Library, package και subprograms

- Libraries
- Packages
- Subprograms
- Procedures
- Functions
- Overloading
- Αριθμητικά πακέτα
- Type conversion
- Shift operators
- Παράδειγμα Library - Package
- Ασκήσεις-Προβλήματα

Libraries

- Μέχρι τώρα στην αρχή κάθε προγράμματος χρησιμοποιούμε την ακόλουθη βιβλιοθήκη
  - **library** IEEE;

  μέρος της οποίας είναι το πακέτο (package)
  - **use** IEEE.std_logic_1164.**all**;

  το οποίο περιέχει τους ορισμούς των data types std_logic, std_ulogic, std_logic_vector και std_ulogic_vector.
- Εκτός της παραπάνω βιβλιοθήκης υπάρχουν οι
  - **library** work;
  - **library** std;

  οι οποίες πάντοτε είναι ορατές από τον χρήστη.
- Μιά βιβλιοθήκη αποτελείται από πολλά components αλλά και packages τα οποία περιέχουν functions, procedures, constants, types κτλ.

Packages

- Συνήθως σε μεγάλα ηλεκτρονικά σχέδια υπάρχουν πολλά functions, procedures, types, constants, components κτλ τα οποία πρέπει να χρησιμοποιηθούν από πολλά άλλα components ή διαφορετικούς σχεδιαστές. Αυτά όλα μπορούν να περιληφθούν σε ένα **πακέτο** (**package**).

Syntax for a package:

```
package <package_name> is
        [exported_subprogram_declarations]
        [exported_constant_declarations]
        [exported_components
        [exported_type_declarations]
        [attribut_declarations]
        [attribut_specifications]
end [<package_name>];

package body <package_name> is
        [exported_subprogram_bodies]
        [internal_subprogram_declarations]
        [internal_subprogram_bodies]
        [internal_constant_declarations]
        [internal_type_declarations]
end [<package_name>];
```

Declaration

Body

Packages

- Παράδειγμα

```
package mypack is
  function minimum (a,b:in  std_logic_vector)
              return  std_logic_vector;

  -- declarations of exported constant and types:
  constant maxint: integer := 16#FFFF#;
  type arithmetic_mode_type is (signed, unsigned);

end  mypack;

package body mypack is
  function minimum (a,b:in  std_logic_vector)
              return  std_logic_vector is
  begin
    if a<b then
       return a;
    else
       return b;
    end if;
  end  minimum;
end mypack;
```

Packages

- Ας υποθέσουμε ότι το προηγούμενο πακέτο έγινε compiled στην βιβλιοθήκη **mylib**. Για να το χρησιμοποιήσουμε πρέπει να δηλωθεί ως εξής

  Library ieee;

  Library mylib;

  Use ieee.std_logic_1164.**ALL**;

  Use mylib.mypack.**ALL**;

- Εάν θέλουμε να χρησιμοποιήσουμε μόνο την function minimum τότε μπορούμε να την δηλώσουμε ως εξής

  Library ieee;

  Library mylib;

  Use ieee.std_logic_1164.**ALL**;

  Use mylib.mypack.minimum;

Subprograms

- Υπάρχουν δύο ειδών υποπρογράμματα
  - **Procedures**
  - **Functions**
- Οι functions παράγουν μόνο μία τιμή ενώ οι procedures παράγουν περισσότερες ή και καμία.
- Τα υποπρογράμματα μπορούν να ορισθούν σε τρία μέρη μέσα στον κώδικα της VHDL.
  - Package
  - Architecture
  - Process
- Μέσα στα υποπρογράμματα ο κώδικας της VHDL είναι ακολουθιακός (sequential). Απαγορεύονται οι συντρέχουσες (concurrent) εντολές όπως **process**, **when-else**, **with-select** καθώς και η εντολή **wait** μέσα σε functions.

