KU_logo KU_soeng



**The University of Kansas**

**School of Engineering**

**Department of Electrical Engineering and Computer Science**

EECS 645 – Computer Architecture

Fall 2016

Homework 02 (Resource Sharing)

Student Name: Tim Elvart Student ID: 2760606

**Homework Problem:**

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 preemptive with descending priority. More specifically the policy is as follows:

* Priority is determined by the consumer ID (index), i.e. consumer 1 has the highest priority than all other consumers, and consumer 3 has the lowest priority than all other consumers
* When the resource is idle or being accessed by any of the consumers and the consumers simultaneously request the resource, the consumers are scheduled/preempted according to their priority
* Consumers are responsible for saving and restoring their work if preempted (i.e. the controller is simple and does not handle any context switching issues)
* The controller should also be capable of *self-recovery* and handling *race-conditions*

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 both *structural* and *behavioral* VHDL.
7. Simulation results.

Con\_2

**ACK\_02 REQ\_02**

Con\_3

**REQ\_01 REQ\_03**

Con\_1

Arbiter

**ACK\_01 ACK\_03**

Shared Resource

**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 resource requests |
| 0 | 0 | 1 | Consumer 3 requests resource |
| 0 | 1 | 0 | Consumer 2 requests resource |
| 0 | 1 | 1 | Consumers 2,3 request resource |
| 1 | 0 | 0 | Consumer 1 requests resource |
| 1 | 0 | 1 | Consumers 1,3 request resource |
| 1 | 1 | 0 | Consumers 1,2 request resource |
| 1 | 1 | 1 | Consumers 1,2,3 request resource |

**Table 2: Output Code Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| Output Combinations (Output Codes) | | | Output Description |
| ACK\_01 | ACK\_02 | ACK\_03 |
| 0 | 0 | 0 | Resource is idle |
| 0 | 0 | 1 | Consumer 3 granted access |
| 0 | 1 | 0 | Consumer 2 granted access |
| 0 | 1 | 1 | Forbidden output |
| 1 | 0 | 0 | Consumer 1 granted access |
| 1 | 0 | 1 | Forbidden output |
| 1 | 1 | 0 | Forbidden output |
| 1 | 1 | 1 | Forbidden output |

**Table 3: State Code Assignment**

|  |  |  |
| --- | --- | --- |
| State Description | State Codes | |
| S\_01 | S\_02 |
| Resource is idle (No access) | 0 | 0 |
| Resource used by consumer 1 | 0 | 1 |
| Resource used by consumer 2 | 1 | 0 |
| Resource used by consumer 3 | 1 | 1 |

**000**

Idle

00

000

**1XX**

**X0X**

**0XX 01X**

**001**

**XX0**

Con\_1

01

100

Con\_2

10

010

Con\_3

11

001

**1XX 01X**

**001**

**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** | **11** | **10** | **10** | **01** | **01** | **01** | **01** |
| **01** | **00** | **00** | **00** | **00** | **01** | **01** | **01** | **01** |
| **11** | 00 | 11 | 00 | 00 | 00 | 00 | 00 | 00 |
| **10** | **00** | **00** | **10** | **10** | **00** | **00** | **00** | **00** |

**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** | **0** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **01** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **11** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** |



**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** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** |
| **01** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **11** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |



**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** | **100** | **100** | **100** | **100** | **100** | **100** | **100** | **100** |
| **11** | 001 | 001 | 001 | 001 | 001 | 001 | 001 | 001 |
| **10** | **010** | **010** | **010** | **010** | **010** | **010** | **010** | **010** |

**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** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **01** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |



**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** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **01** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |



**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** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **01** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **11** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **10** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |



**Figure 3: Synchronous logic diagram**

**Structural VHDL code**

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE ieee.std\_logic\_unsigned.all;

ENTITY arbiter\_struct\_3cons IS

PORT(

REQ\_01 : IN std\_logic;

REQ\_02 : IN std\_logic;

REQ\_03 : IN std\_logic;

clk : IN std\_logic;

rst : IN std\_logic;

ACK\_01 : OUT std\_logic;

ACK\_02 : OUT std\_logic;

ACK\_03 : OUT std\_logic

);

END arbiter\_struct\_3cons ;

ARCHITECTURE struct\_priority OF arbiter\_struct\_3cons IS

-- Declare current and next state signals

SIGNAL s1\_current, s2\_current : std\_logic;

SIGNAL s1\_next , s2\_next : std\_logic;

BEGIN

----------------------------------------------------------------------------

memory\_elements : PROCESS(clk, rst)

----------------------------------------------------------------------------

BEGIN

IF (rst = '1') THEN

s1\_current <= '0';

s2\_current <= '0';

ELSIF (clk'EVENT AND clk = '1') THEN

s1\_current <= s1\_next;

s2\_current <= s2\_next;

