Design of An Arbiter on Resource Sharing using FPGA

Lay Nandar Soe¹, Nang Tee Shar²

¹Lay Nandar Soe & ²Nang Tee Shar
¹Lecturer, Department of Electronic Engineering, TU (Loikaw)
²Assistant Lecturer, Department of Electronic Engineering, TU (Loikaw)
Loikaw, Kayah State, Myanmar

Abstract: This paper deals with implementation of arbiter design using FPGA. In computer system, it is often useful to have resource shared by a number of different devices. Usually, the resource can be used by only one device at a time. When various devices need to use the resource, they have to request to do so. These requests are handled by an arbiter circuit. In this system, arbiter circuit is implemented using logic gates with asynchronous design. Arbiter can be employed anywhere a resource needs to be shared. Arbiter can be synchronous or asynchronous, and they work by inputting requests and granting access to resources based on those requests. This system implements asynchronous design solutions to priority arbitration. The system is based on FPGA and writes a code using VHDL text editor and implements the circuit on a device using Altera DE2 Development and Education board with Cyclone IV FPGA chip.

Keywords: Arbiter, FPGA, VHDL & Priority.

1. INTRODUCTION
Arbitration is an important part of any modern computer system. An arbiter is a device that takes as input requests, and outputs a single grant, in the form of a one-hot. A one-hot is a group of bits of arbitrary size consisting of all zeros except for one. In this way, the arbiter looks at its set of inputs and allows a single device access to the resource.
Any devices are competing for the shared resource. Each device communicates with the arbiter by means of two signals, Request and Grant. When a device needs to use the shared resource, it raises its request signal to 1. Then it waits until the arbiter responds with the grant signal. A device initiates the activity by raising a request, \( r = 1 \). When the shared resource is available, the arbiter responds by issuing a grant, \( g = 1 \). When the device receives the grant signal, it proceeds to use the requested shared resource. When it completes its use of the resource, it drops its request by setting \( r = 0 \). When the arbiter sees that \( r = 0 \), it deactivates the grant signal, making \( g = 0 \). The arrows in the figure indicate the cause-effect relationships in this signaling scheme; a change in one signal causes a change in the other signal. The time elapsed between the changes in the cause-effect signals depends on the specific implementation of the circuit. A key point is that there is no need for a synchronizing clock.

2. DESIGN CONSIDERATION OF AN ARBITER

In computer system, it is often useful to have resource shared by a number of different devices. Usually, the resource can be used by only one device at a time. When various devices need to use the resource, they have to request to do so. These requests are handled by an arbiter circuit. When there are two or more outstanding requests, the arbiter may use some priority scheme to choose one of them. In this system, three devices share one resource using arbiter circuit. So, arbiter has three inputs, request signal \( r_1, r_2, r_3 \) and three outputs, grant signal \( g_1, g_2, g_3 \).

Figure (2) State Diagram of the arbiter

Our design is implemented using priority. The priority of \( r_2 \) is higher than \( r_3 \). The priority of \( r_1 \) is higher than \( r_2 \). So, \( r_1 \) is the highest priority. When both request 1 and request 2 are asserted then grant 1 is asserted, otherwise when both request 2 and request 3 are asserted then grant 2 is asserted.

Table 1. Excitation table of the arbiter

<table>
<thead>
<tr>
<th>Present state y3y2y1</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next State</td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>001</td>
<td>100</td>
<td>001</td>
<td>100</td>
<td>001</td>
</tr>
<tr>
<td>Output g3g2g1</td>
<td>000</td>
<td>001</td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>001</td>
<td>010</td>
<td>001</td>
</tr>
</tbody>
</table>

29
From the excitation table, the following next-state and output expressions are derived.

\[
Y_1 = r_1 \\
Y_2 = \text{NOT} \ r_1 \ \text{AND} \ r_2 \\
Y_3 = \text{NOT} \ r_1 \ \text{AND} \ \text{NOT} \ r_2 \ \text{AND} \ r_3 \\
g_1 = y_1 \\
g_2 = y_2 \\
g_3 = y_3
\]

Figure (3). Circuit Diagram of Arbiter Design

A. Introduction to CAD Tools
Logic circuits found in complex systems, such as today’s computers, cannot be designed manually; they are designed using sophisticated CAD tools that automatically implemented the synthesis techniques. To design a logic circuit, a number of CAD tools are needed. They are usually packaged together into a CAD system, which typically includes tools for the following tasks: design entry, synthesis and optimization, simulation, and physical design.

B. Representation of Digital Signals in VHDL
When using CAD tools to synthesis a logic circuit, the designer can provide the initial description of the circuit in several different ways. One efficient way is to write this description in form of VHDL source code. The controller was implemented using VHDL. In VHDL, a design consists at a minimum of an entity which describes the interface and an architecture which contains the actual implementation. In addition, most designs import library modules. The VHDL code of our design is as shown below:

```
library ieee;
use ieee.std_logic_1164.all;
entity arbiter is
  port( 
    r1, r2, r3: IN std_logic;
  );
architecture logic of arbiter is
begin
  g1<= r1;
  g2<= (r2 and not r1);
  g3<= (not r1 and not r2 and r3);
end logic;
```

Figure (4) VHDL code for arbiter

3. TESTS AND RESULT
Arbiter is implemented in VHDL, Quartus II software is used. Other tools like Xilinx ISE, Active-HDL would work, as well. Quartus II is a sophisticated CAD system. As most commercial
CAD tools are continuously being improved and updated, Quartus II has gone through a number of releases. In this system, Quartus II 10 version of the software is used.

![Simulation results with three requests(r1, r2, r3) and output (g1)](image)

![Simulation results with two requests (r2, r3) and output (g2)](image)

![Simulation results with one request (r3) and output (g3)](image)

Figure 8: Implement the design on Altera DE2 development and education board with Cyclone IV FPGA chip
4. CONCLUSIONS

In an embedded world, resources are always limited. An arbiter can be controlled and add priority to competing subsystems, all while increasing system performance. The system introduced the arbiter, with an implementation of a simple priority arbiter in VHDL. This system can be used to control the three CPU cores that need to access shared memory, or three microcontrollers can be controlled to control the use of a communication bus. The system has been designed using VHDL, and implemented on hardware using Altera DE2 development and education board with Cyclone IV FPGA chip. By this paper, Fundamentals of Digital Logic with VHDL Design subject which was teaching for fourth year (EcE) Students can be improved for Practical approach teaching.

5. ACKNOWLEDGMENT

The authors would like to express their thanks to all the members of Board of Study of TU (Loikaw).

6. REFERENCES