Σχεδίαση Ψηφιακών Κυκλωμάτων

Ανασκόπηση του μαθήματος και θέματα πρακτικής εξάσκησης

Νικόλαος Καββαδίας
nkavv@uop.gr

26 Ιανουαρίου 2011

Σκιαγράφηση της διάλεξης

- Αναδρομή στο περιεχόμενο του μαθήματος
- Ενδεικτικά θέματα εξετάσεων (θεωρία και ασκήσεις)
- Θέματα για εξάσκηση (θεωρία και ασκήσεις)

Σύνοψη του μαθήματος

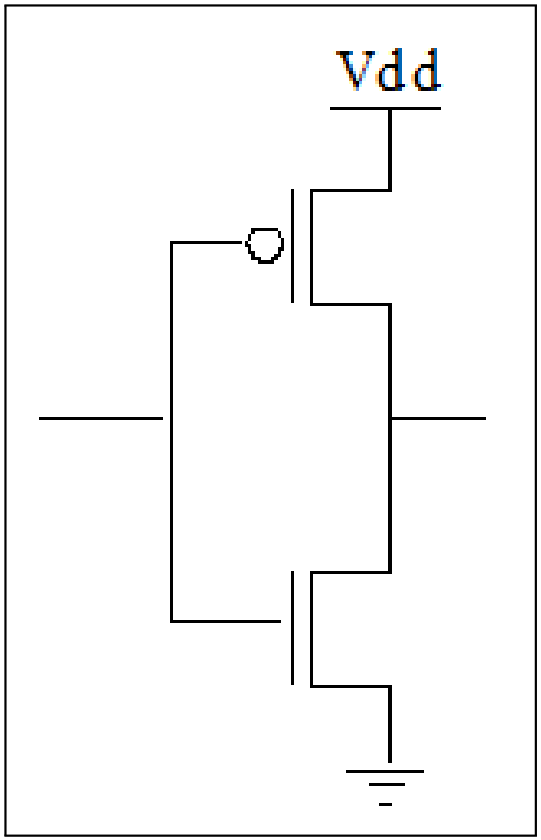
- Διαλέξεις
  1. Η τεχνολογία VLSI - Διεργασίες CMOS
  2. Συνδυαστική και ακολουθιακή λογική
  3. Αριθμητικά κυκλώματα και μνήμες
  4. Εισαγωγή στη γλώσσα περιγραφής υλικού VHDL: Μέρος 1
  5. Εισαγωγή στη γλώσσα περιγραφής υλικού VHDL: Μέρος 2
  6. Η αρχιτεκτονική οργάνωση των FPGA
  7. Οι αρχιτεκτονικές FPGA Xilinx Spartan-3 και Virtex-5
  8. Η φυσική σχεδίαση των FPGA
  9. Μεθοδολογίες σχεδίασης και η ροή λογικής σύνθεσης κυκλωμάτων σε FPGA
- Ιστότοπος του μαθήματος:
  `http://eclass.uop.gr/courses/CST326/`

Δ1: Η τεχνολογία VLSI - Διεργασίες CMOS

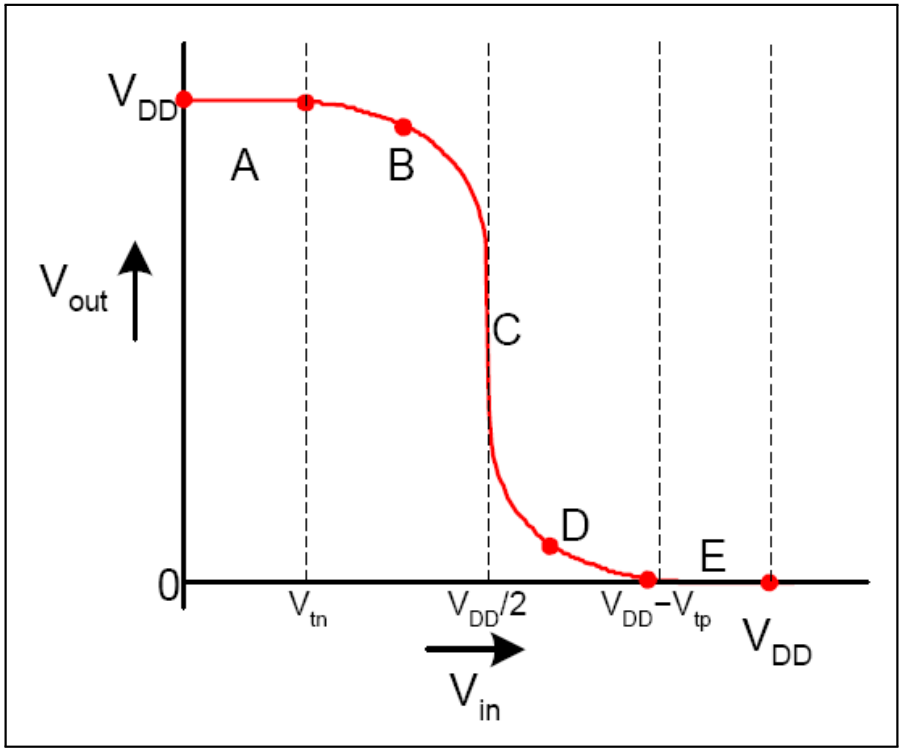
- Προκλήσεις και προβλήματα στην ψηφιακή σχεδίαση
- Μετρικά και κόστη στη σχεδίαση ICs (Integrated Circuits)
- Ο αντιστροφέας CMOS - Χαρακτηριστική V_{in}/V_{out} - Περιοχές λειτουργίας
- Κατανάλωση ισχύος σε κυκλώματα CMOS
- Δραστηριότητα μεταβάσεων κόμβου
- Ενδεικτικά θέματα
  1) Τι είναι η δραστηριότητα μεταβάσεων κόμβου; Δώστε την εξίσωση υπολογισμού της δυναμικής κατανάλωσης ισχύος που οφείλεται σε αυτή.
  2) Αναφέρετε τις περιοχές λειτουργίας και σχεδιάστε τη χαρακτηριστική μεταφοράς τάσης του αντιστροφέα CMOS.

