ΕΙΣΑΓΩΓΙΚΟ ΕΓΧΕΙΡΙΔΙΟ ΓΙΑ ΣΧΕΔΙΑΣΜΟ

ΜΕ ΧΡΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ VHDL

Προετοιμασία: Παπαδόπουλος Γιώργος
Σούρδης Γιάννης

Για το μάθημα Οργάνωσης Υπολογιστών (ΑΡΥ301),

ΕΙΣΑΓΩΓΗ ΣΤΗ STRUCTURAL VHDL

Η VHDL είναι μια γλώσσα προγραμματισμού με την οποία περιγράφουμε την συμπεριφορά, την δομή και την υλοποίηση ενός ψηφιακού κυκλώματος ή συστήματος. Η γλώσσα αυτή συνιστά ένα εργαλείο CAD και έχει καθιερωθεί σαν ένα πρότυπο (standard) στη σχεδίασση ηλεκτρονικών κυκλωμάτων ASIC (Application Specific Integrated Circuits). Το γεγονός αυτό μας εγγυάται ότι οι νεότερες εκδόσεις εργαλείων σχεδίασης θα υποστηρίζουν το πρότυπο αυτό. Έτσι μια περιγραφή ενός κυκλώματος που αναπτύχθηκε με τα σημερινά εργαλεία σχεδίασης θα είναι μεταφέρσιμη μελλοντικά σε νέα εργαλεία σχεδίασης με ελάχιστες τροποποιήσεις. Επίσης οι περιγραφές κυκλωμάτων που αναπτύχθηκαν από διάφορους σσχεδιαστές θα είναι διαθέσιμες σε μια κοινή βάση δεδομένων η χρήση της οποίας θα μας διευκολύνει στην επίλυση παρόμοιων σχεδιαστικών προβλημάτων.

Ο όρος VHDL αποτελεί σύντμηση των λέξεων VHSIC Hardware Description Language και αναπτύχθηκε στις αρχές της δεκαετίας του 1980. Επίσης η λέξη VHSIC είναι σύντμηση των λέξεων Very High Speed Integrated Circuit.

Το κύριο πρότυπο της γλώσσας VHDL είναι το IEEE 1076-1987 και αποτελεί τη βάση για κάθε πρόγραμμα CAD με VHDL simulator και synthesis. Τα βελτιωμένα πρότυπα της γλώσσας αυτής είναι το IEEE 1076-1993 και το IEEE 1164. Επίσης στο τέλος του 1995 προστέθηκε στο IEEE 1074.4 που ονομάστηκε VITAL (VHDL Initiate Toward ASIC Libraries).

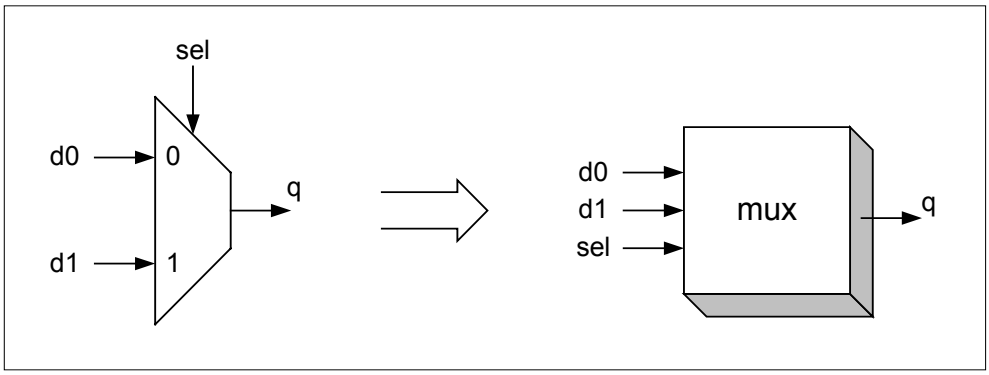
Η δομική περιγραφή σσε VHDL (structural VHDL) χρησιμοποιείται για την περιγραφή ενός κυκλώματος με βάση τα στοιχειώδη εξαρτήματα που περιλαμβάνονται σε μια βιβλιοθήκη εξαρτημάτων. Για παράδειγμα μια περιγραφή ενός κυκλώματος σε επίπεδο πύλης περιλαμβάνει εξαρτήματα όπως λογικές πύλες και

