EITF35 - Introduction to Structured VLSI Design

(Fall 2013)

State Machine Modelling

(Sequence Detector)

Teacher: Dr. Liang Liu

Lab Instructors:

Steffen Malkowsky
Hemanth Prabhu
Chenxin Zhang
Rakesh Gangarajaiah

v.1.0.0

Abstract

This document describes basic state machines for both Mealy and Moore style. This document is an instructional manual for lab work, which is part of course EITF35 "Introduction to Structured VLSI" at EIT, LTH. The lab will give the students a stint to practice developing state machines in VHDL. A simulation that prove functionality will be carried out using QuestaSim VHDL simulator. The files for testbench and stimuli can be downloaded from the course homepage.

Contents

1 Sequence Detector for the sequence '1011' 3

1.1 Introduction 3
1.2 Objectives 3
1.3 Design Example 3
1.4 Testbench 6

2 Assignment 7

1 Sequence Detector for the sequence '1011'

In this lab, you will learn how to model a finite state machine (FSM) in VHDL.

1.1 Introduction

You will create a sequence detector for a given bit sequence. You will develop a sequence detector using Mealy/Moore machine model. This will help you become more familiar with how to implement a FSM based controller in VHDL. This lab is completed using the QuestaSim tool. You will use a typical HDL flow, write the HDL code, and run a behavioural HDL simulation.

1.2 Objectives

After completing this lab, you will be able to:

- Perform the design flow to generate state machines in VHDL.
- Simulate a VHDL based design in QuestaSim.

1.3 Design Example

As an illustrative example a sequence detector for bit sequence '1011' is described. 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, first is Mealy and second is Moore style. We will give you an example for both styles.

Following is the behaviour description of the sequencer for a **Mealy** style implementation and the state diagram is shown in figure 1:

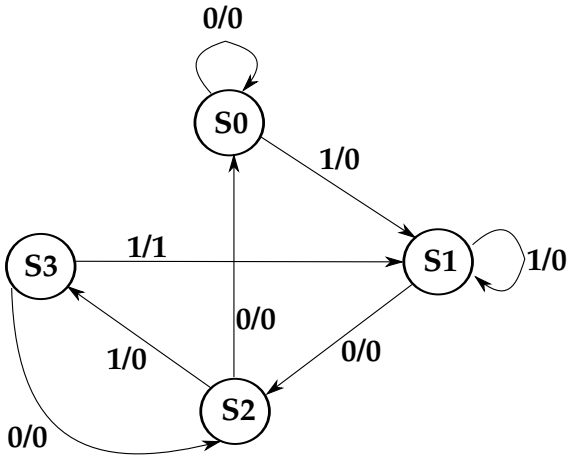

Figure 1: Mealy State Machine for Detecting a Sequence of '1011'

• When in initial state (s0) the machine gets the input of ‘1’ it jumps to the next state with the output equal to ’0’. If the input is ‘0’ it stays in the same state.

• When in 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 behaviour description of the sequencer for a **Moore** style implementation and the 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.

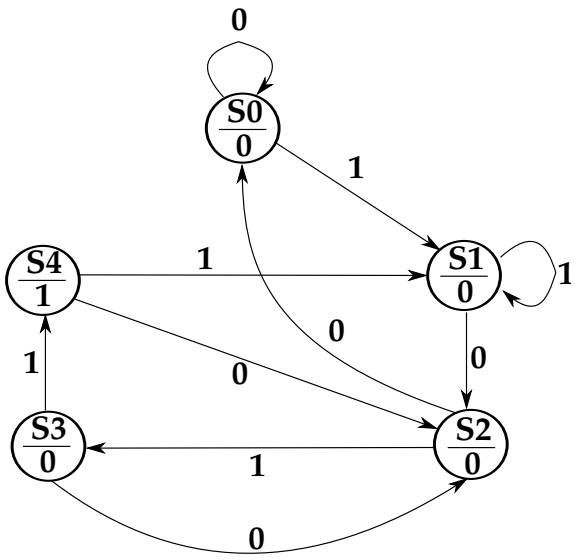

Figure 2: Moore State Machine for Detecting a Sequence of ‘1011’

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 entity for the sequence detector to be developed.

The two blocks inside, i.e., the combinational and the register block are build 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.

