HY220
Εργαστήριο Ψηφιακών Κυκλωμάτων

Εαρινό Εξάμηνο
2025

SystemVerilog:
Επιπλέον Χαρακτηριστικά

Καλύτερη οργάνωση του κώδικα

- Η πολυπλοκότητα των σχεδίων μεγαλώνει και η οργάνωση του κώδικα γίνεται σημαντική για πολλαπλούς λόγους:
  - **Αναγνωσιμότητα (Readability):** Ευκολότερη κατανόηση του κώδικα.
  - **Συντήρηση (Maintainability):** Ευκολότερες αλλαγές και debugging.
  - **Επαναχρησιμοποίηση (Reusability):** Εύκολη χρήση των μπλοκ / components σε διαφορετικά σημεία του σχεδίου.
- Η SystemVerilog έχει αρκετούς μηχανισμούς για καλύτερη οργάνωση του κώδικα:
  - **Typedefs**: Δημιουργία προσαρμοσμένων τύπων (user-defined types).
  - **Enums**: Ορισμός απαριθμήσεων (enumerated types).
  - **Structs**: Οργάνωση σχετικών σημάτων / δεδομένων.
  - **Arrays**: Συλλογές από ομοειδή σήματα / δεδομένα (packed & unpacked).
  - **Packages**: Πακέτα για διαμοιρασμό και επαναχρησιμοποίηση δηλώσεων τύπων, συναρτήσεων, κτλ.

SystemVerilog: Προσαρμοσμένοι Τύποι

- Τα **typedef** επιτρέπουν τη δημιουργία νέων ονομάτων τύπων (aliases) για υπάρχοντες τύπους ή τύπους που ορίζονται από τον σχεδιαστή.

- Βελτιώνουν την αναγνωσιμότητα του κώδικα και διευκολύνουν τις αλλαγές ή προσαρμογές των τύπων αργότερα.

```
typedef existing_data_type new_type_name;

typedef logic [7:0] byte_t;
// 'byte_t' is now an alias for 'logic [7:0]'

typedef integer count_t;
// 'count_t' is now an alias for 'integer'

typedef struct {
  logic valid;
  logic [31:0] data;
} data_payload_t;
// 'data_payload_t' is a new struct type
```

Τα πλεονεκτήματα των typedefs

- Βελτιωμένη Αναγνωσιμότητα
  - Χρησιμοποιώντας περιγραφικά ονόματα για τους τύπους, η κατανόηση του κώδικα γίνεται ευκολότερη
- Απλοποίηση Δηλώσεων
  - Αντί να χρησιμοποιούνται σύνθετοι τύποι, ο σχεδιαστής χρησιμοποιεί το όνομα που επιλέγει να ορίσει με το `typedef`
- Ευκολότερες Αλλαγές / Προσαρμογές:
  - Αν χρειαστεί αλλαγή ενός τύπου, η αλλαγή γίνεται μόνο σε ένα σημείο του κώδικα (τον ορισμό του `typedef`) και όχι σε όλα τα σημεία που χρησιμοποιείται
- Επαναχρησιμοποίηση Κώδικα
  - Οι τύποι που ορίζει ο χρήστης μπορούν εύκολα να επαναχρησιμοποιηθούν σε όλο το σχέδιο (με τη χρήση `packages` – αργότερα…)

```
module top;
  typedef logic [7:0] byte_t;
  typedef byte_t address_t;

  byte_t my_data;
  address_t memory_location;

  initial begin
    my_data = 8'hAA;
    memory_location = 8'h10;

    $display("Data: %h at Address: %h",
             my_data, memory_location);
  end
endmodule
```

Απαριθμήσεις – Enumerations

- Το **`enum`** (enumeration / απαρίθμηση) ορίζει έναν τύπο που μπορεί να παίρνει ένα συγκεκριμένο πεδίο τιμών και ορίζει ονόματα για καθεμιά από αυτές τις τιμές.

