Implementation of a Ultra Wideband Video Streaming MAC with Customizable Microcontroller (CAP)

Hans-Joachim Gelke
Institute of Embedded Systems
Zurich University of Applied Sciences
Technikumstrasse 20/22
CH-8401 Winterthur, Switzerland
hans.gelke@zhaw.ch

Daniel Alberti
Institute of Embedded Systems
Zurich University of Applied Sciences
Technikumstrasse 20/22
CH-8401 Winterthur, Switzerland
daniel.alberti@zhaw.ch

ABSTRACT
Ultra Wide Band, a new wireless transmission technology for high bandwidth (480 Mb/s), short range (10-50m) is currently making its way into mainstream PC [5]. While the main market of UWB is to replace the cable of USB, there are many other applications, for which this technology is ideally suited for. E.g., uncompressed video streaming, wireless data hubs, medical instrumentation. The semiconductor mass market focuses mainly on PC applications, which implies that most commercially available UWB chips are Wireless-USB hubs or PCI bus hosts. Unfortunately, this leaves the market uncovered for applications, which require low overhead data streaming. This paper describes a WiMedia compliant UWB media access controller (MAC) realized with the AT91CAP9S metal mask customizable microcontroller from Atmel. The MAC interfaces to a UWB-PHY module according to the ECMA-369 MAC-PHY interface standard. Data rates of up to 355 Mb/s, utilizing isochronous data transfers, can be achieved. The integrated ARM9 processor of the Atmel CAP allows UWB beacon and distributed reservation protocol (DRP) processing to be controlled by software such that the hardware design can be kept as simple as possible. The MAC is configurable by Software. This means that the same mask programmable SoC can be used in the receiver and the transmitter. The CAP with metal mask programming area has a power consumption as low as standard microcontrollers. Due to these features, the CAP SoC is ideally suited to handle e.g. high quality video wireless transmission in a battery powered portable medical instrument. [1]

1. INTRODUCTION
1.1 Background
For removal of obstacles in the human larynx, or to insert breathing tubes, physicians use so called laryngoscopes. A laryngoscope is an instrument to view the larynx of a patient and consists of a handle and spatula as shown in picture 2. In order to examine the larynx, this instrument must be shaped ergonomically and easy to operate. The physician inserts the instrument into the mouth and lifts the lower jar with a gentle push upwards. This method may cause some complications, since there is the danger to break the incisors of the patients. Additionally, about 20% of patients show an unexpected difficult airway, which means, the instrument can not be used.

Figure 1: Laryngoscope used on a patient

A solution to these problems provides a miniature camera, built into the tip of the spatula, which improves the sight of the laryngoscope (see Fig.2) In systems, currently on the market, a miniature camera is connected to a monitor via a cable; this cable is hindering the operation room or in an emergency car. For this reason, a wireless connection between laryngoscope and camera makes the handling more convenient.

For reasons of picture quality and avoidance of distortions, it has been decided, that the transmission should be digital. For the comfort of the physician, movements of the object should appear on the screen with little delay, which implies
that the video signal may not pass through compression circuits or extensive layer stacks. Due to the high bandwidth requirements (160Mbit/s for a picture in NTSC quality) and the operating range of the near field (1-10m), UWB is the suitable transmission media for this instrument.

1.2 Camera Architecture

The video sensor is a CMOS type. In order to compensate for data rate fluctuations on the transmission line and to allow for repeated transmission of video frames, the image is captured in a frame store before being sent to the UWB MAC. A tightly coupled Microcontroller bus (Amba Host Bus) between the UWB MAC and the Microcontroller allows most UWB functions, like beacon generation, to be software supported. The UWB unit consists of a UWB Transceiver (UWB PHY) and the UWB streaming Media Access Controller. Since the camera unit is battery powered, care must be taken to keep power consumption to a minimum, such that the system can be operated at least 2 hours on one battery charge. The battery charging system is controlled by the microcontroller as well.

