VeriLab

An introductory lab for using ***Verilog*** in digital design
(first draft)

VeriLab

An introductory lab for using ***Verilog*** in digital design

Verilog is a hardware description language useful for designing digital systems at various levels of abstraction, from a pure behavioral description until a full structural description. In between, there are many mixed possibilities allowing us to design systems so detailed as we consider.

This laboratory represents a *module of knowledge* devoted to introduce students in using Verilog only for simple digital projects. The main features of the language are exercised starting
from *small* and *simple* circuits until medium scale *complex* systems.

First, we must establish the distinction between *size* and *complexity* in digital systems. *The size of a digital system is given by the number of elementary components used to build the*
system. A good measure for size could be the total number of inputs in all the gates of the system, or the number of two input gates could be used to realize the system. *The complexity of a digital system is given by the magnitude order of the size of its description.* Thus, we can talk about:

- *simple systems* having the size depending by the number of inputs, but having the complexity in the magnitude class of O(1)
- *complex systems* having the size and the complexity in the same magnitude order.

Another goal of this laboratory is to accomdate students with the existing relation between the *level of structuring* of a digital system and the *functional features* so reached. There are two main structural mechanisms used to build a digital system:

- to **compose**, connecting subsystems serially or/and in parallel
- to **close loops** inside the system, connecting some outputs to some inputs.

As we shall see, each closed loop adds new functional features in a digital systems, thus allowing a more significant classification than the pour distinction, in use, between combinational and sequential circuits. Using the number of loops closed inside a system results the following classification:

- **0-OS**: zero-order systems with *no-loop* circuits - the **combinational circuits**
- **1-OS**: first-order systems with *one-loop* circuits - **memory circuits**
- **2-OS**: second-order systems with *two-loop* circuits - **automata**
- **3-OS**: third-order systems with *three-loop* circuits - mainly represented by **processors**
- **4-OS**: fourth-order systems with *four-loop* circuits - mainly represented by **computers**

In the **first VeriLab** we don't need any knowledge about digital circuits because we will describe structures only at the functional level. We need to know only how behaves the structure we want to design.

The Verilog hardware description language allows us to simulate more or less complex systems using only behavioral descriptions. Many times these behavioral descriptions are sufficient to
generate a synthesisable project, i.e., to have a description of what we intend to do that can be automatically translated into a network of logical gates performing the desired function. But,
some times there are needed more details about the internal structure of the system to be built. For this reason follow five other VeriLabs. In each of them are presented specific techniques
for designing digital structure.

The **second VeriLab** introduces the main techniques to describe structurally the combinational circuits - the digital circuits with NO LOOP, belonging to the 0-OS.

The **third VeriLab** deals with memory circuits - the digital circuits with ONE LOOP, belonging to the 1-OS. These first loop closed in digital circuits, if it is done appropriately, offers a
first degree of autonomy - the autonomy to maintain the internal state between two significant events on the inputs.

In the **fourth VeriLab** we shall describe small automata (for example, T flip-flop, JK flip-flop) or simple automata (automata having the loop closed through simple, recursive defined, combinational circuits, for example, counters). All these are TWO-LOOP circuits, belonging to the 2-OS.

The **fifth VeriLab** continues with automata in their *complex* form, called *finite automata*. Now, the combinational circuit which closes the loop is a random circuits having its size in the same order with its complexity. The behavior of these automata will be more complex, corresponding to the loop's complexity.

Finally, the **sixth VeriLab** ends, this short introduction in using Verilog, with the THREE-LOOP circuits, belonging to the 3-OS. Only simple applications are presented because the main representant of this class - the processor - is too complex to be approached in these introductory labs. The goal of this last VeriLab is to emphasize only the effect of the third loop on the complexity of the resulting circuits in contrast with the solutions, for the same problems, allowed by the two-loop circuits.

The amount of matter presented for each VeriLab is more that can be done in a class. The teacher will select the most appropriate problems to be done in the class and the rest can be used as home works.

The problems dealing with the *synthesisable code*, with the *synthesis* and with the *more complex systems* will be the matter of a future module called **Advanced VeriLab**.

Content

VeriLab 1 BEHAVIORAL DESCRIPTIONS IN VERILOG

Problem no. 1 Expandable 8-bit incrementer
Problem no. 2 The behavioral description of an 8-bit adder/substractor
Problem no. 3 A top level structural description of an 8-bit adder/substractor
Problem no. 4 The structural description of 32-bit incrementer using the module from Problem no.1
Problem no. 5 32-bit adder (to be done by students)
Problem no. 6 The n-bit counter
Problem no. 7 The structural description of the counter
Problem no. 8 Up-down counter (to be done)

VeriLab 2 STRUCTURAL DESCRIPTIONS OF COMBINATIONAL CIRCUITS

Problem no. 1 Equality comparator
Problem no. 2 Full adder
Problem no. 3 4-bit adder with ripple carry
Problem no. 4 16-bit priority encoder
Problem no. 5 The behavioral description of a 16 to 1 multiplexer
Problem no. 6 The behavioral description of a 4-bit decoder
Problem no. 7 The structural description of the same decoder

Problem no. 8 The structural description of the previously described multiplexer
Problem no. 9 The transcoder for 7-segment display

VeriLab 3 MEMORIES: LATCHES, FLIP-FLOPS, REGISTERS, RANDOM ACCESS MEMORIES

Problem no. 1 The unstable loop: a three (odd) inverting level loop
Problem no. 2 The stable loop: elementary latch built with two loop-connected NANDs
Problem no. 3 Elementary latch built with two loop connected NORs (proposed to be done)
Problem no. 4 The clocked latch
Problem no. 5 The master-slave principle
Problem no. 6 D (delay) flip-flop
Problem no. 7 D flip-flop with asynchronous set and reset (to be done structurally)
Problem no. 8 The behavioral description of D flip-flop with asynchronous reset
Problem no. 9 4-bit register structurally described
Problem no. 10 The behavioral description of the n-bit register with asynchronous reset
Problem no. 11 Static random access memory (RAM)
Problem no. 12 Multi-port memory

VeriLab 4 AUTOMATA: T & JK FLIP-FLOPS, COUNTERS AND OTHERS SIMPLE AUTOMATA

Problem no. 1 The simplest automaton: the T flip-flop
Problem no. 2 The biggest flip-flop as the simplest 2-input automaton: the JK flip-flop
Problem no. 3 The asynchronous counter
Problem no. 4 The structure of an n-bit synchronous counter
Problem no. 5 How to design all 2-input, 2-state and one output automata?
Problem no. 6 Multiplier (accumulator)
Problem no. 7 Multiplier using an adder and a shifter in order to save time (as home work)
Problem no. 8 Stack memory

VeriLab 5 FINITE AUTOMATA: THE COMPLEX COMMAND OR CONTROL AUTOMATA

Problem no. 1 Automaton for semaphors
Problem no. 2 Automaton of recognizing a regular sequences
Problem no. 3 Electronic key
Problem no. 4 More patient electronic key

VeriLab 6 LOOP-CONNECTED AUTOMATA: THE DEEPEST SEGREGATION BETWEEN SIMPLE & COMPLEX CIRCUITS

Problem no. 1 Automaton for semaphors - a remake of the first problem from VeriLab 5
Problem no. 2 Stack automaton
Problem no. 3 Electronic key - a remake of the second problem from the previous VeriLab
Problem no. 4 Electronic key - a remake of the third problem from VeriLab 5