AREA-EFFICIENT DIGITAL BASEBAND MODULE FOR BLUETOOTH WIRELESS COMMUNICATIONS

Myoung-Cheol Shin, Seong-Il Park, Sung-Won Lee, Se-Hyeon Kang, and In-Cheol Park

Division of Electrical Engineering, KAIST, Taejon, Korea

ABSTRACT

This paper describes a small and portable digital baseband module developed for Bluetooth wireless technology. To achieve portability and the small size, much of the Bluetooth baseband layer protocols are implemented in software running on the embedded microcontroller while the minimal tasks of low-level baseband processing, UART and USB interfaces, and audio CODEC are performed on the dedicated hardware blocks. The fully synthesizable baseband module was fabricated in 0.25μm CMOS technology occupying 2.25×2.25mm^2 area.

1. INTRODUCTION

Due to progress in related technologies in the past decades, the Bluetooth specification [1] was developed in 1999 to substitute cables connecting portable or desktop devices and to build low-cost wireless networks. It emphasizes low complexity, power consumption, and cost target [2]–[4]. It is crucial to implement digital baseband processing of Bluetooth in hardware and desirable to integrate the whole system on a chip to achieve the power and cost objective [4]. Baseband modules as IP (intellectual property) cores enable those higher levels of integration through SOC (system-on-chip) design and reduce time to market.

The tasks that the baseband module should perform vary significantly depending on the Bluetooth application. For the simplest applications such as wireless headsets or cellular phone add-on dongles, even the entire application as well as a basic part of baseband layer protocols may be implemented in software on the baseband processor. Complex ones, on the other hand, expect the baseband controller to support full baseband and host controller interface (HCI) functions, while the more complex upper layer protocols are processed on a host processor. The baseband module should therefore be very flexible and programmable not to waste the hardware resources and processing power in any case.

This work was supported by the Korea Science and Engineering Foundation through the MICROS (Micro Information and Communication Remote Object-oriented System) research center at KAIST, Taejon, Korea.

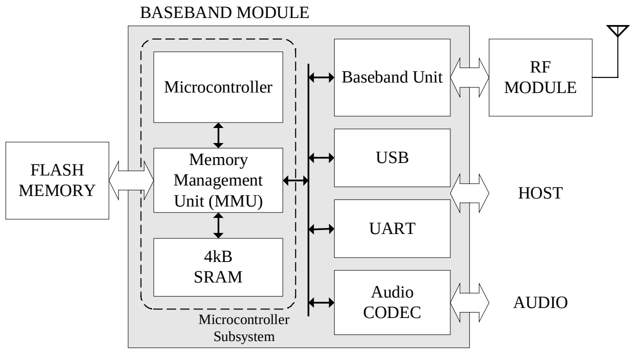

Figure 1. Hardware block diagram of the Bluetooth baseband module

Several baseband hardware cores have been developed either as a part of a system or as an IP [5]–[7]. Their size, however, are large because they perform almost all tasks in massive hardware while idling their microcontrollers [6], or because they use dual-port internal SRAMs [5].

This paper presents a simple, small, and portable Bluetooth baseband module that is suitable for use as an IP core. To reduce hardware area and gain more flexibility, the programmable embedded microcontroller performs as many tasks as possible and the hardware blocks implement only the most essential hardware functions. The module is made up of a logic part of only 85k gates and a 4kB single-port SRAM. Yet it performs all the essential Bluetooth baseband, link controller, link manager, and host controller interface (HCI) [1] functions including point-to-multipoint communications, multi-slot packets, encryption, scatternet, etc.

2. OVERALL ARCHITECTURE

As depicted in Figure 1, the baseband module consists of five major functional units: the microcontroller subsystem, baseband unit, UART, USB, and audio CODEC.

The microcontroller subsystem manages the other units and executes Bluetooth link manager, host controller interface, and some part of link controller protocol software. The baseband unit performs encoding and decoding of

Bluetooth bitstream and low-level timing control. It controls the RF module using a boundary scan serial interface. The USB and UART blocks implement Bluetooth host controller interface (HCI) physical transport layer. The audio CODEC processes voice data, and supports all the three Bluetooth audio coding: A-law, μ-law, and CVSD.

