





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 <u>structural</u> and <u>behavioral</u> VHDL.
- 7) Simulation results.

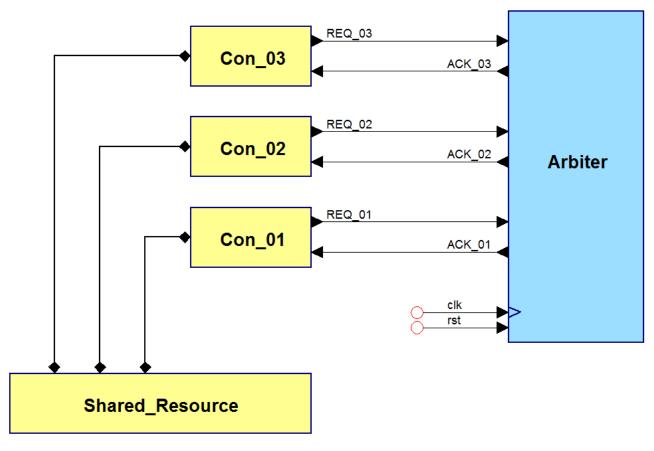


Figure 1: System architecture

Table 1: Input Code Assignment

Input Com	binations (Inp	out Codes)	Input Description
REQ_01	REQ_02	REQ_03	input Description
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 Com	binations (Ou	•	Output Description
ACK_01	ACK_02	ACK_03	Output Description
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			
	State Description	S_01	S_02		
	Resource is idle (No access)	0	0		
Required States	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		

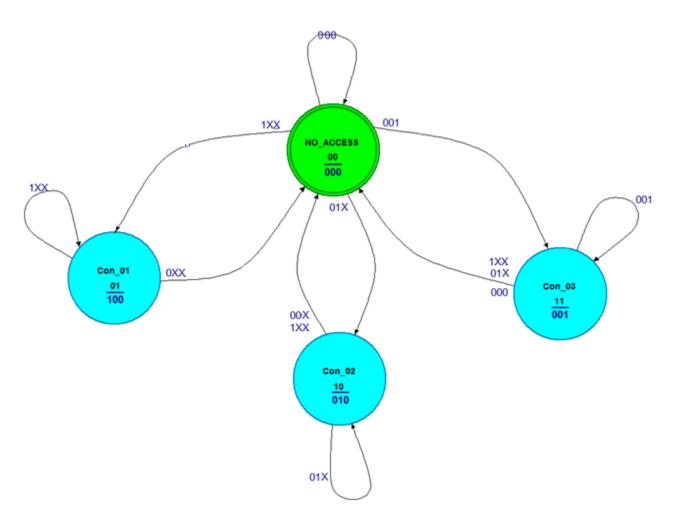


Figure 2: State transition diagram

Table 4: State transition table

Next State (S ₁ , S ₂)		Inputs (REQ ₁ , REQ ₂ , REQ ₃)								
$(\mathbf{S}_1,\mathbf{S}_2)$	000	001	011	010	110	111	101	100		
	00	00	11	10	10	01	01	01	01	
Current State	01	00	00	00	00	01	01	01	01	
(S_1, S_2)	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		Inputs (REQ1 , REQ2 , REQ3)								
(S_1)	000	001	011	010	110	111	101	100		
	00	•	1	1	1					
Current State	01									
(S_1,S_2)	11	ı	1							
	10			1	1					

 $S_{1}^{\textit{next}} = \overline{\textit{REQ}}_{1} \cdot \textit{REQ}_{2} \cdot \overline{S}_{2}^{\textit{current}} + \overline{\textit{REQ}}_{1} \cdot \textit{REQ}_{3} \cdot \overline{S}_{1}^{\textit{current}} \cdot \overline{S}_{2}^{\textit{current}} + \overline{\textit{REQ}}_{1} \cdot \overline{\textit{REQ}}_{2} \cdot \textit{REQ}_{3} \cdot S_{1}^{\textit{current}} \cdot S_{2}^{\textit{current}}$

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		
	00		1			1	1	1			
Current State	01					1	1	1			
(S_1, S_2)	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		
	00	000	000	000	000	000	000	000	000		
Current State	01	100	100	100	100	100	100	100	100		
(S_1, S_2)	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 (ACK ₁)			Inputs (REQ ₁ , REQ ₂ , REQ ₃)								
(ACK1) 000			001	011	010	110	111	101	100		
	00										
Current State	01	A	1	1	1	1	1	1	A		
(S_1, S_2)	11	· · · · · · · · · · · · · · · · · · ·									
	10										

