KU_logo KU_soeng



**The University of Kansas**

**School of Engineering**

**Department of Electrical Engineering and Computer Science**

EECS 645 – Computer Architecture

Fall 2015

Homework 02 (Resource Sharing)

Student Name: Dustin Horvath Student ID: 2729265

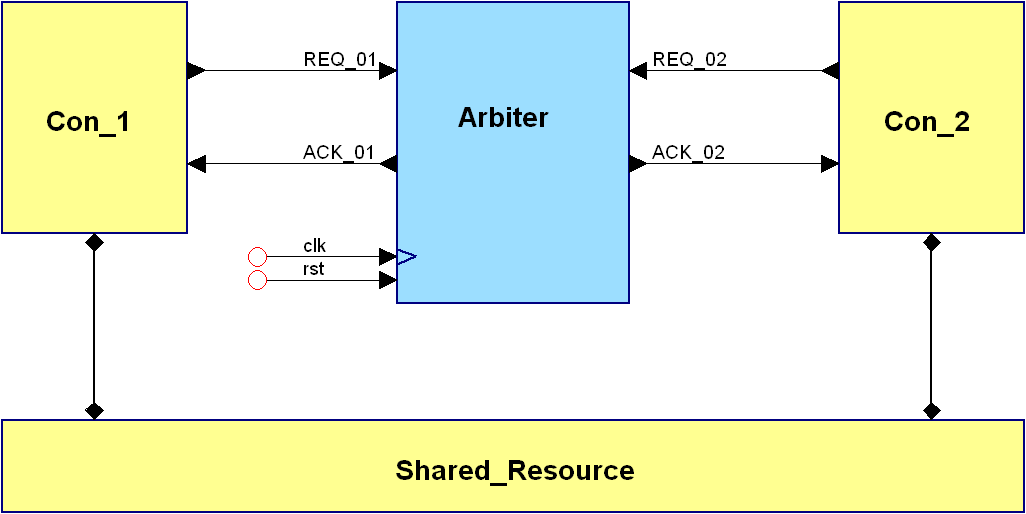
**Lecture Example (Template Answer):**

Given a resource that is to be shared by two consumers such that only one consumer has access to the resource at any given time. The policy of access is non-preemptive with no priority. More specifically the policy is as follows:

* When the resource is idle and the consumers simultaneously request the resource, their requests are ignored until only one request is submitted
* When the resource is being accessed by any of the consumers, the consumers are scheduled as First-Come-First-Served (FCFS)

Design a two-consumer arbiter/controller that controls the access to the shared resource and implements the above policy. In the design process, provide the following:

1. The system architecture (block diagram) showing the interface ports to the arbiter including the clock and reset signals.
2. Finite State Machine (FSM) diagram showing all possible states, transitions, and output values.
3. K-maps for internal state and output variables.
4. Boolean expressions for internal state and output variables.
5. Detailed logic diagram using synchronous memory elements showing internal connections and external interfaces.
6. Complete description of the arbiter using VHDL code.
7. Complete VHDL testbench along with simulation results.

****

**Figure 1: System architecture**

**Table 1: Input Code Assignment**

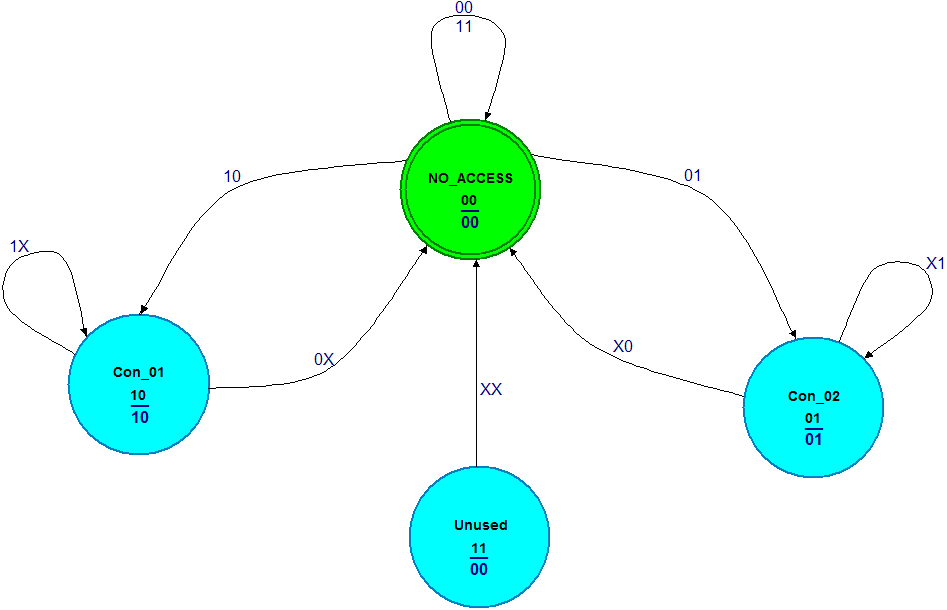
|  |  |  |
| --- | --- | --- |
| Input Combinations (Input Codes) | | Input Description |
| REQ\_01 | REQ\_02 |
| 0 | 0 | No requests |
| 0 | 1 | Consumer 2 requests resource |
| 1 | 0 | Consumer 1 requests resource |
| 1 | 1 | Both consumers request resource |

**Table 2: Output Code Assignment**

|  |  |  |
| --- | --- | --- |
| Output Combinations (Output Codes) | | Output Description |
| ACK\_01 | ACK\_02 |
| 0 | 0 | None granted access to resource |
| 0 | 1 | Consumer 2 granted access to resource |
| 1 | 0 | Consumer 1 granted access to resource |
| 1 | 1 | Forbidden output |

**Table 3: State Code Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| State Description | | State Codes | |
| S\_01 | S\_02 |
| Required States | Resource is idle  (No access) | 0 | 0 |
| Resource is used by consumer 2  (Con\_02) | 0 | 1 |
| Resource is used by consumer 1  (Con\_01) | 1 | 0 |
| Unused States | Extra unused state  (Unused) | 1 | 1 |

****

**Figure 2: State transition diagram**

**Table 4: State transition table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1 , S2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** | **00** | **01** | **00** | **10** |
| **01** | **00** | **01** | **01** | **00** |
| **11** | **00** | **00** | **00** | **00** |
| **10** | **00** | **00** | **10** | **10** |

**Table 5: K-map for the state variable *S1***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  | **1** |
| **01** |  |  |  |  |
| **11** |  |  |  |  |
| **10** |  |  | **1** | **1** |



**Table 6: K-map for the state variable *S2***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  | **1** |  |  |
| **01** |  | **1** | **1** |  |
| **11** |  |  |  |  |
| **10** |  |  |  |  |



**Table 7: Output transition table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1 , ACK2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** | **00** | **00** | **00** | **00** |
| **01** | **01** | **01** | **01** | **01** |
| **11** | **00** | **00** | **00** | **00** |
| **10** | **10** | **10** | **10** | **10** |

**Table 8: K-map for the output variable ACK1**

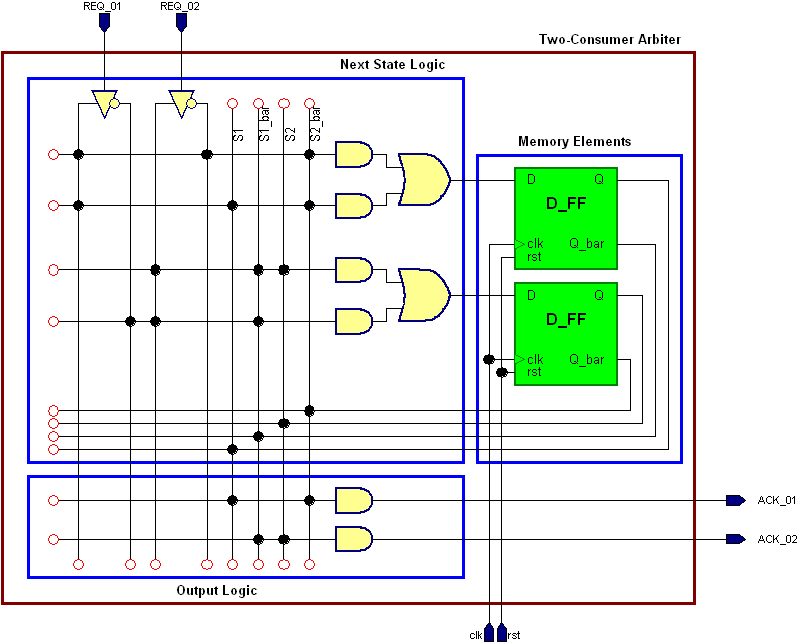
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |
| **01** |  |  |  |  |
| **11** |  |  |  |  |
| **10** | **1** | **1** | **1** | **1** |



**Table 9: K-map for the output variable ACK2**

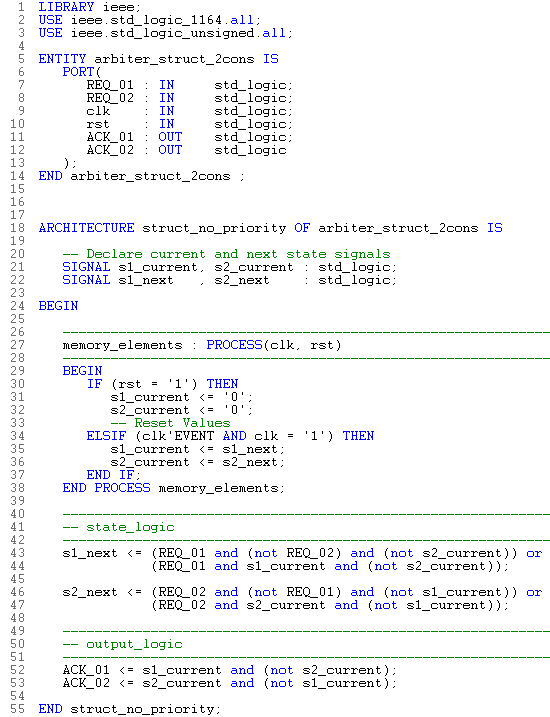
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |
| **01** | **1** | **1** | **1** | **1** |
| **11** |  |  |  |  |
| **10** |  |  |  |  |