Ο αντιστροφέας CMOS

- Κυκλωματική σχεδίαση του αντιστροφέα CMOS
- Τα τρανζίστορ NMOS και PMOS σε διακοπτική λειτουργία



Η χαρακτηριστική μεταφοράς τάσης και οι περιοχές λειτουργίας του αντιστροφέα CMOS



Περιοχή	NMOS	PMOS	Συνθήκη
A	Αποκοπή	Γραμμική	$0 < V_{in} < V_{Tn}$
B	Κόρος	Γραμμική	$V_{Tn} \leq V_{in} < \frac{V_{DD}}{2}$
C	Κόρος	Κόρος	$V_{in} \simeq \frac{V_{DD}}{2}$
D	Γραμμική	Κόρος	$\frac{V_{DD}}{2} < V_{in} < V_{DD} - V_{Tp}$
E	Γραμμική	Αποκοπή	$V_{DD} - V_{Tp} \leq V_{in} \leq V_{DD}$

Δραστηριότητα μεταβάσεων κόμβου (node transition activity)

- Αποτελεί τη συχνότητα των μεταβάσεων ανάμεσα σε δύο δοθείσες λογικές καταστάσεις σε έναν κυκλωματικό κόμβο
- Δίνεται από το λόγο του αριθμού των μεταβάσεων στον κόμβο ως προς τον συνολικό αριθμό των αλλαγών στις τιμές εισόδου που επιδρούν στον κόμβο
- Συμβολίζεται με α και υπολογίζεται για τις μεταβάσεις $0 \rightarrow 1$ ($\alpha_{0\rightarrow1}$) και $1 \rightarrow 0$ ($\alpha_{1\rightarrow0}$)
- Η ενέργεια λόγω της μεταγωγής μιας πύλης CMOS για N κύκλους ρολογιού δίνεται από τη σχέση $E_N = C_L \cdot V_{dd}^2 \cdot n(N)$
- $n(N)$ είναι ο αριθμός των μεταβάσεων $0 \rightarrow 1$ σε N κύκλους ρολογιού
- Για τη μέση δυναμική ισχύ έχουμε:

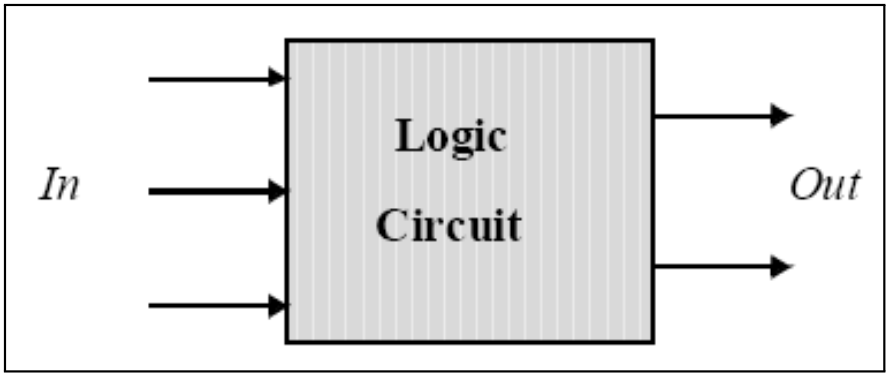
$$\begin{aligned} P_{avg} &= \lim_{N\rightarrow\infty} \frac{E_N}{N} \cdot f_{clk} \\ &= (\lim_{N\rightarrow\infty} \frac{n(N)}{N}) \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} \\ &= \alpha_{0\rightarrow1} \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} \end{aligned}$$

Δ2: Συνδυαστική και ακολουθιακή λογική

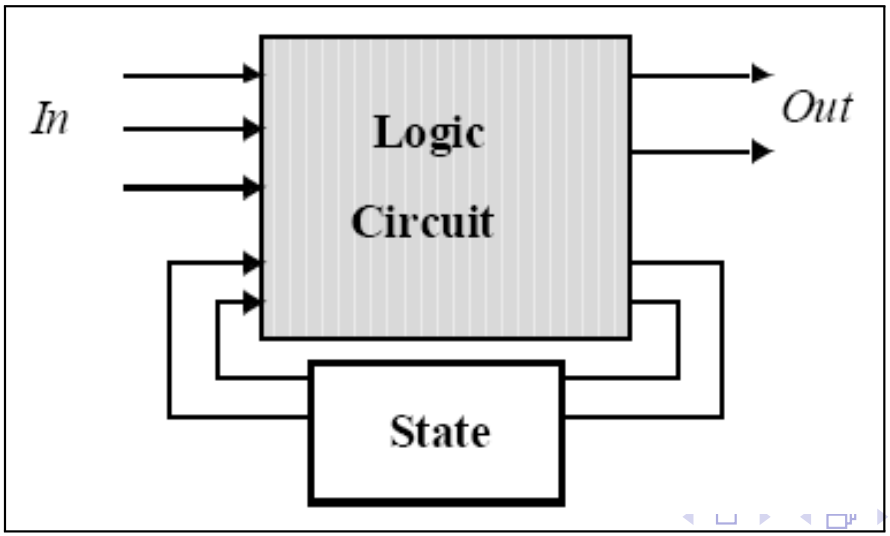
- Ιδιότητες των κυκλωμάτων CMOS
- Λογική CMOS - Σχεδιασμός PUN και PDN
- Πολυπλέκτης και τρισταθής απομονωτής
- Σχεδιασμός/λειτουργία D-ff και D-latch
- Διάφοροι ορισμοί στα χρονικά διαγράμματα συνδυαστικών και ακολουθιακών κυκλωμάτων
- Ενδεικτικά θέματα
  1) Σχεδιάστε κυκλώματα επιπέδου τρανζίστορ για τις εξής λογικές συναρτήσεις:
     $E = (A + \overline{B}) \cdot \overline{C} + D$
     $F = A \cdot B + B \cdot C + A \cdot C$
  2) Περιγράψτε τη λειτουργία του D flip-flop.

