A Fully Synthesizable Bluetooth Baseband Module for a System-on-a-Chip

Ik-Jae Chun, Bo-Gwan Kim, and In-Cheol Park

Bluetooth is a specification for short-range wireless communication using the 2.4 GHz ISM band. It emphasizes low complexity, low power, and low cost. This paper describes an area-efficient digital baseband module for wireless technology.

For area-efficiency, we carefully consider hardware and software partitioning. We implement complex control tasks of the Bluetooth baseband layer protocols in software running on an embedded microcontroller. Hardware-efficient functions, such as low-level bitstream link control; host controller interfaces (HCIs), such as universal asynchronous receiver transmitter (UART) and universal serial bus (USB) interfaces; and audio Codec are performed by dedicated hardware blocks. Furthermore, we eliminate FIFOs for data buffering between hardware functional units. The design is done using fully synthesizable Verilog HDL to enhance the portability between process technologies so that our module can be easily integrated as an intellectual property core on system-on-a-chip (SoC) ASICs. A field programmable gate array (FPGA) prototype of this module was tested for functional verification and realtime operation of file and bitstream transfers between PCs. The module was fabricated in a 0.25-μm CMOS technology, the core size of which was only 2.79 mm×2.80 mm.

Keywords: Bluetooth, baseband module, link controller, intellectual property.

I. Introduction

Due to progress in related technologies in the past decade, wireless telecommunication technology has been applied to telephony services, medical instruments, home electronics, and other applications. It has been replacing existing wired applications for the increased convenience of customers and for opening new applications. Wireless communication using the 2.4 GHz ISM band in particular is forecasted to increase explosively [1] because the band is used without license.

Bluetooth, operating in the ISM band, is a specification [2] for short-range wireless communication. It was developed in 1999 to substitute cables connecting portable or desktop devices and to build low-cost wireless networks for mobile and portable devices. It emphasizes low complexity, low power consumption, and low cost [3]-[5]. It is crucial to implement digital baseband processing in a cheap, small module and desirable to integrate the whole system on a chip to achieve the power and cost targets [5]. Baseband modules as reusable intellectual property (IP) cores [6] enable those higher levels of integration through system-on-a-chip (SoC) design and reduce time-to-market.

The Bluetooth baseband module is in general responsible for carrying out link control and link management tasks. The detailed tasks of the module vary significantly depending on applications. For the simplest applications, such as wireless headsets and cellular phone add-on dangles, the entire application as well as a basic part of the baseband layer protocols may be implemented in software on the baseband processor. For more complex applications expecting high-speed full baseband operation and host controller interface (HCI), most of the baseband protocols would be implemented in hardware while more complex upper layer protocols and

Manuscript received Jan. 15, 2003; revised July 31, 2003.

This work was supported in part by the Korea Science and Engineering Foundation through the MICROS center and IDEC at KAIST, Korea.

Ik-Jae Chun (phone: +82 42 821 7707, email: ijchun@ieee.org) and Bo-Gwan Kim (email: bgkim@cnu.ac.kr) are with the Department of Electronics Engineering, Chungnam National University, Daejeon, Korea.

In-Cheol Park (email: icpark@ee.kaist.ac.kr) is with the Department of Electrical Engineering and Computer Science, KAIST, Daejeon, Korea.

application software are processed on a host processor. Allocating more functions to hardware from software can reduce the load/interrupt frequency, but the trade-off could produce a significant increase in gate count and loss of connection flexibility with a resultant poor interoperability performance. The baseband module should therefore be very flexible so as not to waste the processing power and hardware resource.

This paper describes an area-efficient digital baseband module that is suitable for use as an IP core on SoC ASICs. To gain more flexibility, we used a programmable embedded microcontroller optimized to our Bluetooth core. The programmable embedded microcontroller performs as many tasks for channel control and interface as possible. To reduce the size, we merged the data-transfer FIFO buffers distributed in functional units, such as USB, UART, link controller, and audio Codec, into the internal SRAM. The functional units access the SRAM by direct memory access (DMA) through a memory management unit (MMU). The module is made up of a logic part of only 85,000 gates and a 4 kB single-port SRAM. It conforms to the latest version of the Bluetooth (version 1.1) [2]. In addition, it supports firmware programming capability.

The remainder of this paper is organized as follows. Section II gives the overall architecture of our Bluetooth baseband module. In sections III and IV, we present the structure and design of a microcontroller subsystem and a link controller. Section V describes the host controller interfaces and section VI an audio Codec for the Bluetooth module. Section VII summarizes our experiments and results. Finally, we present the conclusions of the paper.

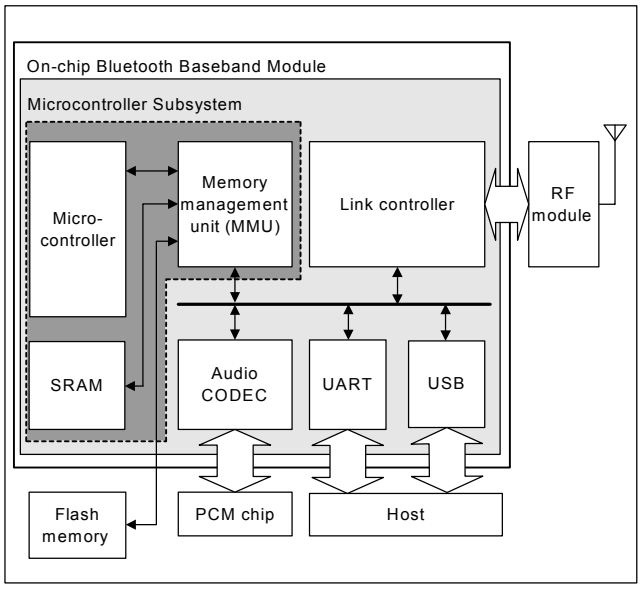

Fig. 1. Overall architecture of the Bluetooth baseband module.

