Σχεδίαση Υπολογιστικών Συστημάτων

Αριθμητικές πράξεις στην VHDL

Μιχάλης Ψαράκης

Τα βασικά της αριθμητικής

- Η αναπαράσταση και η επεξεργασία αριθμητικών δεδομένων είναι μια συνηθισμένη απαίτηση
  - Απρόσημοι ακέραιοι
  - Προσημασμένοι ακέραιοι
  - Πραγματικοί αριθμοί σταθερής υποδιαστολής
  - Πραγματικοί αριθμοί κινητής υποδιαστολής
- Πώς αναπαριστάνονται στην VHDL;

Τύπος integer

```
type integer is range -2147483647 to +2147483647
```

- Τα όρια του τύπου εξαρτώνται από την υλοποίηση του εργαλείου VHDL
  - Τουλάχιστον τους αριθμούς από $-2^{31} + 1$ έως $+2^{31} - 1$

Ακέραιοι τύποι

- Ο χρήστης μπορεί να ορίσει νέους ακέραιους τύπους με περιορισμό εύρους
  - Πολύ χρήσιμο. Γιατί;

Αύξουσα σειρά

```
type bcd_up is range 0 to 9 ;
type bcd_down is range 9 downto 0;
```

Φθίνουσα σειρά

```
variable num1 : bcd_up;
variable num2 : bcd_down := 5;
```

```
bcd_up := bcd_down
```

Σύνθεση ακέραιων τύπων

- Μεταφράζεται σε ένα σύνολο σημάτων ικανά να αναπαραστήσουν όλες τις τιμές του τύπου
- Πόσα σήματα απαιτούνται για να αναπαραστήσουμε τους παρακάτω τύπους integer;

```
type short is range -128 to +127;
type offset is range 30 to 31;
type negative is range -2147483647 to -1;
```


Τύποι signed και unsigned

```
signal x: signed(7 downto 0);
signal y: unsigned (0 to 3);
```

- Ο τύπος signed(7 downto 0) αναπαριστάνει όλους τους προσημασμένους αριθμούς (σε συμπλήρωμα ως προς 2) με 8 bit
- Ο τύπος unsigned(0 to 3) αναπαριστάνει όλους τους απρόσημους (θετικούς) αριθμούς με 4 bit
- Δηλώνονται σαν διανύσματα (όπως ο τύπος std_logic_vector) και όχι σαν ακέραιοι (όπως ο τύπος integer)
- Επιτρέπουν την εκτέλεση αριθμητικών λειτουργιών (σε αντίθεση με τα διανύσματα τύπου std_logic_vector)

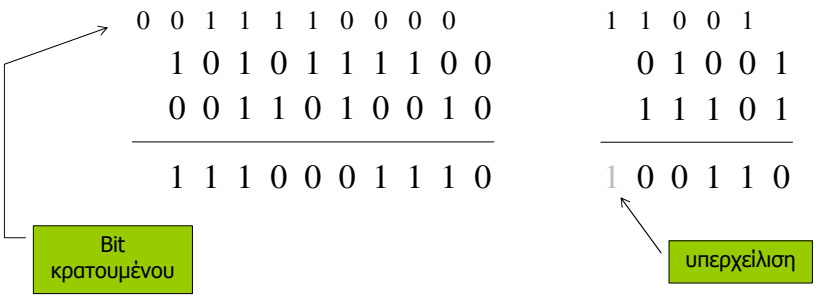
Τύπος unsigned

- Το πακέτο numeric_std παρέχει τύπο απρόσημου διανύσματος και αριθμητικές λειτουργίες

```
library ieee; use ieee.std_logic_1164.all, ieee.numeric_std.all;
entity multiplexer_6bit_4_to_1 is
  port ( a0, a1, a2, a3 : in unsigned (5 downto 0);
         sel            : in std_logic_vector (1 downto 0);
         z              : out unsigned (5 downto 0) );
end entity multiplexer_6bit_4_to_1;

architecture eqn of multiplexer_6bit_4_to_1 is
begin
  with sel select z <= a0 when "00",
                       a1 when "01",
                       a2 when "10",
                       a3 when others;
end architecture eqn;
```


