HY430 – Εργαστήριο Ψηφιακών Κυκλωμάτων

Διδάσκων: Χ. Σωτηρίου, Βοηθός: (θα ανακοινωθεί)

http://inf-server.inf.uth.gr/courses/CE430/

1η διάλεξη – Παρουσίαση του μαθήματος

Περιεχόμενα

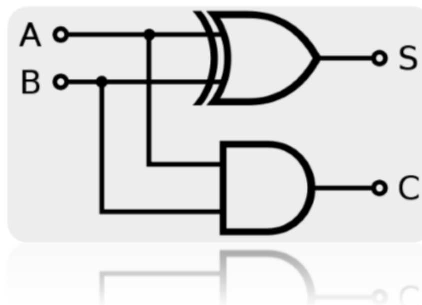
- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Αθροιστής χωρίς Κρατούμενο – Half Adder

a	b	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



s = a'b + ab' = a (+) b
co = ab

Το Λογικό Βάθος (Λ.Β.) για το Άθροισμα s και κρατούμενο co είναι 2 και 1 αντίστοιχα.

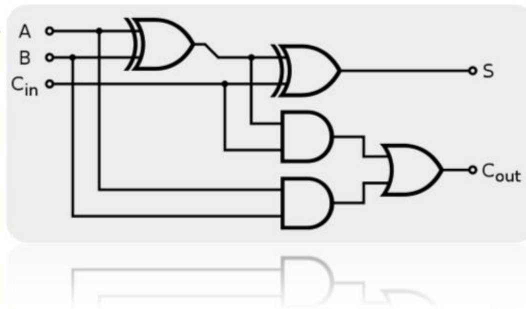
ΠΡΟΣΟΧΗ: οι πύλες XOR ΔΕΝ χρησιμοποιούνται για τον υπολογισμό του Λογικού Βάθους, αλλά μόνο οι 2 βασικές πράξεις and, or.

Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Αθροιστής με Κρατούμενο – Full Adder

a	b	ci	co	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1




```
s = a'b'ci + a'bci' + ab'ci' + abci =
    ci (a'b' + ab) + ci'(a'b + ab') =
    ci (a (+)' b) + ci' (a (+) b) = a (+) b (+) c
co = a'bci + ab'ci + abci' + abci =
     ab (ci + ci') + ci (a'b + ab') = ab + ci(a (+) b)
```


Το Λογικό Βάθος (Λ.Β.) για το Άθροισμα s και κρατούμενο co είναι 4 και 4 αντίστοιχα.

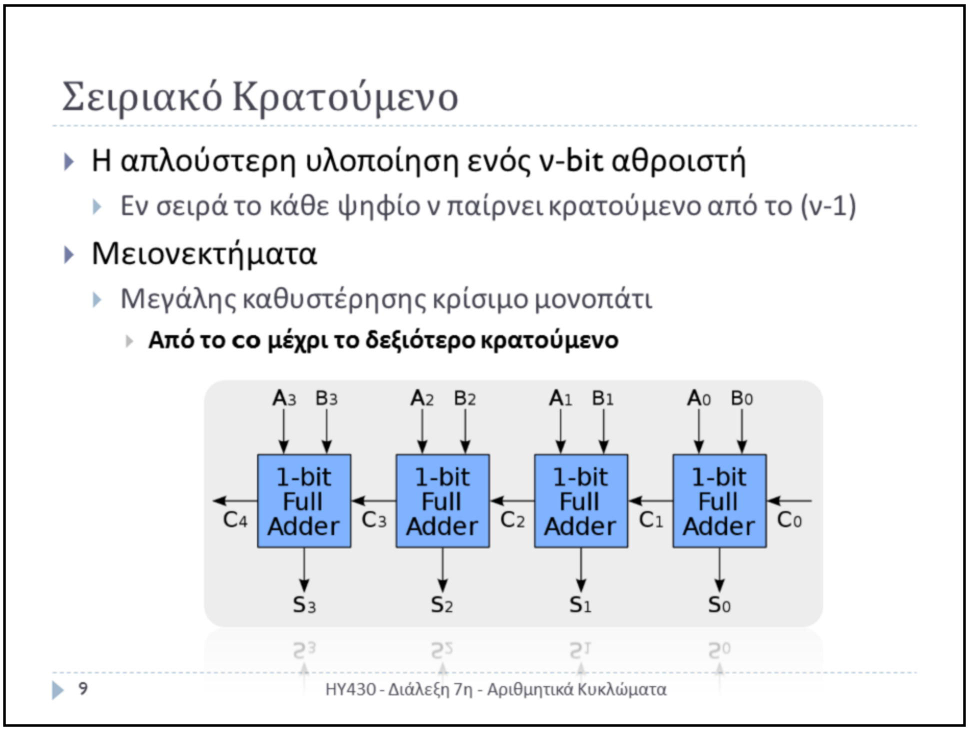
Έμμεση Υλοποίηση και Σήματα

a	b	ci	co	s	κρατούμενο
0	0	0	0	0	αναίρεση
0	0	1	0	1	αναίρεση
0	1	0	0	1	προώθηση
0	1	1	1	0	προώθηση
1	0	0	0	1	προώθηση
1	0	1	1	0	προώθηση
1	1	0	1	0	ανάθεση
1	1	1	1	1	ανάθεση

- Σε κάποιες υλοποιήσεις αθροιστών οι έξοδοι (s, co) προκύπτουν από έμμεσες εκφράσεις:
  - `G = A.B`
  - `D = A'.B'`
  - `P = A + B` ή
  - `P = A (+) B`
- Έτσι, οι εκφράσεις για co, s μετατρέπονται ως εξής:
  - `co = G + P ci` και
  - `s = p (+) ci`

Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση



co = ab (ci + ci’) + ci (a’b + ab’)

Άρα η καθυστέρηση σε Λογικά Επίπεδα είναι 4 (τεχνολογικά ανεξάρτητη καθυστέρηση).