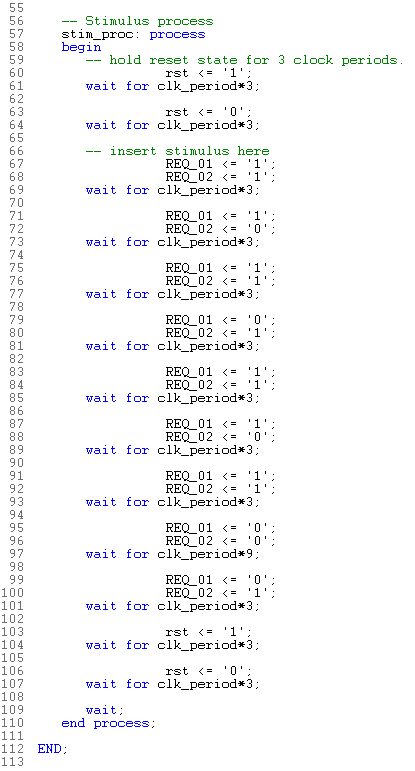
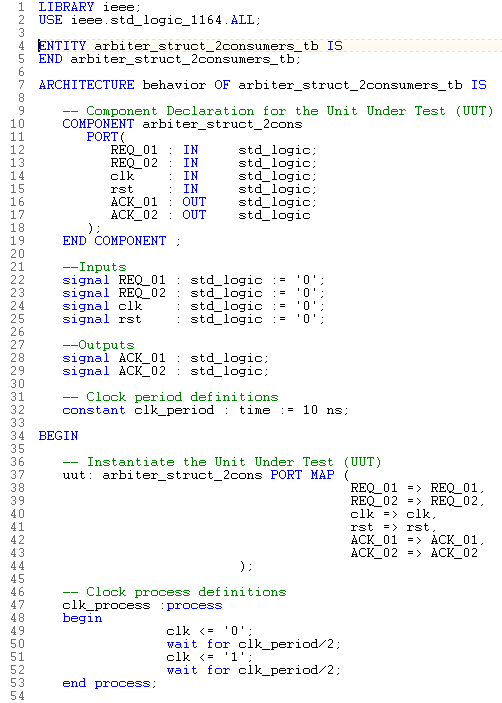


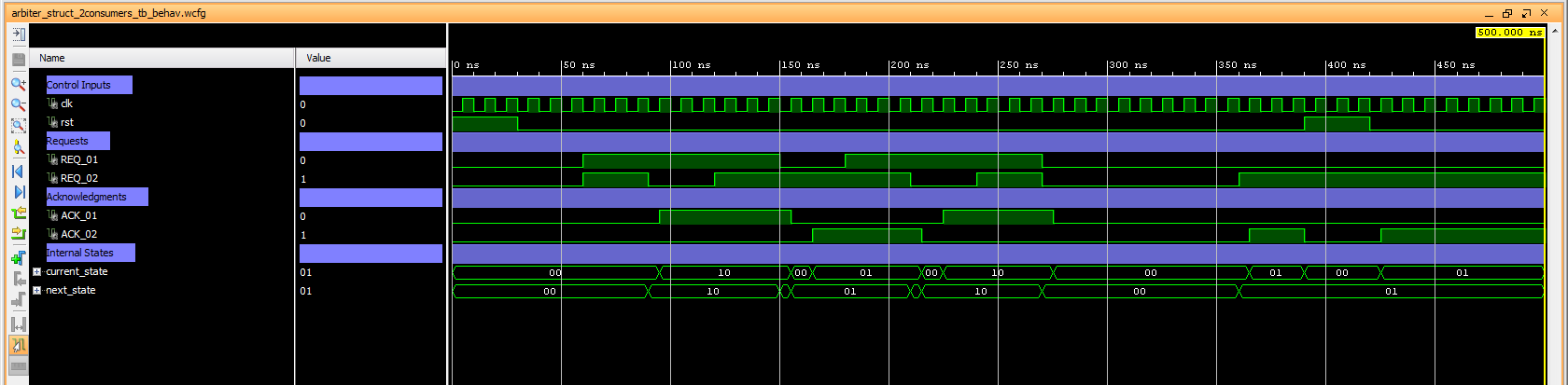
**Figure 3: Synchronous logic diagram**

**Structural VHDL code**

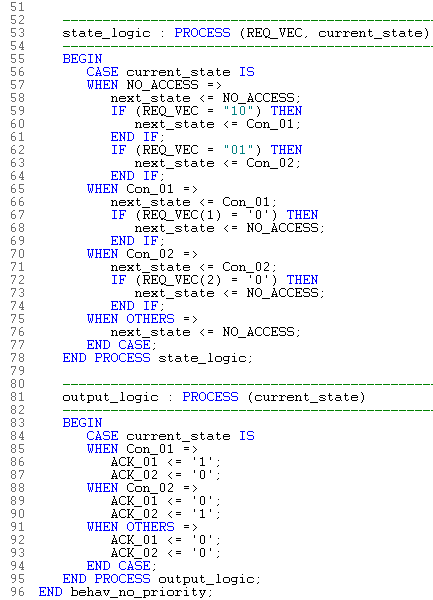
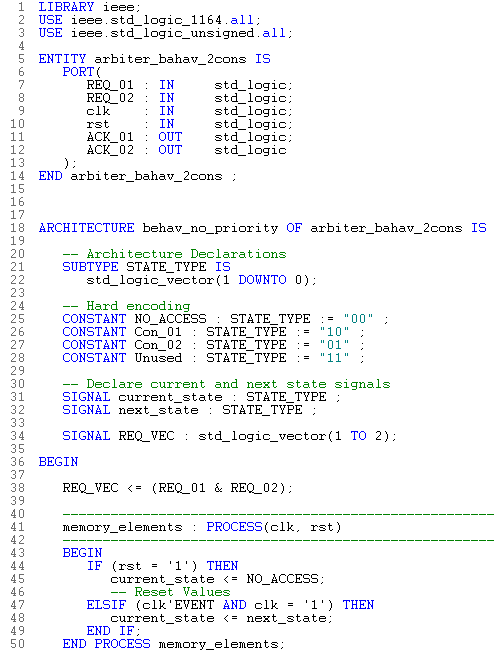
****

**Testbench and Simulation Results**

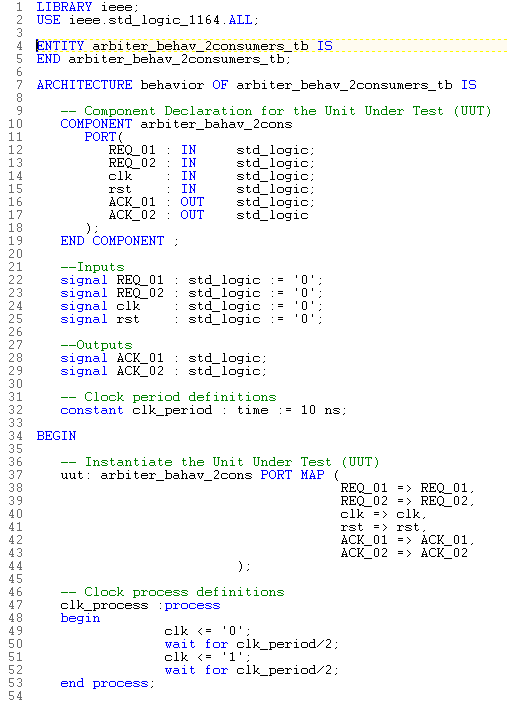
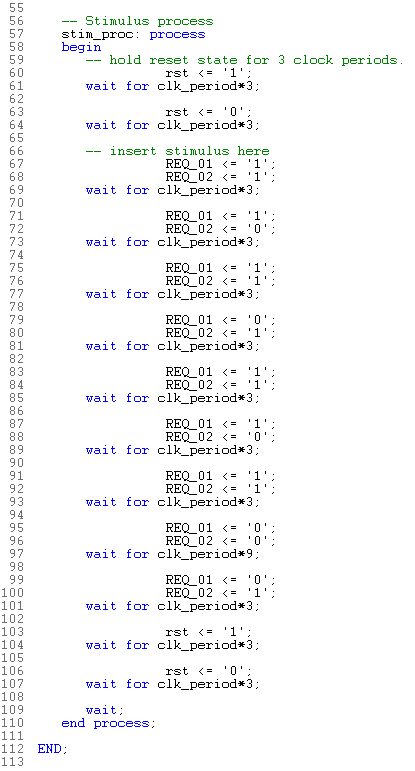


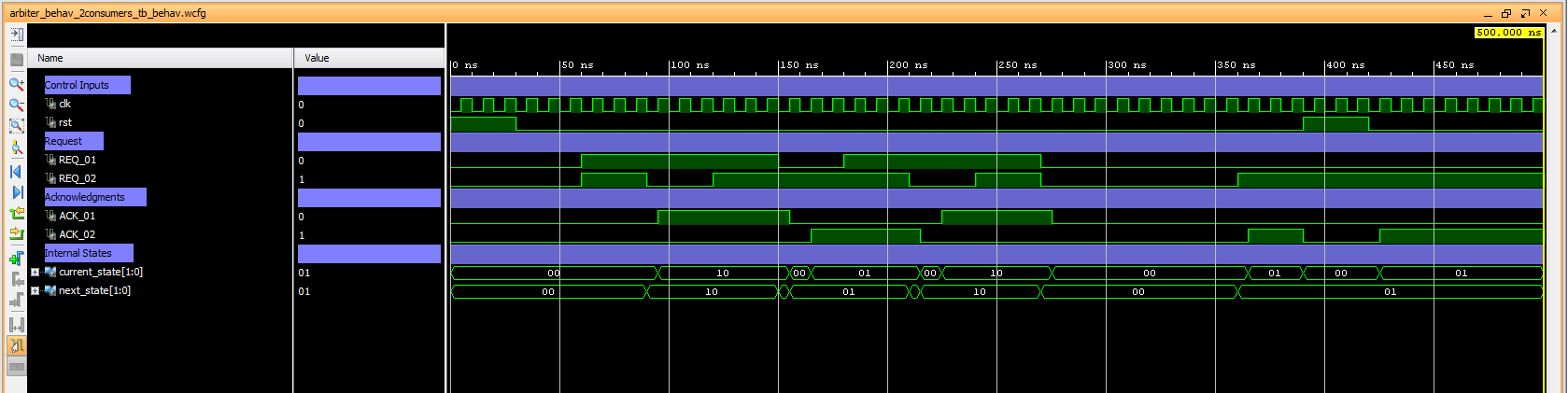
****

**Behavioral VHDL code**

****

**Testbench and Simulation Results**

****

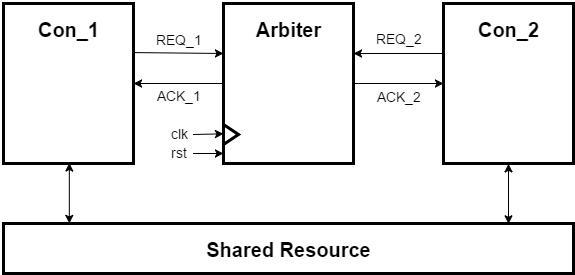
**Homework Problem I:**

Given a resource that is to be shared by two consumers such that only one consumer has access to the resource at any given time. The policy of access is non-preemptive with priority determined by the consumer index. More specifically the policy is as follows:

* When the resource is idle and the consumers simultaneously request the resource, the consumers are prioritized according to their consumer number (i.e. consumer 2 has a higher priority than consumer 1)
* When the resource is being accessed by any of the consumers, the consumers are scheduled as First-Come-First-Served (FCFS)

Design a two-consumer arbiter/controller that controls the access to the shared resource and implements the above policy. In the design process, provide the following:

1. The system architecture (block diagram) showing the interface ports to the arbiter including the clock and reset signals.
2. Finite State Machine (FSM) diagram showing all possible states, transitions, and output values.
3. K-maps for internal state and output variables.
4. Boolean expressions for internal state and output variables.
5. Detailed logic diagram using synchronous memory elements showing internal connections and external interfaces.
6. Complete description of the arbiter using VHDL code.
7. Complete VHDL testbench along with simulation results.

****

**Figure 1: System architecture**

**Table 1: Input Code Assignment**

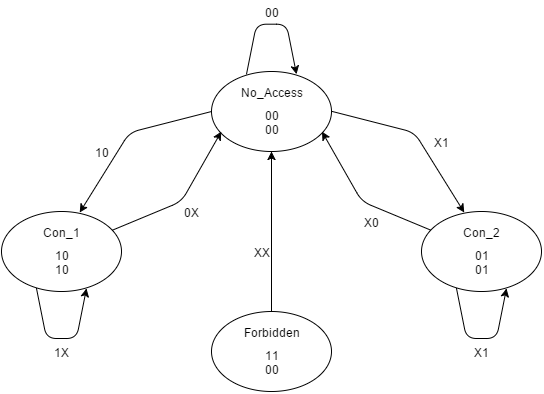
|  |  |  |
| --- | --- | --- |
| Input Combinations (Input Codes) | | Input Description |
| REQ\_01 | REQ\_02 |
| 0 | 0 | No Requests |
| 0 | 1 | Consumer 2 requests resource |
| 1 | 0 | Consumer 1 requests resource |
| 1 | 1 | Both consumers request resource |

**Table 2: Output Code Assignment**

|  |  |  |
| --- | --- | --- |
| Output Combinations (Output Codes) | | Output Description |
| ACK\_01 | ACK\_02 |
| 0 | 0 | None granted access to resource |
| 0 | 1 | Consumer 2 granted access to resource |
| 1 | 0 | Consumer 1 granted access to resource |
| 1 | 1 | Forbidden output |

**Table 3: State Code Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| State Description | | State Codes | |
| S\_01 | S\_02 |
| Required States | Resource is idle  (No access) | 0 | 0 |
| Resource is used by consumer 2  (Con\_02) | 0 | 1 |
| Resource is used by consumer 1  (Con\_01) | 1 | 0 |
| Unused States | Extra unused state  (Unused) | 1 | 1 |

****

**Figure 2: State transition diagram**

**Table 4: State transition table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1 , S2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** | **00** | **01** | **01** | **10** |
| **01** | **00** | **01** | **01** | **00** |
| **11** | **00** | **00** | **00** | **00** |
| **10** | **00** | **00** | **10** | **10** |

**Table 5: K-map for the state variable *S1***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1 )** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  | **1** |
| **01** |  |  |  |  |
| **11** |  |  |  |  |
| **10** |  |  | **1** | **1** |



**Table 6: K-map for the state variable *S2***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  | **1** | **1** |  |
| **01** |  | **1** | **1** |  |
| **11** |  |  |  |  |
| **10** |  |  |  |  |



**Table 7: Output transition table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1 , ACK2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** | 00 | 00 | 00 | 00 |
| **01** | 01 | 01 | 01 | 01 |
| **11** | 00 | 00 | 00 | 00 |
| **10** | 10 | 10 | 10 | 10 |

**Table 8: K-map for the output variable ACK1**

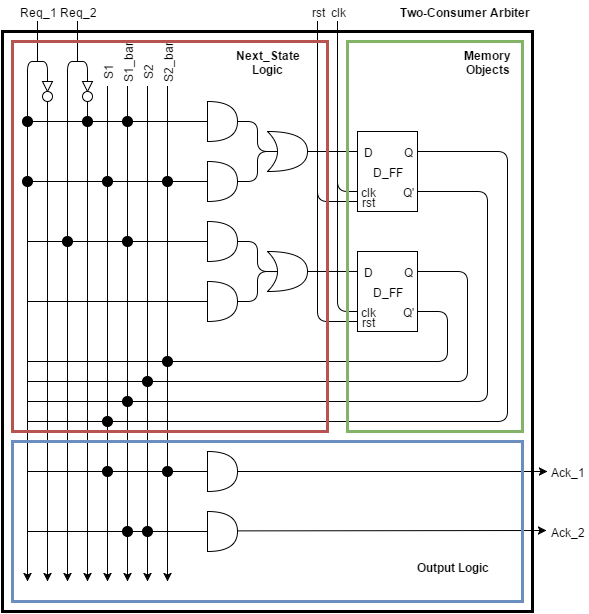
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |
| **01** |  |  |  |  |
| **11** |  |  |  |  |
| **10** | 1 | 1 | 1 | 1 |



**Table 9: K-map for the output variable ACK2**

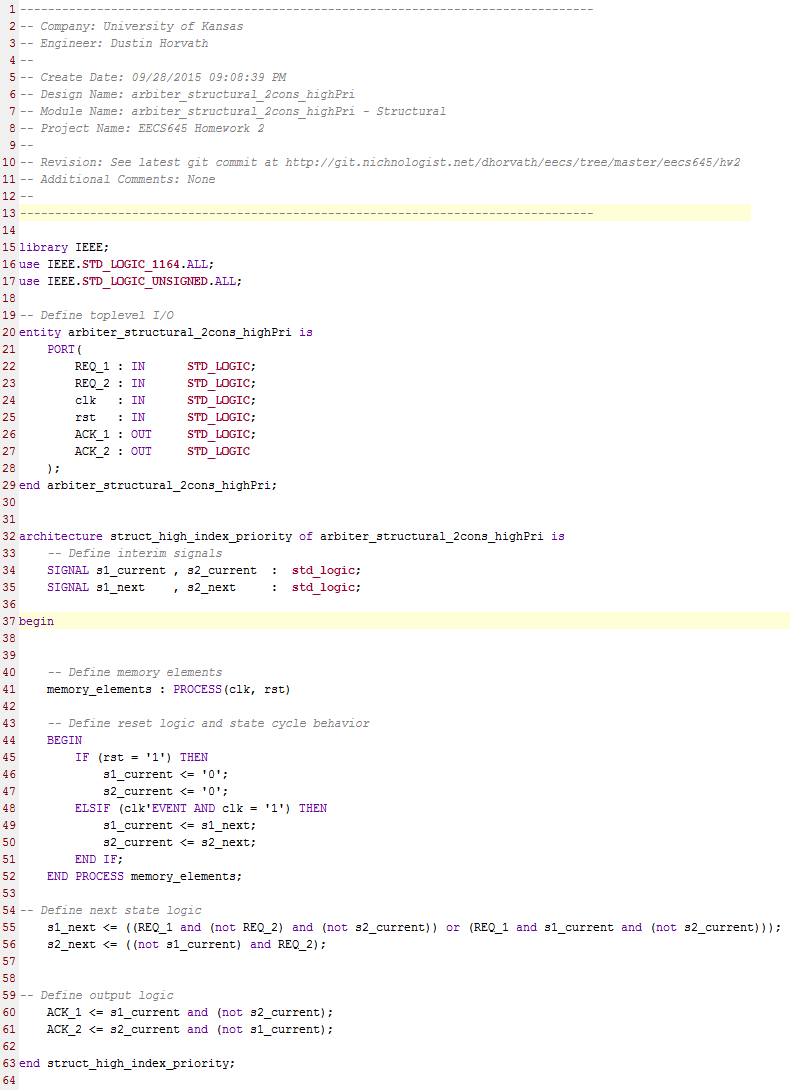
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |
| **01** | 1 | 1 | 1 | 1 |
| **11** |  |  |  |  |
| **10** |  |  |  |  |



