Term Project 2: Report

Project summary

This project's goal is to implement sequential logic for the "missionaries and cannibals" problem. In this problem, three missionaries and three cannibals must cross the river, but at most two can be carried at once. Cannibals must also not outnumber missionaries on one side.

This project is an extension of the combinational logic created for this problem in term project 1, which will be reused in this project as a submodule. In order to include sequential logic, this project will also have registers for the state and direction. Additional logic will also be implemented to determine the direction correctly.

Module description

missionary_cannibal (top module)

missionary_cannibal is the top module. It consists of several submodules and is based on the figure provided in the project description slides. Below is the figure generated by Quartus based on the code.

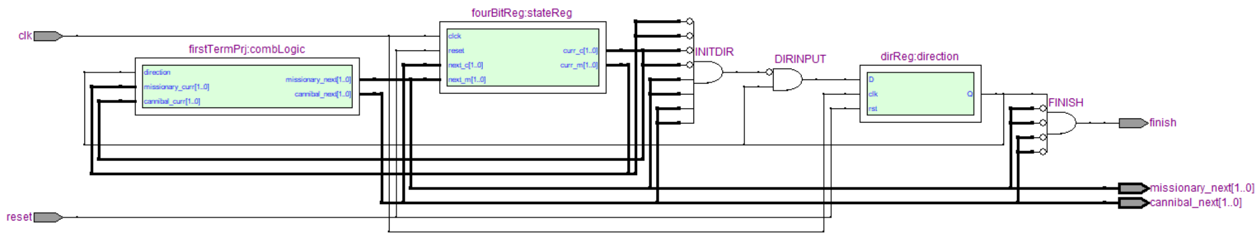

Figure 1: Top Module

As specified in the requirements, it has two inputs – *clck* and *reset*, and the next state and *finish* as output. The submodules are *direction* (instance of *dirReg* module), *stateReg* (instance of *fourBitReg* module) and *combLogic* (instance of *firstTermPrj*) – these will be explained in detail later.

To connect the submodules and ports several wires were used. Wires *curr_m* (current missionary state) and *curr_c* (current cannibal state) were used to connect output ports (with the same names as the wires) of the *stateReg* module to input ports *missionary_curr* and *cannibal_curr* of the *combLogic* module. They were also used in the *INITDIR* gate, which will be described later.

The wire called *dir* (direction) is the wire which will be connected to the *Q* output port of the *direction* module, *direction* input port of *combLogic*, and *FINISH* and *DIRINPUT* gates.

Wires *next_m* (next missionary state) and *next_c* (next missionary state) connects the output ports *missionary_next* and *cannibal_next* of *combLogic* to the output ports *missionary_next* and *cannibal_next* of the top module. Additionally, they are connected to input ports *next_c* and *next_m* of *stateReg*, and also the *FINISH* and *INITDIR* gates in the top module.

The last two wires are *initDir* (1 if direction should be intialized) and *dirInput* (the input chosen for input port *D* of *direction* module). *initDir* is the output from gate *INITDIR*, while *dirInput* is the output from gate *DIRINPUT* which is connected to input gate *D* of *direction* module.

The input ports *clck* and *reset* are also directly connected to the corresponding input ports of submodules *stateReg* and *direction*. Wires were not used to connect them as they could be connected directly.

Besides the wires and submodules, there are three gates in the top module. Firstly, there is a 5-input AND gate (called *FINISH*) which is used to check if the next state and direction is final (next state is 0000 and direction is 1). In other words, finish is 1 if the direction is 1 and the inverse of next state is 1111. This gate is therefore connected to the next state outputs from the *combLogic* module (*next_c* and *next_m* wires), which is inverted, and the *Q* output from the direction module (*dir* wire). This is then connected to output port *finish* of the top module.

The next gate is called *INITDIR*. The project requirements state that state and direction should be set back to initial if it is far from the solution. The code from project 1 sets the next state to initial state, but it does not affect the next direction. Therefore, some additional logic was needed to modify the direction in case state is initialized internally in this module. That is the purpose of this gate – it will be 1 if state could have been initialized internally in *combLogic*. In order to do so, it makes sure that the current state is not initial and that the next state is initial (i.e. a change to initial state has been made inside *combLogic*) – *INITDIR* is therefore an AND gate taking the next state (*next_c* and *next_m* wires) and the inverse of the current state (*curr_c* and *curr_m* wires) as input. The output of this gate is the *initDir* wire.

Lastly, *DIRINPUT* is an AND gate which is used to determine the input *D* of the *direction* module. Its input is the complement of the *initDir* wire and the previous output *Q* from the *direction* module (*dir* wire). The output for this gate will be the next *D* input for *direction* module and is therefore connected to that input port through the *dirInput* wire. The reason why *initDir* is complemented will be explained under the description of the *direction* module.

combLogic

This module was created in term project 1 and is reused without any modification. Therefore, it will therefore not be described in detail in this report (see term project 1 report for a more detailed description of module).

This submodule takes three input – direction and current state of missionaries and cannibals. The output is the next state based on the input. In case the current state is the goal state or far from the solution, the state is set back to the initial state. Otherwise, the appropriate number of missionaries/cannibals are moved according to the solution provided.

dirReg

dirReg is the submodule which determines the direction. This is a D flip/flop with asynchronous reset which is based on the code provided. Its input is *D*, *clk* and *rst*, and output is *Q*.

This module should transition from 0->1 and repeat, unless reset is 1. The code provided already resets output *Q* to 0 in case of *rst* = 1.

In order to make it transition between 0 and 1, the complement of the input *D* is chosen as output *Q*, and output *Q* is then indirectly connected to input port *D* through wire *dir* in top module. The reason why output *Q* is not directly connected to the input port is because the direction may need to be initialized – which is the reason why gate *DIRINPUT* is in the top module. In other words, the previous output of the module should be used as input unless direction should be initialized. *DIRINPUT* gate therefore has *init* (output from *INITDIR* in top module) and *dir* (output *Q*) as input. Because *init* is 1 if it should be initialized, we therefore have that the input *D* for module *direction*, and the output of gate *DIRINPUT*, should be as follows:

init	*Q*	*D*
0	0	0
0	**1**	**1**
1	0	0
1	1	0

D = init’ * Q

Therefore, in order for this module to receive the correct input, there is an AND gate in the top module, with previous output *Q* and the complement of *init* and as input.

fourBitReg

fourBitReg is the submodule which acts as a 4-bit register for the current state. This module consists of four instances of the *stateRegister* module – one for each bit. The purpose of this module is mostly to organize the one-bit registers into one 4-bit register module. The input is *clk*, *reset* and next state (which will be saved as the new current state). This module then distributes the bits of next state to be saved to one *stateRegister* submodule each. The output is then state saved in the 1-bit registers combined.

stateRegister

Lastly, *stateRegister* is a D flip/flop acting as a register for one bit. Again, this is based on the DFF code provided. It has input *D*, *clk* and *rst*, and output *Q*. Unlike the code provided, in the case of reset, the state should be set to initial state which is 1111. Therefore, the code was modified so that *Q* = 1 if *rst* = 1.