- Είναι χρήσιμο σε αναπαράσταση καταστάσεων σε FSM, τύπους εντολών (opcodes), ή σε οποιαδήποτε μεταβλητή πρέπει να δεχτεί ένα περιορισμένο αριθμό από πιθανές τιμές.

- Από default, το πρώτο όνομα παίρνει την τιμή 0, το δεύτερο 1, κτλ. Μπορούμε επίσης να ορίσουμε ρητά τις τιμές για κάθε όνομα.

```
typedef enum {
  value1,
  value2,
  value3 // ...
} enum_name_e;


enum_name_e variable_name;
```

Παραδείγματα Enums

```
typedef enum logic [1:0] { // Specifying a 2-bit encoding
  IDLE = 2'b00,
  READ = 2'b01,
  WRITE = 2'b10,
  ERROR = 2'b11
} state_e;

typedef enum {
  ADD,
  SUB,
  MUL,
  DIV
} opcode_e;

module top;
  state_e current_state;
  opcode_e operation;

  initial begin
    current_state = READ;
    operation = MUL;

    $display("Current State: %s", current_state.name()); // Prints "READ"
    $display("Operation Code: %s", operation.name());    // Prints "MUL"
    $display("State Value: %b", current_state);          // Prints "01"
    $display("Opcode Value: %0d", operation);            // Prints "2"
  end
endmodule
```

Δομές – Structs στην SystemVerilog

- Το **`struct`** επιτρέπει τη δημιουργία γκρουπ «σχετικών» μεταβλητών / σημάτων που έχουν διαφορετικούς τύπους, χρησιμοποιώντας ένα όνομα.
- Είναι χρήσιμο για την αναπαράσταση «σχετικών» κομματιών πληροφορίας / δεδομένων, όπως τα πεδία εντολών ή πακέτων ή για παραμετροποίηση των modules

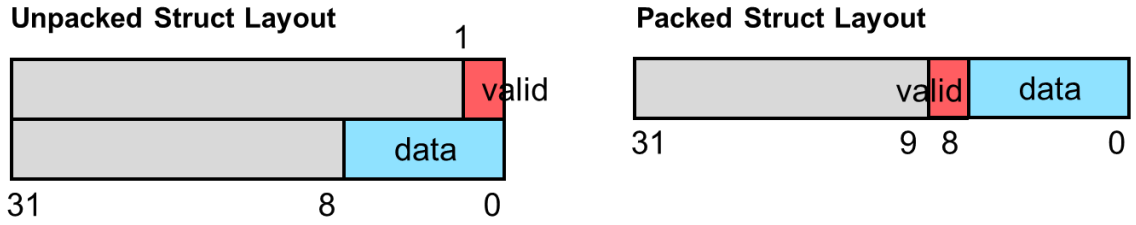
```
typedef struct {
  data_type member1_name;
  data_type member2_name;
  // ...
} struct_name_t; // Optional typedef name

struct_name_t instance_name;
```

Παράδειγμα Struct: Αναπαράσταση Pixel

```
typedef struct {
  logic [7:0] red;
  logic [7:0] green;
  logic [7:0] blue;
} pixel_t;

module top();
pixel_t my_pixel;

initial begin
  my_pixel.red = 8'hFF;
  my_pixel.green = 8'h00;
  my_pixel.blue = 8'h00;
  $display("Red pixel: r=%h g=%h b=%h",
            my_pixel.red, my_pixel.green, my_pixel.blue);
end
endmodule
```