END IF;

END PROCESS memory\_elements;

----------------------------------------------------------------------------

-- state\_logic

----------------------------------------------------------------------------

--- insert your code here ---

s1\_next <= ((not REQ\_01) and (not REQ\_02) and REQ\_03 and s1\_current and s2\_current)

or ((not REQ\_01) and REQ\_03 and (not s1\_current) and (not s2\_current))

or ((not REQ\_01) and REQ\_02 and (not s2\_current));

s2\_next <= ((not REQ\_01) and (not REQ\_02) and REQ\_03 and (not s1\_current) and (not s2\_current))

or ((not REQ\_01) and (not REQ\_02) and REQ\_03 and s1\_current and s2\_current)

or (REQ\_01 and (not s1\_current));

-----------------------------

----------------------------------------------------------------------------

-- output\_logic

----------------------------------------------------------------------------

--- insert your code here ---

ACK\_01 <= (not s1\_current) and s2\_current;

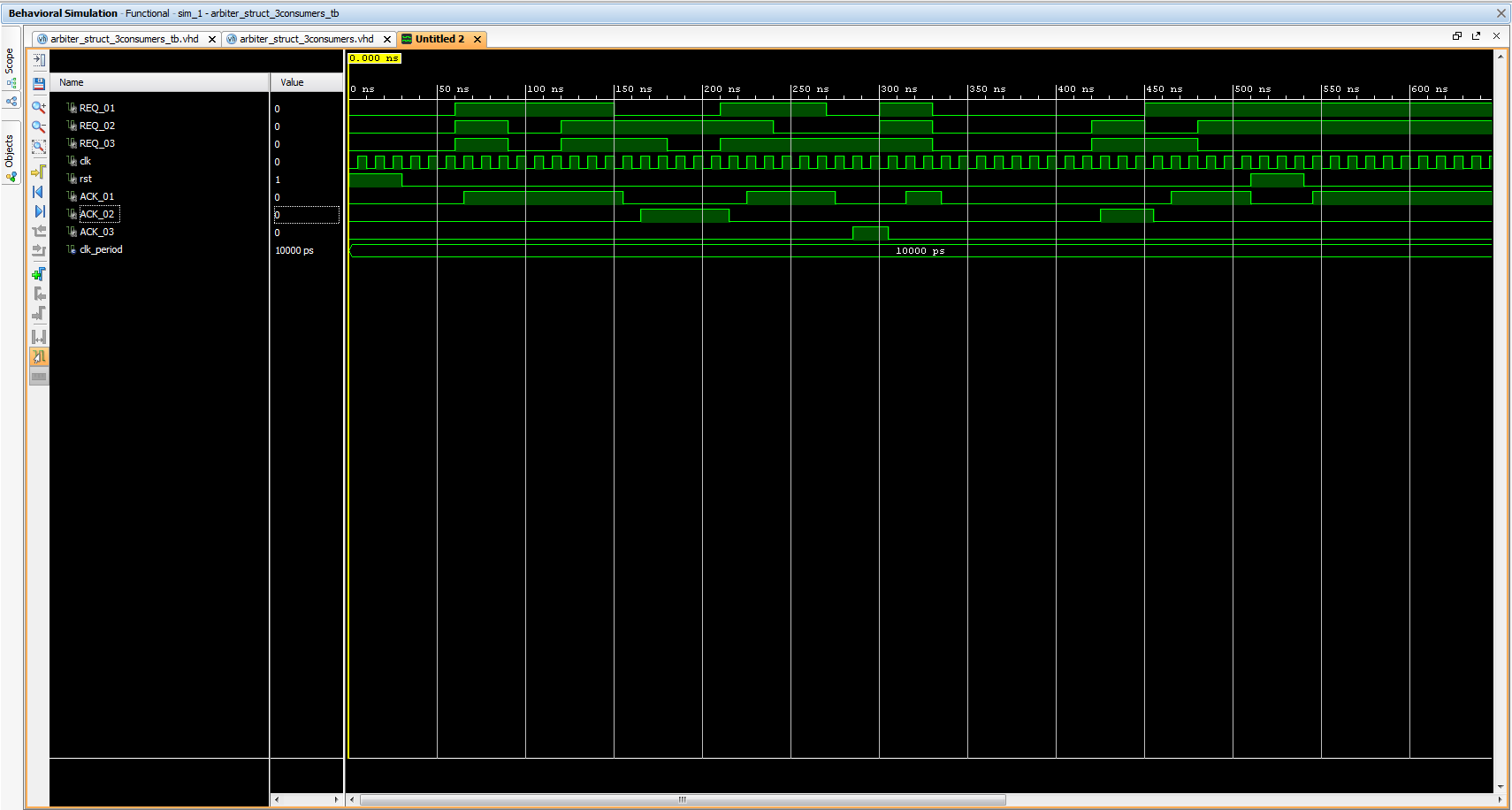
ACK\_02 <= s1\_current and (not s2\_current);