flip-flops τα οποία συνδέονται μεταξύ τους έτσι ώστε να σχηματίσουν τη λογική δομή του κυκλώματος. Η συνδεσμολογία αυτήονομάζεται netlist. Επίσης μία περιγραφή ενός κυκλώματος σε υψηλότερο επίπεδο περιλαμβάνει εξαρτήματα τα οποία είναι μεγάλες λειτουργικές δομές που μας επιτρέπουν να διαχωρίσουμε ένα μεγάλο και πολύπλοκο κύκλωμα σε μικρότερα μέρη που είναι εύκολο να αναλυθούν.

Όταν γράφουμε σε structural VHDL, ορίζουμε την λεκτική περιγραφή ενός netlist που είναι μια περιγραφή της συνδεσμολογίας των εξαρτημάτων με χρήση των κατάλληλων καλωδίων. Το κάθε ένα από τα ανώτερα εξαρτήματα μπορεί να αποτελείται από άλλα υποεξαρτήματα τα οποία είναι επίσης συνδεσμολογημένα σσύμφωνα με μια δεδομένη λογική δομή. Το αποτέλεσμα της χρήσης τέτοιων εξαρτημάτων και υποεξαρτημάτων είναι να δημιουργήσουμε ένα μοντέλο για το κύκλωμά μας με πολλαπλά δομικά επίπεδα. Τότε η ιεραρχία του μοντέλου αποκτά δενδρική μορφή και συνίσταται από δομικά επίπεδα, έκαστο των οποίων περιγράφεται σε structural VHDL.

Παράδειγμα – Δομική περιγραφή του MUX 2-to-1

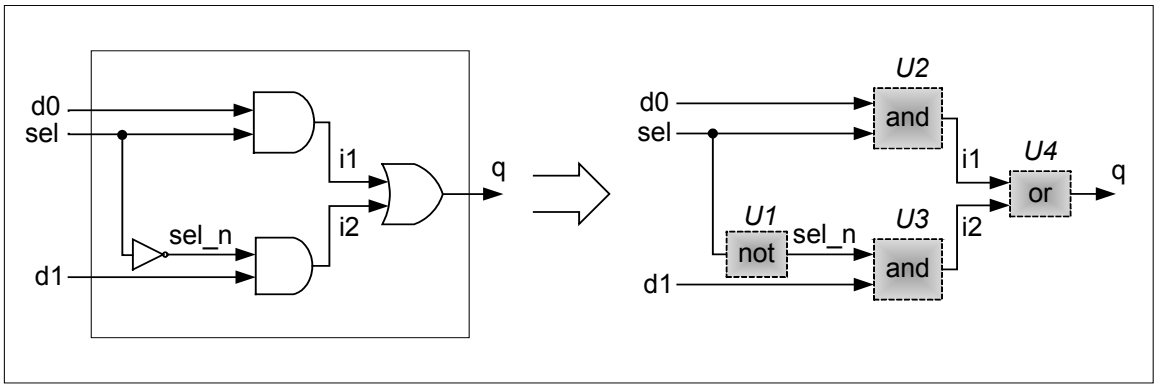
Το γραφικό σύμβολο του πολυπλέκτη δύο εισόδων (MUX 2-to-1) φαίνεται στο ακόλουθο σχήμα.



Ο ανωτέρω πολυπλέκτης συνιστά ένα εξάρτημα το οποίο ονομάζουμε “mux”. Το εξάρτημα αυτό παριστάνεται με ένα κουτί το οποίο έχει τρία σήματα εισόδου *d0, d1* και *sel* ένα σήμα εξόδου *q*, και τα οποία λαμβάνουν δυαδικές τιμές ‘0’ και ‘1’. Η περιγραφή σε VHDL του γραφικού συμβόλου του εξαρτήματος του πολυπλέκτη δίνεται κατωτέρω:

```
entity mux is
     port (
          d0,d1,sel       : in std_logic;
          q               : out std_logic
     );
end mux;
```

Το κυκλωματικό διάγραμμα του πολυπλέκτη με χρήση λογικών πυλών φαίνεται στο ακόλουθο σχήμα.



