```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 29.02.2016 18:42:45
-- Design Name:
-- Module Name: ale_specific - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Location: J:\Ale\Xilinx\MS01\MZ_7010_Basic_System\MZ_Basic_System.srcs\sources_1\new
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ale_specific is
  Port (
    reset                  : in    std_logic;
    clock                  : in    std_logic;
    -- configuration registers
    adr_dest               : in    std_logic_vector (31 downto 0);  -- 20-bit offset in dram area
    allocated by pAxi[2]
    adr_source             : in    std_logic_vector (31 downto 0);  -- 20-bit...
    ctrl                   : in    std_logic_vector (31 downto 0);  -- ctrl(0) starts transfer
    strb_ctrl              : in    std_logic;                       -- edge pulse from cpu write to ctrl
    size                   : in    std_logic_vector (31 downto 0);  -- transfer size
    status                 : out   std_logic_vector (31 downto 0);
    -- local bus
    user_local_bus_address : out   std_logic_vector (19 downto 0);  -- 20-bit offset in dram area
    user_local_bus_busy    : in    std_logic;
    user_local_bus_clock   : in    std_logic;
    user_local_bus_datard  : in    std_logic_vector (31 downto 0);
    user_local_bus_datawr  : out   std_logic_vector (31 downto 0);
    user_local_bus_rd      : out   std_logic;
    user_local_bus_wr      : out   std_logic;
    -- irq
    vhdl_irq               : out   std_logic := '0'
    -- RIK
--  out_rik                : in    std_logic_vector (31 downto 0)
    --pmod_c                   : out   std_logic_vector (7 downto 0);
    --pmod_a3                  : out   std_logic;   -- also goes to LED6 on MS01
    --pmod_a4                  : out   std_logic    --  and LED5
   );
end ale_specific;

architecture logic of ale_specific is

  type   transfer_fsm_state_type is (IDLE, READ, WAIT_READ, WRITE, WAIT_WRITE);
  signal transfer_fsm_state     : transfer_fsm_state_type := IDLE;

  signal start_order      : std_logic := '0';
  signal start_order_1r   : std_logic := '0';
  signal start_order_2r   : std_logic := '0';
  signal busy             : std_logic := '0';
  signal busy_1r          : std_logic := '0';
  signal busy_2r          : std_logic := '0';
  signal tx_pending       : std_logic := '0';
  signal cnt_32bit_words  : unsigned (16 downto 0) := (others => '0');

begin

  ----------------------------------------------
  -- Process
  -- Name         : p_transfer
  ----------------------------------------------
  -- sensitivity list :
  -- local_bus_clock (synchronous process)
  -----------------------------------------------
  p_transfer : process(user_local_bus_clock)
  begin
    if (rising_edge(user_local_bus_clock)) then
      --metastability handling
      start_order    <= ctrl(0) and strb_ctrl;
      start_order_1r <= start_order;
      start_order_2r <= start_order_1r;

      busy    <= user_local_bus_busy;
      busy_1r <= busy;
      busy_2r <= busy_1r;


      -------------------------------------------------------------------
      -- Dummy state machine for transfers
      -------------------------------------------------------------------
      case transfer_fsm_state is

        when IDLE =>
          -- state evolution
          if ((start_order_2r='0' and start_order_1r='1') and busy_1r = '0') then
            transfer_fsm_state <= READ;
          end if;
          -- outputs of fsm
          tx_pending               <= '0';
          cnt_32bit_words          <= (others => '0');
          vhdl_irq                 <= '0';


        when READ =>
          -- state evolution
          transfer_fsm_state      <= WAIT_READ;
          -- outputs of fsm
          tx_pending               <= '1';
          user_local_bus_rd        <= '1';
          user_local_bus_address  <= std_logic_vector((unsigned(adr_source(19 downto 0)) + ('0'&
          cnt_32bit_words&"00") ));

        when WAIT_READ =>
          -- state evolution
          if (busy_1r = '0' and busy_2r='1') then
            transfer_fsm_state    <= WRITE;
          end if;
          -- outputs of fsm
          user_local_bus_rd        <= '0';

        when WRITE =>
          -- state evolution
          transfer_fsm_state      <= WAIT_WRITE;
          -- outputs of fsm
          user_local_bus_datawr   <= user_local_bus_datard;
          user_local_bus_wr        <= '1';
          user_local_bus_address  <= std_logic_vector((unsigned(adr_dest(19 downto 0)) + ('0'&
          cnt_32bit_words&"00") ));
          cnt_32bit_words          <= cnt_32bit_words +1;

        when WAIT_WRITE =>
          -- state evolution
          if (busy_1r = '0' and busy_2r='1') then
            if (cnt_32bit_words = unsigned(size(18 downto 2))) then
              transfer_fsm_state  <= IDLE;
              vhdl_irq            <= '1';
            else
              transfer_fsm_state  <= READ;
              vhdl_irq            <= '0';
            end if;
          end if;
          -- outputs of fsm
          user_local_bus_wr       <= '0';

        when others => transfer_fsm_state <= IDLE;

      end case;

    end if;
  end process p_transfer;

  status(0) <= tx_pending;

  --pmod_c(7 downto 0) <= out_rik(7 downto 0);  -- Just loop to physical I/O for the moment
  --pmod_a3 <= out_rik(0);  -- LED6 on PCB
  --pmod_a4 <= out_rik(1);  -- LED6 on PCB


end logic;
```