Design of Round Robin Arbiter in Verilog

Anvay Pancholiya
Department of Electronics and Communication Engineering
RV College of Engineering
Bengaluru, India
anvayp.ec19@rvce.edu.in

***Abstract*—This report presents the design for a round robin arbiter that handles the resource access requests from 4 devices. It grants the access to the device in the next cycle if it is the turn of that device or no other device is requesting access.**

***Index Terms*—Arbitration, Priority Arbiter, Round Robin order**

I. Description

There are a number of resources in a computer system. They range from networks, storage devices and communicating devices to computer program. It is often the case where a resource is shared by a number of different devices and can be used by only one device at a time. The main objective of arbitration algorithms is to ensure that only one master has access to the bus at any given time, all the other masters are forced to remain in the idle state until they are granted the use of the bus.

II. Design specification

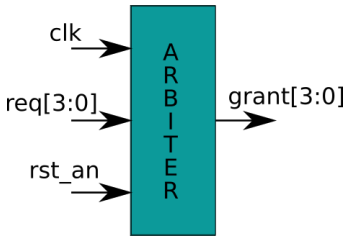

Fig. 1. Block diagram

The *rst_an* is asynchronous reset signal which is used to initialise the variables at the beginning. The 4 bit *req* bus is composed of the resource request lines of the 4 devices. The *grant* bus is composed of the grant status lines of each device. Only one bit in the grant bus can be HIGH at a time as the resource can be granted to only one device at a time. The design is clocked on the rising edges. *rotate_ptr* is the pointer used to move the access order in round robin fashion. *shift_req* is the signal after shifting the *req* bits so that the LSB of *shift_req* is the bit of *req* at index given by *rotate_ptr*. This is done because the LSB is given the highest priority in the simple priority arbiter used. It is used to give a higher priority to the device having the resource access opportunity in the round robin order at that moment over the other devices. *shift_grant* has the bit set to HIGH at index chosen by the simple priority arbiter. *grant_comb* shifts the bits in *shift_grant* based on the *rotate_ptr* value so that the bit set to HIGH comes back to the position that it was at before the *shift_req* step. Finally, *grant_comb* is masked with the 1's complement of current *grant* value to get the new *grant* value.

III. Simulation

The *rst_an* goes LOW at beginning to initialise the variables to 0. Device A requests access for the resource and the arbiter grants access to it in the next cycle. Device A also stops requesting the same cycle. The next cycle both Device A and Device B request access. However since the *rotate_ptr* already incremented, Device B gets the access. This is the round robin arbitration functionality expected out of the design. Now, Device C and Device A request access and the arbiter grants access to Device C in the next cycle. The design works similarly for the further inputs.

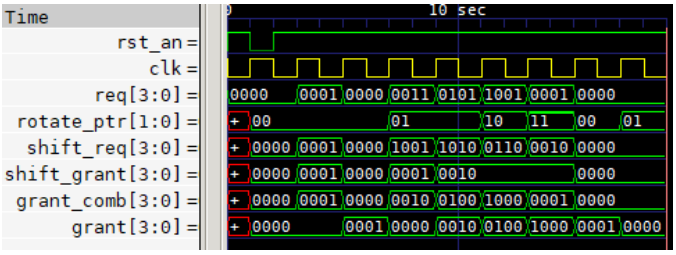

Fig. 2. Round Robin arbitration in action

References

[1] Dongjun, Luo. Round Robin Arbiter. Mar. 2010. *https://opencores.org/projects/round_robin_arbiter*
[2] Rudregowda, Shashidhar. (2014). Implementation of Bus Arbiter Using Round Robin Scheme. IJIRSET. 3297.