THE UNIVERSITY OF KANSAS

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

EECS 645 – Computer Architecture

Fall 2016

Homework 02 (Resource Sharing) Solution

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.

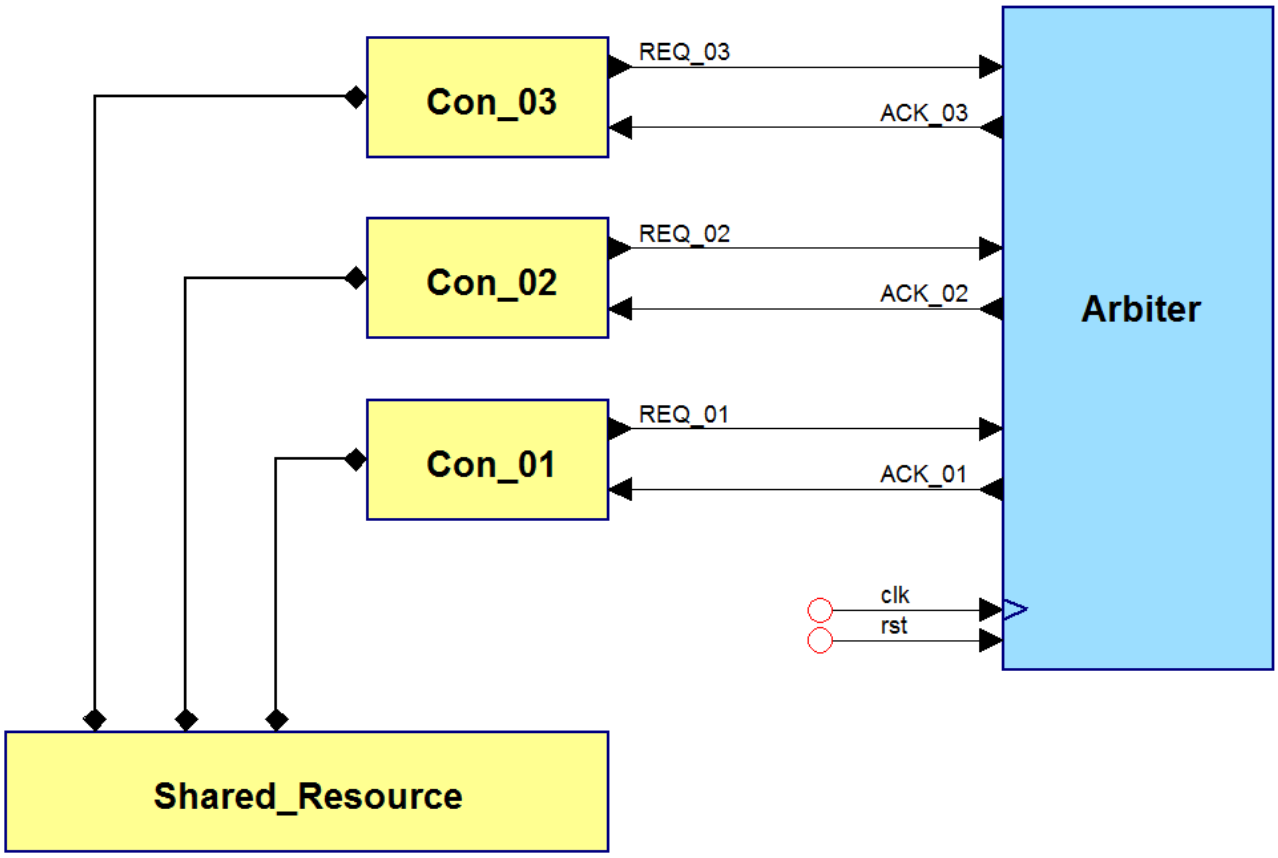


Figure 1: System architecture

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	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	All consumers request 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	0	Consumer 2 granted access to resource
0	1	1	Forbidden output
1	0	0	Consumer 1 granted access to resource
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
Required States	Resource is idle (No access)	0	0
	Resource is used by consumer 1 (Con_01)	0	1
	Resource is used by consumer 2 (Con_02)	1	0
	Resource is used by consumer 3 (Con_03)	1	1

