Lab #9

**Objective:**

To design synchronous as counters, utilize state machine designs, and further utilize VHDL

**Design:**

**Design 1:** A self-starting 3-bit synchronous counter that counts up odd numbers

States:

000 -> 010 -> 100 -> 110 -> 000



Table - Design 1 Truth Table

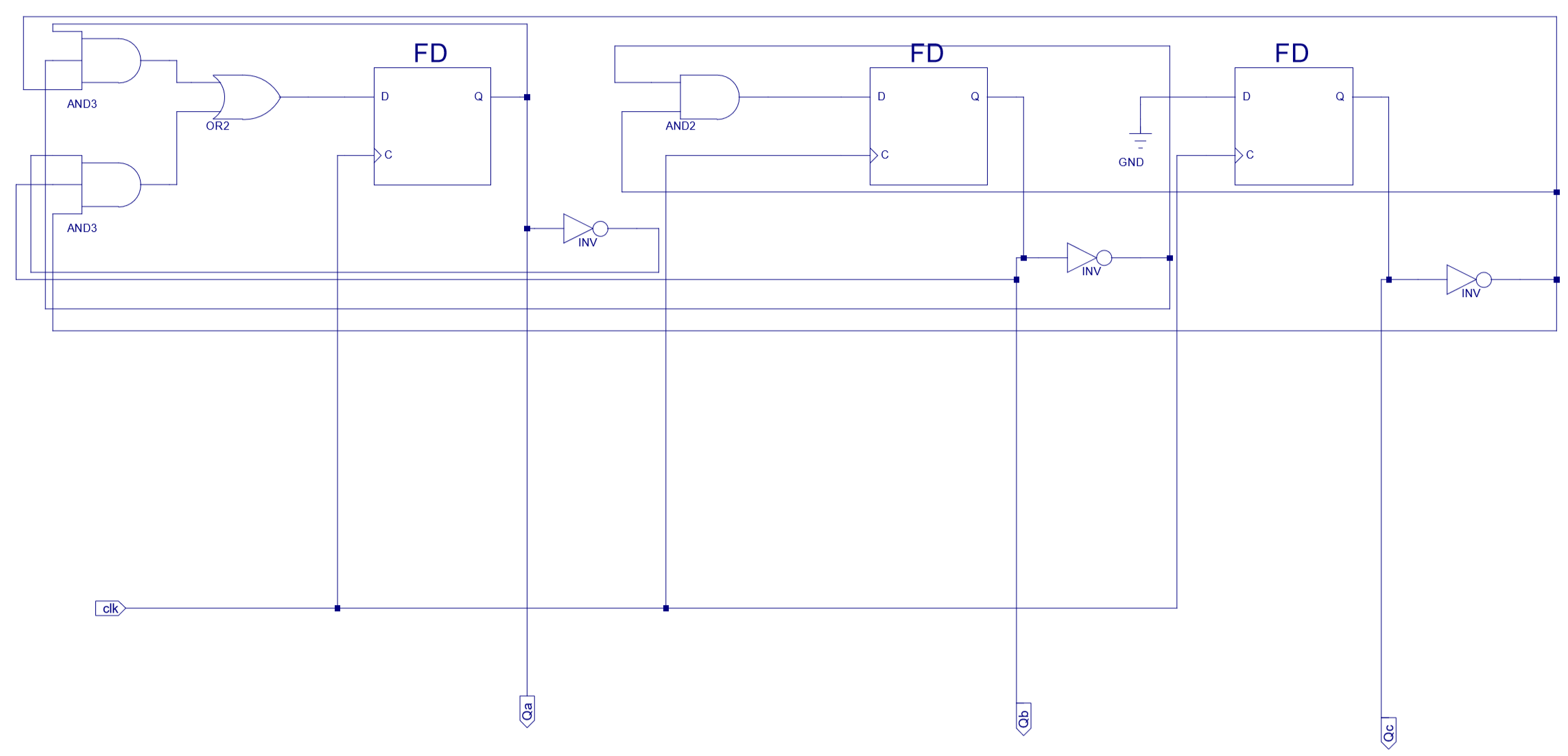


Figure - Design 1 Schematic Diagram

**Design 2:** A self-starting 4-bit synchronous counter that can count up or down even numbers

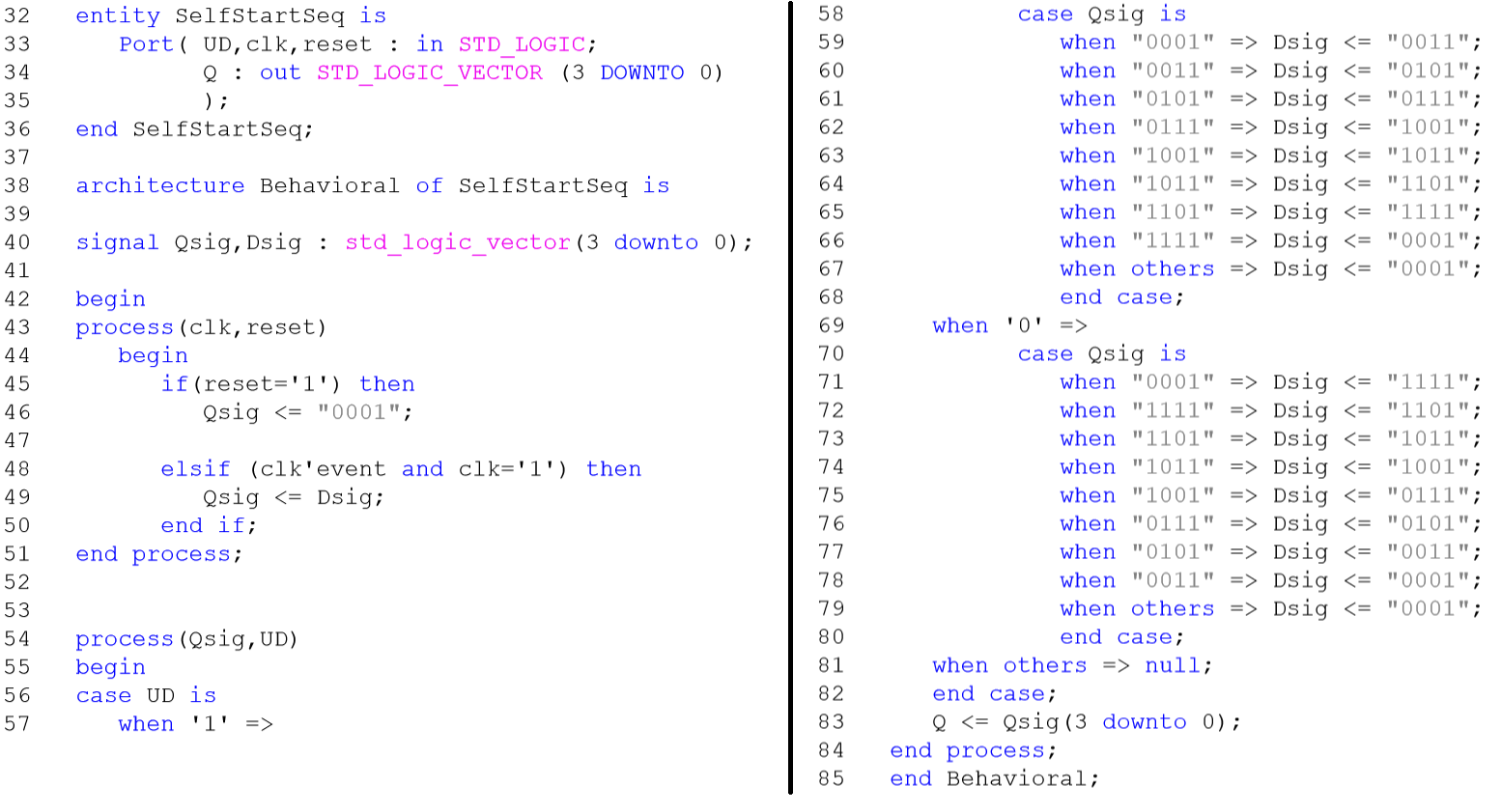


