# Μοντέλο προσομοίωσης λογικών κυκλωμάτων - Switching activity computation 2

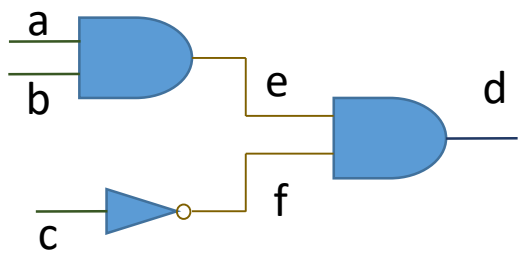
Vasileios Tenentes

University of Ioannina

# Όμως τι γίνεται με κυκλώματα πιο πολύπλοκα από μια λογική πύλη;

- Για να βρούμε το switching activity ενός κυκλώματος πρέπει να βρούμε το switching activity όλων των δομικών του **στοιχείων**. Τα **δομικά στοιχεία** μπορεί να είναι πιο πολύπλοκα από λογικές πύλες αλλά στην περίπτωσή μας θα μας απασχολήσουν μόνο οι λογικές πύλες

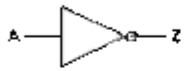
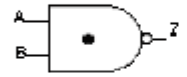
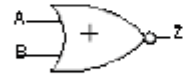
Παράδειγμα




Με γνωστό φόρτο εργασίας

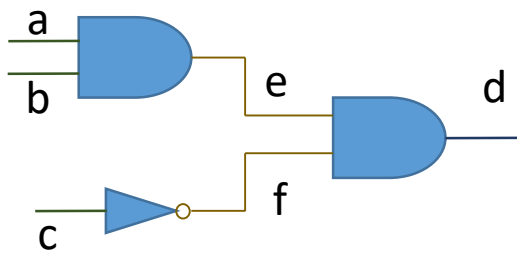
| | φόρτος εργασίας | | | Ενδιάμεσες έξοδοι | | Τελική έξοδος |
|---|---|---|---|---|---|---|
| χρόνος | a | b | c | e | f | d |
| t1 | 0 | 1 | 0 | 0 | 1 | 0 |
| t2 | 1 | 1 | 0 | 1 | 1 | 1 |
| t3 | 1 | 0 | 0 | 0 | 1 | 0 |
| t4 | 0 | 0 | 1 | 0 | 0 | 0 |

**Βασικές Λογικές Πύλες**

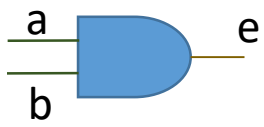
| Ονομασία | Σύμβολο | Σχέση | Πίνακας αληθείας A | B | Z |
|---|---|---|---|---|---|
| AND |  | $Z=A \bullet B$ | 0 | 0 | 0 |
| | | | 0 | 1 | 0 |
| | | | 1 | 0 | 0 |
| | | | 1 | 1 | 1 |
| OR |  | $Z=A+B$ | 0 | 0 | 0 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 1 |
| NOT |  | $Z=\overline{A}$ | 0 | | 1 |
| | | | 1 | | 0 |
| NAND |  | $Z=\overline{A \bullet B}$ | 0 | 0 | 1 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 0 |
| NOR |  | $Z=\overline{A+B}$ | 0 | 0 | 1 |
| | | | 0 | 1 | 0 |
| | | | 1 | 0 | 0 |
| | | | 1 | 1 | 0 |
| XOR |  | $Z=A \oplus B$ | 0 | 0 | 0 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 0 |
| XNOR |  | $Z=\overline{A \oplus B}$ | 0 | 0 | 1 |
| | | | 0 | 1 | 0 |
| | | | 1 | 0 | 0 |
| | | | 1 | 1 | 1 |

# Ανάλυση και των ενδιάμεσων εξόδων

| | φόρτος εργασίας | | | Ενδιάμεσες έξοδοι | | Τελική έξοδος |
|---|---|---|---|---|---|---|
| χρόνος | a | b | c | e | f | d |
| t1 | 0 | 1 | 0 | 0 | 1 | 0 |
| t2 | 1 | 1 | 0 | 1 | 1 | 1 |
| t3 | 1 | 0 | 0 | 0 | 1 | 0 |
| t4 | 0 | 0 | 1 | 0 | 0 | 0 |



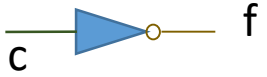

**έξοδος e**




| t | a | b | e |
|---|---|---|---|
| t1 | 0 | 1 | 0 |
| t2 | 1 | 1 | 1 |
| t3 | 1 | 0 | 0 |
| t4 | 0 | 0 | 0 |

**Esw(e)=2/3**

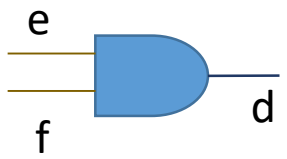
**έξοδος f**




| t | c | f |
|---|---|---|
| t1 | 0 | 1 |
| t2 | 0 | 1 |
| t3 | 0 | 1 |
| t4 | 1 | 0 |

**Esw(f)=1/3**

**έξοδος d**



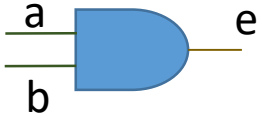

| t | e | f | d |
|---|---|---|---|
| t1 | 0 | 1 | 0 |
| t2 | 1 | 1 | 1 |
| t3 | 0 | 1 | 0 |
| t4 | 0 | 0 | 0 |