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

Table 9: K-map for the output variable ACK2

			•							
Outputs (ACK2)		Inputs (REQ ₁ , REQ ₂ , REQ ₃)								
(ACK)	000	001	011	010	110	111	101	100		
	00									
Current State	01									
(S_1, S_2)	11									
	10	A	11	1	1	1	1	1	A	

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

Table 10: K-map for the output variable ACK₃

Outputs (ACK ₃)			Inputs (REQ ₁ , REQ ₂ , REQ ₃)								
(ACK)	000	001	011	010	110	111	101	100			
	00										
Current State	01										
(S_1, S_2)	11	A	1	1	1	1	1	1	Ų		
	10										

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

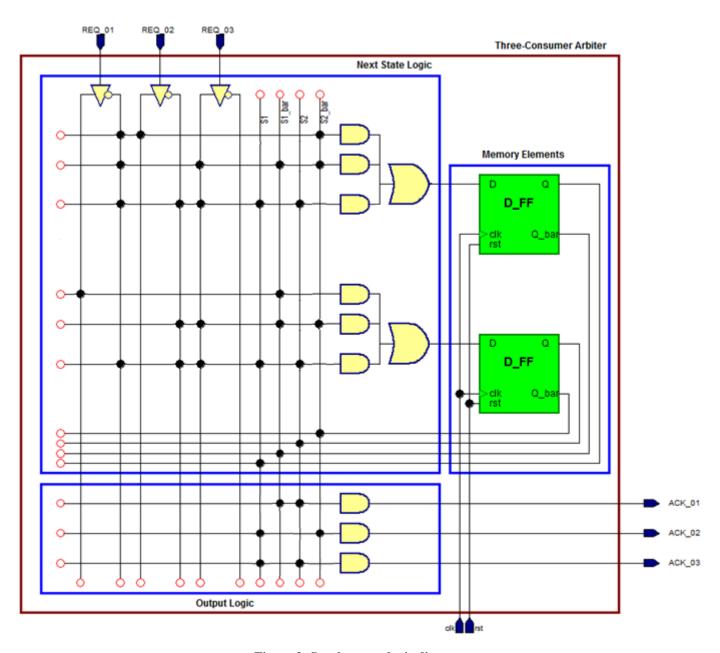


Figure 3: Synchronous logic diagram

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
REQ_02 : IN
REQ_03 : IN
                                    std_logic;
                                   std_logic;
std_logic;
             clk
                                    std_logic;
                                    std_logic;
11
             rst
                          IN
             ACK_01
ACK_02
ACK_03
                          OUT
                                    std_logic;
                                   std_logic;
std_logic
13
                          OUT
    END arbiter_struct_3cons ;
16
17
18
    ARCHITECTURE struct_priority OF arbiter_struct_3cons IS
21
22
23
24

    Declare current and next state signals

         SIGNAL s1_current, s2_current : std_logic; SIGNAL s1_next , s2_next : std_logic;
25
26
27
28
29
30
    BEGIN
         memory_elements : PROCESS(clk, rst)
31
32
33
34
             IF (rst = '1') THEN
             ir (rst = 1 ) IHEN
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:</pre>
35
36
37
38
39
         END PROCESS memory_elements;
40
41
42
43
44
         -- state_logic
         and (not s2_current)) or REQ_03) and (not s1_current) and (not s2_current)) or REQ_03) and ( s1_current) and ( s2_current));
45
                                                         REQ_02)
                         ((not REQ_01) and (not REQ_02) and (
47
48
49
50
                                                                                 and (not s1_current) ) or REQ_03) and (not s1_current) and (not s2_current)) or REQ_03) and ( s1_current) and ( s2_current));
         s2_next <= ((
                                 REQ_01)
                         51
53
54
55
         -- output_logic
56
         57
                                                            s2_current);
58
59
60
    END struct_priority;
61
```

Testbench and Simulation Results

```
62
63
64
65
66
                                                                                                                                         - Stimulus process
                                                                                                                                      stim_proc: process
                                                                                                                                     begin
-- hold reset state for rst <= '1';
                                                                                                                                            rst <= '1';
wait for clk_period*3;
                                                                                                                       67
                                                                                                                      68
69
70
71
72
73
                                                                                                                                            rst <= '0';
wait for clk_period*3;</pre>
      LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
                                                                                                                                            -- insert stimulus here
                                                                                                                                                                  REO_01 <= '1'
REO_02 <= '1'
REO_03 <= '1'
      ENTITY arbiter_struct_3consumers_tb IS
END arbiter_struct_3consumers_tb;
                                                                                                                                            wait for clk_period*3;
      ARCHITECTURE behavior OF arbiter_struct_3consumers_tb IS
           -- Component Declaration for the Unit Under Test (UUT)