II. Overall Architecture of the Baseband Module

A Bluetooth module generally consists of an RF module to generate wireless channels for data transmission and a baseband module to execute link management tasks, link control tasks, and bitstream processing. In the design of the Bluetooth baseband module, the link manager and link controller are very important function blocks. The link manager performs link management tasks, translating commands and data into operations at the link controller and

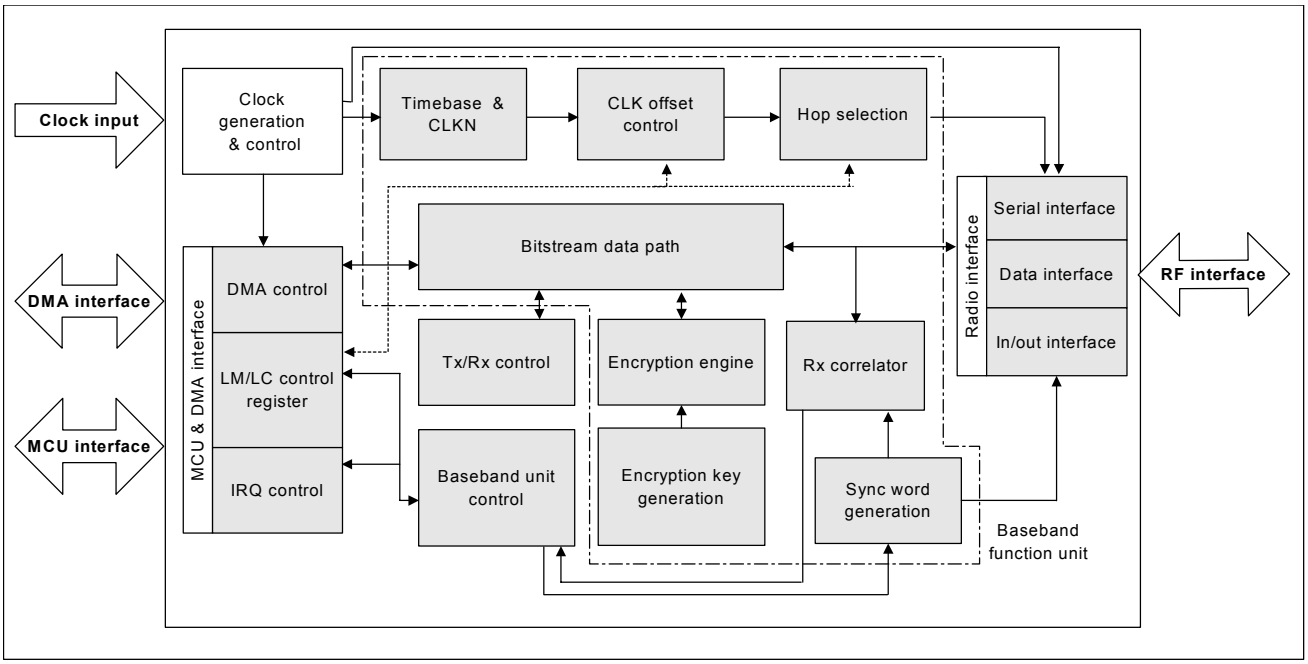

Fig. 2. The block diagram of the link controller.

controlling link establishment, link destruction, and link configuration among Bluetooth devices. This part is generally performed by software running on the embedded microprocessor. The link controller carries out the link control protocol, which is responsible for the mechanics of maintaining a link once established, and performs the bit-intensive and time-critical part [5]. This part is generally implemented in dedicated hardware.

Our baseband module consists of five major functional units: a microcontroller subsystem, a link controller, a universal asynchronous receiver transmitter (UART), a universal serial bus (USB), and an audio Codec. Figure 1 shows the overall architecture of the Bluetooth baseband module. The module is designed so that the microcontroller can easily control each of the peripherals through internal registers and memory mapped I/O.

The microcontroller subsystem consists of a microcontroller, an MMU, and an SRAM (Fig. 1). It manages the other units and executes the Bluetooth link manager, host controller interface, and some part of the link control protocol software. The link controller performs encoding and decoding of Bluetooth bitstream data and low-level timing control. The UART and USB are HCI physical transport layers [2] and operate alternatively. The audio Codec for voice data supports all three of the Bluetooth audio coding methods: A-law, μ-law, and CVSD.

