CASE STATEMENT ENHANCEMENT PROPOSAL IDEA

THE PROBLEM

Related to unique and priority assertion processing is the problem of making a default assignment before entering unique (or priority) case and then only doing update assignments.

Don & Heath are well aware of the simple 2-to-4 decoder example that we use when teaching synthesis with full_case, priority case or unique case

```
always_comb begin
  y = '0;
  unique case ({en,a})
    3'b100: y[a]='1;
    3'b101: y[a]='1;
    3'b110: y[a]='1;
    3'b111: y[a]='1;
  endcase
end
```

In this example, if you remove the unique keyword, this simulates and synthesizes to a very efficient 2-to-4 decoder, but with the unique keyword, this still simulates like a 2-to-4 decoder and issues a run-time warning whenever en=0 and the always_comb block is executed. It also optimizes away the en-input to give the wrong logic when synthesized.

If the default (y = '0;) assignment could be moved to just inside of the case statement and always executed before testing the other case items, the problem would be solved.

I had looked into putting the "default" statement before the other case items and requiring the default to be executed always before the case items, but this coding style was already legal in Verilog and I believe Steve Sharp mentioned that some users had coded that way expecting default actions to be taken only if another case items did not execute (dang!)

I have tried to think of a solution that does not require another keyword and that would still make coding sense, and I think I have a possible solution. There is also precedent for the solution inside of UDPs.

PROPOSAL

Allow the keyword **initial** to be used inside of a case statement, immediately after the case expression. If used, the initial statement, which could include begin-end code, shall be required to be listed first after the case expression. Placing the initial keyword after any case item shall be a syntax error.

The initial statement would always be executed any time the case statement is executed, then testing of subsequent case items would proceed as normal with only one additional case-item or default match permitted.

The syntax (not semantics) is similar to the initial statement that is allowed inside of a UDP, so the syntax would not be particularly strange.

Possible restrictions?

- Perhaps no LHS delays or RHS blocking delays should be permitted(??)
- No @(edge) or wait statements should be permitted(??)
- Nonblocking assignments SHOULD be permitted
- Nonblocking assignments with RHS delays SHOULD be permitted

Adding initial to a case statement effectively kills any priority assertion testing (because code *WILL* be executed every time the case statement is executed - similar to adding a default-case). Unique testing shall still look for overlap between all case items after the initial statement. Unique shall not fail if both the initial statement and one other case item or default can execute.

Re-examining the above example using the case-initial statement:. Note that there is no warning if enable is 0 (unlike example above) and pre-synthesis & post-synthesis results will be the same (no synthesis mis-match).

```
always_comb begin
  unique case ({en,a})
    initial: y   ='0; // clear all y outputs -
    3'b100:  y[a]='1; // set just one y-output
    3'b101:  y[a]='1; // if enable is true
    3'b110:  y[a]='1;
    3'b111:  y[a]='1;
  endcase
end
```

Another decoder example - before and after initial

```
// Infers latches because only one output is set
// and default is never executed
always_comb begin
  unique case ({en,a})
    3'b100:  y[a]='1; // set just one y-output
    3'b101:  y[a]='1; // if enable is true
    3'b110:  y[a]='1;
    3'b111:  y[a]='1;
    default: y   ='0; // skip if enable is true
  endcase
end

// NO latches because all outputs are cleared each time
// the case statement is executed before one output is set
always_comb begin
  unique case ({en,a})
    initial: y   ='0; // clear all y outputs -
    3'b100:  y[a]='1; // set just one y-output
    3'b101:  y[a]='1; // if enable is true
    3'b110:  y[a]='1;
    3'b111:  y[a]='1;
  endcase
end
```

The follow-on question is, should this capability be added to if-else-if statements??

Cliff tends to say "no" because it is not as clean to make an initial output assignment after the first if-expression and expect it to execute even if the first if-expression is false. Case statements postpone if-testing until the first case item is compared to that which was placed in the case expression (cleaner for this purpose).