Απρόσημη πρόσθεση

- Εκτελείται όπως στο δεκαδικό



Πρόσθεση στην VHDL

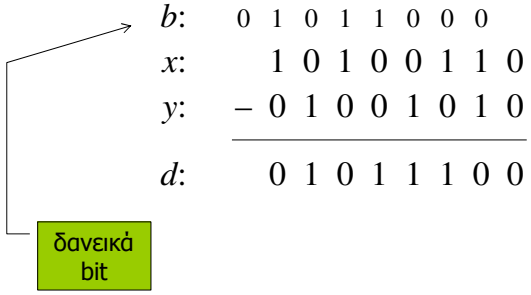
- Χρήση λειτουργιών από το numeric_std

```
library ieee; use ieee.numeric_std.all;
...
signal a, b, s: unsigned(7 downto 0);
…
s <= a + b;
```

```
signal tmp_result : unsigned(8 downto 0);
signal c : std_logic;
…
tmp_result <= ('0' & a) + ('0' & b);
c <= tmp_result(8);
s <= tmp_result(7 downto 0);
```


Απρόσημη αφαίρεση

- Όπως στο δεκαδικό



Αφαίρεση στην VHDL

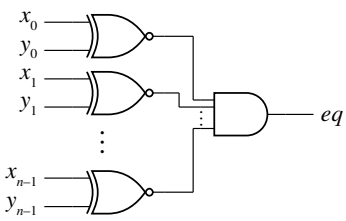
```
library ieee; use ieee.std_logic_1164.all, ieee.numeric_std.all;
entity adder_subtracter is
  port ( x, y : in unsigned(11 downto 0);
         s : out unsigned(11 downto 0);
         mode : in std_logic;
         error : out std_logic );
end entity adder_subtracter;

architecture behavior of adder_subtracter is
  signal s_tmp : unsigned(12 downto 0);
begin
  s_tmp <= ('0' & x) + ('0' & y) when mode = '0' else
           ('0' & x) - ('0' & y);
  s <= s_tmp(11 downto 0);
  error <= s_tmp(12);
end architecture behavior;
```


Πρόσθεση, αφαίρεση

- Διαφορετικές υλοποιήσεις:
  - minimum area, minimum delay
- Αρχιτεκτονικές αθροιστών:
  - Ripple carry
  - Carry-Look-Ahead
  - Brent-Kung

Σύγκριση ισότητας

- Πύλη XNOR: Ισότητα δύο bit
  - Εφαρμογή σε κάθε bit δύο απρόσημων αριθμών

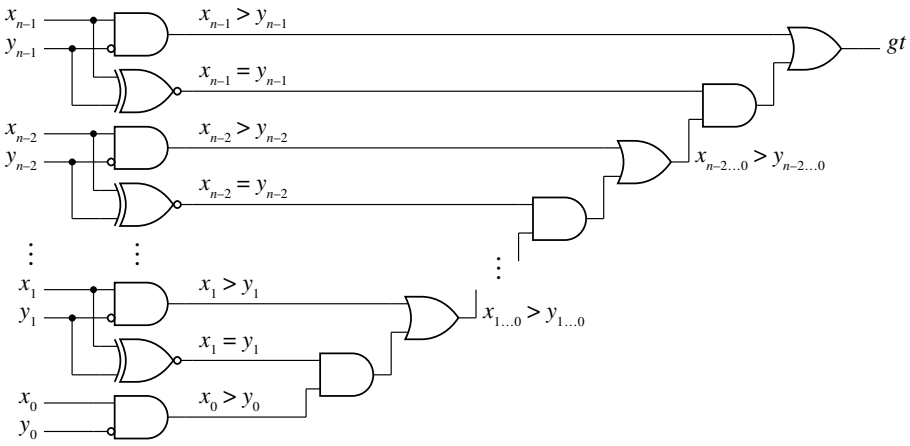


- Στην VHDL, το x = y δίνει boolean αποτέλεσμα
  - Ψευδές ή αληθές
  - Δεν μπορεί να γίνει ανάθεση σε σήμα std_logic

```
eq <= '1' when x = y else '0';
```


