Implementing an ADSL to USB Interface Using Spartan Devices

XAPP171 March 26, 1999 (Version 1.0) Application Note

Summary

This application note illustrates the use of Spartan devices in an ADSL modem. The Spartan device is used to implement the complex system level glue logic required for the modem's USB interface and manages DMA transfers of ATM cells. The design example shows how cost effective a Spartan device can be in these applications. While the design is targeted at solving a specific problem, it illustrates solutions to a number of general technical issues. These include implementing Utopia interfaces for ATM devices and remote configuration of Spartan devices. The following topics are covered:

- Introduction
- Overview
- Alcatel DynaMiTe Chipset
- National Semiconductor USBN9602
- Microcontroller Selection
- Spartan Device Selection
- Interface Architecture
- Conclusion

Xilinx Products

Spartan FPGAs

Introduction

This application note illustrates the use of Spartan devices in an Asymmetric Digital Subscriber Line (ADSL) modem. In this application the Spartan device is used to implement the complex system level glue logic required for the modem's Universal Serial Bus (USB) interface. The Spartan device sits between the CPU, USB interface controller, and the ADSL modem, and manages DMA transfers of ATM cells. The design example shows how cost effective a Spartan device can be in these applications.

ADSL technology can expand the useable bandwidth of existing copper telephone lines, delivering high speed data communications at rates of up to 8 Mbps. The recent G.Lite standard allows for a lower-speed, lower-cost implementation. The USB interface is becoming the standard on both PC and Macintosh-based platforms. This document assumes that the reader is familiar with ADSL, USB and ATM technologies.

While the design is targeted at solving a specific problem, interfacing an ADSL chipset to USB, it illustrates solutions to a number of general technical issues. These include implementing Utopia interfaces for ATM devices and remote configuration of Spartan devices.

Overview

The design objectives for this application were threefold. First was the creation of a solution with the lowest possible cost. In this case the target was a semiconductor bill of materials for the USB interface that is significantly less than $10 in volume.

The second objective was to deliver a solution that would deliver the best possible performance. Current solutions are able to deliver 2 to 3 Mbps of bandwidth across USB at a much higher price point. The minimum target for this design was to be able to support the full 1.5 Mbps data rate of G.Lite and at the same time get as close to the full G.992.2, 6.1 Mbps data rate.

The third objective was to configure the Spartan device from the host via the USB interface. This has the dual benefit of eliminating the requirement for FPGA configuration memory in the modem and the ability to update the configuration in the field.

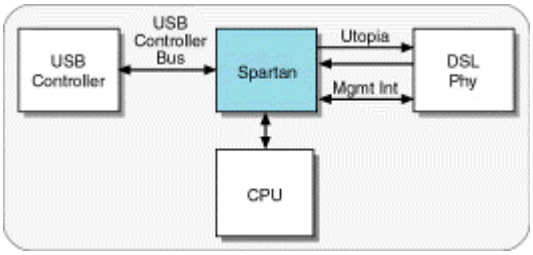

Figure 1: ADSL Modem System Block Diagram

Before we examine the functions implemented in the Spartan device, let's get a overview of the devices that were chosen for the ADSL PHY, USB controller, and CPU blocks.

Alcatel DynaMiTe Chipset

The ADSL chipset chosen for this design is the Alcatel DynaMiTe chipset. This three chip set consists of the MT-20134 Analog Front End, the MT-20135 ADSL Modem and ATM Framer, and the MT-20136 ADSL Transceiver Controller. Of these devices the Spartan device interfaces to the MT-20135, and MT-20136.