**Esw(d)=2/3**

Θυμηθείτε: **Esw=(# of switches) / (max # of switches)**

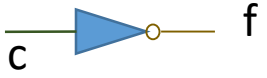
# Ανάλυση και των ενδιάμεσων εξόδων

**έξοδος e**



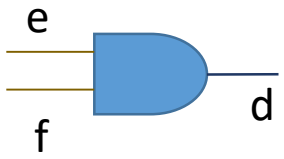

**Esw(e)=2/3**

**έξοδος f**



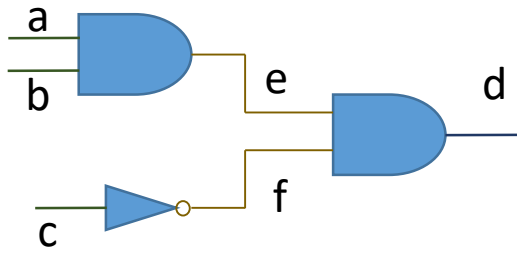

**Esw(f)=1/3**

**έξοδος d**




**Esw(d)=2/3**

**circuit**




Πρέπει να εφαρμόσουμε τον παρακάτω τύπο σε κάθε στοιχείο ξεχωριστά με βάση το δικό του $C_L$ ώστε να βρούμε τα Pd(e), Pd(f) και Pd(d)

$$P_d = V^2_{dd} * F_{clk} * C_L * E_{SW} * 0.5$$

$F_{clk}$ = clock frequency, $C_L$ = capacitance seen by gate,
$E_{SW}$ = gate switching activity (0-1 toggles)

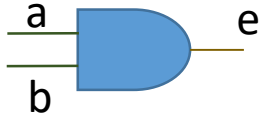
Και στο τέλος να βρούμε την τιμή:
**Pdynamic(circuit)= Pd(e)+Pd(f) +Pd(d)**

Σύνοψη: Για να βρούμε το switching activity ενός κυκλώματος με γνωστό φόρτο εργασίας πρέπει:

1. Να προσομοιώσουμε το κύκλωμα για το φόρτο εργασίας
2. Για κάθε έξοδο κάθε στοιχείου του κυκλώματος να υπολογίσουμε το switching activity

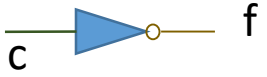
# Ανάλυση και των ενδιάμεσων εξόδων

**έξοδος e**



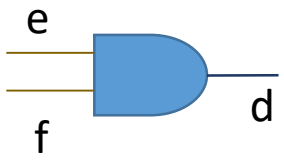

**Esw(e)=2/3**

**έξοδος f**



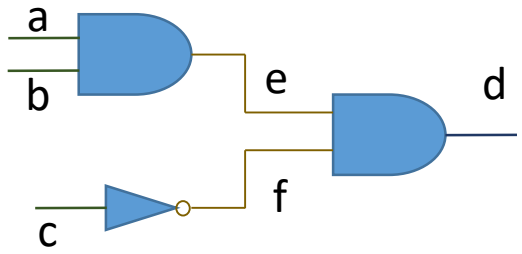

**Esw(f)=1/3**

**έξοδος d**




**Esw(d)=2/3**

**circuit**




```
//Αρχείο περιγραφής κυκλώματος
e =AND(a,b)
f  =NOT(c)
d =AND(e,f)
// Αρχείο περιγραφής κυκλώματος 2
AND(d, e, f)
NOT(f, c)
AND(e, a, b)
```

Πρέπει να εφαρμόσουμε τον παρακάτω τύπο σε κάθε στοιχείο ξεχωριστά με βάση το δικό του $C_L$ ώστε να βρούμε τα Pd(e), Pd(f) και Pd(d)

$$P_d = V^2_{dd} * F_{clk} * C_L * E_{SW} * 0.5$$

$F_{clk}$ = clock frequency, $C_L$ = capacitance seen by gate,
$E_{SW}$ = gate switching activity (0-1 toggles)

Και στο τέλος να βρούμε την τιμή:
**Pdynamic(circuit)= Pd(e)+Pd(f) +Pd(d)**

```
function {e,f,d}=circuitmodel(a,b,c)
begin
  e=spAND(a,b)
  f=spNOT(c)
  d=spAND(e,f)

  return {e,f,d}
end
```

# Περιγραφή Στη Μνήμη και Προσομοίωση Κυκλωμάτων

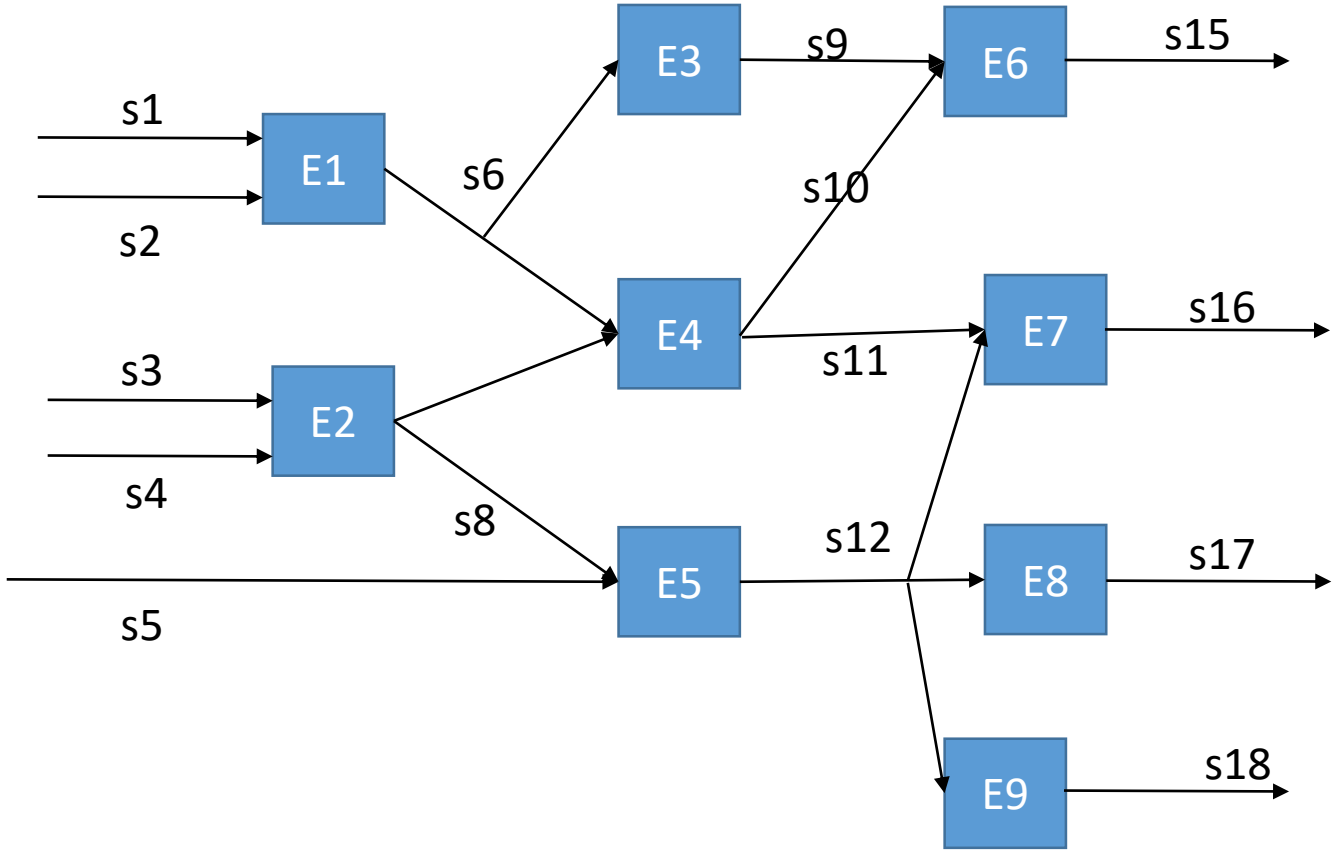
Έστω ότι έχουμε όλα τα Elements σε έναν πίνακα:

ElementsTable={E1, E2, E3, E4, E5, E6, E7, E8, E9} με τη σωστή σειρά που πρέπει να τα επεξεργαστούμε.

Ομοίως χρειαζόμαστε έναν πίνακα με τα signals:

SignalsTable={s1,s2,s3,s4,….,s18}

Και χρειαζόμαστε έναν τρόπο να ξέρουμε ποια Elements διαβάζουν ποια signals και ποια Elements γράφουν ποια signals.




Hints/κανόνες για απλούστευση του simulator:

1. ένα σήμα γράφεται ΜΟΝΟ από ΕΝΑ Element
2. Κάθε Element γράφει ΜΟΝΟ ένα signal (αυτό είναι απλούστευση, υπάρχουν πολύπλοκες λογικές πύλες με περισσότερες εξόδους)
3. Ένα SIGNAL μπορεί να διαβαστεί από πολλά Elements

# Περιγραφή Πολύπλοκων Κυκλωμάτων στη Μνήμη

Και χρειαζόμαστε έναν τρόπο να ξέρουμε ποια Elements διαβάζουν ποια signals και ποια Elements γράφουν ποια signals.

**//μπορεί να είναι πίνακας από κωδικοποιημένες τιμές (enumeration) ή πίνακας από strings**

**ElementTypes={NOT, AND, OR, XOR, NAND, NOR, XNOR}**

**// Πίνακας που κρατάει στοιχεία της δομής Element**

**ElementsTable={E1, E2, E3, E4, E5, E6, E7, E8, E9}**

**//πίνακας που κρατάει τις τιμές των σημάτων**

**SignalsTable={s1,s2,s3,s4,….,s18}**

//Χρειαζόμαστε μια αναπαράσταση για το Element

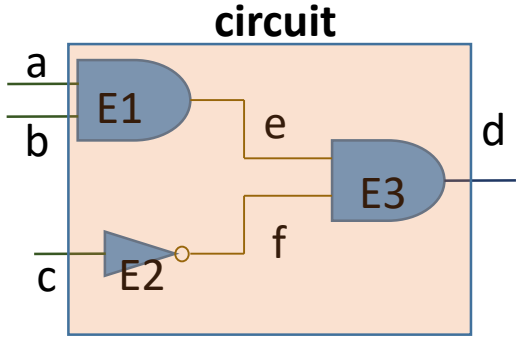
//μπορεί να είναι κλάση/δομή/κτλ.

*Δομή* Element

τιμή τύπος; //index προς το ElementTypes

πίνακας inputs; // πίνακας από indexes προς το SignalsTable

τιμή output; // index προς το SignalsTable




**Παράδειγμα Περιγραφής**:

**SignalsTable={0,0,0,0,0,0}**

**//a,b,c,d,e,f**

**E1.type='AND'; E1.inputs=[1,2]; E1.output=5;**

**E2.type='NOT'; E2.inputs=[3]; E2.output=6;**

**E3.type='AND'; E3.inputs=[5,6]; E3.output=4;**

**ElementsTable={E1,E2,E3}**

//μετά ακολουθεί το **processing**

*for i=1:size(ElementsTable)*

*process(ElementsTable[i])*

***Σημείωση για αρχικοποίηση***

Αν έχουμε βρει Top_inputs

Top_inputs=[1,2,3]

Τότε μπορούμε να αρχικοποίησουμε π.χ. σε a=1,b=0, c=1

Με τον εξής τρόπο

a=1 → **SignalsTable[1]=1**

b=0 -> **SignalsTable[2]=0**

c=1 → **SignalsTable[3]=1**

# Το Βασικό Processing της Προσομοίωσης

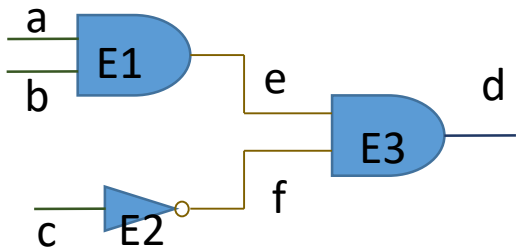
Αν ο ElementsTable είναι ταξινομημένος τότε το simulation γίνεται με το σχήμα:

*for i=1:size(ElementsTable)*

*process(ElementsTable[i])*

Η *process* είναι μια συνάρτηση που κοιτάει το Type, τα inputs και outputs του Element που εκτελεί την κατάλληλη συνάρτηση μεταφοράς με τις κατάλληλες εισόδους εξόδους. Πρέπει να γράψει το αποτέλεσμα για κάθε Element στον πίνακα SignalsTable

**circuit**




**Περιγραφή:**

topInputs={1,2,3}

SignalsTable={0,0,0,0,0,0}

//a,b,c,d,e,f

E1.type='AND'; E1.inputs=[1,2]; E1.output=5;

E2.type='NOT'; E2.inputs=[3]; E2.output=6;

E3.type='AND'; E3.inputs=[5,6]; E3.output=4;

ElementsTable={E1,E2,E3}

**Παράδειγμα της process με ψευδοκώδικα:**

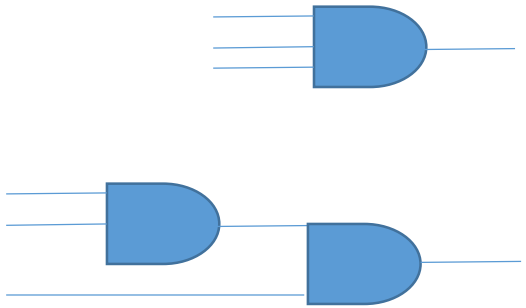
```
function process(element)
  if(element.type=='AND')
        SignalsTable[element.output]=spAND(SignalsTable[element.input[1]], SignalsTable[element.input[2]])
 elseif(element.type=='NOT')
       SignalsTable[element.output]=spNOT(SignalsTable[element.input[1]])
 elseif(element.type=='XOR')
       SignalsTable[element.output]=spXOR(SignalsTable[element.input[1]],
            SignalsTable[element.input[1]])
.
```

**Απλό παράδειγμα αρχικοποίησης τιμών σημάτων:**

```
Giveinputs(input1val, input2val, input3val)
  SignalsTable[topInputs[1]]=input1val
  SignalsTable[topInputs[1]]=input2val
 SignalsTable[topInputs[1]]=input3val
```

# Βοηθητικός κώδικας για N εισόδους

```
% N input gates
function s = spAND(varargin)
  printf("%d input AND\n", nargin)
  temp = 1;
  for i = 1:nargin
     temp = temp * varargin{i};
  end
  s = temp;
endfunction
```



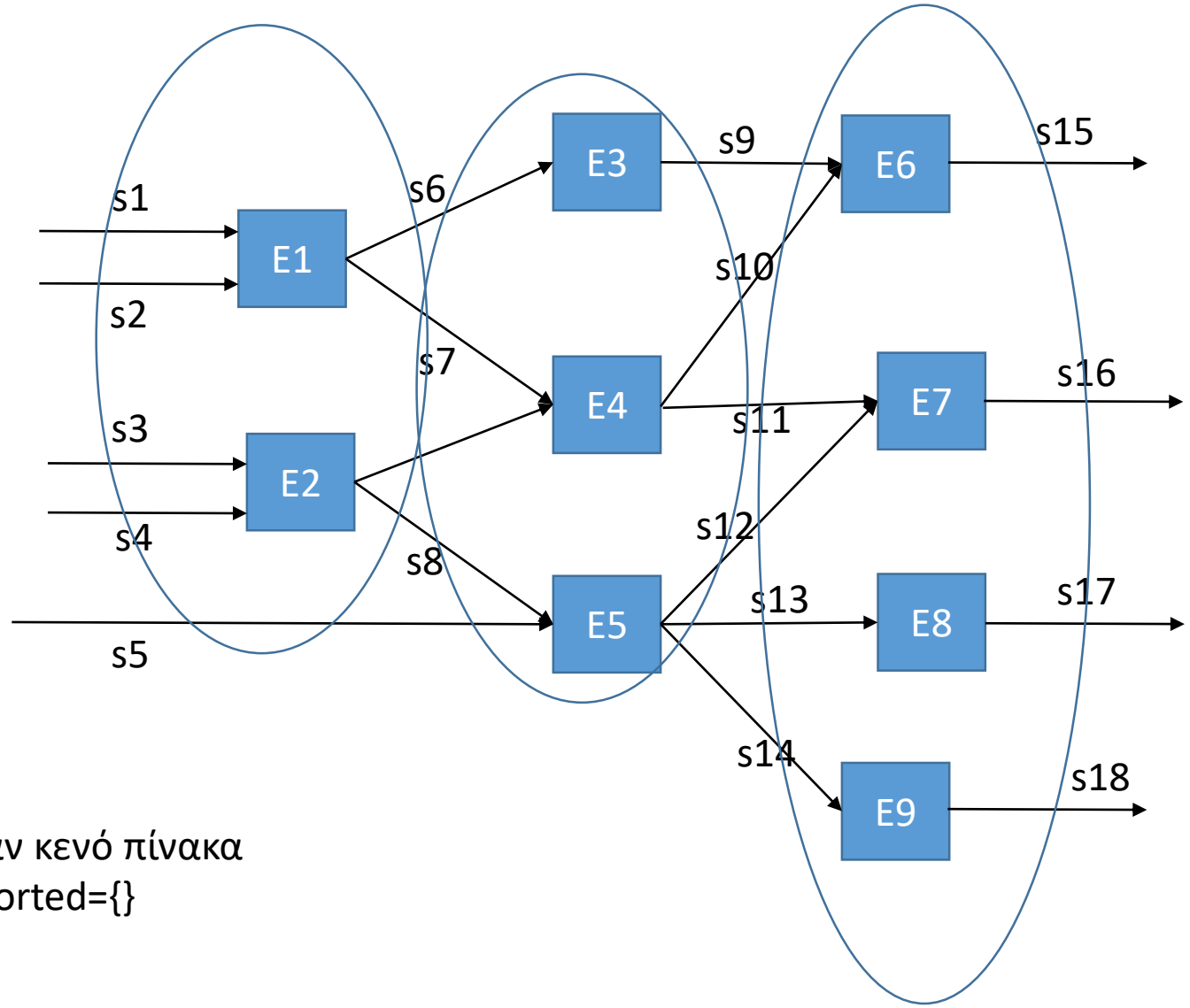
## Βοηθητικός κώδικας για wrapper sp συναρτήση

```
function s=spByType(type, varargin)
  if (type == 'NOT')
    s=spNOT(varargin{:});
  elseif (type == 'AND')
    s=spAND(varargin{:});
  elseif (type == 'OR')
    s=spOR(varargin{:});
  elseif (type == 'XOR')
    s=spXOR(varargin{:});
  elseif (type == 'NAND')
    s=spNAND(varargin{:});
  elseif (type == 'NOR')
    s=spNOR(varargin{:});
  elseif (type == 'XNOR')
    s=spNOR(varargin{:});
  else
    printf("unsupported  ")
    type
  endif
endfunction
```

