Electronic Circuits

Week 13: Interfaces

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

Interface, birbirleri ile ilişkili sinyalleri gruplamak amacıyla kullanılmaktadır. Tüm ilişkili sinyaller tek bir grupta tanımlanıp, yönetilebilmektedir.

- Manual olarak birçok sinyalin bağlanması gereksinimini ortadan kaldırır.
- Port tanımları her bir dosyada kopyaları oluşmaktadır ve yönetilmesi zordur
- Tüm portlar hakkında bilgi sahibi olunmalıdır.
- Interface kullanımı ile kodun yeniden kullanılabilirliği artmaktadır.

Course Plan

```
interface [name] ([port_list]);
    [list_of_signals]
endinterface
```

Interface Syntax'i

```
interface axis (input axisclk);
    logic [31:0]    axisdata;
    logic           axisready;
    logic           axisvalid;
endinterface
```

AXIS Interface Örneği

Course Plan

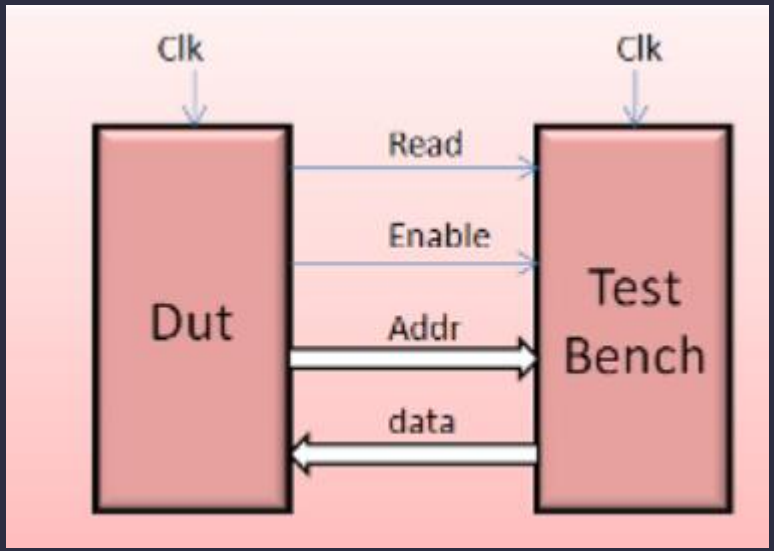
Interface'lerde;

- Function
- Task
- Variable
- Parameter
- Initial ve Always blokları

gibi yapılar bulunabilir.

Course Plan

Interface yaklaşımı ile modülleri birbirine bağlamak için bir örnek verilmektedir.



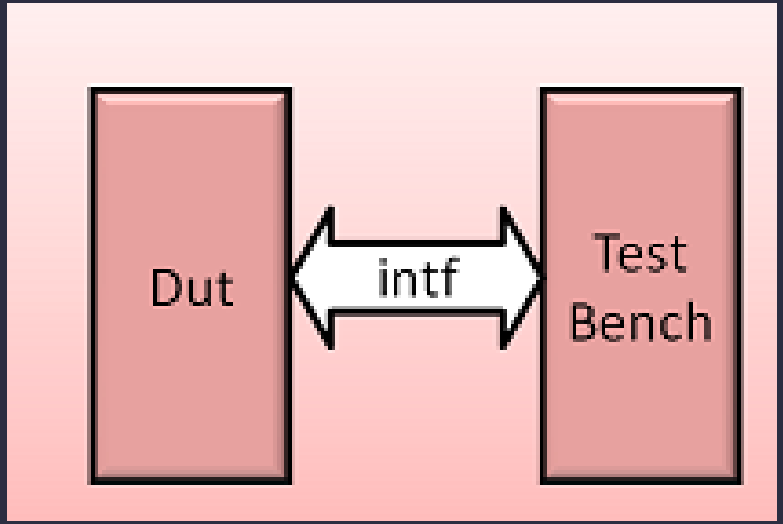
Course Plan

```
interface intf #(parameter BW = 8)(input clk);
    logic read, enable;
    logic [BW -1 :0] addr,data;
endinterface :intf

module Dut (intf dut_if);

    always @(posedge dut_if.clk)
        if(dut_if.read)
            $display("Read is asserted");

endmodule

module tb(intf tb_if);

    initial begin
        tb_if.read = 0;
        repeat(3) #20 tb_if.read = ~tb_if.read;
        $finish;
    end

endmodule
```

```
module tb_top();

bit clk;

initial
forever #5 clk = ~clk;

intf bus_if(clk); // interface instantiation
Dut d(bus_if); // use interface for connecting D and TB
tb tb (bus_if);

endmodule
```

Course Plan

Interface kullanımı ile sinyallerin tek tek tanımlanıp bağlanması yerine, daha önceden tanımlanmış bir sinyal grubunun bağlanması olarak ifade edilmiştir.



Course Plan

```
interface myInterface ();
    reg          gnt;
    reg          ack;
    reg [7:0]    irq;
endinterface

module myDesign ( myInterface dut_if,
                  input logic clk);

    always @(posedge clk)
        if (dut_if.ack)
            dut_if.gnt <= 1;

endmodule
```

Course Plan

```
module tb;
    reg clk = 0;
    always #5 clk = !clk;

    myInterface     if0 ();
    myInterface     wb_if [3:0]();

    myDesign top (if0, clk);

    myDesign md0 (wb_if[0], clk);
    myDesign md1 (wb_if[1], clk);
    myDesign md2 (wb_if[2], clk);
    myDesign md3 (wb_if[3], clk);

endmodule
module tb;
    reg clk;

    myInterface     dut_if();

    myDesign        top (.*);

endmodule
```

Course Plan

Modport, interface bloğunun içinde tanımlanan ve arayüzü kullanacak olan modülün sinyallerinin giriş veya çıkış olacağının tanımlanmasını sağlanabildiği bir bloktur.

```
modport  [identifier]  (
        input  [port_list],
        output [port_list]
);
```

Course Plan

Tablo'da interface bloğunda tanımlanmış olan modport örneği verilmektedir.

```
interface intf (input clk);
    logic read, enable;
    logic [7:0] addr,data;

    modport dut (input clk,
read,enable,addr,output data);
    modport tb (output read,enable,addr,input
clk, data);
endinterface :intf

module Dut (intf.dut dut_if);

    assign dut_if.data = 8'd128;

    always @(posedge dut_if.clk)
        if(dut_if.read)
            $display("Read is asserted");

endmodule
```

```
module tb(intf.tb tb_if);
    initial begin
        tb_if.read = 0;
        repeat(3) #20 tb_if.read = ~tb_if.read;
        $finish;
    end
endmodule
module tb_top();

    bit clk;

    initial
    forever #5 clk = ~clk;

    intf bus_if(clk);
    Dut d(bus_if.dut);
    tb tb(bus_if.tb);

endmodule
```