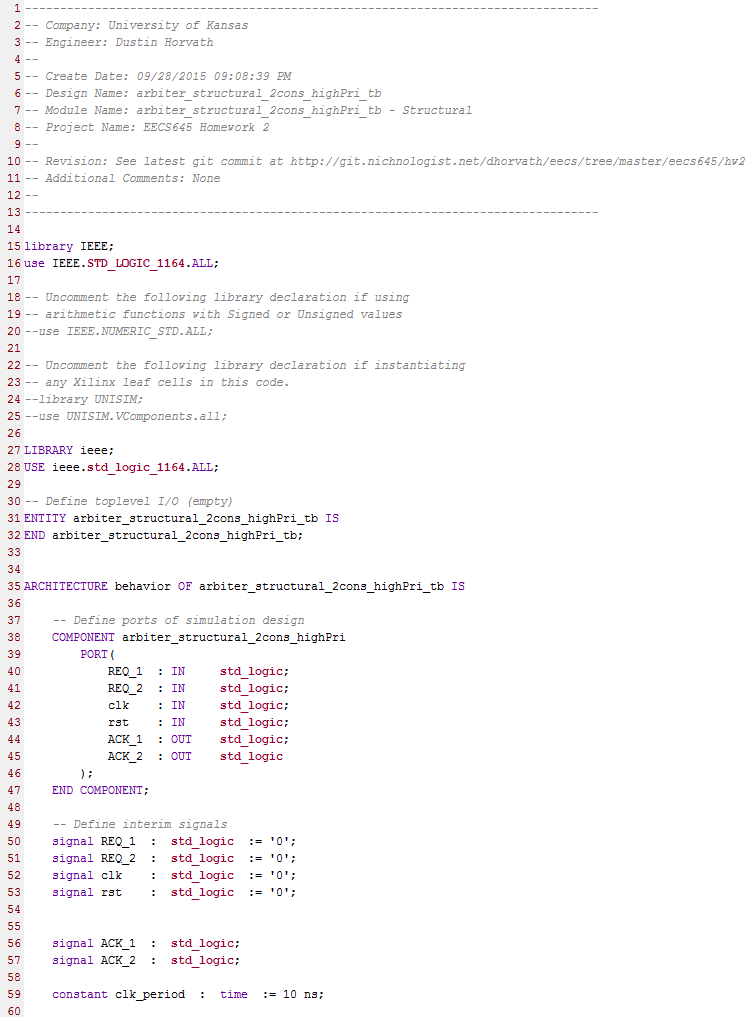
****

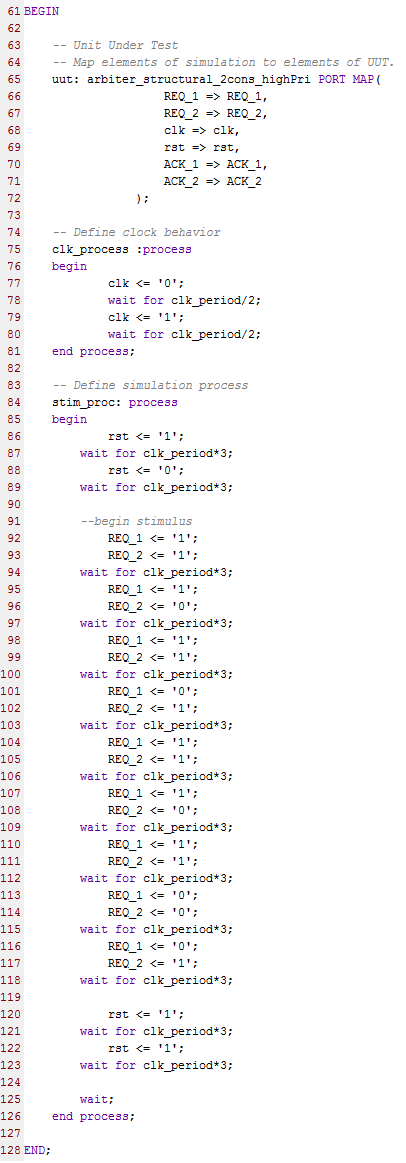
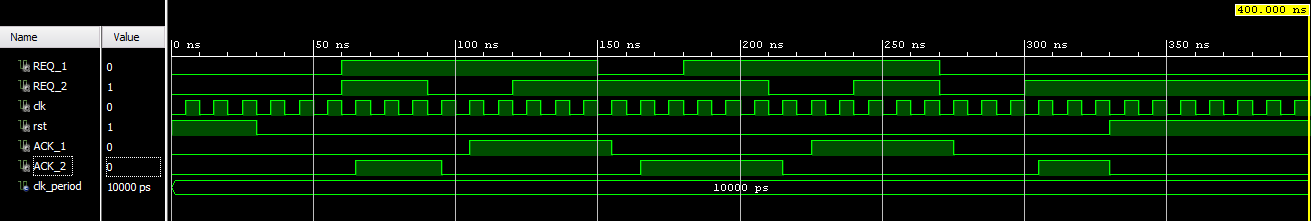
**Figure 3: Synchronous logic diagram**

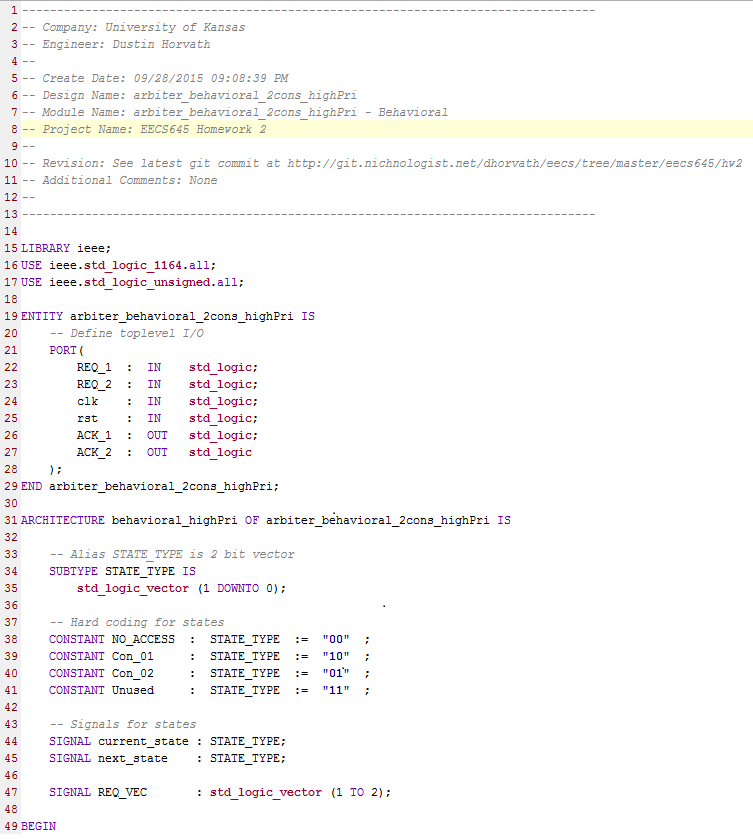
**Structural VHDL code**

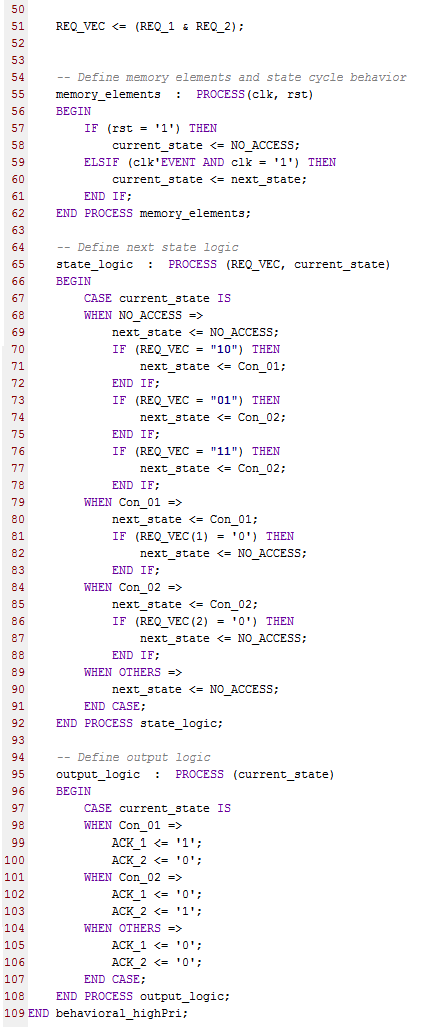
****

**Testbench and Simulation Results**

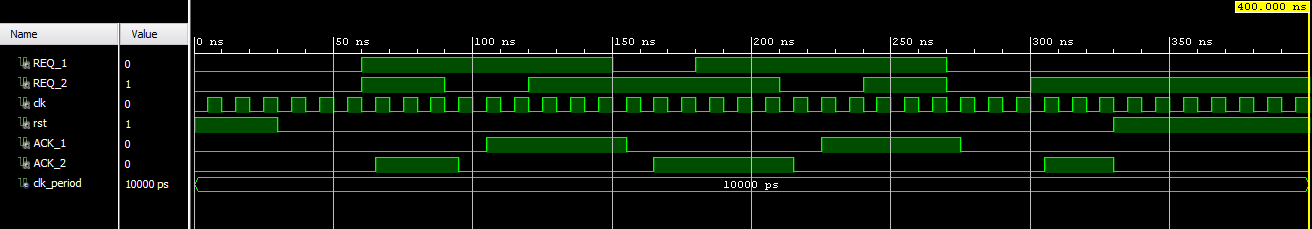
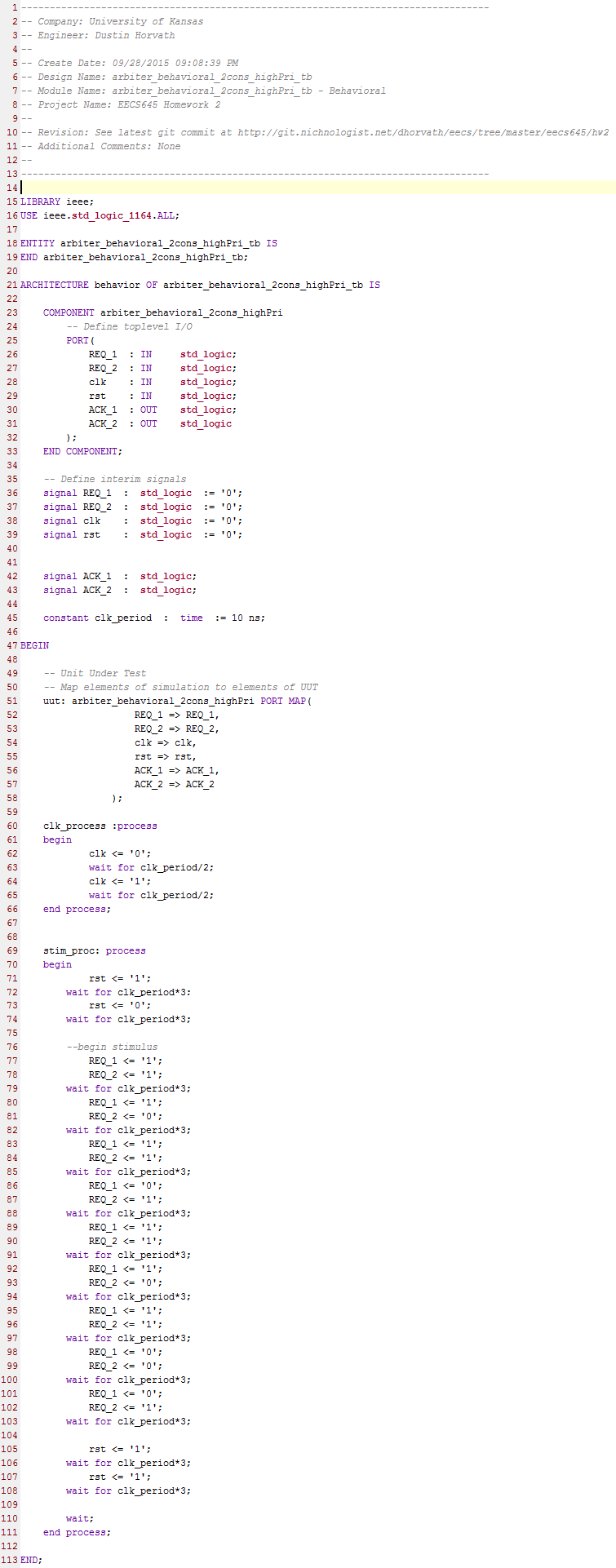
****

