One Hot encoding for FSMs

Computer Architecture

November 17, 2019

© 2019 Oprițoiu Flavius. All Rights Reserved.
Introduction

Objectives:

- Construct control units using the One Hot encoding

For reading:


The **One Hot encoding** for a Finite State Machine (FSM) with \( n \) states uses \( n \) storage elements. Each storage element is associated with one state. In consequence, at any given moment, only one of the \( n \) storage elements has active output.

The FSM implemented using a One Hot encoding uses a large number of storage elements, however its design and debug is straightforward.
Divide-by-$n$ counter

A divide-by-$n$ counter divides the input clock frequency by the given factor $n$ and it is constructed around a modulo-$n$ counter.

A modulo-$n$ counter is composed of:

- an $r$-bit register ($r = \lceil \log_2 n \rceil$)
- a component realizing addition of 1 to register’s content
- the synchronous signal for clearing the register when it reaches content of $n - 1$

The divide-by-$n$ counter has two synchronous inputs: $c_{up}$, for activating advancement in the counting sequence and $clr$, for clearing its content.

The counter has a 1-bit output $dclk$ (divided clock), activated once every $n$ clock cycles.
Transition diagram of a divide-by-5 counter
Divide-by-5 counter design using the One Hot encoding

The design uses 5 D-type flip-flops, \( FF_0, FF_1, FF_2, FF_3 \) and \( FF_4 \), with inputs \( D_i \) and outputs \( Q_i \), associated with the counter’s 5 states \( S_0, S_1, S_2, S_3 \) and \( S_4 \).

At any moment only one of the 5 flip-flop is active (having the output, \( Q_i \), equal to 1). The present state is indicated by the flip-flop with the active output. If \( Q_1 \) is active than the current state is \( S_1 \), if \( Q_4 \) is 1 then \( S_4 \) is the current state, and so on.

On inputs \( D_i \) are connected the boolean equations activating state \( S_i \). FSM’s implementation reduces to writing these equations.

Example: Starting from transition diagram the next state is \( S_1 \) if:

- the present state is \( S_0 \), \( c_{\text{up}} \) is 1, and \( clr \) is 0, or
- the present state is \( S_1 \) and neither \( c_{\text{up}} \) nor \( clr \) are active

Thus \( D_1 = Q_0 \cdot c_{\text{up}} \cdot \overline{clr} + Q_1 \cdot (\overline{clr} \cdot \overline{c_{\text{up}}}) \)

The remaining 4 inputs are constructed similarly.
module d5cntr(
input clk,
input rst_b, //async
input clr,
input c_up,
output dclk
);

//vector of inputs D for all 5 flip flops (FFs); din[0] is input to FF0;
//FF0 indicates whether the current state is S0 or not
reg [4:0] din;
//next value for all 5 FFs; corresponds to the next state
wire [4:0] din_nxt;

//boolean equation setting the next state to S1:
// D1 = Q0 * c_up * (~clr) + Q1 * (~c_up) * (~clr)
assign din_nxt[1] = (din[0] & c_up & (~clr)) | (din[1] & (~c_up) & (~clr));

//boolean equations for activating states S0, S2, S3, S4
//...

//output activation
assign dclk = din[0]; //dclk activated once every 5 clock cycles;
//one can choose any state to activate the output

//update the current states vector
always @ (posedge clk, negedge rst_b)
    if (!rst_b) din <= 5’d1; //set current state to S0 by
    //setting FF0 and clearing all other FFs
    else din <= din_nxt;
endmodule
References