COMPONENT arbiter_struct_3cons IS

FORT(

REO_01: IN std_logic;

REO_02: IN std_logic;

REO_03: IN std_logic;

clk: IN std_logic;
                                                                                                                                                                  REQ_01 <= '1'
REQ_02 <= '0'
REQ_03 <= '0'
                                                                                                                       78
79
                                                   std_logic;
std_logic;
std_logic;
std_logic;
std_logic;
std_logic;
std_logic;
std_logic;
std_logic
IN
IN
IN
IN
OUT
OUT
                                                                                                                                            wait for clk_period*3;
                                                                                                                       81
                                                                                                                                                                  REQ_01 <= '1'
REQ_02 <= '1'
REQ_03 <= '1'
                      Clk :
rst :
ACK_01 :
ACK_02 :
ACK_03 :
                                                                                                                       83
                                                                                                                       84
85
                                                                                                                       86
87
                                                                                                                                            wait for clk_period*3;
           END COMPONENT
                                                                                                                                                                  REQ_01 <= '0';
REQ_02 <= '1';
REQ_03 <= '1';
                                                                                                                       88
                                                                                                                       89
           --Inputs
signal REQ_01
signal REQ_02
signal REQ_03
signal clk
signal rst
                                                                                                                       90
                                        std_logic := '0';
                                                                                                                      91
92
93
94
                                                                                                                                            wait for clk_period*3;
                                                                                                                                                                  REQ_01 <= '0'
REQ_02 <= '1'
REQ_03 <= '0'
                                                                                                                       95
96
97
98
           --Outputs
signal ACK_01
signal ACK_02
signal ACK_03
                                                                                                                                            wait for clk_period*3;
                                                                                                                                                                                                                                                   REQ_01 <= '0';
REQ_02 <= '1';
REQ_03 <= '1';
                                                                                                                                                                                                      123
                                     : std_logic;
: std_logic;
                                                                                                                                                                  REQ_01 <= '1'
REQ_02 <= '1'
REQ_03 <= '1'
                                                                                                                       99
           -- Clock period definitions constant clk_period : time := 10 ns;
                                                                                                                                                                                                       126
                                                                                                                                                                                                                              wait for clk_period*3;
                                                                                                                     101
102
103
                                                                                                                                            wait for clk_period*3;
                                                                                                                                                                                                                                                   REQ_01 <= '1';
REQ_02 <= '0';
REQ_03 <= '1';
                                                                                                                                                                  REQ_01 <= '1';
REQ_02 <= '0';
REQ_03 <= '1';
                                                                                                                     104
           -- Instantiate the Unit Under Test (UUT) uut: arbiter_struct_3cons PORT MAP (
                                                                                                                     105
                                                                             (REO_01 => REO_01,

REO_02 => REO_02,

REO_03 => REO_03,

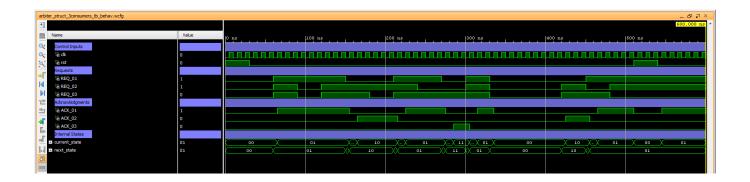
clk => clk,

rst => rst,

ACK_01 => ACK_01,

ACK_02 => ACK_02,

ACK_03 => ACK_02
                                                                                                                                                                                                       131
132
                                                                                                                                                                                                                              wait for clk_period*3;
                                                                                                                                            wait for clk_period*3;
                                                                                                                                                                                                                                                   REQ_01 <= '1';
REQ_02 <= '1';
REQ_03 <= '0';
                                                                                                                                                                                                       133
                                                                                                                                                                  REQ_01 <= '0';
REQ_02 <= '0';
REQ_03 <= '1';
                                                                                                                     109
                                                                                                                                                                                                       136
137
                                                                                                                                                                                                                              wait for clk_period*3;
                                                                                                                     111
                                                                                                                                            wait for clk_period*3;
                                                                                                                     112
113
                                                                                                                                                                                                       138
                                                                                                                                                                  REQ_01 <= '1
REQ_02 <= '1
REQ_03 <= '1
                                                                                                                                                                                                                              wait for clk_period*3;
           -- Clock process definitions clk_process :process
                                                                                                                                                                                                       140
                                                                                                                     115
                                                                                                                                                                                                                                                     rst <= '0'
                                                                                                                                                                                                      141
142
                                                                                                                                            wait for clk_period*3;
                                                                                                                                                                                                                              wait for clk_period*3;
           begin
                                   clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
                                                                                                                                                                                                       143
                                                                                                                                                                  REQ_01 <= '0';
REQ_02 <= '0';
REQ_03 <= '0';
                                                                                                                     118
119
                                                                                                                                                                                                                              wait;
                                                                                                                                                                                                     145
146
                                                                                                                                                                                                                       end process;
                                                                                                                     120
           end process:
                                                                                                                                            wait for clk_period*9;
                                                                                                                                                                                                                END:
                                                                                                                                                                                                      147
```



Behavioral VHDL code