ACK\_03 <= s1\_current and s2\_current;

-----------------------------

END struct\_priority;

**Simulation Results**

**Behavioral VHDL code**

**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.all;**

**USE ieee.std\_logic\_unsigned.all;**

**ENTITY arbiter\_bahav\_3cons IS**

**PORT(**

**REQ\_01 : IN std\_logic;**

**REQ\_02 : IN std\_logic;**

**REQ\_03 : IN std\_logic;**

**clk : IN std\_logic;**

**rst : IN std\_logic;**

**ACK\_01 : OUT std\_logic;**

**ACK\_02 : OUT std\_logic;**

**ACK\_03 : OUT std\_logic**

**);**

**END arbiter\_bahav\_3cons ;**

**ARCHITECTURE behav\_priority OF arbiter\_bahav\_3cons IS**

**-- Architecture Declarations**

**SUBTYPE STATE\_TYPE IS**

**std\_logic\_vector(1 DOWNTO 0);**

**-- Hard encoding**

**CONSTANT NO\_ACCESS : STATE\_TYPE := "00" ;**

**CONSTANT Con\_01 : STATE\_TYPE := "01" ;**

**CONSTANT Con\_02 : STATE\_TYPE := "10" ;**

**CONSTANT Con\_03 : STATE\_TYPE := "11" ;**

**-- Declare current and next state signals**

**SIGNAL current\_state : STATE\_TYPE ;**

**SIGNAL next\_state : STATE\_TYPE ;**

**SIGNAL REQ\_VEC : std\_logic\_vector(1 TO 3);**

**BEGIN**

**REQ\_VEC <= (REQ\_01 & REQ\_02 & REQ\_03);**

**---------------------------------------------------------------------------**

**memory\_elements : PROCESS(clk, rst)**

**----------------------------------------------------------------------------**

**BEGIN**

**--- insert your code here ---**

**IF (rst = '1') THEN**

**current\_state <= NO\_ACCESS;**

**ELSIF (clk'EVENT AND clk = '1') THEN**

**current\_state <= next\_state;**

**END IF;**

**-----------------------------**

**END PROCESS memory\_elements;**

**----------------------------------------------------------------------------**

**state\_logic : PROCESS (REQ\_VEC, current\_state)**

**----------------------------------------------------------------------------**

**BEGIN**

**--- insert your code here ---**

**CASE current\_state IS**

**WHEN NO\_ACCESS =>**

**next\_state <= NO\_ACCESS;**

**IF (REQ\_VEC = "000") THEN**

