ZLAN-88

UTOPIA Interface Between MT90222/3/4 and Specific ATM Controllers

Application Note

March 2004

Contents

1.0 Introduction

2.0 Background

2.1 Compatible Type of ATM Controllers

2.2 ATM Controllers Which Support Back-to-Back Mode

3.0 Interface Options

3.1 Using the Priority Polling in the ATM Controller

3.2 The Simple CPLD Interface Between ATM Controller and MT90223/2

3.2.1 Sample Programming of TC Links

3.3 Interface Considerations When Using MT90224

3.4 ZL30226/7/8 and MT90225/6 Support

4.0 VHDL Code for the CPLD

1.0 Introduction

The purpose of this application note is to allow users of MT90222/3/4 to connect to a specific type of ATM controller on UTOPIA interface. One such implementation is the MPC8260 also known as Power QUICC II - PQII

2.0 Background

MT90222/3/4 device is connecting to the UTOPIA bus as an ATM slave (PHY). This application note assumes that the ATM controller depicted in Figure 1 is the master on UTOPIA bus and that it is generating and receiving traffic to/from MT90222/3/4.

The UTOPIA bus handshaking is defined in the AF-PHY-0039-000 standard. The user should refer to this specification for detailed signal description. For the transmit communication in the direction from the ATM controller towards the PHY both chips communicate as described in the standard. In the receive direction, there might be some interoperability issues. Whether those will occur depends on the implementation of ATM controller. This application note describes in details the observed behaviour and the solution.

The definition of the UTOPIA spec is such that it leaves freedom to the design of ATM master to implement when RxEn* is deasserted at the end of cell transfer. Depending on when the RxEn* is deasserted, there may be a need for glue logic circuitry or software changes.

2.1 Compatible Type of ATM Controllers

Certain ATM controllers vendors implement the deassertion of the RxEn* at the very end of the cell transmission during the clock cycle 53 (or cycle 27 for 16-bit mode) of the cell transmission as shown in Figure 4.4 of the UTOPIA spec. In this case the UTOPIA transfer is as described in the specification and there is no need for adding the glue logic circuitry or implementing specific polling schemes (this is detailed in paragraph 3). One such ATM controller implementation is Agere APC device.

