https://doi.org/10.7840/kics.2025.50.3.489

VDES 수신기를 위한 Turbo 복호기의 SIMULINK 모델링 및 FPGA 구현

김 재 형◆

SIMULINK Modeling and FPGA Implementation of Turbo Decoder for VDES Receiver

Jae Hyung Kim◆

요 약

VDES(VHF Data Exchange System)는 link ID 1~34에 대하여 부호율, 정보길이, 인터리빙(interleaving) 및 펑처링(puncturing) 방법 등을 정의하고, 다양한 MCS(Modulation and Coding Scheme)을 기반으로 대용량의 데이터의 고속 전송을 디지털 해상 통신 시스템이다. 본 논문에서는 VDES 수신기를 위한 Turbo 복호기를 FPGA에 구현을 하였다. VDES의 모든 link ID를 지원할 수 있는 Turbo 복호기 알고리즘을 SIMULINK를 이용하여 모델링하고 기능을 검증하였다. 검증된 SIMULINK 모델은 HDL 컴파일러에 의하여 Verilog로 변환하였고, FILS(FPGA In the Loop Simulation) 테스트벤치로 설계된 VDES 송신기 및 수신기를 이용하여 FPGA로 구현된 VDES Turbo 복호기의 성능을 시험하였다. VDES는 link ID에 따라 다양한 인터리빙 패턴은 물론, 최대 길이 6032의 부호기 입력을 지원해야 한다, 따라서 Turbo 복호기에서 메모리 사용량을 줄이기 위한 방법으로 길이가 32인 슬라이딩 블록 기법과 인터리빙 인덱스를 실시간으로 계산하는 방식을 적용하였다. 설계된 VDES 송수신기는 43.008MHz의 클럭으로 동작을 하면서 VDES 슬롯 구간 내에서 복조 및 복호를 완료할 수 있음을 테스트벤치 시험을 통하여 확인하였다.

Key Words : VDES, Turbo decoder, SIMULINK HDL compile, FILS

ABSTRACT

VDES(VHF Data Exchange System) defines the code rate, information length, interleaving, and puncturing methods for link ID 1~34, and is a digital maritime communication system that transmits large amounts of data at high speed based on various MCS (Modulation and Coding Schemes). In this paper, a Turbo decoder for VDES receivers is implemented in an FPGA. The Turbo decoder algorithm, which can support all link IDs of VDES, was modeled and verified using SIMULINK. The validated SIMULINK model was converted to Verilog by an HDL compiler, and the performance of the VDES Turbo decoder implemented in FPGA was tested using a VDES transmitter and receiver designed as a FPGA In the Loop Simulation test bench. The VDES must support encoding inputs with a maximum length of 6032 as well as various interleaving patterns depending on the link ID, so the Turbo decoder uses a sliding block with a length of 32 and a method of calculating the interleaving index in real time as a way to reduce memory usage. The designed VDES transceiver operates at a clock of 43.008 MHz and can complete demodulation and decoding within the VDES slot duration.

※ 이 논문은 2023~2024년도 창원대학교 자율연구과제 연구비 지원으로 수행된 연구결과임
◆ First Author : Changwon National University School of Mechatronics, hyung@changwon.ac.kr, 종신회원
논문번호 : 02409-217-B-RU, Received September 21, 2024; Revised October 27, 2024; Accepted November 12, 2024

Ⅰ. 서 론

최근 해상에서의 인터넷, 이메일 및 팩스 등의 디지털 통신에 대한 수요의 증가에 대응하기 위하여, IALA(The International Association of Maritime Aids to Navigation and Lighthouse Authorities)에서는 2013년 ITU-R Working Party 5B 회의를 통하여 VDES (VHF Data Exchange System)을 제안하였다[1]. VDES는 AIS(Automatic Identification System), ASM (Application Specific Messages) 및 VDE(Very High Frequency Data Exchange)를 통합한 시스템이며, 향후 e-navigation의 현대화를 위한 핵심 요소이다. 특히, VDE-TER과 VDE-SAT는 각각 지상 및 위성 통신을 위한 시스템이며, AIS에 비하여 다양한 MCS (Modulation and Coding Scheme)를 기반으로 대용량의 데이터를 고속으로 전송할 수 있는 시스템이다[2]. 이러한 요구 조건을 수용하기 위하여 VDES에 채택한 Turbo 부호는 34개 link ID에 따라 최대 6032의 코드 입력과 다양한 인터리빙(Interleaving) 패턴을 지원해야 한다.

그림 1은 VDES의 Turbo 부호기의 구조를 나타낸 것이며, 2개의 RSC (Recursive Systemic Convolutional) 부호기의 출력을 병렬로 결합한다. 단, 하나의 RSC에는 인터리빙을 한 입력을 사용함으로써 패리티 비트를 랜덤하게 생성하며, 따라서 Turbo 부호의 오류정정 능력을 Shannon limit에 근접하게 만들 수 있다[3]. 또한 link ID에 따라 1/2~5/6의 부호율을 제공하기 위하여 패리티 비트를 제거하는 펑처링 (puncturing) 기능을 사용한다. VDES의 link ID는 6개의 부호율, 16개의 펑처링 패턴 그리고 고유의 인터리빙 패턴 및 변조 방식을 정의한다[4].

Jung, Lee는 고속 Turbo 복호기를 FPGA에 구현하였다[5,6]. 이 연구는 radix-4 알고리즘, 병렬 디코딩 등의 기법을 적용하였으며 부호 길이가 긴 경우 약간의 성능 저하를 보이지만 디코딩 지연을 줄여 고속 무선 통신에서 실시간 복호가 가능하도록 설계를 하였다. 또한 Yang은 VDES를 위한 Turbo 복호기의 FPGA 구현을 하였다[7]. 그러나 Log-MAP의 근사적 방법인 Max-Log-MAP을 이용하였고, VDES의 최대 부호길이에 비례하는 많은 메모리를 사용하였다. 본 논문에서는 메모리 및 로직 자원을 효율적으로 사용하기 위하여 Log-MAP과 Max-Log-Map을 선택적으로 사용하며, VDES의 모든 link ID를 지원할 수 있는 Turbo 복호기를 FPGA에 구현하여 VDES 송수신 시스템 개발에 적용하고자 한다.

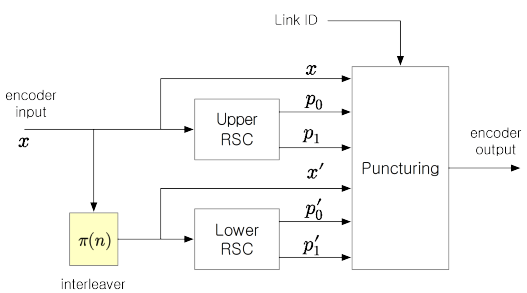


그림 1. VDES Turbo 부호기의 구조
Fig. 1. Structure of VDES Turbo encoder

본문은 다음과 같이 구성된다. 2장에서는 VDES의 Turbo 코드 규격을 기술하고, 3장에서 SIMULINK를 이용하여 VDES에 최적화된 Turbo 복호기의 모델을 설계하고 시뮬레이션을 수행한다. 4장에서는 HDL 변환된 Turbo 복호기의 기능을 FPGA In the Loop Simulation(FILS)을 이용하여 검증을 하고, 그리고 5장에서는 VDES-TER 송신기 및 수신기를 FIL 기반 테스트베드로 구성하고 설계된 VDES Turbo 복호기를 적용하여 성능을 확인한다. 마지막으로 6장에서 시험 결과를 평가하고 결론을 맺는다.