The base interface of the module complies with the one of the ARM7TDMI controller [7], which is commonly used in Bluetooth systems. As shown in Fig. 2, an RF interface connects the baseband module with an RF module. The RF interface consists of a serial interface, a data interface, and an in/out interface. It was designed on the basis of Ericsson's RF module interface [8]. However, for compatibility with other RF chip-solutions [9], [10], the data interface and the in/out interface were designed considering both common interface signals and unique interface signals each commercial RF chip has. The baseband module controls the RF modules through the serial interface based on the IEEE standard 1149.1 boundary scan architecture. The timing of the RF interface signals for bitstream data transmission is controlled by the LM/LC control register of the link controller in Fig. 2. Therefore, the baseband module can be connected directly to various RF modules via the RF interface.

Several baseband hardware modules have been reported either as a part of a system or as an IP [11]-[14]. Their size, however, is large either because they have a distributed buffer (the total dedicated buffer size is about 585 bytes) in each module, namely, the baseband, USB, audio Codec, and UART [14], or because massive hardware is adopted to perform almost all the tasks while the embedded microcontroller is idling for most of the time (the gate count of the baseband IP is 280,000 gates in 0.18-μm technology) [11] or because they use area-occupying dual-port internal SRAMs [12]. However, since our module has the minimum number of dedicated buffers, supports transmission of data via DMA, and consists of several modules designed as IP blocks, our baseband core is a simple, small, and portable Bluetooth core that is suitable for use as an IP core.

The main input clock is 48 MHz. The USB unit runs at the 48 MHz external clock in order to gain synchronization with a 12 MHz Rx bitstream. The other units use a 12 MHz clock that is generated by dividing the 48 MHz by four to save power consumption, but the interface with the radio module operates by the other clocks. The other subclocks, 3.2 kHz, 4 MHz, and 1 MHz clocks, are used in relation to the RF interface between the link controller and an external RF module. The 3.2 kHz is provided from the RF module and used for timing synchronization to the link connection. The 4 MHz frequency is used for the serial interface between the link controller and the RF module. Transmission is synchronized to the 1 MHz provided by the RF module, and reception is synchronized to the 1 MHz clock extracted from the phase-locked loop block of the RF interface block in the link controller.

III. Microcontroller Subsystem

The microcontroller in Fig. 1 controls the other units via a memory-mapped I/O interface and interrupts. The other important task of the microcontroller is to run the Bluetooth link manager, the HCI, and a part of the link control protocol software. The microcontroller performs the complex part of the link control protocol that requires flexibility, such as decision making on received baseband packets and context switching between links, while the link controller performs the bit-intensive and time-critical part.

The MMU in Fig. 1 manages the memory interface and memory-mapped I/O interface of the microcontroller. One of the most important tasks of the MMU is DMA of peripheral units. If the link controller and HCI units have their own data-transfer buffers as in reported designs [14], the buffers will dominate the size of those units when implemented with flip-flops and impose a sizable burden on the microcontroller to move the data. To solve this problem, we merged the distributed large buffers into the internal SRAM, which already existed for the program and data of the microcontroller, which resulted in a great area reduction. Compared to the distributed buffer-based architecture, the logic gate count in functional units and the microcontroller without SRAM was reduced from 132,000 to 85,000, a 35.7% reduction. When 4 kB of on-chip SRAM was counted together, the net area reduction was

27.4%. A 4 kB SRAM was sufficient to run a whole simple application program on the microcontroller, whereas the memory intensive logical link control and adaptation protocol segmentation and reassembly of complex applications may be performed on a host.

In the memory access, considering the load of the microcontroller, the peripheral has a low load. Therefore, although the microcontroller has priority over the other peripherals, a processing delay or transmitting and receiving error on the peripheral's operation is not generated. This low gate count is made possible by the DMA architecture. This architecture simplifies the SoC design and nearly eliminates buffer requirements.

For this Bluetooth baseband module, we used a clone of the Advanced RISC Machines ARM7TDMI core as the microcontroller. We used a single-port on-chip SRAM, which was half as large as a dual-port SRAM of the same capacity. As the ARM7 architecture does not access the RAM while fetching instructions from the flash memory, DMA can be easily implemented with a small single-port SRAM. The instruction fetch and the fixed data load of the ARM processor were obtained from flash memory through a 16-bit interface. In addition, while a peripheral accessed the SRAM (data memory), the processor could fetch instructions from flash memory.

The MMU also provides flash memory programming capability through a UART interface. At power-up, a dedicated pin is used to select the loading of a new program from the UART interface.

IV. Link Controller

The link controller is a part of the baseband module for processing the bit-intensive baseband protocol functions, i.e., Bluetooth bitstream processing and encryption, which are power-efficient if implemented in hardware. In addition, the most time-critical portions of the link control task, such as low-level timing control and frequency hop calculation, are processed by the link controller. The link controller also exactly transmits the processing information about the link connection, such as Tx/Rx timing, interrupt, and event information, to the link manager. The link controller conforms to the latest version of the Bluetooth specification (version 1.1) and supports all of the six asynchronous connectionless (ACL), four synchronous connection-oriented (SCO), and four common packet types. Figure 2 shows a block diagram of the Bluetooth link controller.

As the figure depicts, the link controller has a baseband function unit. The baseband function unit is responsible for the most important tasks of the link controller. The tasks are to perform bit-intensive baseband protocol and the time-critical portion. A Tx/Rx control block and a baseband unit control block control the baseband function unit by generating Tx/Rx timing and packet control signals. The baseband function unit implemented by hardware processes the most time-critical portion of the link control tasks. For this, the baseband function unit has been designed to perform as many hardware-efficient tasks as possible: low-level timing control, frequency hop calculation, encryption, decryption, access code generation and detection, correlation, Bluetooth clock control, etc.

