A FEC architecture for UWB system

Sung-Woo Choi¹, Sang-Sung Choi¹, Han-Ho Lee²
Wireless Home Network Research Team, ETRI¹
{csw9908, sschoi}@etri.re.kr¹
School of Information & Communication Engineering, In-ha University²
hhlee@inha.ac.kr²

Abstract — The MB-OFDM UWB for IEEE802.15.3a uses a R=1/3 convolutional code for the error correcting code. Furthermore, it uses a Reed-Solomon code to protect the PLCP header from impairment. In this paper, we show a detailed hardware architecture and implementation results of FEC (forward error correction) for the MB-OFDM UWB. We propose a Viterbi decoder with radix-4 ACS to resolve the timing problem and we show important parameters which are necessary for the hardware design. The estimated hardware size of the Viterbi decoder is 120k gates and the clock frequency is 144MHz. The proposed Reed-Solomon decoder uses a modified euclidean algorithm to solve a key equation. This RS decoder has low latency and small area using simple and regular processing elements for the key equation solve block. The RS decoder is 27k gates and it can operate at a clock frequency of 232MHz, and has the latency of 46 symbol clocks. When RS code is applied to the whole data field, the area increased to about 3 times.

Keywords — UWB, Viterbi, Reed-Solomon, Channel code

I. INTRODUCTION

The Ultra Wideband (UWB) is a wireless technology for transmitting digital data at very high rates over a wide spectrum of frequency bands using very low power. The UWB is ideally suited for wireless communications, particularly short-range and high-speed data transmissions for local area network applications. This technology has advantage of high speed enabling multimedia streaming in the home.

The UWB is being standardized in IEEE 802.15.3a task group. The MB-OFDM use a convolutional code for the channel code and use the Viterbi decoder for decoding that code. In this system, the Viterbi decoder should provide the maximum data rate over 400Mbps. To implement this Viterbi decoder, a very high speed operation clock is needed. Additionally, the MB-OFDM UWB adopts the (23,17) RS code, which is the shortened architecture of the (255,249) Reed-Solomon code to protect a PLCP header. In this system, Reed-Solomon encoding is mandatory but decoding is optional. So, we focused on the hardware architecture of the RS decoder which has low latency and small area.

This paper is organized as follows. Section 2 summarizes the specification of the MB-OFDM UWB. Section 3 proposes the architecture of the Viterbi decoder and RS decoder. And section 4 presents the implementation result of the proposed architecture using an ASIC technology. Finally, the conclusions are given in section 5.

II. UWB SYSTEM

The MB-OFDM UWB uses 3.1-10.6 GHz frequency as 14 sub-bands that have a spacing of 528 MHz. In this system, bands 1 through 3 are used for mandatory mode. The remaining channels are reserved for future use. In Figure 1, a band plan for MB-OFDM is illustrated[1].

Channelization for different piconets is achieved by using different time-frequency codes for different piconets. In addition, different preamble patterns are used for the different piconets. Table 1 shows the TF code and preamble pattern for each piconet.

The MB-OFDM should support various data rates as described in Table2. The system uses 122 sub-carriers that are modulated using QPSK or DCM. 100 sub-carriers are used for data signal and 22 sub-carriers are used for pilot signal. Forward error correction coding is used with a coding rate of 1/3, 1/2, 5/8 and 3/4. Total 31 OFDM symbols, which is the preamble, shall be added prior to data to aid the receiver in timing synchronization, carrier-offset recovery, and channel estimation.