Σύγκριση ανισότητας

- Συγκριτής μεγέθους για x > y



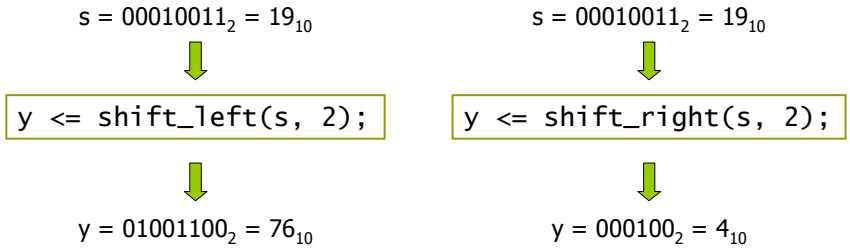
Παράδειγμα σύγκρισης στην VHDL

- Θερμοστάτης με επιθυμητή θερμοκρασία
  - Θέτει σε λειτουργία το καλοριφέρ (heater) ή το κλιματιστικό (cooler) όταν η πραγματική (actual) θερμοκρασία διαφέρει περισσότερο από 5° από την επιθυμητή (target)

```
entity thermostat is
  port ( target, actual : in unsigned(7 downto 0);
         heater_on, cooler_on : out std_logic );
end entity thermostat;

architecture rtl of thermostat is
begin
  heater_on <= '1' when actual < target - 5 else '0';
  cooler_on <= '1' when actual > target + 5 else '0';
end architecture rtl;
```


Ολίσθηση στην VHDL

- Λειτουργίες shift_left και shift_right
  - Αποτέλεσμα ίδιου μεγέθους με τον τελεστέο



Τελεστές ολίσθησης

`sll, srl, sla, sra, rol, ror`

- Προστέθηκαν στην έκδοση VHDL'93 και δεν είναι συνθέσιμοι από εργαλεία που υποστηρίζουν μόνο την έκδοση VHDL'87
- Όταν το δεξί τελούμενο είναι σταθερή τιμή
  - κανένα λογικό κύκλωμα, αναδιάταξη του bus
- Όταν το δεξί τελούμενο είναι σήμα ή μεταβλητή
  - χρήση κυκλώματος ολισθητή

Τελεστές ολίσθησης: παράδειγμα

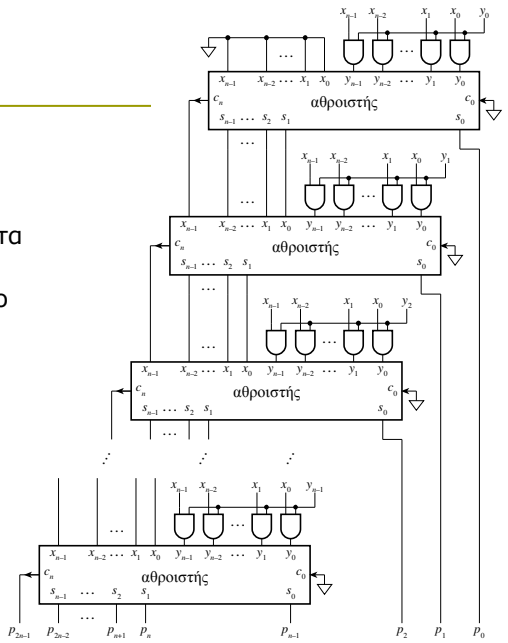
```
entity shift is
  port (a : in bit_vector (3 downto 0);
        z : out bit_vector (3 downto 0));
end entity shift;

architecture beh of shift is
begin

  z <= a sll 2;

end architecture beh;
```

Τελεστές ολίσθησης: παράδειγμα

```
entity shift is
  port (a : in bit_vector (3 downto 0);
        b : in integer range 0 to 1;
        z : out bit_vector (3 downto 0));
end entity shift;

architecture beh of shift is
begin

  z <= a sll b;

end architecture beh;
```


Απρόσημος πολλαπλασιασμός

- Συνδυαστικός πολλαπλασιαστής
  - Πύλες AND σχηματίζουν τα μερικά γινόμενα
  - Αθροιστές σχηματίζουν το πλήρες γινόμενο
- Οι αθροιστές μπορούν να έχουν οποιαδήποτε αρχιτεκτονική
- Βελτιστοποιημένοι πολλαπλασιαστές συνδυάζουν τμήματα γειτονικών αθροιστών