Συνδυαστική και ακολουθιακή λογική

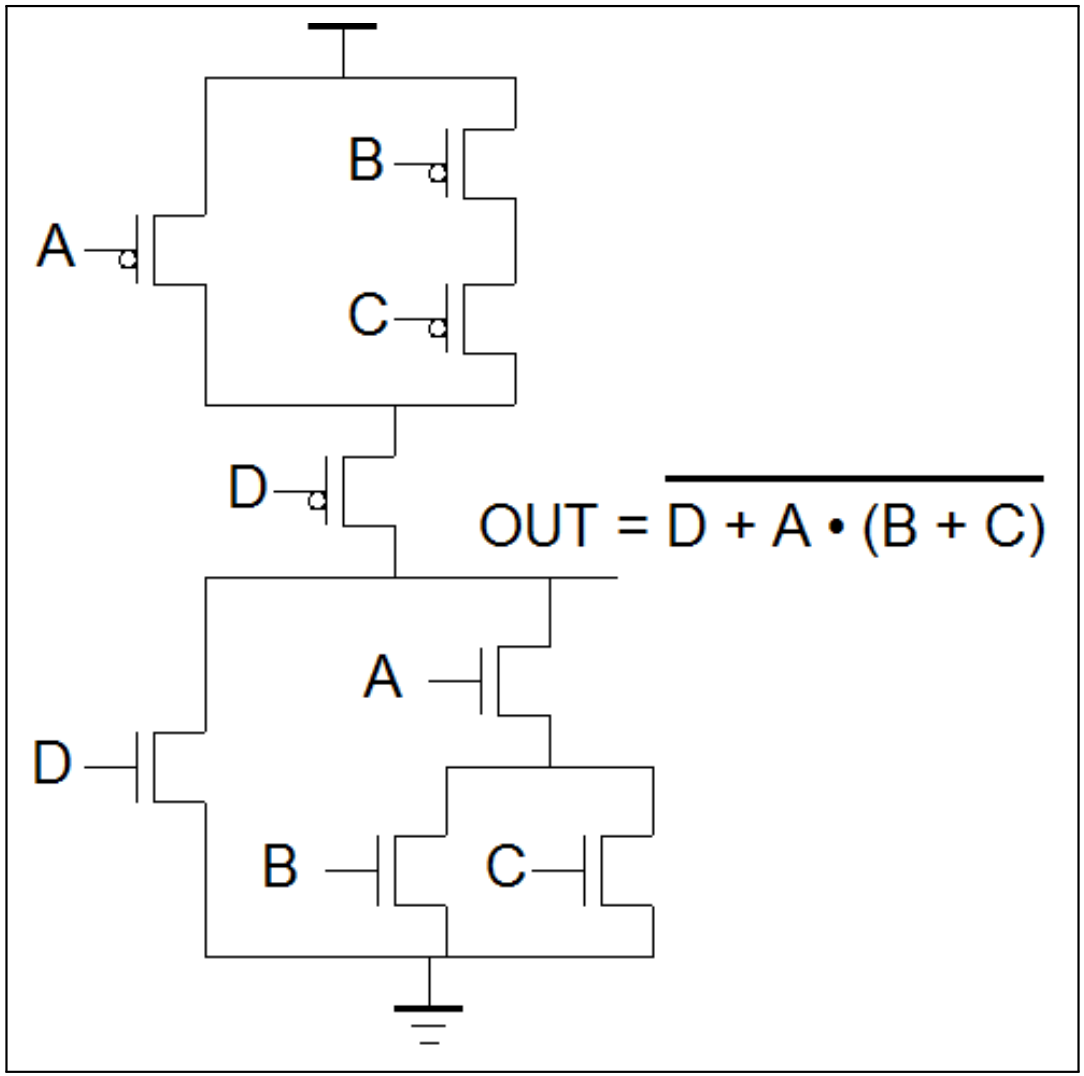
- Συνδυαστική λογική: οι έξοδοι αποτελούν συνάρτηση μόνο των εισόδων



- Ακολουθιακή λογική: οι έξοδοι αποτελούν συνάρτηση των εισόδων και της τρέχουσας κατάστασης (state) η οποία εξαρτάται από προηγούμενες εισόδους

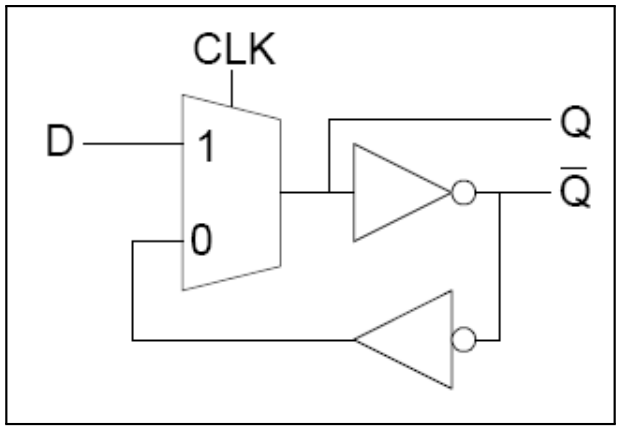


Παράδειγμα: Σύνθετη πύλη



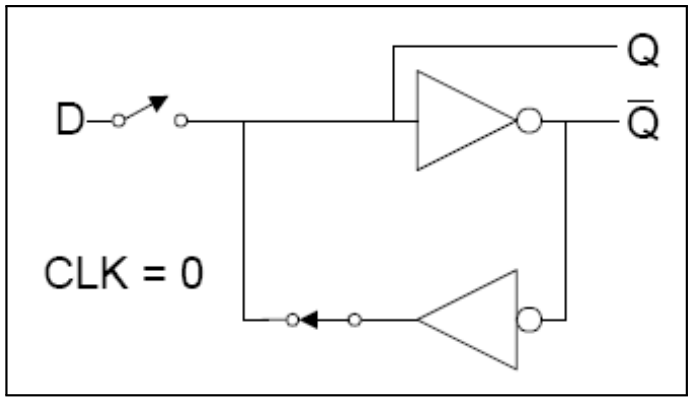
Σχεδιασμός και λειτουργία του μανδαλωτή D