Table 1. Time-Frequency Code and Preamble Pattern for different Piconets

<table>
<thead>
<tr>
<th>Channel Number</th>
<th>Preamble Pattern</th>
<th>Mode 1 : Length 6 Time Frequency code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1 2 3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1 3 2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1 1 2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1 1 3</td>
</tr>
</tbody>
</table>

The MB-OFDM uses frequency spreading and time spreading to enhance performance. Frequency spreading makes complex conjugate input to IFFT on the half of sub-carrier. For rates of 53.3, 80, 106.7, 160 and 200 Mbps a time-domain spreading operation is performed with a spreading factor of 2. The time-domain spreading operation consists of transmitting the same information over two OFDM symbols. These two OFDM symbols are transmitted over different sub-bands to obtain frequency diversity as described in Table1.
1. FEC ARCHITECTURE

The MB-OFDM UWB system uses the convolutional code for FEC. The convolutional encoder shall use the rate R=1/3 code with the generator polynomials, \( g_0 = 133_8 \), \( g_1 = 145_8 \), \( g_2 = 175_8 \). Additional coding rates are derived from the rate R=1/3 convolutional code by employing puncturing. In the receiver, the FFT input data contains the time and frequency domain spreading overhead. After the de-puncture process, the data is transferred to the Viterbi decoder without the spreading overhead. Because the coding rate of Viterbi decoder is 1/3, the minimum required clock of operation is 1/3 of the input data rate. Data rates of this system and corresponding clocks are summarized as follows.

Table 2. Rate-dependent Parameters

<table>
<thead>
<tr>
<th>Data rate (Mb/s)</th>
<th>Modulation</th>
<th>Coding rate</th>
<th>Conj. Symmetric to IFFT</th>
<th>Time spreading</th>
<th>spreading gain</th>
<th>Coded bits / 6 OFDM symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>53.3* QPSK</td>
<td>1/3</td>
<td>Yes</td>
<td>2</td>
<td>4</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>80 QPSK</td>
<td>1/2</td>
<td>Yes</td>
<td>2</td>
<td>4</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>106.7* QPSK</td>
<td>1/3</td>
<td>No</td>
<td>2</td>
<td>2</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>160 QPSK</td>
<td>1/2</td>
<td>No</td>
<td>2</td>
<td>2</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>200* QPSK</td>
<td>5/8</td>
<td>No</td>
<td>2</td>
<td>2</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>320 DCM</td>
<td>1/2</td>
<td>No</td>
<td>1</td>
<td>1</td>
<td>1200</td>
<td></td>
</tr>
<tr>
<td>400 DCM</td>
<td>5/8</td>
<td>No</td>
<td>1</td>
<td>1</td>
<td>1200</td>
<td></td>
</tr>
<tr>
<td>480 DCM</td>
<td>3/4</td>
<td>No</td>
<td>1</td>
<td>1</td>
<td>1200</td>
<td></td>
</tr>
</tbody>
</table>

(* - mandatory Data rate)

2. Viterbi decoder

The Viterbi decoder’s critical path is in the ACS(add compare select) block. Because it contains a feed-back loop, there is a limit of pipelining data process. However, since the critical path still cannot be made arbitrarily short, the throughput rate depends on the clock rate of the realization, and that is limited. The classical high throughput implementation for such decoders is the radix-2 fully parallel approach, where ACS unit is assigned to each state and organized in pairs to iterate one stage of a two-state trellis. An alternative approach to high throughput is to exploit new forms of concurrency within a decoder by reformulation of the algorithm. However, since new algorithm needs much more hardware, we adopt one-stage look-ahead to radix-2 algorithm and use a radix-4 ACS architecture[2]. Because the radix-4 architecture is adopted, two symbols are processed at the half rate of conventional architecture. Consequently the operation clock is the half of radix-2 architecture as described in Table 3. For the mandatory data rate, the maximum clock speed is 100MHz, and that could be implemented using current ASIC technology.

In Table3, the maximum data rate(480Mbps) needs an operation clock frequency over 240MHz. It is very hard to solve critical path in the ACS block using an current ASIC technology. So, for the maximum data rate, more parallel architecture such as radix-2^2 ACS unit should be considered or bit-wise ACS optimization should be conducted.

