EITF35 - Introduction to Structured VLSI Design

(Fall 2022)

State Machine Modelling (Sequence Detector)

Lecturer:

Liang Liu

TA:

Sijia Cheng

Ilayda Yaman

Abstract

This document describes basic state machine design for both Mealy and Moore styles. This document is an instructional manual for lab work, which is part of the EITF35 “Introduction to Structured VLSI” course at EIT, LTH. The lab will give the students a chance to practice developing state machines in VHDL. A simulation that proves functionality will be carried out using Xilinx Vivado's in-built simulator. The files for the testbench and stimuli can be downloaded from the course homepage.

Sequence Detector for a Binary Sequence

In this lab, you will learn how to convert a parallel data to serial data and model a finite state machine (FSM) in VHDL that is capable of detecting the occurrences of a given binary sequence in a stream of bits.

Introduction

You will create a sequence detector for a given 4-bit sequence. You will develop a parallel to serial data converter and a sequence detector using Mealy and Moore machine models. This will help you become more familiar with how to implement an FSM-based controller in VHDL. This lab is completed using Vivado' s simulator. You will use a typical HDL flow, write the HDL code, and run a behavioral HDL simulation.

Objectives

After completing this lab, you will be able to:

- How to convert parallel data to serial data
- Perform the design flow to generate state machines in VHDL.
- Simulate a VHDL-based design in Vivado.

Before starting the lab:

1. Draw a timing and block diagram of a 4-bit shift register.
2. Read the following example of a counter (taken from the textbook: “RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability”):

Listing 8.12 Free-running binary counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity binary_counter4_pulse is
   port(
      clk, reset: in std_logic;
      max_pulse: out std_logic;
      q: out std_logic_vector(3 downto 0)
   );
end binary_counter4_pulse;

