COMIT SYSTEMS
The Contract Engineering Company
Design Advantage®
Comit Systems, Inc. Newsletter
We build them for YOU.SM
Volume 4 Number 4 Jul - Aug 2001

Verifiable RTL Design - Fully Specified Case Statements

By Harry Foster, Hewlett Packard *and John Emmitt,* Verplex Systems

As discussed in the book *Principles of Verifiable RTL Design* by Lionel Bening and Harry Foster, one of the requirements for producing verifiable designs is fully specified case statements. This approach has verification advantages and design implementation consequences which will be presented here. This article also illustrates an assertion module approach, which aids verification in both simulation and formal methods.

Full specification of case statements requires designers to specify the output state for all 2^N input conditions, where N is the width of the (Verilog®) case expression. Example 1 shows a typical use of the incompletely specified case statement, in conjunction with a synthesis full_case pragma. Example 2 shows the fully specified counterpart for the same case statement in example 1. The incompletely specified case statement creates don't-care conditions which the synthesis tool can take advantage of to optimize the gate-level design. A design error, that allows the unspecified case to occur, means that there can be a functional error in the gate-level implementation.

Example 1 Incompletely specified case

```
module ctr0_2 (in1, out1);
       input [1:0] in1;
       output [1:0] out1;
       reg [1:0] out1;
       always @(in1)
    case (in1) //synthesis_pragma
    full_case
           2'b00: out1 = 2'b01;
           2'b01: out1 = 2'b10;
           2'b10: out1 = 2'b00;
           //2'b11 should never occur
         endcase
  endmodule
```

Example 2 Fully specified case

```
module ctr0_2 (in1, out1);
       input [1:0] in1;
       output [1:0] out1;
       reg [1:0] out1;
       always @(in1)
         case (in1)
          2'b00: out1 = 2'b01;
          2'b01: out1 = 2'b10;
          2'b10: out1 = 2'b00;
          2'b11: out1 = 2'b00;
         endcase
endmodule
```

Here are some of the reasons why using fully-specified case statements is preferred, from a verification standpoint:

- Better performance from Boolean Equivalence Checkers: elimination of don't-cares can boost equivalence checking performance by 5-10x.
- RTL – gate simulation alignment: full specification of cases means that the RTL and gate-level designs will be the same on a cycle-by-cycle and state-by-state basis. This allows the manufacturing test vectors to be run against the RTL model, which provides a performance advantage of 5-10x over the gate-level simulation.
- Better RTL simulation performance: full specification of cases eliminates the need for an X assignment for the default ("impossible") states, which improves RTL simulation performance.

While the verification advantages heavily favor using fully specified case statements, there are design implementation consequences. Two of these are:

- Non-optimized synthesized netlists: fully specified cases with a **default** will result in higher gate counts in the synthesized design. (See example 3).
- Loss of simplicity: full specification of case statements may require the designer to use more (Verilog®) statements, in order to achieve minimal gate counts and optimized timing.

Example 3 Fully specified case with default

```
module one_hot (inhot, out_st);
  input [7:0] inhot;
  output [2:0] out_st;
  reg [2:0] out_st;
  always @(inhot) begin
    case (inhot) //synthesis_pragma full_case
       8'b10000000: out_st =3'b000;
       8'b01000000: out_st =3'b001;
       8'b00100000: out_st =3'b010;
       8'b00010000: out_st =3'b011;
       8'b00001000: out_st =3'b100;
       8'b00000100: out_st =3'b101;
       8'b00000010: out_st =3'b110;
       8'b00000001: out_st =3'b111;
       default: out_st = 3'b000;
     endcase
  end //always
endmodule
```

Examples 4 and 5 shows ways to implement the one_hot module above so that the synthesized design gate count is minimized and the error condition is encapsulated separately.

Example 5 also shows an instantiation of the assert_one_hot module for verification of the one_hot module. The assert_one_hot assertion is usable in both standard RTL simulation and formal verification*, since it is a standard Verilog® module. This assertion is part of the Open Verification Library (OVL) set of assertion modules available as an open source at http://www.verificationlib.org

Example 4 Fully specified one_hot module for minimized gate count

```
module one_hot (inhot, out_st);
     input [7:0] inhot;
     output [2:0] out_st;
     reg [2:0] out_st;
     reg [2:0] code0, code1,code2,
               code3,code4,code5,code6;
     always @(inhot) begin
      code6 = (inhot[6]) ? 3'b001: 3'b000;
      code5 = (inhot[5]) ? 3'b010: 3'b000;
      code4 = (inhot[4]) ? 3'b011: 3'b000;
      code3 = (inhot[3]) ? 3'b100: 3'b000;
      code2 = (inhot[2]) ? 3'b101: 3'b000;
      code1 = (inhot[1]) ? 3'b110: 3'b000;
      code0 = (inhot[0]) ? 3'b111: 3'b000;
      out_st= code0|code1|code2|
              code3|code4|code5|code6;
   end //always
endmodule
```

Example 5 One_hot error case checking using assertion module instantiation

```
   …
   assert_one_hot inhot_one_hot
          #(0,8) (clk, rst_n, inhot);
endmodule
```

* OVL assertions can be formally verified in the BlackTie™ functional checker product from Verplex Systems, Inc

You can learn more about creating verifiable RTL designs and using assertions to verify them by attending a **free tutorial:**

Primary speaker: Harry Foster
- Hewlett Packard, Richardson, TX
co-author of Principles of Verifiable RTL Design

(*FREE* copy of the book to all qualified attendees)

Registration: http://www.verplex.com/rtl
Phone: 408-586-0382
Or send an email with your full name, title, email address, mailing address, phone number and location you wish to attend to info@verplex.com. ■

John Emmitt is Product Marketing Manager at Verplex Systems, Inc. Verplex Systems is a member of the Comit Technology Alliance Program. Opinions expressed in this article are those of the authors.

FREE Harry Foster Tutorials
FREE Book!

Time: 9:00AM – Noon
Locations and dates:

- Dallas, TX	July 23rd	Westin Galleria Hotel
- Chelmsford, MA	July 24th	Radisson Heritage Hotel
- Tigard, OR	July 25th	Embassy Suites
- Sunnyvale, CA	July 26th	Four Points Sheraton
- Irvine, CA	July 27th	Hyatt Regency Hotel