# Και αν ο Πίνακας Elements δεν είναι ταξινομημένος;

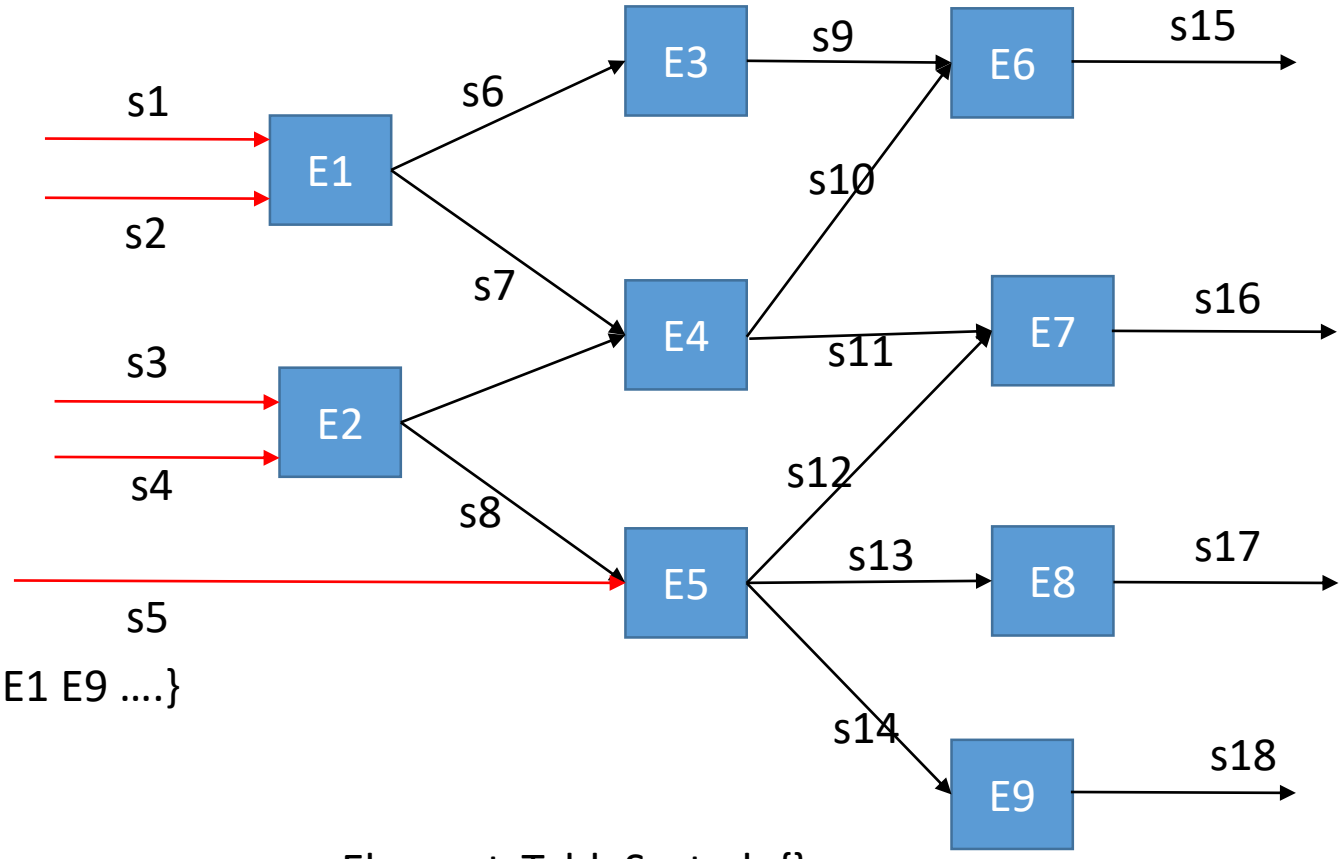
Όλα αυτά ισχύουν αν ο πίνακας είναι ταξινομημένος με τη σωστή σειρά επεξεργασίας

Πως ταξινομούμε τον πίνακα;