Αναλυτικά $1^{ο}$ επίπεδο (χειρότερο) είναι το a’b, ab’, $2^{ο}$ επίπεδο το a’b + ab’, $3^{ο}$ επίπεδο το ci (a’b + ab’), και $4^{ο}$ ολόκληρο το co.

Άρα, η καθυστέρηση του co είναι 4 Λογικά Επίπεδα, ενώ το κόστος σε εμβαδό είναι 9.

Το κόστος σε εμβαδό ισούται με το πλήθος των εμφανίσεων των δυαδικών όρων-μεταβλητών.

Έτσι:

Καθυστέρηση c1: 4 t.u. (1 t.u. – time unit = 1 επίπεδο λογικής AND/OR = 1 g.d. – gate delay)

Καθυστέρηση c2: (4 + 4) t.u.

Καθυστέρηση c3: (4 + 4 + 4) t.u.

Καθυστέρηση c4: (4 + 4 + 4 + 4) t.u.

Καθυστέρηση ci : (4 * I) t.u.

Η καθυστέρηση του 4-bit αθροιστή είναι 4*3 ($4^{ο}$ bit) + 4 (s3) = 16 λογικά επίπεδα/t.u.

Καθυστέρηση c16 (16-bit αθροιστή) = 4 * 16 = 64 t.u., δηλ. 64 λογικά επίπεδα
(ή καθυστερήσεις τεχνολογικά-ανεξάρτητων πυλών)

Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Πρόγνωση Κρατουμένου

- Ο βασικός στόχος είναι να μειωθεί η μεγάλη καθυστέρηση της αλυσίδας των κρατουμένων των ν-bit
- Βασική ιδέα
  - Ακριβής πρόγνωση κρατουμένου πριν αυτό προκύψει από τις εξισώσεις
- Για ομάδες ν-bit (όπου συνήθως ν ~ 4)
  - Υπολογίζεται η πρόγνωση του κρατουμένου
  - αυτή προωθείται στην επόμενη ομάδα
  - Η καθυστέρηση της πρόγνωσης είναι σημαντικά μικρότερη από τον σειριακή προώθηση του κρατουμένου

Πρόγνωση Κρατουμένου - Γένεση, Προώθηση

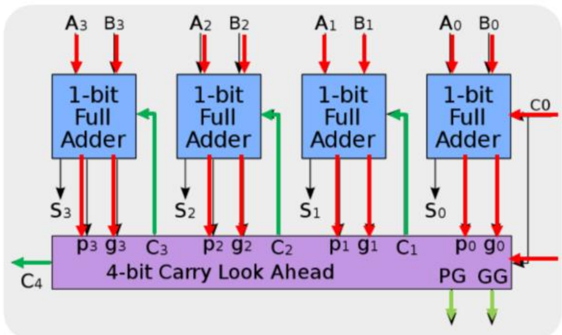
- Στην πρόσθεση A + B παράγεται κρατούμενο μόνο όταν AB:
  - $G = AB$
- Ένα κρατούμενο προάγεται στο επόμενο ψηφίο όταν:
  - $P = A + B$
- Το κρατούμενο μπορεί να εκφραστεί ως:
  - $co = G + P\ ci$

- Για v=4-bits:
  - $c1 = G0 + P0\ c0$
  - $c2 = G1 + P1\ c1 = G1 + P1(G0 + P0\ c0) = G1 + G0P1 + c0P0P1$
  - $c3 = G2 + G1P2 + G0P1P2 + C0P0P1P2$
  - $C4 = G3 + G2P3 + G1P2P3 + C0P0P1P2P3$

Ομοιόμορφη εξίσωση

Πρόγνωση Κρατουμένου – 4-bit Αθροιστής

- C4 = (G3 + G2P3 + G1P2P3 + G0P1P2P3) + C0(P0P1P2P3)
- **PG = P0P1P2P3**
- **GG = G3 + G2P3 + G1P2P3 + G0P1P2P3**
- C4 = GG + C0 PG



Καθυστέρηση για Pi = AiBi, Gi = Ai + Bi = 1 Λογικό Επίπεδο

Καθυστέρηση pi, gi : 1 t.u.

Καθυστέρηση για co, c1, c2, c3, c4:

c1 = G0 + P0 c0, δηλαδή 3 λογικά επίπεδα.

Αναλυτικά τα G0,P0 έχουν καθυστέρηση 1 λογικό επίπεδο, το 2ο είναι το P0c0,

και τρίτο ολόκληρο το c1

Ομοίως για το c4:

c4 = G3 + G2P3 + G1P2P3 + C0P0P1P2P3

Τα P0, P1, P2, P3, G2, G1, G2 έχουν καθυστέρηση 1 λογικό επίπεδο, το 2ο είναι το C0P0P1P2P3,