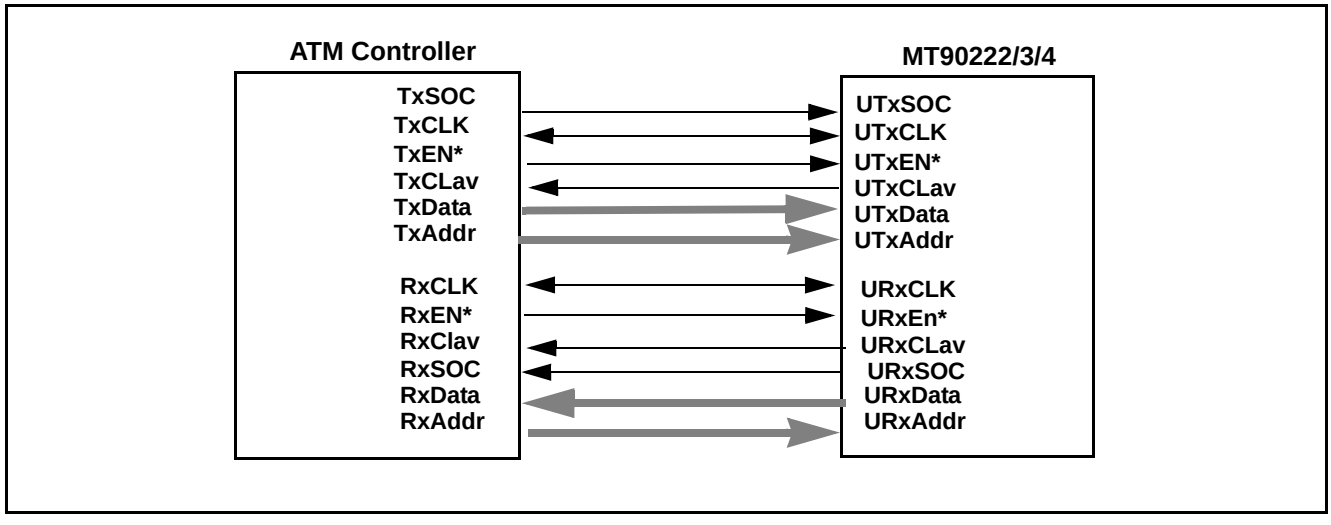

Figure 1 - UTOPIA Interface Between an ATM Controller and MT90222/3/4

2.2 ATM Controllers Which Support Back-to-Back Mode

Some vendors of ATM controllers are choosing to implement the UTOPIA interface in such way that if there are no more cells available to receive, their device deasserts the RxEn* signal few clock cycles after the end of the transfer of the cell data (as shown on Figure 4.5 of the af-phy-0039-000). Motorola's PQII ATM controller implementation is one of those. It deasserts the RxEn* during the clock cycle 55 (in case of 8-bit mode). That may cause the MT90222/3/4 to get into the situation where the ATM cells will be skewed. Therefore, adjustments need to be made in order to ensure error free communication between MT90222/3/4 and ATM controller on Rx UTOPIA side. The options are listed in the next section.

3.0 Interface Options

3.1 Using the Priority Polling in the ATM Controller

In order to overcome the interoperability issue on the Rx side, one option is to use the priority polling. The priority polling is implemented in such a way where the polling always starts from the first PHY in the polling list and finishes with the last PHY in the polling list. When the PHY is selected then the polling will start again from the first PHY. The polling list of ATM controller should be defined in the following manner. It should start with active addresses and then it should be padded with inactive PHY addresses on UTOPIA bus. For example, if there are four active groups on UTOPIA bus, these should be programmed first and then the padding will continue with inactive addresses. The goal of padding with inactive addresses on UTOPIA bus is to make sure that at the end of cell transmission during the ATM cell byte 53 (payload byte 48) for 8-bit bus or ATM word 27 (for 16-bit bus) and after the end of the cell data, while the RxEn* is low, no active PHYs are responding with the cell available signal - RxCLAV. This is depicted in Figure 2 for 8-bit bus.