Structs: Packed vs. Unpacked

- Τα struct στην SystemVerilog είναι unpacked από default.
  - Unpacked σημαίνει ότι ο compiler έχει ευελιξία στο πώς θα τοποθετήσει τα πεδία στη μνήμη και πιθανώς να προσθέσει padding για ευθυγράμμιση / alignment
- Μπορούμε να χρησιμοποιήσουμε το `packed` keyword για να ζητήσουμε τα πεδία να τοποθετηθούν συνεχόμενα στη μνήμη και χωρίς padding
- Αυτό χρησιμοποιείται συχνά όταν τα structs πρέπει να απεικονιστούν απευθείας ένα hardware interface ή πρέπει να έχουν ένα συγκεκριμένο memory layout
- **«Υποχρεωτική» χρήση struct packed για συνθέσιμο κώδικα!**




```
// Unpacked Struct Example
typedef struct {
  logic valid;
  logic [7:0] data;
} unpacked_data_t;

unpacked_data_t my_unpacked_data;

// Packed Struct Example
typedef struct packed {
  logic valid;
  logic [7:0] data;
} packed_data_t;

packed_data_t my_packed_data;
```

Structs σε πόρτες των modules

- Η χρήση structs για τις πόρτες των modules βελτιώνει την οργάνωση και την αναγνωσιμότητα των διεπαφών των modules. Ειδικά για γκρουπ από «σχετικά» σήματα
  - Αντί για ξεχωριστές πόρτες, μπορούμε να ομαδοποιούμε τα σήματα σε ένα struct

```
typedef struct packed {
  logic [7:0] address;
  logic [7:0] data;
  logic       read_enable;
  logic       write_enable;
} bus_interface_t;

module memory_controller (
  input  logic         clk,
  input  logic         rst_n,
  input  bus_interface_t req_if,
  output bus_interface_t tgt_if
);
// Internal logic using
// req_if.address, req_if.data, etc.
// also do full assignment between structs
assign tgt_if = req_if;
endmodule
```

```
module top;
logic clk;
logic rst_n;
bus_interface_t request_bus;
bus_interface_t target_bus;

memory_controller dut (
  .clk(clk),
  .rst_n(rst_n),
  .req_if(request_bus),
  .tgt_if(target_bus)
);

initial begin
  clk = 0;
  rst_n = 1;
  request_bus.address = 8'h20;
  request_bus.data = 8'h55;
  request_bus.read_enable = 0;
  request_bus.write_enable = 1;
  #10;
  $display("Target Address: %h, Target Data: %h",
           target_bus.address, target_bus.data);
end

endmodule
```

Πίνακες - Unpacked

- Κάθε πίνακας unpacked είναι μια συλλογή στοιχείων του ιδίου τύπου δεδομένων.
- Κάθε στοιχείο μπορεί να προσπελαστεί με ένα δείκτη (index)

- Σύνταξη:

```
data_type array_name [dimension1] [dimension2] … ;
```

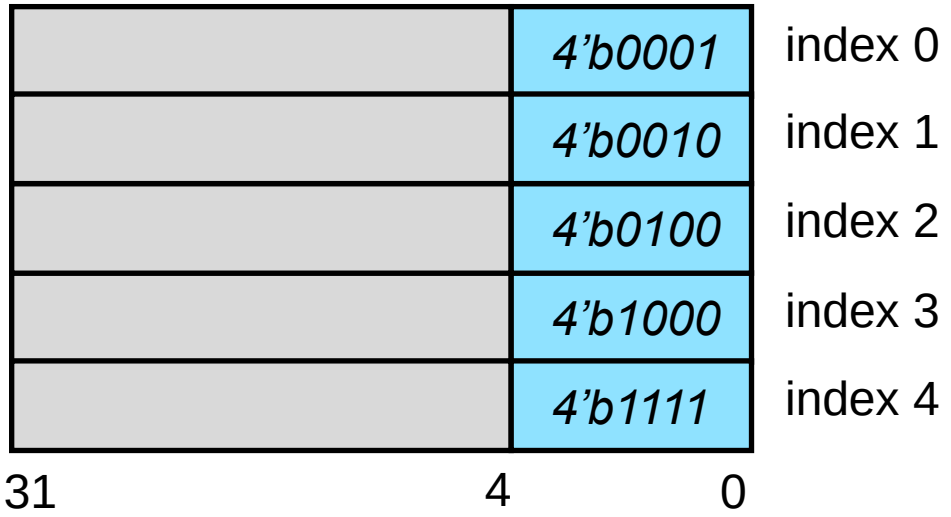
- Χρήση:

```
logic [7:0] byte_array [11:0]; // Array of 12 bytes
integer count [10];            // Array of 10 integers
```