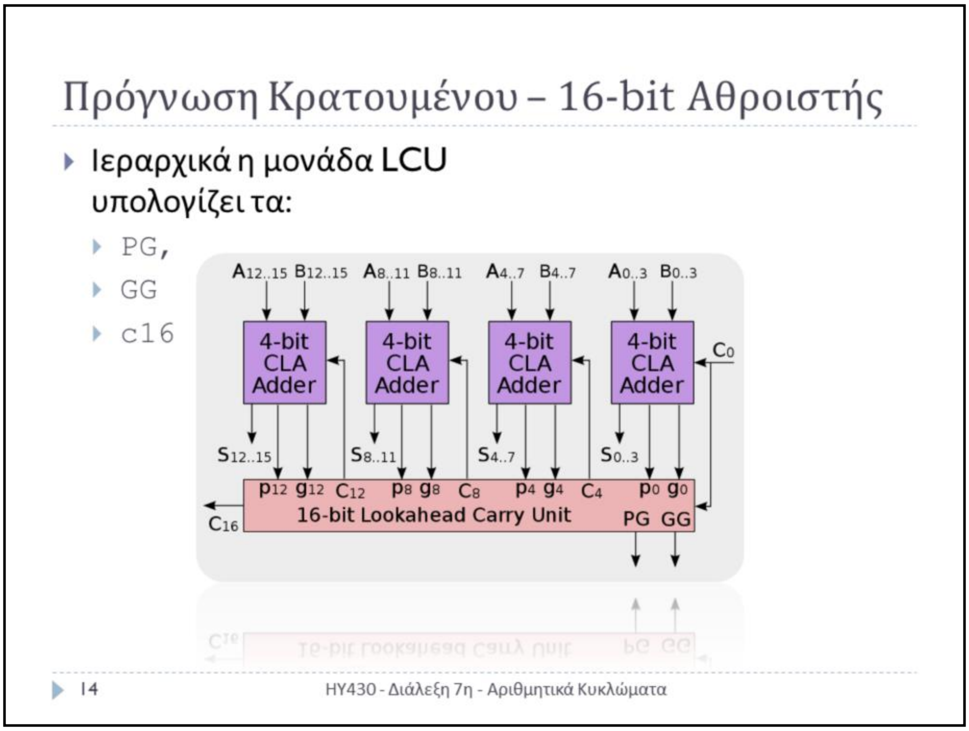
και τρίτο ολόκληρο το c4.

Έτσι:

Καθυστέρηση για c1-c4 : 3 t.u. (για υλοποίηση 2-επίπεδης λογικής)

Καθυστέρηση για PG, GG = 2 t.u.

Άρα η καθυστέρηση για τον 4-bit αθροιστή είναι 3 + 4 = 7 λογικά επίπεδα (αντί για 16 του σειριακού κρατουμένου)



Για τον 16-bit αθροιστή με 4 των 4-bit με Πρόγνωση και 1 μονάδα Ιεραρχικής Πρόγνωσης (LCU), οι καθυστερήσεις έχουν ως εξής:

PPi (4-bit) = P0P1P2P3 = 2 λογικά επίπεδα
GGi (4-bit) = G3 + G2P3 + G1P2P3 + G0P1P2P3 = 3 λογικά επίπεδα

Ιεραρχικό ci (c4, c8, c12, c16) από τα PPi, GGi:
c4 = GG1 + c0 PP1 **-> 4 επίπεδα**
c8 = GG2 + c4 PP2 = GG2 + GG1 PP2 + c0 PP1 PP2 **-> 5 επίπεδα λόγω του GG1 . PP2**
c12 = GG3 + c8 PP3 = GG3 + (GG2 + GG1 PP2 + c0 PP1 PP2) PP3 = GG3 + GG2 PP3 + GG1 PP2 PP3 + c0 PP1 PP2 PP3 **-> 5 επίπεδα ομοίως**
[όπου GGi, PPi είναι για τον i ($1^{ο}$, $2^{ο}$ και $3^{ο}$) 4-bit αθροιστή]

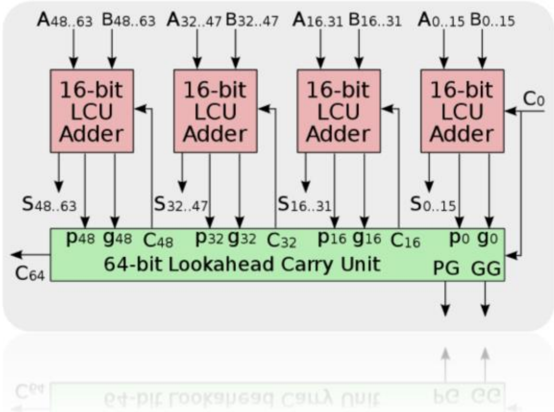
Άρα, τα c8, c12 έχουν βάθος 5 λογικά επίπεδα, το c4 έχει 4.

s0…3 : 4 t.u.
s4…7 : 8 t.u. (*4 για το κρατούμενο και 4 για το άθροισμα*)
s8…11 : 9 t.u (*5 για το κρατούμενο και 4 για το άθροισμα*)
s2, s3, s4 : 9 t.u., (*ομοίως*)
c16 επίσης σε 5 t.u.

Η μέγιστη καθυστέρηση είναι βάθους 9 (9 t.u.).
Στον 16-bit σειριακό αθροιστή θα ήταν 4*15 (κρατούμενου του $15^{ου}$ bit) + 4 = 64 λογικά επίπεδα/t.u.

