Computer Engineering Program

College of Engineering and Computer Science

Fall 2020



Exercise 4: The Arbiter

EGCP 446 - 11/20/2020

# Prepared by: Levi Randall and James Samawi

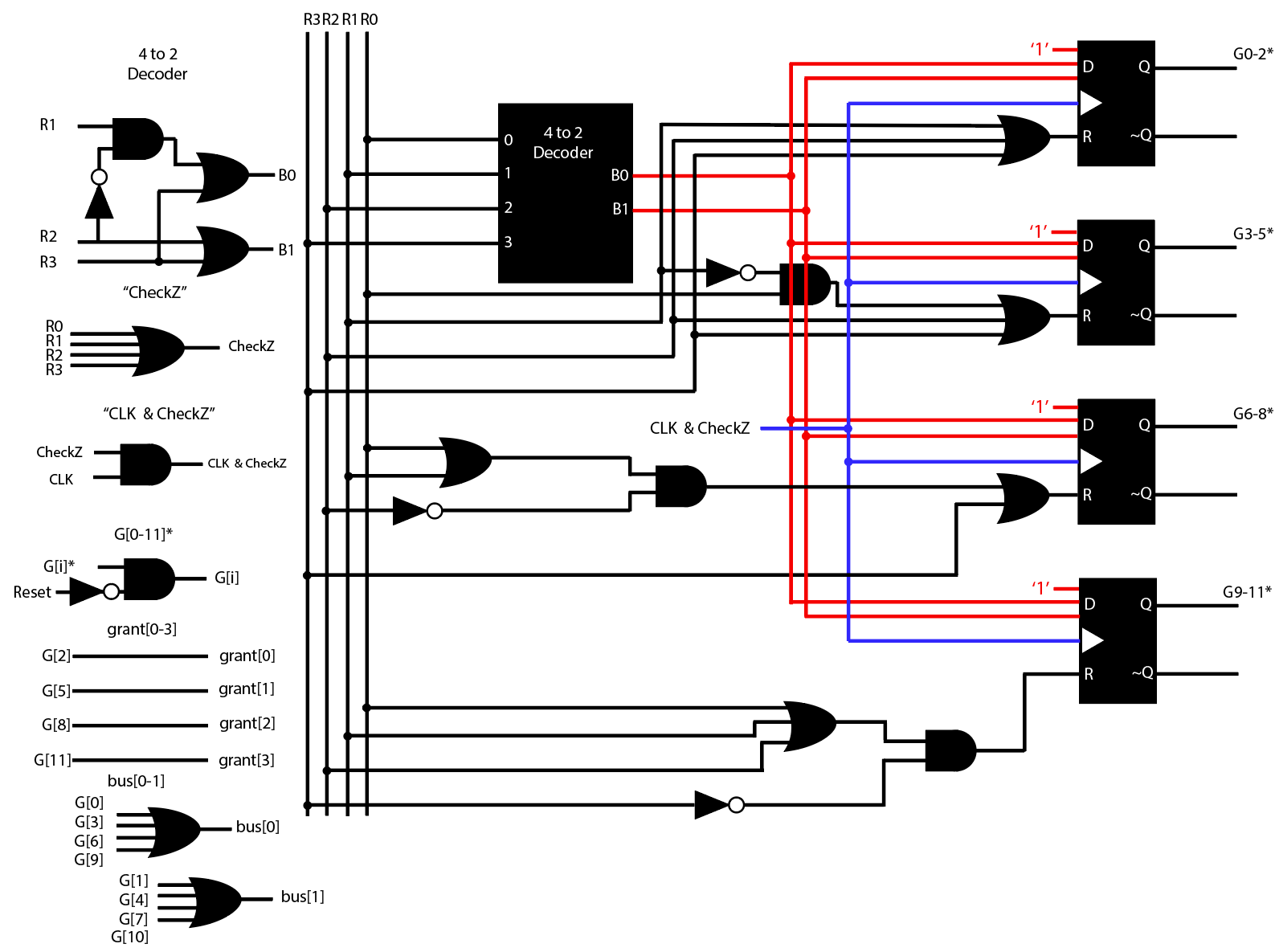
Abstract

Within many integrated circuits, bus systems are used in order to transmit data. These busses are very useful for transferring data and have the advantage of being one physical wire, but with many possible directions of transferring data. This in mind, the bus must be managed so two devices don’t try to transmit data at the same time. The hardware to do this is called an Arbiter and they manage request signals from various devices and grant access to the bus using a grant signal.