Ⅱ. VDES의 Turbo 부호

VDES의 Turbo 부호는 ETSI EN 302 583의 규격을 따르며 RSC, 인터리빙 및 펑처링 모듈로 구성된다[8,9]. 그림 1은 VDES Turbo 부호기의 구조를 보여준 것이다. RSC의 시스테믹(systemic) 부호 x와 패리티(parity) 부호 p_0, p_1, p_0', p_1'는 펑처링 블록에서 VDES link ID에 따라 다양한 부호율로 출력된다. 단, x'는 x의 인터리빙된 시퀀스를 표현한다.

2.1 RSC 부호기

RSC 부호기의 전달 함수는 식(1)로 표현되며, 그림 2의 구조를 가진다. 정보 비트 길이의 입력구간 이후에는 스위치의 위치를 데이터에서 테일로 변경하고 6 클럭 동안 테일 비트를 출력함으로써 부호기는 상태는 항상 0으로 종료 시킨다. 시작과 종료를 모두 0 상태로 강요하여 MAP 알고리즘을 적용할 수 있도록 한다.

$$G(D) = \left[1 \quad \frac{1+D+D^3}{1+D^2+D^3} \quad \frac{1+D+D^2+D^3}{1+D^2+D^3}\right] \quad (1)$$

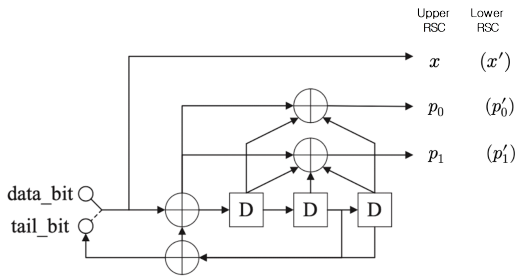


그림 2. RSC 부호기
Fig. 2. RSC encoder

2.2 VDES Turbo 부호의 인터리빙 및 펑처링 알고리즘

Turbo 부호는 버스트 오류를 랜덤화 하기 위하여 코드의 입력 순서에 인터리빙 알고리즘을 적용한다. 복호기가 인터리빙 순서를 복구하는 과정에서 버스트 오류를 산발적으로 분산된다. DES의 인터리빙 알고리즘은 n번째 순서의 입력을 식(2)를 이용하여 구해진 $\pi(n)$번째 입력으로 치환(permutation) 시킨다[9].

$$\pi(n) = 2(t + ck_1/2 + 1\ ) - m$$

$$\begin{aligned} \text{단,}\quad m &= (n-1) mod\ 2 \\ i &= floor\big((n-1)/(2k_2)\big) \\ j &= floor\big((n-1)/2\big) - ik_2 \\ t &= (19i+1) mod\ (k_1/2) \\ q &= t\ mod\ 8 + 1 \\ c &= \big(p_q j + 21m\big) mod\ k_2 \end{aligned} \quad (2)$$

여기서, k_1, k_2 및 p_q는 link ID에 의하여 고유한 값으로 정의된다. VDES에서 다양한 부호율은 ITU-R 표준안 [4]의 표 4에 정의된 펑처링 id의 조합에 따라 표준안의 표 5와 표 6에 정의된 패턴으로 출력 비트를 삭제하거나 또는 반복을 한다. 아래 표 1은 본 논문에서 시험하게 될 link ID 11, 17 및 19에 대한 채널 부호율, 인터리빙 파라미터 및 펑처링 패턴 파라미터를 보여준 것이다.

Ⅲ. Turbo 복호기 Log-MAP 알고리즘

3.1 Turbo 복호기의 구조[10,11]

Turbo 복호기는 디펑처링(de-puncturing) 블록과 반복적 연산을 수행하는 2개의 SISO (single-input-soft-output) RSC 디코더로 구성이 된다. 수신기에서 복조된 심볼은 $\tilde{x}$, $\tilde{p}_0$, $\tilde{p}_1$과 $\tilde{p}_0'$, $\tilde{p}_1'$로 분리되어 두개의 RSC로 각각 입력된다. Upper RSC 디코더의 출력은 인터리빙 후 lower RSC 디코더로 입력되고, lower RSC 디코더의 출력은 디인터리빙 후 다시 upper RSC 디코더로 입력되어 반복적으로 복호를 수행한다. 지정된 횟수의 반복적인 복호가 수행된 후, 소프트 출력 $\tilde{x}_e$의 하드 판정이 실행되어 복호 데이터로 출력한다. 그림 3은 재귀적 반복 연산을 수행하는 Turbo 복호기의 구조를 보여준 것이다.

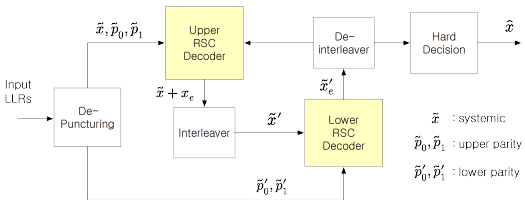


그림 3. Turbo 복호기의 구조
Fig. 3. The structure of Turbo decoder

3.2 SISO 디코딩 알고리즘

Turbo 복호기를 위한 SISO 디코딩 알고리즘으로는 MAP (Maximum-a-Posteriori) 또는 Soft-Output Viterbi Algorithm (SOVA) 기반의 알고리즘이 사용된다. SOVA 디코더는 하드웨어 구현이 어려운 단점 때문에 MAP 기반의 SISO 디코더가 Turbo 복호기의 하드웨어 구현에 주로 사용되고 있다. 그러나 MAP 기반의 디코더는 많은 메모리를 사용하고, 복호를 시작하려면 마지막 심볼이 필요하기 때문에 디코딩 지연이 커진다는 단점을 가지고 있다[12]. 이러한 단점은 슬라이딩 윈도우를 이용하여 효율적으로 구현이 가능하다는 것을 문헌[11]에서 제안하였다.

표 1. VDES link ID 11,17,19의 채널 부호 파라미터
Table 1. Channel encoding parameters for VDES link ID 11,17,19

Link id	변조방식	부호율	정보길이	k_1/k_2	p_q (q = 1, 2, …, 8)	Puncturing id (data/tail)
11	π/4 *QPSK*	1/2	432	2/216	127,191,241,5,83,109,107,179	6/6
17	π/4 *QPSK*	1/2	1872	6/312	211,61,227,239,181,79,73,193	6/6a
19	16 QAM	3/4	5616	16/351	137,101,223,41,67,131,61,47	8/8

3.3 Log-MAP 알고리즘

본 논문에서는 RSC 부호의 디코딩을 위하여 MAP 알고리즘이 사용하고자 한다. $\boldsymbol{u}$는 정보 비트 시퀀스, $\boldsymbol{y}$는 수신 심볼 시퀀스라 가정하면, MAP 디코더는 $\boldsymbol{k}$ 번째 수신 비트의 복호를 위하여 식(3)의 posteriori probability ratio를 계산한다[10-12].