Σχεδιασμός του D-latch

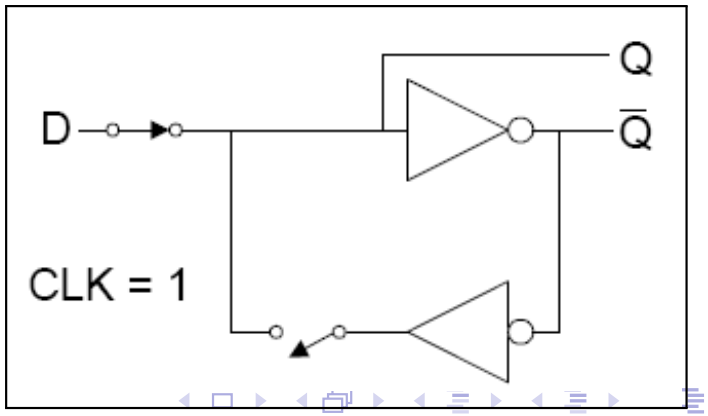


Λειτουργία

Για CLK = 0

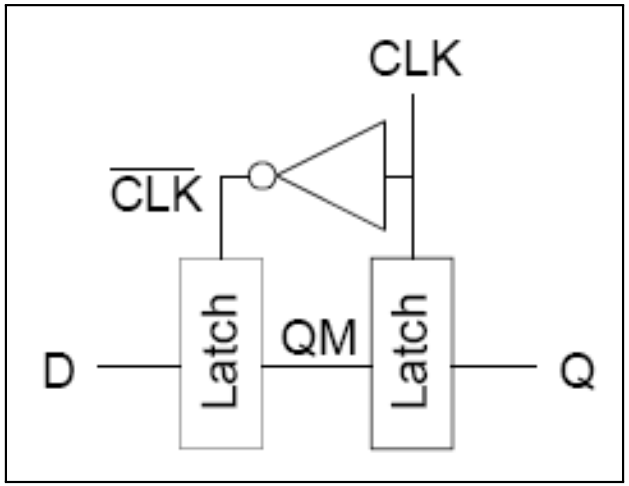


Για CLK = 1



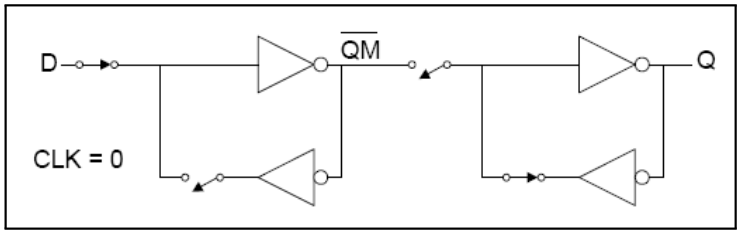
Σχεδιασμός και λειτουργία του D flip-flop

Σχεδιασμός του D flip-flop

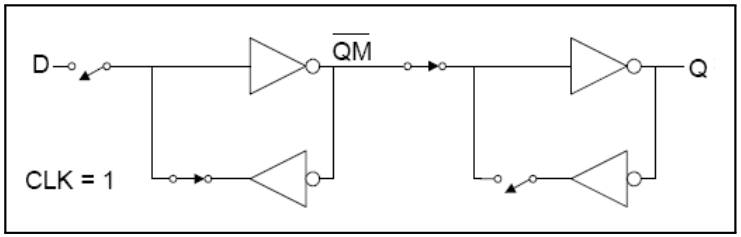


Λειτουργία

Για CLK = 0



Για CLK = 1



Δ3: Αριθμητικά κυκλώματα και μνήμες

- Πλήρης αθροιστής, RCA, CLA, πολυπλοκότητα χώρου και χρόνου
- Πολλαπλασιασμός: γενική μορφή, Baugh-Wooley (τύπου πίνακα)
- Συγκριτές
- Μνήμη ROM και υλοποίηση, γενικά στοιχεία για μνήμες RAM
- Ενδεικτικά θέματα
  1) Να σχεδιαστεί συγκριτής ισότητας ($A = B$) και ανισοισότητας ($A \geq B, A \leq B$) ο οποίος βασίζεται σε δομή αφαίρεσης και χρησιμοποιεί τον υπολογισμό των σημαιών Z (μηδενικό), C (κρατούμενο) και V (υπερχείλιση).
  2) Να σχεδιαστεί μνήμη ROM 4 λέξεων των 5 bit με τα εξής περιεχόμενα: `11011, 00001, 10000, 10101`

Δ4-Δ5: Η γλώσσα περιγραφής υλικού VHDL (1)

- Θεμελιώδεις γνώσεις της VHDL
  - `ENTITY, ARCHITECTURE, COMPONENT, PACKAGE, PORT`
- Συχνά χρησιμοποιούμενα πακέτα (`std_logic_1164, std_logic_unsigned`)
- `CONSTANT, VARIABLE, SIGNAL` - Δηλώσεις και αναθέσεις
- Τελεστές της VHDL
- Δομές ακολουθιακού κώδικα
  - `PROCESS`, λίστα ευαισθησίας
  - Δηλώσεις ελέγχου: `IF-ELSIF-ELSE, CASE-WHEN`
  - Δηλώσεις επανάληψης: `LOOP: FOR scheme or WHILE scheme`
  - Περιγραφή ακολουθιακών κυκλωμάτων (χρήση `clock, reset`)
- Δομές συντρέχοντος κώδικα
  - `WHEN-ELSE, WITH-SELECT`
- Συνένωση σημάτων (concatenation)

Δ4-Δ5: Η γλώσσα περιγραφής υλικού VHDL (2)

- Συναρτήσεις (`FUNCTION`) και διαδικασίες (`PROCEDURE`)
- Παραμετρικές περιγραφές με `IF-GENERATE` και `FOR-GENERATE`
- Γενικές σταθερές `GENERIC`
- Απαριθμητοί τύποι δεδομένων
- Πίνακες - Δήλωση και ανάθεση
- Κανόνες για τη σύνταξη συνθέσιμων περιγραφών
- Σχεδίαση μνήμης ROM (synchronous/asynchronous read)
- Σχεδίαση μνήμης RAM (synchronous/asynchronous read)
- Δήλωση `ASSERT`
- Περιγραφές ελέγχου/επαλήθευσης (testbenches)
- Όλα τα κυκλώματα των παραδειγμάτων (`vhdl4dcd.zip`)
- Μεθοδολογία σχεδίασης FSM
- Μεθοδολογία σχεδίασης FSMD

Δ4-5: Ενδεικτικά θέματα (1)

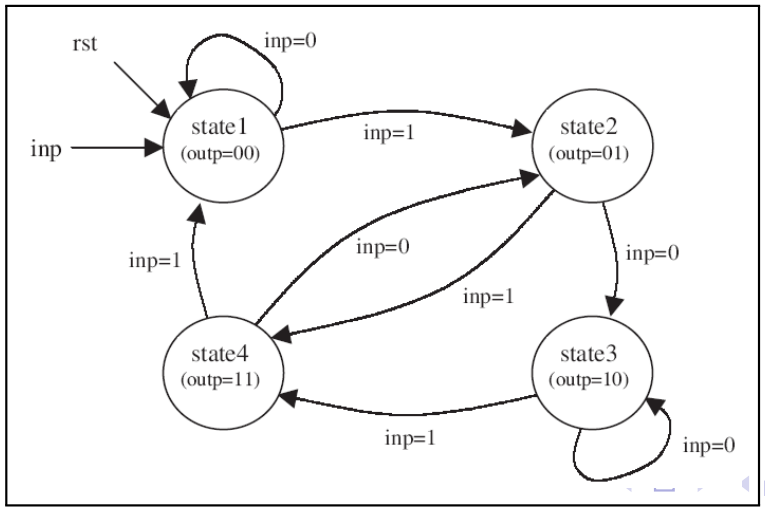
1. Τι περιγράφεται στην `ENTITY` ενός κυκλώματος·
   Α. οι μηχανισμοί λειτουργίας του Β. η διεπαφή του
   Γ. ο πίνακας αληθείας του Δ. τα `COMPONENT` από τα οποία αποτελείται
2. Τι περιγράφεται στην ARCHITECTURE ενός κυκλώματος·
   Α. η διεπαφή του Β. ο πίνακας αληθείας του
   Γ. τα COMPONENT που το συνιστούν Δ. οι μηχανισμοί λειτουργίας του
3. Επιλέξτε την ισοδύναμη έκφραση για την αρνητική ακμοπυροδότηση:
   `falling_edge(clk)`
   Α. `clk'STABLE and clk = '1'` Β. `clk'EVENT and clk = '0'`
   Γ. `clk'EVENT and clk = '1'` Δ. `clk = '0' or clk'EVENT`
4. Τι είναι ένα αρχείο `testbench`·
   Α. Το top-level αρχείο του κυκλώματος
   Β. Αρχείο για τον έλεγχο του κυκλώματος
   Γ. Ένα πακέτο με δηλώσεις του χρήστη
   Δ. Εναλλακτική περιγραφή του κυκλώματος
5. Μία `VARIABLE` δεν μπορεί:
   Α. Να διασυνδέσει δύο αντίτυπα `COMPONENT` (υποκυκλώματα)
   Β. Να χρησιμοποιηθεί μέσα σε μία `PROCESS`
   Γ. Να διαβαστεί και να γραφεί μέσα στην ίδια `PROCESS`
   Δ. Να δηλωθεί στην περιοχή δηλώσεων της `PROCESS`

Δ4-5: Ενδεικτικά θέματα (2)

1. Να γραφεί ο κώδικας VHDL κυκλώματος πλειοψηφίας (majority voter) τριών εισόδων του 1-bit.
2. Να γραφεί ο κώδικας VHDL για την αρχιτεκτονική κυκλώματος αντιμετάθεσης ψηφίων (bit swapper) για διανύσματα των 8-bit. Το κύκλωμα θα υλοποιηθεί με χρήση της δήλωσης FOR...GENERATE και έχει την παρακάτω ENTITY.

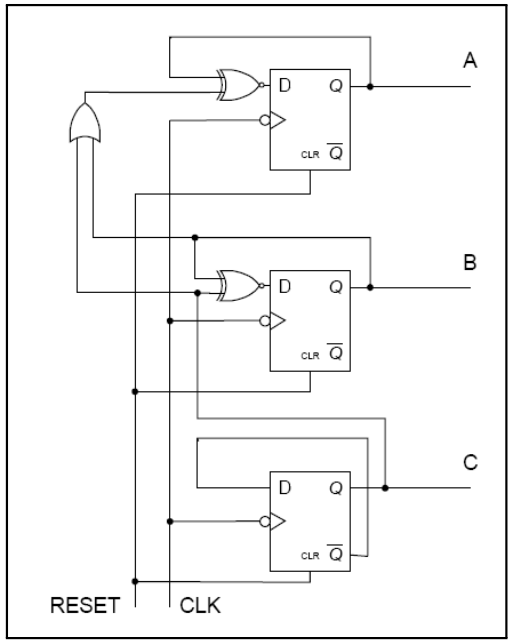
```
ENTITY bit_swapper IS
  PORT (
    din  : IN  std_logic_vector(7 DOWNTO 0);
    dout : OUT std_logic_vector(7 DOWNTO 0)
);
END bit_swapper;
```

3. Να σχεδιαστεί το FSM με το διάγραμμα καταστάσεων του σχήματος σε VHDL.



Δ4-5: Ενδεικτικά θέματα (3)

4. Δίνεται το κύκλωμα του σχήματος. Να γραφεί ο αντίστοιχος κώδικας VHDL. Για την υλοποίηση του D flip-flop χρησιμοποιήστε την απάντησή σας στο προηγούμενο υποερώτημα.



5. Θεωρήστε ότι το παραπάνω κύκλωμα δέχεται είσοδο `RESET = '1'` κατά την πρώτη περίοδο λειτουργίας του, και στη συνέχεια είναι `RESET = '0'`. Δώστε τις τιμές των `A, B, C` για τις πρώτες 11 περιόδους λειτουργίας του κυκλώματος. Σχολιάστε τα αποτελέσματα.

Επίλυση της άσκησης 1

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

entity majority_voter is
  port (
    in1  : in  std_logic;
    in2  : in  std_logic;
    in3  : in  std_logic;
    outp : out std_logic
  );
end majority_voter;

architecture comb of majority_voter is
  signal temp : std_logic_vector(2 downto 0);
begin
  process (in1, in2, in3)
  begin
    temp <= in1 & in2 & in3;
    case temp is
      when "011" | "101" | "110" | "111" => outp <= '1';
      when others                        => outp <= '0';
    end case;
  end process;
end comb;
```

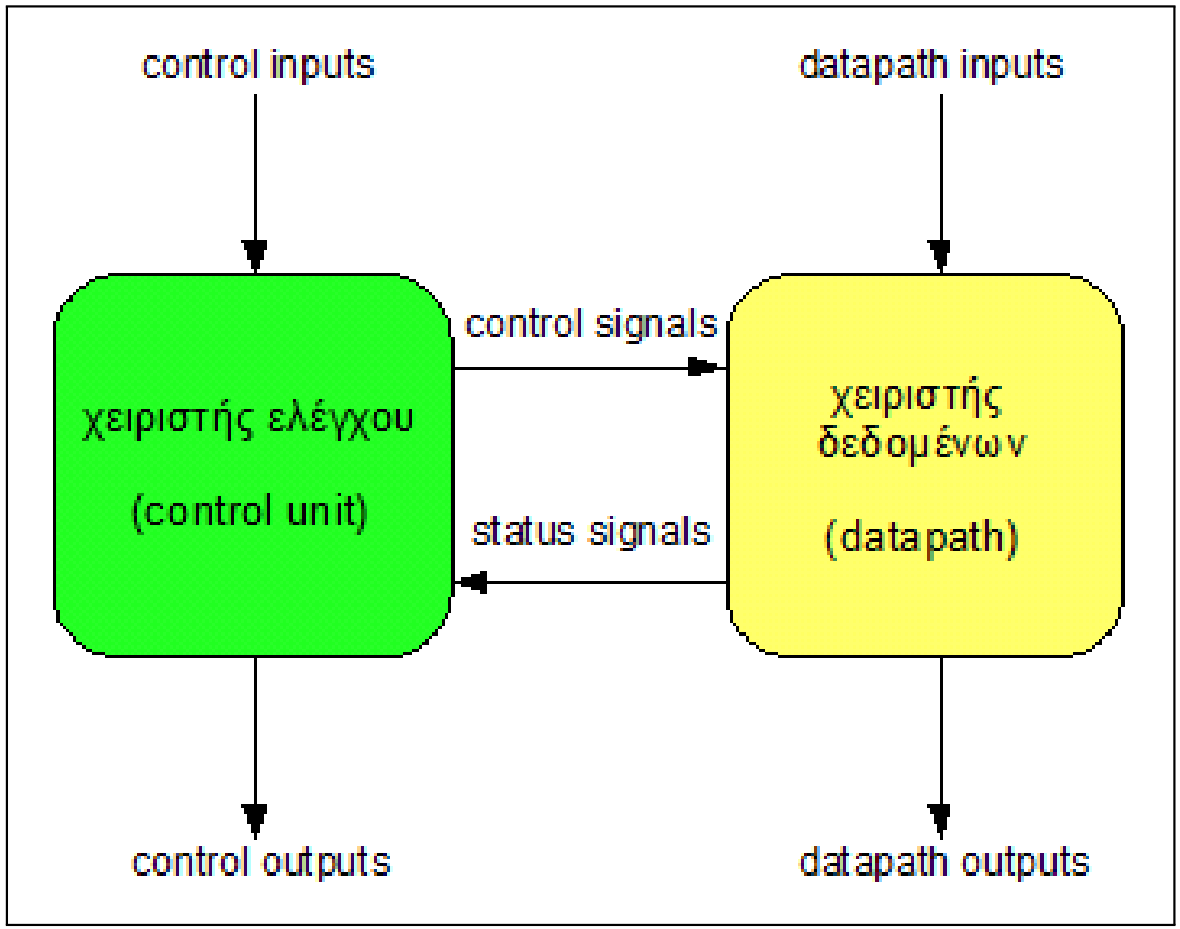
Επίλυση της άσκησης 4