```
LIBRARY ieee:
     USE ieee.std_logic_1164.all;
     USE ieee.std_logic_unsigned.all;
 5
    ENTITY arbiter_bahav_3cons IS
         PORT(
             RÈQ_01
                                    std_logic;
 8
             REQ_02 :
                          IN
                                     std_logic;
                                    std_logic;
std_logic;
std_logic;
std_logic;
                          IN
 9
             REQ_03 :
10
             clk
                          ĪŃ
             rst
11
             ACK_01 :
ACK_02 :
                          OUT
13
                          OUT
                                     std_logic;
             ACK_03 : OUT
14
                                    std_logic
15
         ):
16
    END arbiter_bahav_3cons ;
18
20
    ARCHITECTURE behav_priority OF arbiter_bahav_3cons IS
21
         -- Architecture Declarations
SUBTYPE STATE_TYPE IS
22
23
24
             std_logic_vector(1 DOWNTO 0);
25
26
27
             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" ;
28
29
30
31
         -- Declare current and next state signals SIGNAL current_state : STATE_TYPE ; SIGNAL next_state : STATE_TYPE ;
32
33
34
35
36
         SIGNAL REQ_VEC : std_logic_vector(1 TO 3);
37
38
    BEGIN
39
40
         REQ_VEC <= (REQ_01 & REQ_02 & REQ_03);
41
42
         memory_elements : PROCESS(clk, rst)
44
45
         BEGIN
             IF (rst = '1') THEN
46
                 current_state <= NO_ACCESS;
-- Reset Values
47
48
             ELSIF (clk'EVENT AND clk = '1') THEN current_state <= next_state;
49
50
         END IF;
END PROCESS memory_elements;
51
52
```

```
54
55
56
57
58
59
                    state_logic : PROCESS (REQ_VEC, current_state)
                           next_state <= NO_ACCESS;</pre>
                           next_state < NO_ACCESS,
CASE current_state IS
WHEN NO_ACCESS =>
IF (REQ_VEC(1) = '1') THEN
next_state <= Con_01;
END IF;
IF (DEO VEC(1 to 2) = "01"
  60
 61
62
63
64
65
66
                                   END IF:

IF (REQ_VEC(1 to 2) = "01") THEN

next_state <= Con_02;

END IF:
                                    IF (REQ_VEC = "001") THEN
                          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;
UNITED CON_03 =>

 68
69
70
71
72
73
74
75
76
77
78
                           WHEN Con_03 =>
IF (REQ_VEC = "001") THEN
 80
81
82
83
                           next_state <= Con_03;
END IF;
WHEN OTHERS =>
                   next_state <= NO_ACCESS;
END CASE;
END PROCESS state_logic;
 84
85
 86
87
 88
                   output_logic : PROCESS (current_state)
                   BEGIN
  90
91
92
93
94
95
                           SIN 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';
  96
97
98
99
                           ACK_U3 <= '0';
WHEN COD_U3 =>
ACK_U1 <= '0';
ACK_U1 <= '0';
ACK_U3 <= '1';
WHEN OTHERS =>
ACK_U1 <= '0';
ACK_U2 <= '0';
ACK_U2 <= '0';
ACK_U3 <= '0';
100
102
104
105
107
                   END CASE;
END PROCESS output_logic;
109
          END behav_priority;
```

Testbench and Simulation Results

