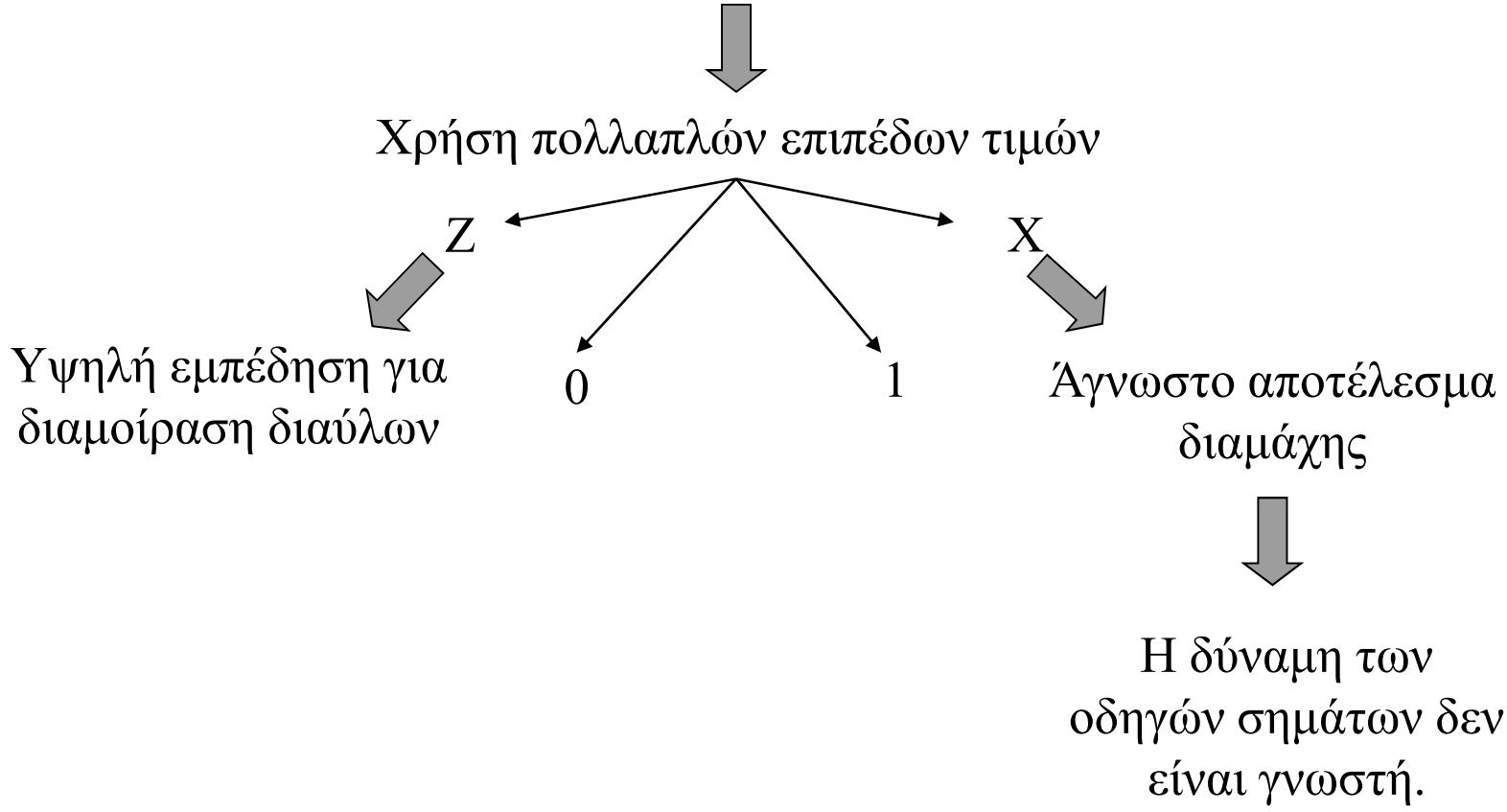
ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης
ΕΛΛΑΔΑ 2008
Ανάπτυξη παντού. Ανάπτυξη για όλους.