```
library IEEE;
use IEEE.std_logic_1164.all;

entity dcounter_3b is
  port (
    clk, rst : in  std_logic;
    a, b, c  : out std_logic
 );
end dcounter_3b;

architecture gatelevel of dcounter_3b is
  component dff
    port (
      clk   : in std_logic;
      rst   : in std_logic;
      d     : in std_logic;
      q     : out std_logic;
      q_n   : out std_logic);
  end component;
  signal q_a, q_b, q_c : std_logic;
  signal q_n_a, q_n_b, q_n_c : std_logic;
  signal t0, d_a, d_b : std_logic;
begin
```

```
  U_dff_A: dff port map (
      clk => clk,   rst => rst,
      d   => d_a,   q   => q_a,
      q_n => q_n_a
    );

  U_dff_B: dff port map (
      clk => clk,   rst => rst,
      d   => d_b,   q   => q_b,
      q_n => q_n_b
    );

  U_dff_C: dff port map (
      clk => clk,   rst => rst,
      d   => q_n_c, q   => q_c,
      q_n => q_n_c
    );

  t0  <= q_b or q_c;
  d_a <= q_a xnor t0;
  d_b <= q_b xnor q_c;

  a   <= q_a;
  b   <= q_b;
  c   <= q_c;
end gatelevel;
```

Η οργάνωση ενός μη-προγραμματιζόμενου επεξεργαστή

- Γενικό σχηματικό διάγραμμα ενός μη προγραμματιζόμενου επεξεργαστή



Το πρόβλημα του μέγιστου κοινού διαιρέτη δύο αριθμών

- Δεχόμαστε ότι: $gcd(n, 0) = gcd(0, n) = gcd(0, 0) = 0$
- Εύρεση αριθμού m ο οποίος να είναι ο μεγαλύτερος θετικός ακέραιος ο οποίος διαιρεί και τους δύο αριθμούς
- Στα αρχαία Ελληνικά μαθηματικά αντιπροσωπεύει το πρόβλημα εύρεσης κοινής αναφοράς για τη μέτρηση δύο ευθύγραμμων τμημάτων
- Αλγόριθμος:

```
int gcd(int a, int b) {
  int result, x, y;
  x = a; y = b;
  if (x!=0 && y!=0) {
    while (x != y) {
      if (x >= y) {
        x = x - y;
      } else {
        y = y - x;}
    }
    result = x;
  } else {
    result = 0;
  }
  return (result);
}
```

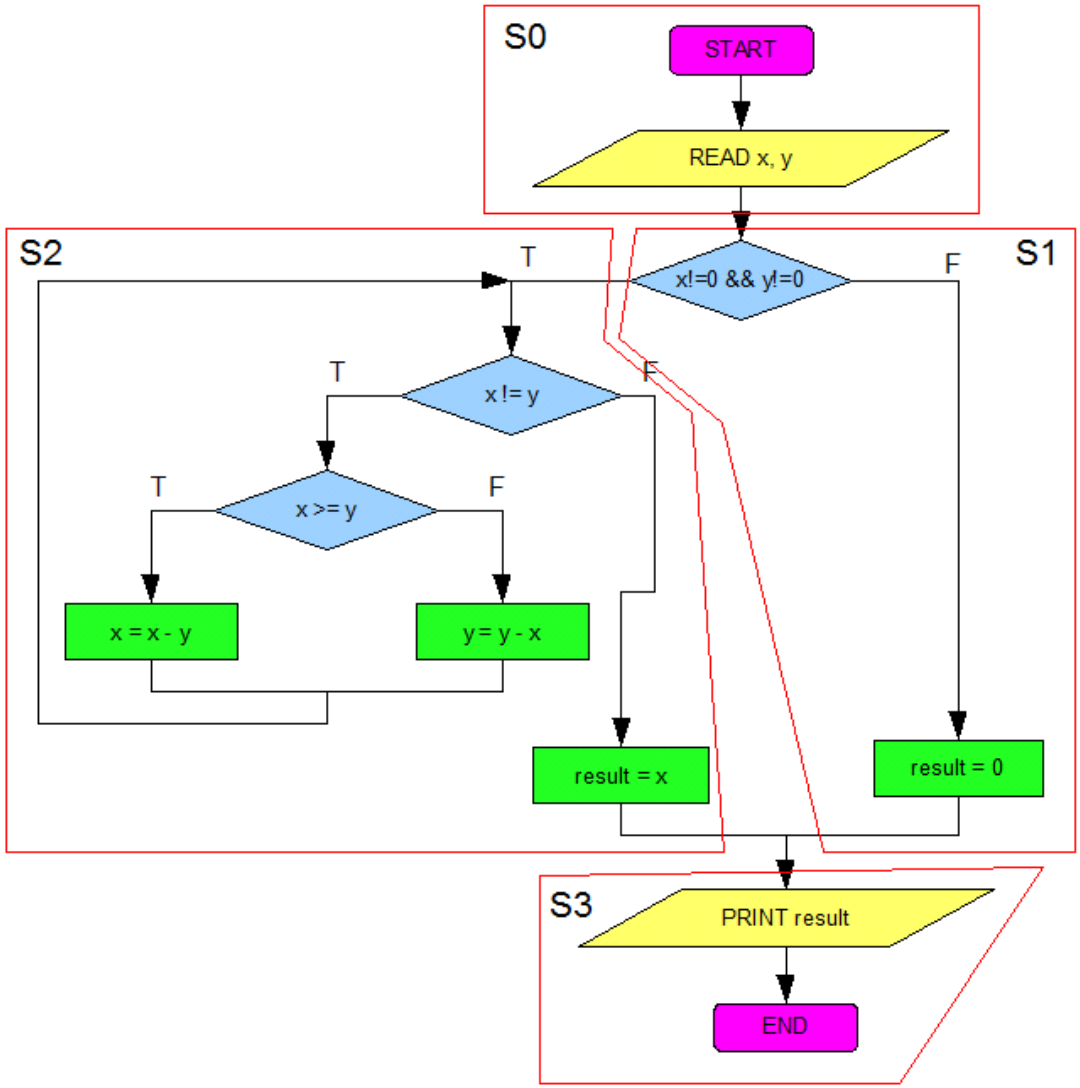
Αριθμητικό παράδειγμα υπολογισμού του GCD

- Ο μικρότερος αριθμός από δύο αριθμούς a, b αφαιρείται από τον μεγαλύτερο σε διαδοχικά βήματα
- Όταν οι δύο αριθμοί γίνουν ίσοι, τότε ισούνται με το Μέγιστο Κοινό Διαιρέτη τους
- Σε περίπτωση που η διαδικασία φτάσει μέχρι το σημείο που $a = 1$ ή $b = 1$ τότε οι δύο αριθμοί δεν έχουν μη τετριμμένο GCD, δηλαδή μεγαλύτερο του 1
- Παράδειγμα ($a = 196, b = 42$)

Βήμα	A	B
1	196	42
2	154	42
3	112	42
4	70	42
5	28	42
6	28	14
7	14	14

- Το αποτέλεσμα είναι: $gcd(196, 42) = 14$

Αλγοριθμικό διάγραμμα ροής για τον αλγόριθμο GCD



Περιγραφή της υλοποίησης FSMD του επεξεργαστή GCD (1)

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

entity gcd is
  generic (
    WIDTH : integer
  );
  port (
    clock : in std_logic;
    reset : in std_logic;
    start : in std_logic;
    a     : in std_logic_vector(WIDTH-1 downto 0);
    b     : in std_logic_vector(WIDTH-1 downto 0);
    outp  : out std_logic_vector(WIDTH-1 downto 0);
    done  : out std_logic
  );
end gcd;

architecture fsmd of gcd is
  type state_type is (s0,s1,s2,s3);
  signal state: state_type;
  signal x, y, res : std_logic_vector(WIDTH-1 downto 0);
begin
```