Πρόγνωση Κρατουμένου – 64-bit Αθροιστής

- **<u>Ίδια ιδέα με 2ο επίπεδο ιεραρχίας</u>**



15 HY430 - Διάλεξη 7η - Αριθμητικά Κυκλώματα

Ισχύει η ίδια ιεραρχική ανάλυση με τον 16-bit Αθροιστή για τις ομάδες των 16 πλέον bits…

Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

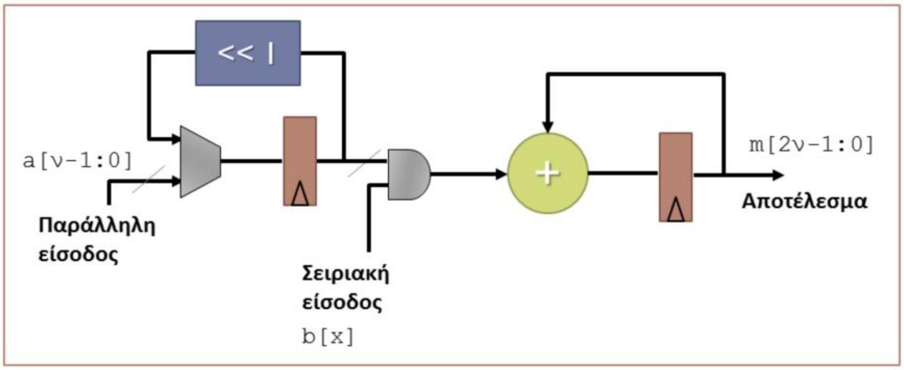
Ολίσθηση και Πρόσθεση

- **1011001 x 1101 :**

```
        1011001    Πολλαπλασιαστέος
           1101 x  Πολλαπλασιαστής
----------------
1 :     1011001
0 :    0000000
1 :   1011001
1 :  1011001     +
----------------
    10010000101
```

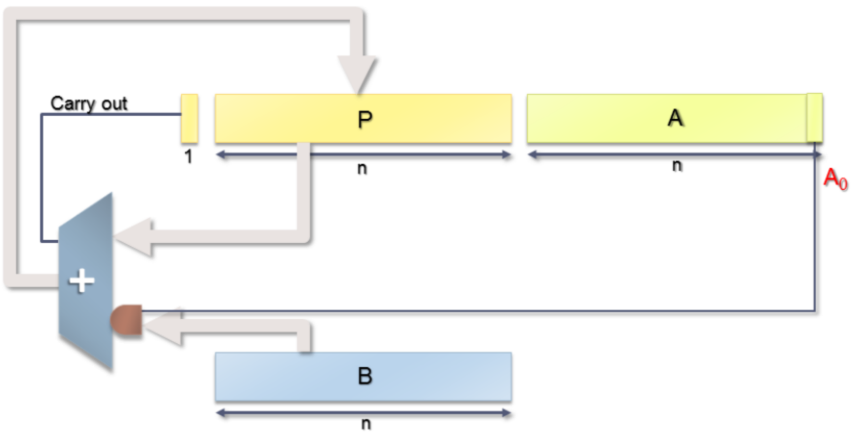
- Προσθέτουμε με τον πολλαπλασιαστή από δεξιά προς αριστερά

Ολίσθηση και Πρόσθεση



Δυαδικός Πολλαπλασιαστής

- Εκτελεί Μη-Προσημασμένο Πολλαπλασιασμό



Δυαδικός Πολλαπλασιαστής

- Χρησιμοποιεί:
  - 1 n-bit Αθροιστή
  - 3 Καταχωρητές: P, A, B
  - Έλεγχο αν το A_0 είναι 0/1
- Αρχικοποίηση
  - P = 0, A = πρώτος αριθμός, B = δεύτερος αριθμός
- Αλγόριθμος:
  1. Αν το A_0 είναι 1, $P = P + B$ ($B = b_{n-1}b_{n-2}\ldots b_0$)
  2. Ολίσθηση του Ζεύγους (P, A) κατά 1 δεξιά
     το τελευταίο bit του A δεν χρησιμοποιείται

Παράδειγμα Πολλαπλασιασμού

P	A	Step Explanation
0000	0101	Multiply 5=0101 with 3=0011. B always 0011
+0011		step 1(i): A0=1, add B to P
0011	0101	step 1(ii): shift (P,A) right one bit
0001	1010	step 2(i): A0=0, add 0 to P (or do nothing)
0001	1010	step 2(ii): shift (P,A) right one bit
0000	1101	step 3(i): A0=1, add B to P
+0011		
0011	1101	step 3(ii): shift (P,A) right one bit
0001	1110	step 4(i): A0=0, add 0 to P
0001	1110	step 4(ii): shift (P,A) right one bit
0000	1111	result is 1111_2=15 in (P,A)

23
X 102
46
00
+ 23
2346

0011
X 0101
0011
0000
+ 0011
001111

Αλγόριθμος Πολλαπλασιασμού

```
MULTIPLY(x, y, m)
// Είσοδοι - x : πολλαπλασιαστέος, y : πολλαπλασιαστής, Έξοδος – m : γινόμενο
{
   n = LENGTH(y);
   m = 0;
   t = x;  // ολισθητής //
   for i in 1 to n  // για κάθε ψηφίο του y //
   {
      if (y[i] == 1)
          m = m + t;  // πρόσθεση μερικού παράγοντα //
      t = t << 1;  // ολίσθηση 1 ψηφίο δεξιά για κάθε ψηφίο του y //
   }
   return r;
}
```

Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Πολλαπλασιασμός με Πίνακα (LUT)

- Γρήγορη μνήμη για λίγα ψηφία
  - Χρησιμοποιείται για FPGAs
  - Μικρή καθυστέρηση
  - Εκθετικό μέγεθος

3x3 LUT	000	001	010	011	100	101	110	111
000	000000	000000	000000	000000	000000	000000	000000	000000
001	000000	000001	000010	000011	000100	000101	000110	000111
010	000000	000010	000100	000110	001000	001010	001100	001110
011	000000	000011	000110	001001	001100	001111	010010	010101
100	000000	000100	001000	001100	010000	010100	011000	011100
101	000000	000101	001010	001111	010100	011001	011110	100011
110	000000	000110	001100	010010	011000	011110	100100	101010
111	000000	000111	001110	010101	011100	100011	101010	110001

