WP146 (v1.0) May 8, 2001

UART to 1394 Bridging for Bluetooth

Author: Saeid Mousavi

Introduction

The distribution of video, audio, and PC data has evolved using a variety of different networking technologies. Various factors drive these applications including cost, performance, quality of service required, regulatory issues, geographic building trends, etc. Therefore, Ethernet, IEEE 1394, Bluetooth, USB, HomePNA, HomePlug, HomeRF, HiperLAN2, and Wireless LAN are available to network homes and/or small offices. Each one of these technologies has its own pros and cons and is suitable for specific applications.

The emergence of digital video is a recent trend in consumer electronics that is spawning a new generation of entertainment products. And, as video is a very bandwidth intensive application IEEE 1394 has established a strong presence in Digital Cameras, DTV, HDTV, digital hard disk recorders, etc. And, as high end products these are rapidly evolving into the centerpiece components of modern entertainment systems.

Bluetooth on the other hand is a modest performance wireless technology that is seeing widespread adoption in low end consumer applications. One of the more popular of these is digital audio, a subset of digital video (MP3 is the audio format for MPEG). Thus there is a compelling case to integrate these technologies so that PDA's, laptops, MP3 players, and myriad other Bluetooth products can access the audio content of 1394 based systems.

A Xilinx based fast UART to 1394 bridging solution is the ideal mechanism for integrating these technologies. Such a solution can be realized quickly, can leverage a wide variety of low cost Bluetooth components, and can be optimized to integrate neatly into the 1394 environment. The result is a cost effective solution with fast time to market.

With Xilinx programmable logic, on-chip designs will also benefit from Bluetooth's unprecedented flexibility and reprogrammability. FPGA and CPLD based systems can better deal with the inevitable unknowns designers will encounter from the benchtop in the lab to deployed products in the field. Further, programmable solutions can be quickly and efficiently leveraged into derivative configurations to rapidly address new market opportunities. When fully exploited, these benefits will reap large rewards and greatly improve the return on your engineering investment.

Bluetooth to HCI Bridging

Bluetooth is one of the fastest growing wireless technologies in the marketplace. By offering an alternative to cabled systems that currently connect most electronic devices, it promises a new level of customer convenience and demand. More than 2,000 companies currently participate in the Bluetooth Special Interest Group, and Bluetooth product shipments are expected to approach one billion units per year by the year 2005.

The Bluetooth specification defines a device communications interface that requires minimal user intervention. It supports wireless peer-to-peer connections as well as wireless access to LANs, PSTNs, mobile phone networks, and the Internet. At present, however, most standard Bluetooth silicon solutions simply provide a UART and USB host controller interface (HCI).

Figure 1 illustrates a standard interconnection between a host system and a Bluetooth module.

The traditional Bluetooth module consists of the following three components:

- RF - Radio frequency component
- BB - Baseband processor

- µC - Microcontroller

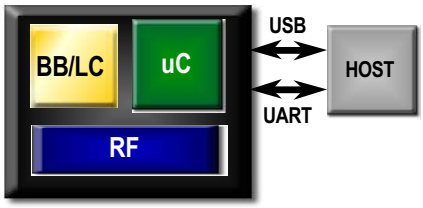

Figure 1: **Zero-Glue Bluetooth Interface**

In many cases, the traditional approach may not be practicable, for instance:

- The host system may not have an available USB or UART port
- The host system may not have a UART port with adequate performance
- The host system is MIPS limited and implementing these port interfaces in the typical manner imposes excessive processing overhead
- The target application is intended to integrate additional capabilities not supported in the standard components

In any of these situations, a Xilinx Spartan FPGA can be used to implement an alternative interface. In this paper, we focus on how you can efficiently bridge from the UART to PCI to Bluetooth enable a legacy system with a PCI bus (which often exhibit the constraints listed above).

While Spartan FPGAs can implement either a USB or UART port, the simpler transport protocol of the UART results in a more cost-effective solution with better system level performance. Since a UART operates in a single mode with eight bits of data, no parity, and one stop bit, the FPGA implementation can be very simple and operate at very high speed. In addition, unlike USB or RS232, the UART transport layer does not need external level shifters or transceivers when implemented in an FPGA. Finally, reduced implementation complexity and better system level performance can be achieved by tailoring the solution specifically to the characteristics of the target components.