Περιγραφή της υλοποίησης FSMD του επεξεργαστή GCD (2)

```
  process (clock, reset)
  begin
    done <= '0';
--
    if (reset = '1') then
      state <= s0;
      x <= (others => '0');
      y <= (others => '0');
      res <= (others => '0');
    elsif (clock='1' and clock'EVENT) then
      case state is
        when s0 =>
          if (start = '1') then
            x <= a;
            y <= b;
            state <= s1;
          else
            state <= s0;
          end if;
        when s1 =>
          if (x /= 0 and y /= 0) then
            state <= s2;
          else
            res <= (others => '0');
            state <= s3;
          end if;
```

Περιγραφή της υλοποίησης FSMD του επεξεργαστή GCD (3)

```
        when s2 =>
          if (x > y) then
            x <= x - y;
            state <= s2;
          elsif (x < y) then
            y <= y - x;
            state <= s2;
          else
            res <= x;
            state <= s3;
          end if;
        when s3 =>
          done <= '1';
          state <= s0;
      end case;
    end if;
  end process;

  outp <= res;

end fsmd;
```

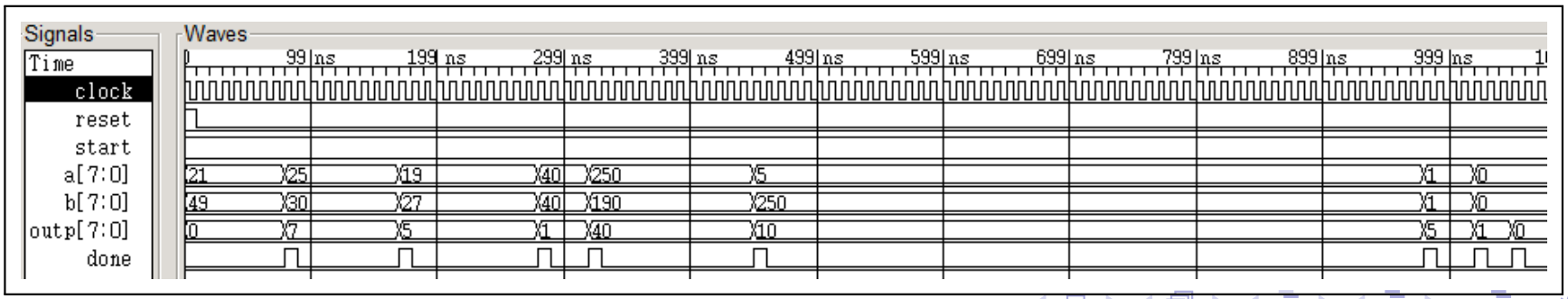
Προσομοίωση του επεξεργαστή GCD

- Δηλώσεις FILE για λήψη εισόδων από αρχείο και εκτύπωση διαγνωστικής εξόδου σε αρχείο

```
file TestDataFile: text open read_mode is "gcd_test_data.txt";
file ResultsFile: text open write_mode is "gcd_alg_test_results.txt";
```

- Περιεχόμενα του "gcd_test_data.txt" (A, B, *result*)

```
21 49 7
25 30 5
19 27 1
40 40 40
250 190 10
5 250 5
1 1 1
0 0 0
```



Δ6: Η αρχιτεκτονική οργάνωση των FPGA

- Οι προγραμματιζόμενες συσκευές και τα χαρακτηριστικά τους
- Τι είναι FPGA - Λογικό κύτταρο
- Αντίστροφη μηχανική λογικού κυττάρου (π.χ. Xilinx XC3000)
- Τυπικές ιδιότητες των CLB
- Ενδεικτικά θέματα
  1) Αναφέρετε γενικά χαρακτηριστικά των προγραμματιζόμενων συσκευών.

