# W7. Control Unit Construction

# One Hot Encoding for the implementation of FSMs

## 1. Laboratory objective

*Implementation of control units using One Hot Encoding*

## 2. Theoretical Background

### 2.1 One Hot encoding

***One Hot Encoding*** for an FSM having ***n*** states uses ***n*** storage elements. Each storage item is associated with a state. Consequently, at each time point, only one of the n storage elements has the output active. The FSM implemented using One Hot encoding uses an increased number of storage elements, but benefits from a direct implementation and easy troubleshooting.

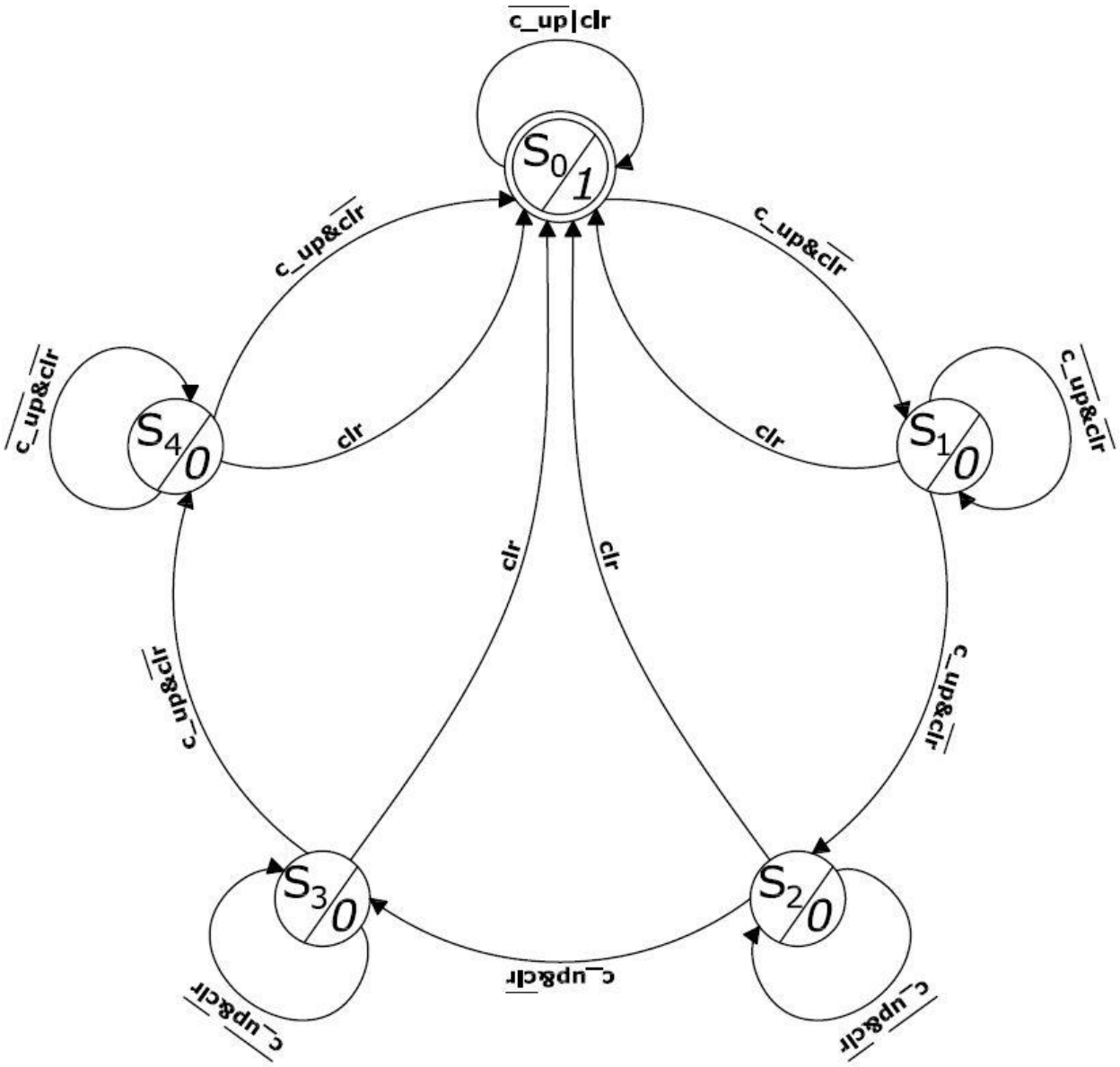
### 2.2.1 Divide-by-N Counter

A divide-by-***n*** counter divides the frequency of the input clock signal by the factor ***n*** and is built around a modulo-***n*** counter. A modulo-***n*** counter is composed of:

- A register on ***r*** bits ( $r = [log_2 n]$ )
- A component for incrementing the contents of the register
- A synchronous ***clear*** signal activates when its contents reach ***n-1***.

The divide-by-***n*** counter has 2 synchronous inputs: ***c_up***, to activate the counting sequence and ***clr***, to delete the register's content. Also, the divide-by-***n*** counter has the output on 1-bit ***dclk*** (divided clock), activated once every n clock cycles.

### 2.2.2 State transition diagram: divide-by-5 counter




The design uses 5 flip-flops type ***D***, ***FF0, FF1, FF2, FF3, and FF4***, with the inputs ***D***i and the outputs ***Q***i, associated with the 5 states ***S0, S1, S2, S3, and S4*** (as illustrated in the diagram above). At any time only one of the 5 flip-flops is active, with the ***Q***i output equal to 1. The current status is indicated by the flip-flop with the active output. If ***Q***1 is active the current state is ***S1***, if ***Q4*** is 1 then ***S4*** is the current state, and so on. The boolean equations that activate the ***Si*** states are connected to the **Di** inputs. The design of the ***FSM*** unit is reduced to writing these equations. Example: From the transition diagram, the next state is ***S1*** if the current state is ***S0***, ***c_up*** is 1 and ***clr*** is 0, or the current state is ***S1*** and neither ***c_up*** nor ***clr*** is 1. So we can write accordingly:

$$D_1 = Q_0 \cdot c\_up \cdot \overline{clr} \; + \; Q_1 \cdot (\overline{clr} \cdot \overline{c\_up})$$

The other 4 D inputs are built similarly.

# 3. Divide-by-5 counter: One Hot encoding and Verilog code

## Laboratory Task

Complete the template below with the missing Boolean equations deduced from the state transition diagram on page 2:

```
1   module d5cntr(
2     input clk,
3     input rst_b,  //asynch
4     input clr,
5     input c_up,
6     output dclk
7   );

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

15    //boolean equation setting the next state to S1:
16    //     D1         =    Q0   * c_up * (~clr)  +    Q1   * (~c_up) * (~clr)
17    assign din_nxt[1] = (din[0] & c_up & (~clr)) | (din[1] & (~c_up) & (~clr));
18    //boolean equations for activating states S0, S2, S3, S4
19    //...

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

25    //update the current states vector
26    always @ (posedge clk, negedge rst_b)
27      if (! rst_b) din <= 5'd1; //set current state to S0 by
28      //setting FF0 and clearing all other FFs
29      else din <= din_nxt;
30  endmodule
```

# 4. References

**[Vlad12]** M. Vlăduțiu, "*Computer Arithmetic: Algorithms and Hardware Implementations*" Springer, 2012.