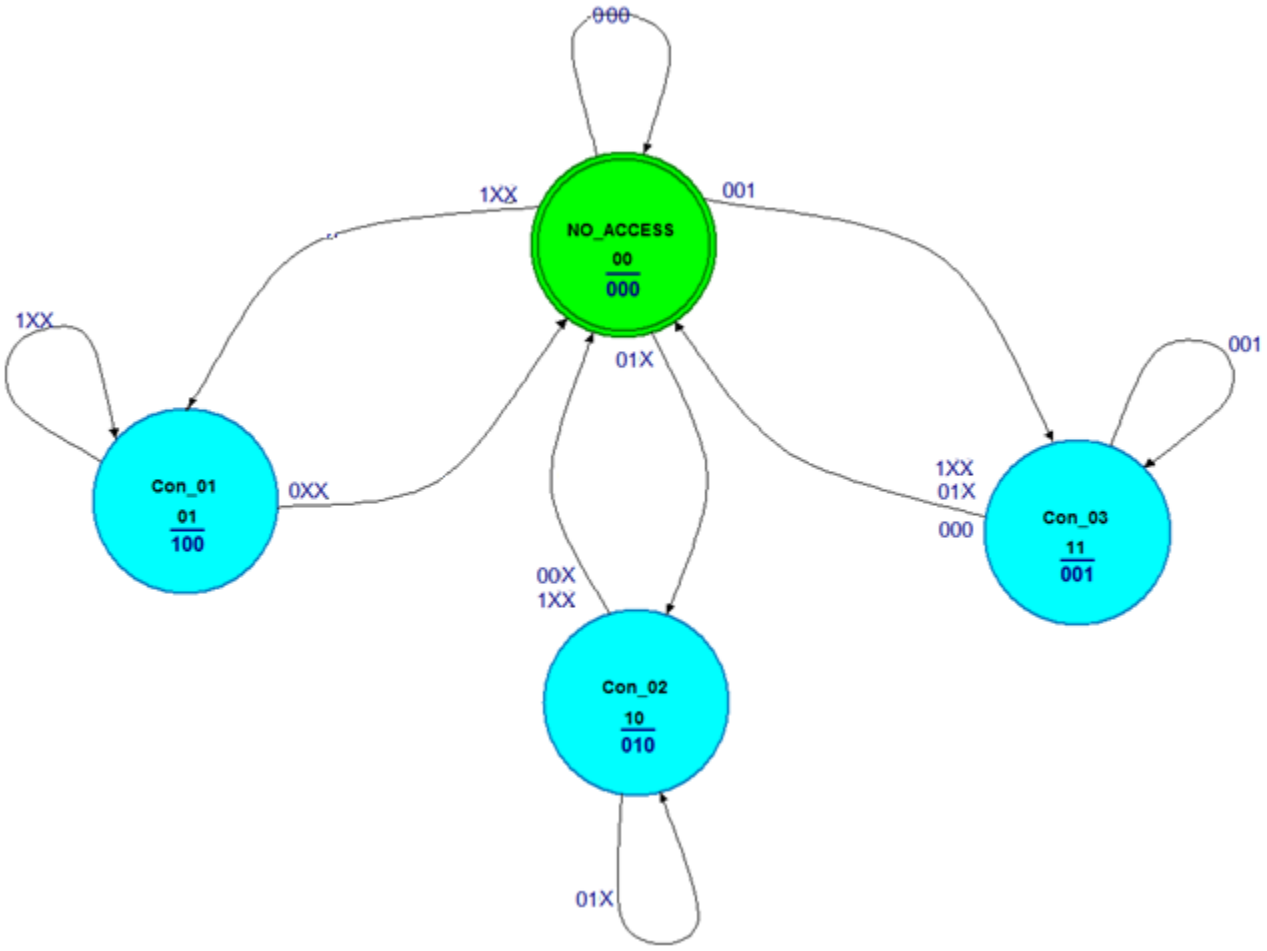


Figure 2: State transition diagram

Table 4: State transition table

Next State (S_1 , S_2)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	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 S_1