Μέγεθος γινομένου

- Μεγαλύτερο αποτέλεσμα για τελεστέους των n bit :

$$(2^n-1)(2^n-1)=2^{2n}-2^n-2^n+1=2^{2n}-\left(2^{n+1}-1\right)$$

  - Απαιτεί 2^{2n} bit για αποφυγή υπερχείλισης
  - Γινόμενο τελεστέων των n bit και m bit
    - Απαιτεί $n+m$ bit

```
signal x : unsigned(7 downto 0);
signal y : unsigned(13 downto 0);
signal p : unsigned(21 downto 0);
…
p <= x * y;
```


Πολλαπλασιασμός

- Διαφορετικές υλοποιήσεις:
  - signed, unsigned multiplication
  - minimum area, minimum delay
- Αρχιτεκτονικές πολλαπλασιαστών:
  - Carry-save, carry-propagate array
  - Booth-recoded
  - Wallace tree

Διαίρεση

- Πιο πολύπλοκη από τον πολλαπλασιασμό
  - Μεγαλύτερη επιφάνεια κυκλώματος, κατανάλωση ισχύος
  - Κάποιες βιβλιοθήκες παρέχουν κυκλώματα διαίρεσης (π.χ. Synopsys' DesignWare)
- Οι σύνθετες πράξεις συχνά εκτελούνται ακολουθιακά
  - Σε μια ακολουθία από βήματα, ένα σε κάθε κύκλο ρολογιού
  - Συμβιβασμός κόστους / απόδοσης / κατανάλωσης ισχύος

Προσημασμένοι ακέραιοι

- Θετικοί και αρνητικοί αριθμοί (και το 0)
- Κώδικας *προσημασμένου μεγέθους* των n bit
  - 1 bit για το πρόσημο: 0 $\Rightarrow$ +, 1 $\Rightarrow$ –
  - n – 1 bit για το μέγεθος
- Η αναπαράσταση προσημασμένου μεγέθους σπάνια χρησιμοποιείται σήμερα για ακεραίους
  - Τα κυκλώματα είναι ιδιαίτερα πολύπλοκα
- Χρήση δυαδικού κώδικα *συμπληρώματος ως προς* 2 (2s-complement)

Προσημασμένοι ακέραιοι στην VHDL

- Τύπος signed από το numeric_std

```
library ieee; use ieee.numeric_std.all;
...
s : signed(15 downto 0);
```

- Οι τύποι signed και unsigned είναι ξεχωριστοί

```
signal s1 : unsigned(11 downto 0);
signal s2 : signed(11 downto 0);
...
s1 <= s2; -- illegal
```

```
s1 <= unsigned(s2); -- s2 is known to be non-negative
...
s2 <= signed(s1); -- s1 is known to be less than 2**11
```


Προσημασμένη πρόσθεση

$$x = -x_{n-1}2^{n-1} + x_{n-2\ldots0} \qquad y = -y_{n-1}2^{n-1} + y_{n-2\ldots0}$$

$$x + y = -(x_{n-1} + y_{n-1})2^{n-1} + \underbrace{x_{n-2\ldots0} + y_{n-2\ldots0}}_{\text{δίνει το } c_{n-1}}$$

- Εκτέλεση της πρόσθεσης, όπως για απρόσημους αριθμούς
  - Υπερχείλιση εάν το c_{n-1} διαφέρει από το c_n
- Μπορεί να χρησιμοποιηθεί το ίδιο κύκλωμα για προσημασμένη και απρόσημη πρόσθεση

Παραδείγματα προσημ. πρόσθεσης

0 0 0 0 0 0 0 0	1 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0
72: 0 1 0 0 1 0 0 0	–63: 1 1 0 0 0 0 0 1	–42: 1 1 0 1 0 1 1 0
49: 0 0 1 1 0 0 0 1	–32: 1 1 1 0 0 0 0 0	8: 0 0 0 0 1 0 0 0
121: 0 1 1 1 1 0 0 1	–95: 1 0 1 0 0 0 0 1	–34: 1 1 0 1 1 1 1 0
χωρίς υπερχείλιση	χωρίς υπερχείλιση	χωρίς υπερχείλιση
0 1 0 0 1 0 0 0	1 0 0 0 0 0 0 0	1 1 1 1 0 0 0 0
72: 0 1 0 0 1 0 0 0	–63: 1 1 0 0 0 0 0 1	42: 0 0 1 0 1 0 1 0
105: 0 1 1 0 1 0 0 1	–96: 1 0 1 0 0 0 0 0	–8: 1 1 1 1 1 0 0 0
1 0 1 1 0 0 0 1	0 1 1 0 0 0 0 1	34: 0 0 1 0 0 0 1 0
θετική υπερχείλιση	αρνητική υπερχείλιση	χωρίς υπερχείλιση

