https://doi.org/10.7840/kics.2022.47.12.2179

Simulink model을 이용한 NB-IoT 하향링크 송신기 FPGA 설계

김 재 형◆

FPGA Design of an NB-IoT Downlink Transmitter Using a Simulink Model

Jae Hyung Kim◆

요 약

본 논문에서는 NB-IoT 하향링크의 기지국 송신기 물리 계층을 FPGA에 Simulink model을 기반으로 설계 및 구현을 하였다. 적용된 규격은 3GPP TS36 V.15에서 standalone 및 FDD 모드를 기준으로 구현하였다. 송신기는 L1 인터페이스 레지스터에 의하여 전달되는 채널 파라미터를 이용하여 NPBCH, NPSS, NSSS, NPDCCH 및 NPDSCH의 subframe grid와 시간 영역의 파형을 발생시킬 수 있도록 하였다. 기지국 송신기의 물리 계층의 기능들은 HDL 변환이 가능한 Simulink 블록과 MATLAB 스크립트를 이용하여 설계하였다. MATLAB Simulink를 이용하여 설계한 모델은 논리 시뮬레이션, Verilog HDL 생성 및 FILS (FPGA In the Loop) 검증 과정을 모두 동일한 Simulink 환경에서 실행을 한 후, Xilinx Zynq Ultrascale 개발보드에서 FPGA 구현을 하고 설계 검증을 하였다. 구현된 하향링크 송신기는 Amarisoft의 NB-IoT UE emulator를 이용하여 수신 및 복조를 하여 송신 신호의 오류 여부를 최종 확인하였다.

키워드 : 협대역 사물인터넷, 모델기반 설계, FIL 시뮬레이션 , FPGA, Testbed
Key Words : NB-IoT, Model based design, FIL simulation, FPGA, Testbed

ABSTRACT

In this paper, the base station transmitter physical layer of the NB-IoT downlink was designed and implemented based on the Simulink model on the FPGA. The physical layer specification adopted is 3GPP TS36 V.15 and is designed with standalone and FDD modes assumed. The transmitter could generate subframe grids and time-domain waveforms of NPBCH, NPSS, NSSS, NPDCCH, and NPDSCH using channel parameters written in L1 interface registers by the L1 controller. In this paper, all the functions of the base station transmitter are designed by MATLAB Simulink models supporting HDL generation. The model developed using Simulink performed logic simulation, Verilog HDL generation, and FPGA In the Loop (FIL) simulation in the same Simulink environment. Then the FPGA was implemented on the Xilinx Zynq Ultrascale Development Board, and the design was validated. The implemented downlink transmitter was finally verified by receiving using Amarisoft's NB-IoT UE emulator.

※ 이 논문은 2021~2022년도 창원대학교 자율연구과제 연구비 지원으로 수행된 연구결과임
◆ First Author : Changwon University School of Mechatronics, hyung@changwon.ac.kr, 종신회원
논문번호 : 202208-189-D-RU, Received August 23, 2022; Revised September 21, 2022; Accepted September 26, 2022

Ⅰ. 서 론

근래에, Internet of Things (IoT)[1]는 우리의 생활의 많은 부분에서 그 활용도가 급격하게 높아지고 있다[2]. Narrow-band IoT (NB-IoT)는 3GPP에서 협대역의 IoT 응용을 위하여 개발한 무선기술 표준이며 미래 5G 이동통신의 중요한 부분으로 포함된다 [2,3]. NB-IoT는 많은 장치를 연결할 수 있으며, 특히 작은 대역폭과 매우 긴 배터리 수명을 요구하는 응용에 최적화되어 있다. 이를 위하여 하향 링크는 180KHz의 대역폭을 사용하지만, 상향 링크에서는 3.75KHz와 15KHz의 single-tone 모드와 15KHz의 배수의 대역폭을 가지는 multi-tone 모드를 가진다. 또한 동작 주파수 대역은 응용 시나리오(standalone, guard-band 또는 in-band)에 따라 선택될 수 있다[2,3].