UART

A Universal Asynchronous Receiver and Transmitter (UART) is used for communication with serial input/output devices. Serial communication is needed either for devices such as modems and telephone lines, which are inherently serial, or when the cabling cost has to be reduced at the expense of operating speed (e.g., a twisted pair in laboratory instrumentation).

Typically, the UART is connected between a central processor and a serial device. To the processor, the UART appears as an 8-bit parallel port, which can be written to or read from. To the serial device, the UART presents two data wires, one for input and one for output, which serially communicate 8-bit data. The rate of data communication depends on the peripheral device. Some devices operate at a single clock speed (e.g., old modem at 9600 baud) and generate an internal clock. Other devices operate at multiple clock rates and get their clock input from the UART.

A UART is used for building serial communication devices such as modems and serial ports. It has a transmitter section and a receiver section. The transmitter converts the (8-bit) bytes into a serial stream of data bits as they are prepared for transmission. The receiver takes the incoming stream of bits and groups them into 8-bit chunks so they can be reconstructed as bytes.

The UART also monitors input control lines and has the ability to change the state of output lines depending on the program code running at the time. UARTs can be wired as either Data Terminal Equipment (DTE) or Data Communication Equipment (DCE). They are controlled by a clock usually running at different speeds. A buffer is also used to temporarily hold incoming data. This buffer varies by design and is usually very small, ranging anywhere from 1 byte to 128 bytes.

For example, a 16-byte FIFO may not sound like much, but it allows up to 16 characters to be received before the host has to service the interrupt. This increases the maximum bps rate the host can process reliably from 9,600 to 153,000 bps, if it has a 1 ms interrupt dead time. A 32-byte FIFO increases the maximum rate to over 300,000 bps.

The two different types of RAM that can be implemented on a Xilinx device are Distributed SelectRAM and Block SelectRAM. Distributed SelectRAM is implemented in LUTs and is suitable for shallow memory structures and small FIFOs. Block SelectRAM are dedicated 4K-bit RAM blocks, in which depths and widths are parameterizable. These are suitable for applications that require larger blocks of memory. The memory resources available on even the smallest Spartan-II are more than sufficient to build highly efficient fast UARTs.

As illustrated in Figure 2, a typical UART architecture consists of the following blocks:

- Transmit
- Receive
- Control Logic
- Baud Rate Generator
- Internal Control
- Modem Control

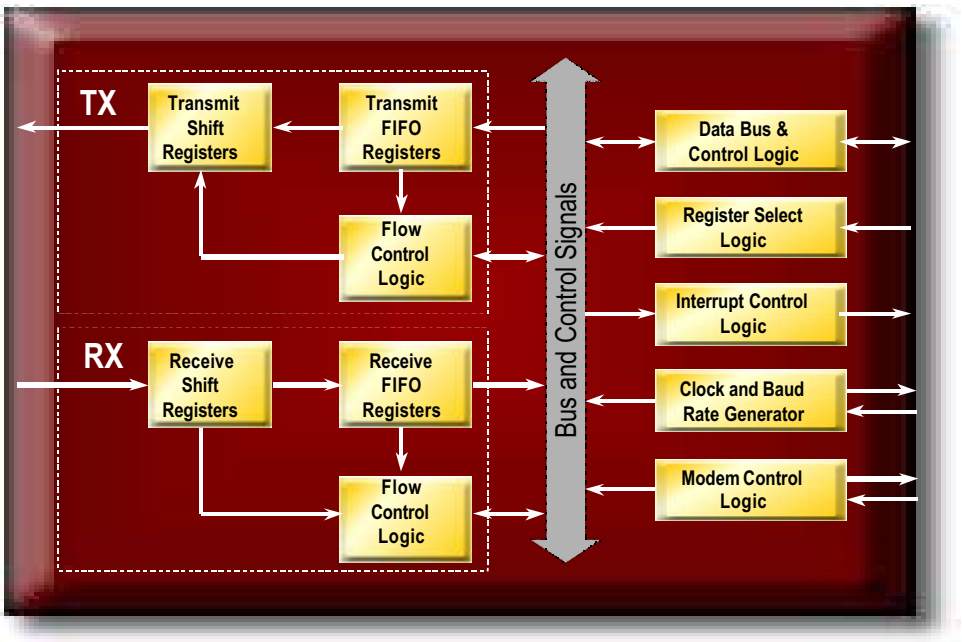

Figure 2: **Standard UART Functional Blocks**

Transmit Block

The Transmit block includes the Transmitter Holding Register (THR), Transmitter Shift Register (TSR), Transmitter FIFO Register (TFR),and Transmitter Flow Control Logic. Whenever the CPU writes parallel data into the Transmitter Holding Register or the Transmitter FIFO, it is immediately transferred to the Transmitter Shift Register. Data is shifted out serially from the TSR.