Παρατηρούμε ότι το δομικό διάγραμμα αποτελείται από δύο πύλες AND (U2, U3), μια πύλη NOT (U1) και μια πύλη OR (U4) διασυνδεδεμένες μεταξύ τους με τα εσωτερικά καλώδια i1, i2 και sel_n, και με τα εξωτερικά καλώδια d0, d1, sel και q. Η περιγραφή του ανωτέρω δομικού διαγράμματος του πολυπλέκτη σε VHDL δίνεται παρακάτω:

```
architecture struct_mux of mux is

-- component declaration
    component and_comp
     port (
          a,b  : in std_logic;
          c    : out std_logic
     );
    end component;
    component or_comp
     port (
          a,b  : in std_logic;
          c    : out std_logic
     );
    end component;
    component inv_comp
     port (
          a    : in std_logic;
          c    : out std_logic
     );
    end component;

-- internal signal declaration
     signal i1,i2,sel_n  : std_logic;

begin
-- component instantiation
     U1   : inv_comp
     port map (
          a    => sel,
          c    => sel_n
     );
     U2   : and_comp
     port map (
          a    => d0,
          b    => sel,
          c    => i1
     );
     U3   : and_comp
     port map (
          a    => sel_n,
          b    => d1,
          c    => i2
     );
     U4   : or_comp
     port map (
          a    => i1,
          b    => i2,
          c    => q
     );
end struct__mux;
```

Επίσης μια πιο απλη περιγραφή δίνεται στον κώδικα που ακολουθεί:

```
architecture struct_mux of mux is

begin
     q <= (d0 and sel) or (d1 and not sel);
end struct__mux;
```

ΕΙΣΑΓΩΓΗ ΣΤΗ BEHAVIORAL VHDL

Κάθε περιγραφή ενός σχεδίου κυκλώματος στη γλώσσα VHDL αποτελείται τουλάχιστον από ένα ζεύγος entity/architecture. Σε μεγάλα σχέδια γράφουμε πολλά ζεύγη entity/architecture τα οποία συνδέουμε μεταξύ τους για να σχηματίσουμε το πλήρες κύκλωμα.

Η σύνταξη του "entity" είναι:

```
entity entity-name is
     [port (interface-signal-declaration);]
end entity-name;
```

Η σύνταξη του "architecture" είναι:

```
architecture architecture-name of entity-name is
     [declarations] -- signals
     begin

          architecture-body

end architecture-name;
```

Στην behavioral VHDL, το architecture περιγράφει τη λειτουργία του κυκλώματος και έχει μορφή κατάλληλη για simulation και synthesis.

Η κύρια εντολή που χρησιμοποιείται μέσα στο σώμα του architecture και περιγράφει την συμπεριφορά του κυκλώματος σε συνάρτηση με το χρόνο είναι η "process".

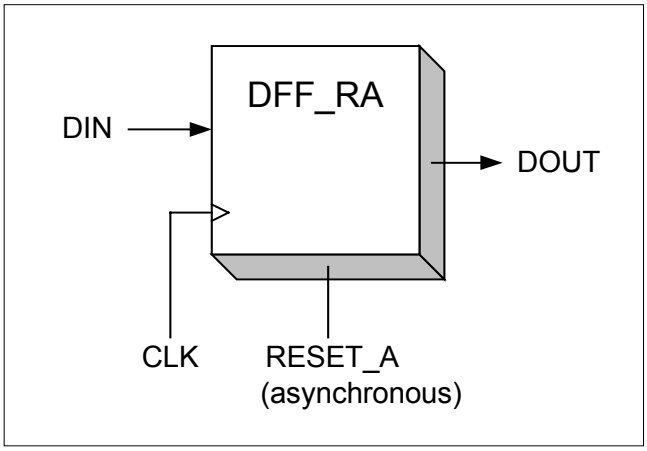
Η σύνταξη της “process” είναι:

```
[proces-label:] process (sensitivity-list)
     [declarations] -- variables
     begin

         sequential-statements

     end process [process-label];
```

