EE362 Introduction to Computer Architecture
Lab 1
VHDL Revisited

CAPT Charles B. Cameron

1 Introduction

The purpose of this lab is to reintroduce the student to the use of VHDL for hardware design. For complex digital circuits, design by traditional methods and the use of schematic capture is too cumbersome. VHDL—the VHSIC Hardware Definition Language[1]—is a widely used example of a hardware definition language, one defined as IEEE standard IEEE Std 1076-1993.

2 Equipment

Altera Quartus II software

3 Description

Use VHDL to design and implement circuits for each of the following.

1. Combinational logic implementing the seven functions needed to operate an active-low seven-segment display. The seven functions are a, b, c, d, e, f, and g, corresponding to the letters on the segments in Figure 1 on the following page. The VHDL module takes four inputs, w, x, y, and z. Taken as a four-digit binary number $wxyz$, cause the seven-segment display to show only the 10 decimal digits. When any input in the range 10–15 is applied, the display should be blank.

   Note: *active-low outputs!*

   Simulate the operation of the circuit for all possible four-bit inputs. The simulation should look similar to that in Figure 2 on the next page

2. A state machine to implement the state diagram in Figure 3 on page 4. The purpose of the machine is to convert a binary number in two's complement format to its two's complement, i.e., to its negative. For example, the two's complement of the six-bit number 010100 is $101011+1 = 101100$ whereas that of 101100 is $010011 + 1 = 010100$.

[1] VHSIC is itself an acronym that stands for Very High Speed Integrated Circuits

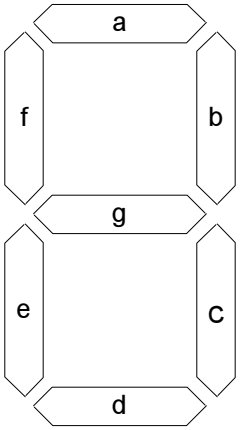

Figure 1: A seven-segment display. The bars are identified by the letters shown within each segment.

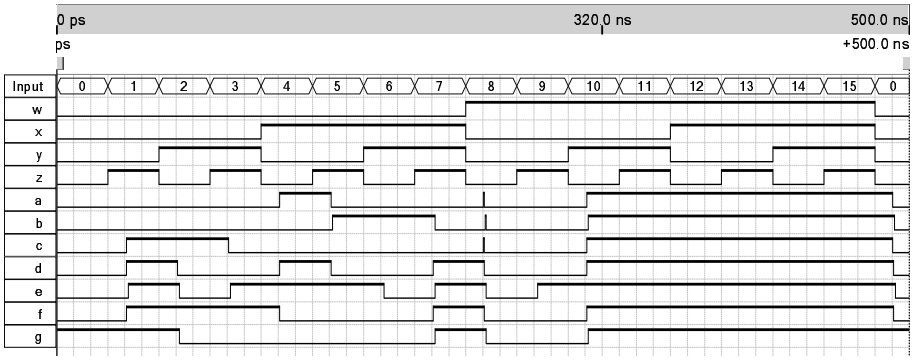

Figure 2: Output from a module to operate a seven-segment display with active-low inputs a through g. The inputs are w, x, y, and z. The Quartus II software lets you group signals, as has been done here for the inputs, and then allows you to have them displayed in other formats. Here, decimal is used instead of binary.

Each number in a pair is the additive inverse of its partner, as can be seen by adding them together. For example, $010100 + 101100 = 000000$. Two numbers that, when added together, yield zero are additive inverses of each other.

The machine works by accepting a single bit at a time, starting with the least significant bit. Bits are copied without alteration until a 1 is encountered. That first 1-bit, too, is copied without alteration, but all bits thereafter are complemented.

States are to change only when the main clock, Clock, changes. This signal does not appear in the state diagram.

The data clock is signal DClock. The data bits come in as signal D. The input bits change when DClock goes high; they are sampled and output when DClock goes low. This convention ensures that data bits are stable when they are sampled.

A third input signal, Done, is 0 while valid data are being transmitted from an outside circuit. The start of an input word occurs at the moment when Done is low and when DClock goes high. The end of the input word occurs at the moment when Done is high and when DClock goes high. Signal Done is not to be examined when DClock goes low, but only when it goes high.

The output data appear as signal E. The state diagram includes notations such as ER↑SB↓SE. This notation implies that the output signal E is controlled by an SR latch. The S-input to the latch is to be asserted when the state begins and the R-input is to be asserted when the state ends.

Figure 4 shows the timing diagram for the state machine when it is presented with the number 0010100 and generates the output number 1101100.

Although the main clock, Clock is shown in the timing diagram as having a frequency four times greater than that of DClock, do not assume this is the case. Assume only that this clock has a substantially higher frequency than the data clock. A ratio of 10:1 or higher should suffice.

Simulate the operation of your circuit using enough two's complement numbers of various lengths to demonstrate that your implementation of the state machine works satisfactorily. Make sure your test cases include the binary equivalents of the decimal numbers 5, 11, 29, 53, and 119. Unlike the sample timing diagram of Figure 4, yours must show the main clock, too.