1.3 Main Principle of the UWB Streaming MAC

User data (mainly video content) and the tables required for UWB network management are contained in memory. This memory could be internal to the MAC ASIC or bulk memory outside of the MAC device. To provide a continuous data flow from the data source (camera) to the destination (display), the UWB MAC utilizes isochronous data streaming. To reach this goal, a concept consisting of a fast user data path and a scheduler was developed. (Figure 4 Fast User Data Path formats the data stream according to the UWB protocol defined in the ECMA-368 standard. A scheduler, controlled by the firmware, sends the packets to the UWB PHY according to the UWB MAC timing requirements. This way, the non time critical functions like network management and resource negotiation, between the MAC on the camera side and the MAC on the display side can be implemented in the firmware of a microcontroller. The Fast User Data Path and the scheduler need to be implemented in hardware. In addition to classic MAC layer functionality, the firmware also provides simple link control functionality for point to point transfers like opening a connection with the display device and specifying the desired bandwidth.

1.4 Firmware Beacon Processing

UWB transmission is organized in super frames[7] (see Figure 5). A super frame is divided in Beacon Period (BP) and payload. Beacons and payload occupy 256 Medium Access Slots of the super frame.

Starting at the end of the beacon period, the firmware processes the received beacons and calculates its own beacon for the next super frame.

Figure 6 shows the Beacon Processing of two devices A and B during two super frames. The horizontal axis is the timeline from left to right. Both devices send Beacons during the beacon period (BP) of each super frame[4]. During the time the payload of super frame[n-1] is transmitted, the firmware of device-A processes the beacon information element sent by device-B. At the time super frame[n-1] has finished transmitting, the firmware completed the information for device-B, which appears in the Beacon Information Element (IE) for device-A in super frame [n]. As the MAC program can
be seen as a real-time thread, a request and response queue is used for communication, with the application.

Figure 7 (see top of next page) shows a communication cycle between device-A and -B. The Application of device-A makes a request. Within SF\[n-1\] the Firmware of device-A generates the Beacon IE Request. The request will be transmitted in the BP of SF\[n\]. Device-B receives this beacon and calculates its answer in the data period SF\[n\]. The beacon received in SF\[n+1\] can now be interpreted by device-A.

2. IMPLEMENTATION

2.1 Overview

Since the device is battery operated, power consumption becomes an issue. As described in Section 1.2, the architecture of the MAC requires a tight coupling between scheduler and micro controller. The CMOS sensor is controlled by an I2C bus, and the battery management requires A/D converters and a Pulse Width Modulator (PWM). Parts count should be reduced to a minimum, since everything has to fit into the handle of the laryngoscope. Therefore a standard cell ASIC would be the obvious choice; however volumes don’t justify the development costs. For these reasons, the Atmel CAP, an ARM based micro controller with mask programmable custom logic, combining high integration, low NRE, low parts cost and low power, is the ideal solution for this application.

The following chapter describes the implementation of the UWB MAC fitted to the architecture of the Atmel CAP.

Figure 8 shows the Block Diagram of the AT91CAP9. The UWB MAC is implemented in the user metal programmable block, which is connected to the ARM9 AHB through a 6-layer AHB matrix.

To take advantage of cheap bulk memory, the RAM for Data and UWB management (see also section 1.3) is located external to the CAP and connected to the EBI interface. The multi layered AHB matrix of the CAP9 Architecture supports this very well, by allowing AHB bus masters to be implemented into the metal programmable block. The MAC becomes a bus master when accessing data from external bulk memory via the External Bus Interface (EBI). For memory that must be accessed without delay, like beacon frame generation, SRAM on the CAP is used.

Figure 9 shows a conceptual overview of the UWB MAC architecture. The top half of the picture shows the fast data path between the MAC-PHY Interface (MPI) and the AMBA Advanced Host Bus (AHB) of the CAP. External Memory lookups are performed via the RX- and TX-Endpoints. Each endpoint contains an address pointer, which determines the start address, end address and read/write of the memory location addressed by the AHB master. Each table or memory area requires an endpoint. The MAC PHY Interface (MPI) is the interface to the UWB physical layer; it complies with the ECMA-369 standard [2].

