



COMPUTER ENGINEERING PROGRAM

California Polytechnic State University

©Copyright: 2007 by Bryan Mealy



CPE 269 Experiment 2



Implementing FSMs Using the Equation Method

Objectives:

- To learn the Equation Method to implement FSMs
- To learn the one-hot encoding method of representing states
- To gain further practice with the VHDL structural modeling approach to circuit implementation

Somewhat Meaningful Comments: This implements a FSM using the so-called equation method. Although implementing FSMs using VHDL behavioral modeling has proved useful and straight-forward, it is never-the-less instructive to show that FSMs can also be implemented using flips-flops, excitations equations, and other basic low-level FSM items.

The Equation Method is one of the various methods used to implement FSMs. The Equation Method utilizes the implied storage approach to implement the flip-flops required for state variable storage in FSMs. This method essentially requires two modifications or substitutions to the VHDL code for a standard D flip-flop. The first modification involves the naming of the state variables while the second involves the assignment of asynchronous inputs. A simple listing of the excitation equations for each of the flip-flops in a FSM design effectively generates the required flip-flops. The equations can be generated by any one of a number of different techniques including the *Set or Hold 1* method. Figure 1 shows an example of this approach using two D flip-flops (two state variables). You can extrapolate this method for any number of state variables.

Note that in Figure 1 two intermediate signals have been used as temporary storage for the state variables Y1 and Y2. This approach needs to be taken with working with excitation equations because the mode of the Y1 and Y2 signals has been declared as “out” in the entity declaration. Since these variables were declared as “outs”, they cannot appear on the right side of the signal assignment operator. It is expected that when working with excitation equations that the state variables will need to appear on the right side of the signal assignment operator because the next state of the FSM is generally a function of the present state. Use of the intermediate signals inside the process and subsequent assignment of these variables to the Y1 and Y2 state variables outside the process allows for the next state to be a function of the present state. Since the process statement and the two signal assignment statements outside of the process are all concurrent signal assignment statements, the FSM operates as expected. Note that the model shown in Figure 1 looks a lot like a D flip-flop (there is an if statement with no else statement).

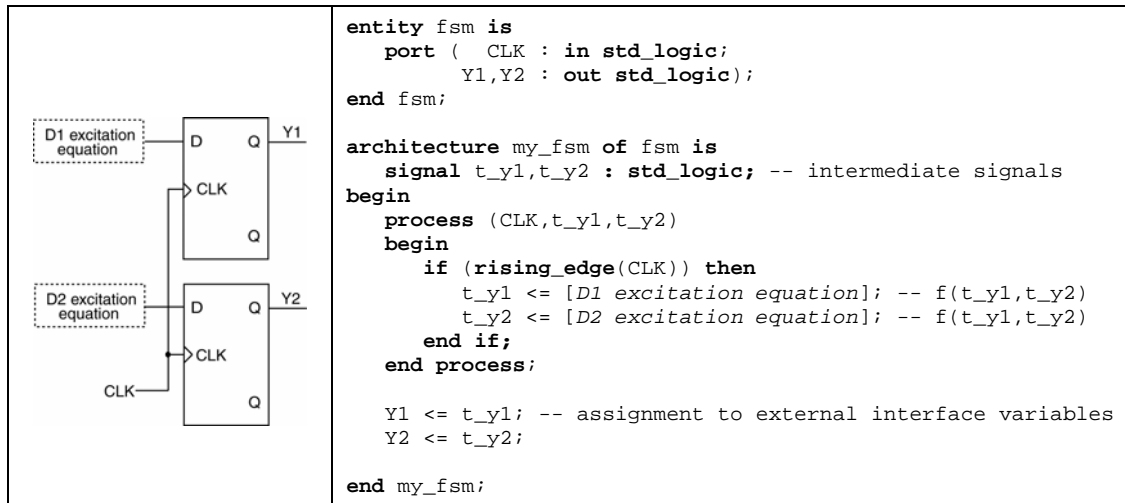


Figure 1: Representing multiple D flip-flops using VHDL.