Προσημασμένη πρόσθεση στην VHDL

- Το αποτέλεσμα του + έχει ίδιο μέγεθος με τους τελεστέους

```
signal v1, v2 : signed(11 downto 0);
signal sum : signed(12 downto 0);
...
sum <= resize(v1, sum'length) + resize(v2 sum'length);
```

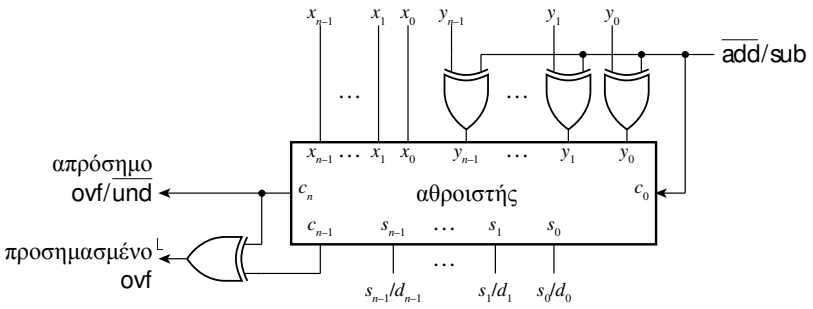
- Για έλεγχο υπερχείλισης, σύγκριση προσήμων

```
signal x, y, z: signed(7 downto 0);
signal ovf : std_logic;
...
z <= x + y;
ovf <= (not x(7) and not y(7) and z(7))
       or (x(7) and y(7) and not z(7));
```

Προσημασμένη αφαίρεση

$$x - y = x + (-y) = x + \overline{y} + 1$$

- Χρήση αθροιστή συμπληρώματος ως προς 2
  - Υπολογίζει το συμπλήρωμα του y και θέτει το c_0 = 1



Άλλες προσημασμένες πράξεις

- Σύγκριση
  - =, παρόμοια με την απρόσημη
  - >, σύγκριση bit προσήμου, χρήση του $\overline{x_{n-1}} \cdot y_{n-1}$
- Πολλαπλασιασμός
  - Πιο πολύπλοκος λόγω της ανάγκης για επέκταση προσήμου στα μερικά γινόμενα

Μετατροπή δεδομένων

- Στο πακέτο std_logic_arith της βιβλιοθήκης ieee library υπάρχουν οι συναρτήσεις μετατροπής δεδομένων:
- conv_integer(p) :
  - Μετατρέπει μια παράμετρο p τύπου INTEGER, UNSIGNED, SIGNED, ή STD_ULOGIC σε τιμή τύπου INTEGER
- conv_unsigned(p, b):
  - Μετατρέπει μια παράμετρο p τύπου INTEGER, UNSIGNED, SIGNED, ή STD_ULOGIC σε τιμή τύπου UNSIGNED με b bit
- conv_signed(p, b):
  - Μετατρέπει μια παράμετρο p τύπου INTEGER, UNSIGNED, SIGNED, ή STD_ULOGIC σε τιμή τύπου SIGNED με b bit
- conv_std_logic_vector(p, b):
  - Μετατρέπει μια παράμετρο p τύπου INTEGER, UNSIGNED, SIGNED, ή STD_LOGIC σε τιμή τύπου STD_LOGIC_VECTOR με b bit