Μοντελοποίηση Επιπέδου Πύλης

(Peter Ashenden, The Students Guide to VHDL)

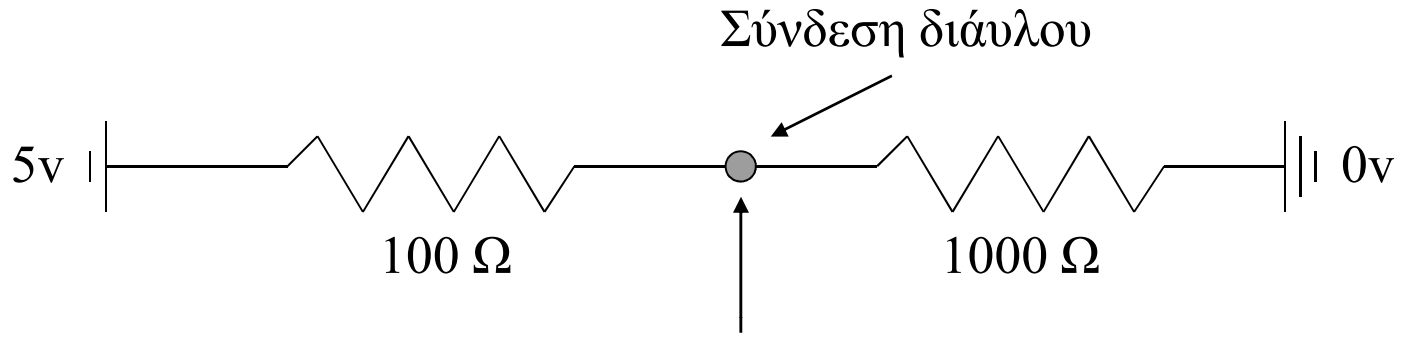
Πολλαπλά Επίπεδα Τιμών

Η κατάσταση μίας γραμμής δεν είναι πάντα 0 ή 1. Διαμάχες οδηγούν σε απροσδιοριστία.



Πολλαπλά Επίπεδα Τιμών

Σε πολλές τεχνολογίες η σύγκρουση οδηγών δίνει γνωστό αποτέλεσμα



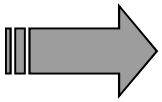
Ισχυρό 1, Αδύνατο 0

Ισοδυναμεί με 1

Ανάγκη μοντελοποίησης ισχύος σήματος

Πολλαπλά Επίπεδα Τιμών

```
type BIT_7 is
(    'x',      -- strong X (strong unknown)
     '0',      -- strong low
     '1',      -- strong high
     'Z',      -- high impedance
     'W',      -- weak unkonwn
     'L',      -- weak low
     'H');     -- weak high
```



7 επίπεδα τιμών

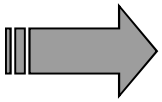
Η διαφοροποίηση τους θα διαφανεί στην επίλυση

7 Επίπεδα Τιμών

X	0	1	Z	W	L	H	
X	X	X	X	X	X	X	X
X	0	X	0	0	0	0	0
X	X	1	1	1	1	1	1
X	0	1	Z	W	L	H	Z
X	0	1	W	W	W	W	W
X	0	1	L	W	L	W	L
X	0	1	H	W	W	H	H

Std_Ulogic

```
type std_ulogic is
(    'U',     -- uninitialized
     'x',     -- strong X (strong unknown)
     '0',     -- strong low
     '1',     -- strong high
     'Z',     -- high impedance
     'W',     -- weak unkonwn
     'L',     -- weak low
     'H',     -- weak high
     '-');    -- don't care
```



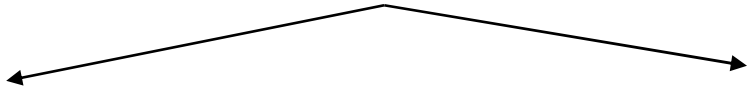
9 επίπεδα τιμών

9 Επίπεδα Τιμών

U	X	0	1	Z	W	L	H	-	
U	U	U	U	U	U	U	U	U	U
U	X	X	X	X	X	X	X	X	X
U	X	0	X	0	0	0	0	X	0
U	X	X	1	1	1	1	1	X	1
U	X	0	1	Z	W	L	H	X	Z
U	X	0	1	W	W	W	W	X	W
U	X	0	1	L	W	L	W	X	L
U	X	0	1	H	W	W	H	X	H
U	X	X	X	X	X	X	X	X	-

Μοντέλο ασύμμετρου χρονισμού

Σύμφωνα με τα εγχειρίδια κατασκευαστών μία τυπική πύλη έχει δύο μορφές καθυστέρησης



TPLHio: ο χρόνος που απαιτείται ώστε η έξοδος να αλλάξει από 0 σε 1 (ή από L σε H) όταν αλλάζει μία είσοδος.