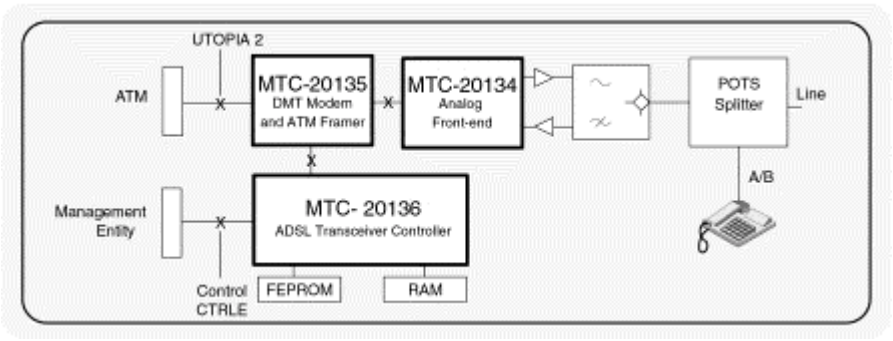

Figure 2: Alcatel DynaMiTe Chipset Block Diagram

Figure Courtesy Alcatel Microelectronics

The MT-20135 sends and receives data in the form of ATM over a standard Utopia level 2 interface. This interface was defined to transfer ATM cells between physical layer devices like the MT-20135 and segmentation and re-assembly (SAR) controllers or ATM switching fabrics. The function of the Spartan device in this design is to handle the handshaking required to convert the full duplex ATM cell stream from the Utopia interface into half-duplex transfers to and from the USB controller. Since both the MT-20135 and the USB controller have internal FIFOs capable of storing a complete ATM cell, the transfers consist of moving data one cell at a time between these FIFOs.

The MT-20135 supports non-ATM transport schemes by way of another interface referred to as the Synchronous Link Access Protocol (SLAP) interface. The SLAP interface is not used in this application.

The MT-20136 contains an ARM processor, a memory controller, and an interface bus that is connected directly to the MT-20135. The MT-20136 is essentially a single chip processor dedicated to managing the ADSL modem. The Spartan device interfaces to this device via a specialized management interface that Alcatel refers to as CTRLE. This interface is used to control the modem and to query status.

National Semiconductor USBN9602

The USB interface in the design is based on a National Semiconductor USBN9602 controller. This device, packaged in a 28 pin SOIC package, contains all of the logic necessary to transfer data frames to and from the host with minimal processor intervention. It contains several endpoint FIFOs, two of which are 64 bytes deep. These FIFOs are large enough to contain a complete 53-byte ATM cell plus a control header if required.

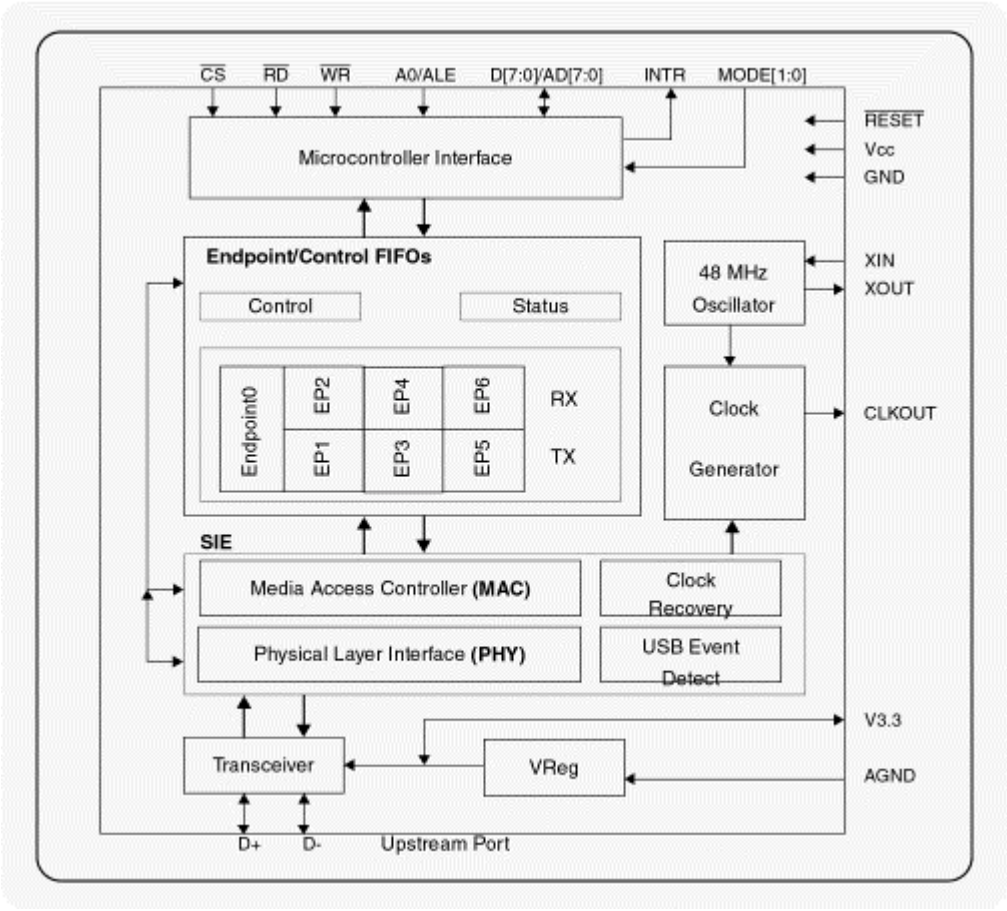

Figure 3: National USBN9602 Block Diagram

Figure Courtesy National Semiconductor

The USB interface could be implemented in a Xilinx FPGA, but for this application the dedicated USB chip provides the most cost-effective solution.