The microcontroller can manage all of the link controller functions, for instance bitstream processing, interrupt, and encryption, by setting the internal registers of the link controller. When a critical event occurs in relation to transmission or reception of packets, the link controller calls the microcontroller interrupt, and then the event information is transferred to the microcontroller via an interrupt register in the LM/LC control register of the link controller. The RF interface consists of a serial interface, a data interface, and an in/out interface. The data and in/out interfaces are responsible for connection for packet transmission from the link controller to an RF module. The serial interface controls the RF module. The RF interface also has a digital phase-locked loop logic for Rx synchronization and is responsible for connection with an RF module. For the detection of a designated packet of data, the link controller has to periodically perform synchronization with a syncword. For this, a correlator using a 64-bit syncword is designed in the baseband function unit.

Before transmission through the RF channel, a bitstream data path block is essential for protecting the data against an imperfect channel. The encryption/decryption block deals with the security of information. Encryption is used as a safeguard against eavesdropping. The bitstream data path block consists of several channel coding blocks, such as HEC, CRC, Whiten, and FEC. The bitstream data path block is also designed so that data can continuously stream through the channel coding blocks without any bitstream buffers between them, which results in an area reduction. To achieve the continuous stream, in the bitstream data path block shown in Fig. 3, we designed the sequencers that control the timing of each function block, such as HEC, CRC, Whiten, Encryption, and FEC. The sequencers can be programmed according to the different packet types and carry out the required processing functions without further microcontroller intervention. In addition, we heavily apply a clock-gating scheme in order to achieve low power consumption. Figure 3 shows the data transmission and reception flow with the sequencer logic in the bitstream data path block.

Rx processing is different from Tx processing. Because we do not know the packet type and length in advance, we must recognize such information during reception. An Rx block

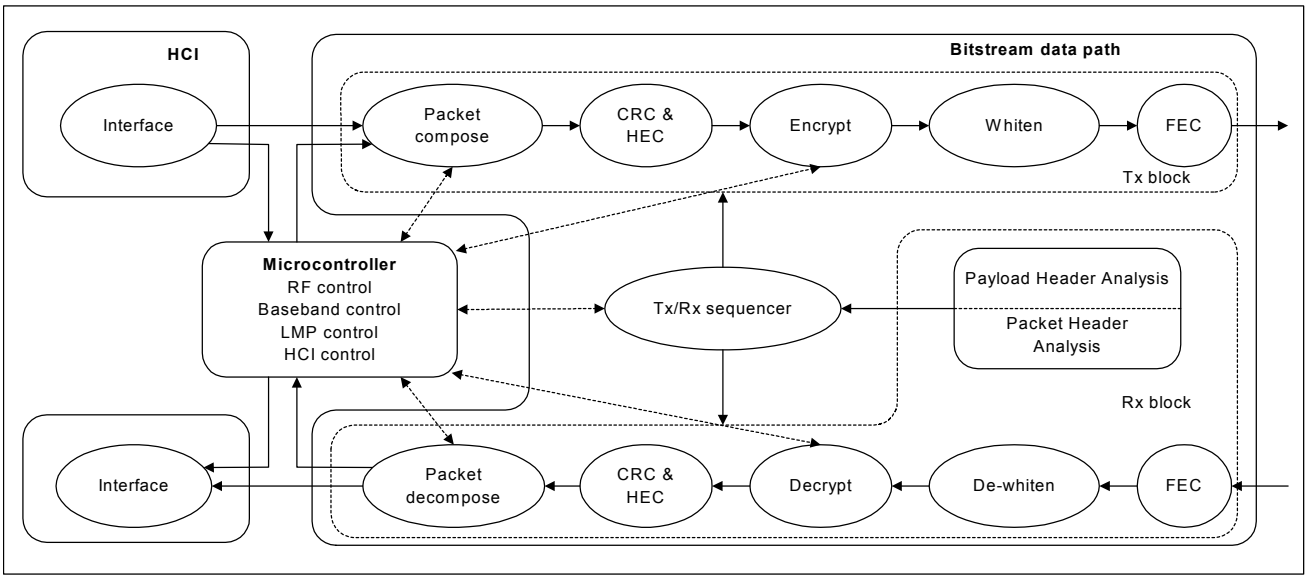

Fig. 3. Bitstream flow in the bitstream data path block.

therefore requires a block that analyzes the form of received data. To perform this task, we designed the packet header analysis block and the payload header analysis block in the bitstream data path block in Fig. 3. The packet header analysis block first analyzes the control information associated with the packet, such as the address of the Bluetooth slave device for which the packet is intended and information on the packet type, and then the payload header analysis block analyzes the logical link control information, such as information on the length of the message in the packet.

The clock control is one of the most important parts in the design of a Bluetooth unit. When two Bluetooth units are to establish a communication channel, the clock (slave clock) and the phase of the slave device must be synchronized to the clock (master clock) of the master device. For this, the baseband function unit uses a 3.2 kHz clock and counts it with a 28 bit counter. The CLK offset control logic in Fig. 2 controls three Bluetooth-specified clocks: CLKN, CLKE, and CLK [2], [3], [5]. CLKN, the native clock, is used as the basis of the other clocks. CLKE and CLK, which represent the estimated clock and master clock, respectively, are obtained by adding an offset to CLKN. The Bluetooth clocks feed the hop selection logic that generates a hopping sequence for the 79-hop system.