Ξεκινάμε με έναν κενό πίνακα
ElementsTableSorted={}

# Και αν ο Πίνακας Elements δεν είναι ταξινομημένος;




ElementsTable={E4 E6 E1 E9 ….}

ElementsTableSorted={}

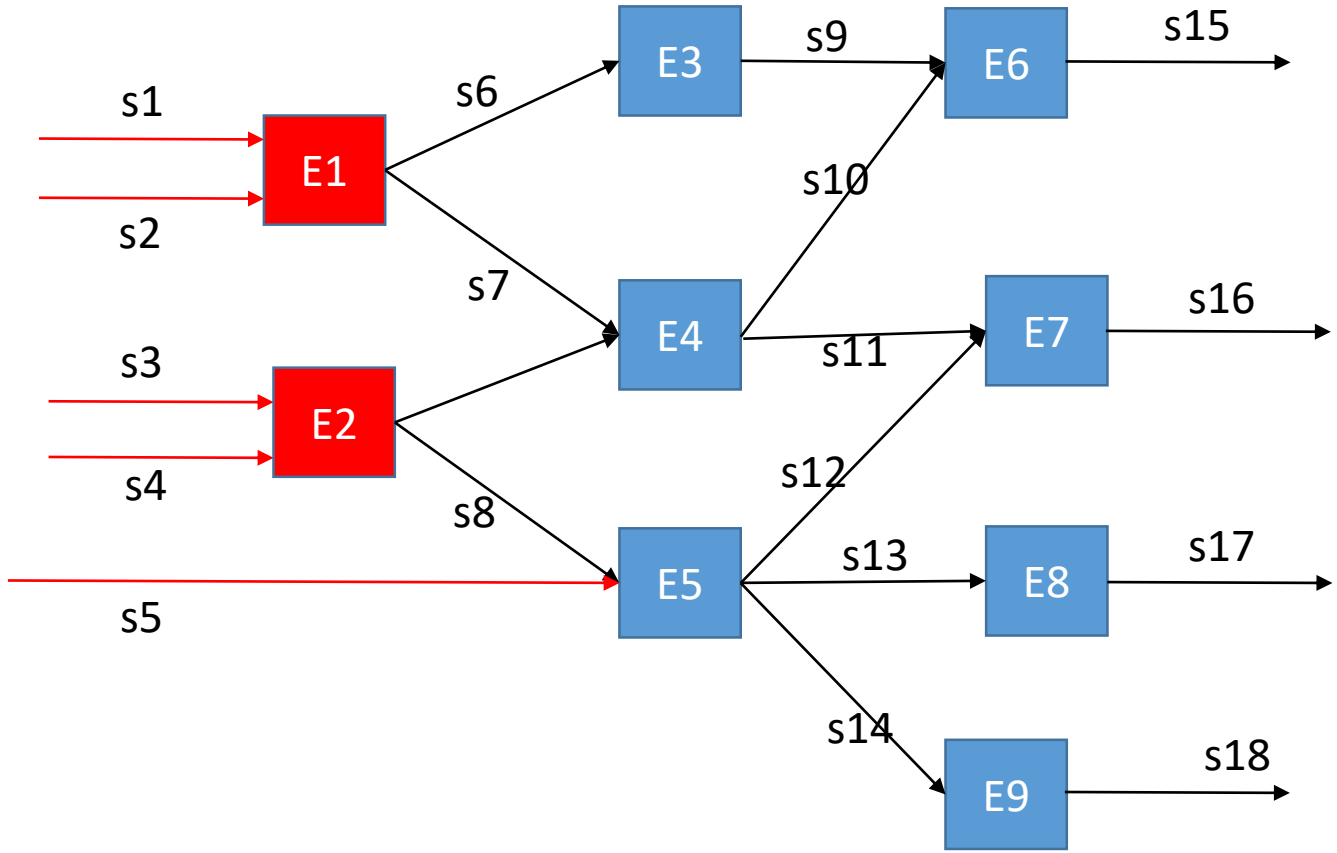
Βήμα 1ο: μαρκάρουμε τα **top level pin inputs**

Πώς μπορούμε να βρούμε τα top level pin inpits?

Α) είτε μπορούμε να τα δηλώσουμε με έναν πίνακα: top_inputs s1 s2 s3 s4 s5

Β) είτε μπορούμε να τα βρούμε αυτόματα: είναι τα signals που δεν εγγράφονται από κάποιο στοιχείο/πύλη

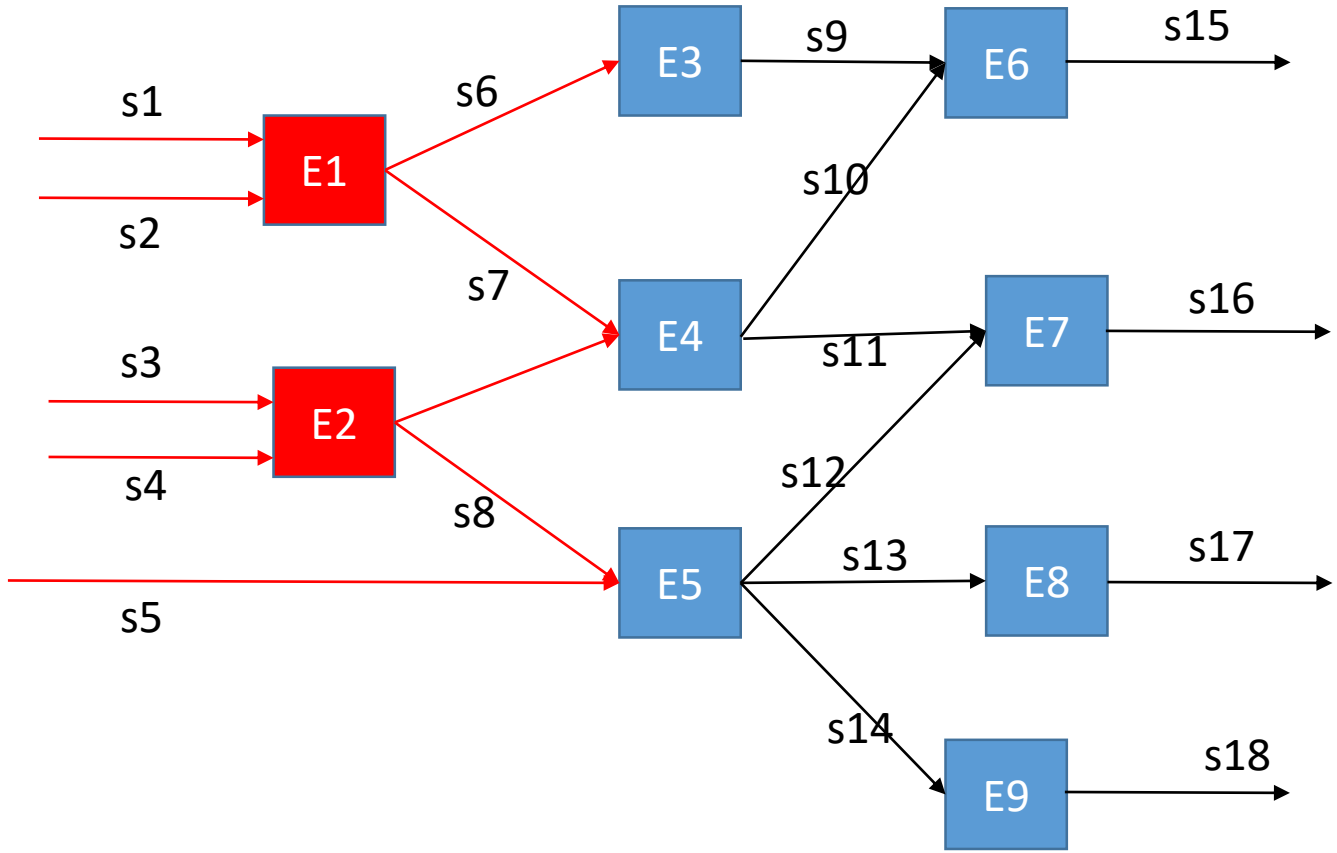
# Και αν ο Πίνακας Elements δεν είναι ταξινομημένος;