**next\_state <= NO\_ACCESS;**

**ELSIF (REQ\_VEC = "001") THEN**

**next\_state <= Con\_03;**

**ELSIF (REQ\_VEC = "011") THEN**

**next\_state <= Con\_02;**

**ELSIF (REQ\_VEC = "010") THEN**

**next\_state <= Con\_02;**

**ELSE**

**next\_state <= Con\_01;**

**END IF;**

**WHEN Con\_01 =>**

**next\_state <= Con\_01;**

**IF (REQ\_VEC = "110") THEN**

**next\_state <= Con\_01;**

**ELSIF (REQ\_VEC = "111") THEN**

**next\_state <= Con\_01;**

**ELSIF (REQ\_VEC = "101") THEN**

**next\_state <= Con\_01;**

**ELSIF (REQ\_VEC = "100") THEN**

**next\_state <= Con\_01;**

**ELSE**

**next\_state <= NO\_ACCESS;**

**END IF;**

**WHEN Con\_02 =>**

**next\_state <= Con\_02;**

**IF (REQ\_VEC = "011") THEN**

**next\_state <= Con\_02;**

**ELSIF (REQ\_VEC = "010") THEN**

**next\_state <= Con\_02;**

**ELSE**

**next\_state <= NO\_ACCESS;**

**END IF;**

**WHEN Con\_03 =>**

**next\_state <= Con\_03;**

**IF (REQ\_VEC = "001") THEN**

**next\_state <= Con\_03;**

**ELSE**

**next\_state <= NO\_ACCESS;**

**END IF;**

**WHEN OTHERS =>**

**next\_state <= NO\_ACCESS;**

**END CASE;**

**-----------------------------**

**END PROCESS state\_logic;**

**----------------------------------------------------------------------------**

**output\_logic : PROCESS (current\_state)**

**----------------------------------------------------------------------------**

**BEGIN**

**--- insert your code here ---**

**CASE current\_state IS**

**WHEN NO\_ACCESS =>**

**ACK\_01 <= '0';**

**ACK\_02 <= '0';**

**ACK\_03 <= '0';**

**WHEN Con\_01 =>**

**ACK\_01 <= '1';**

**ACK\_02 <= '0';**

**ACK\_03 <= '0';**

**WHEN Con\_02 =>**

**ACK\_01 <= '0';**

**ACK\_02 <= '1';**

**ACK\_03 <= '0';**

**WHEN Con\_03 =>**

**ACK\_01 <= '0';**

**ACK\_02 <= '0';**

**ACK\_03 <= '1';**

**WHEN OTHERS =>**

**ACK\_01 <= '0';**

**ACK\_02 <= '0';**

**ACK\_03 <= '0';**

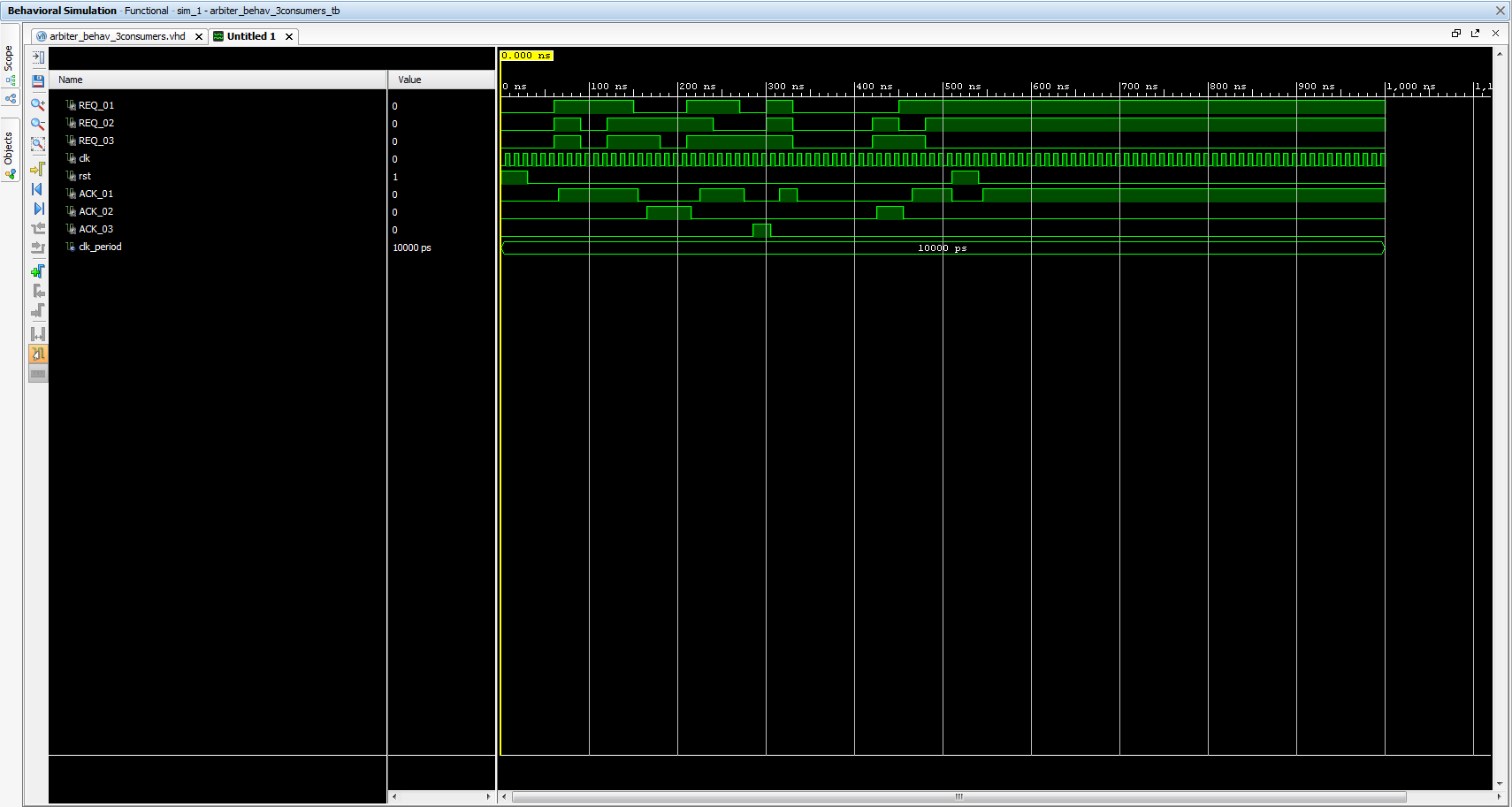
**END CASE;**

**-----------------------------**

**END PROCESS output\_logic;**

**END behav\_priority;**

**Simulation Results**

****