Receive Block

The Receive block includes the Receiver Buffer Register (RBR), Receiver Shift Register (RSR), Receiver FIFO, and Receiver Flow Control Logic. Whenever the Receiver Shift Register has received a complete character, it is immediately transferred to the Receiver Buffer Register (in character mode) or to the receiver FIFO (in FIFO mode). The CPU reads parallel data either from the RBR or from the RFR depending upon the mode of operation.

Control Logic Block

The Control Logic block consists of two sub-blocks — **Select** and **Control** logic blocks. They decode the address lines and chips select lines and generate enables to various UART internal registers. The Register Control block includes two control registers — the line control register and the FIFO control register. The bits set in these two control registers control the operation of the UART.

Internal Registers

A typical UART device provides internal registers for monitoring and control. These registers function as data holding registers (THR/RHR), interrupt status and control registers (IER/ISR), FIFO control registers (FCR), line status and control registers (LSR/LCR), modem status and control registers (MSR/MCR), programmable data rate (clock) control registers (DLL/DLM), and as a user assessable scratchpad register (SPR).

- **FIFO Control Register (FCR)** is used to enable the FIFOs, clear the FIFOs, set the transmit/receive FIFO trigger levels, and select the DMA mode.
- **Interrupt Status Register (ISR) —** A UART device should provide six levels prioritized interrupts to minimize external software interaction. The Interrupt Status Register (ISR) provides the user with six interrupt status bits. Performing a read cycle on the ISR will provide the user with the highest pending interrupt level to be serviced. No other interrupts are acknowledged until the pending interrupt is serviced. Whenever the interrupt status register is read, the interrupt status is cleared.
- **Line Control Register (LCR)** is used to specify the asynchronous data communication format. The word length, the number of stop bits, and the parity are selected by writing the appropriate bits in this register.
- **Line Status Register (LSR)** provides the status of data transfers between the UART device and the CPU
- **Modem Control Register (MCR)** controls the interface with the modem or a peripheral device.
- **Modem Status Register (MSR)** provides the current state of the control interface signals from the modem or other peripheral device that the UART device is connected to.

Modem Control Block

The Modem Control block includes the modem control register and the modem status register. It monitors changes in the modem input signals and sets corresponding bits in the modem status register. The CPU sets bits in the modem control register whenever it wishes to transmit data to another terminal or handshake to the modem input signals.

Interrupt Control Block

The Interrupt Control block includes the interrupt enable register (which masks the interrupts from the receiver ready, transmitter empty, line status and modem status registers) and the interrupt identification register. The function of this block is to generate an interrupt whenever:

- Data is received
- Error in the received character
- Character timeout occurs
- Transmitter holding register becomes empty
- Change in the modem input signals

Programmable Baud Rate Generator

A single baud rate generator is provided for the transmitter and the receiver, allowing independent TX/RX channel control. The programmable Baud Rate Generator is capable of accepting an input clock up to 24 MHz, as required for supporting a 1.5 Mbps data rate.

The generator divides the input 16X clock by any divisor from 1 to 16. A UART device divides the basic crystal or external clock by 16. Further division of this 16X clock provides two table rates to support low and high data rate applications using the same system design. Customized (using an FPGA) baud rates can be achieved by selecting the proper divisor values for the MSB and LSB sections of baud rate generator. Programming the Baud Rate Generator Registers DLM (MSB) and DLL (LSB) gives the user the capability for selecting the desired final baud rate.

IEEE 1394 (FireWire)

IEEE 1394 (FireWire or iLink) is a hardware and software standard for transporting data at 100, 200, 400, or 800 megabits per second (Mbps). The new revision of 1394 (1394b) will be capable of transferring data at 3.2 Gbps in the 100-meter range.

1394 is a digital interface. Since the video data is digital, each 1394 device can process the video without the expense and image quality lost from digitization. There is no need for a video capture card or any analog-to-digital video conversion within the computer.