The primary clock input of the module is 48MHz. Whereas the USB runs at the 48MHz external clock in order to gain synchronization with 12MHz RX bitstream, the other units use a 12MHz clock that is generated by dividing the 48MHz clock by four to save power consumption. The other sub clocks that are used for interface between the baseband unit and an external RF module, such as 1MHz, 4MHz, and 3.2kHz, are provided by the RF module.

3. MICROCONTROLLER SUBSYSTEM

The microcontroller controls and manages the other units via memory-mapped I/O interface and interrupts. The other important task of the microcontroller is to run Bluetooth link manager, host controller interface, and a part of link controller protocol software. The microcontroller performs the complex part of the link control that require flexibility such as decision-making on received baseband packets and context switching between links, while the baseband unit performs bit-intensive, time-critical part. Although the link control tasks add some processing burden to the microcontroller, it still has excess processing power that may be used to run an entire protocol stack up to an application program for simple applications.

It is a clone of Advanced RISC Machines ARM7TDMI core [8] that we have developed as the microcontroller of our Bluetooth baseband module. We have designed the clone in order to take advantage of small die size and good code density of the ARM7TDMI, and to utilize its development environment. The 32-bit microcontroller core is more suitable for our goal of minimizing hardware through much flexibility than 8- or 16-bit low-performance cores. We used only public-domain documents and information of the core to develop the clone. The most part of the software is coded in Thumb instructions, which are 16-bit instructions requiring less bus width and memory size than 32-bit ARM instructions.

Because the microcontroller, especially its register file, occupies major portion of chip area, we have adopted a couple of schemes to reduce its size. Using a latch-based single clock register file structure that stores register data in slave latches and has one master flip-flop at input, we achieved a 27% area and 40% power reduction of the datapath with respect to the conventional flip-flop-based structure. Furthermore, we removed exception modes of ARM7TDMI unnecessary to our design, i.e. fast interrupt request (FIQ), abort, and undefined modes. Removal of those modes eliminates several banked registers and greatly simplifies the register file decoder, which results in additional 7% area reduction of the datapath.

The memory management unit (MMU) in Figure 1 manages memory interface and memory-mapped I/O interface of the microprocessor. One of the most important tasks of the MMU is direct memory access (DMA) of peripheral units. If the baseband unit and HCI units have their own buffers implemented with flip-flops, the buffers will dominate the size of those units and impose a sizeable burden on the microcontroller to move the data. We moved the large buffers into the internal SRAM and implemented a DMA, which results in a great area reduction. Compared to the distributed buffer architecture, the logic gate count is reduced from 132,000 to 85,000 (35.7% reduction). The net area reduction is 6.1% taking the added 4kB on-chip SRAM into account. However, we can also use the RAM for storing microcontroller program data as well as for buffering. A 4kB or 8kB RAM is sufficient to run a whole simple application program on the microcontroller, whereas the memory-intensive L2CAP (logical link control and adaptation protocol) segmentation and reassembly of complex applications such as PC Internet access may be performed on a host.

We use a single-port on-chip SRAM, which is 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, the DMA can be easily implemented with a small single-port SRAM.

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

4. BASEBAND UNIT

The baseband unit performs the bit-intensive baseband protocol functions that are power-efficient if implemented in hardware, i.e. Bluetooth bitstream processing and encryption. In addition, the most time-critical portion of the link controller tasks, such as low-level timing control and frequency hop calculation, are processed by the baseband unit. The complex part of the link controller requiring flexibility runs on the microcontroller in order to make the hardware small and less complex. The baseband unit conforms to the latest version of the Bluetooth specification [1] and supports all of the six ACL, four SCO, and four common packet types.

Figure 2 shows a block diagram of the baseband unit. TX and RX buffer blocks in Figure 2 transfer data received through DMA to the bitstream processing block using double buffering. At the same time they may generate interrupt signals that occur during data transmission and reception. CLK Offset Control logic controls three Bluetooth-specified clocks: CLK, CLKN, and CLKE. The Bluetooth clocks feed the hop selection logic that generates a hopping sequence for the 79-hop system.

