FSM Enumerated Types

There is one large problem with enumerated types and state machines and I think the problem also exists in VHDL. For a binary-encoded FSM, enumerated types work fine. For a Onehot-encoded FSM, enumerated types do not work well (same issue applies to the proposed "state" type). Explanation below.

Binary-Encoded

Enumerated types work well because the encoding is assigned to the enumerated state names. Displaying the enumerated names in a waveform viewer is a matter of matching the enumerated name to value on the state bus.

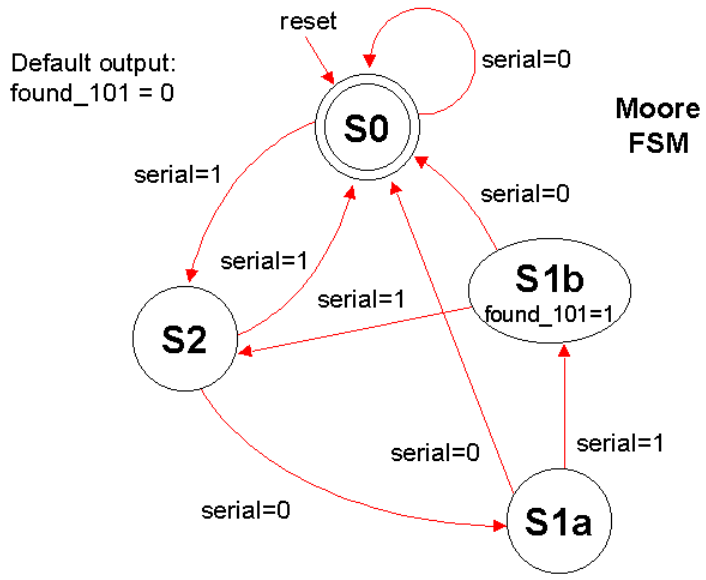
Onehot-Encoded

Although it is possible to encode onehot patterns into state names, this synthesizes to a very inefficient gate-level implementation and largely defeats the purpose of using onehat state encodings. The problem with encoding onehot patterns into state names is that then all combinational comparisons (such as "case (state)") are now multi-bit instead of single bit comparisons. The synthesis tool obliges by inferring multi-bit comparison logic, causing the combinational always block to grow and the design to become slower. This is especially costly when doing FPGA designs since the size of the combinational block, when spread across multiple FPGA logic-blocks, largely determines how fast the design will run.

For this reason, the superior Verilog onehot coding style encodes indexes into the state names, as opposed to using state encodings.

The Verilog code for the Moore state machine below is shown on the next slide. Also note the synthesis-excluded always block at the end of the module that is used to make state names viewable in a simulation waveform window. Showing these statenames automatically does not seem possible using enumerated types, since the names would correspond to an index variable and not to the state encoding itself.

In VHDL, I believe an efficient onehot FSM coding style requires if-else-if comparisons of individual state bits and suffers from the same problem as Verilog, not providing the capability to show the state names in a waveform viewer automatically generated from enumerated types. Seems like it would be useful to have both enumerated types for assigned values and for an index into a variable. There are a number of onehot applications that would benefit from the ability to enumerate and display index information as opposed to variable-value information. I am open to suggestions.



```
module FSM3b (output reg found_101, input serial, clock, reset);

  parameter S0 =0, // state bit 0 (0001), not 4'b0000
            S1a=1, // state bit 1 (0010), not 4'b0001
            S1b=2, // state bit 2 (0100), not 4'b0010
            S2 =3; // state bit 3 (1000), not 4'b0011

  reg [3:0] state, next;

  always @(posedge clock or posedge reset)
    if (reset) begin
                state     <= 4'b0;
                state[S0] <= 1'b1;
    end
    else        state     <= next;

  always @* begin
    next = 0;
    found_101 = 0;
    case (1'b1)
      state[S0] : if ( serial)   next[S2]  = 1'b1;
      state[S2] : if (!serial)   next[S1a] = 1'b1;
                  else           next[S0]  = 1'b1;
      state[S1a]: if ( serial)   next[S1b] = 1'b1;
                  else           next[S0]  = 1'b1;
      state[S1b]: begin
                    found_101 = 1;
                    if ( serial) next[S2]  = 1'b1;
                    else         next[S0]  = 1'b1;
                  end
    endcase
  end

  `ifndef SYNTHESIS // equivalent to translate_off
  always @* begin: display
    reg [3*8:0] statename; // display this ASCII variable in the wave window
    case (1'b1)
      state[S0] : statename="S0"
      state[S1a]: statename="S1a"
      state[S1b]: statename="S1b"
      state[S2] : statename="S2"
    endcase
  end
  `endif
endmodule
```

Index into the `state` and `next` variables, NOT state encodings!

Only 1-bit comparisons, NOT 4-bit comparisons

Notice how each individual state bit is tested to determine which state name to display