In this design the Spartan device contains logic to support two primary functions. The first of these is DMA logic that manages the transfer of ATM cells between the USBN9602 and the Utopia interface.

The second function is arbitrating access to the USBN9602 by an 8051 microcontroller. Although the bus on the USBN9602 is designed for direct attachment to the 8051, doing so would eliminate the ability of the Spartan device to directly transfer cells to and from the USBN9602. This is because there is no mechanism available to force the 8051 to relinquish control of the bus when the DMA logic in the Spartan device needs to perform a transfer. Microcontrollers with this feature are significantly more expensive than the 8051.

Microcontroller Selection

The microcontroller chosen for this design was an 80C51. This device is available from multiple vendors and is very inexpensive. Aside from low cost, the features that were required for this application were as follows:

- On chip ROM and RAM: to reduce parts count.
- External memory bus: to facilitate reading and writing registers in the USBN9602 and MT-20135.
- 13 general-purpose I/O pins: used for accessing the USBN9602 before the Spartan device is configured, more on this in the following section.

Although the 8051 microcontroller meets these requirements, a variety of other devices such as the Motorola 68HC11 could have been used.

The performance that is required of the microcontroller is minimal. This is due to the fact that the Spartan device handles the real-time transfer of ATM cells between the ADSL chipset and the USB interface. The functions handled by the microcontroller include initializing the Spartan device at startup and responding to status queries from the host received via USB messages.

Spartan Device Selection

Spartan devices are available in a range of densities and packages. The following criteria were used to select the device used in this application:

- I/O Pins
  - The design requires a total of 60 pins.
- Voltage
  - The design operates at 3.3V.
- Density
  - The estimated size of the design is 8K gates, broken down as follows: 4K for the USB controller interface, 1K each for the Utopia Tx and Rx state machines, 1K for the microcontroller interface and 1K for the remaining logic.
- Performance
  - The highest clock speed used in the device is 48 MHz, used to clock the USB interface bus state machine. The remaining logic runs at 3 MHz, the Utopia data rate, or 16 MHz or less for the microcontroller.
- Packaging
  - The size constraints imposed on most modem designs dictates a high density surface mount package.

Based on these criteria the device selected for this design is the XCS10XL-4VQ100C. This device offers 10K gate density, 3.3V operation with 5V compatibility, 77 user I/O, and is packaged in a space saving VQ100 package. The -4 speed grade is sufficient for this design's performance requirements.

Spartan Device Configuration

As was mentioned earlier, a design goal was the ability to configure the Spartan device via the USB interface. This is accomplished by using the microcontroller to load configuration data into the Spartan device using slave serial mode. The data comes from messages received from the host via the USB interface. On the host side the configuration data image is stored in a data structure in the device driver. The advantages of this approach are that the modem does not need dedicated configuration storage for the Spartan device and updating its configuration is accomplished by distributing an updated device driver.

Since the microcontroller does not normally interact directly with the USBN9602, an alternate mechanism is needed to let the microcontroller receive configuration data during startup. This is accomplished by connecting thirteen general purpose I/O pins on the microcontroller directly to the I/O bus on the USBN9602. At startup the USB interface pins on the Spartan device are tri-stated and the microcontroller can read and write registers in the USBN9602 by manipulating the I/O port pins. When the Spartan device has been configured and is ready to take control of the USB interface the microcontroller reconfigures these port pins as inputs to avoid contention.

Interface Architecture

Figure 4 illustrates how all of this fits together. Sitting in the middle of the interface, the Spartan device is truly the glue that pulls the whole thing together. The design lets the USB controller and the ADSL modem directly interchange ATM cells with no intervention from the microcontroller. At the same time the micro- controller has access to control and status registers in all of the devices. This allows the software in the host driver to directly manage the modem. This means the software on the microcontroller only needs to implement "peek" and "poke" level functions.

