

BINARY ENCODING VERSUS ONE-HOT ENCODING NORMAL VERSUS BUFFERED OUTPUTS

Consider the four-state FSM shown. It has no inputs for simplicity since inputs are not the issue here.

We may use binary encoded state variables or one-hot encoding for this machine. We may use regular or buffered outputs. What are the practical differences?

With BINARY ENCODING, two flip-flops are needed and the encoding would likely be such that the encoding corresponded with the state numbers. The equations generated are:

```
bolts = sreg_1.Q * sreg_0.Q
tacks = /sreg_1.Q * sreg_0.Q
nuts = /sreg_1.Q * sreg_0.Q
      + sreg_1.Q * /sreg_0.Q
sreg_1.D = nuts
sreg_0.D = /sreg_0.Q
```

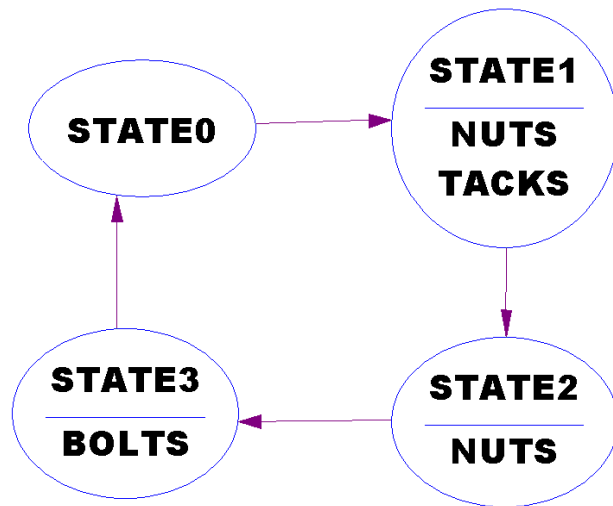
With one-hot encoding, there is one flip-flop (potentially) for each state. The equations obtained are:

```
state0.D = bolts.Q
tacks.D = state0.Q
state2.D = tacks.Q
bolts.D = state2.Q
nuts     = tacks.Q + state2.Q
```

Here, only two of the four state variable appear (state0 and 2). This is because the flip-flops which are used for states 1 and 3 supply TACKS and NUTS respectively and so separate flip-flops are not needed. The total number of macrocells is the same in each case. Furthermore, there is no additional delay in TACKS and BOLTS (but there still is with NUTS). Using Buffered outputs with the one-hot machine we obtain the following:

```
bolts.D = state2.Q ***
tacks.D = state0.Q ***
state0.D = state3.Q
state1.D = state0.Q
state2.D = state1.Q
state3.D = state2.Q
nuts.D   = state0.Q + state1.Q
```

In this case, NUTS is a registered signal and is set when the machine enters a state where it is to



be asserted. There is now no additional delay. Note that we should go in an “alias” BOLTS to State 3 and TACKS to State 1 (as was done before). With the aliasing done, still only 5 macrocells are needed.

Finally, for the sake of completeness, using buffered outputs with the binary encoded state assignment yields:

$$\begin{aligned} \text{bolts. D} &= \text{sreg_1. Q} * \text{/sreg_0. Q} \\ \text{nuts. D} &= \text{/sreg_1. Q} \\ \text{tacks. D} &= \text{/sreg_1. Q} * \text{/sreg_0. Q} \\ \text{sreg_0. D} &= \text{/sreg_0. Q} \\ \text{sreg_1. D} &= \text{sreg_1. Q} * \text{/sreg_0. Q} + \text{/sreg_1. Q} * \text{sreg_0. Q} \end{aligned}$$

The equations for the s register are the same (if you substitute the equivalent of nuts into the first set). The equations for the outputs are not since they are set by the conditions leading to the state where they are to be asserted. Hence there is no additional delay for the outputs.

One may summarize this as follows:

One-hot encoding is generally preferable for machines with many outputs, many of which are asserted in only one state (at least when CPLD's are used, with PAL's the advantage is sometimes not realized). If speed is important, buffered outputs should be used which, in some cases, may require additional macrocells but typically do not.