Περιεχόμενα

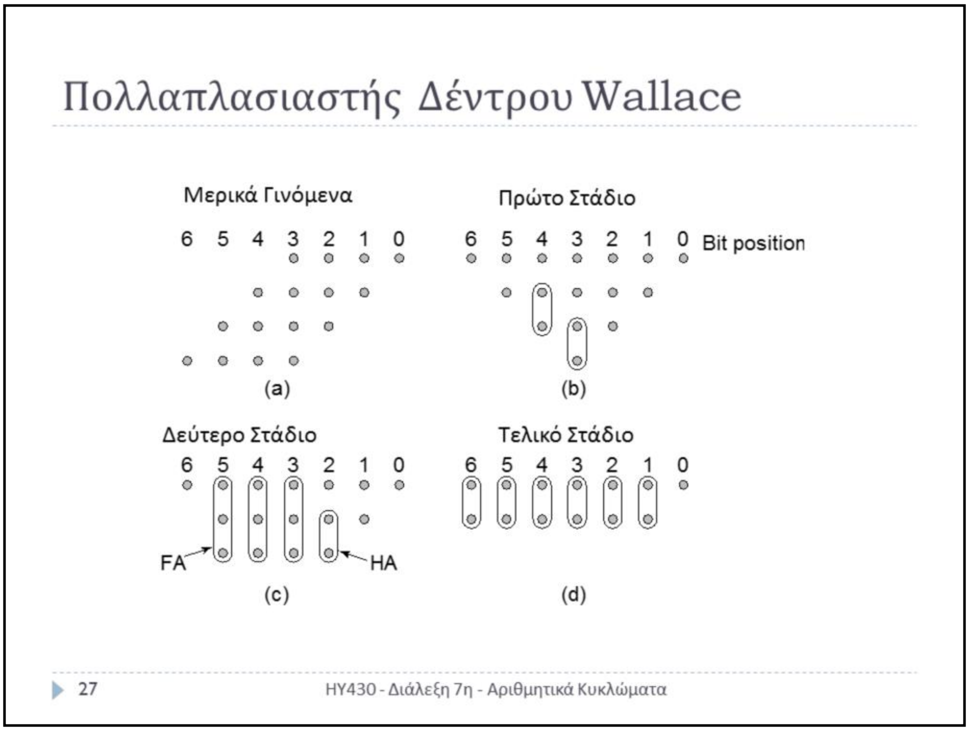
- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Πολλαπλασιασμός Μερικών Παραγόντων

- **67 x 54**

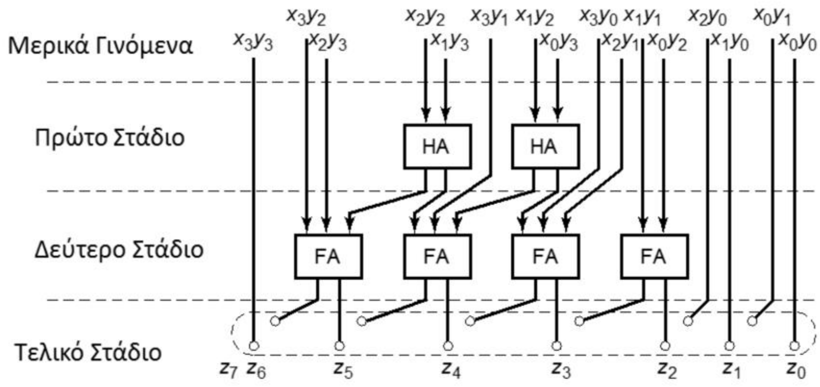
1ο με 1ο	2ο με 1ο	1ο με 2ο	2ο με 2ο
6**7**	**6**7	6**7**	**6**7
X 5**4**	X 5**4**	X **5**4	X **5**4
28	28	28	28
240	**240**	240	240
350	350	**350**	350
3000 +	3000 +	3000 +	**3000** +
3618	3618	3618	3618

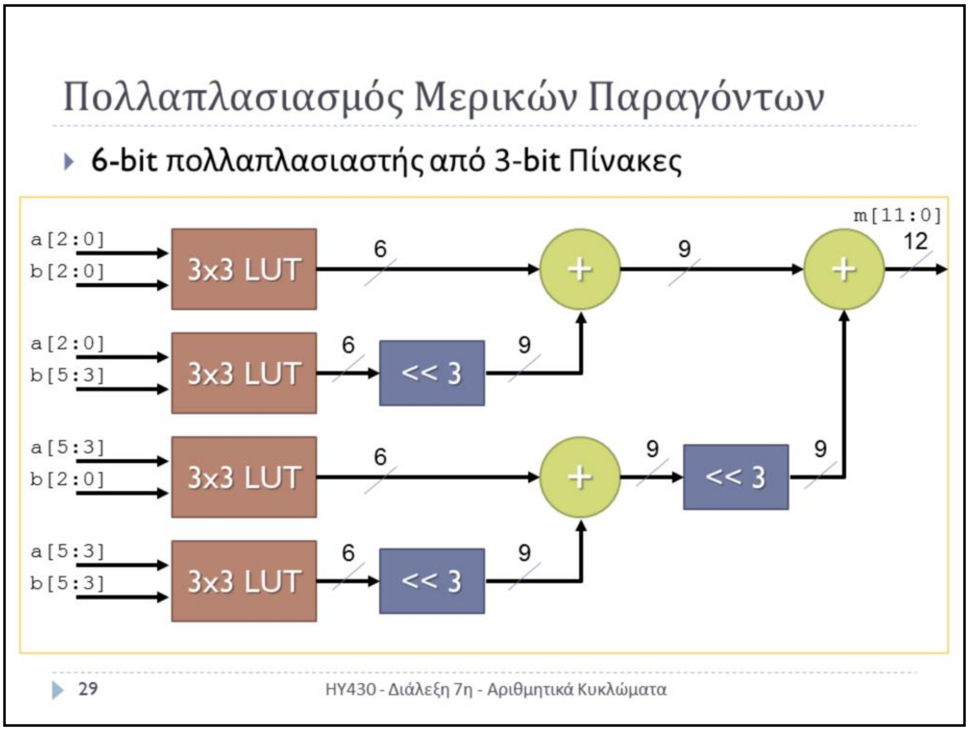
- **οι τέσσερις αυτοί συνδυασμοί μπορούν να γίνουν σε σύνολα από δυαδικά ψηφία**