Παράδειγμα Unpacked Array

```
module top;
  logic [3:0] nibble_array [4:0];
  // Array of 5 4-bit values

  initial begin
    // Initialize individual elements
    nibble_array[0] = 4'b0001;
    nibble_array[1] = 4'b0010;
    nibble_array[2] = 4'b0100;
    nibble_array[3] = 4'b1000;
    nibble_array[4] = 4'b1111;

    // Initialize using an assignment pattern
    logic [3:0] another_array [5] = '{4'b1111, 4'b1010, 4'b0101, 4'b1100, 4'b0000};

    $display("nibble_array[2]  = %b", nibble_array[2]);
    $display("another_array[1] = %b", another_array[1]);
  end
endmodule
```

Unpacked Array Layout



Packed Arrays

- Ένας packed πίνακας αναπαριστά μια συνεχόμενη ακολουθία από bits
- Χρησιμοποιείται συχνά για τη μοντελοποίηση σημάτων και μνήμης όπου η σειρά των bits και η «συνέχεια» είναι σημαντικές

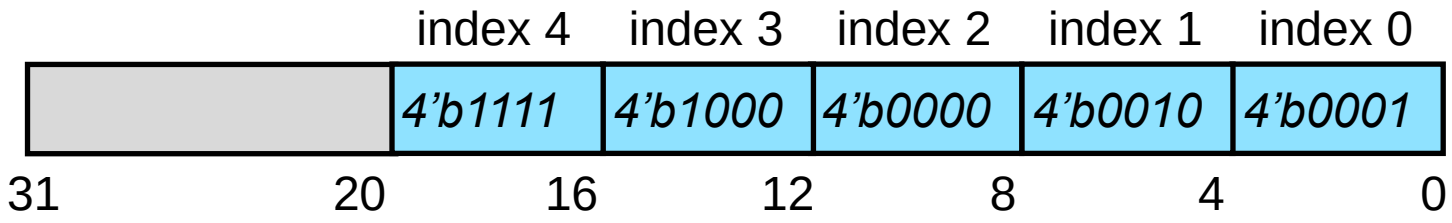
- Σύνταξη:

```
data_type [msb:lsb] [dimension1] [dimension2] ... array_name;
```

Παράδειγμα Packed Array

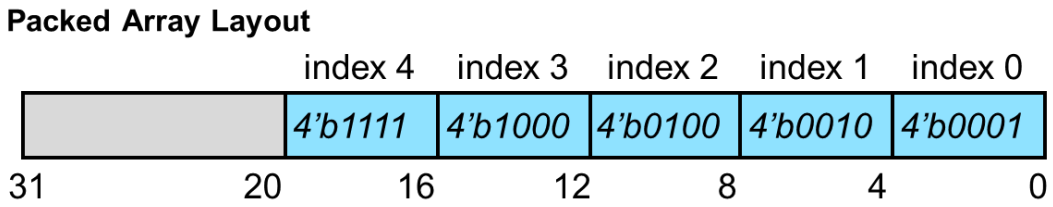
```
module top;
  logic [3:0][4:0] nibble_array; // Array of 5 4-bit values

  initial begin
    nibble_array = '0; // possible to initialize them all together as a flat signal!
    // Initialize individual elements
    nibble_array[0] = 4'b0001;
    nibble_array[1] = 4'b0010;
    // nibble_array[2] = 4'b0100;
    nibble_array[3] = 4'b1000;
    nibble_array[4] = 4'b1111;
    $display("nibble_array[2]  = %b", nibble_array[2]);
  end
endmodule
```