Βήμα 2ο: μαρκάρουμε όλα τα elements των οποίων όλες οι είσοδοι είναι μαρκαρισμένες και τα βάζουμε μέσα στον πίνακα ElementsTableSorted

ElementsTableSorted={E1,E2}

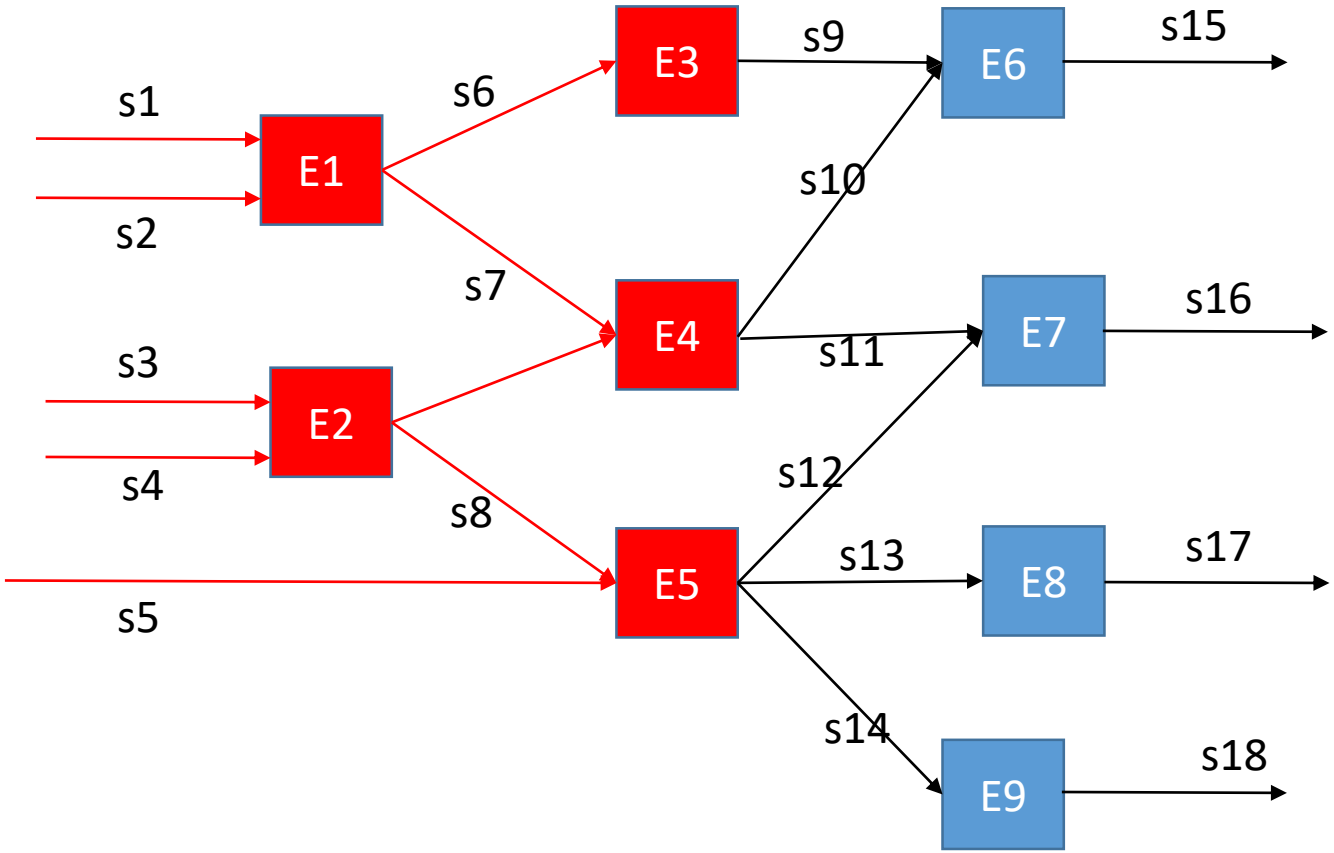
# Και αν ο Πίνακας Elements δεν είναι ταξινομημένος;




Βήμα 3ο: μαρκάρουμε τις εξόδους των Elements που βάλαμε στον πίνακα ElementsTableSorted