V. Host Controller Interfaces

For data transmission and reception between the Bluetooth baseband module and a host, such as a PC and mobile or portable devices, two serial interfaces, UART and USB, are provided. The serial interfaces send and receive bit sequences on the status of these bits to and from another unit that processes the bit sequences [15]. The UART and USB units constitute the physical layer of the Bluetooth HCI. Each unit has a special function for Bluetooth data transfer.

1. UART

The Bluetooth HCI UART transport layer is the most general serial interface between the host and the Bluetooth device. The UART unit is designed on the basis of industry-standard 16C450. It supports baud rates from 300 bps to 1.5 Mbps by a numerical controlled oscillator, and the default bit rate is 57.6 kbps. It also provides firmware-programming capability to meet a modified higher protocol.

The UART unit consists of a Tx unit, an Rx unit, an interrupt block, a flow control block, and an interface block (Fig. 4). The Tx unit converts the parallel data into a serial form to transmit them to the host. The data from the DMA interface are stored in buffer registers, converted into a serial form at the shift register, and transmitted. The Rx unit processes the serial data received from RXD input. Unlike an Rx unit in a general UART, this unit includes a data check block that detects the start bit of data and a packet decoder that finds the packet type and length of the received HCI packets to help HCI processing of the microcontroller.

2. USB

The USB unit complies with USB Specification 1.1 [16] and the HCI USB transport layer specification of Bluetooth v1.1 [2] and supports a full-speed 12 Mbps interface.

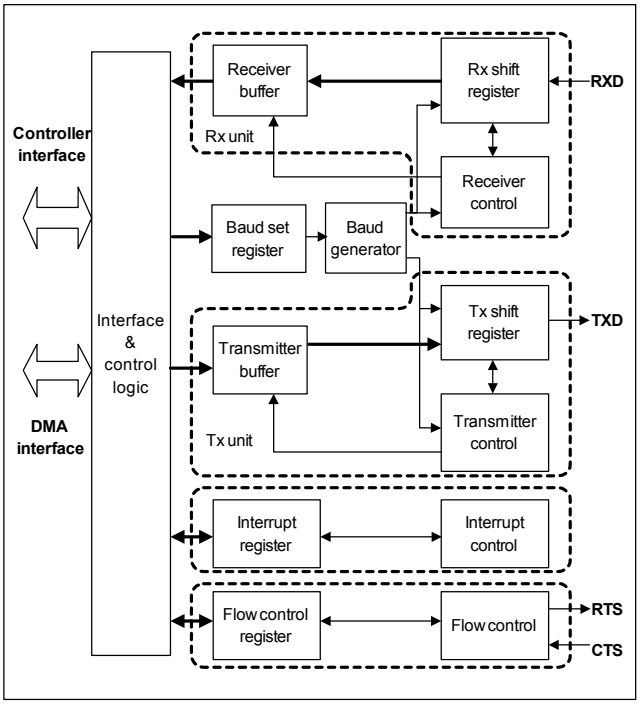

Fig. 4. The block diagram of the UART unit.

The USB controller unit consists of a transceiver interface, serial interface engine, protocol layer handler, registers/endpoint manager, and parallel interface (Fig. 5). The transceiver interface block generates output enable signals for the transceiver to drive the signal line when sending data, and it contains an Rx clock recovery circuit.

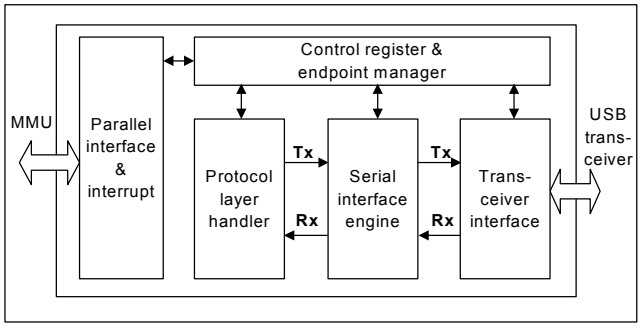

Fig. 5. The block diagram of the USB unit.

The serial interface engine encodes, decodes, and samples signals at the recovered clock. The protocol layer handler works as a transaction sequencer, which performs the control to send or receive expected packets. If it receives an unexpected packet, it will ignore the packet. When it receives an expected error-free packet, it stores the packet in the corresponding endpoint FIFO via DMA and writes related information to registers. The parallel interface and interrupt are provided for data exchange with memory. The different types of HCI packets are mapped onto different USB endpoints according to the Bluetooth specification [2].

