CPE 322 Digital Hardware Design Fundamentals
Homework Assignment #2

Design and Implementation of Sequential Logic that is described as a Finite State Machine

Purpose
The purpose of this homework is to review traditional sequential Finite State Machine, FSM, design methodologies that were introduced in the EE 202, Digital Logic Design class and in Chapter 1 of the course text. The homework also illustrates how such designs can be created using ROM type lookup tables and multiplexers to implement the combinational portion of the logic. Implementing combinational logic in this way is a common method employed by CAD tools when they synthesize designs for FPGA implementation.

Assignment
This assignment requires that students develop multiple Mealy and Moore Finite State Machine implementations of a functional unit that performs two's complement addition of two numbers that are of arbitrary length.

Background
This homework assignment focuses on the design of a sequential Boolean network whose function is to perform a two's complement addition of two arbitrary length binary numbers **A** and **B** producing as sum **S** of the same length. These two inputs are assumed to be sent-bit serially in time least significant bit first. The output **S** is also assumed to produce the sum in a bit serial manner least significant bit first (see Figure 1). The network also contains a **Reset** and **Enable** input. The **Reset** input is the means by which an external controller can put this design back into its initial state so it can be used to sum up another two numbers. The **Enable** input allows the external controller to pause your design (in effect skipping clock cycles) so that it ignores inputs that are invalid. It is the job of the controller to make sure the Boolean network has been reset when a new number is sent and the network will only input valid bits. Your job is to develop the sequential network in a manner the considers only the inputs **A**, and **B**, and the output **S**. In portions of this homework that require flip-flop state assignments assume that the initial state occurs when all of the flip-flops are zero. This would be the case when all the *reset* and *enable* signals are connected together as shown in the figure.

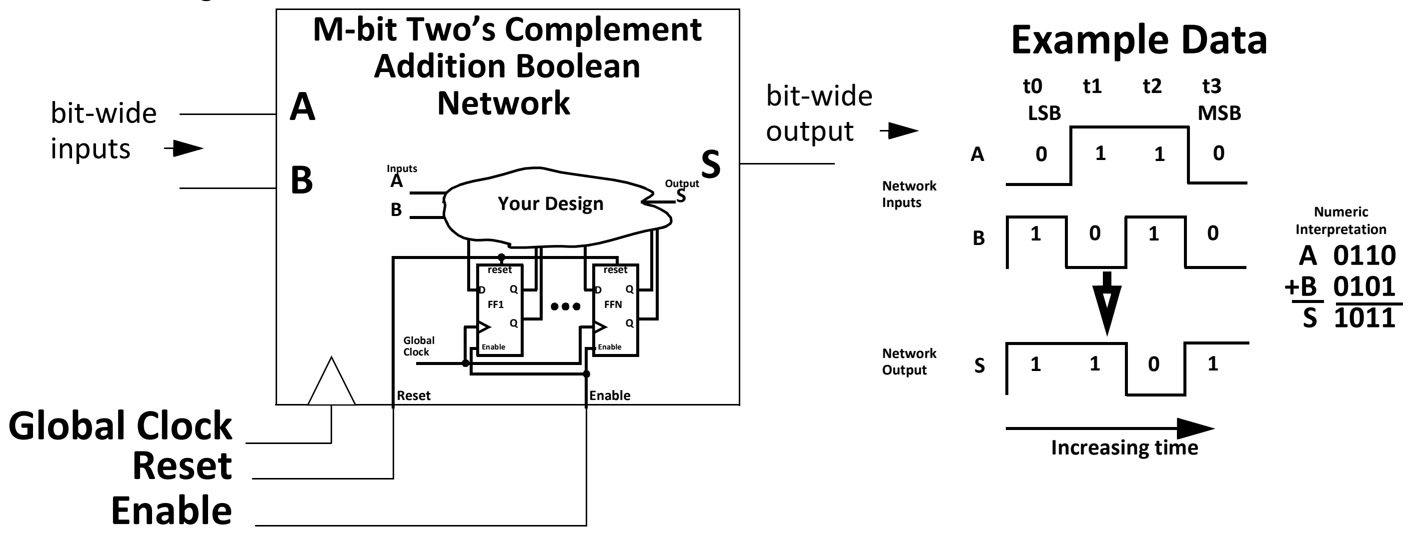

Figure 1: Functional Overview of Sequential Adder

Part 1: Mealy Implementation of the FSM State Graph

Develop a Mealy FSM State Graph (SG) for the Two's Complement Design shown in Figure 1. The design should contain the minimum possible number of states possible, which is 2, to implement the desired function. Identify the initial state and fully label all states in the diagram. Also label the input(s) and output of the design.

Part 2: Mealy minimum 2-level AND/OR/NOT logic & D-flip-flop Implementation

Implement the SG design in Part 1 using the minimum set of discrete logic gates (AND, OR, and NOT) and D-flip-flop for the state assignment that you have chosen. Show all step in this process. Evaluate the correctness of your design by neatly comparing the output of your final implementation with that which is predicted by the SG for the sequence that you developed in Part 1.

Part 3: Mealy ROM logic realization & D-flip-flop Implementation

Replace the combinational logic potion (AND, OR and NOT gates) of Part 2 with a ROM that is of minimal size to implement the SG of Part 1 with a minimal number of D-flip-flops. Clearly identify the Address and Data buses of the ROM element and specify its complete ROM Table.

Part 4: Mealy 4-to-1 MUX logic realization & D-flip-flop Implementation

Replace the combinational logic potion (AND, OR and NOT gates) of Part 2 with one or more 4-to-1 multiplexers that will implement the SG of Part 1 with a minimal number of D-flip-flops. Clearly identify the inputs and outputs of the multiplexers. Assume positive logic with a Logic 1 representing VCC and a Logic 0 representing GND. You may also use inverters (NOT gates) as needed.

Part 5: Moore Implementation of the FSM State Graph

Develop a Moore State Graph, SG, for the Two's Complement Design shown in Figure 1. The design should contain the minimum possible number of states possible, which is 4, to implement the desired function. Identify the initial state and fully label all states in the diagram. Also label the input(s) and output of the design.

Part 6: Moore One-hot encoded logic realization & D-flip-flop Implementation

Implement the SG design in Part 5 using again the minimum set of discrete logic gates (AND, OR, and NOT) but this time implement your state assignments using one-hot encoding. In this encoding scheme each state has a corresponding D-flip-flop which is at a Logic 1 whenever the design is in that state and is a Logic 0 otherwise. Assume in this case that the D-flip flops have both presets and clears which will be used to make sure that the state is initialized with only the flip-flop associated with the initial state being a Logic 1 and the remaining flip-fops being a Logic 0. Show all step in this process. Evaluate the correctness of your design by neatly comparing the output of your final implementation with that which is predicted by the SG for the sequence that you developed in Part 5.

Submission

Place all components of this design in as single pdf file and upload it on Canvas.