$$\Lambda_k = \frac{P(u_k = +1 \mid \boldsymbol{y})}{P(u_k = -1 \mid \boldsymbol{y})} = \frac{\sum_{\boldsymbol{u}:u_k=+1} p(\boldsymbol{u},\boldsymbol{y})}{\sum_{\boldsymbol{u}:u_k=-1} p(\boldsymbol{u},\boldsymbol{y})} \quad (3)$$

MAP과 등가적이지만 로그(logarithm) 영역에서 동작하는 Log-MAP 알고리즘 연산은 FPGA와 같은 장치에서 현실적으로 구현이 가능하다. Log-MAP 소프트 판정 메트릭(soft-decision metric)은 식(4)와 같이 주어진다[10].

$$\begin{aligned} L_k &= \ln \sum_{\boldsymbol{u}:u_k=+1} p(\boldsymbol{u},\boldsymbol{y}) - \ln \sum_{\boldsymbol{u}:u_k=-1} p(\boldsymbol{u},\boldsymbol{y}) \\ &= \ln \sum_{\boldsymbol{u}:u_k=+1} e^{\ln p(\boldsymbol{u},\boldsymbol{y})} - \ln \sum_{\boldsymbol{u}:u_k=-1} e^{\ln p(\boldsymbol{u},\boldsymbol{y})} \\ &= \ln \sum_{\boldsymbol{u}:u_k=+1} e^{M(\boldsymbol{u},\boldsymbol{y})} - \ln \sum_{\boldsymbol{u}:u_k=-1} e^{M(\boldsymbol{u},\boldsymbol{y})} \end{aligned}$$

$$\text{단, } M(\boldsymbol{u},\boldsymbol{y}) = \ln p(\boldsymbol{u},\boldsymbol{y}) = \ln p(\boldsymbol{u}) + \ln p(\boldsymbol{y}|\boldsymbol{u}) \quad (4)$$

식(4)에서 볼 수 있듯이, Log-MAP의 계산에서는 곱셈이 덧셈으로 변환이 되었지만 아직도 비선형 지수함수 연산이 필요하다. 지수함수의 합은 Jacobian logarithm을 이용하여 식(5)로 다시 표현이 가능하다[12-14].

$$\begin{aligned} \max{*}\,(x,y) &\triangleq \ln(e^x + x^y) \\ &= \max(x,y) + \ln\left(1 + e^{-(|x-y|)}\right) \end{aligned} \quad (5)$$

3.4 Max-Log-MAP 알고리즘

$|x-y| > 7$의 영 역에서 $e^{-(|x-y|)} \cong 0$가 되므로 식(5)는

$$L_k \approx \max_{\boldsymbol{u}:u_k=+1} M(\boldsymbol{u},\boldsymbol{y}) - \max_{\boldsymbol{u}:u_k=-1} M(\boldsymbol{u},\boldsymbol{y}) \quad (6)$$

로 근사화 될 수 있다[13].

Max-Log-MAP 알고리즘의 소프트 판정 메트릭은 식(6)을 이용하여 식(7)로 주어진다.

$$L_k \approx \max_{\boldsymbol{u}:u_k=+1} M(\boldsymbol{u},\boldsymbol{y}) - \max_{\boldsymbol{u}:u_k=-1} M(\boldsymbol{u},\boldsymbol{y}) \quad (7)$$

식(7)이 식(5)에 비하여 단순하다는 장점이 있지만 성능의 저하가 발생한다. 본 논문에서는 4.2절에 기술한 방법으로 Log-MAP에 가까운 성능을 효율적으로 구현하고자 한다.

3.5 Log-MAP 알고리즘 적용[13,15]

Turbo 부호에 Log-MAP 디코딩 알고리즘을 적용하기 위해서 부호기를 알고 있는 상태(0 상태)로 종료 시킴으로써, 시작 상태와 종료 상태를 모두 0으로 지정할 수 있다.

Log-MAP 연산은 심볼 판정을 위한 소프트 메트릭의 연산에 Log-BCJR 알고리즘이 자주 이용되고 있다. s'와 s를 각각 (k- 1)번째와 k번째 노드의 상태라 가정하면, Log-BCJR 알고리즘의 k번째 소프트 판정 메트릭 L_k는 식(5)를 이용하여 식(8)과 같이 주어진다[12,15].

$$\begin{aligned} L_k &= \ln\left[\sum_{\substack{s',s: \\ u_k=+1}} \exp\left(a_{k-1}(s') + c_k(s',s) + b_k(s)\right)\right] \\ &\quad - \ln\left[\sum_{\substack{s',s: \\ u_k=+1}} \exp\left(a_{k-1}(s') + c_k(s',s) + b_k(s)\right)\right] \\ &= \max_{\substack{s',s: \\ u_k=+1}}{*}\left(a_{k-1}(s') + c_k(s',s) + b_k(s)\right) \\ &\quad - \max_{\substack{s',s: \\ u_k=-1}}{*}\left(a_{k-1}(s') + c_k(s',s) + b_k(s)\right) \end{aligned} \quad (8)$$

여기서 $c_k(s', s) = \ln p(y_k \mid s, s') + \ln p(s \mid s')$는 k번째 노드에서 s' 상태와 s상태를 연결하는 브랜치(branch)

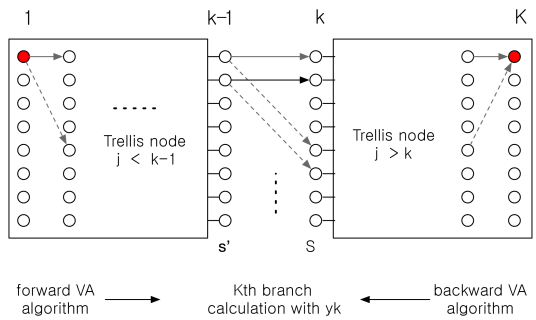


그림 4. MAP 연산을 위한 상태 메트릭 구조
Fig. 4. State metric framework for MAP algorithm

메트릭이다. 그리고 $a_{k-1}(s')$는 초기 노드부터 시작하여 VA(Viterbi Algorithm)으로 계산한 $(k-1)$번째 노드의 순방향 상태 메트릭(forward state metric)이다. 그리고 $b_k(s)$는 마지막 노드로부터 시작하여 역순으로 계산한 k번째 노드의 역방향 상태 메트릭(backward state metric)이다.

3.6 Max-Log-MAP 알고리즘의 적용

식(6)~(7)을 이용하면, Turbo 복호를 위한 Max-Log-MAP의 메트릭은 식(9)로 주어진다.

$$L_k \approx \max_{\substack{s',s: \\ u_k=+1}} [a_{k-1}(s') + c_k(s',s) + b_k(s)] - \max_{\substack{s',s: \\ u_k=-1}} [a_{k-1}(s') + c_k(s',s) + b_k(s)] \quad (9)$$

식(9)의 모든 연산은 곱셈 또는 비선형 연산을 포함하지 않는다. 따라서 식(8)의 Log-MAP 연산에 비하여 상대적으로 구현이 용이하다.