4 Lab Report

See the ECE Department's web site[2] for a model of how to record design information in a lab notebook and how to generate a suitable formal lab report from

[2] `http://www.usna.edu/Users/ee/cameronc/Miscellaneous/StudentHelp/`

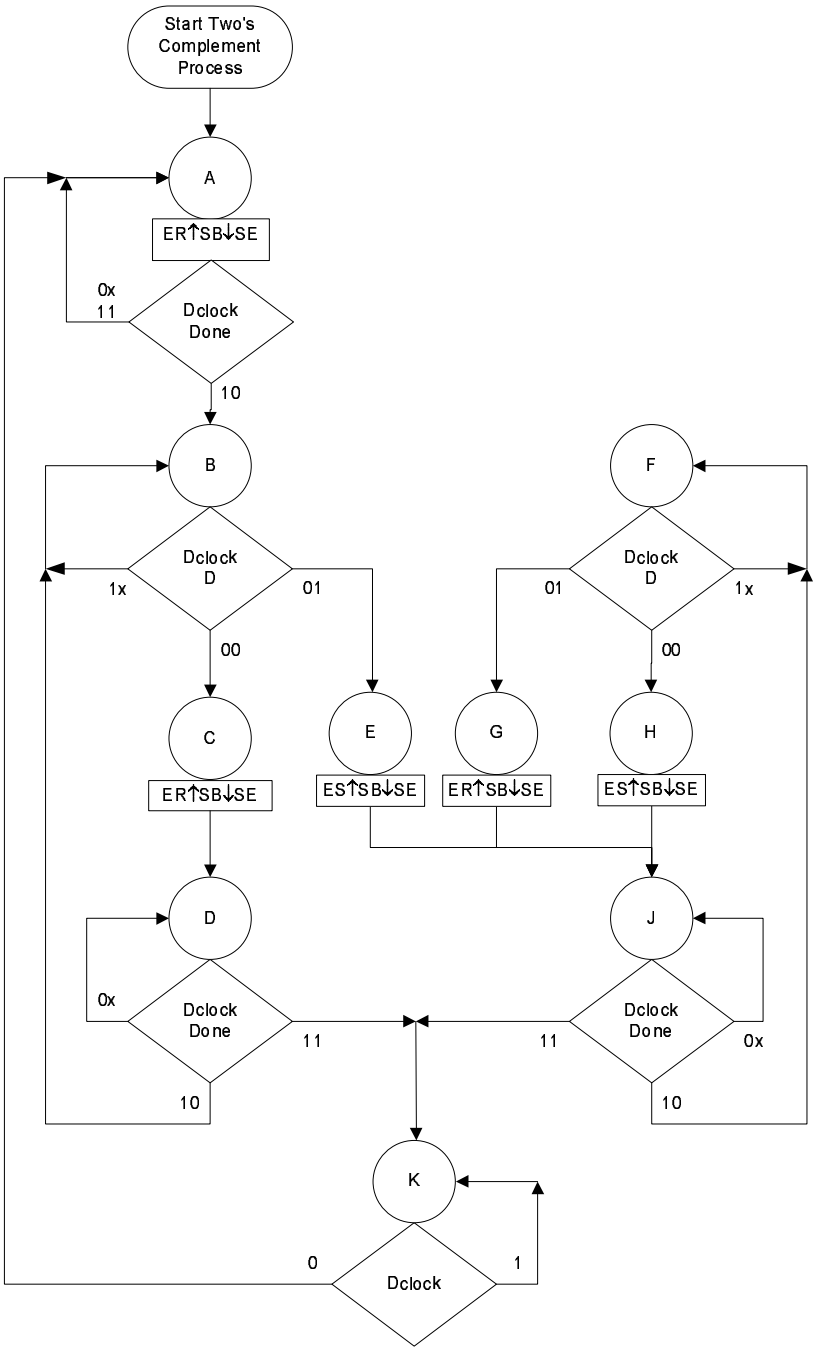

Figure 3: State diagram for taking the two's complement of a number, least-significant bit first. A notation such as ER↑SB↓SE implies that the output signal E is controlled by an SR latch. The R-input to the latch is to be asserted when the state begins and the it to be de-asserted when the state ends. Similarly, ES refers to the S-input to the latch that controls E.

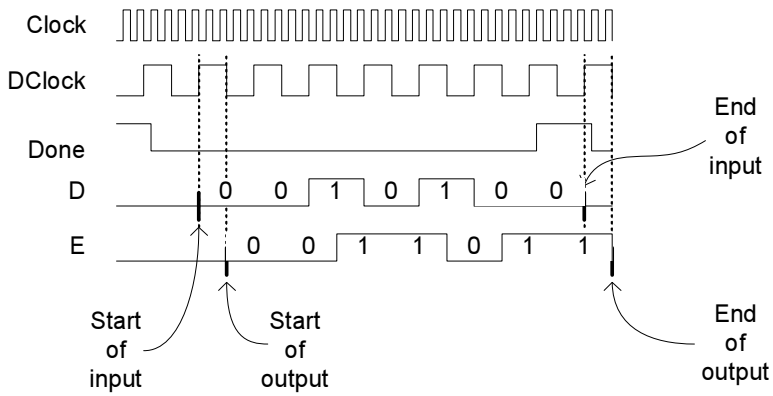

Figure 4: Timing diagram showing how the number 0010100 (representing 20_{10}) is negated by the state machine, yielding 1101100 (which represents -20.) Both numbers appear in the graph with the least-significant bit first.

it. See the EE362 web site [3] for an example of a hierarchical VHDL design with a formal lab report based on it.

Your lab report must contain your fully commented VHDL source files along with timing simulations. The report itself should provide commentary on the VHDL code and the simulations so that they both can be easily understood.

[3] http://www.usna.edu/EE/ee362/Labs/