Δ7: Οι αρχιτεκτονικές FPGA Xilinx Spartan-3 και Virtex-5

- Αρχιτεκτονικά χαρακτηριστικά των συσκευών Xilinx Spartan-3
- Σχεδίαση μνήμης ROM και RAM (γενικά)
- Τρόποι χρήσης ενσωματωμένων πολλαπλασιαστών
- Αρχιτεκτονικά χαρακτηριστικά των συσκευών Xilinx Virtex-5
- Ο ενσωματωμένος χειριστής δεδομένων DSP48E
- Τρόποι χρήσης της μονάδας DSP48E

Δ7: Ενδεικτικά θέματα (1)

1) Αναφέρετε τα αρχιτεκτονικά χαρακτηριστικά των συσκευών FPGA Xilinx Spartan-3.
2) Αναφέρετε τα αρχιτεκτονικά χαρακτηριστικά των συσκευών FPGA Xilinx Virtex-5.
3) Θεωρήστε ότι διαθέτετε μία βιβλιοθήκη μονάδων RTL η οποία απαρτίζεται από αθροιστές (`ADD`), αφαιρέτες (`SUB`) και αριστερούς (`SHL`) και δεξιούς (`SHR`) λογικούς ολισθητές κατά σταθερή ποσότητα n. Ζητείται να σχεδιαστεί το σχηματικό διάγραμμα μονάδας για τον πολλαπλασιασμό μιας εισόδου x με τις σταθερές 5, 33, και 77. Η μονάδα θα διαθέτει τις αντίστοιχες εξόδους u, v, w. Σημειώνεται ότι η ολίσθηση κατά n θέσεις αριστερά ισοδυναμεί με πολλαπλασιασμό με το 2^n και η ολίσθηση κατά n θέσεις δεξιά, με διαίρεση με το 2^n. Αναφέρετε γενικά χαρακτηριστικά των προγραμματιζόμενων συσκευών.

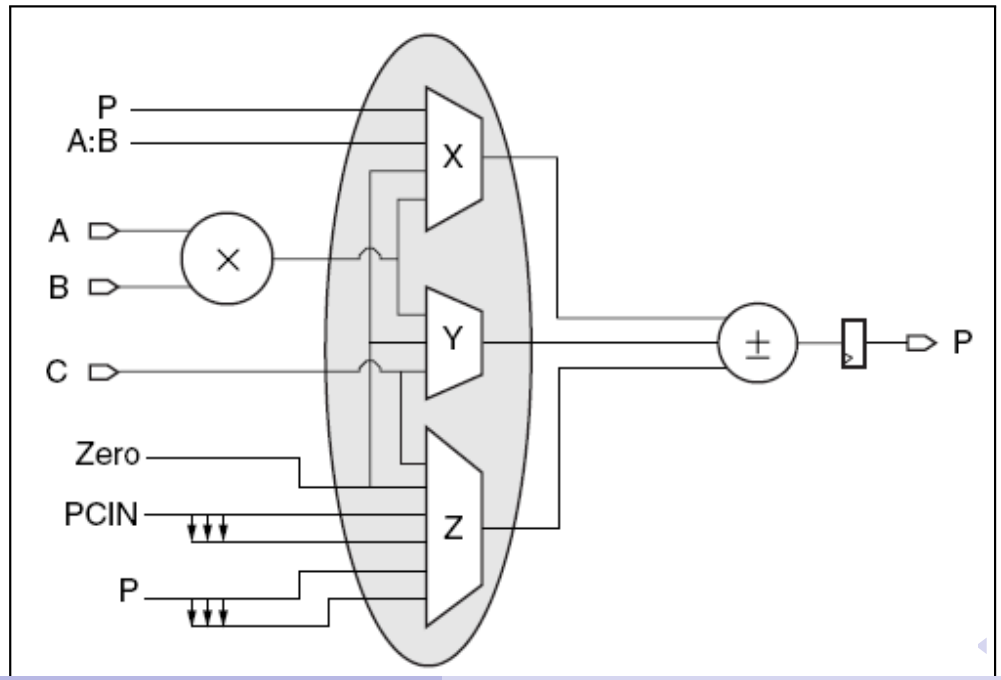
Δ7: Ενδεικτικά θέματα (1)

4) Θεωρήστε το απλοποιημένο σχηματικό διάγραμμα της μονάδας Virtex-4 DSP48 του παρακάτω σχήματος. Ζητείται να σχεδιαστεί πολλαπλασιαστής μιγαδικών αριθμών, ακρίβειας πραγματικού και φανταστικού μέρους των 18-bit, με μονάδες DSP48. Αιτιολογήστε πόσες μονάδες θα χρειαστείτε και στη συνέχεια δώστε το αντίστοιχο σχηματικό διάγραμμα.

ΣΗΜΕΙΩΣΗ: Έστω οι μιγαδικοί αριθμοί $z_1 = a + i \cdot b = (a, b)$ και

$z_2 = c + i \cdot d = (c, d)$. Το γινόμενο τους z_3 δίνεται από τη σχέση:

$z_3 = z_1 \times z_2 = (ac - bd) + i \cdot (ad + bc) = (ac - bd, ad + bc)$



Δ8: Η φυσική σχεδίαση των FPGA

- Γνωρίσματα των επαναδιαμορφώσιμων αρχιτεκτονικών
- Γενικά για τους προγραμματιζόμενους διακόπτες
- Κύτταρα SRAM
- Αντιασφάλειες (antifuses)
- Προγραμματιζόμενες διασυνδέσεις Xilinx
- Ενδεικτικά θέματα
  1) Ποια η διαφορά ανάμεσα στις διαμορφώσιμες και στις επαναδιαμορφώσιμες συσκευές;
  2) Αναφέρετε γενικά χαρακτηριστικά των προγραμματιζόμενων διακοπτών.

Δ9: Μεθοδολογίες σχεδίασης και η ροή λογικής σύνθεσης κυκλωμάτων σε FPGA

- Λογική σύνθεση
- Διαδικασίες στο frontend και στο backend ενός εργαλείου λογικής σύνθεσης
- Προσομοίωση - Τι και ποια είναι τα συνηθέστερα είδη προσομοίωσης;
- Χωροθέτηση - Τοποθέτηση - Διασύνδεση
- Ερμηνεία παραγόμενων αρχείων από εργαλεία λογικής σύνθεσης
- Ενδεικτικά θέματα
  1) Τι είναι η λογική σύνθεση; Να δοθεί διάγραμμα ροής της τυπικής ροής λογικής σύνθεσης;

Η ροή της διαδικασίας λογικής σύνθεσης

- Η συγκεκριμένη ροή εστιάζει στις τεχνολογίες FPGA. Τα περισσότερα εργαλεία CAD χωρίζονται σε frontend και backend
  - frontend: μετατροπή της περιγραφής του κυκλώματος σε netlist
  - backend: λογική σύνθεση της netlist