In current implementation, the Viterbi decoder is consist of 3 blocks such as BM(branch metric), ACS(add compare select), and TB(trace back). The BM block calculates branch metric from input 2 symbols. A symbol means successive 3 encoded signals according to 1 message because of the base code rate(R=1/3). We use integers to calculate symbol metrics. In fact, use of the integers as symbol metrics instead of log likelihoods results in negligible performance degradation with 2-, 4-, or 8-level receiver quantization [3]. We choose 3 bit soft decision for the Viterbi decoder input after fixed point simulation. The relation between input soft bits and branch metric is Table4.

Table 3. Viterbi decoder input data rate and Min. clock

<table>
<thead>
<tr>
<th>Data rate (Mb/s)</th>
<th>Depuncture out (Mb/s)</th>
<th>Spreading gain</th>
<th>VD Input Data Rate (Mb/s)</th>
<th>Min. clock for VD (MHz)</th>
<th>Min. clock for Radix-4 (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>53.3* QPSK</td>
<td>640</td>
<td>4</td>
<td>160</td>
<td>53.3</td>
<td>27.5</td>
</tr>
<tr>
<td>106.7* QPSK</td>
<td>640</td>
<td>2</td>
<td>320</td>
<td>106.7</td>
<td>53.3</td>
</tr>
<tr>
<td>200* DCM</td>
<td>1200</td>
<td>2</td>
<td>600</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>480 DCM</td>
<td>1440</td>
<td>1</td>
<td>1440</td>
<td>480</td>
<td>240</td>
</tr>
</tbody>
</table>

(* - mandatory Data rate)

The proposed Viterbi decoder uses radix-4 ACS as mentioned before. Every 64 states have ACS unit, find optimal previous state and send current decision value to the TB unit. Every state contains 4-way ACS block as depicted in Fig 4. 4-way ACS have four adders, one four-input comparator, one 4:1 multiplexer. A four-input comparator consist of 3 adders(2 stage). So total critical path is 3 adder and one multiplexer over two symbol time, which is compared to conventional radix-2 ACS’s critical path that is 2 adders and one multiplexer over one symbol time.

The relation between input soft bits and branch metric is Table4.

Table 4. Branch metric according to input soft bits

<table>
<thead>
<tr>
<th>bm0</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>111 (3)</td>
<td>011 (2)</td>
<td>001 (1)</td>
<td>000 (0)</td>
<td>111 (7)</td>
<td>110 (6)</td>
<td>101 (5)</td>
</tr>
<tr>
<td>bm1</td>
<td>100 (4)</td>
<td>101 (5)</td>
<td>110 (6)</td>
<td>111 (7)</td>
<td>000 (0)</td>
<td>001 (1)</td>
<td>01 (0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The high speed Viterbi decoder, path metric normalization should be avoided. So we adopt proven modular
normalization algorithm[4]. To use that algorithm, path metric bit width \( (\Gamma_{\text{ hiret}}) \) should be decided using the equation(3)[2]. Because this Viterbi decoder receive 3 bit soft decision, and uses R=1/3 code, the maximum branch metric \( (\lambda_{\text{max}}) \) is 21. So maximum dynamic range \( (\Delta_{\text{max}}) \) is 126 . And the proposed architecture uses Radix-4 , then \( \Gamma_{\text{hi}} \) equals

\[
\Gamma_{\text{hi}} = \left\lceil \log_2(\Delta_{\text{max}} + k\lambda_{\text{max}}) \right\rceil + 1 \quad (3)
\]

\[
\Gamma_{\text{hi}} = \left\lceil \log_2((126 + 2 \times 21) + 1) \right\rceil = 9 \quad (4)
\]

\[
\Delta_{\text{max}} \leq \lambda_{\text{max}} \log_2 N \quad (5)
\]

\( \Delta_{\text{max}} \) : maximum dynamic range
\( \lambda_{\text{max}} \) : maximum branch metric
\( N \) : number of state
\( k \) : radix-2

Decoded output bits are obtained using a trace back method. The trace back method spends less power than register exchange algorithm, so we choose the trace back method. There are several known trace back algorithm. Among them, we choose K-pointer even algorithm with K=3. Because one-pointer algorithm needs a higher clock frequency than K-pointer algorithm, it is difficult to implement in this high speed Viterbi decoder. And K-point odd algorithm is difficult to control, it does not have advantage than K-pointer even and one-pointer algorithm.