VI. Audio Codec

A major application for the Bluetooth is as a carrier of audio information. The audio data are carried via SCO channels and through the use of several coding schemes. Our

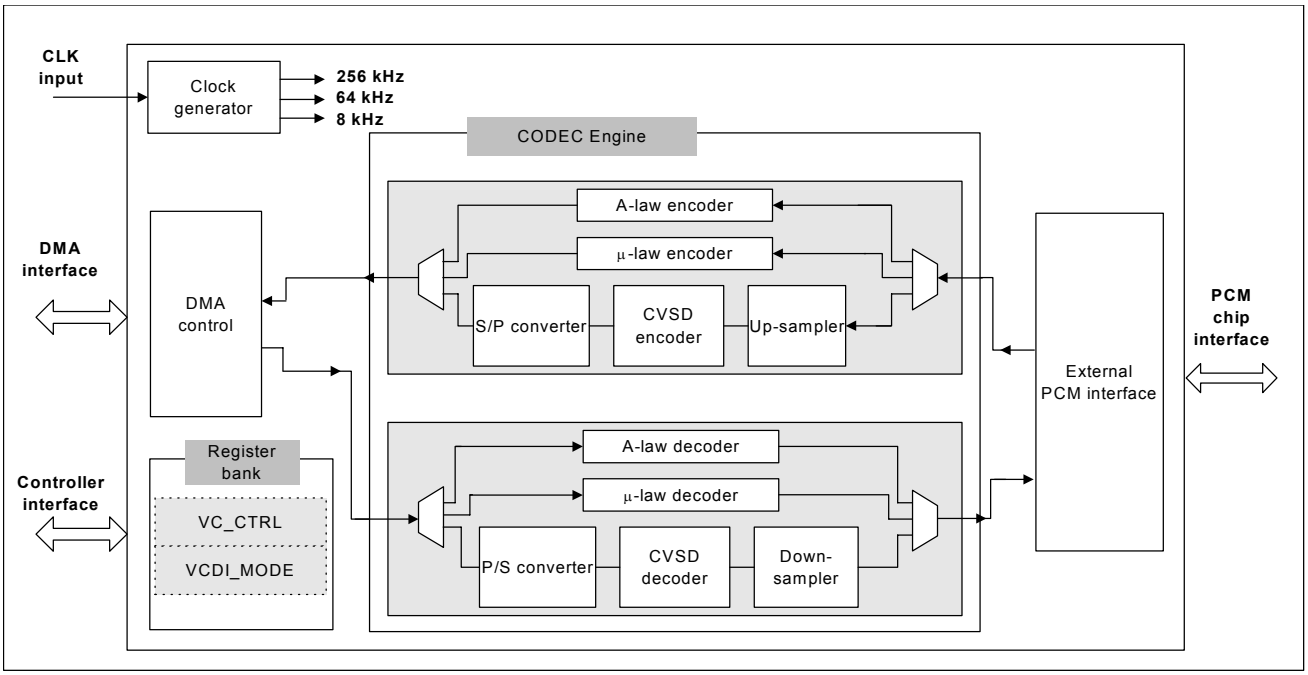

Fig. 6. The block diagram of the audio CODEC.

module supports direct PCM connection by the baseband audio Codec.

The Bluetooth specifies three audio coding techniques: log PCM coding using either A-law or μ-law [17] and continuous variable slope delta modulation (CVSD) [2], [18]. Since the table lookup of the log PCM and the low-pass filtering necessary in CVSD to avoid aliasing are appropriate for hardware implementation, we implemented all three coding methods in a hardware audio Codec and designed the external interface based on a general commercial PCM chip. Figure 6 shows the hardware implementation. In simple audio applications, the audio Codec can access voice data without using an HCI. Its interface is designed to pass 8- to 16-bit decoded linear PCM signals. For PC applications, a coded audio bitstream can also be transmitted via SCO data packets through USB or UART.

In the Bluetooth specification, the sampling frequencies for log PCM and CVSD are not the same: 8 kHz for log PCM and 64 kHz for CVSD. We configured the PCM interface of the audio subsystem to operate at 8 kHz and implemented interpolation with linear interpolation and decimation with low-pass filtering for the CVSD block in the Codec engine (Fig. 6).

The audio Codec is so small that it requires only 5,000 gates, where the 5-tap elliptical IIR low-pass filter occupies half of the entire unit.

VII. Performance Evaluation and Implementation

We moved the dedicated buffer blocks in the Bluetooth baseband module into the SRAM and adopted a DMA architecture. However, these operations may induce performance degradation in running an application and influence data transfer in the baseband module. Therefore, the performance degradation of the baseband module due to removing dedicated buffer blocks should be carefully analyzed to meet the performance specification. According to [5], the CPU performance required for the baseband layer (link controller, link manager, and HCI) is about 10 to 15 million instructions per second (MIPS). However, this is not the optimum value, and there is a trade-off between the gate count and CPU performance. Adding more functions to hardware can reduce the CPU performance. Figures 2 and 3 show our solution for the trade-off. The operating clock is 12 MHz except for the USB unit, the unit of data transfer among peripherals of the baseband module is 8 bits, and the MMU takes 2 cycles for DMA of 8 bits. When the microcontroller does not request RAM, DMA is started, and if the microcontroller request is asserted during the DMA operation, DMA is interrupted. In addition, in the case of a continuous stack operation (stack read/write) in the microcontroller, the MMU stops the stack operation and starts the DMA operation in order to prevent starvation. The following is the equation for CPU performance degradation (PD_{CPU}) by the DMA in a memory operation:

$$PD_{\mathrm{CPU}} = (TI_{\mathrm{USB}} + TI_{\mathrm{LC}} + TI_{\mathrm{Audio}}) \times (1 \times P_{\mathrm{RAM}} + 2 \times P_{\mathrm{Stack}}).$$

Here, the terms TI_{USB}, TI_{LC}, and TI_{Audio} are the numbers of the direct memory access per clock cycle of each peripheral. The constant 1 means 1 cycle stall when the microcontroller accesses the RAM. The constant 2 means 2 cycles stall when the microcontroller executes a stack operation. The term P_{RAM} is the frequency of memory access for the total cycle, and P_{Stack} is the frequency of the stack operation for the total cycle.

