CS373 LM9: Introduction to Sequential VHDL   
A True Divide-By-Three 50% Duty Cycle Clock

Name(s) Shane Snediker

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| DivideBy3FSM |  | 50 |
| TOTAL |  | 50 |

## Learning Goals for This Study Guide

* Gain an understand of how to create VHDL from a hardware diagram.
* Understand how to simulate and implement a true divide by 3 circuit
* Learn how to implement sequential circuits in VHDL
* Challenge critical thinking skills by implementing a new project.

## Instructions for This Project

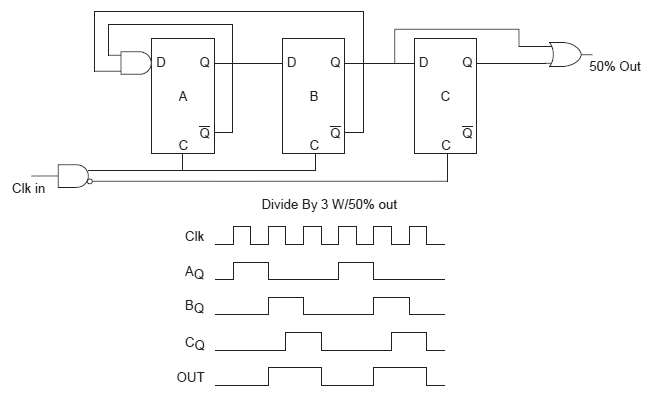
* You may work by yourself or with another person. You must **both** submit the code to whitgit along with your changes to this document. You must each answer the questions in this document by yourself, but you may consult each other on your answers.
* Answer the questions on the last page of this document and submit this document (with your answers in it) to Blackboard.
* Follow the instructions given in class to install Vivado on your machine (or use the machines in the lab – if you use lab machines which already have Vivado installed, you will need to install the board files for the BASYS 3)
* Create the majority project using the TCL/Bash scripts provided in class. You will need to configure the TCL script so that it builds your majority project.

## Divide by 3 Project

Now it’s time to put your newly learned skills to use. For this project you will design and implement the VHDL for a 50% duty cycle divide by three circuit.

## DivideByThree Project Specifications

* Pull the LM9\_Sequential\_VHDL starter project folder that contains this document from whitgit using vscode and git.
* **Complete the VHDL code in the DivideBy3FSM\_50.vhd file.**
* Run the provided testbench simulation to verify that it works correctly.
* Program the BASYS 3 Board and verify LED(1) and LED(0) are lighting up appropriately.



**reset**

## Divide by Three Circuit Design

**y**

**RESET**

**RESET**

**RESET**

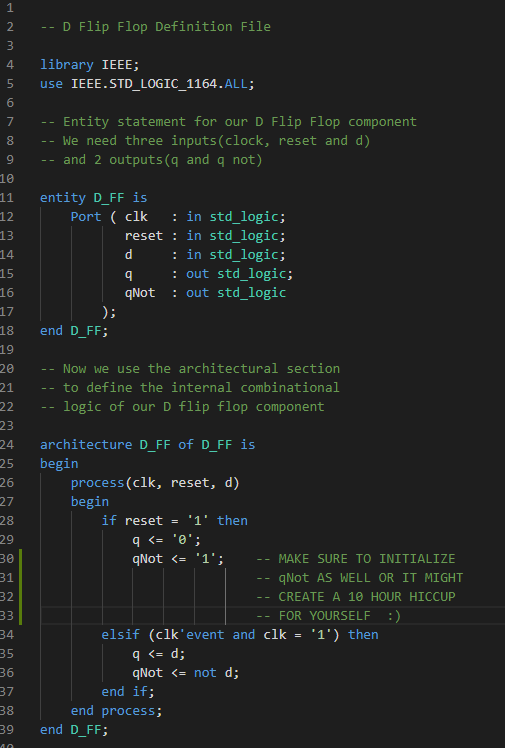
**VHDL Design**

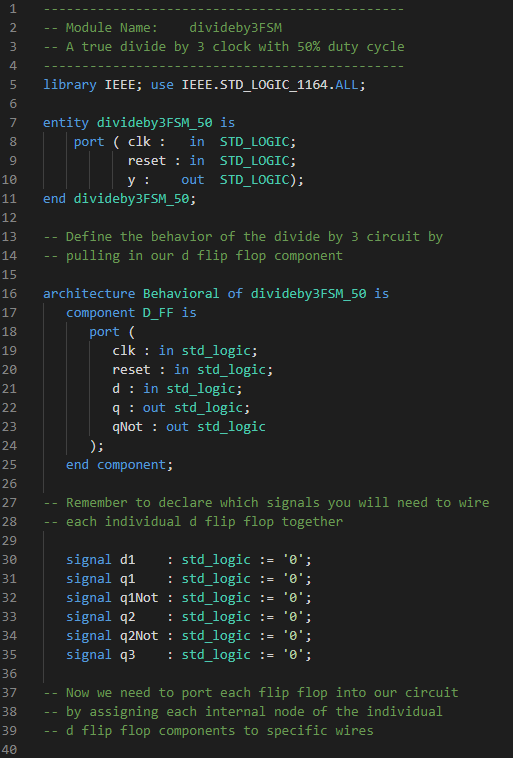
Implement the circuit as shown in this diagram.