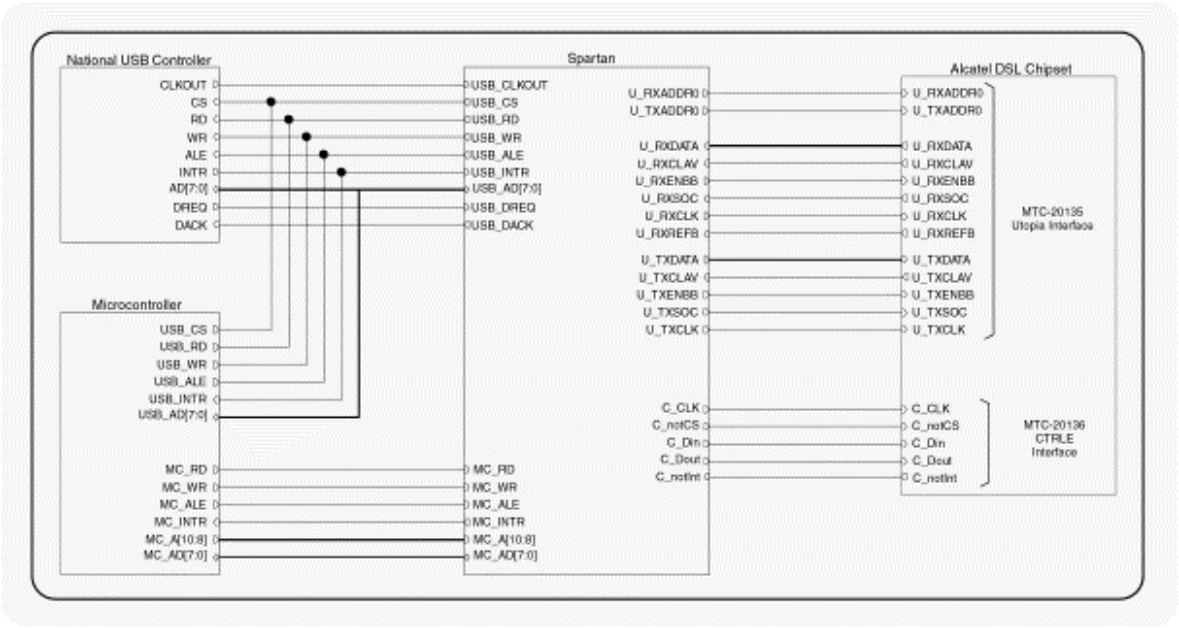

Figure 4: USB Interface Connections

Spartan Device Implementation

Figure 5 shows the internal architecture implemented in the Spartan device for this application.

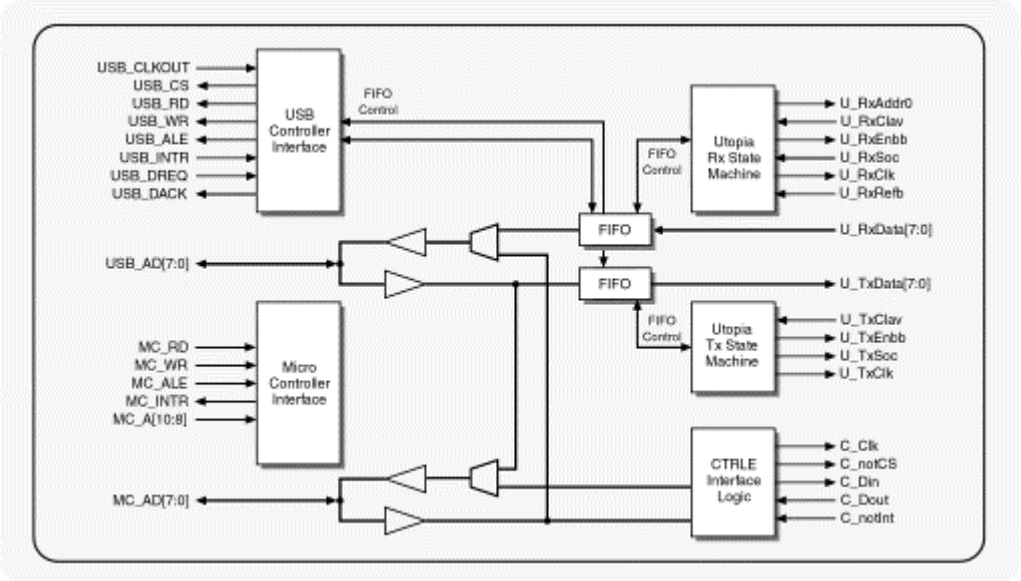

Figure 5: FPGA Logic Block Diagram

The Utopia Tx and Rx state machines interact with the USB interface by way of small (2 word) 9 bit wide FIFOs, which we will refer to as interface FIFOs. The ninth bit is set to mark the start of a cell during transfers and reflects the state of the Tx and Rx Start Of Cell (SOC) indicators on the Utopia interface. The use of FIFOs provides a simple means of abstracting the interface between the Utopia state machines and the USB controller interface state machine. The SpartanXL architecture's SelectRAM memory is used for the creation of the FIFOs.