3.7 VDES 규격에 따른 Turbo 디코더의 성능 시뮬레이션

VDES 규격에 따른 Turbo 디코더의 구하기 위하여 MATLAB 스크립트를 이용하여 부동 소수점 시뮬레이션을 수행하였다. 표 1에 주어진VDES의 link ID 11,17 및 19의 변조 방식과 채널 부호 구조를 적용하였으며, 복조는 이상적인 심볼 타이밍과 위상동기를 가정하여 수행하였다. Turbo 디코더 알고리즘은 Log-MAP과 Max-Log-MAP 그리고 4.2절에서 설명한 부분적으로 LUT (Look up Table)를 이용한 Log-MAP 알고리즘을 적용하여 시뮬레이션한 BER(Bit Error Rate)과 PER(Packet Error Rate)을 그림 5와 그림 6에 각각 나타내었다. 결과에서 볼 수 있듯이 낮은 SNR에서

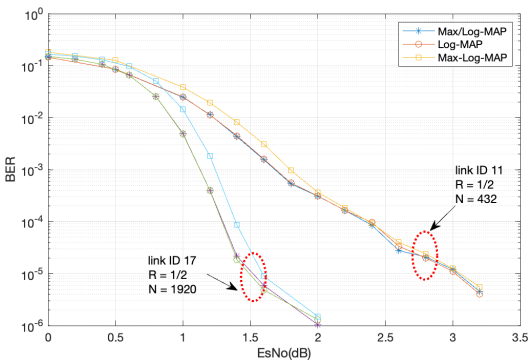


그림 5. Link ID에 대한 Turbo 복호기 BER 성능
Fig. 5. BER of Turbo decoder for different link IDs

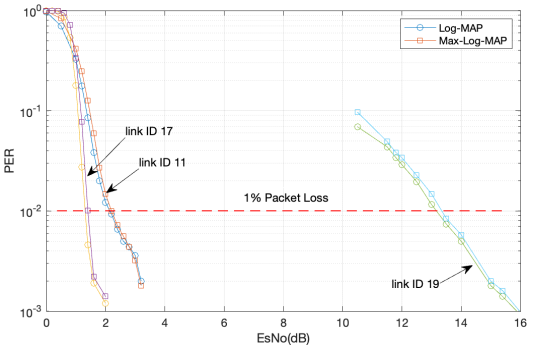


그림 6. Link ID에 대한 Turbo 복호기 PER 성능
Fig. 6. PER of Turbo decoder for different link IDs

Max-Log-MAP은 비교적 큰 성능 저하가 발생하지만, LUT를 이용한 Log-MAP (그림 5의 Max/Log-MAP)은 성능 저하가 거의 없다.

VDES의 감도는 변조 방식에 따라 ITU-R M2092-1[4]에 1% 이하의 PER을 얻을 수 있는 감도(sensitivity)의 요구조건을 정의하고 있다. 수신기의 잡음지수 (Noise Figure)를 6dB, 수신 대역폭을 VDE-TER의 최대 채널 대역폭인 100KH로 가정하면 PER = 0.01에 대한 SNR의 최대 요구치를 표 2와 같이 얻을 수 있다. 그림 6에서 볼 수 있듯이 Turbo 복호기의 PER 성능은 SNR 요구조건과 최소 3dB(ID 19의 경우)의 여유를 가지고 있음을 알 수 있다.

표 2. PER 1% 감도 요구 조건
Table 2. Sensitivity requirement for PER 1%

Link ID	11	17	19
Sensitivity	-111 dBm	-105dBm	-102dBm
EsNo (required)	7dB	13dB	16dB
EsNo (obtained)	2.1dB	1.5dB	13dB

Ⅳ. Turbo 복호기의 SIMULINK 모델링

Turbo 복호기(그림 3)의 RSC 디코더는 반복적(iterative) 연산을 수행하기 위하여, 그림 7와 같은 구조로 설계하였다. 정보길이가 K인 부호의 디펑처링 출력 시퀀스를 $\tilde{x}$, $\tilde{p}_0$, $\tilde{p}_1$, $\tilde{p}_0$, $\tilde{p}_1$이라 가정하면, Upper RSC 디코딩에서는 입력 $(\tilde{x}+x_{eL})$, $\tilde{p}_0$, $\tilde{p}_1$을 선택하여 메트릭 $x_{eU}(1:K)$를 계산한다. 단, 첫번째 반복 연산에서는 $x_{eL}=0$로 설정한다. 이어지는 Lower RSC 디코딩에서는 입력 $(\tilde{x}+x_{eU})'$, $\tilde{p}_0$, $\tilde{p}_1$ 를 선택하여 x_{eL}'을 계산한다. x_{eL}'은 디인터리빙된 후 다음 반복 연산에서 $\tilde{x}$에 대해진다. 여기서 p'는 시퀀스 p의 인터리빙 시퀀스를 표현한다.

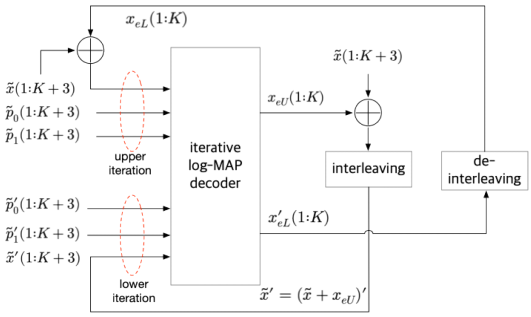


그림 7. 반복 연산 Log-MAP 디코더 구조
Fig. 7. Iterative Log-MAP decoder structure

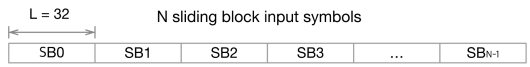
4.1 Sliding block 구조의 Log-MAP 알고리즘

Log-MAP 알고리즘은 식(12)에서 알 수 있듯이,마지막 수신 심볼로부터의 역방향 메트릭을 계산해야 한다. 또한 심볼 판정을 위해서는 모든 노드 및 모든 상태에 대한 순방향 메트릭 a_k를 저장해야 하므로, VDES와 같이 매우 긴 부호를 가지는 경우에 많은 메모리가 필요하고 및 마지막 부호 심볼을 수신해야 판정 값의 계산을 시작할 수 있다는 지연 문제를 가지고 있다. 이에 대한 해결 방법으로 Viterbi가 제안한 슬라이딩 블록(sliding block) 알고리즘[11]을 적용하여 복호기를 설계하고 구현하고자 한다. 본 논문에 적용한 슬라이딩 블록 알고리즘은 다음과 같이 요약할 수 있다.

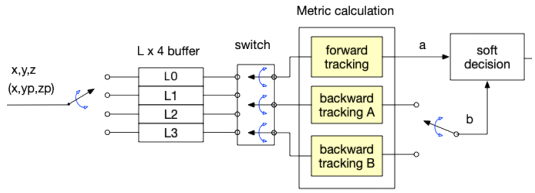
먼저, 입력 심볼은 그림 8(a)와 같이 L개의 심볼 길이를 가지는 N개의 블록으로 나눈다. 초기상태 정보가 없이 상태 메트릭을 계산하는 역방향 메트릭 연산의 경우도 RSC 부호의 구속장(constraint length) 보다 충분히 긴 습득구간 (learning period) 이후는 상태 메트릭의 신뢰도가 충분히 높아질 수 있도록 L = 32로 설정하였다.