The proposed architecture is simulated using a fixed point C program over AWGN channel. To decide soft input bit width, a BER curve is obtained as the input bit width of the Viterbi decoder varies.

![Fig. 5. Performance comparison for soft input bit width](image)

It shows that 3 bits soft input makes negligible performance degradation compared to 4 or 5 bits. In Fig5, soft decision makes 5.7dB coding gain, this is equivalent to theoretical coding gain. Soft decision is 2dB superior to hard decision at BER=10\(^{-5}\).

2. RS decoder

The MB-OFDM UWB transmit data frame composed of the PLCP preamble, the PLCP Header and the PSDU. At the receiver, the PLCP preamble and PLCP header serve as aids in the demodulation, decoding, and delivery of the PSDU. A PLCP header shall be added after a PLCP preamble to convey information about both the PHY and the MAC that is needed at the receiver in order to successfully decode the PSDU. Figure 6 shows the structure of the PLCP header.

![Fig. 6. PLCP Header structure](image)

In the modem transmitter, Reed-Solomon code is adopted for outer code to protect header data from impairment. In Fig. 6, the PHY header, MAC header and HCS, 17 bytes, are scrambled and encoded using RS(23,17) code and residual 6 parity bytes are appended. This code enables to correct maximum 3 bytes error including parity bytes. This RS(23,17) code is a shortened form of RS(255,249) code. The generator polynomial of the RS(255,249) is as follows.

\[
g(x) = \sum_{i=0}^{6} (x-\alpha^i) = x^6 + 126x^5 + 4x^4 + 158x^3 + 58x^2 + 49x + 117
\]

The codeword polynomial \( c(x) \) has relation with the message \( m(x) \) and the generator polynomial \( g(x) \) as follows.

\[
c(x) = x^{N-k} m(x) + \left[x^{N-k} m(x) \mod g(x) \right] = q(x)g(x)
\]

Because a RS code is one of systematic codes, it makes possible to use it as a mandatory code in the transmitter but optional in the receiver. Because the maximum data rate is 480Mbps and the one message of RS(255, 249) code is composed of 8 bits, the RS decoder shall process data at a clock frequency about 60 MHz. And the RS code is used only for small part of the whole data frame. So, the moderate speed and low complexity and low latency RS decoder is needed.

If no erasure is taken into consideration, a syndrome-based RS decoder consists of three components. First part is a syndrome computation (SC) block. It generates a syndrome polynomials \( S(x) \) that is used in the key-equation solver (KES) block for solving a key equation \( S(x)\sigma(x) = (x^{N-k} m(x) \mod g(x)) \). Either the modified Euclidean (ME) and Berlekamp-Massey algorithm (BM) can be used to solve a key equation for an error-locator polynomials \( \sigma(x) \) and an error-value polynomials \( \delta(x) \). Then in the third component, these two polynomials are used to find out the error locations and the corresponding error values according to the Chien search and Forney algorithm and correct the errors as the received word is being read out of the decoder. In addition, a first-in first-out (FIFO) memory is used in order to buffer the received symbols according to the latency of these components[5].

In our RS decoder, because modified Euclidean (ME) have no division operation and easily pipelined for high speed, we choose modified Euclidean algorithm (ME) for KES block. Figure 7 is the architecture of the proposed RS decoder using modified Euclidean algorithm.
In the case of other RS decoder with higher error correcting capability, a KES block consumes 60–70% of hardware of all blocks. But, this system uses the (23, 17) RS code correcting up to 3 errors, so the KES block of this system consumes much less hardware than other system. Consequently, in-out latency is the major factor to resolve the excellence of the RS decoder.

For the ME algorithm block, we choose a systolic array architecture of a linked algorithm processing cell which is easily pipelined, therefore, it is more suitable to high speed operation. Because the correctable error $t = 3$, total 6 ME processing cells are used. Fig. 8 is the block diagram of the ME calculation block.