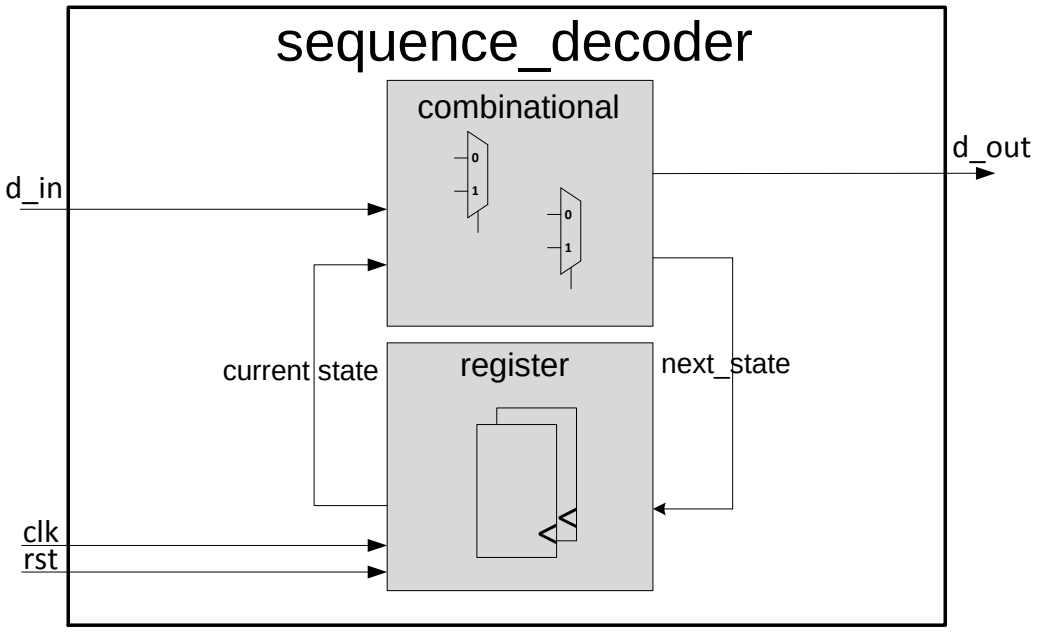

Figure 3: Block diagram clarifying the basic building blocks of an FSM

An example code is shown in Listing 1. This template can be used to code the FSM for this first assignment. All names for entity, processes and signals have been chosen so they are comparable to the block diagram in Fig. 3.

Listing 1: An example code for coding an FSM in VHDL

```
1  use ieee.std_logic_1164.all;
2
3  entity sequence_decoder is
4
5    port (
6      clk   : in  std_logic;
7      n_rst : in  std_logic;
8      d_in  : in  std_logic;
9      d_out : out std_logic);
10
11 end sequence_decoder;
12
13 architecture sequence_decoder_arch of sequence_decoder is
14
15   -- Define a enumeration type for the states
16   type state_type is (s0, s1, s2, s4);
17
18   -- Define the needed internal signals
19   signal current_state, next_state : state_type;
20
21 begin  -- sequence_decoder_arch
22
23   -- purpose: Implements the registers for the sequence decoder
24   -- type   : sequential
25   -- inputs : clk, rst, next_state
26   -- outputs: current_state
27   reg: process (clk, rst)               -- The register process
28   begin  -- process register
29     if rst = '1' then                   -- asynchronous reset (active high)
30       current_state <= ???;
31     elsif clk'event and clk = '1' then  -- rising clock edge
```

```
32        current_state <= ???;
33      end if;
34    end process register;
35
36    -- purpose: Implements the next_state logic as well as the output logic
37    -- type   : combinational
38    -- inputs : d_in, current_state
39    -- outputs: next_state, d_out
40    comb: process (d_in, current_state)   -- The combinational process
41    begin  -- process combinational
42
43      -- set default value
44      d_out <= '0';
45      next_state <= current_state;
46
47      -- next-state logic
48      case current_state is
49        when s0 =>
50          if d_in = '0' then
51            next_state <= s1;
52          else
53            next_state <= s0;
54        end if;
55        when s1 =>
56                    ???
57                    ???
58                    ???
59
60        when others => null;
61      end case;
62    end process combinational;
63
64  end sequence_decoder_arch;
```

1.4 Testbench

To verify the function of the coded hardware it is necessary to simulate the design. In this first assignment all necessary files for testing will be provided. However, the written VHDL design has to be integrated in the test architecture (FSM_test.vhd). Just uncomment the prepared code which instantiates and connects your written FSM with the testing environment. 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 FILE_READ. 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 component CLOCKGENERATOR whereas the reset signal is fed from the test architecture in FSM_test.vhd.

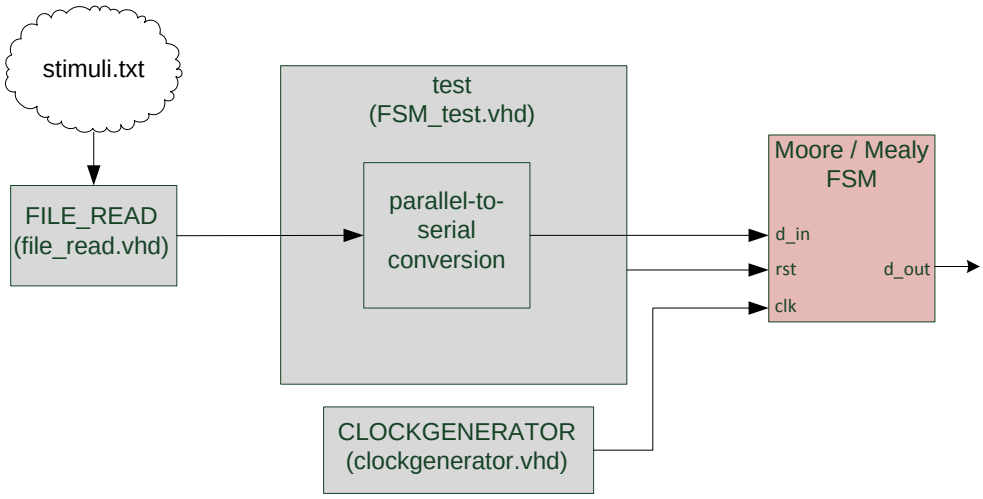

Figure 4: Block diagram of the testing environment

2 Assignment

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

1. **Submit the state diagram for both Mealy and Moore implementation before lab for the assigned sequence. This is a prerequisite to get access to the labs. Deadline 2013-09-10.**
2. **Implement the same algorithm for detection of the sequence in VHDL using both Mealy and Moore implementation.**
3. **Show the simulation of the design in the LAB using QuestaSim. Deadline 2013-09-13.**

The sequences will be distributed for each lab group during the second lecture (2013-09-03).

(All the files needed to test the implementations are placed in "S:\assignment_1\" folder, a stimuli file is also provided. You can also download these files from the course web page.)