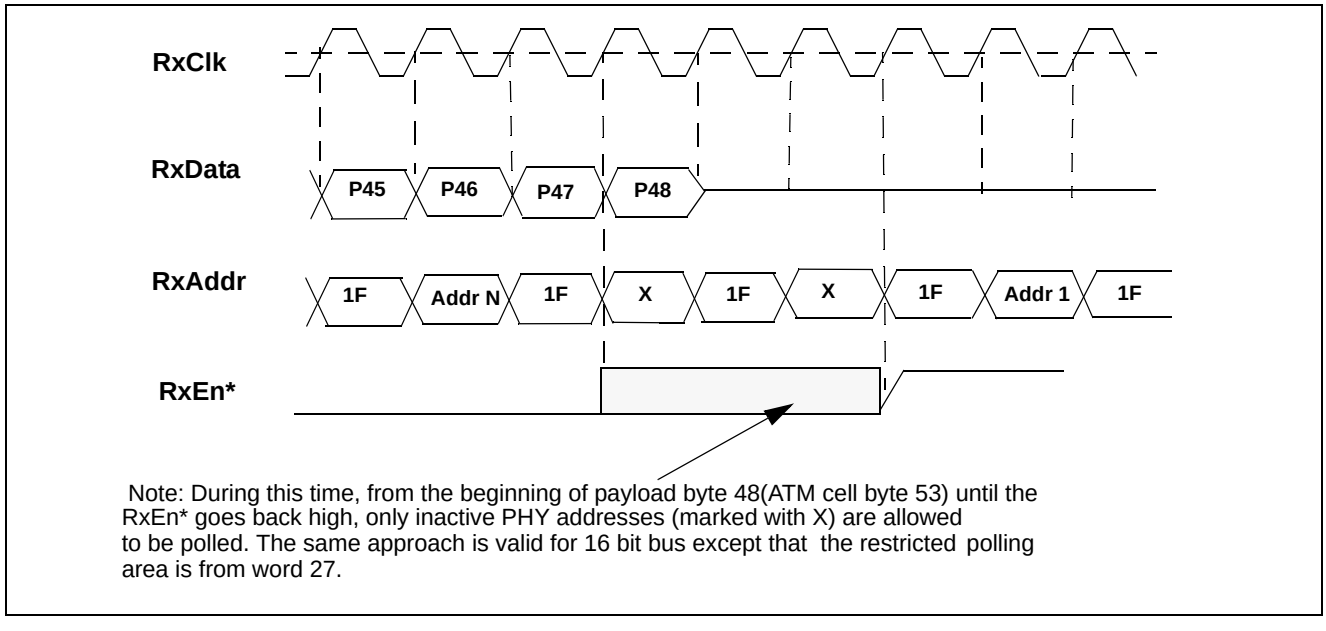

Figure 2 - Polling restriction at the end of the transfer of ATM Cell

Another assumption made for priority polling to be used is that the RxEn* is deasserted by the ATM master not more than few clock cycles after the end of the cell. In different ATM controller architectures it might be possible that the RxEn* stays asserted longer than just two cycles after the end of cell transmission. In this situation, the priority polling can still be used only to make sure that there are no active addresses to be polled during the last cycles of the cell transmission. If RxEn* stays low for more than few clock cycles after the end of the cell transmission - for unknown amount of time or indefinitely, then the priority polling is not enough. A simple CPLD glue logic needs to be implemented as it is discussed in the chapter below.

3.2 The Simple CPLD Interface Between ATM Controller and MT90223/2

Another option to overcome interoperability issues is to interface the ATM controller (from section 2.2) with MT90223/2 with the help of a simple CPLD as shown in Figure 3.

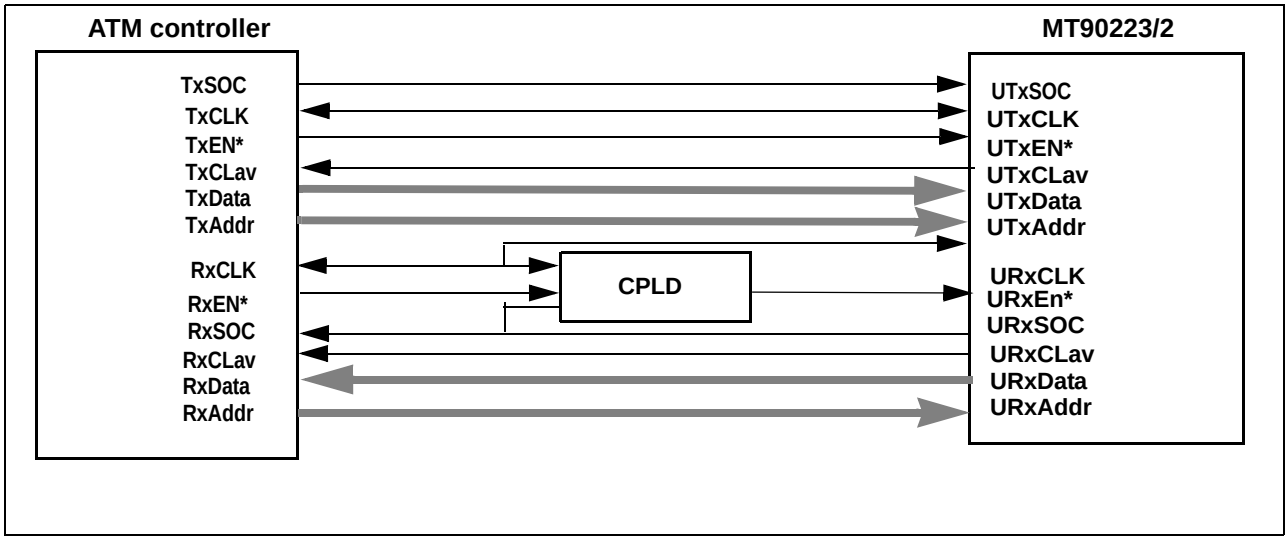

Figure 3 - Glue Logic on UTOPIA Interface Between Specific ATM Controller and MT90223/2 Family

This glue logic overcomes the obstacle where the ATM controller may choose to keep RxEn* low indefinitely after the transfer of one cell in the case that there are no other cells ready to be received. The inputs to glue logic are RxEn*, RxSOC and RxCLK and the output is URxEn*. There is also a selection pin to define whether it is 8-bit or 16-bit UTOPIA bus.

The function of the CPLD is to overwrite the RxEn* from ATM controller towards the MT90223/2 in case that at the end of the transfer of the cell, RxEn* stays asserted for longer. During the time when the RxEn* is low (please refer to Figure 2) and there are no other PHYs ready to send the cells, the last selected PHY port would always stay selected. This is because, once the RxEN* is overwritten after 53 data byte (27data word) of the cell, the MT90223/2 will always see the URxEn* inactive (while the RxEn* is actually asserted) therefore preventing the transmission of any further cells. In order to avoid that, the idea is to generate idle traffic using dummy PHYs, so there will always be cell ready for transmission by the polled dummy PHY. To achieve this, the user should enable special test features of MT90223/2.