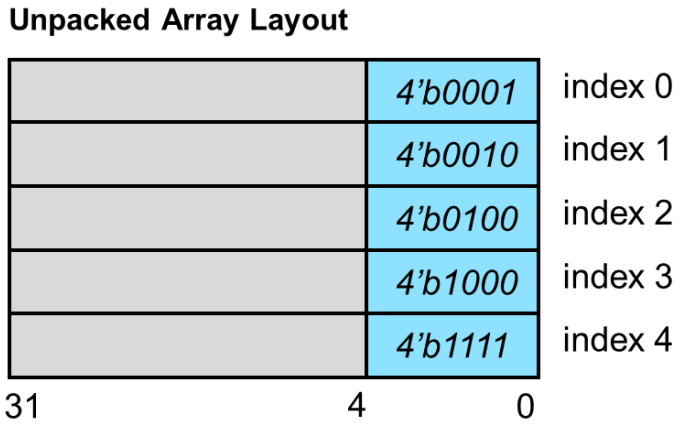
Packed Array Layout



Packed vs Unpacked Arrays

Χαρακτηριστικό	Packed Array	Unpacked Array
Χώρος	Συνεχόμενη ακολουθία από bits	Τα στοιχεία αποθηκεύονται ξεχωριστά στην μνήμη
Δήλωση	**data_type** [packed_dims] **array_name**	**data_type array_name** [dims]
Χρήση	Μοντελοποίηση σημάτων, μνημών, και ευέλικτος χειρισμός των bits	Συλλογή δεδομένων, όπως οι κλασσικοί software πίνακες





Τα Packages της SystemVerilog

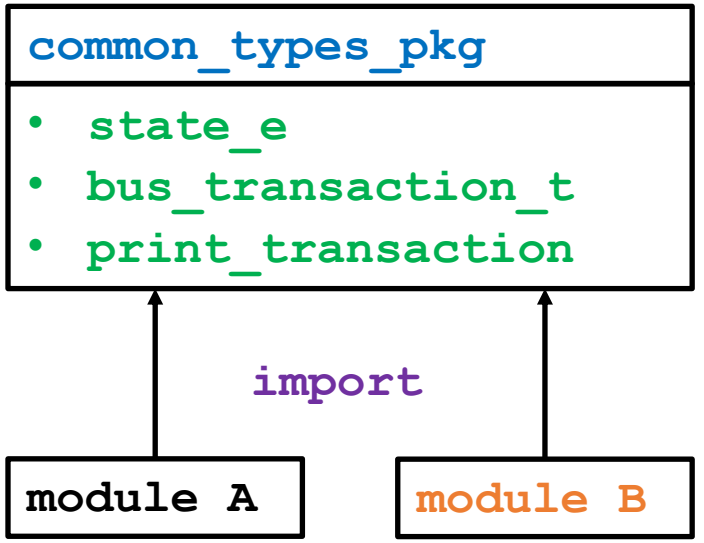
- Τα **`packages`** παρέχουν ένα μηχανισμό να ενθυλακώνουμε (encapsulate) και να διαμοιραζόμαστε δηλώσεις (typedefs, structs, enums, functions, tasks, etc) σε όλη την ιεραρχία ενός σχεδίου.
  - Βοηθάει στην επαναχρησιμοποίηση και την αποφυγή «διπλών» δηλώσεων και «διπλότυπων» ονομάτων

- Σύνταξη:

```
package package_name;
  // Declarations (typedefs, functions, tasks, etc.)
endpackage
```

Παράδειγμα Package: Κοινοί τύποι και συναρτήσεις

```
package common_types_pkg;
  typedef enum logic [1:0] {
    IDLE,
    READ,
    WRITE,
    ERROR
  } state_e;

  typedef struct packed {
    logic [7:0] address;
    logic [7:0] data;
  } bus_transaction_t;

  function void print_transaction(input bus_transaction_t trans);
    $display("Address: %h, Data: %h", trans.address, trans.data);
  endfunction
endpackage
```



Χρήση των Packages

- Για να χρησιμοποιήσετε το περιεχόμενο ενός package σε ένα module πρέπει να το κάνετε **`import`**.
- Σύνταξη:

```
import package_name::*;          // Import all items from the package
import package_name::item_name; // Import a specific item
```