Μπορούμε να μετασχηματίσουμε την δομή των μερικών γινομένων του πολλαπλασιασμού σε ένα δέντρο πολλαπλών επιπέδων. Παραπάνω φαίνεται ο μετασχηματισμός των γινομένων για αριθμούς 4-bit.

Πολλαπλασιαστής Δέντρου Wallace





Πολλαπλασιάζουμε τα:

a[5:3], a[2:0]

x b[5:3], b[2:0]

b[2:0] x a[2:0] → **1^{o} με 1^{o}**

b[2:0] x a[5:3] (<< 3) → **1^{o} με 2^{o}**

b[5:3] x a[2:0] (<< 3) → **2^{o} με 1^{o}**

b[5:3] x a[5:3] (<< 3) x 2 → **2^{o} με 2^{o}**

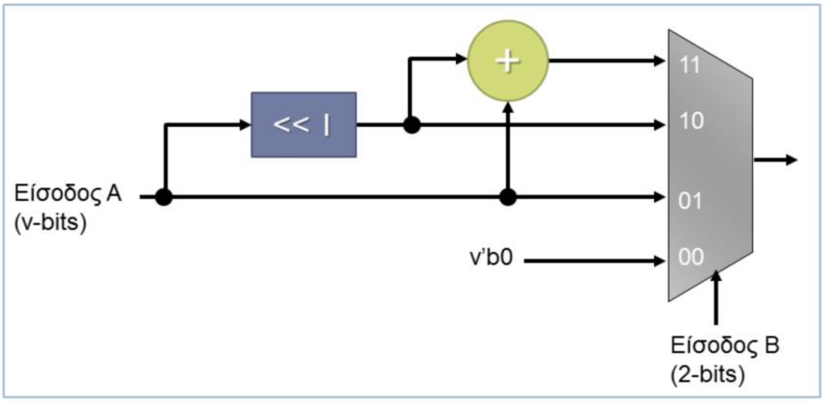
Ο ολισθητής δεξιά συνδυάζει σε 1 πράξη την $2^{η}$ ολίσθηση του $4^{ου}$ παράγοντα και του $2^{ου}$.

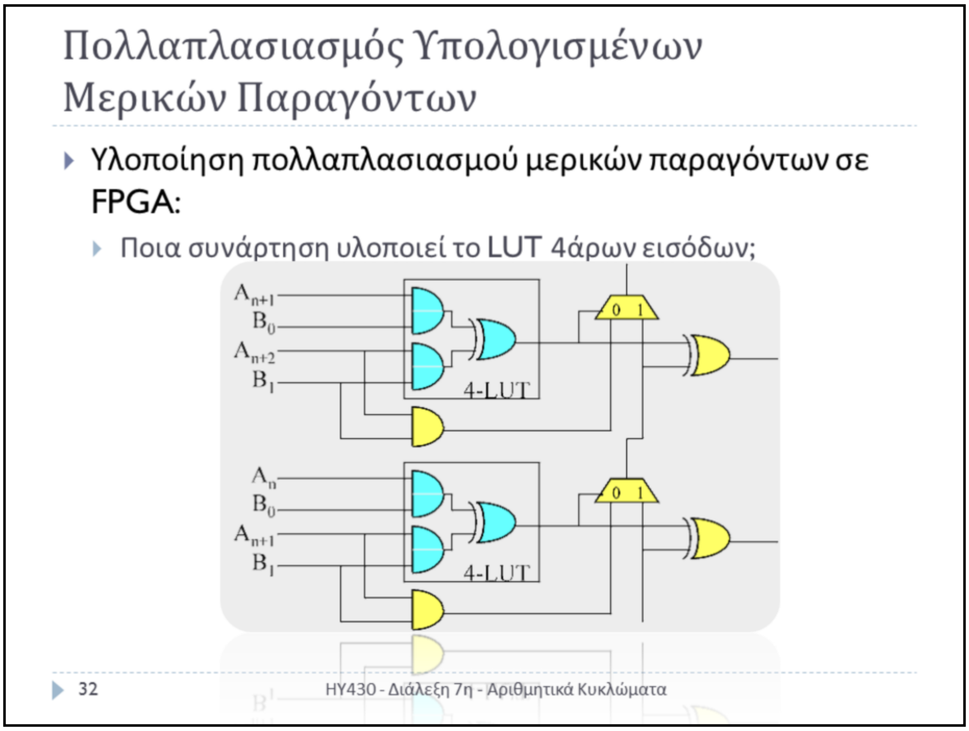
Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Πολλαπλασιασμός Υπολογισμένων Μερικών Παραγόντων

- Ειδική περίπτωση του πίνακα
  - όπου η συνδυαστική λογική είναι απλή
- Βασική ιδέα - πολλαπλασιασμός ν-bit με 2-bit:





Η συνάρτηση που υλοποιεί λ.χ. το κάτω LUT είναι η:

f = An . B0 (+) An+1 . B1, δηλαδή:

{B1, B0} = 00 => f = 0,

{B1, B0} = 01 => f = An (+) 0 = An

{B1, B0} = 10 => f = 0 (+) An+1 = An+1, άρα (x2)

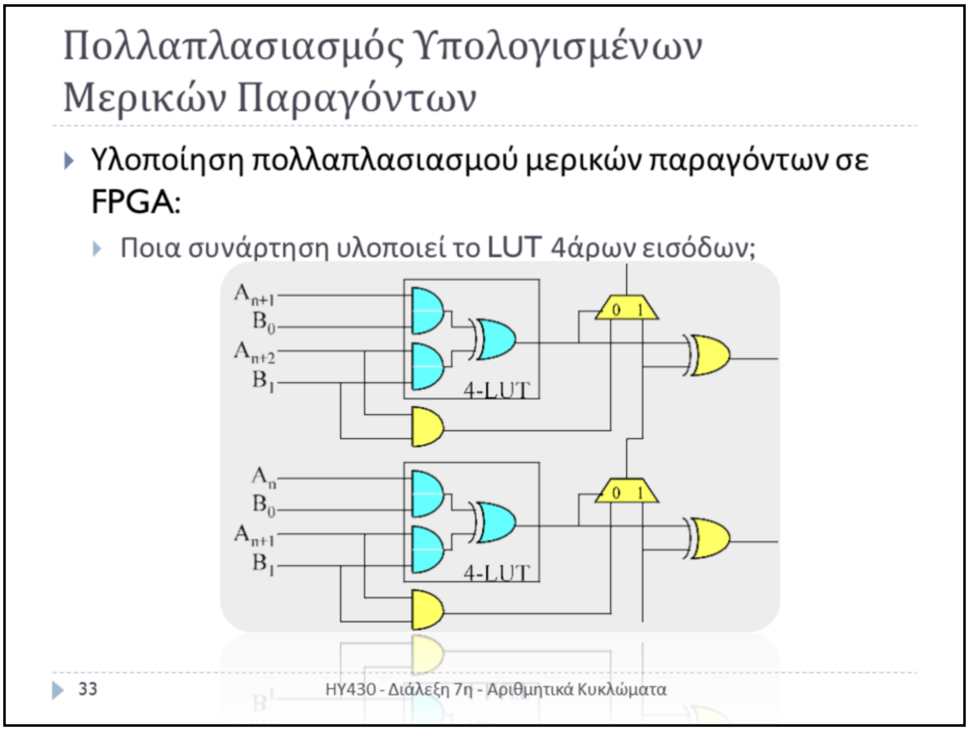
{B1, B0} = 11 => f = An (+) An+1, **πρόσθεση των 2-bit.**

Συνεπώς το LUT λειτουργεί ως ο πολυπλέκτης της προηγούμενης διαφάνειας.

Προσέξτε επίσης ότι αν B1 =1, δηλ. {B1, B0} = {1-} (όπου - = DC), τότε πραγματοποιείται ολίσθηση,

άσχετα αν το αποτέλεσμα είναι το άθροισμα ή ο αρχικός πολλαπλασιαστέος.

ΠΡΟΣΟΧΗ ότι η έξοδος της f είναι ισοδύναμη με την η συνθήκη της προώθησης, εφόσον έχουμε κρατούμενο
με τις συνθήκες **(f = 0 ΚΑΙ B1 . An+1) || (f = 1 ΚΑΙ cin)**.



Η έξοδος του LUT συνδέεται με τον πολυπλέκτη και την XOR. Η XOR (έξω από το LUT) χρησιμοποιείται για την

πρόσθεση του κρατούμενου. Η AND υπολογίζει τον 1ο υποψήφιο της ολίσθησης (B1 x An+1), ο 2ος προέρχεται

από τα προηγούμενα ψηφία. Ο πολυπλέκτης πραγματοποιεί την ολίσθηση του αποτελέσματος σε υψηλότερη ομάδα ψηφίων.

Παράδειγμα:

y7y6y5y4y3y2y1y0 **-> A7-A0**

x {10}{11}{01}{00} **-> B1 B0**

{00} x y7y6y5y4y3y2y1y0 = 00000000

{01} x y7y6y5y4y3y2y1y0 = y7y6y5y4y3y2y1y0 (και μετά << 2)

{11} x y7y6y5y4y3y2y1y0 = (y7y6y5y4y3y2y1y0 << 1) + y7y6y5y4y3y2y1y0 (και μετά << 4)

{10} x y7y6y5y4y3y2y1y0 = (y7y6y5y4y3y2y1y0 << 1) (και μετά << 6)