Procedures

In the declaration part of the package:

```
procedure <procedure_name>
        [ ( [object_class] arg_name {, arg_name}:
        [mode] type [ ( index_range [, index_range] ) ];
        { [object_class] arg_name {, arg_name}:
        [mode] type [ (index_range [, index_range] ) ] } ) ];
```

In the package body:

```
procedure <procedure_name>
        [ ( [object_class] arg_name {, arg_name}:
        [mode] type [ ( index_range [, index_range] ) ];
        { [object_class] arg_name {, arg_name}:
        [mode] type [ (index_range [, index_range] ) ] } ) ] is
        [constant_declarations]
        [variable_declarations]
        [type_declarations]

begin
        sequence_of_statements
end [<procedure_name>];
```

Procedures

- Μία procedure μπορεί να αλλάξει τις τιμές των **arguments** . Δέχεται ως **arguments** constants, variables και signals τα οποία μπορεί να χαρακτηρίζονται ως **in**, **out** και **inout**.

- Παράδειγμα

```
procedure ff ( signal clk, data : in   std_logic;
               signal q :         out std_logic) is
begin
  loop
     wait until clk='1';
     q <= data;
  end loop;
end ff;
```

Procedures

- Παράδειγμα

```
Architecture rtl of ex3 is
procedure calc ( a,b:                in  integer;
                 signal avg,max:     out integer) is
begin
   avg<=(a+b)/2;
   if a>b then
      max<=a;
   else
      max<=b;
   end if;
end calc;

begin
   calc(d1,d2,q1,q2);          -- Concurrent procedure call
   process(d3,d4)
   begin
     calc(d3,d4,q3,q4);        -- Sequential procedure call
   ...
   end process;
   ...
end;
```

Functions

In the declaration part of the package:

```
function <function_name>
        [ ( [ object_class] arg_name {, arg_name}:
        [IN ] type [ (index_range [, index_range] ) ]
        { [object_class] arg_name {, arg_name}:
        [IN ] type [ (index_range [, index_range] ) ] } ) ]
return type;
```

In the package body:

```
function <function_name>
        [ ( [ object_class] arg_name {, arg_name}:
        [IN ] type [ (index_range [, index_range] ) ]
        { [object_class] arg_name {, arg_name}:
        [IN ] type [ (index_range [, index_range] ) ] } ) ]
return type is
        [constant_declarations]
        [variable_declarations]
        [type_declarations]
begin
        sequence_of_statements
        return(expression);
end [<function_name>];
```

Functions

- Μία function δεν μπορεί να αλλάξει τις τιμές των **arguments** . Δέχεται ως **arguments** constants ή signals τα οποία μπορεί να χαρακτηρίζονται μόνο ως **in**.
- Παράδειγμα σε package

```
package mypack is
  function max (a,b:in std_logic_vector) return
    std_logic_vector;
end;

package body mypack is
  function max (a,b:in std_logic_vector) return
    std_logic_vector is
  begin
    if a>b then
      return a;
    else
      return b;
    end if;
  end;
end;
```

```
Use work.mypack.ALL;
...
Entity ex is
 port(...
end;

Architecture rtl of ex is
begin
  ...
  q<=max(d1,d2);              -- Concurrent function call

  process(data,g)
  begin
    data_out<=max(data,g);    -- Sequential function call
  end process;

  ...
end;
```

Functions

- Παράδειγμα σε architecture

```
Architecture rtl of ex2 is
   function max (a,b: in std_logic_vector) return
   std_logic_vector is
   begin
      if a>b then
         return a;
      else
         return b;
      end if;
   end max;
   ...
begin
   q<=max(d1,d2);                -- Concurrent function call
   process(data,g)
   begin
     data_out<=max(data,g);      -- Sequential function call
   end process;
  ...
end;
```

Functions

- Παράδειγμα σε process

```
Architecture rtl of ex3 is
begin
  process(data,g)
    function max (a,b: in std_logic_vector) return
    std_logic_vector is
    begin
      if a>b then
        return a;
      else
        return b;
      end if;
    end max;
  begin                                -- Start of process
    data_out<=max(data,g);
  end process;
  ...
end;
```