1394 is physically small. The thin serial cable can replace larger and more expensive interfaces. The 6-pin connectors have two data wires and two power wires for devices that derive their power from the 1394 bus. Following are the main advantages of IEEE 1394:

- **It is easy to use.** There is no need for terminators, device IDs, or elaborate setup. It is unlike other technologies such as Ethernet that require repeaters, terminators, and device IDs.
- **It is hot pluggable.** Users can add or remove devices with the bus active and there is no need to reset the bus.
- **It is inexpensive.** It is priced for consumer products.1394 has a scaleable architecture . One may mix 100, 200, and 400 Mbps devices on a bus. With 1394b, it can go up to 3.2 Gbps. It has a flexible topology supporting daisy chaining and branching for true peer-to-peer communication.

As Figure 3 illustrates, the 1394 protocol stack has the following components.

Physical Layer — Provides the electrical and mechanical connection between the 1394 device and the 1394 cable. Besides the actual data transmission and reception tasks, the Physical Layer provides arbitration to insure all devices have fair access to the bus.

1394 Physical Layer is physically point to point and logically a bus (each node is a repeater) The Physical layer transmits the unstructured raw bit stream over a physical medium, and describes the electrical, mechanical, and functional interface to the carrier. It is this layer that provides the linking to the upper sessions via signaling. The Physical layer provides the initialization and arbitration services necessary to assure that only one node at a time is sending data.

The Physical layer of the 1394 protocol includes:

- The electrical signaling
- The mechanical connectors and cabling
- The arbitration mechanisms
- The serial coding and decoding of the data being transferred or received
- Transfer Speed detection

Link Layer — Provides data packet delivery service for the two types of packet delivery; asynchronous and Isochronous. Asynchronous is the conventional transmit-acknowledgment protocol and isochronous is a real-time guaranteed-bandwidth protocol for just-in-time delivery of information.

Transaction Layer — Supports the asynchronous protocol write, read, and lock commands. A write sends data from the originator to the receiver and a read returns the data to the originator.

Serial Bus Management — Provides overall configuration control of the serial bus in the form of optimizing arbitration timing, guarantee of adequate electrical power for all devices on the

bus, assignment of which 1394 device is the cycle master, assignment of Isochronous channel ID, and basic notification of errors. Bus management is built upon IEEE 1212 standard register architecture

As Figure 3 illustrates, a 1394 PHY has the following functional blocks:

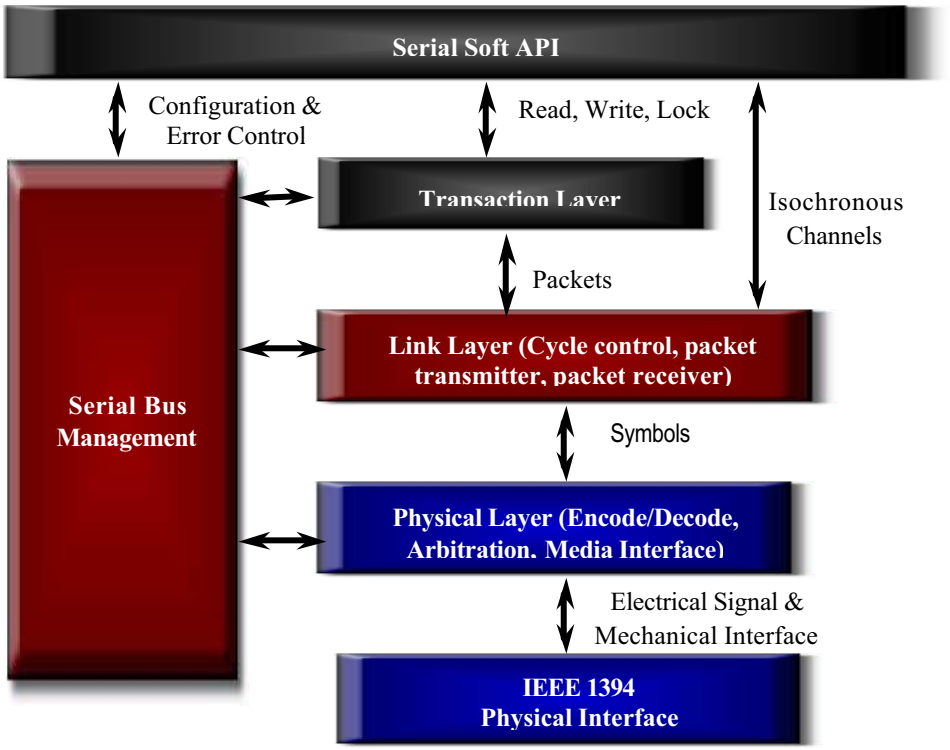

Figure 3: **1394 Protocol Stack**

Link Interface — The PHY-Link interface manages all the PHY-Link interface communications. It handles arbitration requests, packet transmission, packet reception, status transmission,and multi-speed packet concatenation (Figure 4).