그림 8(a)는 입력 시퀀스를 N개의 슬라이딩 블록으로 나눈 것이다. Log-MAP 복호기에는 하나의 순방향 트레킹 프로세스와 2개의 역방향 트레킹 프로세스가 동시에 각각 서로 다른 블록을 처리하면서 다음 블록으로 이동한다. 따라서 그림 8(b)와 같이 현재 처리중인 3개의 블록과 다음 슬라이딩 구간 처리를 위하여 추가되는 하나의 블록으로 구성이 된다. 각 프로세스는 4개의 메모리 블록에서 처리할 블록의 오프셋 인덱스를 변경하는 방법으로 선택하며 다음 구간의 입력 블록은 이미 처리가 완료된 블록 위치에 기록이 되도록 한다. 4개의 순환 버퍼를 이용한 이러한 과정은 그림 9에 설명이 되어 있다.

슬라이딩 버퍼 메트릭 연산기는 아래에 설명한 방식에 따라 길이 L의 블록 단위로 처리를 하여 소프트 판정



(a) 슬라이딩 블록으로 나눈 입력 시퀀스



(b)슬라이딩 블록 방식 메트릭 연산기 구조

그림 8. 상태 메트릭 연산기의 구조
Fig. 8. The structure of state metric calculator

메트릭을 출력한다[11].

순방향 트레킹 (forward tracking) 프로세스는 블록 0 (그림 8의 SB0)부터 순차적으로 순방향 메트릭 $a_{k-1}(s')$을 계산하며, 길이 L의 상태 메트릭을 버퍼에 저장한다.

역방향 메트릭 트레킹은 초기상태에 대한 정보 없이 수신 코드의 중간부터 연산을 시작하기 때문에 한 블록의 습득구간이 필요로 한다. 따라서 2개의 프로세서가 한 블록 어긋난 상태로 홀수와 짝수 블록에 대한 역방향 메트릭을 출력한다

- 역방향 트레킹 (reverse tracking) A 프로세스는 홀수 번째($2n+1$)의 블록부터 시작하여 초기 상태 정보 없이 역방향으로 상태 메트릭을 계산하며 짝수 번째($2n$) 블록을 계산할 시점에서는 충분히 높은 신뢰도를 가지는 메트릭을 출력한다.
- 역방향 트레킹 (reverse tracking) B 프로세스는 짝수 번째($2n$)의 블록부터 시작하여 역방향으로 상태 메트릭을 계산하며, 짝수 번째($2n-1$) 블록을 계산할 시점에서는 충분히 높은 신뢰도를 가지는 상태 메트릭을 출력한다.
- 마지막으로 소프트 판정(soft decision) 프로세스는 버퍼에 저장된 순방향 메트릭과 2개의 역방향 프로세스에서 교대로 출력되는 상태 메트릭을 이용하여 각 슬라이딩 블록의 마지막 심볼부터 역방향으로 디코딩을 실행하면서 L_K을 출력한다.

그림 9는 코드 길이를 6개의 블록으로 나눈 경우의 예를 들어 각 블록의 디코딩 처리 시간 T_0, T_1, ⋯, T_6에서의 순환 버퍼의 저장 값, 각 프로세스의 블록 처리 과정을 보여주고 있다. 역방향 메트릭 연산과 소프트 판정이 동시에 수행 되므로, 역방향 상태 메트릭은 마지막 값만 저장하면 된다. 본 논문에서 적용한 슬라이딩

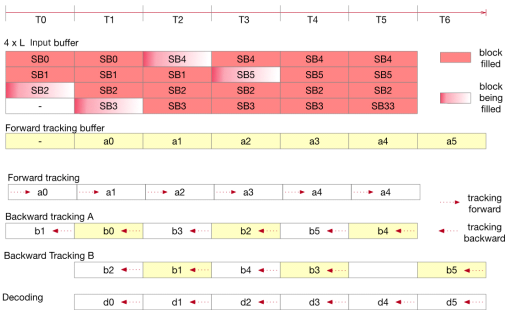


그림 9. 슬라이딩 블록 메트릭 연산 처리 과정
Fig. 9. Sliding block metric calculation process

블록 방식의 Log-MAP 알고리즘은 $L = 32$의 짧은 길이의 버퍼만으로 구성되며, 디코더 입력 $\tilde{x}$, $\tilde{p}_0$, $\tilde{p}_1$, $\tilde{p}_0$, $\tilde{p}_1$에 대한 $5 \times 4 \times L$의 슬라이딩 블록 메모리와 길이 L의 순방향 상태 메트릭 메모리가 필요하므로 W_b 비트를 사용하는 Log-MAP 디코더는 부호 길이와 무관하게 $(20L + 8L) * W_b$ 비트의 메모리를 사용한다.

4.2 Log-MAP 알고리즘의 HDL 모델링

식(8)을 이용하여 Log-MAP 알고리즘을 구현하기 위해서는 비선형 연산 $\ln(1 + e^{-(|x-y|)})$의 HDL 모델이 필요하다. 함수 $\ln(1 + e^{-(|x-y|)})$는 그림 10에 보여준 바와 같이, $|x - y|$가 증가할 때 급격하게 감소한다. 메트릭 x, y를 양수의 범위에서 동작하도록 설계할 경우, 예를 들어 $|x - y| \geq 3.35$이면 $\log(1 + e^{-(|x-y|)})/|x - y| \leq 10^{-2}$ 또는 $\log(1 + e^{-(|x-y|)})/\max(x, y) \leq 10^{-2}$가 된다. 따라서 $|x - y| < 4$인 경우는 $\max(x, y)$에 128 크기의 LUT(Look Up Table)에 저장된 $\ln(1 + e^{-(|x-y|)})$를 더해 주고, $|x - y| \geq 4$인 경우는 $\max(x, y)$만 사용하는 $\max\_sel(x, y)$ 함수를 정의하였다. 이 함수는 Max-Log-MAP에 최소한의 LUT 로직만 추가하며, 그림 5에 보여 주었듯이 Log-MAP에 가까운 연산을 수행한다.

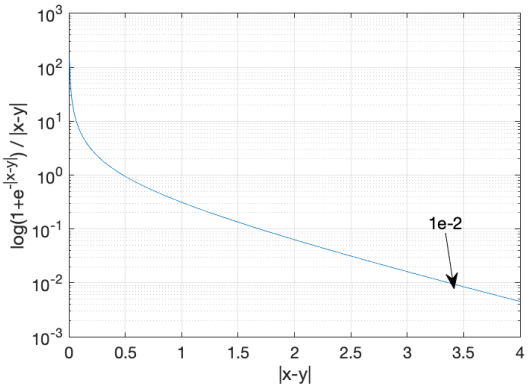


그림 10. $\ln(1 + e^{-(|x-y|)})/|x - y|$ 함수
Fig. 10. $\ln(1 + e^{-(|x-y|)})/|x - y|$ function

```
FUNCTION MAX_STAR = MAX_SEL(X,Y)
  MAX_TH = 4;     % THRESHOLD
  IF ABS(X-Y) > MAX_TH
   STAR(:)= 0;          % MAX(X,Y)
  ELSE                  % USE 128 POINT LUT
   STAR(:)= LUT( FI( ABS(X-Y)*32,0,8,0)+1);
  END
   MAX_STAR (:) = MAX(X,Y) + STAR;
END
```

4.3 인터리버 및 디인터리버 모델

VDES Turbo 부호의 인터리빙 및 디인터리빙 알고리즘은 Link ID에 따라 주어지는 파라미터와 식(2)에 의하여 정의되며, 그림 11은 HDL 변환이 가능한 SIMULINK 블록들을 이용하여 인터리빙 알고리즘을 모델링을 한 것이다. 설계된 모델은 6 클럭의 지연(latency)을 가지면서 입력 인덱스 n과 인터리빙 인덱스 $\pi(n)$을 동시에 출력한다.