Summary/Theory

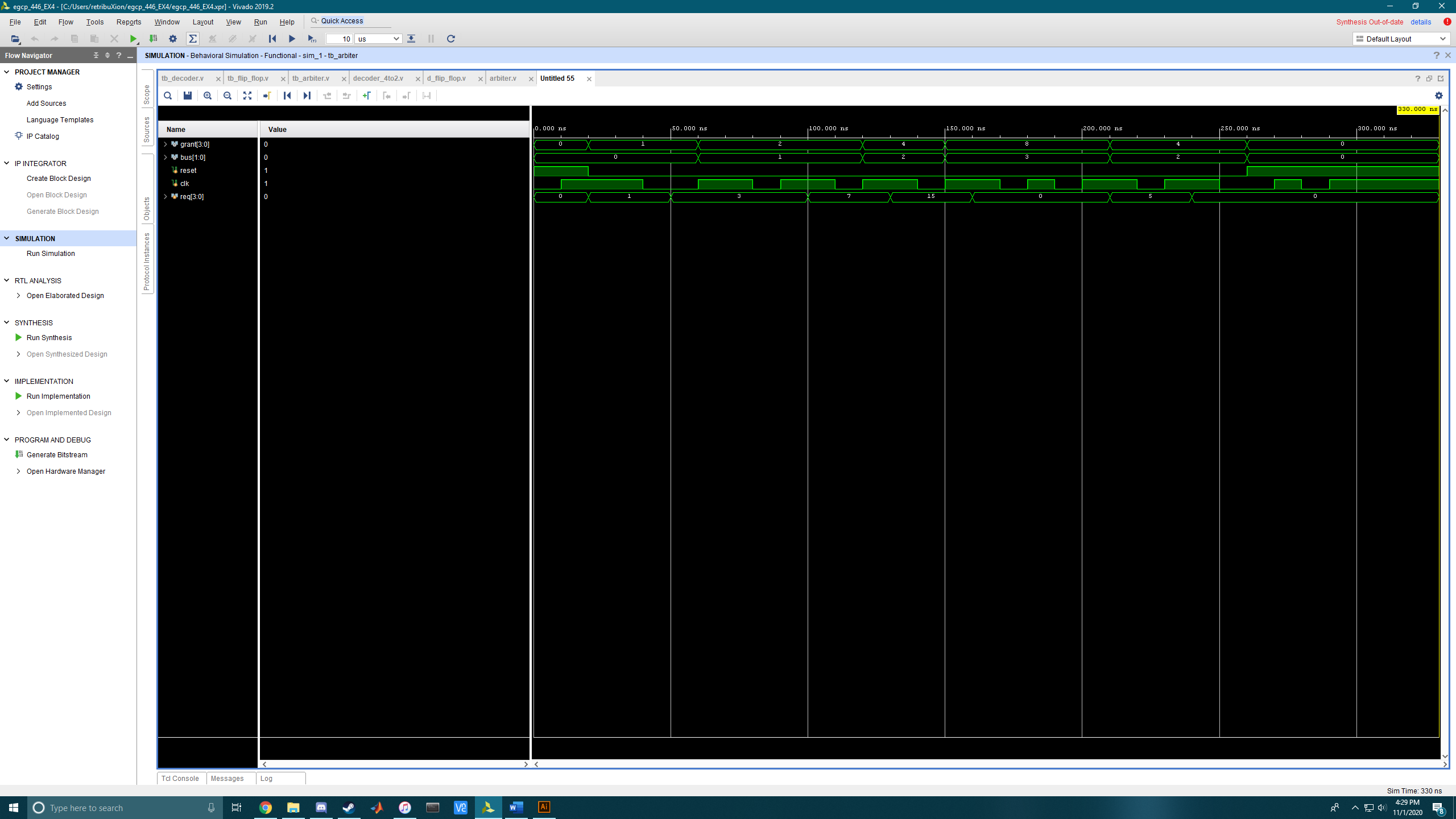
Our Arbiter must be created with specific constraints. It must use One-Hot State coding and has to be able to continue granting access to a device if no other signals are requested. It also has a reset that resets all grant values to 0 in the event of a reset. The arbiter takes 4 inputs and also issues a grant signal to a bus that outputs a value tied to the device granted. The arbiter also prefers the device with the higher number and will allow it signal over other devices. The arbiter starts with all outputs low and lastly, only one output can be high at one time.

To accomplish our task, I first created a behavioral of a flip flop with a synchronous reset and a 3-bit wide input. Of these three bits, the last was always high to represent a grant signal and the other two bits were going to be sent from a decoder that model the bus signal. This decoder has the preference for higher valued numbers and is made from two OR gates, an AND gate, and a single inverter. Depending on the highest bit, the output will relate to the highest valued number, so say R[3] and R[1] are both high, then the output will be 3 or in binary “11.” The main element of control in this circuit is done by abusing the reset of the flip flop. Each flip flop reset is tied to a circuit that also values higher valued numbers as well as contains logic to switch the reset off if the input correlating to this flip flop is not high and another input is high. With this control, we can allow certain flip flops to be on while others remain off. In order to make sure access is continuously granted when all other signals are low, we combined the clock signal with an OR gate that checks to see if all inputs are zero. If all inputs are zero, the clock signal is halted and the values coming from the flip flops are held static. To recreate a new reset because all the resets of the flip flops are used for managing the bus info, all outputs of each flip flop is put through an AND gate that is controlled by an inversion of the reset signal, meaning if the reset signal is high then the outputs of all flip flops will be zero. All of these ideas can be visualized in our model below,



RESULTS

After we had discovered all the ideas mentioned above, we were able to complete all requirements of the system given. The system can be forced to all low inputs using the reset signal and from there inputs can be inputted. The flip flops require that the input must be high or low before the clock’s positive edge in order to transmit info. So, we make sure our clock’s frequency is high enough to detect when an input is given. Moving forward, no matter the lower inputs the highest bit will always be chosen, and this is reflected in both the bus and grant signal. If no input is given at a time, then the last granted input will be continued. And lastly everything can be set low asynchronously if the reset signal is high.



Conclusion

This arbiter was far more challenging then either of us had considered. The concept of the arbiter is fairly simple but getting as much functionality as possible out of the flip flops was challenging and our design still is not perfect. Perhaps one of the greatest unforeseen challenges was the odd way that Verilog handles timing events. Learning that the signal must be high or low before the clock edge took a while to understand and implement around. We also had to alter the input, the output, the reset, and the clock of the flip flop with more gates to establish the correct functionality. Making sure the highest bit given was the output is another incredibly challenging part and that prompted our use of gates on the reset signal of the flip flops. If we were to do this lab again, we would model things more around the decoder to avoid the confusion when using the flip flops, but overall are happy with our results. Now, we know that sometimes simple objects can become much harder when asked for additional functionality.