In the ME calculation cell, the input data is computed and transferred to output after 2 delays. So, the total output delay of the ME calculation block is 12. In fig. 5, the control block examine the STOP signals from all cells and decide when the output buffer outputs the error locator polynomial and the error value polynomial.

### IV. IMPLEMENTATION RESULTS

The proposed FEC block was modeled in Verilog hardware description language(HDL) and simulated to verify its functionality. After that, it was synthesized using a TSMC 0.18um library.

The proposed Viterbi decoder uses 120k gates and operating clock frequency is 144MHz which satisfy the mandatory mode operation clock of 100MHz. But this result is not enough for the maximum data rate. The full rate Viterbi decoder needs 240MHz for clock frequency. This clock frequency is too difficult to suffice using the proposed architecture, so another algorithm or pipelined VLSI research for the Viterbi decoder must be conducted.

The synthesis result shows that the area of the proposed RS decoder is 27k gates and it can operate at clock frequency of 232MHz, which means it has a data processing rate of 1.8Gbps. This means that high speed RS decoder can be implemented using small area. The total latency of the proposed RS decoder is 46 symbol clocks. The FIFO of RS decoder is made by dual port RAM of the same size to the latency.

Also, we expanded the RS code to the data field using a RS(255,239) code and estimated the hardware complexity. This expansion can be easily done because the ME calculation cell is a systolic array architecture. The estimated area increased to 78k gates.

We make comparison in the case of a RS(255,239) code between our design and other existing chip solutions as listed in Table 5. The ME calculation block has more area than the architecture of using a recursive cell block[7] but has less latency. Compared to the high pipelined ME block[6], it has less speed but has less latency and area.

### Table 5. The comparison table of performance of RS decoder

<table>
<thead>
<tr>
<th></th>
<th>Tech. (um)</th>
<th>Latency (Cycles)</th>
<th>Clock (MHz)</th>
<th>Total of # Gates</th>
<th>Through put (Gbits/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>0.18</td>
<td>n+46</td>
<td>232</td>
<td>78,000</td>
<td>1.856</td>
</tr>
<tr>
<td>AP-ASIC’04[7]</td>
<td>0.18</td>
<td>n+(2t)^2+1</td>
<td>400</td>
<td>20,614</td>
<td>3.2</td>
</tr>
<tr>
<td>ISCAS’00[8]</td>
<td>0.25</td>
<td>n+3t+42</td>
<td>75</td>
<td>55,240</td>
<td>0.6</td>
</tr>
<tr>
<td>ISVLSI’03[6]</td>
<td>0.13</td>
<td>355</td>
<td>770</td>
<td>115,500</td>
<td>6.16</td>
</tr>
</tbody>
</table>

### V. CONCLUSION

This paper shows detailed hardware architecture and implementation results of the FEC (forward error correction) for the MB-OFDM UWB. The MB-OFDM UWB uses both...
convolutional code and Reed-Solomon code for FEC. The proposed Viterbi decoder uses the radix-4 ACS to resolve the timing problem and it is enough to support the mandatory data rate of the Viterbi decoder. The estimated hardware size of the Viterbi decoder is 120k gates. We also researched the RS(23,17) code of the MB-OFDM UWB. We showed the appropriate decoding algorithm and suggested the architecture with small area and low latency. From the synthesis result, the area estimation is 27k gates and the latency is 46 symbol clocks. The proposed architecture adopts a Modified-Euclidean algorithm for the key equation solver block which consumes most area in the decoder. For the ME block, we proposed the pipelined ME calculation cell with 2 delays. Because the ME calculation cell is a systolic array architecture, it is easily expanded to a higher RS code. Using this merit, we expanded a RS code to the data field of the MB-OFDM UWB and simulated function and synthesized the architecture. The synthesis result shows that the hardware increased to about 3 times when uses the RS(255,239) code in the data field.

REFERENCES