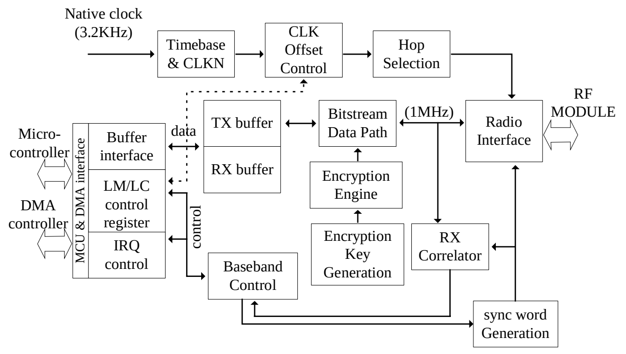

Figure 2. The block diagram of baseband unit

The baseband unit is directly connected to an RF module via Ericsson's RF module interface. The control of the RF module is achieved through a serial control interface based on the IEEE standard 1149.1 boundary scan architecture. The interface can be easily retargeted to other RF front-end modules.

5. HOST CONTROLLER INTERFACE UNITS

For data transmission between the Bluetooth baseband module and a host such as a PC, PDA, or cellular phone, two serial interfaces, USB and UART, are provided. The USB and UART units constitute the physical part of Bluetooth HCI. To reduce chip area, they implement only the most basic hardware parts while the microcontroller performs the complex flow control.

5.1. USB

The USB (Universal Serial Bus) unit complies with USB Specification 1.1 [9] and HCI USB transport layer specification of Bluetooth v1.1 [1], and supports full-speed 12Mbit/s host controller interface.

As shown in Figure 3, the USB unit consists of transceiver interface, serial interface engine, protocol layer handler, registers/endpoint manager, and parallel interface. The transceiver interface block drives the transceiver while sending data, and contains an RX clock recovery circuit. The serial interface engine encodes, decodes, and

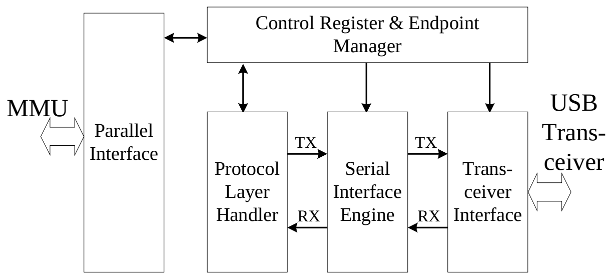

Figure 3. USB unit block diagram

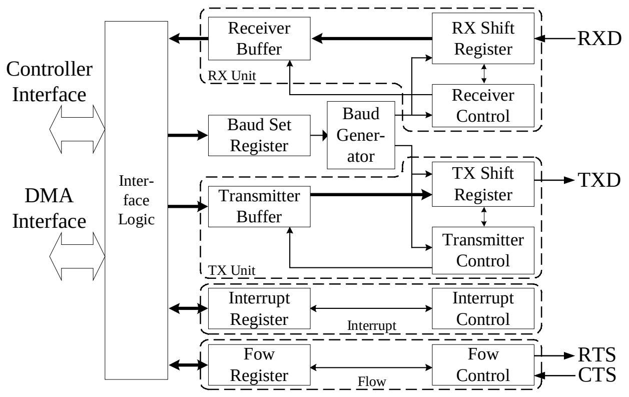

Figure 4. Architecture of the UART unit

samples signals at the recovered clock. Protocol layer handler works as a transaction sequencer, which performs the control to send or receive packets. The packets are stored in the corresponding endpoint buffer memory via DMA. The different types of HCI packets are mapped onto different USB endpoints according to the Bluetooth specification [1].

5.2. UART

The UART (Universal Asynchronous Receiver Transmitter) unit is designed based on industry-standard 16C450. It supports from 300bit/s to 1.5Mbit/s, and the default bit rate is 57.6kbit/s.

As shown in Figure 4, the UART unit consists of TX unit, RX unit, interrupt block, flow control, and interface block. The TX unit converts the parallel data into a serial form to transmit them. 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 TX unit, 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.

6. AUDIO CODEC

Bluetooth specifies three audio coding techniques: Log PCM coding using either A-law or μ-law [10] and CVSD (continuous variable slope delta modulation) [1]. Since the table lookup of log PCM and low-pass filtering required in CVSD to avoid aliasing is appropriate for hardware implementation, we implemented all the three coding methods in a hardware audio CODEC. For simple audio applications, the audio unit interfaces directly to PCM audio devices without using HCI. The interface is designed to pass 8- to 16-bit decoded linear PCM signals. For PC applications, coded audio bitstream can also be transmitted