Overloading

- Πολλές φορές απαιτείται η εκτέλεση της ίδιας function αλλά με διαφορετικά data types στα arguments. Σε αυτή την περίπτωση η ίδια function πρέπει να ορισθεί πολλές φορές. Αυτό ονομάζεται overloading.
- Παράδειγμα

```
Library ieee;
Use ieee.std_logic_1164.ALL;

Package ex_pack is
 function max(a,b: in std_logic_vector) return
   std_logic_vector;
 function max(a,b: in vlbit_vector) return vlbit_vector;
 function max(a,b: in integer) return integer;
end;

Package body ex_pack is
 function max(a,b: in std_logic_vector) return
   std_logic_vector is
 begin
   if a>b then
     return a;
   else
     return b;
   end if;
 end;

function max(a,b:in vlbit_vector) return vlbit_vector is
 begin
   if a>b then
     return a;
   else
     return b;
   end if;
 end;

function max(a,b:in integer) return integer is
begin
 if a>b then
     return a;
   else
     return b;
   end if;
 end;
end;
```

Overloading

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use work.ex_pack.ALL;

Entity ex is
 port( a1,b1:  in   std_logic_vector(3 downto 0);
       a2,b2:  in   vlbit_vector(4 downto 0);
       a3,b3:  in   integer;

       c1:     out  std_logic_vector(3 downto 0);
       c2:     out  vlbit_vector(4 downto 0);
       c3:     out  integer);
end;

Architecture ex_beh of ex is
begin
  c1<=max(a1,b1);  -- Function max(a,b:std_logic_vector)
  c2<=max(a2,b2);  -- Function max(a,b:vlbit_vector)
  c3<=max(a3,b3);  -- Function max(a,b:integer)
end;
```

Αριθμητικά πακέτα

- **std_logic_unsigned** και **std_logic_signed**.

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_unsigned.ALL;
...
Architecture rtl of ex is
begin
   q<=a + b;      -- unsigned add
end;
```

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_signed.ALL;
...
Architecture rtl of ex is
begin
   q<=a + b;      -- signed add
end;
```

- **std_logic_arith**

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_arith.ALL;
...
Architecture rtl of ex is
begin
  q1<=unsigned(a) + unsigned(b);    -- unsigned add
  q2<=signed(a) + signed(b);        -- signed add
end;
```

Type conversion

- Παραδείγματα συναρτήσεων μετατροπής
  - **function** to_stdlogicvector (s : bit_vector)
    return std_logic_vector;
  - **function** conv_integer(arg : std_logic_vector)
    return integer;
  - **function** conv_std_logic_vector(arg : integer; size integer)
    return std_logic_vector;

- **Κανόνας** : Σε ένα καλό σχέδιο θα πρέπει να χρησιμοποιούνται όσο το δυνατόν λιγότερο οι συναρτήσεις μετατροπής.