Tx Decoder — Receives data to be transmitted over the bus from two or four parallel data paths to the Link Controller. These data paths are latched and synchronized with the clock. The parallel bit paths are combined serially, encoded, and transmitted at either 98.304 Mb/s or 196.608 Mb/s, depending whether the transaction is a 100 Mb/s or 200 Mb/s transfer, respectively. The transmitted data is encoded as data-strobe information with the data information being transmitted on the TPB cable pairs and the strobe information transmitted on the TPA cable pairs.

Tx Decoder and Timer — During packet reception the TPA and TPB transmitters of the receiving cable port are disabled, and the receivers for that port are enabled. The encoded data information is received on the TPA cable pair and the strobe information is received on the TPB cable pair. The combination of the data and strobe signals is decoded to recover the receive clock signal and the serial data stream. The serial data stream is converted to two or four parallel bit streams, resynchronized to the internal clock and sent to the associated link controller.

Arbitration and Control Logic — The cable status, bus initialization, and arbitration states are monitored through the cable interface using differential comparators. The outputs of these comparators are used by internal logic to determine cable and arbitration status.

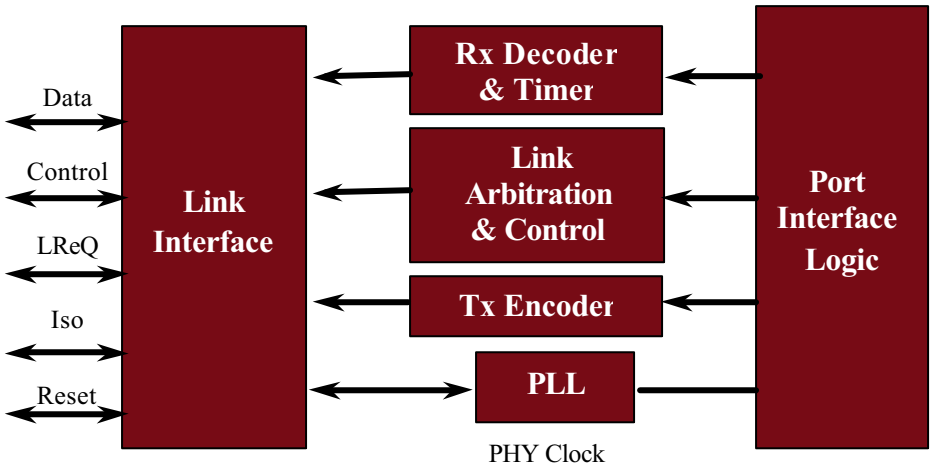

Figure 4: **1394 PHY**

IEEE 1394 Operation

IEEE 1394 supports two types of data transfer: Isochronous and Asynchronous.

Isochronous data transfer puts the emphasis on the guaranteed timing of the data, and less emphasis on delivery. Isochronous transfers are always broadcast in a one-to-one or one-to-many fashion. No error correction or retransmission is available for Isochronous transfers. Up to 80% of the available bus bandwidth can be used for Isochronous transfers. The delegation of bandwidth is tracked by a node on the bus Isochronous channel IDs are transmitted followed by the packet data. The receiver monitors the incoming data's channel ID and accepts only data with the specified ID. Isochronous transfers are the best choice for sending time-critical and error-tolerant video or audio stream data.

Asynchronous data transfer puts the emphasis on guaranteed delivery of data, with less emphasis on guaranteed timing. Asynchronous transfers are targeted to a specific node with an explicit address. They are not guaranteed a specific amount of bandwidth on the bus. They are guaranteed a fair shot at gaining access to the bus when asynchronous transfers are permitted. Asynchronous transfers are acknowledged and responded to. This allows error-checking and retransmission mechanisms to take place.

If the data isn't error-tolerant, such as a disk drive, then asynchronous transfers are preferable.

UART-To-1394 Bridge

Different networking and interconnection technologies must co-exist due to different performance criteria, distribution methods, and cost considerations. A technology bridge successfully connects these disparate technologies together. Reprogrammability features of an FPGA provide excellent flexibility and reliability for bridging and translating these technologies.

