```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity Stream2ToDDR is
   port(
      --------------------
      -- System Signals --
      --------------------
      sysclk                 : in std_logic; -- 100 MHz system clock.
      reset                  : in std_logic;
      DPRAMBaseAddr          : in std_logic_vector(31 downto 0);

      -----------------------------------
      -- DPRAM Data From ECU to SoC --
      -----------------------------------
      Stream2_RxValid        : in std_logic;
      Stream2_Data           : in std_logic_vector(48 downto 0);

      -----------------------------
      -- Avalon Master Interface --
      -----------------------------
      avm_master_waitrequest  : in std_logic;
      avm_master_write        : out std_logic;
      avm_master_address      : out std_logic_vector(29 downto 0);
      avm_master_writedata    : out std_logic_vector(31 downto 0);
      avm_master_byteenable   : out std_logic_vector(3 downto 0);
      avm_master_burstcount   : out std_logic_vector(7 downto 0)
   );
end entity;

architecture rtl of Stream2ToDDR is


   ---------------------------
   -- State Machine Signals --
   ---------------------------
   type SM_type is (IDLE, ASSERT_WRITE);
   signal SM_State              : SM_type;

   signal DPRAMBaseAddrValid           : std_logic;
      signal DPRAMWriteAddrValid       : std_logic;
   signal ECUPrivateRAMWen             : std_logic;
   signal DPRAMWriteData               : std_logic_vector(31 downto 0);
   signal ECUPrivateRAMWrData          : std_logic_vector(31 downto 0);
   signal DPRAMWriteAddr               : std_logic_vector(16 downto 0);

      signal s_avm_master_address      : std_logic_vector(29 downto 0);
   signal s_avm_master_writedata    : std_logic_vector(31 downto 0);


begin

   avm_master_byteenable <= "1111";
   avm_master_burstcount <= x"01";

   DPRAMWriteData <= Stream2_Data(31 downto 0);
   DPRAMWriteAddr <= Stream2_Data(48 downto 32);

   ------------------------
   -- DDR Memory Control --
   ------------------------
   DDRCtrl:process(reset, sysclk)
   begin
      if reset = '1' then
         SM_State <= IDLE;
        avm_master_write <= '0';
        s_avm_master_writedata <= (others => '0');
        s_avm_master_address <= (others => '0');
      elsif rising_edge(sysclk) then

         case SM_State is

      -- When the Data and address are valid, enter the address and data values and be prepared for writting
```

```
            when IDLE =>
                    avm_master_write <= '0';
                    -- DPRAMBaseAddr(0) and DPRAMWriteAddr(16) are constantly '1' and '0' respectively
               if Stream2_RxValid = '1' and DPRAMBaseAddr(0) = '1' and DPRAMWriteAddr(16) = '0' then
                  SM_State <= ASSERT_WRITE;
                    s_avm_master_writedata <= DPRAMWriteData;
                    s_avm_master_address <= DPRAMBaseAddr(31 downto 2) + DPRAMWriteAddr(15 downto 0);
                    avm_master_write <= '1';
               end if;
           --------------------------------------------------------------------------
                    -- Write to SDRAM HPS --
            when ASSERT_WRITE =>
                    if avm_master_waitrequest /= '1' then
                      avm_master_write <= '0';
                     SM_State <= IDLE;              -- when "waitrequest" de-asserts look for the next valid address/data
                    end if;
       --------------------------------------------------------------------------
            when others =>
             SM_State <= IDLE;
                    avm_master_write <= '0';
                    s_avm_master_writedata <= (others => '0');
                    s_avm_master_address <= (others => '0');
         end case;

       end if;
    end process;

       avm_master_writedata <= s_avm_master_writedata;
       avm_master_address <= s_avm_master_address;

end rtl;
```