2.2 TX Data Stream

The TX data stream (see Fig. 10) is a high speed data path for all frames sent from the application to the PHY. The data source can be memory space or a single register in the AHB addressing space. The AHB Master acts like a DMA. On request, the TX AHB master begins reading from the memory buffer and transfers the data to the MPI-interface [3]. When the bus is granted, the Master begins a sequential transfer. Burst size, start address, direction and wrap boundaries are provided by the TX endpoint through the AHB Master configuration interface. When transfers are pending on both,
the RX AHB Master and the TX AHB master, the TX AHB master has the priority. The TX-Endpoint block updates the MAC Header Fields according to its own configuration and the information provided by the fragmentation logic. The MPI-Interface is responsible for updating the PHY header fields according to the scheduler (For information contained in the MAC and PHY header fields see [1]). Header information is supplied by the Status/Header RAM, which is part of the tightly coupled memory blocks of the CAP. Alternatively, the header generation can be bypassed and the headers can be supplied directly by user data, which is sourced by the memory.

Figure 7: Device A requests data from device B

The endpoint implements a handshake interface with the scheduler. The Status/Header RAM stores status and configuration information for multiple endpoints. If the scheduler requests a frame, the FSM begins to load the configuration from the RAM to the AHB Master. Subsequently the status and configuration registers are loaded. If set by software, the endpoint now transmits the header from RAM to the TX-STREAM bus. Afterwards the FSM starts the AHB Master, which transfers data from external memory to the TX-STREAM bus. When the transmit operation successfully ends, the scheduler issues a success command. The FSM now writes the status information back to the RAM. If frame transmit is not successful, the error command prevents the FSM from writing status back. If B-ACK is used ([1]), the scheduler may send a request command before a success or error command is sent.

2.3 RX Data Stream

Figure 11 shows the RX data stream is a high speed data path for all Frames received by the PHY and sent to the application. When the Frame RX, TX Logic (FRT) receives a Frame from the PHY, it will be stored in the buffer of the FRT, until it is determined that the frame is addressed to this device and the CRC32 matches. If this is the case, the FRT signals the scheduler a newly available frame. The RX endpoint configures the RX AHB Master. Data is now transmitted from the FRT Buffer to the Memory space (or a Register within the AHB address space) by the AHB Master. If enabled, the RX Endpoint can discard the header before data is transmitted. Alternatively, the endpoint can completely discard the received Frame.

Figure 10: TX Data Stream with Endpoint

The AMBA AHB Master is responsible for storing data to an external memory from the RX Endpoint. When the bus is granted, the Master begins a sequential transfer. Burst size, start address, direction and wrap boundaries are provided by the RX Endpoint through the AHB Master Configuration interface.

The Frame Receive Transmit Logic (FRT) provides frame information, like UWB addressing, configuration and endpoint status to the RX-endpoint. When a new frame reception is signaled by the FRT, the FSM begins to search for a matching filter. If a valid filter was found, the Endpoint FSM loads the AHB Configuration data from the configuration RAM to the RX AHB Master. The frame will now be transmitted to an external memory. The RX counter fields within the RAM will be updated. Alternatively the endpoint does not search for a matching filter, but the scheduler provides a fix endpoint number. This is useful, when Frames/Links are distinguished only by time (for example DRP). The endpoints may be enabled independent of auto search and scheduler preselect. Reassembly of fragmented frames is done by simply storing the frames sequentially to the EBI data memory, whereas only the first frame stores its header. This reassembly works only when an endpoint is set to receive frames from a single address. Broadcast endpoints and out of order transmits are currently not supported by the design. To fully support reassembly, an intermediate storage is needed (either internal or external).
Figure 8: Blockdiagramm of CAP9
2.4 Frame Receive Transmit Logic (FRT)

![Figure 12: Block Diagram of the Frame Receive Transmit Logic (FRT)](image)

Figure 12 shows the FRT Interface. A CRC32 logic calculates the checksum of the frames. The calculation is done on the fly during receive or transmit of the frame. The FSM in the FRT receives commands from the Scheduler and initiates RX or TX operations. It also controls appending of the CRC for transmit frames. The status of the FRT-FSM is reported to the scheduler through the FRT Status Interface. The FRT-FSM is in lock step with the scheduler and controlled through the command lines and status register.

Data received through the FRT are stored in the RX Buffer of the FRT Interface. Only after the CRC and the destination address of the frame are valid, the FRT-FSM signals to the RX control block that a new frame has arrived. The RX control logic reads the header of the newly received frame and feeds information like size; type and source address to the RX endpoints. The RX endpoint controls reading the data out of the RX Buffer. An additional bit in RX Buffer signals synchronization of frame boundaries.