Μετατροπή δεδομένων: παράδειγμα

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
...
signal a: in unsigned (7 downto 0);
signal b: in unsigned (7 downto 0);
signal y: out std_logic_vector(7 downto 0);
...
y <= CONV_STD_LOGIC_VECTOR ((a+b), 8);
```

Αριθμοί σταθερής υποδιαστολής

- Αναπαριστάνονται ως ακέραιοι με έμμεση κλιμάκωση κατά μια δύναμη του 2
- Στο δεκαδικό

$$10{,}24_{10} = 1\times10^{1} + 0\times10^{0} + 2\times10^{-1} + 4\times10^{-2}$$

- Στο δυαδικό

$$101{,}01_{2} = 1\times2^{2} + 0\times2^{1} + 1\times2^{0} + 0\times2^{-1} + 1\times2^{-2} = 5{,}25_{10}$$

- Αναπαράσταση ως διάνυσμα bit : 10101
  - Η δυαδική υποδιαστολή είναι έμμεση

Απρόσημοι σταθερής υποδιαστολής

- Απρόσημος σταθερής υποδιαστολής των n bit
  - m bit πριν και f bit μετά την δυαδική υποδιαστολή

$$x = x_{m-1}2^{m-1} + \cdots + x_0 2^0 + x_{-1}2^{-1} + \cdots + x_{-f}2^{-f}$$

  - Εύρος: 0 έως $2^m - 2^{-f}$
  - Ακρίβεια: 2^{-f}
  - Το m μπορεί να είναι ≤ 0, δίνοντας μόνο κλασματικό μέρος
    - π.χ., m= –2: 0,0001001101

Προσημασμένοι σταθερής υποδ/στολής

- Προσημασμένος σταθερής υποδιαστολής των n bit σε συμπλήρωμα ως προς 2
  - m bit πριν και f bit μετά την δυαδική υποδιαστολή

$$x = -x_{m-1}2^{m-1} + \cdots + x_0 2^0 + x_{-1}2^{-1} + \cdots + x_{-f}2^{-f}$$

- Εύρος: -2^{m-1} έως $2^{m-1} - 2^{-f}$
- Ακρίβεια: 2^{-f}
- Π.χ., 111101, προσημασμένος σταθερής υποδιαστολής, $m = 2$
  - $11{,}1101_2 = -2 + 1 + 0{,}5 + 0{,}5 + 0{,}0625 = -0{,}1875_{10}$

Σταθερή υποδιαστολή στην VHDL

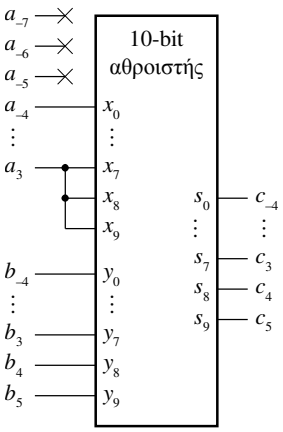
- Χρήση του προτεινόμενου πακέτου fixed_pkg
  - Βρίσκεται σε στάδιο προτυποποίησης από το IEEE
  - Τύποι ufixed και sfixed
  - Αριθμητικές λειτουργίες, αλλαγή μεγέθους, μετατροπή