Next State (S_1)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	00		1	1	1				
	01								
	11		1						
	10			1	1				

$$S_1^{next} = \overline{REQ_1} \cdot REQ_2 \cdot \overline{S_2^{current}} + \overline{REQ_1} \cdot REQ_3 \cdot \overline{S_1^{current}} \cdot \overline{S_2^{current}} + \overline{REQ_1} \cdot \overline{REQ_2} \cdot REQ_3 \cdot S_1^{current} \cdot S_2^{current}$$

Table 6: K-map for the state variable S_2

Next State (S_2)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	00		1			1	1	1	1
	01					1	1	1	1
	11		1						
	10								

$$S_2^{next} = REQ_1 \cdot \overline{S_1^{current}} + \overline{REQ_2} \cdot REQ_3 \cdot \overline{S_1^{current}} \cdot \overline{S_2^{current}} + \overline{REQ_1} \cdot \overline{REQ_2} \cdot REQ_3 \cdot S_1^{current} \cdot S_2^{current}$$

Table 7: Output transition table

Outputs (ACK_1 , ACK_2 , ACK_3)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	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 ACK_1

Outputs (ACK_1)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	00								
	01	1	1	1	1	1	1	1	1
	11								
	10								

$$ACK_1 = \overline{S_1^{current}} \cdot S_2^{current}$$

Table 9: K-map for the output variable ACK_2

Outputs (ACK_2)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	00								
	01								
	11								
	10	1	1	1	1	1	1	1	1

$$ACK_2 = S_1^{current} \cdot \overline{S_2^{current}}$$

Table 10: K-map for the output variable ACK_3

Outputs (ACK_3)		Inputs (REQ_1 , REQ_2 , REQ_3)							
		000	001	011	010	110	111	101	100
Current State (S_1 , S_2)	00								
	01								
	11	1	1	1	1	1	1	1	1
	10								