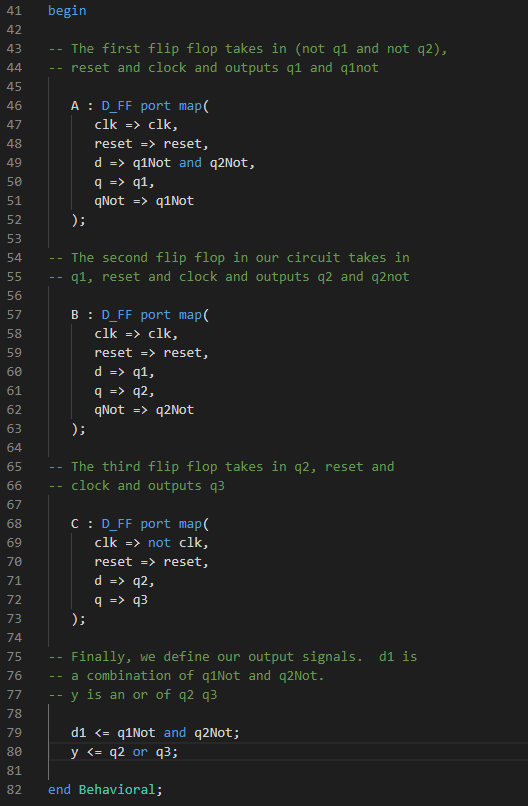
## Complete the Following:

1. Copy and paste the VHDL for your divide by 3 circuit here:

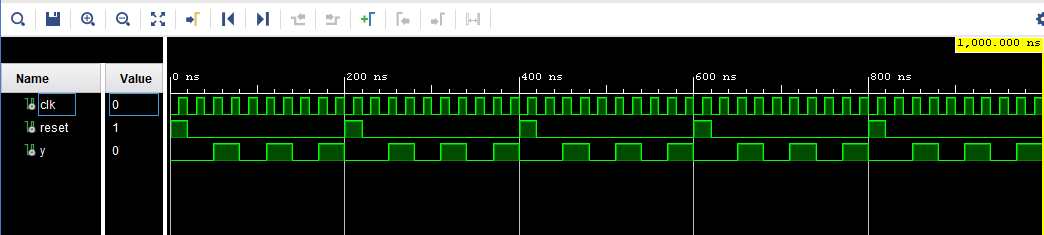
The following are screenshots of my dflipflop component vhd file and the completed divideBy3FSM\_50.vhd file:







1. Paste a screen shot of your divide by 3 simulation working here:



1. How could you modify this circuit to become a divide by 6?

The divide by 3 with a 50% duty cycle uses 3 edge triggered d flip flops to produce the clock division. It does this by routing the first 2 d flip flops q not signals as feedback loops back into the first d flip flop and giving the y output of the disjunction of the second flip flop and the third flip flop. In order to slow the clock down even more to produce a divide by 6 circuit, we would need to produce a slower feedback loop, I believe. I’m sure there’s multiple ways that this could be accomplished, but one way would be to keep our design mostly intact with flip flop A, flip flop B, flip flop C and a clock signal. Only we could give each of them the same clock signal (as opposed to giving flip flop C clkNot), do not output qNot for flip flop A or flip flop B, give flip flop C’s q signal to Y and provide flip flop C’s qNot as the feedback loop connected as flip flop A’s input D. This would output 1/6 of a frequency cycle, I believe:

DA QA DB QB DC  QC Y

~QA ~QB ~QC

CLK

1. Explain how the main clk signal is generated that feeds this circuit. Hint: Look in the top level module file. How is the clk that connects to this circuit generated? Explain how it works here:

The main clock signal is generated by creating a 26-bit variable signal. Upon every rising edge of the clock, the variable is incremented by 1 bit. We give our divideBy3 clk signal the 26th bit of the variable so that every 26th rising edge of the internal clock, our circuit clock will go high. This divides the frequency of the internal clock allowing us to generate a slower clock signal.

Finally, a picture of the Basys3 board working:

