Lab 5: A serial two's complementer using ASM Chart and Datapath/Control Unit Design Methodology
Winter 2019 COEN 313: Digital Systems Design II
Department of Electrical and Computer Engineering
Concordia University

Objectives:

• To become familiar with the ASM chart and control unit/datapath design methodology.
• To become well-versed in VHDL coding of datapath components and finite state machines at the RTL level.

A serial two's complementer design:

Consider the following algorithm for generating the two's complement of an binary number: starting from the least significant bit position, retain all the 0's and the first 1, after which complement each of the remaining bits.

Figure 1 shows an architecture for a two's complementer based on this algorithm. It consists of an 8-bit shift register with parallel load capability, a counter, and a control unit. The control unit generates the 7 required control signals : **D**, **Shift**, **Load**, **Inc**, **Clr**, **ld_done**, and **clr_done**.

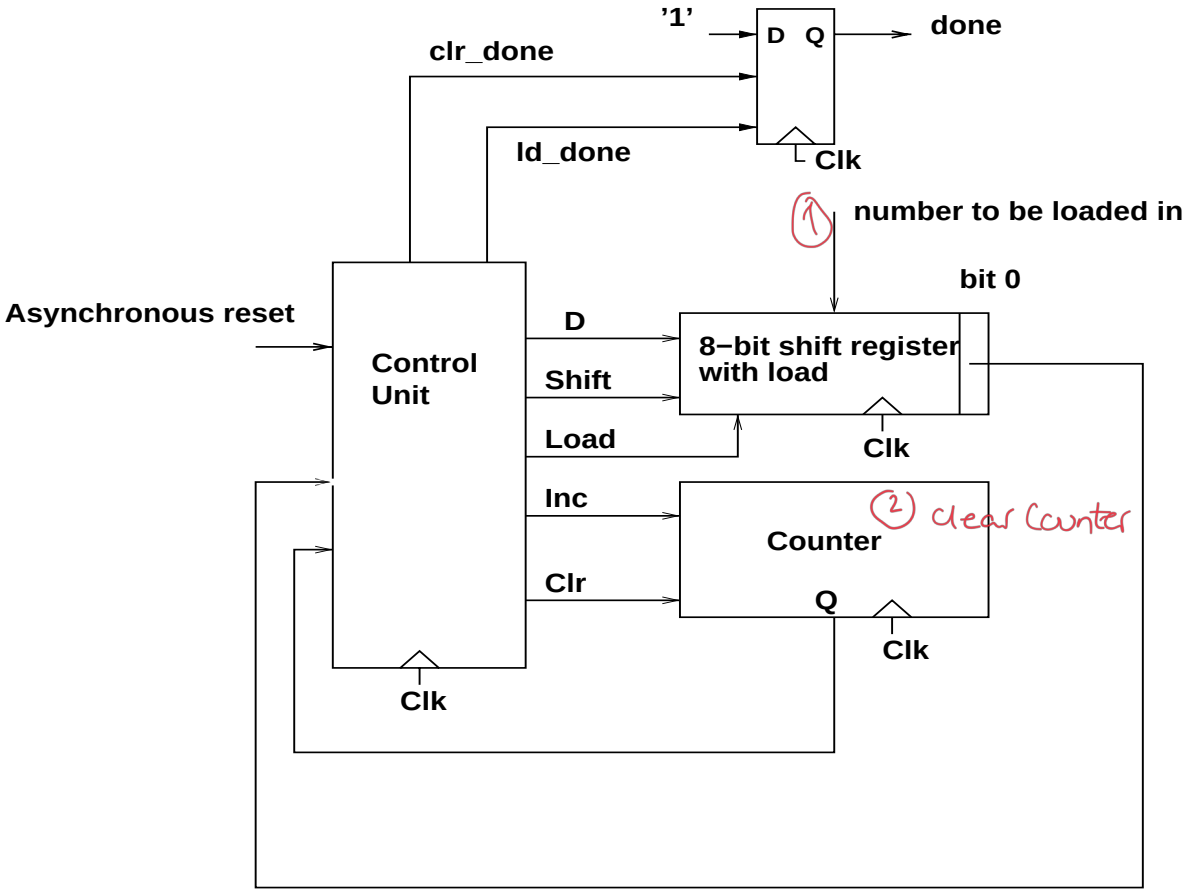

Figure 1: Serial two's complementer block diagram.

The algorithm begins by first loading a number into the 8-bit shift register and clearing the counter and done flip-flop to 0. The least significant bit of the shift register is examined, and based upon its value the control unit then sets the D signal to the appropriate value and shifts it into the most significant position of the shift register. This process is repeated until all 8 bits of the original number have been examined. At this point, the shift register will contain the two's complement of the original number. A "Done" flip-flop is used to indicate that the algorithm is complete.

