(op=”11”) then
27:                                         res <= A or B;
28:                                 else
29:                                         res <= “XXXX”;
30:                                 end if;
31:                         end if;
32:                 end process;
33:                 R <= std_logic_vector(res(3 downto 0));
34: end arch;
```