• Overall performance

This is the common case for data transmission and reception. In this case, the USB and the link controller have a 1 Mbps transfer rate (1 DMA per 96 cycles), and the audio Codec has a 128 kbps transfer rate (1 DMA per 750 cycles). The memory access is below 10% of the total cycle, and the stack operation is below 1% of the total cycle. Therefore, the CPU performance degradation by the DMA in the memory operation is 0.27%.

• Worst-case performance

In this case, the USB and the link controller have a 12 Mbps (1 DMA per 8 cycles) and 1 Mbps (1 DMA per 96 cycles) transfer rate, respectively, and the audio Codec has a 128 kbps transfer rate (1 DMA per 750 cycles). In a program with many memory operations, the memory access is 50% of the total cycle in the case of continuous store, and the stack operation is 10% of the total cycle. In the worst case, the CPU performance degradation by the DMA operation of the peripheral units in memory operation is 9.57%.

The microcontroller used in this paper takes about 10 MIPS at 12 MHz. The performance degradation of the microcontroller by DMA is a maximum of 9.57% in the memory operation. In the common case, using UART for the HCI, the performance degradation by DMA is very small, and the processing performance is sufficient for a file transfer application. The result is demonstrated by a field programmable gate array (FPGA) prototype. However, in the USB operation with a maximum transfer rate (12 Mbps), the memory access through DMA takes a large part of the memory operations. The 9.57% performance degradation is considerable. A solution for this worst-case is to extend the data width of the USB interface. A 16-bit data width of the

USB interface results in about 50% reduction of the performance degradation in the worst-case memory operation. Thus, though the dedicated buffers in the peripherals are removed, data transmission and reception and packet control have been performed without the reduction of the data transfer rate in the file transfer application.

We constructed an FPGA prototype to validate the hardware and software in a realtime environment. We mapped our design into a Xilinx one-million gate Virtex chip and implemented a test board. The test PCB board contained the FPGA, flash memory, external RAM, Ericsson RF front-end module, PBA313 01/2 [8], and antenna. Figure 7 shows the FPGA test board for realtime operation testing. We confirmed and verified the point-to-point connection [19] capability of the baseband module with two test boards. Each board was connected to a PC through a UART interface. Two boards successfully established a connection and transferred bitstreams and files. The maximum data transfer rate measured in the test was 723 kbps in the DH5 packet using a 5-slot size.

In parallel with the FPGA prototype, an ASIC chip of the baseband module was fabricated in a 0.25-μm 5metal CMOS technology. The ASIC chip had no on-chip SRAM. All the blocks of the chip were soft cores described in RTL with Verilog HDL and fully synthesizable. We used a semi-custom method for implementation. The chip size had a 2.79 mm×2.80 mm area and the operation clock was 48 MHz. The module was made up of a logic part of only 85,000 gates. The chip was fully tested using an IMS ATS2 test station to verify its functionality and timing. Table 1 shows the characteristics of the prototype chip.

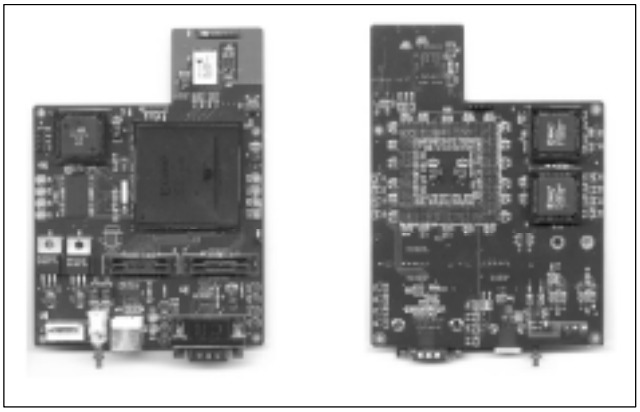

Fig. 7. The FPGA test board (front and rear views).

Table 1. Major chip characteristics.

Technology	0.25 μm CMOS 5LM
Core size	2.79 mm ×2.80 mm
Gate count	85,000
Supply voltage	2.5 V
Operating clock	48 MHz
CPU performance	10 MIPS
UART	Max. 1.5 Mbps
USB (USB spec. ver 1.1)	12 Mbps

VIII. Conclusion

We have presented a small, flexible baseband module for Bluetooth wireless communication. The complex control tasks of the Bluetooth baseband layer protocols were implemented in software running on the embedded microcontroller. The hardware-efficient functions, such as low-level bitstream link control functions; HCIs, such as UART and USB interfaces; and audio Codec were performed by dedicated hardware blocks. Furthermore, we eliminated FIFOs for data buffering between hardware functional blocks.