ElementsTableSorted={E1,E2}

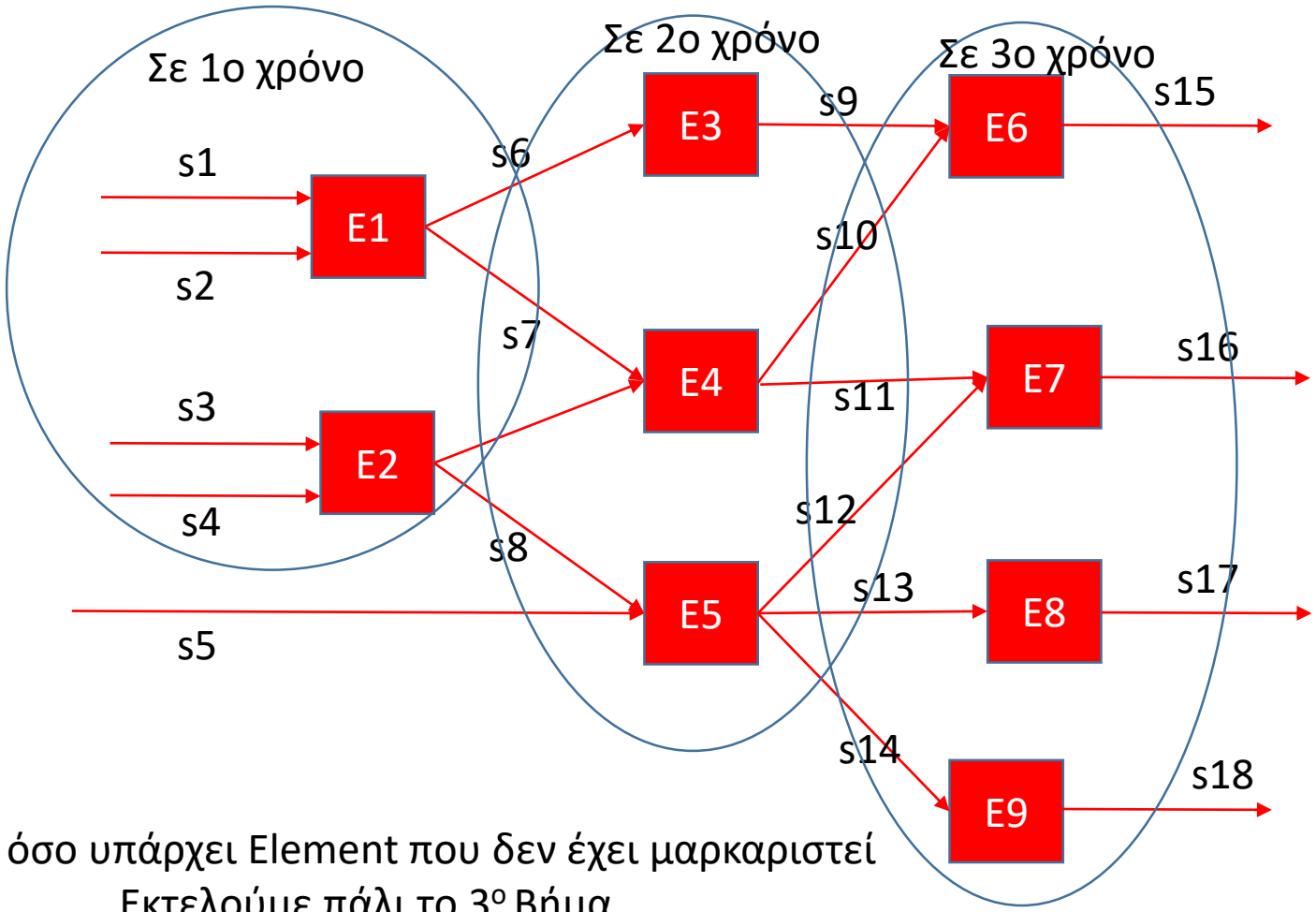
# Και αν ο Πίνακας Elements δεν είναι ταξινομημένος;




Βήμα 4ο: μαρκάρουμε όλα τα NEA elements (που δεν έχουμε ήδη μαρκάρει) των οποίων όλες οι είσοδοι είναι μαρκαρισμένες και τα βάζουμε μέσα στον πίνακα ElementsTableSorted

ElementsTableSorted={E1,E2,E4,E5,E3}

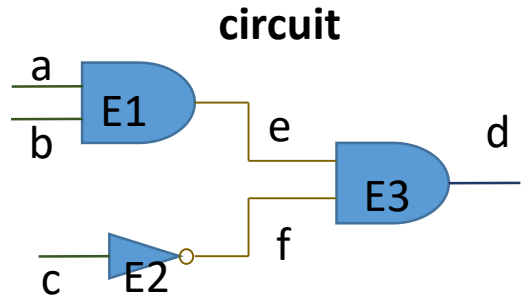
# Και αν ο Πίνακας Elements δεν είναι ταξινομημένος;




Για όσο υπάρχει Element που δεν έχει μαρκαριστεί
Εκτελούμε πάλι το 3ο Βήμα
Εκτελούμε πάλι το 4ο Βήμα

ElementsTableSorted={E1,E2,E3,E4,E5,E6,E7,E8,E9}

# Περιγραφής κυκλώματος από αρχείο




Αρχεία δομικής περιγραφής κυκλωμάτων

//Αρχείο περιγραφής κυκλώματος (Μορφή 1)
e =AND(a,b)
f  =NOT(c)
d =AND(e,f)

// Αρχείο περιγραφής κυκλώματος (Μορφή 2)
top_inputs a b c
AND d e f
AND e a b
NOT f c

# Παράδειγμα ψευδοκώδικα αρχικοποίησης εισόδων με τιμές για την περίπτωση Ν εισόδων

```
Circuit a;


a.loadFromFile(“circutfile.txt”)
print(a.topInputs)

a.giveInputs(timh1, timh2, time3,….)

Giveinputs(varargin)
  for i in inputsNumber
    SignalsTable[topInputs[i]]=varargin{i};
```