Figure - VHDL Design for Design 2

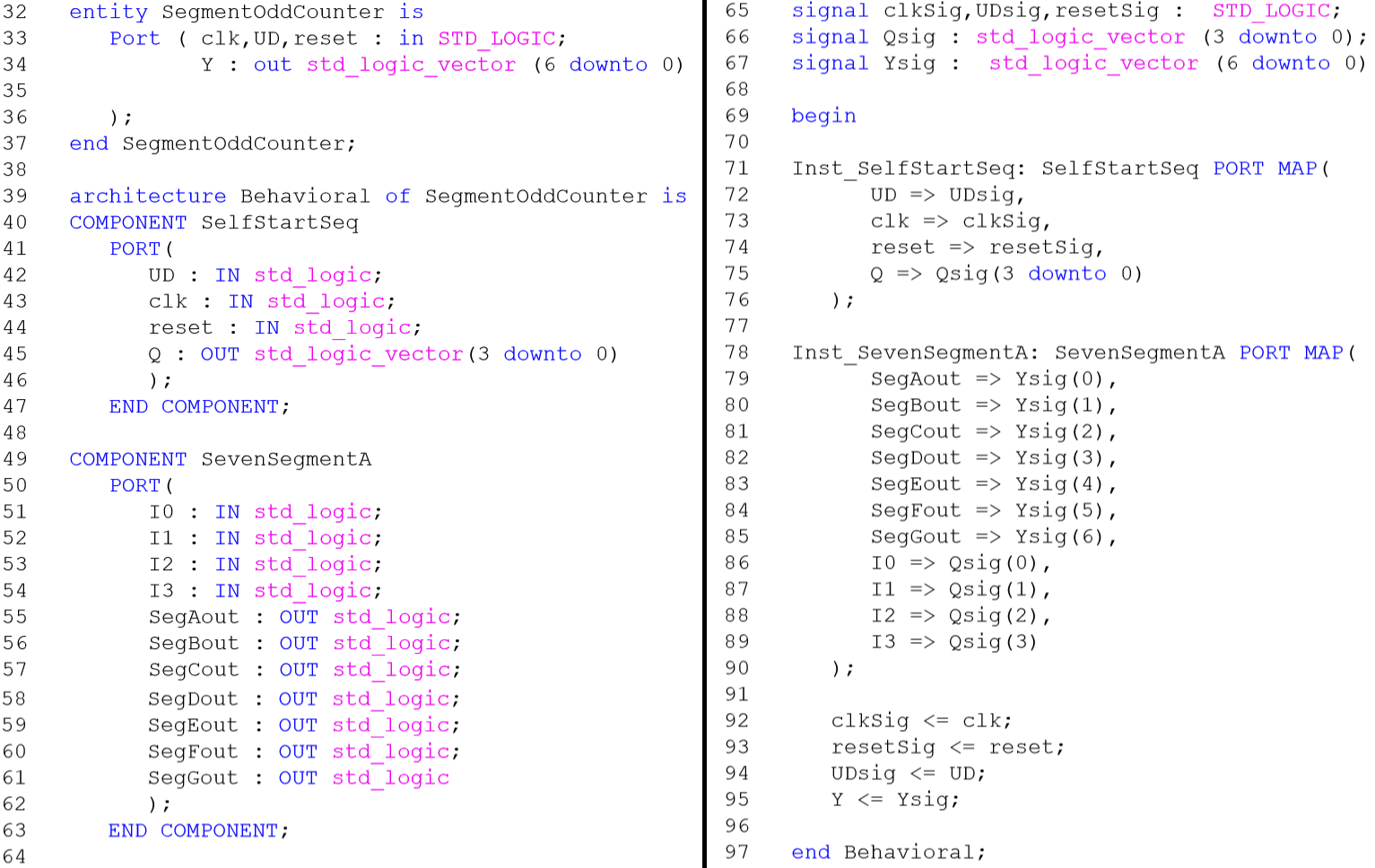


Figure - VHDL Design for Design 2 with 7-Segment Decoder

**Procedure:**

**Design 1:**

* Create a new project
* Create a schematic file of the counter from reference
* Run behavioral simulation with the clock set to a 2ns period and run for 20ns
* Import the previously designed 7-segment decoder into the project folder
* Create a schematic symbol of the 7-segment decoder
* Add the 7-segment decoder to the counter schematic
  + Connect the 3 outputs from the counter to the lower 3 bits of the 7-segment decoder
  + Connect the remaining 7-segment input to ground
* Save the schematic
* Assign pin numbers with the clock input connected to a switch
* After making the .ucf file
  + Open the .ucf in notepad or another text editor
  + Copy the below line to the end of the text file
    - NET "XXXXX" CLOCK\_DEDICATED\_ROUTE = "FALSE";
  + Replace “XXXXX” with the input name in double quotes
* Download to the FPGA and test