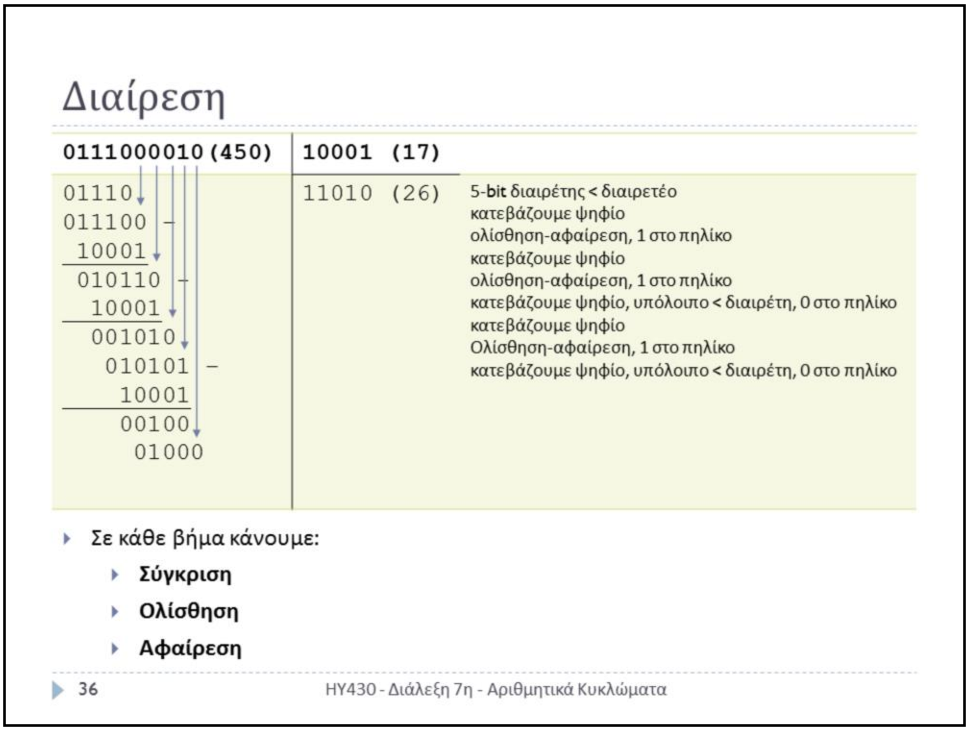
Περιεχόμενα

- Κυκλώματα Πρόσθεσης
  - Half-adder
  - Full-Adder
  - Σειριακό Κρατούμενο – Ripple Carry
  - Πρόγνωση Κρατουμένου – Carry Lookahead
- Κυκλώματα Πολλαπλασιασμού
  - Ολίσθηση και Πρόσθεση – Shift and Add
  - Πολλαπλασιασμός με Πίνακα Αποτελεσμάτων
  - Μερικών Παραγόντων – Partial Products
  - Υπολογισμένων Μερικών Παραγόντων – Computed Partial Products
- Διαίρεση

Διαίρεση

0111000010 (450)	10001 (17)

Πώς κάνουμε διαίρεση;



Αφαίρεση αλλάζοντας το πρόσημο με το δυικό συμπλήρωμα (2’s complement):

11100 (28) – = 11100 + = 011100 (28)

10001 (17) = (01110 + 1) = 101111 (-32+15=-17) = [1]001011

Το αριστερότερο ψηφίο (κρατούμενο υπερχείλισης) δεν χρησιμοποιείται, η τιμή 1 προκύπτει από την αναπαράσταση των αρνητικών.

Άλλα Παραδείγματα Διαίρεσης:

101010/10,

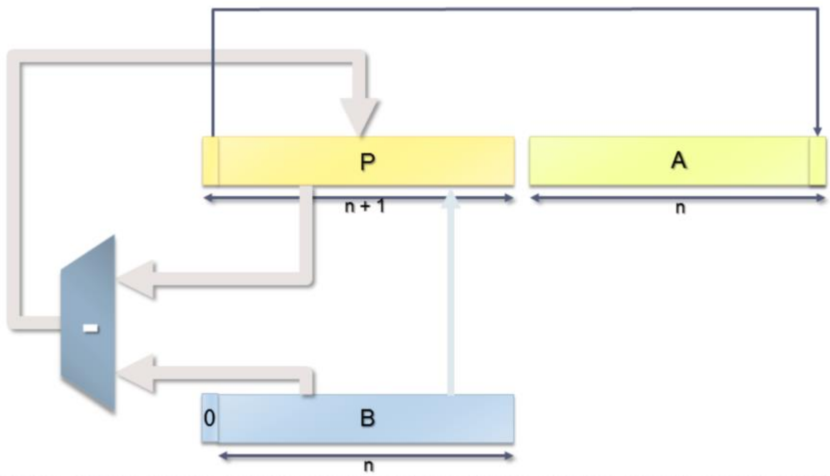
Δεκαδικό: 1500/15

Αλγόριθμος Διαίρεσης

```
LONG_DIVISION(D, d, q, r)
// Είσοδοι - D : Διαιρετέος, d : διαιρέτης, Έξοδοι – q : πηλίκο, r : υπόλοιπο
{
   n = MSB(D); m = (n - LENGTH(d)); x = 0; Dt = D;
   do
   {
      while (Dt[n:m] < d) // υπόλοιπο < διαιρέτη //
         q[x++] = 0; m = m - 1; // 0 στο πηλίκο, κατεβάζουμε ψηφίο //
      q[x++] = 1; // 1 στο πηλίκο //
      r[n-m:0] = Dt[n:m] - d; // νέο υπόλοιπο //
      m = m - 1; // κατεβάζουμε ψηφίο //
      Dt[n:0] = {r, D[m-1:0]};// συνένωση υπολοίπου με διαιρετέο //
   } while (r > d);
   return (q[0:x], r[n-m:0]);
}
```

Δυαδικός Διαιρέτης

- Πανομοιότυπη Διάταξη με τον Πολλαπλασιαστή



Δυαδικός Διαιρέτης

- Για να Υπολογίσουμε το A/B
- Αρχικοποίηση:
  - Αποθηκεύουμε το A στον Καταχωρητή A,
  - Το B στον Καταχωρητή B,
  - Μηδενίζουμε τον P
- Αλγόριθμος:
  - Για n bits, n επαναλήψεις:

1. Ολισθαίνουμε το Ζεύγος (P,A) 1 bit αριστερά
2. Εκτελούμε P = P – B
3. Αν το αποτέλεσμα < 0, θέτουμε A_0 σε 0, αλλιώς σε 1
4. Αν το αποτέλεσμα < 0, επαναφέρουμε την παλιά τιμή του P Εκτελώντας P = P + B