NB-IoT의 큰 시장성과 많은 응용 분야에 대한 기대가 커지면서 저전력 저가의 단말기용 트랜시버 개발에 대한 요구가 매우 높아졌으며, 다수의 연구가 진행되었다[4-8]. NB-IoT의 매우 빠른 확장 추세는 설계 및 설계 검증방법에서의 변화를 역시 요구한다. 즉, 빠르고 쉬운 프로토타입 설계와 검증방법이 NB-IoT의 시장 진입을 위한 주요 요건 중의 하나라고 볼 수 있다.

본 논문에서는 NB-IoT 기지국 하향 링크 송신기의 물리 계층을 모델 기반의 설계를 이용하여 FPGA에 구현을 하고자 한다. 송신기 블록은 MATLAB Simulink를 기반으로 모델링 및 시뮬레이션을 수행하고, Simulink HDL (Hardware Description Language) 컴파일러를 이용하여 HDL을 생성한다[9]. 설계된 블록의 HDL은 FPGA에 구현되어 실행하기 전에 HDL 시뮬레이션에 의한 검증이 가능하지만 HDL testbench를 작성하기 위해서는 많은 노력과 시간이 필요하다. 본 연구에서는 MATLAB의 FPGA in the Loop Simulation (FILS)[10]을 이용하여 FPGA에서의 송신기 동작을 보다 빠르고 편리하게 검증을 하는 방법을 사용한다. FILS에서는 Simulink가 프로세서의 제어기능을 대신하면서 결과를 확인 할 수 있다. 즉, FILS은 DUT(device under test) 블록의 FPGA에서의 동작을 SIMULINK 환경의 testbench에서 검증을 할 수 있다. 최종적으로 구현된 송신 시스템에서는 Xilinx Zynq SoC의 프로세서에서 물리 계층 블록의 제어를 수행하게 되며, 구현된 송신 블록의 출력은 MATLAB 5G toolbox[11] 스크립트를 이용하여 작성된 Link Level Simulator(LLS)의 waveform과 비교하여 검증한 후 NB-IoT UE emulator에서 수신 시험을 한다[12].

본 논문에서는 3GPP TS36 V15 표준[9,10]을 기반으로 NB-IoT 기지국 송신기의 물리 계층을 설계하고자 하며, 주요 설계 규격을 표 1에 요약하였다.

본 연구에서는 stand alone 모드, FDD 방식, 2개의 송신 안테나 다이버시티 만을 고려 하였다. 물리 채널은 NPBCH, NPSS, NSSS, NPDCCH 그리고 NPDSCH을 생성하고 전송한다. 송신기는 크게 채널 부호화 (Channel encoding), 채널 변조 (Channel modulation) , 동기 신호 발생 및 IFFT 기능 블록들로 구성된다. 채널 부호화 블록에서는 모든 채널의 CRC 코딩, TBCC(Tail Byte Convolutional Coding) 및 rate matching을 수행한다. 채널 변조 블록은 NPBCH, NPDSCH 및 NPDCCH 채널을 위한 독립적인 모듈로 구성이 된다. 각 모듈들은 채널 부호화 블록에서 전달받은 코드 워드를 저장하고 3GPP TS36 표준에 의거하여 scrambling과 QPSK 변조를 하고, layer mapping과 precoding을 하여 2 개의 안테나 송신 diversity 신호를 생성한다. 그리고 자원 할당 및 반복 전송 제어를 수행한다. 동기 신호 발생 블록에서는 초기 동기를 위한 NPSS와 NSSS 신호를 생성한다. Cell ID에 의해서 생성되는 NRS(Narrowband Reference Signal)는 물리 채널의 복조를 위하여 NPSS, NSSS를 제외한 모든 subframe의 특정 RE (resource element)에서 전송이 된다. 채널 신호, 동기 신호들은 모든 subframe에서 독립적으로 항상 생성되지만 subframe의 채널 type에 따라 선택되어 subframe grid buffer에 저장한다. 마지막으로 subframe grid buffer는 IFFT 블록에서 OFDM 기저대역 신호로 변조하여 출력한다. 그림 1은 기지국 송

표 1. NB-IoT 기지국 송신기 주요 설계 규격
Table 1. NB-IoT downlink design specifications.