인터리버로 사용할 경우는 그림 11(b)의 인터리빙 구간과 같이 $x(n)$의 위치에서 $x(\pi(n))$으로 재배치하며, 디인터리버로 사용할 경우는 그림 11(b)의 디인터리빙 구간과 같이 $x(\pi(n))$의 위치에서 $x(n)$으로 재배치를 한다.

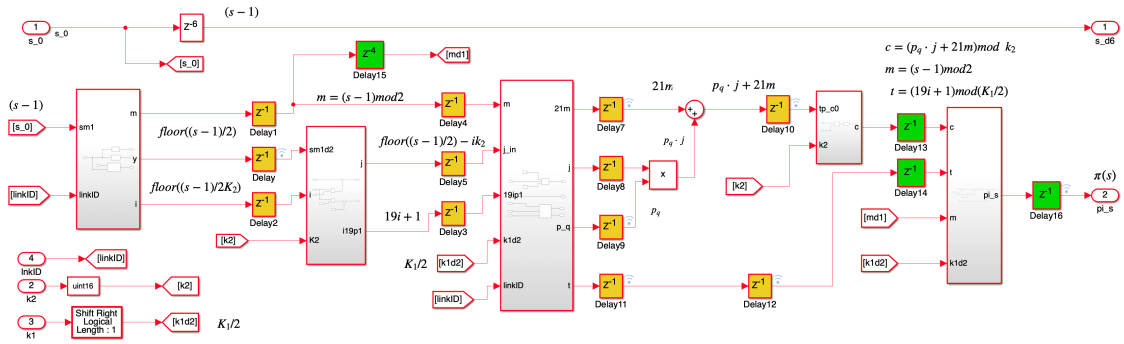
V. 모델링, HDL 컴파일, FPGA 구현 및 시험

Turbo 디코딩 알고리즘의 FPGA 구현 과정을 그림 12에 나타내었다. 먼저 MATLAB 스크립트를 이용하여 VDES 송수신기를 포함하는 VDES Turbo 복호기의 성능 분석을 위한 시뮬레이션을 수행한다. 이 단계에서 고정 소수점 기반의 입력 LLR이 생성되어 SIMULINK 모델의 입력으로 활용된다. SIMULINK 모델의 설계와 시뮬레이션 검증이 완료되면 HDL 컴파일러를 이용하여 Verilog 또는 VHDL을 생성한다. 이과정에서 설계 블록의 FPGA 자원사용량도 평가될 수 있다. 마지막으로 FIL 시뮬레이션 과 FPGA 구현 및 시험을 수행한다.

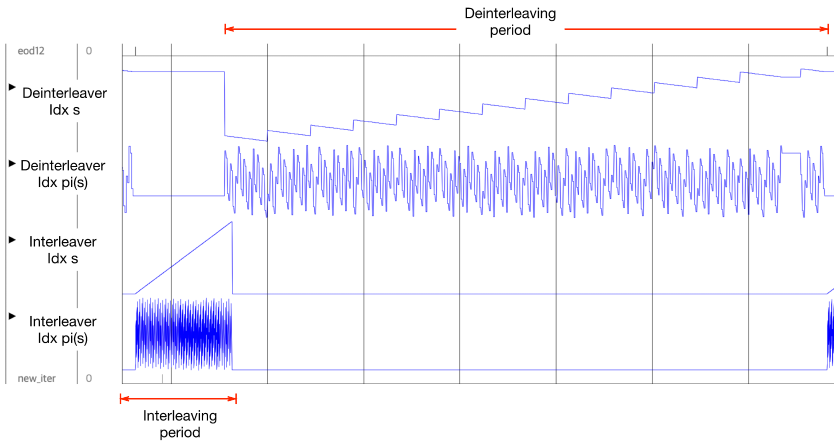
5.1 SIMULINK 모델링 및 HDL 컴파일

Turbo 복호기의 모든 알고리즘은 HDL 변환이 가능한 SIMULINK 블록과 MATLAB 스크립트로 표현할 수 있는 사용자 정의 함수 블록으로 모델링을 한 후, 시뮬레이션과 HDL 컴파일을 수행하였다. 그림 13은 HDL 컴파일러로 생성된 Turbo 복호기 모듈의 자원 사용량을 보여준 것이다.

FPGA는 Xilinx의 MPSoC ZCU104 개발 보드를 사용하였으며, 비교를 목적으로 MATLAB LTE toolbox에 포함된 LTE Turbo 복호기의 자원 사용량도 같이



(a) HDL 변환을 위한 인터리버 SIMULINK 모델



(b) 인터리빙 및 디인터리빙 인덱스 변환 파형 (id 11)

그림 11. 인터리빙 알고리즘의 SIMULINK 모델
Fig. 11. SIMULINK model of interleaving algorithm

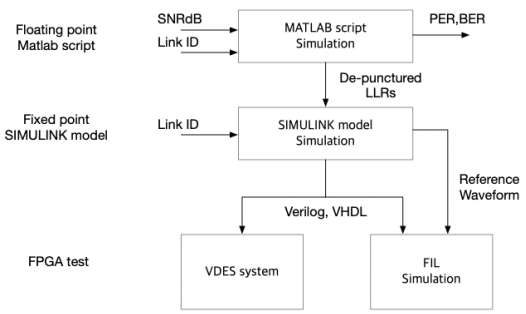
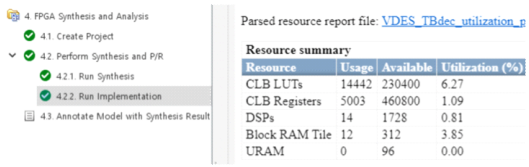
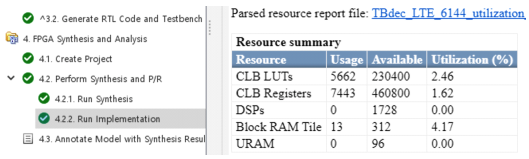


그림 12. Turbo 복호기의 설계 및 시험 과정
Fig. 12. Design and test procedure of Turbo decoder

보여주었다. 비교한 두 복호기 모두 슬라이딩 윈도우 방식을 적용하였기 때문에 메모리 (BRAM) 사용량은 거의 동일하며, VDES는 Link ID에 따라 인터리빙 인덱스를 실시간으로 계산하기 위하여 DSP 자원을 더 사용하고 있다는 것을 알 수 있다. 참고로 문헌[7]에서



(a) VDES Turbo decoder



(b) LTE Turbo decoder

그림 13. Turbo 복호기의 FPGA 자원 사용량
Fig. 13. Resource utilization of Turbo decoder

구현한 Max-Log-MAP 방식의 VDES Turbo 복호기는 82개의 BRAM을 사용하고 있다.