The test functionality within MT90222 and MT90223 allows programming of up to 8 TC links in addition to the used number of links/groups. Depending on whether MT90222 or MT90223 is used, either four or eight of these extra links need to be programmed. These TC links can only work in digital loopback. Their sole purpose is to generate idle cells continuously. By generating this idle traffic, it is ensured that RxEn* is not kept low indefinitely by ATM controller, as at the end of the cell reception, there will always be another cell ready to transfer. The priority polling scheme described in the previous section must still be used and the ATM contoller must include these dummy PHYs in its polling list. This idle traffic must be discarded on the ATM controller side. Due to ample bandwidth of the UTOPIA bus, there will be no impact on user traffic. The sample programming of 8 TC links to transmit only idle cells in digital loopback is described below. For detailed description on how to program the TC links, please refer to Technical Note - MT90225/6 Programming Manual.

The VHDL code for the CPLD is presented in the Section 4.0.

3.2.1 Sample Programming of TC Links

```
  /* This is the example programming of 8 dummy TC links for MT90223. With this
feature the 8 additional dummy TC links will transmit idle cells all the time. The
cells will be passed to Rx UTOPIA interface. This dummy active links are the part of
testing feature and can only be used in digital loopback. Those links are
1,3,5,7,9,11,13,15 for MT90223 and 1,5,9,13 for MT90222. For MT90222 only four dummy
TC links need to be programmed. */


  void Initialization(void) // resets and initializes the device
  {
     int16 link;
     int16  i;
      // soft reset
      write_reg( 0x0299, 0x00E0 );
      // Set link FIFO size, 4 in this example
     for (i=0; i<8; i++) write_reg( 0x008B + i, 0x0404 );
     // set Utopia mode, 16-bit in this example
     write_reg( 0x0011, 0 );
     write_reg( 0x0052, 0 );
     /* Set Tx and Rx UTOPIA PHY addresses */
     for (i=1; i<8; i+=2)  {
         write_reg( 0x0040 + i, 0x0800 + 0x0101 * i );
         write_reg( 0x0000 + i, 0x0800 + 0x0101 * i );
      }
      /* Tx TDM */
      for (link=1;  link<16; link +=2 )  {
          // Tx TDM mapping, 0xFFFF to enable all timeslots
           write_reg( 0x0610 + link, 0xFFFF );
           write_reg( 0x0620 + link, 0xFFFF );
            // Config Tx TDM link - Generic TDM mode 2.048Mbps
           write_reg( 0x0600 + link, 0x48A4 );
     }
      for (i=1; i<8; i+=2) write_reg( 0x0318 + i, 0x0808 );    //Tx link control
      write_reg( 0x0741, 0x0036 );  // Rx sync
      for (link=1; link<16; link+=2)    {
          // Rx TDM mapping, 0xFFFF to enable all timeslots
          write_reg( 0x0710 + link, 0xFFFF );
          write_reg( 0x0720 + link, 0xFFFF );
          // Config Rx TDM link - Generic TDM mode 2.048Mbps, digital loopback
          write_reg( 0x0700 + link, 0x01A4 );
       }
       /* Rx link control  - set to TC mode, do not discard the idle cells*/
```

```
      for (i = 1; i < 8;  i += 2)     write_reg( 0x00C0 + i, 0x9090 );
      //Enable the RX UTOPIA
      write_reg( 0x0010, 0x00FF);
  }
```

3.3 Interface Considerations When Using MT90224

For the applications where MT90224 is used, it is best that customer choose an ATM controller that has behavior described in section 2.1. If some other type of ATM controller is chosen (as described in 2.2), then the MT90224 can still be used, but the priority polling must be supported by the ATM controller. Also, the ATM controller needs to deassert the RxEn* after the end of the cell in the finite amount of time (known number of clock cycles - as described in figure 4.5 of the UTOPIA spec). The polling list needs to be padded with inactive addresses so that during the period while the RxEN* is low, during the last byte/word of the cell and after the end of the cell, no active addresses are polled. The polling of active addresses should resume when RxEN* goes back to high.

With suggested polling scheme, if there is only one MT90224 on UTOPIA bus, then the maximum number of PHY addresses to be used is 16. If more than one MT90224 is connected to UTOPIA bus, then the number of PHY addresses per one MT90224 is limited to 15 - as one UTOPIA address is chosen to be dummy PHY address.

3.4 ZL30226/7/8 and MT90225/6 Support

This application note applies to ZL30226/7/8 and MT90225/6. The problem description and solutions described for MT90224 also refer to MT90225 and ZL30228. The same way, MT90223 is cross-referenced with ZL30227 and MT90226 as well as MT90222 is cross-referenced with ZL30226.

