Computer Architecture

Week 6: System Verilog Tutorial III

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

TA: Arş. Gör. Uğur Özbalkan

Office: 311

Email: ugur.ozbalkan@fbu.edu.tr

Course Plan

- System Verilog for Synthesis III

Struct and Unions

- Structs are using for grouping related datas to each other and can be derived many objects from this.

Struct and Unions

- Unions, group related objects like structs, but each element in union has a common memory.
- Since the elements in union do not have the separate memory, the value assigned to one element actually updates the value of the other element.

Struct and Unions

```
struct packed {
logic [ 1:0] parity;
logic [63:0] data;
} data_word;
```

```
union packed {

    struct packed {
        logic [31:0] data;
        logic [31:0] address;
    } data_packet;

    struct packed {
        logic [31:0] data;
        logic [31:0] operation;
    } instruction_packet;

} packet_u;
```

Package

- It can contain structures such as classes, structs, functions, parameters; It is a structure that holds together structures that can be used repeatedly in design.

Package

- Used as a library file.

Package

```
package alu_types;

    localparam DELAY = 1;
    typedef logic [31:0] bus32_t;
    typedef logic [63:0] bus64_t;
    typedef enum logic [3:0] {ADD, SUB, ...} opcode_t;

    typedef struct {
    bus32_t i0, i1;
    opcode_t opcode;
    } instr_t;

    function automatic logic parity_gen(input d);
        return ^d;
    endfunction

endpackage
```

Interfaces

- Interface is a structure that allows the signal definitions to be covered in a single structure.
- These grouped signals can be defined in the desired module with a single definition.

System Verilog

```
interface my_int;
    logic sel;
    logic [9:0] data1, data2, result;

    modport b1 (input result, output sel,
data1, data2);
    modport b2 (input sel, data1, data2,
output result);

endinterface : my_int
```

```
module top (
input clk,
input s1,
input [9:0] d1, d2,
output my_sel,
output equal);

logic [9:0] data1, data2, result;
logic sel;

my_int int3();

assign my_sel = int3.sel;

bottom1 u0 (int3, clk, d1, d2, s1, equal);
bottom2 u1 (int3, clk);


endmodule
```

System Verilog

```
module bottom1 (
my_int.b1 int1,
input clk,
input [9:0] d1, d2,
input s1,
output logic equal);

always@(posedge clk) begin
if (d1 == d2)
    equal <= 1;
else
    equal <= 0;
end

always@(posedge clk) begin
     if (s1 == 1) begin
        int1.data1 <= d1;
    end
     else begin
       int1.data2 <= d2;
    end
end

always@(posedge clk) begin
     int1.sel <= ^int1.result;
end

endmodule
```

```
module bottom2 (
my_int.b2 int1,
input clk);

always@(posedge clk) begin
if (int1.sel == 1)
    int1.result <= int1.data1;
else
    int1.result <= int1.data2;
end

endmodule
```