LINT Your Design with SILOS-X

Introduction

The LINT tool is used to check language errors and potential design issues (race condition, un-synthesizable design, unnecessary components, etc.) in a logic design. Since it can detect potential design issues (which would emerge much further downstream) in the early stages of design, the LINT tool can greatly improve the efficiency of logic design. Turbolint produced by Simucad was such a tool. Turbolint was embedded into the new SILOS-X as a LINT function. Because of this, SILOS-X is not just a simulation tool, but also a powerful tool in helping improve design quality. With examples, this article analyzes how the LINT function does the job.

LINT Capability

Almost every simulation tool can do basic syntax and semantic check. However, passing such a check only shows that a design is grammatically correct, but tells little about the quality of the design. LINT function has many more check rules and checks many more categories than syntax and semantic:

(1), LINT checks possible race condition in a design.

Example:

```
module m(clk, a, z_r);
input clk;
input a;
output z_r;
reg z_r;
reg q_r;
always @(posedge clk) begin
z_r = q_r;
end
always @(posedge clk) begin
q_r = a;
end
endmodule
```

In this example, there are two always blocks, one that reads reg “q_r” and the other that writes the reg. The order of evaluation of the always blocks is not deterministic so the read can occur either before or after the write. The two orders of execution (read-then-write and write-then-read) can yield different results in simulation. Hence, this is a Verilog race condition, which LINT will catch.

(2) LINT performs hardware analysis

LINT gives warnings when registers, latches, state-machines and other sequential elements are inferred. One therefore can check whether extra synchronous hardware was synthesized where not wanted.

Example:

```
module m(SELECT, IN_A, IN_B, X, Y);
input [1:0] SELECT;
input IN_A;
input IN_B;
output X;
output Y;

reg X;
reg Y;

always @(SELECT or IN_A or IN_B) begin
   if (SELECT==2'b00) begin
       X = IN_A | IN_B;
       Y = 0;
   end
   else if(SELECT ==2'b01) begin
       Y = IN_A & IN_B;
   end
   else if (SELECT==2'b10) begin
       X = 0;
       Y = 1;
```

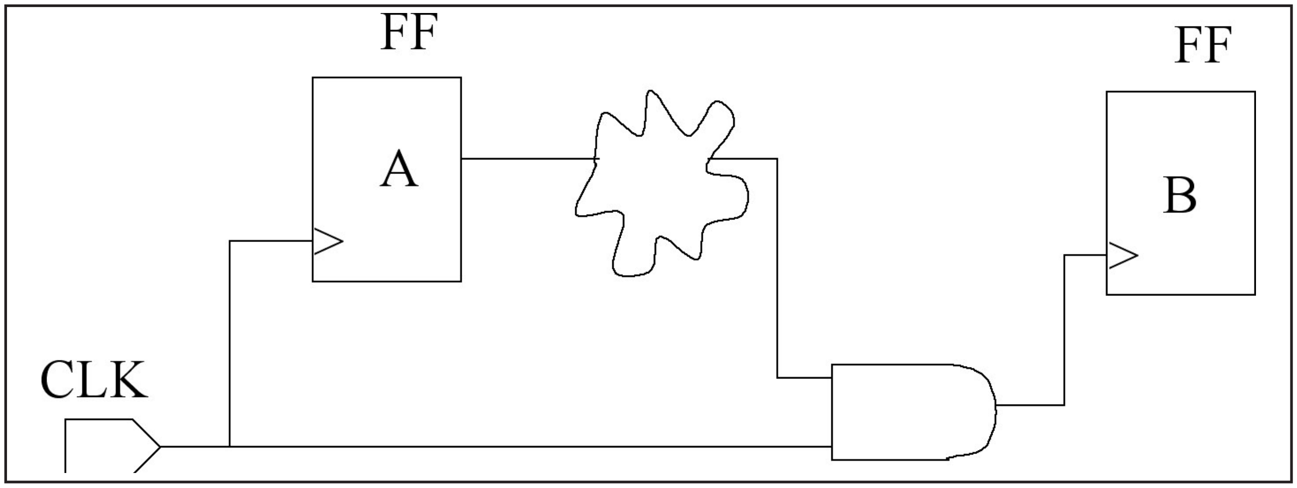

Figure 1.

```
        end
        else begin
            X = 0;
            Y = 0;
        end
    end

    endmodule
```

In this example, assignment expression to X is not defined when SELECT = 2’b01, therefore a latch will be generated to hold the value of X for that case. LINT will remind users of this issue for further checking.

LINT also points out un-synthesizable designs and designs that may cause mismatched simulation results before and after synthesize.

(3), LINT checks for DFT

Design For Test (DFT) is very important for a successful design. It can facilitate fault test and improve the quality of a design. LINT catches suspicious parts in a design that may be difficult or impossible to test.

In this design, since the clock pin of the second FF can not be directly controlled from an external input port, the scan insert tool will exclude this FF from the scan. So it will be difficult to detect faults for this part using the ATPG tool. LINT will point out this issue at a very early stage.

(4), LINT can improve the portability and readability of a design.

LINT has checking rules to help users to write more readable and easily maintained designs.

Example:

```
    output [31: 0]   X;
```

In this example, LINT will give a suggestion to define a constant with the numerical value and use the constant rather than using the hard-coded numerical constant directly. Particularly when such a constant appears in many places, using a defined constant can make maintenance much easier.

Example:

```
    Reg  Abc;
    Reg  abc;
```

In this case, LINT will warn that identity names are distinguished by lower or upper cases, which is error prone and makes the code difficult to read.

Above we listed several categories that LINT checks, but not all. The LINT function in SILOS-X has more than 500 check rules and can also check design compatibility with STARC.

Conclusion

The previous examples show that the LINT function is powerful in improving design quality and efficiency. For inexperienced designers, LINT is especially helpful in teaching them how to write a design in the correct style and how to avoid potential mistakes. For experienced designers, LINT can improve their design efficiency by catching suspicious code in a complicated design and help to improve the quality of the code. So, with LINT, SILOS-X becomes a powerful design tool. All designers should learn to take advantage of this function to improve their design quality.