Standard	3GPP TS36 v15
Mode	stand alone 모드
Duplex	FDD (Frequency Division Duplex)
Diversity	2 Tx. antenna diversity
Physical Channel	NPBCH(NB Physical Broadcasting CH)
	NPDCCH(NB Physical Downlink Control CH)
	NPDSCH (NB Physical Downlink Shared CH)
	NPSS(NB Primary Synchronization Signal)
	NSSS(NB Secondary Synchronization Signal)

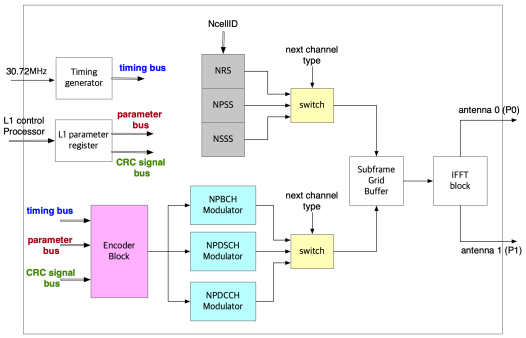


그림 1. NB-IoT 하향링크 송신기 블록도
Fig. 1. NB-IoT downlink transmitter block diagram

신기 물리 계층의 블록도를 보여준 것이다.

본 논문의 구성은 2장에서 Simulink를 이용한 모델 기반 송신기 블록의 구조 및 알고리즘 설계에 대하여 기술하고, 3장에서는 FILS를 이용한 기능 검증 과정 및 FPGA 구현 결과에 대하여 기술하였다. 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 모델 기반 기지국 송신기 설계

2.1 채널 부호화 (Channel encoding)블록

Encoding 블록은 L1 제어 프로세서에 의하여 주기적으로 갱신되는 L1 레지스터에서 MIB, DCI 및 DLSCH[7] 데이터 및 채널 파라미터들을 읽은 후 채널 부호화 과정을 실행한다. 그림 2는 채널 부호화 블록의 구조를 나타낸 것이다

CRC encoder와 TBCC는 HDL code의 생성이 가능한 Simulink 블록을 사용하였으며, rate matching 블록은 MATLAB 스크립트를 이용하여 설계하였다.

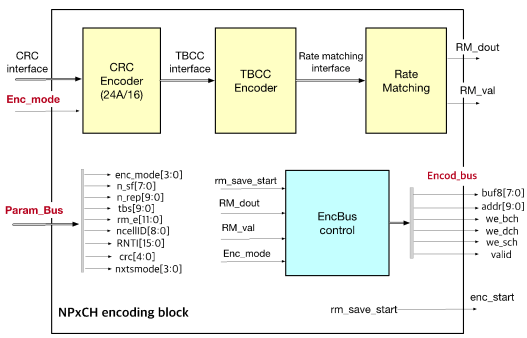


그림 2. 채널 부호화 블록의 구조
Fig. 2. The structure of channel encoding block

2.1.1 CRC 부호 모듈

CRC 부호기는 Simulink HDL 생성 블록을 이용하여 설계하였으며, NB-IoT 데이터 형에 따라 다음과 같이 CRC 다항식과 mask가 적용된다. [13]

표 2. 데이터 형에 따른 CRC 다항식과 mask
Table 2. CRC polynomial and mask.