```
library ieee_proposed; use ieee_proposed.fixed_pkg.all;
entity fixed_converter is
  port ( input : in ufixed(5 downto -7);
         output : out sfixed(7 downto -7) );
end entity fixed_converter;
```

Λειτουργίες σταθερής υποδιαστολής

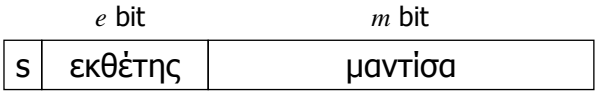
- Απλά, χρήση υλικού για ακεραίους
  - π.χ., πρόσθεση:

$$x + y = (x \times 2^f + y \times 2^f) / 2^f$$



- Θα πρέπει εξασφαλιστεί ότι οι δυαδικές υποδιαστολές είναι ευθυγραμμισμένες

Μορφή κινητής υποδιαστολής IEEE



$$x = M \times 2^E = (-1 \times s) \times 1, \mu\alpha\nu\tau\acute{\iota}\sigma\alpha \times 2^{\varepsilon\kappa\theta\acute{\varepsilon}\tau\eta\varsigma - 2^{e-1}+1}$$

- s: bit προσήμου (0 ⇒ μη αρνητικό, 1 ⇒ αρνητικό)
- Κανονικοποίηση: 1,0 ≤ |M| < 2,0
  - Το M πάντα έχει ένα αρχικό bit 1 πριν την δυαδική υποδιαστολή, έτσι δεν χρειάζεται να αναπαρίσταται ρητά (κρυμμένο bit)
- Εκθέτης: αναπαράσταση υπέρβασης: E + 2^{e-1}–1

Εύρος κινητής υποδιαστολής

- Οι εκθέτες 000...0 και 111...1 είναι δεσμευμένοι
- Μικρότερη τιμή
  - εκθέτης: 000...01 $\Rightarrow E = -2^{e-1} + 2$
  - μαντίσα: 0000...00 $\Rightarrow M = 1{,}0$
- Μεγαλύτερη τιμή
  - εκθέτης: 111...10 $\Rightarrow E = 2^{e-1} - 1$
  - μαντίσα: 111...11 $\Rightarrow M \approx 2{,}0$
- Εύρος: $2^{-2^{e-1}+2} \le |x| < 2^{2^{e-1}}$

Ακρίβεια κινητής υποδιαστολής

- Η σχετική ακρίβεια είναι κατά προσέγγιση 2^{-m}
  - Όλα τα bit της μαντίσα είναι σημαντικά
- m bit ακρίβειας
  - $m \times \log_{10}2 \approx m \times 0{,}3$ δεκαδικά ψηφία

Παραδείγματα μορφών

- Απλής ακρίβειας IEEE, 32 bit
  - e = 8, m = 23
  - εύρος ≈ ±1,2 × 10^{-38} έως ±3,4 × 10^{38}
  - ακρίβεια ≈ 7 δεκαδικά ψηφία
- Προσαρμοσμένη μορφή, 22 bit
  - e = 5, m = 16
  - εύρος ≈ ±6,1 × 10^{-5} έως ±6,6 × 10^{4}
  - ακρίβεια ≈ 5 δεκαδικά ψηφία

Μη κανονικοποιημένοι αριθμοί

- Εκθέτης = 000...0 ⇒ το κρυμμένο bit είναι 0

$$x = M \times 2^{E} = (-1 \times s) \times 0,\mu\alpha\nu\tau\acute{\iota}\sigma\alpha \times 2^{-2^{e-1}+1}$$

  - Μικρότεροι από τους κανονικοποιημένους
    - Επιτρέπουν μια βαθμιαία ανεπάρκεια (gradual underflow), με φθίνουσα ακρίβεια
  - Μαντίσα = 000...0

$$x = M \times 2^{E} = (-1 \times s) \times 0,0 \times 2^{-2^{e-1}+1} = \pm 0,0$$

Άπειρα και NaN

- Εκθέτης = 111...1, μαντίσα = 000...0
  - ± Άπειρο
  - Μπορεί να χρησιμοποιηθεί σε υπολογισμούς που ακολουθούν, αποφεύγοντας την ανάγκη για έλεγχο υπερχείλισης
- Εκθέτης = 111...1, μαντίσα ≠ 000...0
  - Not-a-Number (NaN)
  - Ένδειξη μη επιτρεπτού αποτελέσματος ή αποτελέσματος που δεν ορίζεται
    - π.χ., 0,0 / 0,0
  - Μπορεί να χρησιμοποιηθεί σε υπολογισμούς που ακολουθούν

Λειτουργίες κινητής υποδιαστολής

- Αρκετά πιο πολύπλοκες από τις ακέραιες λειτουργίες
  - Π.χ., πρόσθεση
    - αποσύμπτυξη των πεδίων, ευθυγράμμιση δυαδικών υποδιαστολών, προσαρμογή εκθετών
    - πρόσθεση μαντίσα, έλεγχος για εξαιρέσεις
    - στρογγυλοποίηση και κανονικοποίηση του αποτελέσματος, προσαρμογή εκθέτη
- Συνδυαστικά κυκλώματα δεν είναι εφικτά
  - Ακολουθιακά κυκλώματα με διοχέτευση

Κινητή υποδιαστολή στην VHDL

- Χρήση του προτεινόμενου πακέτου float_pkg
  - Βρίσκεται σε στάδιο προτυποποίησης από το IEEE
  - Τύποι float, float32, float64, float128
  - Αριθμητικές λειτουργίες, αλλαγή μεγέθους, μετατροπή
- Δεν θα περνά από σύνθεση
  - Αντίθετα, θα χρησιμοποιείται για την επαλήθευση κυκλωμάτων που έχουν βελτιστοποιηθεί με το χέρι