5.2 Turbo 복호기 블록의 FILS 시험

SIMULINK 모델을 이용한 설계의 장점은 설계 블록을 다양한 단계에서 시험 및 검증을 쉽게 할 수 있다는 점이다. 본 연구에서는 MATLAB 및 SIMULINK 시뮬레이션 과정에서 저장된 Turbo 복호기 입력과 출력 벡터를 FILS 시험 벡터로 사용하여 비교 검증을 하였다. 또한, Turbo 복호기의 출력은 CRC 디코더 블록을 이용하 최종확인을 하였다. 그림 14는 FPGA에서 동작하는 Turbo 복호기 모듈과 SIMULINK를 연동시킨 FILS 모델을 보여준 것이다.

FILS 블록의 테스트 포인트는 SIMULINK의 스코프 블록, Logic Analyzer 또는 Data Inspector 기능 등의 방법으로 결과를 확인할 수 있다. 그림 15는 Link ID 11과 19에 대한 FILS 실행 과정의 파형을 Data Inspector 기능을 이용하여 Log-MAP 디코더의 출력 x_{eU}, x_{eL}과 CRC 디코더의 결과를 관측한 것이다. Turbo 복호기는 43.008MHz로 동작하며 6회의 iteration을 수

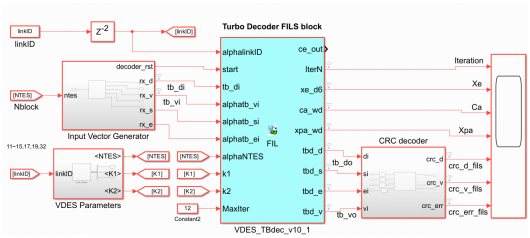
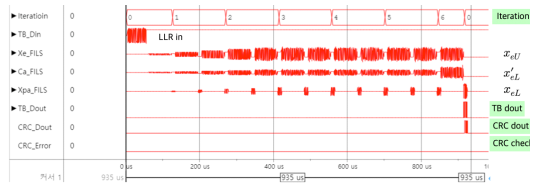
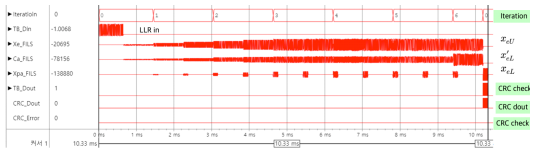


그림 14. Turbo 복호기 모듈의 FILS 모델
Fig. 14. FILS model of Turbo decoder module



(a) FILS waveforms of Link ID 11



(b) FILS waveforms of Link ID 19

그림 15. Turbo 복호기 모듈의 FILS 테스트
Fig. 15. FILS test of Turbo decoder module

행한 후, 소프트 메트릭 x_{eL}'에 대한 하드 판정 값을 CRC 디코딩하여 오류를 검출하는 과정을 확인할 수 있다. 최대 디코딩 시간은 10ms 이하이며, 1 슬롯 시간 26.667ms 이내에 복조와 복호를 모두 완료하도록 설계하였다.

5.3 VDES 테스트벤치 설계 및 Turbo 복호기의 FPGA 시험

설계된 Turbo 복호기의 VDE-TER 시스템과의 연동 시험을 위하여 Link ID 11, 17 및 19를 지원하는 테스트 벤치를 SIMULINK 모델로 설계하고 5.2절에서 시

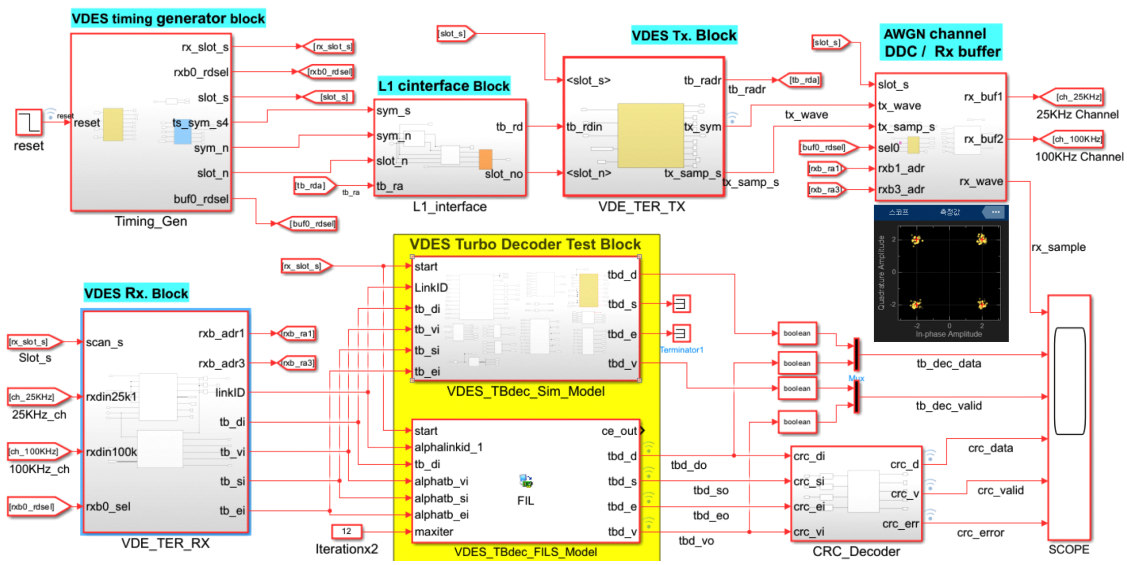


그림 16. VDE-TER 송수신 테스트 벤치
Fig. 16. Testbench for VDE-TER transceiver

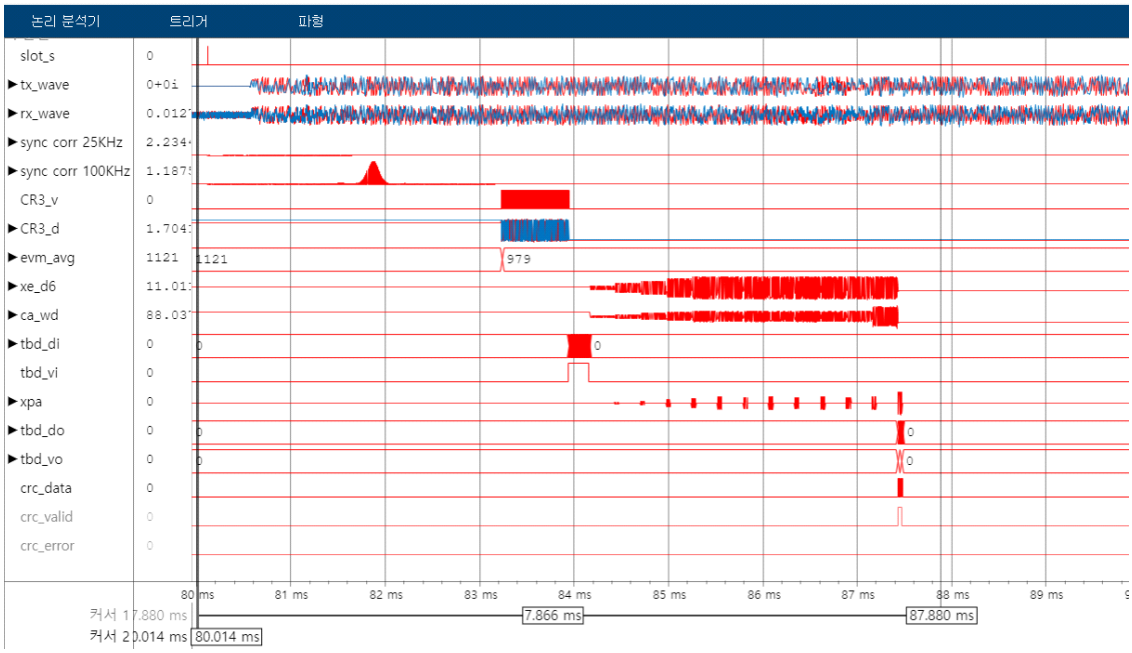


그림 17. VDE-TER 송수신 테스트 벤치의 FILS 실행 파형 (link ID 17)
Fig. 17. FILS execution waveform of testbench for VDE-TER transceiver (link ID 17)

