

Speeding up large FPGA based design using One Hot state machine encoding

The traditional methods like binary encoding used to generate state machine logic result in highly encoded states. State machines with highly encoded state variables typically have a minimum number of flip-flops and wide combinatorial functions.

On the other hand, one hot state machine encoding allows you to create state machine with one flip-flop per state as only one flip flop is "Hot" (active) at one time in this encoding style.

FPGAs have narrow function generator. If we use binary encoding for designing state machines, it results in more combinatorial logic and less number of flip-flops. This can affect the frequency and density of the whole design. By using one hot encoding for state machine implementation, we can decrease the width of combinatorial logic. We can effectively speed up the FPGA design using this approach.

Most FPGA synthesis tools by default uses one hot encoding for better speed. But we can make synthesis tool infer one hot encoded state machine by adopting particular style while encoding state machine. For reference part of complete RTL code is shown below to show the typical style of coding one hot encoded state machine. However there can be other ways possible.

```
module one_hot_encoding (CLOCK, RESET, A, B, C, D, E,  
    MULTI, CONTIG);
```

```
input CLOCK, RESET;  
input A, B, C, D, E;  
output MULTI, CONTIG;  
reg MULTI, CONTIG;
```

```
//One hot encoding
```

```
parameter [6:0]  
    S1 = 7'b0000001,  
    S2 = 7'b0000010,  
    S3 = 7'b0000100,  
    S4 = 7'b0001000,  
    S5 = 7'b0010000,  
    S6 = 7'b0100000,  
    S7 = 7'b1000000;
```

```
// Declare current state and next state variables
```

```
reg [6:0] CS;  
reg [6:0] NS;
```

```
always @ (posedge CLOCK or posedge RESET)
```

```
begin  
    if (RESET == 1'b1)  
        CS <= S1;  
    else  
        CS <= NS;
```

```
end
```

```
always @ (CS or A or B or C or D or D or E)
```

```
begin  
    case (CS)  
        S1 :  
            begin  
                MULTI = 1'b0;  
                CONTIG = 1'b0;  
                if (A && ~B && C)  
                    NS = S2;  
                else  
                    NS = S1;
```

```
end
endcase
end
endmodule
```

One hot encoded state machine requires more area than other state machine encoding styles, as we need more number of flip-flops. This fact doesn't affect the performance much as FPGAs have more flip-flops than function generators. Synthesizing large FPGA based state machines with one hot encoding style yields better performance in terms of speed.