- Χρήση:

```
module my_module;
  import common_types_pkg::*; // Import everything from the package

  state_e current_state;
  bus_transaction_t my_transaction;

  initial begin
    current_state = READ;
    my_transaction.address = 8'h10;
    my_transaction.data = 8'hAA;

    $display("Current state: %s", current_state.name());
    print_transaction(my_transaction); // Calling the function from the package
  end
endmodule
```

Χρήσιμες συναρτήσεις (1/2)

- Πολλές χρήσιμες built-in συναρτήσεις της SystemVerilog (ξεκινούν με $). Κάποιες είναι συνθέσιμες.
- **`$bits(expression)`:**
  - Χρήσιμη για παραμετρικό και ευέλικτο κώδικα RTL
  - Επιστρέφει τον αριθμό των bits μιας έκφρασης
  - Πολύ χρήσιμο για τύπους ορισμένους από το χρήστη (typedefs) ή για τον προσδιορισμό του μεγέθους δυναμικά

```
typedef logic [15:0] word_t;

typedef struct packed {
  logic valid;
  word_t data;
} message_t;

module top;
  word_t my_word;
  message_t my_message;
  integer word_size;
  integer message_size;

  initial begin
    word_size = $bits(my_word);
    // word_size will be 16

    message_size = $bits(my_message);
    // message_size will be 1 + 16 = 17

    $display("Size of my_word: %0d bits", word_size);
    $display("Size of my_message: %0d bits", message_size);
  end
endmodule
```

Χρήσιμες συναρτήσεις (2/2)

- **`$clog2(integer_expr)`**
  - Υπολογίζει το «ταβάνι (ceiling)» του λογάριθμου με βάση 2 για την ακέραια έκφραση

- Χρησιμοποιείται συχνά για να υπολογίσει τον ελάχιστο αριθμό από bits που χρειάζεται για να αναπαρασταθεί ένα συγκεκριμένο εύρος τιμών ή για το «πλάτος» των διευθύνσεων μιας μνήμης με συγκεκριμένο «ύψος/αριθμό λέξεων»

```
module top;
parameter MEMORY_WORDS = 256;
// Example memory size

localparam ADDRESS_BITS = $clog2(MEMORY_WORDS);
// ADDRESS_BITS will be $clog2(256) = 8

initial begin
  $display("Memory Size: %d", MEMORY_SIZE);
  $display("Address Bits: %d", ADDRESS_BITS);
end

endmodule
```

Επισκόπηση

- **Typedefs** επιτρέπουν τη χρήση «συνωνύμων» για υπάρχοντες τύπους ή τύπους ορισμένους από τον χρήστη. Βελτιώνουν την αναγνωσιμότητα και την συντήρηση του κώδικα.
- **Enums** ορίζουν σύνολα από ονοματισμένες τιμές. Χρήσιμα για αναπαράσταση καταστάσεων και επιλογών/options.
- **Structs** επιτρέπουν τη δημιουργία γκρουπ από «σχετικά» σήματα χρησιμοποιώντας ένα μόνο όνομα (packed ή unpacked). Μπορούν να χρησιμοποιηθούν και σε πόρτες των modules για να απλοποιήσουν τις διεπαφές και τον αριθμό των συνδέσεων/πορτών.
- **Unpacked arrays** είναι συλλογές στοιχείων ιδίου τύπου και η προσπέλαση γίνεται με δείκτες/indexes.
- **Packed arrays** αναπαριστούν συνεχόμενες ακολουθίες από bits και είναι χρήσιμα για μοντελοποίηση κάποιων HW components.
- **Packages** δίνουν τη δυνατότητα για «ενθυλάκωση/encapsulation» και παρέχουν ένα τρόπο να διαμοιραστούμε δηλώσεις, ορισμούς, συναρτήσεις. Βοηθούν την οργάνωση των σχεδίων και την επαναχρησιμοποίηση κώδικα.