You are to design an ASM chart (Moore style) which implements this algorithm using the given datapath.

Implement the datapath and control unit using VHDL processes. Use the following entity specification:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity twos_complementer is
port(din : in std_logic_vector(7 downto 0);
     reset : in std_logic ;
     clk   : in std_logic ;
     done_out : out std_logic ;
     counter_out :  out std_logic_vector(3 downto 0);
     reg_out  : out std_logic_vector(7 downto 0));
end twos_complementer ;
```

Use VHDL processes to describe the datapath and the interconnections (using internal signals) between the various components in the datapath. Use a Moore style of finite state machine to implement the control unit. Note that the control signals will ultimately be the outputs of this Moore style state machine.

Your VHDL code should consist of only processes (with the possible exception of using concurrent signal assignment statements to perform negation of internal signals to drive the active-low LED outputs).

Simulate your VHDL code for several different values of inputs to verify that your design works. When writing the Modelsim DO file, pay attention to the timing of when the main asynchronous reset signal is asserted and then deasserted prior to the state machine entering the initial beginning state. As an aid in understanding the algorithm, Figure 2 shows the simulation results for the test input `din` = "00000101". Note the assertion of the asynchronous reset input at the beginning of the simulation. Note also that many clock cycles are required before the system enters the `"finished"` state with the done flip-flop becoming '1' indicating that the algorithm is complete. Depending on how you implemented your ASM chart, your design may require a different number of clock cycles.

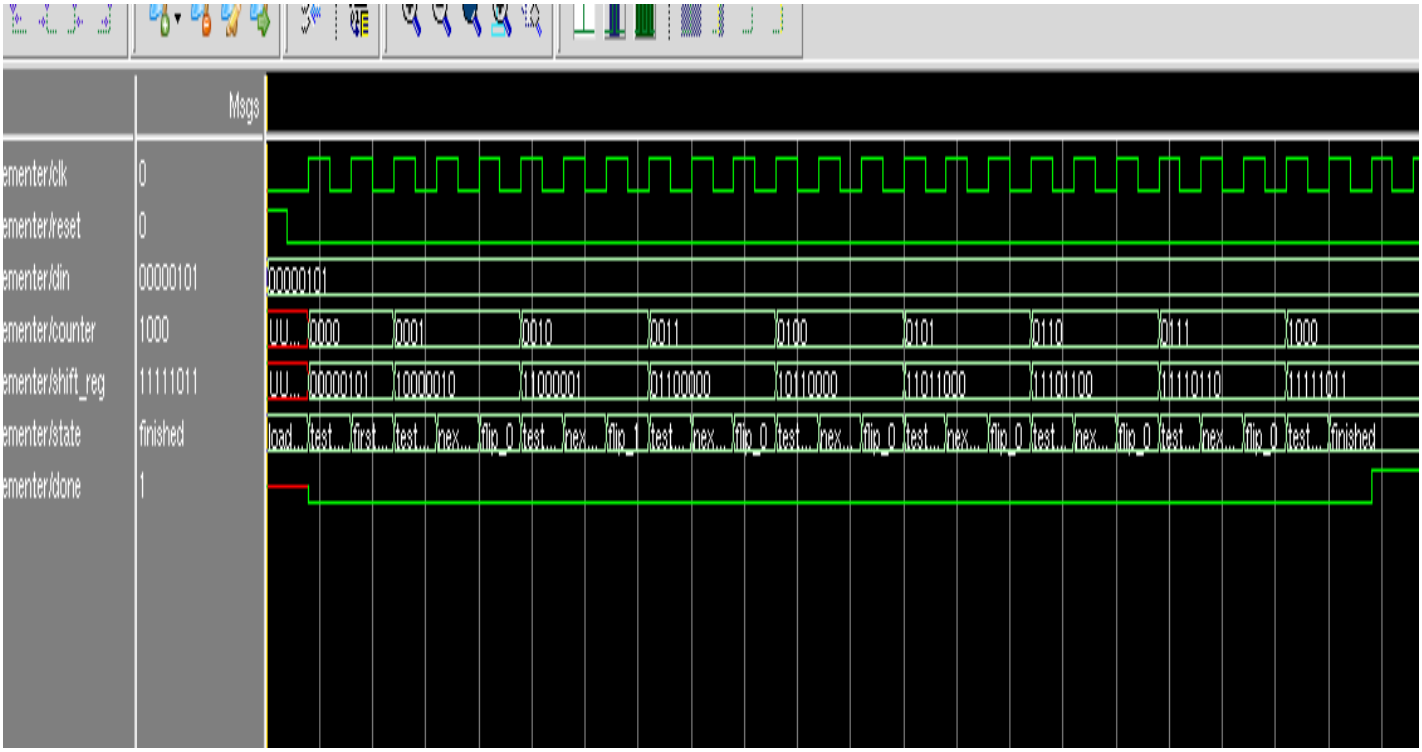

Figure 2: Typical Modelsim simulation results.

Synthesize your design with Precision RTL and implement your synthesized design with Xilinx ISE and Impact. **You are not required to demonstrate your downloaded design to the lab TA as this is the last lab of the session and there is no subsequent lab session two weeks henceforth.** Although you are not required to demonstrate the working design, you are **required to submit** as part of the written lab report the following:

• the `_impact.log` file created by the Xilinx Impact software.

• the `precision.log` file created by the Precision RTL logic synthesis software

• A listing of the directory contents containing the System ACE file generated by the Xilinx Impact software for Lab 5. For example:

```
ted@deadflowers rev0 1:37pm >pwd
/nfs/home/t/ted/SYNOPSYS_2000/Xilinx/Winter_2019/twos_complementer/ACE/rev0
ted@deadflowers rev0 1:37pm >ls -al
total 2856
drwx------ 2 ted ted    4096 Mar 19 19:57 .
drwx------ 3 ted ted    4096 Mar 19 19:57 ..
-rw------- 1 ted ted 1449758 Mar 19 19:57 lab5.ace
```

```
-rw------- 1 ted ted 1449758 Mar 19 19:57 rev0.ace
```

Report Submission:

Please note that the due date for Lab 5 is **Friday, April 12, 2019** before 5:00PM. Submit a **hardcopy printout** of your lab report to the mailbox of T. Obuchowicz in room EV5.139. Please ask the receptionist at the front desk in EV5.139 to place it in my mailbox. **CLEARLY INDICATE ON THE FRONT COVER OF YOUR LAB REPORT YOUR LAB SECTION**.

Requirements:

Submit the following together with your lab report:

• The complete ASM chart.

• The VHDL code for the complete system.

• Simulation results for several input values as produced by Modelsim. Trace through the sequence of states your control unit progresses through and show the control signals produced by the control unit during each clock cycle as well as the values of the data contained in the datapath.

Questions:

1. Examine the `precision.log` file contained in the implementation directory created by Precision RTL. Your design contains a finite state machine. Comment on the state encoding used by the synthesis tool.

2. Comment on the advantages/disadvantages of a binary state encoding vs. a one-hot state encoding. Do other state encoding techniques exist?

3. Consider a datapath which contains a register which has a control signal called “LOAD_MICK” which acts as a synchronous load control signal for the register . Explain by means of a timing diagram and in words the error contained in the following ASM chart (Figure 3) :

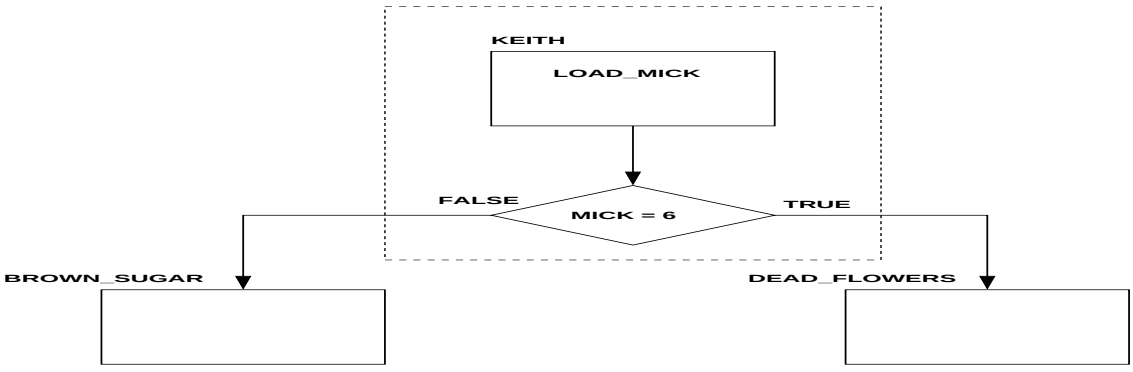

Figure 3: An ASM chart containing an error.

Sources of information:

Chapter 10 of *"RTL Hardware Design Using VHDL"* by Pong P. Chu contains a complete discussion of finite state machines, their implementation with VHDL, state encoding, Mealy and Moore style machines. Chapter 11 provides an overview of the ASM chart design methodology. Chapter 8 of *"Digital Design, 4th ed."* by Mano/ Ciletti contains an easily read overview of register transfer level design with ASM chart. Either text can be referred should you require supplemental information to what has been presented in the classroom lectures.

"It's only VHDL, but I like it, like it , yes I do."

T. Obuchowicz
March 21, 2019