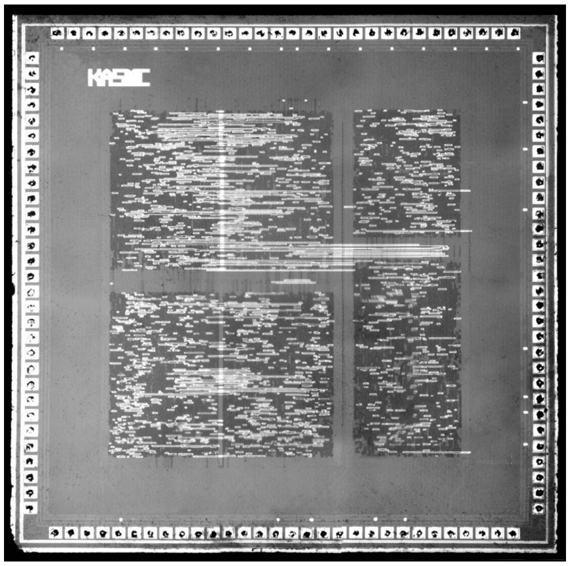

Figure 5. Die microphotograph of the prototype chip

via synchronous connection-oriented (SCO) HCI data packets through USB or UART.

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

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

7. CHIP IMPLEMENTATION

We have implemented a prototype baseband module in 0.25μm CMOS technology. A die microphotograph is shown in Figure 5, and the characteristics of the chip is summarized in Table I. The prototype chip contains no on-chip SRAM.

All the blocks of the chip are described in Verilog HDL and fully synthesizable. The synthesized gate-level design is autorouted to implement the chip. The prototype chip is fully tested using IMS ATS2 test station to verify its functionality and timing.

In parallel with the fabrication of the chip, we constructed an FPGA test board in order to validate the hardware and software in real-time environment. Including our design mapped onto a Xilinx Virtex XCV1000 FPGA chip, the test PCB board contains a flash memory, external RAM, Ericsson PBA313 01 RF front-end module, and antenna.

We confirmed the point-to-point connection capability of the baseband module with two test boards, each of which is connected to a PC through UART. Two boards successfully established a connection and transferred bit-streams and files.

Table I. Major chip characteristics

Technology	0.25μm CMOS 5LM
Core size	$2.25\times2.25\text{mm}^2$
Gate count	85,000
Supply voltage	2.5V
Operating clock	48MHz

8. CONCLUSIONS

A small, flexible baseband module for Bluetooth wireless connection technology has been presented.

We implemented a prototype chip in 0.25μm CMOS technology. The chip occupies only $2.25\times2.25\text{mm}^2$ core area, and its functionality and timing characteristics are tested using a test station.

As the module is implemented in the way that the most of the tasks are implemented in software except the indispensable hardware-efficient functions, it can be adjusted to various Bluetooth applications without wasting hardware resources and processing power. Since the module is a small, fully synthesizable soft core, it can be easily integrated as an IP core on SOC ASICs for the applications related to Bluetooth communications, which embodies the baseband module, RF front-end, and even host application circuits.

REFERENCES

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

[2] J.C. Haartsen, and S. Mattisson, “Bluetooth—A New Low-Power Radio Interface Providing Short-Range Connectivity,” *Proceedings of the IEEE*, vol. 88, pp. 1651–1661, Oct. 2000.

[3] R. Shorey and B.A. Miller, “The Bluetooth Technology: Merits and Limitations,” *IEEE International Conference on Personal Wireless Communications*, pp. 80–84, 2000.

[4] J. Bray and C.F. Sturman, *Bluetooth: Connect Without Cables*, Prentice Hall PTR, Upper Saddle River, NJ, 2001.

[5] F. O. Eynde, et al., “A Fully-Integrated Single-Chip SOC for Bluetooth,” IEEE International Solid-State Circuits Conference, pp. 196–197, and 446, 2001.

[6] Ericsson Microelectronics, *EBCP CherryRed Bluetooth Baseband IP*, May 2001.

[7] Cambridge Silicon Radio, *BlueCore01b Product Data Sheet*, July 2001.

[8] Advanced RISC Machines, *ARM7TDMI Data Sheet*, Aug. 1995.

[9] *Universal Serial Bus Specification 1.1*, Sep. 1998.

[10] International Telecommunications Union, *ITU-T Recommendations G.711 Pulse Code Modulation (PCM) of Voice Frequencies*, Nov. 1988