architecture two_seg_arch of binary_counter4_pulse is
   signal r_reg: unsigned(3 downto 0);
   signal r_next: unsigned(3 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic (incrementor)
   r_next <= r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_pulse <= '1' when r_reg="1111" else
                '0';
end two_seg_arch;
```

Design Example for the FSM

As an illustrative example a sequence detector for the bit sequence ‘1011’ is described here. Every clock-cycle a value will be sampled, if the sequence ‘1011’ is detected a ‘1’ will be produced at the output for 1 clock-cycle. There are two methods to design state machines, one is Mealy and the other is Moore. Here we will discuss both methods. Following is the behavior description of the sequencer for a Mealy style implementation and the state diagram is shown in Figure 1:

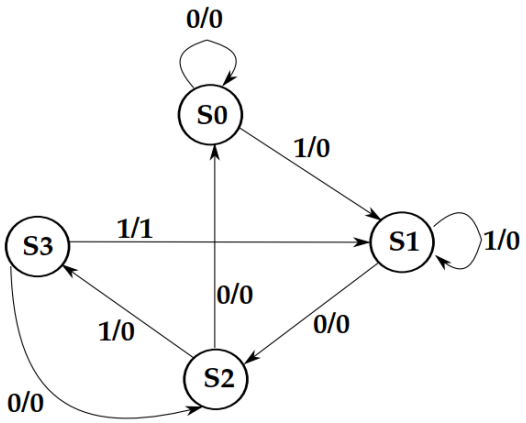

Figure 1: Mealy State Machine for Detecting a Sequence of '1011'

- When in initial state (s0) the machine receives a '1' at its input, it jumps to the next state (s1) with the output equal to '0'. If the input is '0' it stays in the same state.
- When in the 2nd state (s1) the machine gets an input of '0' it jumps to the 3rd state with the output equal to '0'. If it gets an input of '1' it stays in the same state.
- When in the 3rd state (s2) the machine gets an input of '1' it jumps to the 4th state with the output equal to '0'. If the input received is '0' it goes back to the initial state.
- When in the 4th state (s3) the machine gets an input of '1' it jumps back to the 2nd state, with the output equal to '1'. If the input received is '0' it goes back to the 3rd state.

Following is the behavior description of the sequence detector for a Moore style implementation and its state diagram is shown in Figure 2:

- In initial state (s0) the output of the detector is '0'. When machine gets the input of '1' it jumps to the next state. If the input is '0' it stays in the same state.
- In 2nd state (s1) the output of the detector is '0'. When machine gets an input of '0' it jumps to the 3rd state. If it gets an input of '1' it stays in the same state.
- In the 3rd state (s2) the output of the detector is '0'. When machine gets an input of '1' it jumps to the 4th state. If the input received is '0' it goes back to the initial state.
- In the 4th state (s3) the output of the detector is '0'. When machine gets an input of '1' it jumps to the 5th state. If the input received is '0' it goes back to the 3rd state.
- In the 5th state the output of the detector is '1'. When machine gets an input of '0' it jumps to the 3rd state, otherwise it jumps to the 2nd state.