$$ACK_3 = S_1^{current} \cdot S_2^{current}$$

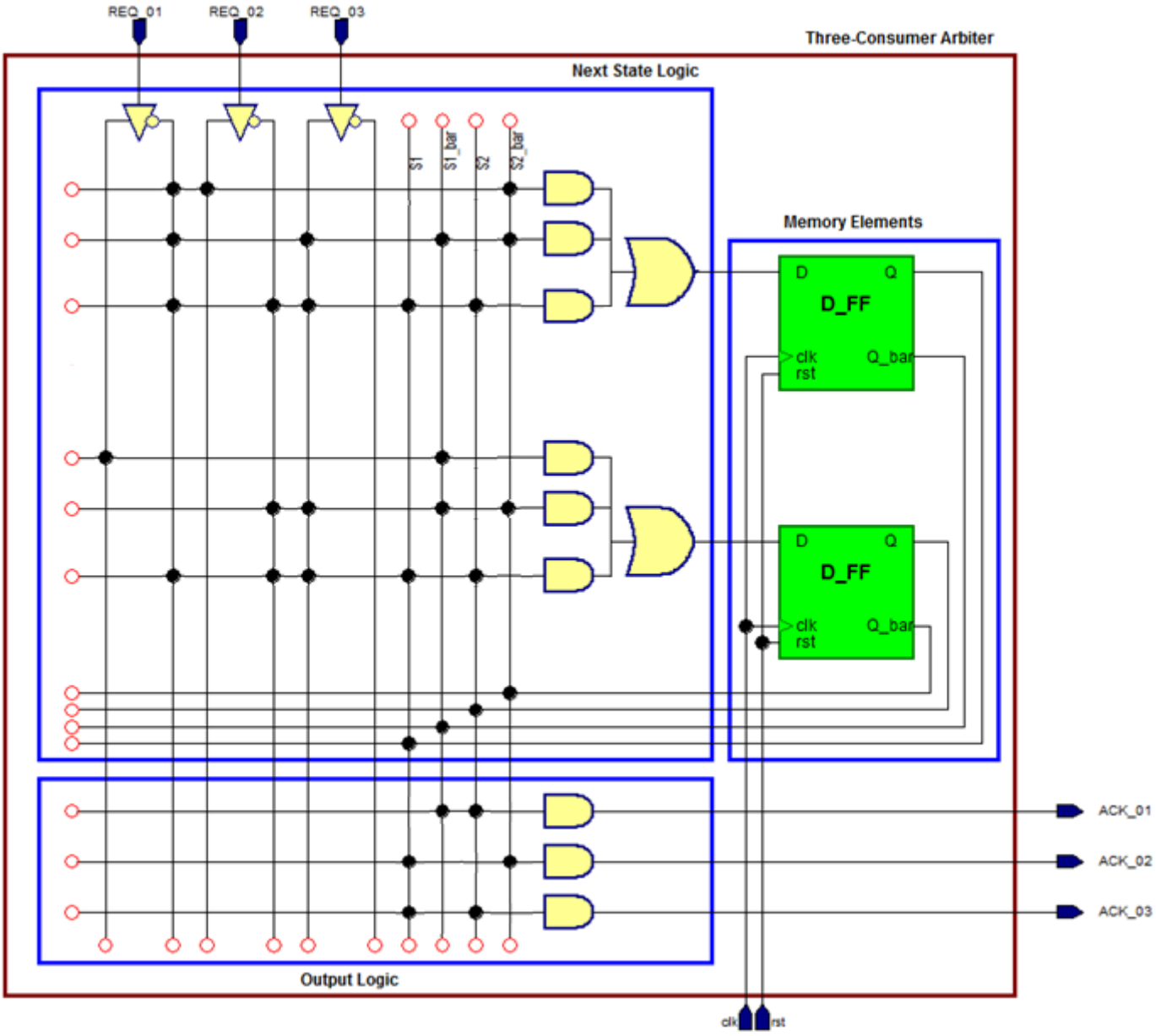


Figure 3: Synchronous logic diagram

Structural VHDL code

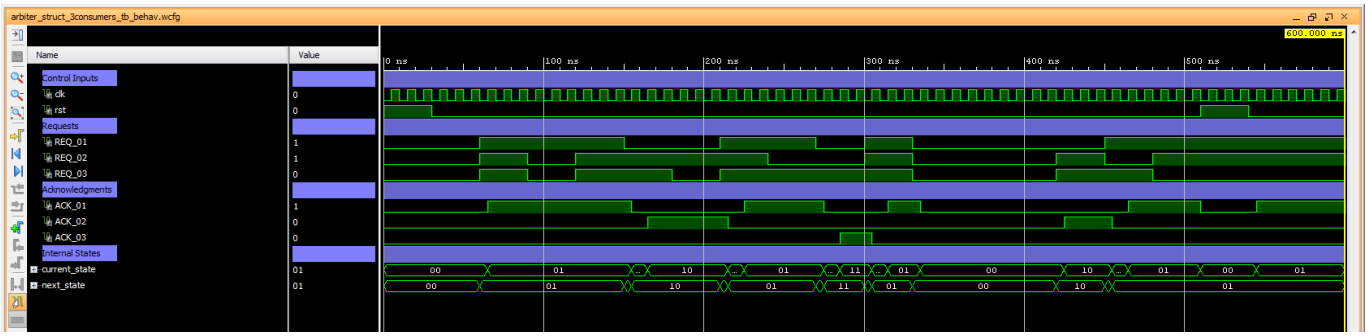
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

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';
         -- Reset Values
      ELSIF (clk'EVENT AND clk = '1') THEN
         s1_current <= s1_next;
         s2_current <= s2_next;
      END IF;
   END PROCESS memory_elements;

   -----------------------------------------------------------------------
   -- state_logic
   -----------------------------------------------------------------------
   s1_next <= ((not REQ_01) and (    REQ_02)                                    and (not s2_current)) or
              ((not REQ_01)                  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));


   s2_next <= ((    REQ_01)                                    and (not s1_current)                    ) or
              (                (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));