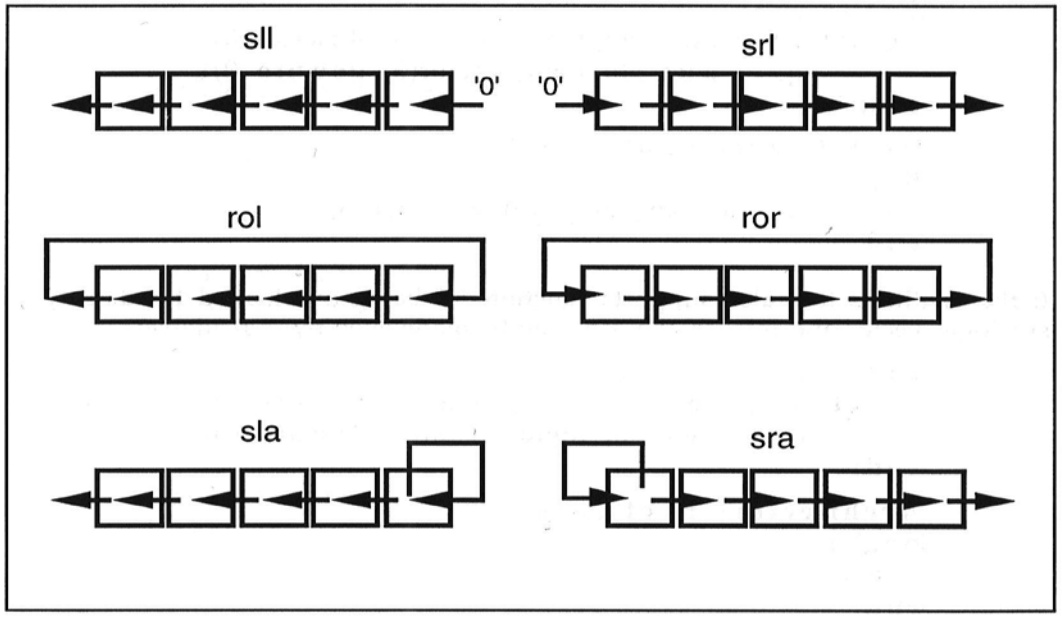
Shift operators

- Οι ακόλουθοι shift operators προστέθηκαν στην VHDL-93

  sll --Shift left **ror** --roll over right

  srl --Shift right **sla** --shift left, and keep value ‘right

  rol --roll over left **sra** --shift right, and keep value ‘left



Shift operators

- Παράδειγμα
  - **architecture** beh **of** ex **is**

    begin

      a <= “01101”;

      q1 <= a **sll** 1; -- q1 = “11010”

      q2 <= a **srl** 3; -- q2 = “00001”

      q3 <= a **rol** 2; -- q3 = “10101”

      q4 <= a **ror** 1; -- q4 = “10110”

      q5 <= a **sla** 2; -- q5 = “10111”

      q6 <= a **sra** 1; -- q6 = “00110”

    end;

Παράδειγμα Library - Package

- το ακόλουθο παράδειγμα είναι ένα **package** (κώδικας VHDL)
  το οποίο περιέχει δύο functions και δύο procedures
- το package εντάσσεται έπειτα σε μία **Library** (αντικείμενο του λογισμικού ISE)

```
--  File: c:\my designs\ex_mypack\SRC\mypack.VHD
--  Library - Package. Created by Design Wizard: 02/01/01 12:12:20
Library ieee;
Use ieee.std_logic_1164.ALL;
package mypack is
      function minimum (a, b : in std_logic_vector)
                              return std_logic_vector;
      function maximum (a, b : in std_logic_vector)
                              return std_logic_vector;
      procedure my_flip_flop (signal clk, data : in std_logic;
                                          signal q : out std_logic);
      procedure my_latch (signal enable, data_in : in std_logic;
                                        signal data_out : out std_logic) ;
end mypack;
```

Παράδειγμα Library - Package

```
package body mypack is
    function minimum (a, b : in std_logic_vector)
                        return std_logic_vector is
    begin
        if a<b then
                        return a;
        else
                        return b;
        end if;
    end minimum;
    function maximum (a, b : in std_logic_vector)
                        return std_logic_vector is
    begin
        if a<b then
                        return b;
        else
                        return a;
        end if;
    end maximum;
```