** Behavioral VHDL code**

****

****

**Testbench and Simulation Results**

****

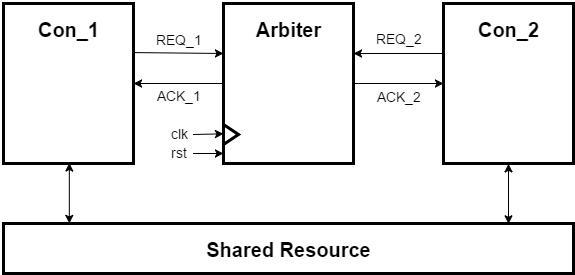
**Homework Problem II:**

Given a resource that is to be shared by two consumers such that only one consumer has access to the resource at any given time. The policy of access is non-preemptive with priority determined by the consumer index. More specifically the policy is as follows:

* When the resource is idle and the consumers simultaneously request the resource, the consumers are prioritized according to their consumer number (i.e. consumer 1 has a higher priority than consumer 2)
* When the resource is being accessed by any of the consumers, the consumers are scheduled as First-Come-First-Served (FCFS)

Design a two-consumer arbiter/controller that controls the access to the shared resource and implements the above policy. In the design process, provide the following:

1. The system architecture (block diagram) showing the interface ports to the arbiter including the clock and reset signals.
2. Finite State Machine (FSM) diagram showing all possible states, transitions, and output values.
3. K-maps for internal state and output variables.
4. Boolean expressions for internal state and output variables.
5. Detailed logic diagram using synchronous memory elements showing internal connections and external interfaces.
6. Complete description of the arbiter using VHDL code.
7. Complete VHDL testbench along with simulation results.

****

**Figure 1: System architecture**

**Table 1: Input Code Assignment**

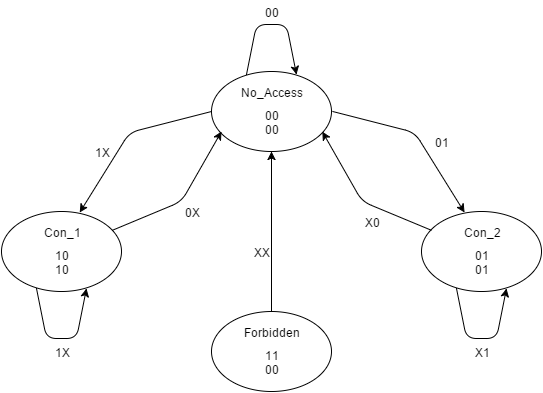
|  |  |  |
| --- | --- | --- |
| Input Combinations (Input Codes) | | Input Description |
| REQ\_01 | REQ\_02 |
| 0 | 0 | No Requests |
| 0 | 1 | Consumer 2 requests resource |
| 1 | 0 | Consumer 1 requests resource |
| 1 | 1 | Both consumers request resource |

**Table 2: Output Code Assignment**

|  |  |  |
| --- | --- | --- |
| Output Combinations (Output Codes) | | Output Description |
| ACK\_01 | ACK\_02 |
| 0 | 0 | None granted access to resource |
| 0 | 1 | Consumer 2 granted access to resource |
| 1 | 0 | Consumer 1 granted access to resource |
| 1 | 1 | Forbidden output |

**Table 3: State Code Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| State Description | | State Codes | |
| S\_01 | S\_02 |
| Required States | Resource is idle  (No access) | 0 | 0 |
| Resource is used by consumer 2  (Con\_02) | 0 | 1 |
| Resource is used by consumer 1  (Con\_01) | 1 | 0 |
| Unused States | Extra unused state  (Unused) | 1 | 1 |

****

**Figure 2: State transition diagram**

**Table 4: State transition table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1 , S2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** | **00** | **01** | **10** | **10** |
| **01** | **00** | **01** | **01** | **00** |
| **11** | **00** | **00** | **00** | **00** |
| **10** | **00** | **00** | **10** | **10** |

**Table 5: K-map for the state variable *S1***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1 )** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  | **1** | **1** |
| **01** |  |  |  |  |
| **11** |  |  |  |  |
| **10** |  |  | **1** | **1** |



**Table 6: K-map for the state variable *S2***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Next State**  **(S2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  | **1** |  |  |
| **01** |  | **1** | **1** |  |
| **11** |  |  |  |  |
| **10** |  |  |  |  |



**Table 7: Output transition table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1 , ACK2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** | **00** | **00** | **00** | **00** |
| **01** | **01** | **01** | **01** | **01** |
| **11** | **11** | **11** | **11** | **11** |
| **10** | **10** | **10** | **10** | **10** |

**Table 8: K-map for the output variable ACK1**

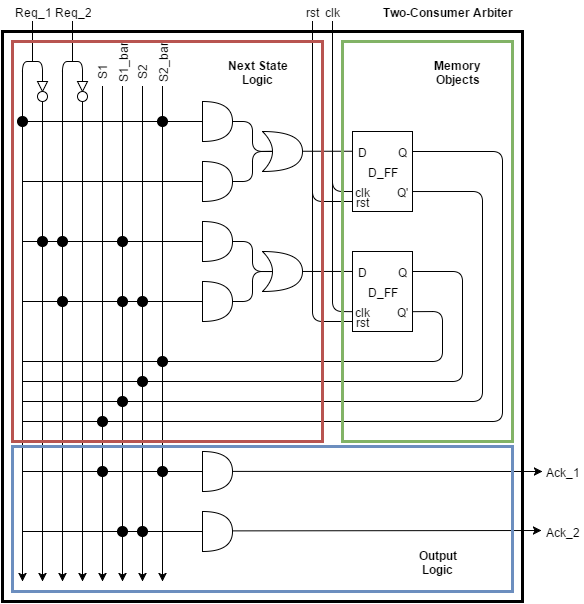
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |
| **01** |  |  |  |  |
| **11** |  |  |  |  |
| **10** | **1** | **1** | **1** | **1** |



**Table 9: K-map for the output variable ACK2**

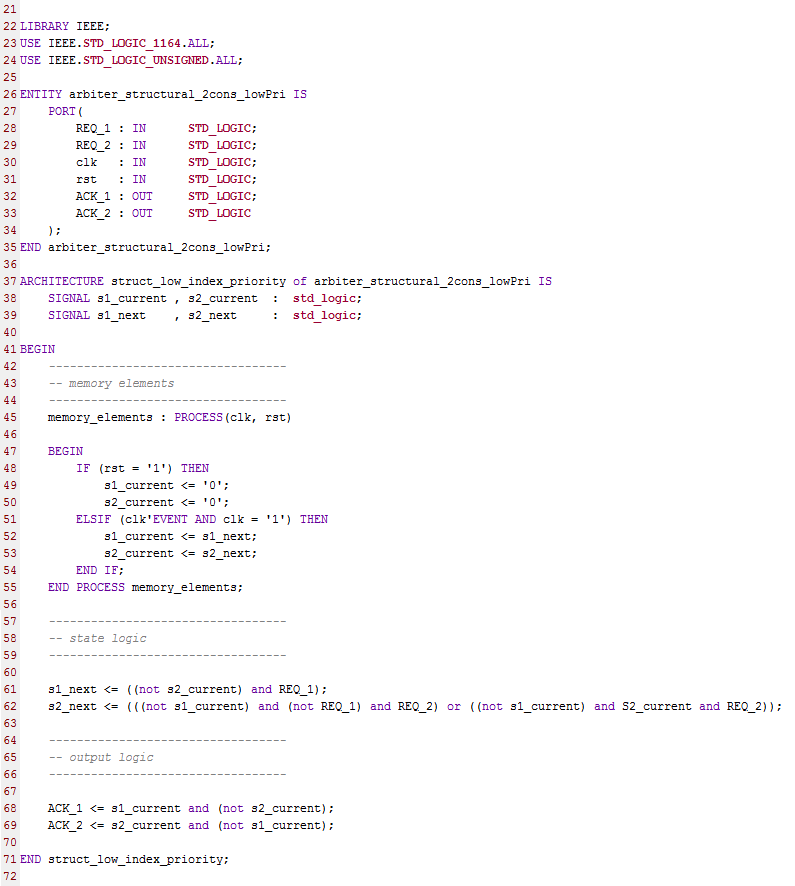
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK2)** | | **Inputs**  **(REQ1 , REQ2)** | | | |
| **00** | **01** | **11** | **10** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |
| **01** | **1** | **1** | **1** | **1** |
| **11** |  |  |  |  |
| **10** |  |  |  |  |