   -----------------------------------------------------------------------
   -- output_logic
   -----------------------------------------------------------------------
   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;
```

Testbench and Simulation Results

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY arbiter_struct_3consumers_tb IS
END arbiter_struct_3consumers_tb;

ARCHITECTURE behavior OF arbiter_struct_3consumers_tb IS

   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT 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 COMPONENT ;

   --Inputs
   signal REQ_01 : std_logic := '0';
   signal REQ_02 : std_logic := '0';
   signal REQ_03 : std_logic := '0';
   signal clk    : std_logic := '0';
   signal rst    : std_logic := '0';

   --Outputs
   signal ACK_01 : std_logic;
   signal ACK_02 : std_logic;
   signal ACK_03 : std_logic;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: arbiter_struct_3cons PORT MAP (
                                       REQ_01 => REQ_01,
                                       REQ_02 => REQ_02,
                                       REQ_03 => REQ_03,
                                       clk => clk,
                                       rst => rst,
                                       ACK_01 => ACK_01,
                                       ACK_02 => ACK_02,
                                       ACK_03 => ACK_03
                                     );

   -- Clock process definitions
   clk_process :process
   begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
   end process;
```

```
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 3
                rst <= '1';
      wait for clk_period*3;

                rst <= '0';
      wait for clk_period*3;

      -- insert stimulus here
                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '0';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '0';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '0';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '0';
      wait for clk_period*9;
```

```
                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '0';
      wait for clk_period*3;

                rst <= '1';
      wait for clk_period*3;

                rst <= '0';
      wait for clk_period*3;

      wait;
   end process;

END;
```



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
      IF (rst = '1') THEN
         current_state <= NO_ACCESS;
         -- Reset Values
      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
      next_state <= NO_ACCESS;
      CASE current_state IS
      WHEN NO_ACCESS =>
         IF (REQ_VEC(1) = '1') THEN
            next_state <= Con_01;
         END IF;
         IF (REQ_VEC(1 to 2) = "01") THEN
            next_state <= Con_02;
         END IF;
         IF (REQ_VEC = "001") THEN
            next_state <= Con_03;
         END IF;
      WHEN Con_01 =>
         IF (REQ_VEC(1) = '1') THEN
            next_state <= Con_01;
         END IF;
      WHEN Con_02 =>
         IF (REQ_VEC(1 to 2) = "01") THEN
            next_state <= Con_02;
         END IF;
      WHEN Con_03 =>
         IF (REQ_VEC = "001") THEN
            next_state <= Con_03;
         END IF;
      WHEN OTHERS =>
         next_state <= NO_ACCESS;
      END CASE;
   END PROCESS state_logic;

   ----------------------------------------------------
   output_logic : PROCESS (current_state)
   ----------------------------------------------------
   BEGIN
      CASE current_state IS
      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;
```

Testbench and Simulation Results

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY arbiter_behav_3consumers_tb IS
END arbiter_behav_3consumers_tb;

ARCHITECTURE behavior OF arbiter_behav_3consumers_tb IS

   -- Component Declaration for the Unit Under Test (UUT)
   COMPONENT arbiter_behav_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 COMPONENT ;

   --Inputs
   signal REQ_01 : std_logic := '0';
   signal REQ_02 : std_logic := '0';
   signal REQ_03 : std_logic := '0';
   signal clk    : std_logic := '0';
   signal rst    : std_logic := '0';

   --Outputs
   signal ACK_01 : std_logic;
   signal ACK_02 : std_logic;
   signal ACK_03 : std_logic;

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: arbiter_behav_3cons PORT MAP (
                                      REQ_01 => REQ_01,
                                      REQ_02 => REQ_02,
                                      REQ_03 => REQ_03,
                                      clk => clk,
                                      rst => rst,
                                      ACK_01 => ACK_01,
                                      ACK_02 => ACK_02,
                                      ACK_03 => ACK_03
                                    );

   -- Clock process definitions
   clk_process :process
   begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 3
                rst <= '1';
      wait for clk_period*3;

                rst <= '0';
      wait for clk_period*3;

      -- insert stimulus here
                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '0';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '0';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '0';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '1';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '0';
                REQ_02 <= '0';
                REQ_03 <= '0';
      wait for clk_period*9;

                REQ_01 <= '0';
                REQ_02 <= '1';
                REQ_03 <= '0';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '1';
      wait for clk_period*3;

                REQ_01 <= '1';
                REQ_02 <= '0';
                REQ_03 <= '0';
      wait for clk_period*3;

                rst <= '1';
      wait for clk_period*3;

                rst <= '0';
      wait for clk_period*3;

      wait;
   end process;

END;
```