Η εντολή “process” εκτελείται κατά τη διάρκεια του simulation μία φορά αρχικά και έπειτα κάθε φορά που αλλάζει τιμή σε ένα σήμα (signal) του sensitivity-list. Οι variables είναι σήματα εσωτερικά στην process και χρησιμοποιούνται για μεταφορά δεδομένων στα sequential statements. Τα sequential statements χρησιμοποιούνται για να περιγράψουν ακολουθιακή λογική που περιλαμβάνει στοιχεία μνήμης και συνδυαστική λογική.

Παράδειγμα – Δομική περιγραφή του D - FLIP FLOP

i) Το γραφικό σύμβολο του “D – FLIP FLOP with asynchronous RESET” και ο πίνακας αληθείας που περιγράφει τη λειτουργία του φαίνονται παρακάτω:



Inputs			*Output*	*Σχόλια*
RESET_A	**DIN**	**CLK**	**DOUT**	
1	X	X	0	Clear
0	1	↑	1	Load '1' (SET)
0	0	↑	0	Load '0' (RESET)

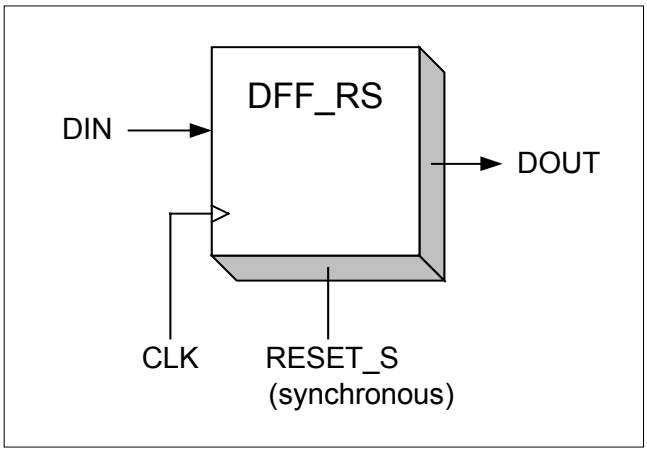
↑ = clock transition LOW to HIGH
X = don't care

Η περιγραφή συμπεριφοράς σε VHDL του μοντέλου του "D – FLIP FLOP with asynchronus RESET" δίνεται παρακάτω:

```
entity DFF_RA is
     port (
          CLK,DIN,RESET_A: in std_logic;
          DOUT           : out std_logic
     );
end DFF_RA;
```

```
architecture BEHAVIORAL of DFF_RA is

     begin

          process (CLK,RESET_A)
          begin
               if RESET_A='1' then
                    DOUT<='0';
               elsif (CLK'event and CLK='1') then
                    DOUT<=DIN;
               end if;
          end process;

end BEHAVIORAL;
```

ii) Το γραφικό σύμβολο του "D – FLIP FLOP with synchronous RESET" και ο πίνακας αληθείας που περιγράφει τη λειτουργία του φαίνονται παρακάτω:



Inputs			***Output***	***Σχόλια***
RESET_A	**DIN**	**CLK**	**DOUT**	
1	X	X	DOUT	Hold (do nothing)
1	X	↑	0	Clear
0	1	↑	1	Load '1' (SET)
0	0	↑	0	Load '0' (RESET)

↑ = clock transition LOW to HIGH
X = don't care

Η περιγραφή συμπεριφοράς σε VHDL του μοντέλου του "D – FLIP FLOP with synchronus RESET" δίνεται παρακάτω:

```
entity DFF_RS is
     port (
          CLK,DIN,RESET_S: in std_logic;
          DOUT           : out std_logic
     );
end DFF_RS;
```

```
architecture BEHAVIORAL of DFF_RS is

     begin

          process (CLK,RESET_S)
          begin
               if CLK'event and CLK='1' then
                    if RESET_S='1' then
                         DOUT<='0';
                    else
                         DOUT<=DIN;
                    end if;
               end if;
          end process;

end BEHAVIORAL;
```