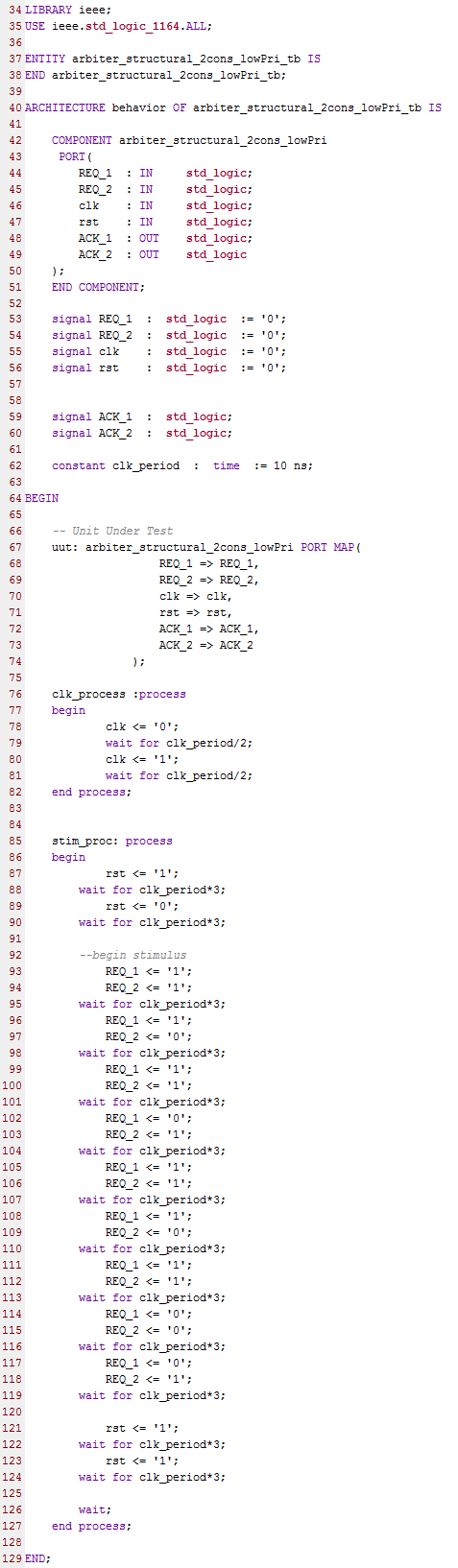
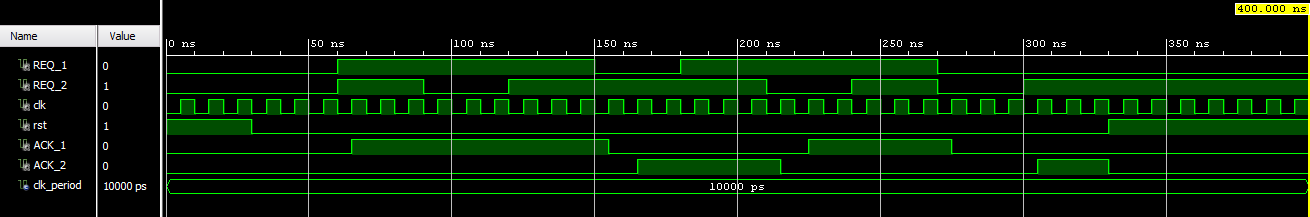
****

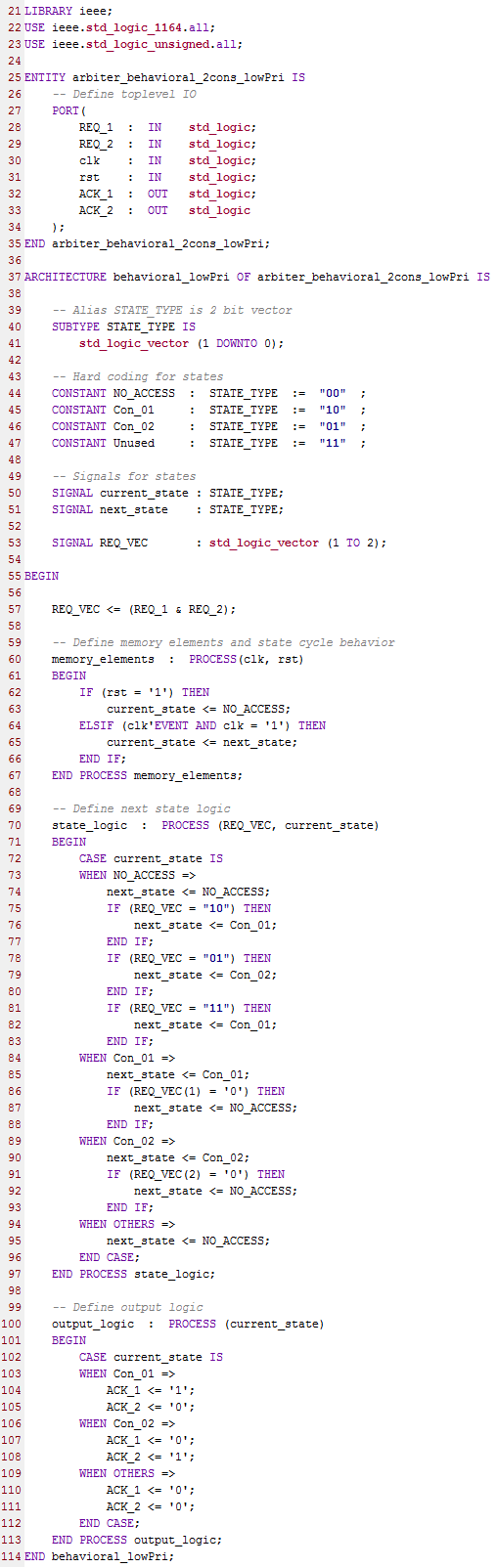
**Figure 3: Synchronous logic diagram**

**Structural VHDL code**

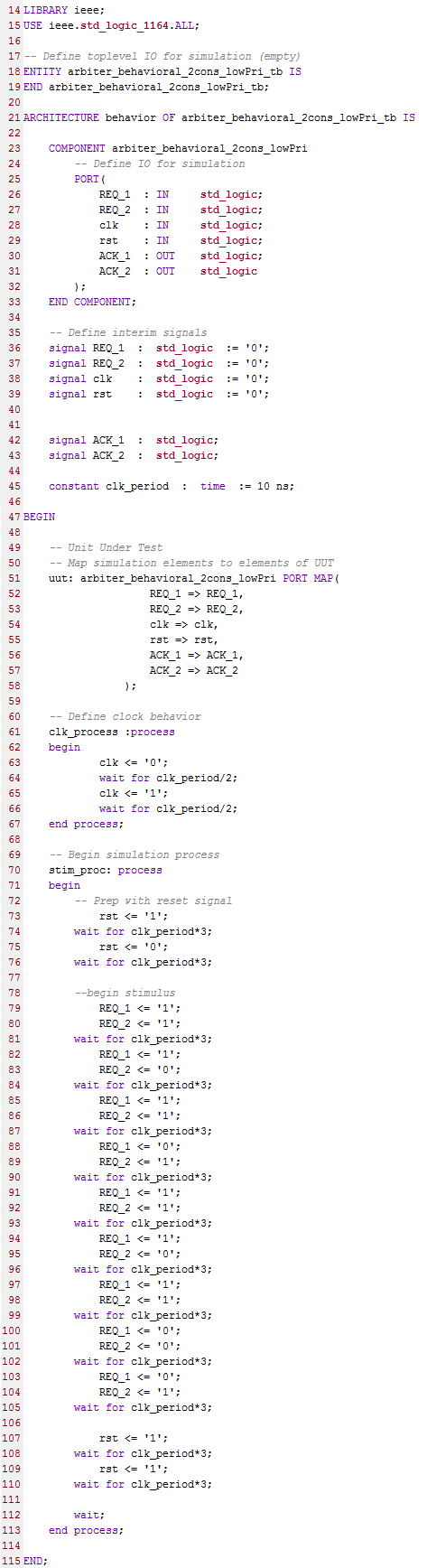
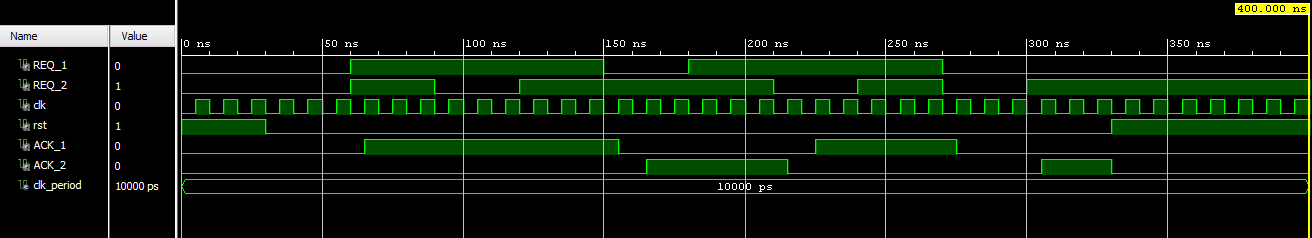
****

**Testbench and Simulation Results**

** Behavioral VHDL code**

****

**Testbench and Simulation Results**

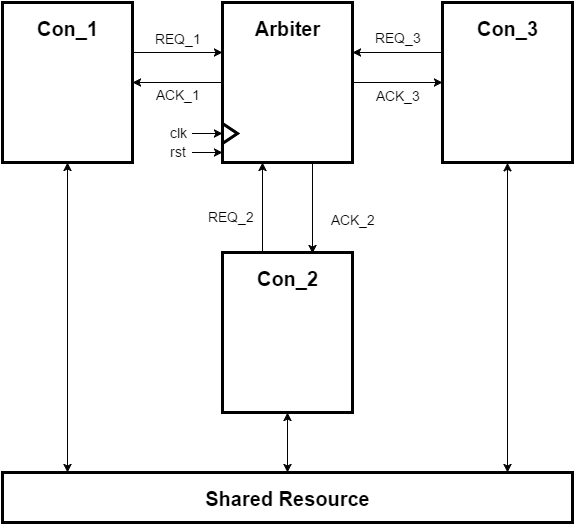
** Homework Problem III:**

Given a resource that is to be shared by three consumers such that only one consumer has access to the resource at any given time. The policy of access is non-preemptive with no priority. More specifically the policy is as follows:

* When the resource is idle and the consumers simultaneously request the resource, their requests are ignored until only one request is submitted
* When the resource is being accessed by any of the consumers, the consumers are scheduled as First-Come-First-Served (FCFS)

Design a three-consumer arbiter/controller that controls the access to the shared resource and implements the above policy. In the design process, provide the following:

1. The system architecture (block diagram) showing the interface ports to the arbiter including the clock and reset signals.
2. Finite State Machine (FSM) diagram showing all possible states, transitions, and output values.
3. K-maps for internal state and output variables.
4. Boolean expressions for internal state and output variables.
5. Detailed logic diagram using synchronous memory elements showing internal connections and external interfaces.
6. Complete description of the arbiter using VHDL code.
7. Complete VHDL testbench along with simulation results.

