Lab 5: Clocked Logic

TA: Andrew White
Oct 1, Oct 5, Oct 7

Overview

This lab will serve as an introduction to VHDL features that will enable the realization of synchronous sequential logic storage structures.

Sequential Circuits Review

Sequential circuits consists of combinational logic interconnected with storage elements. A sequential circuit holds state information: the values stored within the storage elements at any point determines the *state* of the sequential circuit at that point in time. The output of a sequential circuit is a function of the state of the device, as well as the inputs. A block diagram of a sequential circuit is shown in Figure 1. In this lab we will only be concerned with *synchronous* (or clocked) sequential circuits. Synchronous sequential circuits consist of a set of registers controlled by a common clock, along with a combinational circuit to compute the next state and outputs.

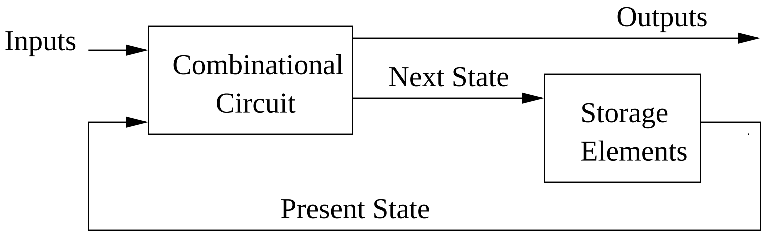

Figure 1: Block Diagram of a Sequential Circuit.

VHDL Signal Attributes

Attributes are a feature of VHDL that allow for additional information about an object (signal, variable, or type) that may not be directly related to the value that the object carries. Attributes also allow for assigning of additional information to objects. For signals, attributes allow VHDL descriptions that take into account dynamic signal behavior like *events* (transitions from one *signal_name'signal_attribute*

Figure 2: Syntax for a signal attribute.

logic level to another). The syntax for obtaining the value of a signal attribute is shown in Figure 2. Common signal attributes are given in Figure 3. An example of a conditional statement to detect a falling clock edge is given in Figure 4.

Attribute	
S'event	true if *event* occurs (signal transition)
S'active	true if *transaction* (signal assignment)
S'stable	true if no *event* occurs
S'stable(t)	true if *no event* has occured for t time units
S'quiet	true if *no event* this simulation cycle
S'quiet(t)	true if signal has been quiet for t time units

Figure 3: Signal Attributes for signal S.

VHDL for Synthesis

In discussing VHDL it is important to mention that the tools that we are using to analyze the VHDL source code effects the type of code that can be written. Every VHDL synthesis tool, like *XST* used in the Xilinx tools for these labs, has a specific way to realize structures like storage elements. This is because unlike a VHDL compiler, such as that which *Modelsim* uses, the synthesis tool is synthesizing the VHDL description into an actual hardware limited circuit that can then be implemented. Remember, Xilinx manufactures FPGAs and their tools are for targeting those devices. Since there are limitations to what code can be synthesized for actual hardware implementation on a particular device, synthesizable VHDL is a limited subset of VHDL. For example, proper VHDL code can use signals of data types such as *real*, *float*, and *integer* and will compile and simulate without error using Modelsim, but this cannot be synthesized in Xilinx. If no standard way was specified for describing such things as registers, each implementation would have to be built by the synthesis tool using combinational logic and would not be optimal in implementation and *may not behave* the way it is expected to. Each synthesis tool will come with supporting documentation that will

if *clk*'**event AND** *clk* = '0' **then**

...

Figure 4: Example of if statement to check for falling clock edge.

indicate what VHDL constructs are supported and how to write VHDL to implement common logic structures. Since the purpose of these labs is to explore VHDL and industry CAD tools, we will limit as best as possible our use of VHDL to that subset which is synthesizable by the CAD tools we are using. In this lab we will look at the supporting documentation for the Xilinx synthesis tool, *Xilinx Synthesis Technology* (XST) .

Lab Procedure

For this lab, you are to use the Xilinx tool's documentation to create a behavioral design. To access the documentation, Click on **Online Documentation** under the **Help** menu in the Xilinx ISE environment. To find information about the synthesis tool and it's VHDL support, find the section for **Xilinx Synthesis Technology User Guide** and find the section titled **VHDL Language Support**. For this lab, you are to create a 16-bit positive-edge triggered register in the format specified in the Xilinx tool's documentation. The register will use one of the signal attributes from the

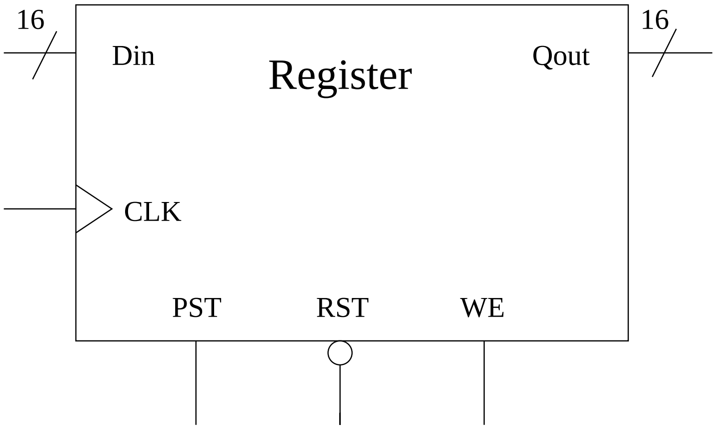

Figure 5: Block Diagram of D Flip-Flop for this lab.

table in Figure 3 and should have an **asynchronous active-low reset** and a **synchronous preset** (set all bits to one) and a **Write Enable**. The synchronous preset will need to have priority over the write enable (if both are asserted, the output should be all '1's). See Figure 5 for the block level view of this register. You will need to adapt the examples given in the documentation to implement the features required for this register. Once you have created your register, create a testbench and test the functionality (reset, preset, write-enable, etc.) through simulation using Modelsim.

Lab Report

The lab report will consist of a cover sheet, your VHDL code for the register and testbench, and the simulation waveforms. Indicate on the waveform where the output is being reset, preset, and where it is registering new values. For the asynchronous active-low reset, test by asserting reset in the middle of a stable period of the clock to show its asynchronous behavior.