험을 완료한 Turbo 복호기의 FILS 블록을 그림 16과 같이 연결하였다. 테스트 벤치는 VDE-TER 송신기, AWGN 채널, VDE-TER 복조기 및 Turbo 복호기 블록을 포함하고 있다. FILS 모드로 동작하는 Turbo 복호기의 파형과 테스트 벤치의 파형을 실시간으로 비교 측정하여 기능을 검증하였다. 그림 17은 FILS 시험에서 측정된 VDE-TER 모델의 link ID 17에서 송수신 신호, 동기 및 복조 신호와 FPGA의 파형을 Data Inspector를 이용하여 보여준 것이다.

Ⅵ. 시험 결과 및 결론

본 논문은 VDES 수신기를 위한 Turbo 복호기를 설계하고 FPGA에서의 동작을 시험하는 방법에 대하여 기술하였다. VDES는 34개의 Link ID에 따라 부호 길이, 부호율 및 인터리빙 방식을 다르게 정의하고 있기 때문에 매우 유연하게 동작하는 펑처링, 인터리빙 및 디코딩 알고리즘 구현 방식이 요구된다. 모델링, 시뮬레이션, HDL 컴파일, FPGA 구현 및 FIL 테스트 벤치 시험을 모두 순차적으로 SIMULINK 환경에서 진행하였다. Xilinx의 MPSoC ZCU104 개발 보드에 구현된 Turbo 복호기는 SIMULINK 모델로 작성한 테스트 벤치를 이용하여 FILS 모드로 시험을 하였으며, VDE-TER 시스템에서 동작하는 것을 확인할 수 있었다. FPGA 자원 사용량은 MATLAB toolbox의 LTE Turbo 복호기와 비교하여 유사한 크기의 메모리를 사용하고 있으나 인터리빙 알고리즘의 차이로 DSP를 더 사용하였다. 테스트 벤치로 작성한 VDE-TER 송수신기 모델은 모두 FPGA로 쉽게 변환이 될 수 있으므로 전체 VDE-TER 송수신 시스템의 동작을 FPGA에서 확인하였으나 본 논문의 범위를 벗어나므로 상세한 내용은 생략하였다. 위에 언급한 바와 같이 본 논문에서 구현한 VDES용 Turbo 복호기는 다양한 단계의 시험 및 검증이 완료됨으로써 VDES 시스템 개발에 유용하게 사용할 수 있을 것으로 본다.

References

[1] F. Lázaro, R. Raulefs, W. Wang, et al., "VHF data exchange system (VDES): An enabling technology for maritime communications," *CEAS space J.*, vol. 11, no. 1, pp. 55-63, 2019. (https://doi.org/10.1007/s12567-018-0214-8)

[2] N. Molina, F. Cabrera, V. Araña, et al., "An overview about the physical layer of the vhf data exchange system (vdes)," *Int. Conf. Comput. Aided Syst. Theory, Springer Int.*

Publishing, pp. 67-74, 2019. (https://doi.org/10.1007/978-3-030-45093-9_9)

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-Codes," in *Proc. IEEE ICC'93,* pp. 1064-1070, Geneva, Switzerland, 1993. (https://doi.org/10.1109/ICC.1993.397441)

[4] Recommendation ITU-R M.2092-1, "*Technical characteristics for a VHF data exchange system in the VHF maritime mobile band*," Feb. 2022. (https://www.e-navigation.nl/content/vdes-documentation)

[5] J.-W. Jung, J.-H. Jung, D.-G. Choi, and I.-K. Lee, "An FPGA design of high-speed turbo decoder," *J. KICS,* vol. 30, no. 6c, pp. 450-456, Jun. 2005.

[6] M. H. Kim, J.-W. Jung, and I. K. Lee, "An FPGA implementation of high-speed adaptive turbo decoder," *J. KICS*, vol. 32, no. 4, pp. 379-388, Apr. 2007.

[7] W. Yang, F. Kong, and J. Qian, "An FPGA implementation of turbo decoding in VDES," *2023 IEEE 5th ICCASIT*, pp. 1375-1383, Dali, China, 2023. (https://doi.org/10.1109/ICCASIT58768.2023.10351645)

[8] Y.-H. Seo, "FPGA Design of Turbo Code based on MAP," *J. KICS*, vol. 32, no. 3, pp. 306-313, Mar. 2007.

[9] IALA Guideline G1139, "*The Technical Specification of VDES*," 3 Ed., Jun. 2019. (https://www.e-navigation.nl/sites/default/files/1139-ed.3-the-technical-specification-of-vdes_june-2019.pdf)

[10] L. H. Ang and S. G. Lim, "SOVA based LTE turbo decoders," M.S. Thesis, Dept. of Electr. and Inf. Technol., Lund Univ, Sweden, Sep. 2009.

[11] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas in Commun.*, vol. 16, no. 2, pp. 260-264, Feb. 1998. (https://doi.org/10.1109/49.661114)

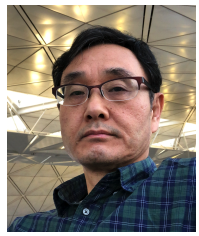
[12] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. ICC'95*, pp. 1009-1013, Seattle, WA, 1995. (https://doi.org/10.1109/ICC.1995.524253)

[13] S. S. Pietrobon, "Implementation and performance of a serial MAP decoder for use in an iterative turbo decoder," in *Proc. IEEE Int. Symp. Inf. Theory*, p. 471, Whistler, B.C., Canada, 1995. (https://doi.org/10.1109/ISIT.1995.550458)

[14] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-output decoding algorithms in iterative decoding of turbo codes," *JPL TDA Progr. Rep.*, vol. 42-124, pp. 63-87, 1995. (https://ntrs.nasa.gov/citations/19960022223)

[15] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Tran. Inf. Theory*, vol. 42, pp. 429-445, 1996. (https://doi.org/10.1109/18.485714)

김 재 형 (Jae Hyung Kim)



1983년 2월 : 고려대학교 전자공학과 졸업

1985년 2월 : 고려대학교 전자공학과 석사

1989년 8월 : 고려대학교 전자공학과 박사

1991년~현재 : 창원대학교 전기전자제어공학부 교수

<관심분야> 전자공학, 통신공학, FPGA설계

[ORCID:0000-0001-9327-1173]