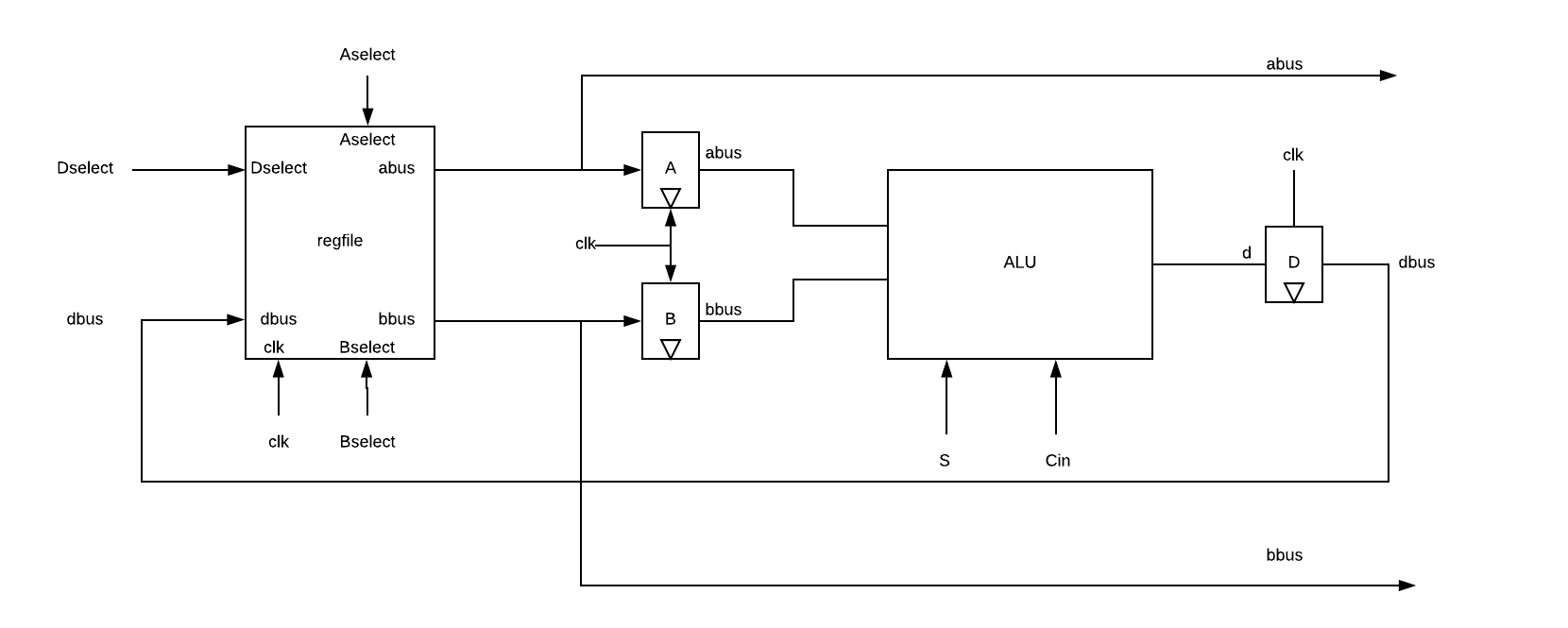
Arjun Gupta

EECE 3324

Lab 3

7/12/2018

1. Description
   1. The main purpose of this lab was to extend the pipelined ALU created in Lab 2 by adding a register file with 32, 32-bit registers. This pipelined ALU plus register file supports writeback from the output bus and represents a 3-stage pipeline. This is the most basic form of parallelism for our purposes. On each rising edge, the data in the output bus gets put into the specific register from the calculated ALU value, and is written on each falling edge. Therefore, there is always a task that needs to be done as long as the appropriate commands are queued up properly.
2. Block Diagram
   1. Below shows the block diagram of the pipelined ALU plus the register file added in Lab 3.



1. Steps needed to complete the Lab
   1. The first step needed to complete this lab was to implement 32, 32-bit registers in the form of a register file. The first register is meant to be a “ground” register, representing logic 0 for the rest of the registers. The registers were modeled as a D Flip-Flop with a trigger on the falling edge of the clock signal. When the correct Dselect signal was given, the data in the “dbus”, or output bus, would be pushed into the register. The output, Q, was used as an input to two tristate buffers going to the two input busses, abus and bbus. The trigger for these tristate buffers is the select lines for each respective input. When the specific register was selected, its value was pushed into the input bus. If a register was not selected, then the value of the bus is an “open circuit”. The structure of the D Flip-Flop and tristate buffer was encapsulated into one module and replicated 30 times. In order to properly represent logic 0, the output was always assigned to either 0 or “open circuit” if 0 register was not chosen. This register file was then connected to the pipelined ALU created in Lab 2.
2. Lessons Learned
   1. Throughout this lab, I realized how to construct a 3 stage pipeline, and learned some helpful Verilog tricks that prevented me from having to instantiate 30 different instances of the register and tri-state buffer module. By creating a vector of instances, my code length dramatically decreased. Navigating the testbench was also a little challenging and required me to look at the simulated output much more closely, which helped my skills at debugging.