World Class Verilog & SystemVerilog Training

The Sunburst Design - "Where's Waldo" Principle of Verilog Coding

by Cliff Cummings of Sunburst Design - www.sunburst-design.com

I am a big fan of very concise coding. In general (but not always), the shorter the code, the better. The more code I can see, nicely spaced and formatted on one page, the easier it is to understand the intent of the design or verification code.

In my opinion, any extra begin-ends that often consume one or two extra lines of code, and the recently popular "// end-always" style of end-comment tags unnecessarily add clutter to the code and just give your eyes more to scan over before finding and recognizing the important details.

I call this the "Where's Waldo" Principle based on the child puzzle-books of the same name. Even though Waldo is dressed in a bright red and white stripped shirt, when he is surrounded by enough additional clutter, he is hard to find. Just as Waldo is hard to find when surrounded by clutter, simple RTL coding bugs can be obscured when surrounded by poorly spaced and formatted RTL code and silly comments that state the obvious.

I have often debugged RTL code by simply reformatting it and removing the silly comments from the code without ever running a simulation.

World Class Verilog & SystemVerilog Training

The Sunburst Design - "Where's Waldo" Principle of Verilog Coding

by Cliff Cummings of Sunburst Design - www.sunburst-design.com

Example DFF (11 lines of code, 129 characters) - verbose and poor spacing:

```
always @(posedge clk or negedge rst_n)
  begin
    if (!rst_n)
      begin
        q <= 0;
      end // end-if-begin
    else
      begin
        q <= d;
      end  // end-else-begin
  end // end-always-begin
```

The **`always`**-block **`begin`**-**`end`** actually encourages a flawed RTL coding style. If extra code is added after the first **`begin`** statement, synthesis will fail.

Example DFF (3 lines of code, 57 characters) - simple, concise and well-spaced:

```
always @(posedge clk or negedge rst_n)
  if (!rst_n) q <= 0;
  else        q <= d;
```

Note how **`q`**-assignments are placed in a well-spaced column for easy recognition.