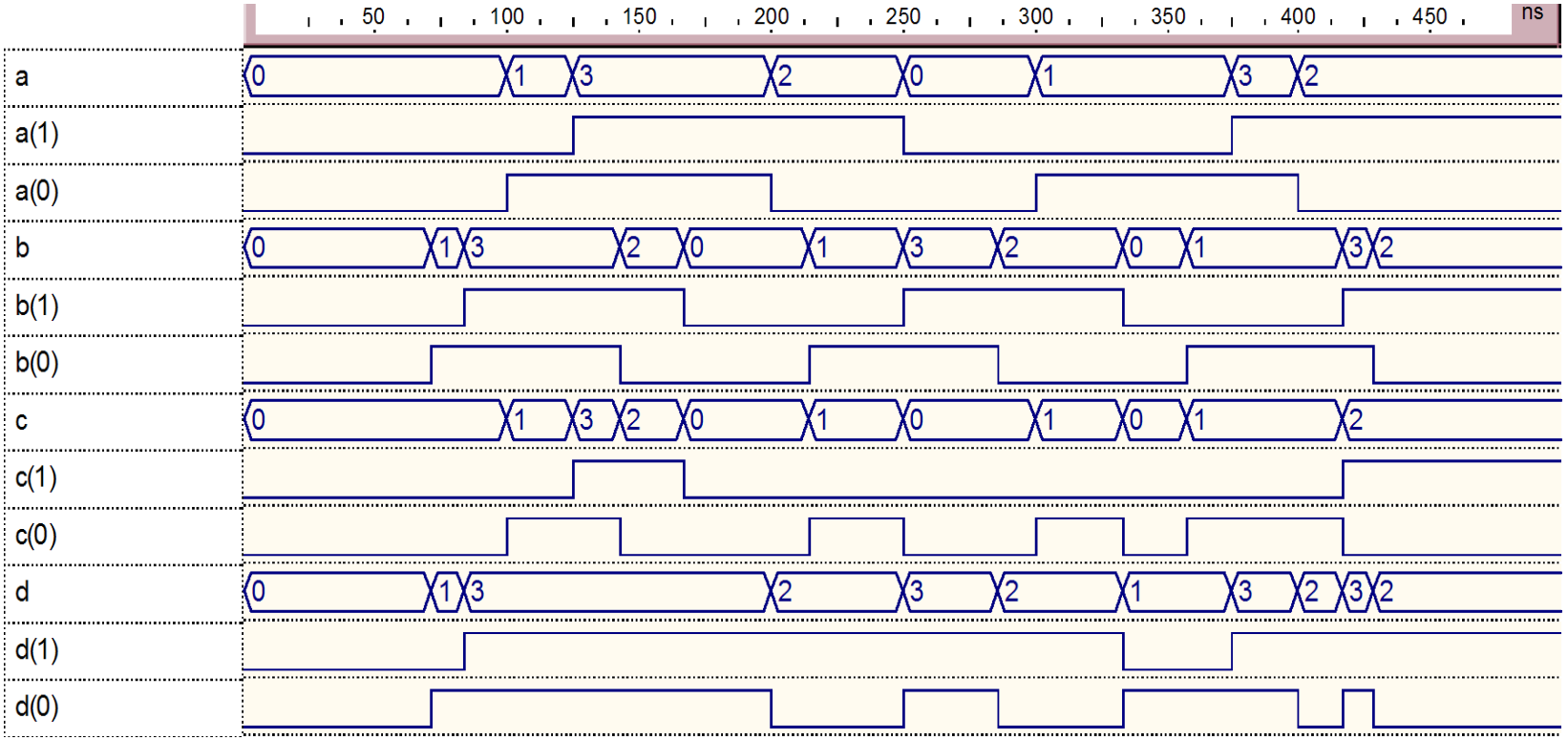
Παράδειγμα Library - Package

```
procedure my_flip_flop (signal clk, data : in std_logic;
                        signal q : out std_logic) is
  begin
    loop
      wait until clk = '1';
          q <= data;
    end loop;
  end my_flip_flop;

 procedure my_latch (signal enable, data_in : in std_logic;
                     signal data_out : out std_logic) is
  begin
    if enable='1' then
       data_out <=  data_in;
    end if;
  end my_latch;

end mypack;
```