The Utopia Rx State machine continually polls the 20135 to determine if there are any cells in its Rx FIFO. If data is present it signals this to the USB controller interface and waits for it to indicate that it is ready to transfer a cell to the USB controller. Upon receiving this indication it transfers the cell to the Rx FIFO one byte at a time. Since the 20135's Utopia interface does not support octet-level handshaking, the transfer of bytes into the Rx FIFO occurs at a rate of one per U_RxClk without interruption.

In order to minimize the depth of the Rx FIFO it is important that the cell transfer rates across the Utopia interface and the USB interface are matched. This can be accomplished by dividing the clock used to sequence the USB interface state machine by the number of clocks required to perform the transfer. Assuming a 48 MHz system clock, analysis has shown that each transfer to or from the USB controller can be done in sixteen clock cycles. Dividing the system clock by sixteen to generate the Utopia U_RxClk will match the data rates in this case. Note that this results in a cell transfer rate of 24 Mbps, twice the USB data rate:

1 / ((1 / 48 MHz) * 16 clocks per transfer) * 8 bits per transfer

The Utopia Tx State machine continually polls the 20135 to determine if there is space for a new cell in its Tx FIFO. When this occurs it signals the USB controller interface that it is ready to start a transfer. When the USB controller interface is prepared to transfer a cell it starts loading the Tx FIFO. When the Utopia Tx state machine detects the assertion of the Tx FIFO full flag it starts transferring the cell data across the Utopia interface.

The USB interface arbitrates between the three functions that need to transfer data across the USBN9602 I/O bus. These are ATM cell read operations, ATM cell write operations, and read or write operations initiated by the microcontroller. The transfer of received cells gets the highest priority due to its higher data rate and the limited amount of cell buffering in the ADSL interface. The transmission of cells gets the next level of priority. Microcontroller accesses are interleaved between cell transfers.

The microcontroller interface includes logic for latching the address, and decoding address ranges for the USBN9602 and CTRLE interfaces. In addition it contains a command register for initiating read and write cycles to the USBN9602. A command response structure is needed due to the long latencies involved in accessing the registers in this device. Recall that the microcontroller may have to wait for the DMA logic to complete the transfer an entire ATM cell before getting access to the USBN9602. Using this arrangement the microcontroller would issue a read command for a specific register in the USBN9602 and then wait until the read operation has completed before reading the read data from a result register. Setting a bit in a status register would indicate completion.

Conclusion

The design that has been outlined meets each of the three original design objectives.

The cost of this solution is well within the design target, coming in at less than $8 when purchased in 100k quantities. The cost of the complete semiconductor solution is less than $80 of which the Alcatel chipset represents almost 90% of the cost. Note that the street price of the Alcatel chipset is well below the budgetary number quoted here.

Supplier	Part Number	Description	Qty	Unit Cost
Alcatel	MTK-20131	DynaMiTe DSL modem chipset	1	$60-70
National	USBN9602	USB Full Speed Function Controller	1	$1-2
Xilinx	XCS10XL-4VQ100C	Spartan FPGA	1	$3-4
Philips	SC80C51BCCB44	8-bit microcontroller	1	$2-3
			Total	$66-79

Notes: Prices are based on budgetary quotes at 100K volume.
Includes semiconductor content only.

In terms of performance, the design is capable of transferring cells at twice the line rate. While this is just one factor in the overall system performance, we believe that the current limitation in the USB host controllers will be the limitation for system level performance.

The third objective of in system configuration was met by supporting the transfer of configuration data across the USB interface. This capability should prove valuable in this and other applications where standard practice if not the standard itself is in a state of flux.

References

- MTC-20135 ADSL DMT Transceiver with ATM Framer Data Sheet, Rev. 2 – April 1998, Alcatel Microelectronics
- MTC-20136 ADSL Transceiver Controller Data Sheet, Rev. 2 – May 1998, Alcatel Microelectronics
- USBN9602 Full Speed Function Controller With DMA Support Data Sheet, May 1998, National Semiconductor
- 80C51 8-bit Microcontroller Family Data Sheet, Oct 1998, Philips Semiconductor
- UTOPIA Specification Level 1 (40 KB), Version 2.01, af-phy-0017.000, March 21, 1994, The ATM Forum