The Spartan-II family of FPGAs provide a flexible and programmable architecture of Configurable Logic Blocks (CLBs) surrounded by a perimeter of programmable Input/Output Blocks (IOBs). There are four Delay-Locked Loops (DLLs) for clock management (can be used for customized UART Baud Rate Generator). Spartan-II has two columns of block RAM between the CLBs and the IOB columns. A powerful hierarchy of versatile routing channels interconnects these functional elements.

The Spartan-II family FPGAs are customized by loading configuration data into internal static memory cells. Unlimited reprogramming cycles are possible with this approach. Stored values in these cells determine logic functions and interconnections implemented in the FPGA.

Configuration data can be read from an external serial PROM (master serial mode), or written into the FPGA in slave serial, slave parallel, or boundary scan modes. These unique features enable designers to develop programmable Application Specific Standard Products (ASSP) in a very short time cycle.

The Spartan-II family devices provide system clock rates up to 200 MHz and internal performance as high as 333 MHz. The Spartan-II family FPGAs offer the most cost-effective solution while maintaining leading edge performance. In addition to the conventional benefits of high-volume programmable logic solutions, Spartan-II FPGAs also offer on-chip synchronous single-port and dual-port RAM (block and distributed form), DLL clock drivers, programmable set and reset on all flip-flops, fast carry logic, and many other features.

A Buffer Manager along with an adjustable Transmit/Receive Shift Registers, and Transmit/Receive FIFO Registers of a UART can act as an actual bridge between UART and 1394 packages. This buffer manager should have a RAM bandwidth of 400Mbps. It provides storage for 1394 and UART data, automatically storing them and passing them to the appropriate destinations, without any intervention from the processor.

The Spartan-II FPGAs dedicated on-chip blocks of 4096 bit dual-port synchronous RAM are ideal for use in FIFO applications. This feature, along with other features, is ideal for building a UART-to-1394 bridge. The FIFOs can be reconfigured in depth and width to accommodate much faster speeds in transmission and reception data rates. This gives designer a great amount of flexibility to customize their UART-to-1394 bridge based on different applications and system performance requirements.

Other advantages of using Spartan-II in UART-to-1394 bridge include:

- Faster Transmit and Receive Shift Registers
- Wider Transmit and Receive Shift Registers
- DMA to allow more efficient bus traffic
- Removable internal baud rate generator
- Customizable CPU interface
- Multiple transmit and receive blocks for independent additional fast serial channels

Spartan-II FPGAs have up to 200K gates which make the multiple implementations of UART possible. DMA channels can also be implemented to increase the performance and

functionality of the bridge. Figure 5 illustrates the implementation of 1394 PHY and UART in a Spartan-II device.