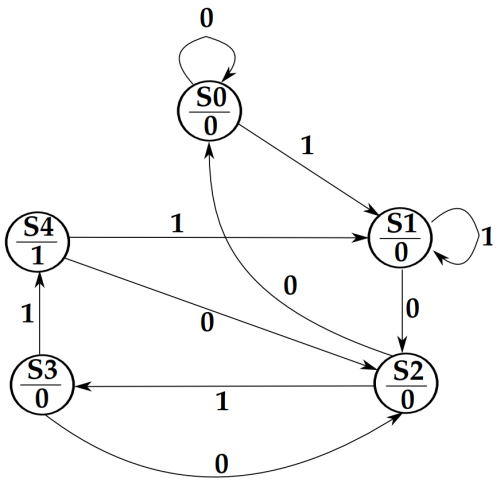

Figure 2: Moore State Machine for Detecting a Sequence of '1011'

After designing the state machines, the models have to be transformed into VHDL code describing the architecture. Therefore, it is helpful to get an understanding about the building blocks. Figure 3 shows the block diagram for the sequence detector to be developed.
The two blocks inside, i.e., the combinational and the register blocks are built out of the two processes used within the architecture in VHDL. The combinational block decides the next state of the FSM according to the current state and the input as well as drives the output according to the state (and input for Mealy implementation). The register block saves the current state of the FSM. This structure can be used to write the VHDL code.

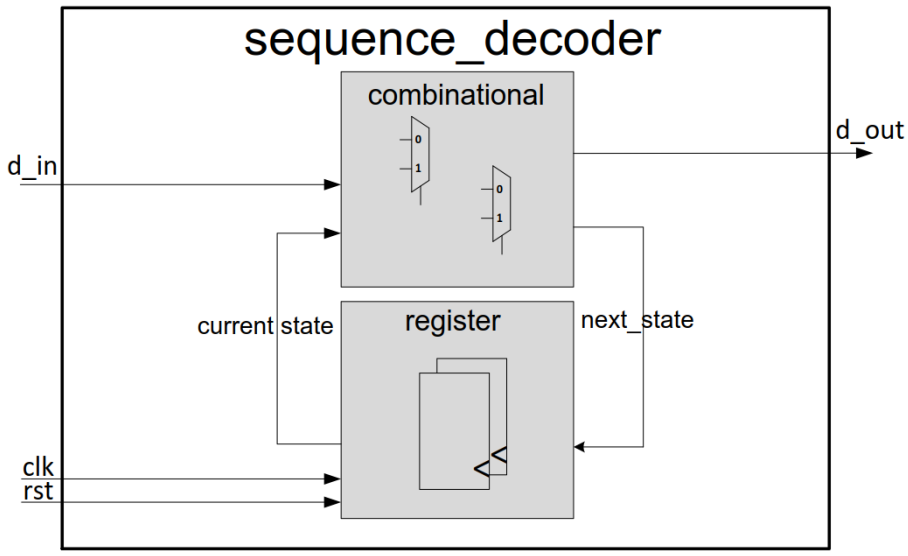

Figure 3: Block diagram clarifying the basic building blocks of an FSM

Testbench

To verify the functionality of the coded hardware it is necessary to simulate the design. In this first assignment all necessary files for testing will be provided. The written VHDL design has to be integrated in the test architecture (sequence_detector_tb.vhd). Figure 4 depicts a block diagram of the testing environment. It is highly recommended to study the provided files to verify that the testbench indeed has the shown structure. The input data is read from a stimuli file using the component stimulus_generator. The read data is then serialized and fed into the instantiated unit-under-test (your design) which is here shown as Moore/Mealy FSM. The clock for the designs is generated by the testbench module.

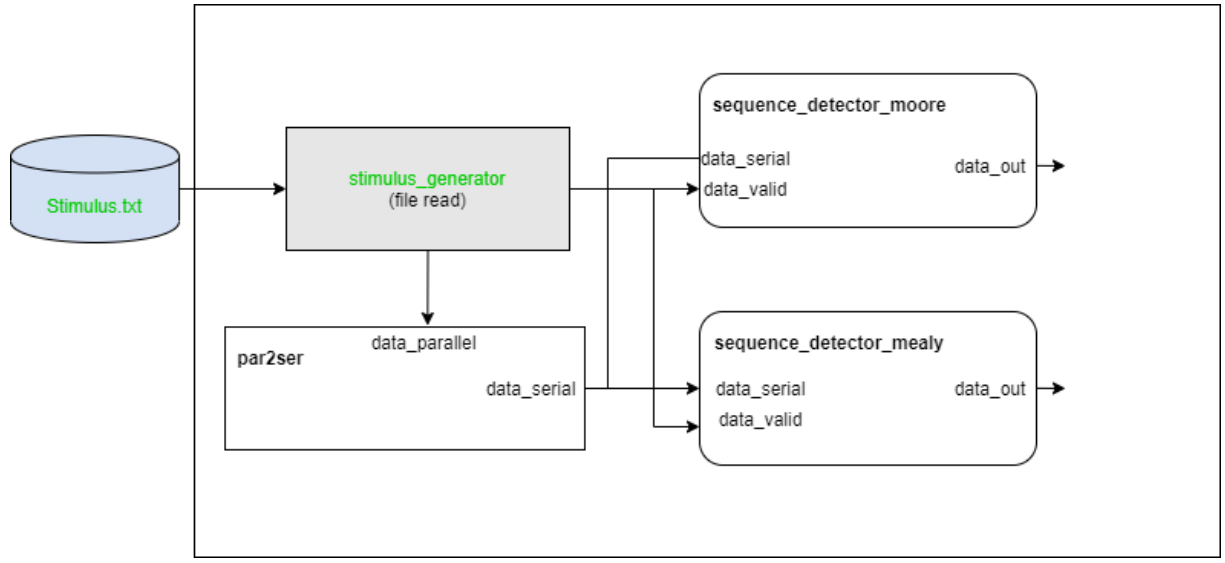

Figure 4

Assignment

You have to perform the following tasks. In order to get approved on this assignment you are required to complete all tasks in time:

1 Draw the timing and block diagrams of the parallel to serial converter and the FSM
2 Cross check your state diagrams, i.e., group 1 checks group 2 and group 2 checks group 1 and so on.
3 Implement the parallel to serial converter
4 Implement the FSM for the detection of the specific bit sequence assigned to your group (available on the website) in VHDL, using both Mealy and Moore implementation. Note that in your design you have to detect the number of occurrences of a specific 4-bit sequence.
5 Check your timing diagrams. Does it look correct? If not, why?
6 Show the simulation of the design in the lab.

(All the files needed to test the implementations can be downloaded from the course web page.)