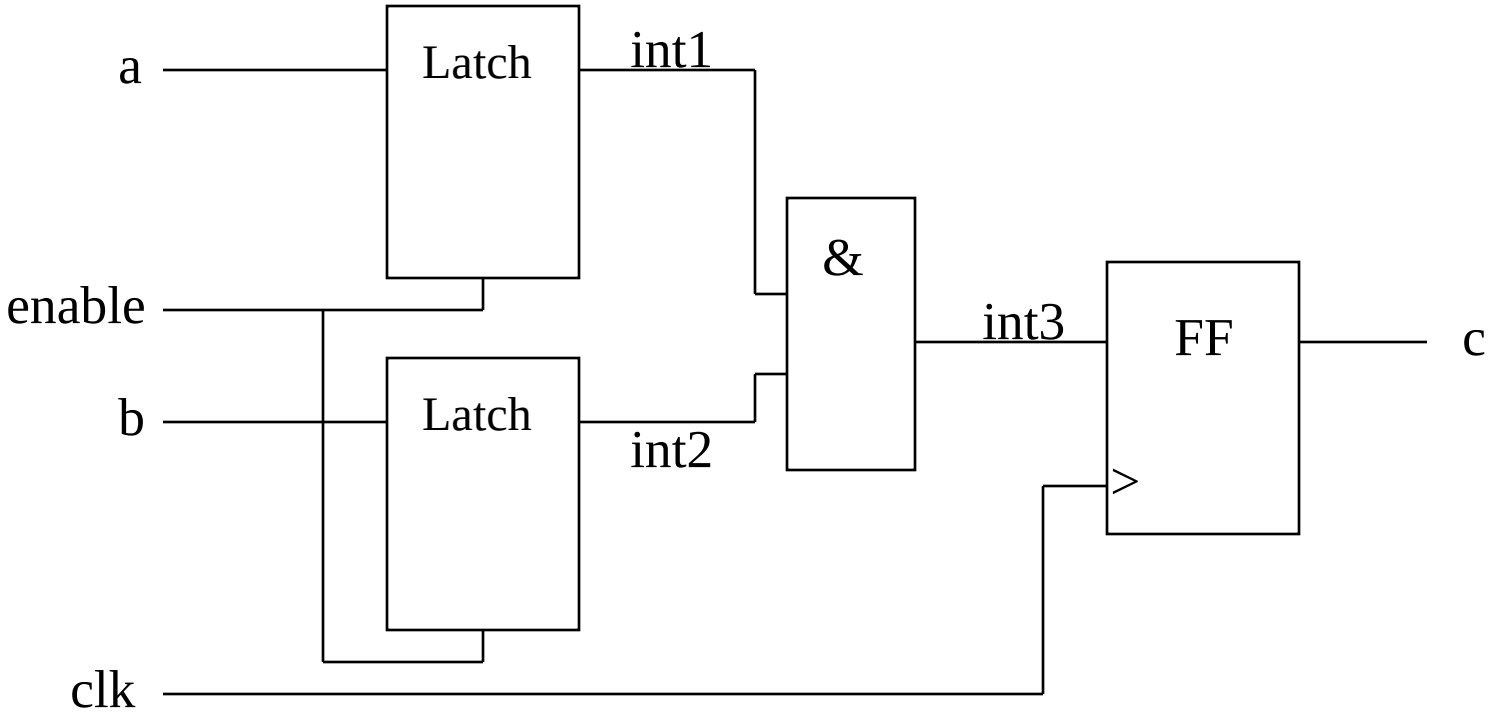
Παράδειγμα 1

```
--  File: c:\my designs\test_ex_mypack\SRC\test_ex_mypack.VHD
--  Created by Design Wizard: 02/01/01 12:27:40
library IEEE;
use IEEE.std_logic_1164.all;
library mylib;
use mylib.mypack.all;

entity test_mylib is
   port ( a, b : in STD_LOGIC_VECTOR (1 downto 0);
          c, d : out STD_LOGIC_VECTOR (1 downto 0));
end test_mylib;

architecture beh of test_mylib is
begin
  c <= minimum(a,b);
  d <= maximum(a,b);
end beh;
```