**Design 2:**

* Create a VHDL file for self-starter 4-bit up/down odd counter with an input to decide direction
* Run behavioral simulation with the clock set to a 2ns period and run for 20ns
* Create a new VHDL file with the 7-segment decoder and the up/down odd counter as components
  + To make a component
    - Select VHDL file
    - Expand Design Utilities
    - Double click VHDL Instantiation Template
  + To implement the template in another VHDL File
    - Copy and paste the component section between the Architecture and Begin statement
    - Copy and paste the component section between the Begin and End Behavioral statement
* Connect the outputs of the counter to the inputs of the 7-segment decoder using signals
* Compile and assign the pins
* After making the .ucf file
  + Open the .ucf in notepad or another text editor
  + Copy the below line to the end of the text file
    - NET "XXXXX" CLOCK\_DEDICATED\_ROUTE = "FALSE";
  + Replace “XXXXX” with the input name in double quotes
* Download to the FPGA and test on the board

**Data:**

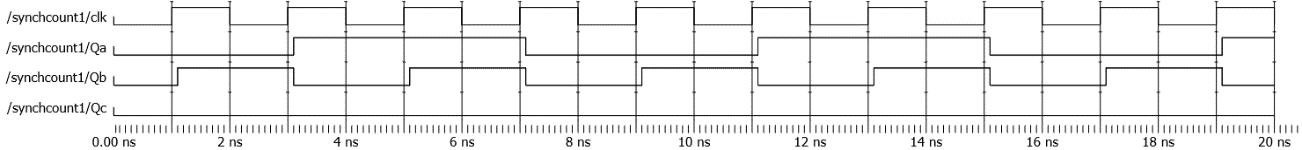


Figure - Design 1 Behavioral Simulation

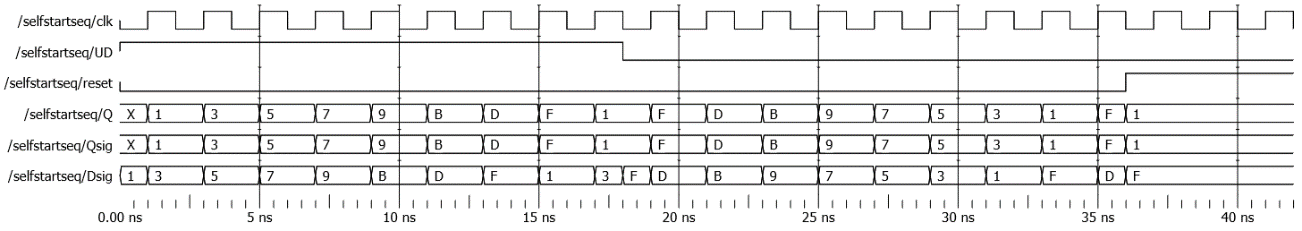


Figure - Design 2 Behavioral Simulation

**Data Analysis:**

The designs performed as expected. One thing that I did not account for in interpreting the first design was the starting what would happen in the starting cycle if the LSB was read as high instead of low. Looking after the fact, the provided circuit would reset the second and third bits to low on the next cycle as ground pushes low to the LSB. In fewer words, it self-starts and resets back to 000 if any unused states (XX1) are entered.

**Conclusion:**

This lab showed the implementation of counters, expanding on the concept of using outputs to drive inputs, in this case driving a specific sequence of outputs. This ties directly to designing with state machines. By designing the outputs or flip-flop states and a specific sequence for them to change between, it makes it much simpler to work backward and design how the inputs need to be affected to reach the next state.

This lab continued previous difficulties for me, namely in VHDL syntax. My initial VHDL design for the self-starting up-down odd counter would not have worked and needed to be almost completely rewritten. The core problem was not understanding implementing multiple processes in a design and once the flip-flops were broken into their own process within the VHDL code the rest more or less fell into place with my original understanding.