4.0 VHDL Code for the CPLD

```
  -- Description: This block emulates a CPLD being proposed.  This CPLD would pull
enable high as soon as a cell is completely transmitted
  -- Notes:  This solution assumes that the timing of the RXEn* signal will still
meet the  UTOPIA spec after passing through the CPLD


  library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

  LIBRARY COMMON;
    USE COMMON.convert_Std_Ulogic.ALL;
    USE COMMON.modules.ALL;
    USE COMMON.common_package.ALL;

  entity cpld_waround3 is
    port   (
      uto_clk :             in  std_ulogic;
      n_enb_in :            in  std_ulogic;
      eight_bit :           in  std_ulogic;
      n_mrst :              in  std_ulogic;
      uto_in_soc:           in std_ulogic;
      n_enb_one_cell_only : out std_ulogic
    );
```

```
end cpld_waround3;

architecture lan of cpld_waround3 is

 SIGNAL s_pull_enable_high : Std_Ulogic;
 SIGNAL s_counter1 : Std_Ulogic_Vector(5 downto 0);

begin
 pull_enable_high : PROCESS(n_enb_in, s_pull_enable_high) is
  BEGIN
      n_enb_one_cell_only <= n_enb_in OR s_pull_enable_high;
      --n_enb_one_cell_only <= n_enb_in;
      -- to bypass this block uncomment this line and comment line above
      -- end process pull_enable_high;
sync : process
  BEGIN
  wait until uto_clk'event and uto_clk = '1';
  if (n_mrst ='0') THEN
     s_counter1 <= "000000";
     s_pull_enable_high <= '0';
     elsif(eight_bit = '1')THEN
      if((n_enb_in = '1') and ((s_counter1 < "000001") or (s_counter1 ="110101" ))) THEN
             -- we want to transmit RxEn* to the  output,
             --unless RxEn* is low and more   than 53
             -- bytes have been transmitted.
              s_counter1 <= "000000";
              s_pull_enable_high <= '0';
            -- next elsif added to support case of late SOC
              elsif(n_enb_in = '0' and uto_in_soc = '1') then
                  s_counter1 <= "000010";
                  s_pull_enable_high <= '0';
             elsif(n_enb_in = '0' and s_counter1 = "110100") THEN
                 s_counter1 <= to_stdulogicvector(unsigned(s_counter1) + '1');
                 s_pull_enable_high <= '1';
        elsif(n_enb_in = '0' and s_counter1 = "110101") THEN
                  s_counter1 <= s_counter1;
                  s_pull_enable_high <= '1';
        elsif(n_enb_in = '0' and s_counter1 < "110100") THEN
                     --s_counter1 <=  s_counter1 + "000001";
                     s_counter1 <= to_stdulogicvector(unsigned(s_counter1) + '1');
                     s_pull_enable_high <= '0';
        else
          s_counter1 <=  s_counter1;
          s_pull_enable_high <= '0';
        end if;
        else -- 16-bit case
        if(n_enb_in = '1') and ((s_counter1 < "000001") or (s_counter1 = "011011" )) THEN
```

```
        -- we  want to transmit enable to the  output, unless RxEn* is low and
       --more  than 27 bytes have been transmitted.
                    s_counter1 <= "000000";
                     s_pull_enable_high <= '0';
              -- next elsif added to support case of late SOC
              elsif(n_enb_in = '0' and uto_in_soc = '1') then
                    s_counter1 <= "000010";
                    s_pull_enable_high <= '0';
              elsif(n_enb_in = '0' and s_counter1 = "011010") THEN
                     s_counter1 <= to_stdulogicvector(unsigned(s_counter1) + '1');
                     s_pull_enable_high <= '1';
              elsif(n_enb_in = '0' and s_counter1 = "011011") THEN
                       s_counter1 <=  s_counter1;
                       s_pull_enable_high <= '1';
              elsif(n_enb_in = '0' and s_counter1 < "011010") THEN
                    --s_counter1 <=  s_counter1 + "000001";
                    s_counter1 <= to_stdulogicvector(unsigned(s_counter1) + '1');
                       s_pull_enable_high <= '0';
               else
                       s_counter1 <=  s_counter1;
                       s_pull_enable_high <= '0';
            end if;
            end if;
end process sync;
end lan;
```

For more information about all Zarlink products
visit our Web Site at

www.zarlink.com

TECHNICAL DOCUMENTATION - NOT FOR RESALE