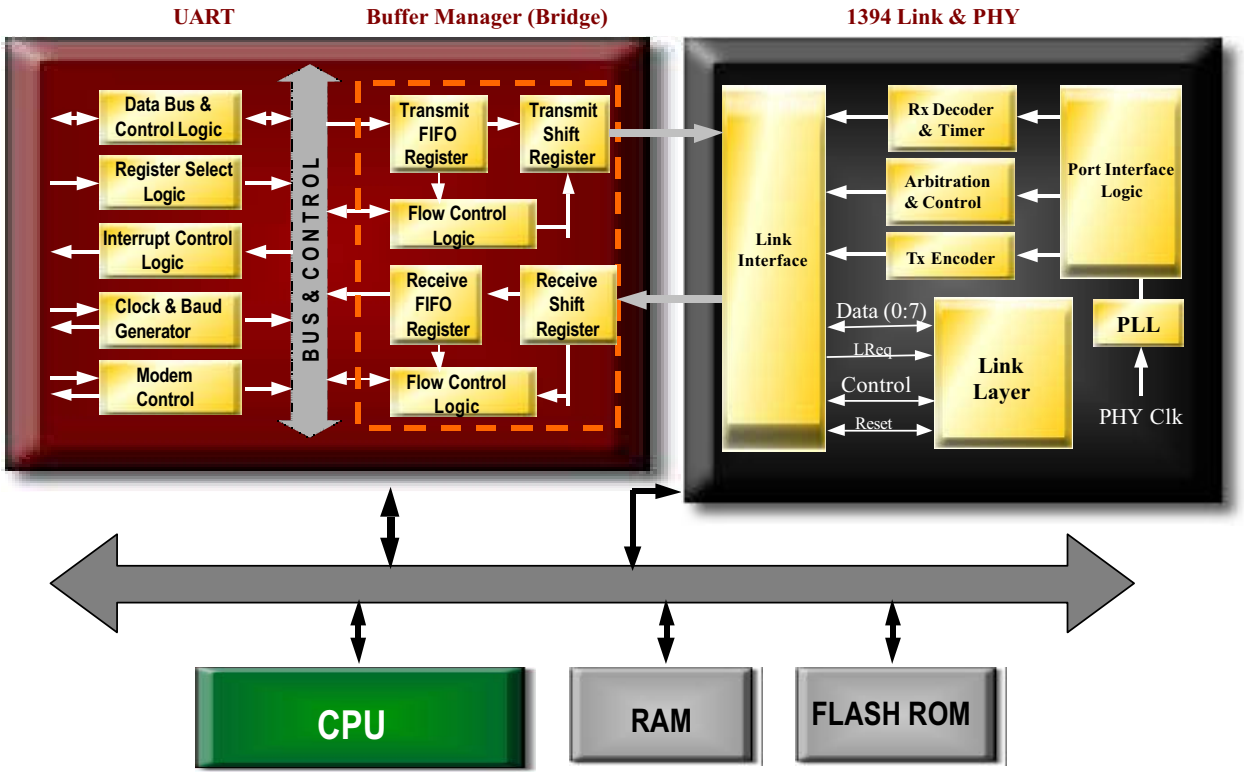

Figure 5: **UART-to-1394 Bridge**

FPGA Advantages

As we have seen, programmable logic provides an excellent platform for integrating Bluetooth technology into embedded systems. Let's highlight the key benefits that they bring:

Time-to-Market

Xilinx programmable logic provides several advantages that reduces time-to-market. First, a broad range of IP support from Xilinx and a ever growing number of third party IP partners provide quick access to key design building blocks. Second, as benchtop programmable solutions, they allow the system designer to achieve a functional hardware platform more rapidly than any alternative. And third, programmable devices are standard parts that are easy to reproduce quickly in limited-to-high volumes to capture a position in strategic accounts before the competition.

Rapid Software Development

Software development is one of the biggest issues in integrating Bluetooth technology. And, obviously, since programmable logic can achieve functional hardware sooner it creates an advantage in this area. However, this advantage can be even greater when you consider the flexibility that programmability brings to the equation. For instance, it is often desirable to use existing or third party drivers and firmware. With a programmable solution, you have full control over the behavior of the interface ensuring a workable approach. This can be particularly valuable when third party code is involved because it may not be well documented or understood and modifications can raise support and maintenance concerns.

Time-in-Market

Product development by its nature is not an exact science. Bugs and incompatibilities are simply a reality that engineering must deal with. Here, especially Xilinx programmable devices can provide a valuable advantage, as our solutions are inherently **reprogrammable**. Thus, patches for known problems can be put into production as soon as they are validated on the existing hardware revision and can also be deployed to installed systems. This allows you to keep your existing design shipping and greatly reduces the risk of obsolete part inventories and expensive field replacement programs.

Rapid Design Derivation

A system design is a corporate asset and in today's world of hyper competition and compressed development cycles, these assets must be flexible. Standards evolve, customers request new features, and experience reveals new business opportunities that can be exploited. Thoughtful designs that incorporate programmable logic are inherently more scalable and are superior platforms for rapid and efficient product derivation. Thus, well exploited programmable logic can make your future product roadmap a strategic competitive advantage.

System Level Cost Reduction

In the past, the use of programmable logic was considered an expensive solution. However, times have changed because Moore's Law has worked to the advantage of programmable solutions. Today, $10 will buy 100,000 system gates in volume, off the shelf, and ready to go. And, as these devices usually replace other functions in your design as well, they can often enable real system level cost reductions. Programmable logic has replaced the small cost reduction ASICs of yesterday and brings many other advantages to your system too!

References

1. Bluetooth HCI Bridging White Paper, January 2001, Kent Dahlgren
2. UART to PCMCIA White Paper, March 2001, Antolin Agatep
3. UART to PCI White Paper, March 2001, Mamoon Hamid
4. 200 MHz UART design by Ken Chapman, http://support.xilinx.com/xapp/xapp223.pdf
5. Specification of the Bluetooth System, Core, version 1.0B December 1999, Bluetooth SIG

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/08/01	1.0	Initial Xilinx release.