Assignment: There are two major steps in the following procedure: 1) design a FSM, and 2) place your design in a circuit for visual testing. As shown in Figure 2, the circuit used for testing the FSM consists of three subcircuits which are connected using VHDL structural modeling. One of the subcircuits is the FSM. The other two subcircuits consist of a clock divider circuit (to effectively reduce the frequency of the system clock provided on the Nexys board) and a seven segment display driver circuit (to display the state variables of the FSM). The VHDL code for the previous two subcircuits is provided and it is your job to use VHDL structural modeling in order to interface with these modules. Design a two-bit up/down counter that adheres to the constraints listed below:

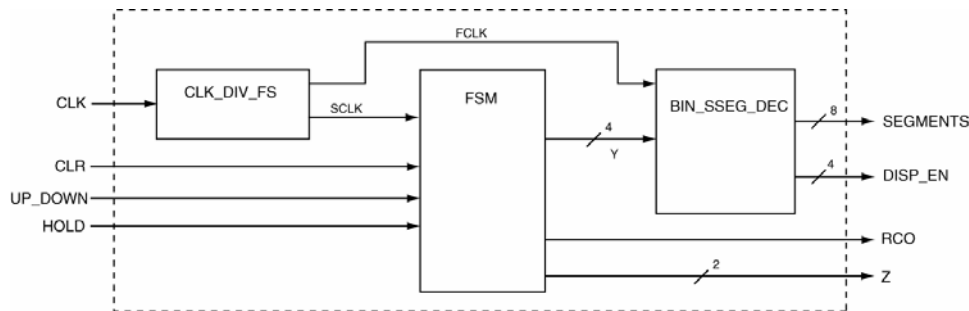


Figure 2: The FSM in a system-level implementation.

1. Design a 4-state counter (FSM) using the Equation Method according to the specifications listed below. Represent the state variables for your design using one-hot encoding. The **CLR** signal causes an asynchronous state transition while all other state transitions are synchronized with the system clock (**CLK**). Start this step by drawing a state diagram for your FSM.

State Variable Designation:

- Use the following state variables in your design: (**Y3 Y2 Y1 Y0**). The corresponding one-hot states should then be: S0=0001, S1=0010, S2=0100, and S3=1000.

Control Inputs:

- The following control inputs listed in order of precedence are:
 - i. **CLR**: when asserted (asynchronous), the counter is initialized to state S0.

- ii. **HOLD**: when asserted, the counter does not change from its present state with the system clock.
- iii. **UP_DOWN**: when asserted, the counter counts up ($S0 \rightarrow S1 \rightarrow S2 \dots$ etc.); otherwise the counter sequences in the opposite direction.

Outputs:

- The state variables (**Y3 Y2 Y1 Y0**) are provided as outputs of the FSM.
- **RCO**: indicates when the FSM is in state S3.
- The outputs **Z1 Z0** are provided an external binary counting sequence and are related to the state variables as shown in Table 1.

State	Output Variables ("Z1 Z0")
S0	"00"
S1	"01"
S2	"10"
S3	"11"

Table 1: Output assignments for the state variables.

2. Simulate your FSM design with ModelSim to verify the FSMs functionality. Include a printout showing your test waveforms and provided annotations showing the important signal transitions. Be sure to test each of the input signals in your design.
3. Using the FSM designed in the previous steps, implement the circuit shown in Figure 2 using VHDL structural modeling. Be sure to comment this VHDL source code and include it with your final circuit. The VHDL descriptions of the CLK_DIV and BIN_SSEG_DEC modules are found in files *bin_sseg_dec.vhd* and *clk_div_fs.vhd*. These files can be downloaded from the class website. **Do not** include printouts of this provided code with your final report.
4. Implement your design on the Nexys board using the I/O assignments shown in Table 2.

	Inputs				Outputs				
CKT Signal	CLR	UP_DOWN	HOLD	CLK	SEGMENTS	DISP_EN	RCO	Z1	Z0
Nexys Signal	BTN3	BTN2	BTN1	CLK	CA,CB,CC,CD CE,CF,CG	AN1, AN2 AN3, AN4	LD0	LD7	LD6

Table 2: I/O assignments for the Nexys development board.

Deliverables:

1. Demonstrate a working design to the lab instructor or Teaching Assistant.
2. A standard lab report. See description of lab report for details.

Questions:

1. This experiment used a special clock divider circuit to reduce the 50MHz system clock frequency. This circuit provides two outputs, *fast_clk* and the *slow_clk*, that represent the system clock frequency divided by two different dividers. Examine the VHDL code calculate the frequencies of the *flck* and *sclk* signals. Provide all work for these calculations with lab report.

2. In this experiment, would it have been possible to simulate the entire circuit using the ModelSim simulator? Briefly explain why or why not. (Hint: it has to do with the clock divider circuit described in the previous question)
3. Generate a formula that represents the number of unused bit combinations as a function of the number of states for a one-hot encoded FSM. Assume the number of states is greater than or equal to two.
4. Because the circuit used in this experiment was one hot encoded, an initialization type pulse was required to put the state variable set into a known one-hot state. Speculate on how the FSM would have acted if this initialization was not done?
5. Although the generation of excitation equations worked fine for this experiment, this technique has one significant drawback. What was it?