TPHLio: ο χρόνος που απαιτείται ώστε η έξοδος να αλλάξει από 1 σε 0 (ή από H σε L) όταν αλλάζει μία είσοδος.

Μοντέλο ασύμμετρου χρονισμού

```
package timing is
type delay_parameters is (TPLHio, TPHLio, delta);
constant t_selection : delay_parameters := delta;
function t_choise (t_selection: delay_parameters)
return time;
end timing;

package body timing is
function t_choise (t_selection: delay_parameters)
return time is
  variable T_TPLHio, T_TPHLio: time;
begin
      case t_selection is
            when TPLHio => return 3ns;
            when TPHLio => return 5ns;
      end case;
end t_choise
end timing;
```

Μοντέλο ασύμμετρου χρονισμού

```
use work.timing.all;
entity and2_gate is
  port (I1, I2: in bit; O: out bit);
end and2_gate;

architecture behave of  and2_gate is
p0: process (I1, I2)
   variable new_and_value, old_and_value:bit;
begin
      new_and_value:= I1 and I2;
      if new_and_value='1' and old_and_value='0' then
            O<=new_and_value after t_choise(TPLHio);
      elsif new_and_value='0' and old_and_value='1' then
            O<=new_and_value after t_choise(TPHLio);
      end if;
      old_and_value:=new_and_value;
end process;
end behave;
```

Μοντέλο λογικής πύλης ΚΑΙ

```
package logic_gates is
  type delay_parameters is (TPLHio, TPHLio, delta);
  constant n: integer:=4;
  constant t_selection : delay_parameters := delta;
  variable t1, t2, t3: time;
  type BIT_7 is ('X', '0', '1', 'Z', 'W', 'L', 'H');
  type BIT_7_vector is array (natural range <>) of BIT_7;
  type BIT_7_table is array (BIT_7, BIT_7) of BIT_7;
  type BIT_7_t is array (BIT_7) of BIT_7;
  function t_choise (t_selection: delay_parameters) return time;
  function and_op (inputs: BIT_7_vector) return BIT_7;
end logic_gates;
```

Μοντέλο λογικής πύλης ΚΑΙ

```
package body logic_gates is
  function t_choise (t_selection: delay_parameters)  return time is
    variable T_TPLHio, T_TPHLio: time;
  begin
     case t_selection is
      when TPLHio => return t1;
      when TPHLio => return t2;
      when delta => return t3;
     end case;
end t_choise;

function and_op (inputs: BIT_7_vector) return BIT_7 is
   variable result: BIT_7:='1';
   constant and_table : BIT_7_table :=
      (('X',  '0',  'X',  'X',  'X',  '0',  'X'),
       ( '0',  '0',  '0',   '0',   '0',   '0',  '0'),
       ('X',  '0',  '1',   'X',  'X',  '0',  '1'),
       ('X',  '0',  'X',  'X',  'X',  '0',  'X'),
       ('X',  '0',  'X',  'X',  'X',  '0',  'X'),
       ('0',   '0',  '0',   '0',   '0',   '0',  '0'),
       ('X',  '0',  '1',   'X',  'X',  '0',  '1'));
begin
       for i in inputs'range loop
          result:=and_table (result, inputs(i));
          exit when result='0';
      end loop;
      return result;
end and_op;
end logic_gates;
```

Μοντέλο λογικής πύλης ΚΑΙ

```
use work.logic_gates.all;

entity and_gate is
   port (Input: in BIT_7_vector (n-1 downto 0); O: out BIT_7);
end and_gate;

architecture behave of and_gate is
begin
    assert n>=2   report "At least 2 inputs are required" severity failure;
    new_value:=and_op(Input);
    if (new_value='1' and old_value='0') or (new_value='H' and old_value='L') then
     O<=new_value after t_choise(TPLHio);
    elsif (new_value='0' and old_value='1') or (new_value='L' and old_value='H') then
     O<=new_value after t_choise(TPHLio);
    else O<=new_value after t_choise(delta);
    end if;
    old_value:=new_value;
end behave;
```