```
1   ------------------------------------------------------------------------------
2   --  AMBA APB SLAVE UNIT
3   --  Author: Laurentiu Acasandrei
4   ------------------------------------------------------------------------------
5   --  This program is free software; you can redistribute it and/or modify
6   --  it under the terms of the GNU General Public License as published by
7   --  the Free Software Foundation; either version 2 of the License, or
8   --  (at your option) any later version.
9   --
10  --  This program is distributed in the hope that it will be useful,
11  --  but WITHOUT ANY WARRANTY; without even the implied warranty of
12  --  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13  --  GNU General Public License for more details.
14  --
15  --  You should have received a copy of the GNU General Public License
16  --  along with this program; if not, write to the Free Software
17  --  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
18  ------------------------------------------------------------------------------
19
20  library ieee;
21  use ieee.std_logic_1164.all;
22  use ieee.numeric_std.all; --solving bug 1
23  library grlib;
24  use grlib.amba.all;
25  use grlib.stdlib.all; -- has the same print function
26
27  entity TFG_APB_SLAVE is
28        generic (--fabtech     : integer := 0; -- Compatibility with GRLIB
29                 --memtech     : integer := 0; -- Compatibility with GRLIB
30                 NAHBIRQ     : integer := 32;  -- DEfault 32 interrupts
31                 PINDEX      : integer := 0;
32                 PADDR       : integer := 0;
33                 PMASK       : integer := 16#fff#;
34                 HIRQ        : integer := 10);
35        port (Rst_n    : in  std_ulogic;
36              Clk      : in  std_ulogic;
37              Apbi     : in  apb_slv_in_type;
38              Apbo     : out apb_slv_out_type;
39              --user interface
40              Irq_user : in std_logic;
41              En       : out std_logic;
42              We       : out std_logic;
43              Di       : out std_logic_vector(31 downto 0);
44              Do       : in std_logic_vector(31 downto 0);
45              Addr     : out std_logic_vector(7 downto 0));
46  end;
47
48  architecture behav of TFG_APB_SLAVE is
49
50        constant REVISION : amba_version_type := 0;
51        constant pconfig : apb_config_type := (
52              0 => ahb_device_reg ( 1, 1, 0, REVISION, 0),
53              1 => apb_iobar(paddr, pmask));
54
55        constant ADDR_MASK : std_logic_vector(15 downto 0):= std_logic_vector(
    to_unsigned(pmask, 16));  --solved bug 1
56        type control_type is record
```

```
 57              irq      : std_logic;
 58              ready    : std_logic;
 59              paddr    : std_logic_vector(7 downto 0);
 60              pwdata   : std_logic_vector(31 downto 0);
 61        end record;
 62
 63   --  type delay_input_type is record
 64   --   apbwrite     : std_logic;
 65   --   apben     : std_logic;
 66   --   ready     : std_logic;
 67   --   paddr     : std_logic_vector(7 downto 0);
 68   --   pwdata    : std_logic_vector(31 downto 0);
 69   --  end record;
 70
 71        signal r, rin : control_type;
 72
 73   BEGIN
 74
 75        apbo.pindex  <= pindex;
 76        apbo.pconfig <= pconfig;
 77
 78    --- Gaisler  â€™two-processâ€™ design method
 79    --- http://www.gaisler.com/doc/vhdl2proc.pdf
 80
 81        control_PROCESS : process (r, Rst_n, apbi, Irq_user, do)
 82              variable v        : control_type;
 83              variable apbrdata : std_logic_vector(31 downto 0);
 84              variable irq      : std_logic_vector(NAHBIRQ-1 downto 0);
 85              variable apbwrite : std_logic;
 86              variable apben    : std_logic;
 87              variable paddr    : std_logic_vector(7 downto 0);
 88              variable pwdata   : std_logic_vector(31 downto 0);
 89        begin
 90
 91              v := r;
 92              -- interrupt handling
 93              irq := (others => '0');
 94              irq(hirq) := r.irq;
 95              apbrdata := do;
 96
 97              apbwrite := apbi.psel(pindex) and apbi.pwrite and apbi.penable;
 98              apben  := apbi.psel(pindex);
 99              v.ready  := apbi.psel(pindex) and (not apbi.penable);
100              pwdata    := apbi.pwdata;
101              paddr  := apbi.paddr(9 downto 2) and ADDR_MASK(9 downto 2);    --solved
     bug 1
102              v.irq  := Irq_user;
103
104              ----------------------------------------------------------------------
105              -- Drive process outputs
106              ----------------------------------------------------------------------
107              rin            <= v;
108              apbo.pirq      <= irq;
109              apbo.prdata    <= apbrdata;
110              we             <= apbwrite;
111              en             <= apben;
112              addr           <= paddr;
```

```
113               di             <= pwdata;
114               --apbo.pready    <= r.ready;
115         end process;
116
117         ----------------------------------------------------------------------------
118         -- Registers in system clock domain
119         ----------------------------------------------------------------------------
120         proc_clk : process(Clk)
121         begin
122               if rising_edge(Clk) then
123                     r <= rin;              -- Control
124               end if;
125         end process;
126         ----------------------------------------------------------------------------
127         ----------------------------------------------------------------------------
128
129         -- pragma translate_off
130         bootmsg : report_version
131         generic map ("TFG_APB_SLAVE" &
132         ": Trabajo de Fin de Grado : Hardware AMBA bus IP for image scaling");
133         -- pragma translate_on
134
135   end;
```