HARDWARE DESCRIPTION LANGUAGE FOR DIGITAL DESIGN

數位設計硬體描述語言

Tasks and Functions

Materials partly adapted from "Digital System Designs and Practices Using Verilog HDL and FPGAs," M.B. Lin.

NCKU EE
LPHP Lab

OUTLINE

- Verilog Basics
- Structural Modeling
- Dataflow Modeling
- Behavioral Modeling
- Tasks and Functions

TASKS AND FUNCTIONS

TASK DEFINITION AND CALLS

task [automatic] task_identifier(task_port_list); … endtask

```
// port list style
task [automatic] task_identifier;
[declarations] // include arguments
procedural_statement
endtask

// port list declaration style
task [automatic] task_identifier ([argument_declarations]);
[other_declarations]  // exclude arguments
procedural_statement
endtask
```

A TASK EXAMPLE

```
// count the zeros in a byte
module zero_count_task (data, out);
input   [7:0] data;
output reg [3:0] out;
always @(data)
    count_0s_in_byte(data, out);
// task declaration from here
task count_0s_in_byte(input [7:0] data, output reg [3:0] count);
integer i;
begin   // task body
   count = 0;
   for (i = 0; i <= 7; i = i + 1)
         if (data[i] == 0) count= count + 1;
end endtask
endmodule
```

TYPES OF TASKS

- (static) task

  task … endtask

- automatic (reentrant, dynamic) task

  task automatic … endtask

A DYNAMIC TASK EXAMPLE

```
// task definition starts from here
task automatic check_counter;
reg [3:0] count;
// the body of the task
begin
    $display ($realtime,,"At the beginning of task, count = %d", count);
    if (reset) begin
        count = 0;
        $display ($realtime,,"After reset, count = %d", count);
end
endmodule
```

FUNCTION DEFINITION AND CALLS

function [automatic] [signed] [range_of_type] … endfunction

```
// port list style
function [automatic] [signed] [range_or_type] function_identifier;
input_declaration
other_declarations
procedural_statement
endfunction

// port list declaration style
function [automatic] [signed] [range_or_type]
function_identifier (input_declarations);
other_declarations
procedural_statement
endfunction
```

A FUNCTION EXAMPLE

```
// count the zeros in a byte
module zero_count_function (data, out);
input  [7:0] data;
output reg [3:0] out;
always @(data)
    out = count_0s_in_byte(data);
// function declaration from here.
function [3:0] count_0s_in_byte(input [7:0] data);
integer i;
begin
   count_0s_in_byte = 0;
   for (i = 0; i <= 7; i = i + 1)
       if  (data[i] == 0) count_0s_in_byte = count_0s_in_byte + 1;
end
endfunction
endmodule
```

TYPES OF FUNCTIONS

- (static) function

  function … endfunction
- automatic (recursive, dynamic) function

  function automatic … endfunction

AUTOMATIC (RECURSIVE) FUNCTIONS

```
// the use of reentrant function
module factorial(input  [7:0] n, output [15:0] result);
// instantiate the fact function
    assign result = fact(7);
// define fact function
   function automatic [15:0]  fact;
   input [7:0] N;

   // the body of function
       if (N == 1) fact = 1;
       else fact = N * fact(N - 1);
   endfunction
endmodule
```

CONSTANT FUNCTIONS

```
module RAM (addr_bus, data_bus);
parameter RAM_depth = 1024;
input [count_log_b2(RAM_depth)-1:0] addr_bus;
output reg [7:0] data_bus;
// function declaration from here
    function integer count_log_b2(input integer depth);
        begin   // function body
            count_log_b2 = 0;
            while (depth) begin
                count_log_b2 = count_log_b2 + 1;
                depth = depth >> 1;
            end
        end
    endfunction
endmodule
```