We implemented and tested the module by both ASIC and FPGA. We tested realtime operations of the FPGA prototype for file and bitstream transfers and point-to-point connection between PCs. The ASIC chip was fabricated in a 0.25 μm CMOS technology. The chip had only a 2.79 mm×2.80 mm core size, and we tested its functionality and timing specification using an IC test station. Furthermore, our baseband module was designed in synthesizable Verilog HDL. Thus, it can be easily integrated as an IP core on SoC ASICs and adjusted to various Bluetooth applications with small size, high speed, and low cost.

A systematic hardware/software trade-off could be further investigated for the optimization of power consumption in addition to area reduction and also for various application environments. Interoperability with other wireless specifications and performance enhancement in the design or in the Bluetooth specification could also be studied.

References

[1] Http://www.bluetooth.com.

[2] Bluetooth Special Interest Group, *Specification of the Bluetooth System Volume 1, Core, Ver. 1.1*, Feb. 2001.

[3] R. Shorey and B.A. Miller, "The Bluetooth Technology: Merits and Limitations," *IEEE Int'l Conf. on Personal Wireless Communications*, 2000, pp. 80-84.

[4] J.C. Haartsen and S. Mattisson, "Bluetooth–A New Low-Power Radio Interface Providing Short-Range Connectivity," *Proc. IEEE*, vol. 88, no. 10, Oct. 2000, pp. 1651-1661.

[5] Jennifer Bray and Charles F. Sturman, *BLUETOOTH Connect*

Without Cables, Prentice Hall PTR, 2001.
[6] T. Grahm and B. Clark, "SoC Integration of Reusable Baseband Bluetooth IP," *Proc. of Design Automation Conf.*, June 2001, pp. 256-261.
[7] Advanced RISC Machines, *ARM7TDMI Data Sheet*, Aug. 1995.
[8] Ericsson Microelectronics, *PBA313 01/2 Bluetooth Radio*, http://www.ericsson.com, Nov. 1999.
[9] Taiyoyuden, *EYSR2SXXX(Radio+Modem) Specification Report*, http://www.stonestreetone.com, Sept. 2001.
[10] Silicon Wave, *SiW1701 Radio Modem Data Sheet*, Sept. 2002.
[11] Ericsson Microelectronics, *EBCP CherryRed Bluetooth Baseband IP*, http://www.ericsson.com, May 2001.
[12] F.O. Eynde et al., "A Fully-Integrated Single-Chip SOC for Bluetooth," *IEEE Int'l Solid-State Circuits Conf.*, Feb. 2001, pp. 196-197, 446.
[13] Cambridge Silicon Radio, *BlueCore01b Product Data Sheet*, July 2001.
[14] Atmel, *Single Chip Bluetooth Controller AT76C551 Data Sheet*, http://www.atmel.com, Aug. 2001.
[15] Yil Suk Yang et al., "A Serial Input/Output Circuit with 8 bit and 16 bit Selection Modes," *ETRI J.*, vol. 24, no. 6, Dec. 2002, pp. 462-464.
[16] Compaq, Intel, Microsoft, NEC, *Universal Serial Bus Specification 1.1*, Sept. 28, 1998.
[17] International Telecommunications Union, *ITU-T Recommendations G.711 Pulse Code Modulation (PCM) of Voice Frequencies*, Nov. 1988.
[18] CML Semiconductor Products, *Continuously Variable Slope Delta Modulation (CVSD) – A Tutorial*, AN/G-Purp/CVSD_1, Nov. 1997.
[19] W. Simpson, "The Point-to-Point Protocol (PPP)," *RFC* 1661, Network Working Group, July 1994.
[20] Ick-Sung Choi et al., "A Kernel-Based Partitioning Algorithm for Low-Power, Low-Area Overhead Circuit Design Using Don't-Care Sets," *ETRI J.*, vol. 24, no. 6, Dec. 2002, pp. 473-476.

Ik-Jae Chun received the BS and MS degrees in electronics engineering from Chungnam National University (CNU) in Daejeon, Korea, in 1998 and 2000. He is currently a PhD student in the Department of Electronics Engineering at CNU. His current research interests include digital system architecture, communication system design, VLSI, and CAD. He is a student member of the IEEE.

Bo-Gwan Kim received the BS degree in electronic engineering from Seoul National University, the MS degree in electrical and electronic engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1976 and 1978. He received the PhD degree in electrical and computer engineering from the University of Wisconsin-Madison in 1989. From March 1978 to February 1980, he worked at Korea Institute of Science and Technology (KIST) as a Researcher, and from March 1980 to February 1991, he was an Assistant Professor of Kum-Oh Institute of Technology, Korea. He joined Chungnam National University in March 1991 as a Professor in the Department of Electronics Engineering. His current research interests include VLSI design for communication systems and CAD for VLSI, such as lower-power logic synthesis, hardware-software co-design, and technology migration.

In-Cheol Park received the BS degree in electronics engineering from Seoul National University in 1986 and the MS and PhD degrees in electrical engineering from KAIST in 1988 and 1992. Since June 1996, he has been with the Department of Electrical Engineering and Computer Science at KAIST first as an Assistant Professor and now as an Associate Professor. Prior to joining KAIST, he was with IBM T. J. Watson Research Center in Yorktown Heights, New York, from May 1995 to May 1996, where he researched on high-speed circuit design. He received the best paper award at the ICCD in 1999 and the best design award at the ASP-DAC in 1997. His current research interests include CAD algorithms for high-level synthesis and VLSI architectures for general-purpose microprocessors. He is a senior member of the IEEE.