```
- Stimulus process
                                                                                                                                      62
63
64
65
66
67
70
71
72
73
74
75
                                                                                                                                                       stim_proc: process
                                                                                                                                                       begin
                                                                                                                                                               -- hold reset state for
                                                                                                                                                              rst <= '1';
wait for clk_period*3;</pre>
                                                                                                                                                                                        rst <= '0'
                                                                                                                                                               wait for clk_period*3;
       LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
                                                                                                                                                               -- insert stimulus here
                                                                                                                                                                                        REO_01 <= '1'
REO_02 <= '1'
REO_03 <= '1'
       ENTITY arbiter_behav_3consumers_tb IS
END arbiter_behav_3consumers_tb;
                                                                                                                                      76
77
78
79
                                                                                                                                                              wait for clk_period*3;
       ARCHITECTURE behavior OF arbiter_behav_3consumers_tH IS
             -- Component Declaration for the Unit Under Test (UUT)
COMPONENT arbiter_behav_3cons IS
FORT(
REC_01: IN std_logic;
REC_02: IN std_logic;
REC_03: IN std_logic;
clk : 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;
                                                                                                                                                                                        REQ_01 <= '1'
REQ_02 <= '0'
REQ_03 <= '0'
                                                                                                                                      80
81
82
wait for clk_period*3;
                                                                                                                                                                                        REQ_01 <= '1';
REQ_02 <= '1';
REQ_03 <= '1';
                                                                                                                                      83
                                                                                                                                      84
85
                                                                                                                                      86
87
                                                                                                                                                               wait for clk_period*3;
             END COMPONENT ;
                                                                                                                                                                                        REQ_01 <= '0';
REQ_02 <= '1';
REQ_03 <= '1';
                                                                                                                                      88
90
91
93
94
95
97
98
99
             --Inputs
signal REQ_01
signal REQ_02
signal REQ_03
signal clk
signal rst
                                              std_logic := '0';
                                                                                                                                                              wait for clk_period*3;
                                                                                                                                                                                        REQ_01 <= '0';
REQ_02 <= '1';
REQ_03 <= '0';
             --Outputs
signal ACK_01 : std_logic;
signal ACK_02 : std_logic;
signal ACK_03 : std_logic;
                                                                                                                                                              wait for clk_period*3;
                                                                                                                                                                                                                                                                                   REQ_01 <= '0';
REQ_02 <= '1';
REQ_03 <= '1';
                                                                                                                                                                                                                                 123
124
                                                                                                                                                                                        REQ_01 <= '1';
REQ_02 <= '1';
REQ_03 <= '1';
                                                                                                                                    100
            -- Clock period definitions constant clk_period : time := 10 ns;
                                                                                                                                                                                                                                  126
                                                                                                                                                                                                                                                           wait for clk_period*3;
                                                                                                                                   101
102
103
104
                                                                                                                                                               wait for clk_period*3;
                                                                                                                                                                                                                                                                                    REQ_01 <= '1';
REQ_02 <= '0';
REQ_03 <= '1';
                                                                                                                                                                                        REQ_01 <= '1';
REQ_02 <= '0';
REQ_03 <= '1';
             -- Instantiate the Unit Under Test (UUT) uut: arbiter_bahav_3cons PORT MAP (
                                                                                                                                   105
106
107
108
109
110
                                                                                      (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/
                                                                                                                                                                                                                                                           wait for clk_period*3;
                                                                                                                                                               wait for clk_period*3;
                                                                                                                                                                                                                                                                                    REQ_01 <= '1';
REQ_02 <= '1';
REQ_03 <= '0';
                                                                                                                                                                                                                                  133
                                                                                                                                                                                        REQ_01 <= '0';
REQ_02 <= '0';
REQ_03 <= '1';
                                                                                                                                                                                                                                 136
137
                                                                                                                                                                                                                                                           wait for clk_period*3;
                                                                                                                                                               wait for clk_period*3;
                                                                                                                                    112
113
                                                                                                                                                                                                                                  138
                                                           );
                                                                                                                                                                                        REQ_01 <=
                                                                                                                                                                                                                                                           wait for clk_period*3;
                                                                                                                                                                                        REQ_02 <= '1'
REQ_03 <= '1'
                                                                                                                                    114
115
                                                                                                                                                                                                                                 140
                   Clock process definitions
                                                                                                                                                                                                                                                                                       st <= '0'
             clk_process :process
                                                                                                                                                                                                                                 141
142
                                                                                                                                   116
117
                                                                                                                                                               wait for clk_period*3;
                                                                                                                                                                                                                                                           wait for clk_period*3;
             begin
                                        clk <= '0'
                                                                                                                                                                                                                                 143
                                        clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;</pre>
                                                                                                                                   118
119
120
                                                                                                                                                                                        REQ_01 <= '0';
REQ_02 <= '0';
REQ_03 <= '0';
                                                                                                                                                                                                                                                            wait;
                                                                                                                                                                                                                                 145
                                                                                                                                                                                                                                                    end process;
             end process:
                                                                                                                                    121
                                                                                                                                                               wait for clk_period*9;
                                                                                                                                                                                                                                           END:
                                                                                                                                                                                                                                147
```