Παράδειγμα 1



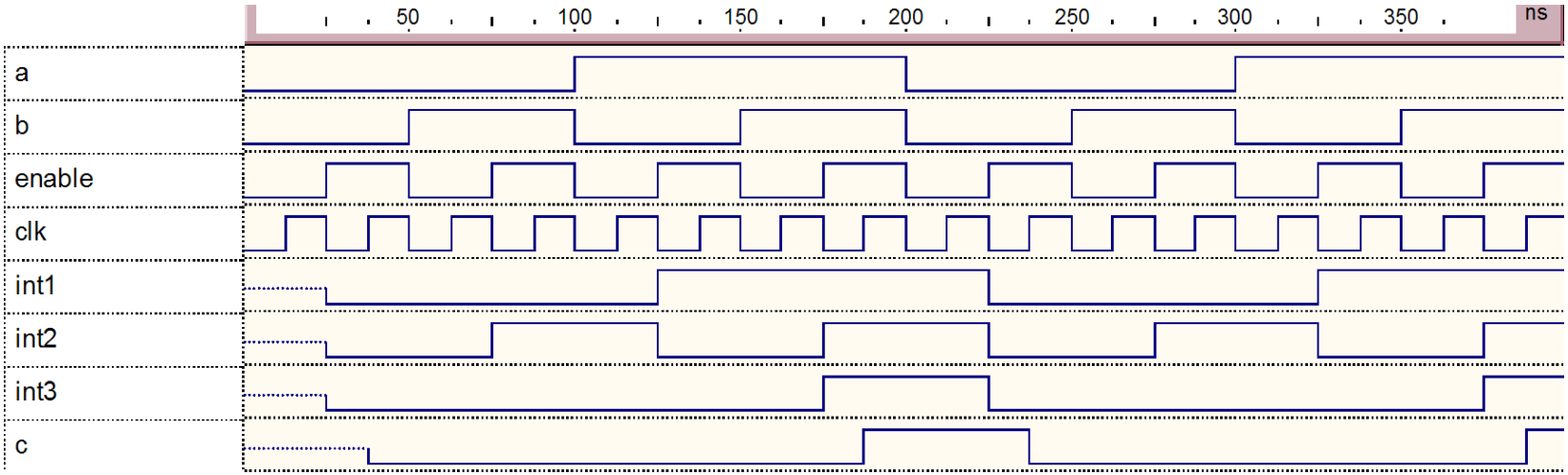
Παράδειγμα 2



Παράδειγμα 2

```
--  File: c:\my designs\test2_ex_mypack\SRC\comp_1.VHD
--  Created by Design Wizard: 02/02/01 18:31:11
library IEEE;
library mylib;
use IEEE.std_logic_1164.all;
use mylib.mypack.all;
entity comp_1 is
   port ( a, b, clk, enable: in STD_LOGIC;
            c: out STD_LOGIC);
end comp_1;
architecture comp_1_beh of comp_1 is
signal int1, int2, int3 : STD_LOGIC;
begin
      my_latch (enable, a, int1);
      my_latch (enable, b, int2);
      int3 <= int1 and int2;
      my_flip_flop (clk, int3, c);
end comp_1_beh;
```

Παράδειγμα 2



Ασκήσεις-Προβλήματα

1. Να δημιουργήσετε μία **Library** με **functions** για τις ακόλουθες συναρτήσεις Boole

AND2	NOT	AND3
NAND2	XOR2	NAND3
OR2	XNOR2	OR3
NOR2		NOR3

2. Να δημιουργήσετε μία **Library** με **procedures** για τις παραπάνω συναρτήσεις Boole

3. Να περιγράψετε σε VHDL τα κυκλώματα των ασκήσεων 1 και 3 του κεφαλαίου “Εισαγωγή στη Γλώσσα” χρησιμοποιώντας τις παραπάνω Libraries. Επαληθεύστε την λογική των κυκλωμάτων με εξομοίωση.

Ασκήσεις-Προβλήματα

- Να δημιουργήσετε μία **Library** που να περιγράφει τα ακόλουθα Latches και Flip-Flops.