****

**Figure 1: System architecture**

**Table 1: Input Code Assignment**

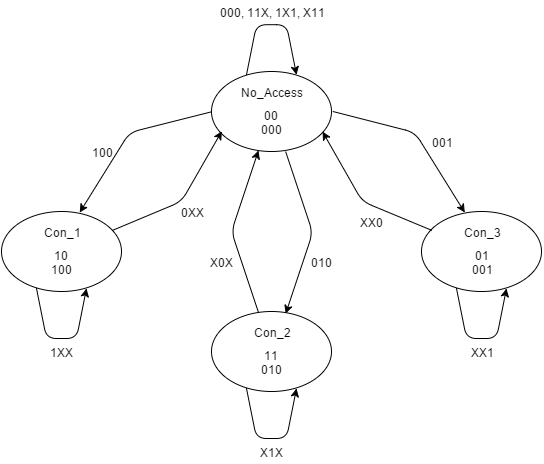
|  |  |  |  |
| --- | --- | --- | --- |
| Input Combinations (Input Codes) | | | Input Description |
| REQ\_01 | REQ\_02 | REQ\_03 |
| 0 | 0 | 0 | No Requests |
| 0 | 0 | 1 | Consumer 3 requests resource |
| 0 | 1 | 1 | Consumers 2 and 3 requesting |
| 0 | 1 | 0 | Consumer 2 Requests resource |
| 1 | 1 | 0 | Consumers 1 and 2 requesting |
| 1 | 1 | 1 | Consumers 1, 2, and 3 requesting |
| 1 | 0 | 1 | Consumers 1 and 3 requesting |
| 1 | 0 | 0 | Consumer 1 requests resource |

**Table 2: Output Code Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| Output Combinations (Output Codes) | | | Output Description |
| ACK\_01 | ACK\_02 | ACK\_03 |
| 0 | 0 | 0 | None granted access to resource |
| 0 | 0 | 1 | Consumer 3 granted access to resource |
| 0 | 1 | 1 | Forbidden output |
| 0 | 1 | 0 | Consumer 2 granted access to resource |
| 1 | 1 | 0 | Forbidden output |
| 1 | 1 | 1 | Forbidden output |
| 1 | 0 | 1 | Forbidden output |
| 1 | 0 | 0 | Consumer 1 granted access to resource |

**Table 3: State Code Assignment**

|  |  |  |
| --- | --- | --- |
| State Description | State Codes | |
| S\_01 | S\_02 |
| Resource is idle | 0 | 0 |
| Resource is used by consumer 3 | 0 | 1 |
| Resource is used by consumer 1 | 1 | 0 |
| Resource is used by consumer 2 | 1 | 1 |

****

**Figure 2: State transition diagram**

**Table 4: State transition table**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1 , S2)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** | **00** | **01** | **00** | **11** | **00** | **00** | **00** | **10** |
| **01** | **00** | **01** | **01** | **00** | **00** | **01** | **01** | **00** |
| **11** | **00** | **00** | **11** | **11** | **11** | **11** | **00** | **00** |
| **10** | **00** | **00** | **00** | **00** | **10** | **10** | **10** | **10** |

**Table 5: K-map for the state variable *S1***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Next State**  **(S1)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  | **1** |  |  |  | **1** |
| **01** |  |  |  |  |  |  |  |  |
| **11** |  |  | **1** | **1** | **1** | **1** |  |  |
| **10** |  |  |  |  | **1** | **1** | **1** | **1** |



**Table 6: K-map for the state variable *S2***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Next State**  **(S2)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** |  | **1** |  | **1** |  |  |  |  |
| **01** |  | **1** | **1** |  |  | **1** | **1** |  |
| **11** |  |  | **1** | **1** | **1** | **1** |  |  |
| **10** |  |  |  |  |  |  |  |  |



**Table 7: Output transition table**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1 , ACK2 , ACK3)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** | **000** | **000** | **000** | **000** | **000** | **000** | **000** | **000** |
| **01** | **001** | **001** | **001** | **001** | **001** | **001** | **001** | **001** |
| **11** | **010** | **010** | **010** | **010** | **010** | **010** | **010** | **010** |
| **10** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** |

**Table 8: K-map for the output variable ACK1**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK1)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |  |  |  |  |
| **01** |  |  |  |  |  |  |  |  |
| **11** |  |  |  |  |  |  |  |  |
| **10** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |



**Table 9: K-map for the output variable ACK2**

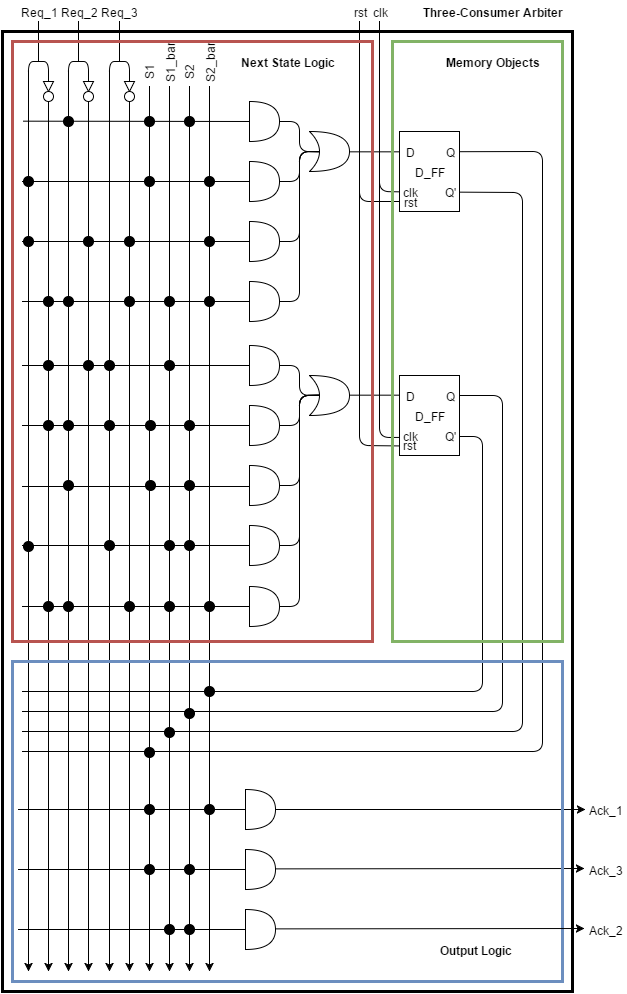
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK2)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |  |  |  |  |
| **01** |  |  |  |  |  |  |  |  |
| **11** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **10** |  |  |  |  |  |  |  |  |



**Table 10: K-map for the output variable ACK3**

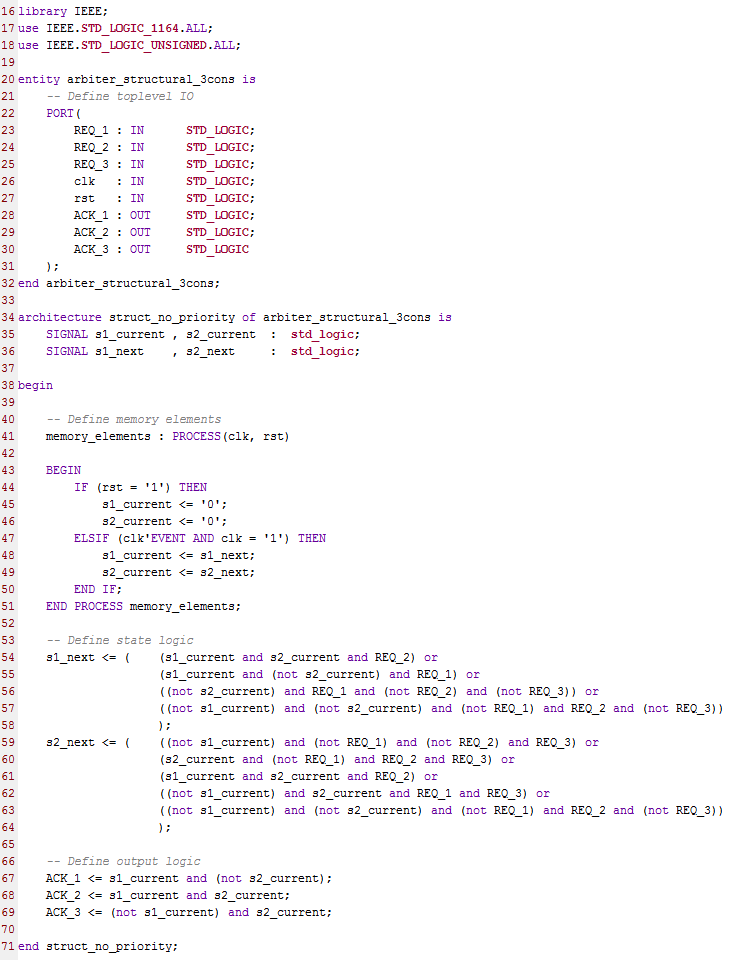
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Outputs**  **(ACK3)** | | **Inputs**  **(REQ1 , REQ2 , REQ3)** | | | | | | | |
| **000** | **001** | **011** | **010** | **110** | **111** | **101** | **100** |
| **Current State**  **(S1 , S2)** | **00** |  |  |  |  |  |  |  |  |
| **01** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **11** |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |



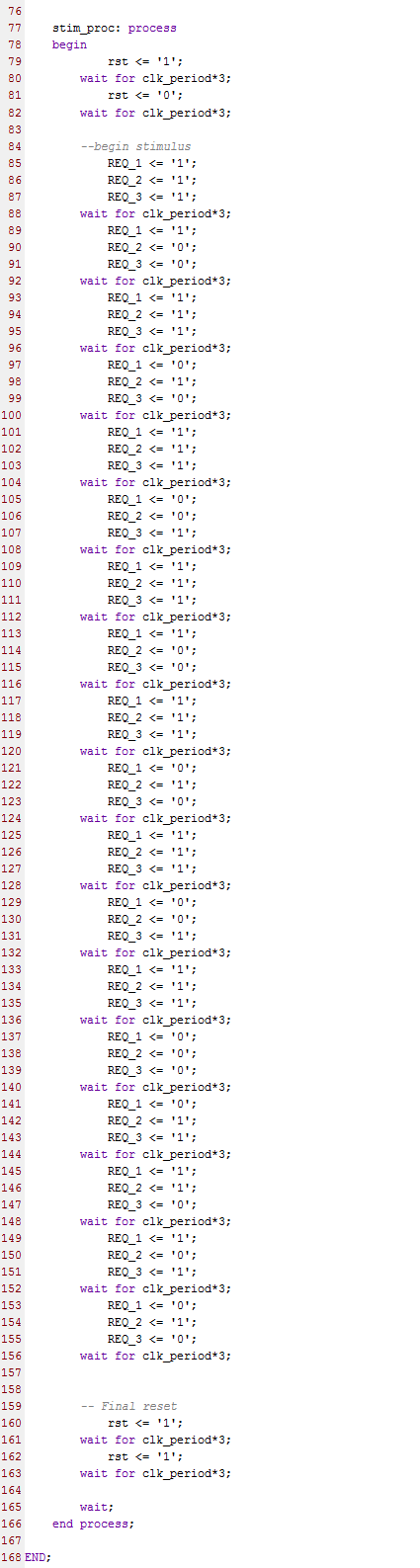
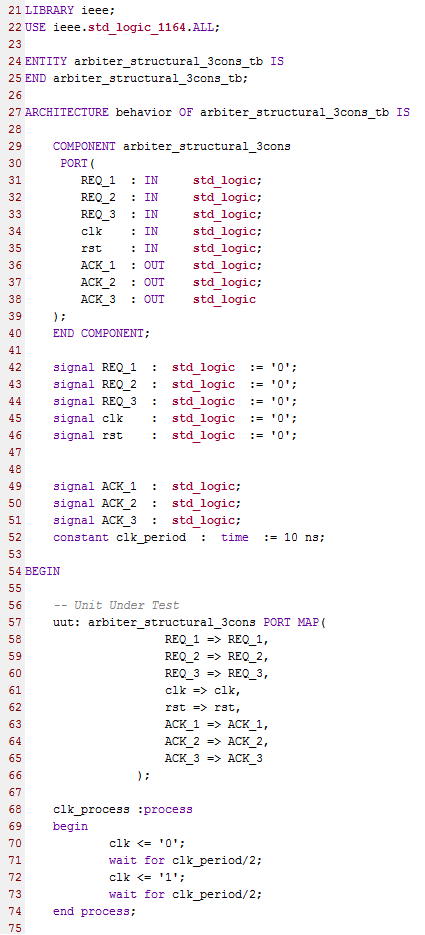
****

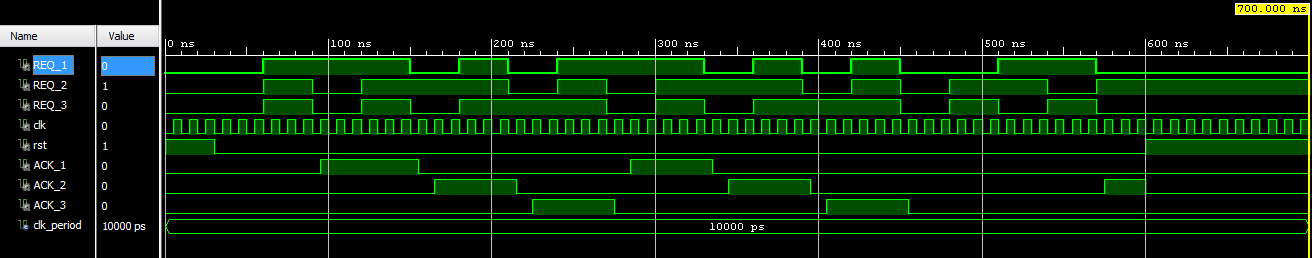
**Figure 3: Synchronous logic diagram**

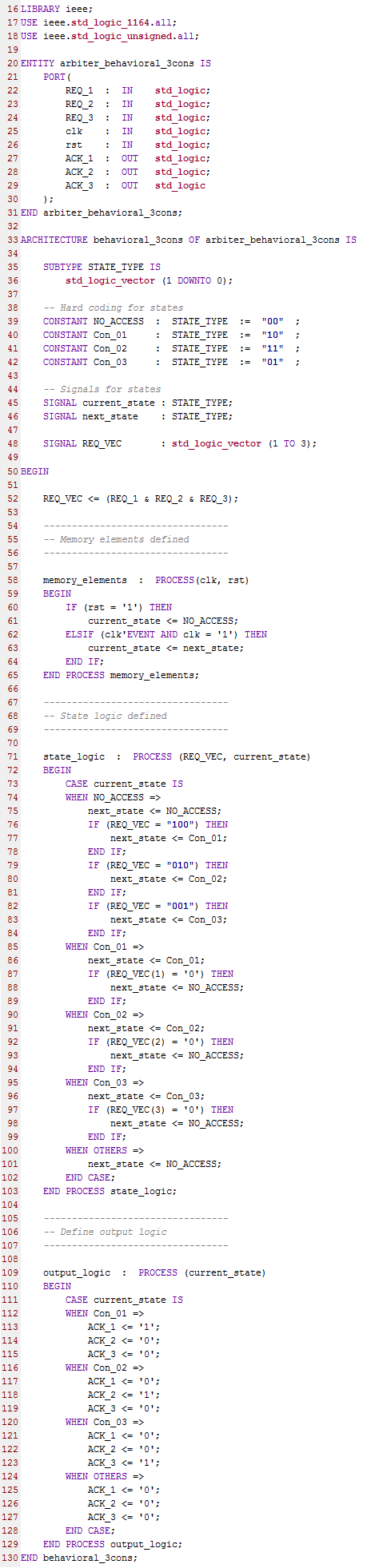
**Structural VHDL code**

****

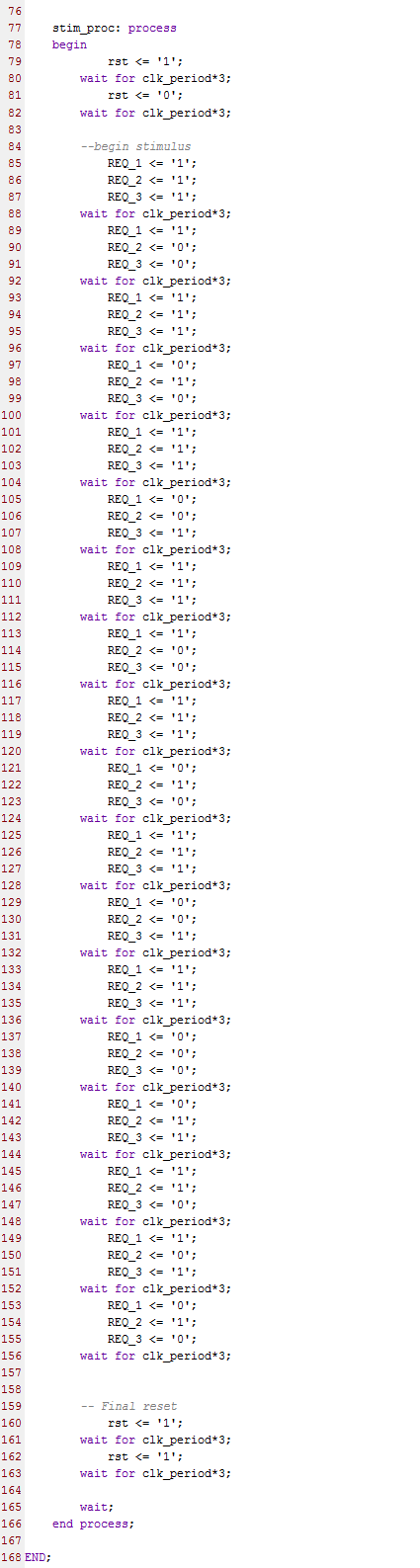
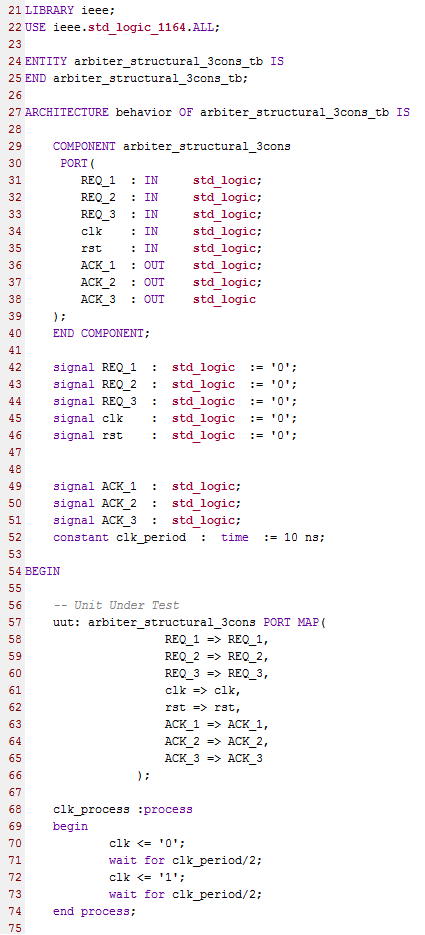
**Testbench and Simulation Results**

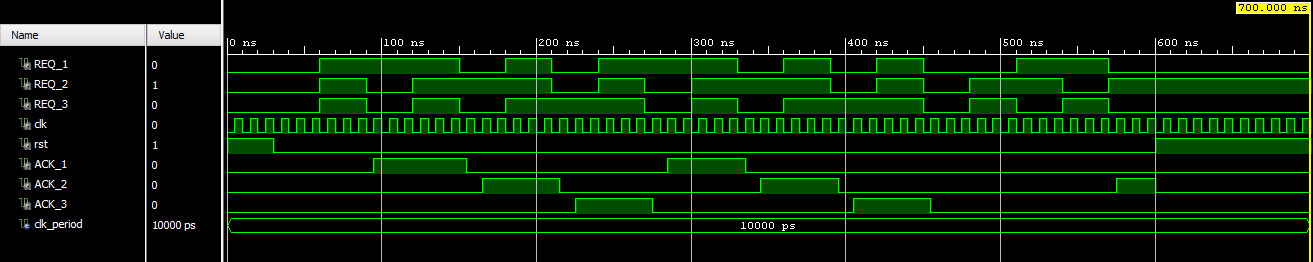
****

**Behavioral VHDL code**

****

**Testbench and Simulation Results**

****

****