TX data will be formatted according to the configuration provided by the scheduler. After wards the data is stored in a TX Buffer. The main purpose of the TX Buffer is to compensate the latency of memory read data caused by EBI and AHB. When the FRT-FSM issues a TX Operation command, data will be transferred from the TX Buffer to the FRT. During the transfer a CRC will be calculated and appended at the end of the stream.

3. MAC Firmware

The UWB MAC firmware implements three protocol layers (see 13), the Link Control Layer, the MAC Layer and the Hardware Access Layer.

3.1 Link Control Layer

Since the firmware controls real time processes, message queues are necessary. The Message queues are implemented as circularly linked lists. Put and Get Operations are atomic. Based on a Link Control Request the Link Control Layer generates a DRP Information Element (IE) Request. A Link has a fixed bandwidth between two devices. A DRP Information Element (IE) describes the reserved MAS (data rate etc.) and is controlled by the MAC layer. The bandwidth is dependent on MAS count, data rate, burst mode and acknowledge policy. Because the maximum data rate, which can be transmitted in a MAS, depends on the received signal strength, the Link Layer needs to adjust the number of reserved MAS. Communication between Link Layers is done through the Link feedback IE.

3.2 MAC Layer

The MAC Layer is responsible for resource management. MAC’s communicate via beacons with each other. Received beacons are interpreted and a own Beacon for transmit is created. A Beacon Period Occupancy MAP and a MAS Availability MAP needs to be up to date, representing the Network allocation state.

3.3 Hardware Access Layer

The Firmware Interface is shown in Figure 14. The hardware is configured by programming of registers. For the Beacon communication at least one RX and one TX Endpoint is needed. The RX Endpoint needs to receive all Beacon frames. The beacons are stored in the memory, and can therefore be processed by the firmware. As the registers are memory mapped, configuring the MAC is easy. For the beacon reception a ring buffer in the memory needs to be managed. The TX Endpoint 0 needs to be updated, when a new own Beacon is available.

3.4 Firmware Execution

Figure 15 shows how the program is executed at the end of the beacon period. After program execution, a new own
beacon is ready for transmit.

Firmware processing is synchronized to Super Frame with an end of "Beacon Period Interrupt". Figure 16 shows the main tasks. It is split into three parts, before Beacon generation (pre), during beacon generation and after beacon generation (post). During beacon generation, the received beacons will be interpreted. First the device state will be updated. Afterwards new requests will be loaded from the request queue. A device generates its own Beacon only when the device is connected.

Figure 15: "End of Beacon Interrupts" trigger firmware processing

<table>
<thead>
<tr>
<th>Device State Update</th>
<th>Get new API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Request (Pre)</td>
<td>Network Update(Pre)</td>
</tr>
<tr>
<td>no</td>
<td>Connected?</td>
</tr>
<tr>
<td>yes</td>
<td>Create own Beacon</td>
</tr>
<tr>
<td></td>
<td>Network Update</td>
</tr>
<tr>
<td></td>
<td>(Beacon Processing)</td>
</tr>
<tr>
<td></td>
<td>API Request (Beacon)</td>
</tr>
<tr>
<td></td>
<td>Complete Beacon</td>
</tr>
<tr>
<td></td>
<td>Network Update (Post)</td>
</tr>
<tr>
<td></td>
<td>API Request (Post)</td>
</tr>
</tbody>
</table>

Figure 16: Firmware flowchart

3.5 Supported Information Elements
So far the following Information Elements will be supported by firmware.

- Distributed Reservation Protokoll (DRP)
- DRP Availability
- Application Specific IE(Video Link Information)
- LinkFeedback
- MAC Capabilities
- PHY Capabilities

3.6 Conclusion
The prototype was equipped with the RTU7010 UWB PHY evaluation module from Realtek. At a brutto data rate of 320Mbits and at a distance of 3m, bit error rates of 0 - 1.5 % could be achieved. The transmission is also able to pass through a 30cm brick wall without degradation. Since the design is intended for a medical application, the volumes are below 10k annually. Since these volumes do not justify a custom radio design, the UWB radio module UWY6500 from AboCom, which is based on the RTU7010, is used in production.

The UWB MAC IP is written in VHDL and can be purchased from Institute of Embedded Systems.

4. REFERENCES