데이터형	CRC 다항식	CRC masking
MIB	$g_{CRC16}(D)$	안테나 수에 따른 mask
NPDSCH	$g_{CRC24A}(D)$	적용하지 않음
DCI	$g_{CRC16}(D)$	RNTI

2.1.2 TBCC 부호 모듈

MIB, DLSCH, DCI의 channel 부호기는 부호율 1/3의 Tail Bite Convolutional Code (TBCC)가 사용된다. 그림 3은 TBCC 부호기의 구조이며 생성 다항식은 $G_0 = 133\ (octal)$, $G_1 = 171\ (octal)$, $G_2 = 165(octal)$로 주어진다[13]. TBCC 부호기는 HDL 생성이 가능한 Simulink 블록을 이용한다.

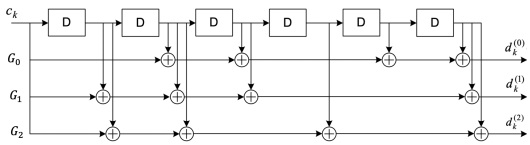


그림 3. TBCC 부호기의 구조
Fig. 3. The structure of TBCC encoder

2.1.3 Rate matching 모듈

Rate matching 모듈에서는 TBCC 부호 모듈에서 출력되는 비트 스트림 $d_k^{(0)}, d_k^{(1)}, d_k^{(2)}$에 대하여 sub-block interleaving과 비트 수집 (bit collection)을 하여 순환 버퍼 (circular buffer)를 형성한다. 채널 파라미터에 따라 subframe에 할당된 자원의 수에 의하여 결정되는 길이의 코드를 출력하기 위하여, 순환 버퍼로부터 비트 선택 (bit selection) 또는 자르기 (pruning)을 실행한다. 그림 4는 rate matching 모듈의 구조이며, MATLAB 스크립트를 이용하여 알고리즘을 작성하고 HDL coder를 이용하여 HDL IP로 변환한다.

TBCC 부호 모듈의 출력 스트림을 3개의 가상 circular buffer에 저장한 후, 그림 4와 같이 column permutation, bit selection 및 pruning을 처리하는 알고리즘[13]을 rate matching 모듈에 구현하였다.

Rate matching 알고리즘의 pseudo code는 다음과

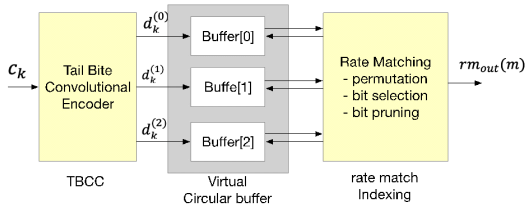


그림 4. Rate matching 모듈의 구조
Fig. 4. The structure of rate matching module

같다. 단, E는 출력 코드의 길이, Rcc 는 permutation matrix의 row, ND는 permutation matrix를 채울 때 prepending할 null의 갯수이다. Ptable은 column permutation table을 의미한다. Null의 위치는 항상 첫 번째 행의 1~ND 열에 존재하므로, null 위치에서는 바로 다음 행으로 건너뛰도록 한다.

Ptable = [1 17 9 25 5 21 13 29 3 19 11 27 7 23 15 31 0 16 8 24 4 20 12 28 2 18 10 26 6 22 14 30];
buf_adr = 0
idxE = **idxC** = **idxR** = **idxB** = 0;

while (idxE < **E**) {
  Cnum = Ptable(idxC);
if (idxR == 0 && Cnum < ND)
  idxR= idxR+1
end
addr_r = Cnum + idxR * 32
rm_out[idxE] = Buffer[idxB][addr_r]
idxR = idxR+1
if (idxR == Rcc)
    idxR = 0;
    idxC = idxC+1;
      if idxC > 31
        idxC = 0;
        idxBuf = idxBuf + 1;
        if idxBuf == 3
          idxBuf = 0;
        end
      end
    end
}

HDL 생성을 위한 채널 부호화 블록의 Simulink 모델은 그림 5와 같다.

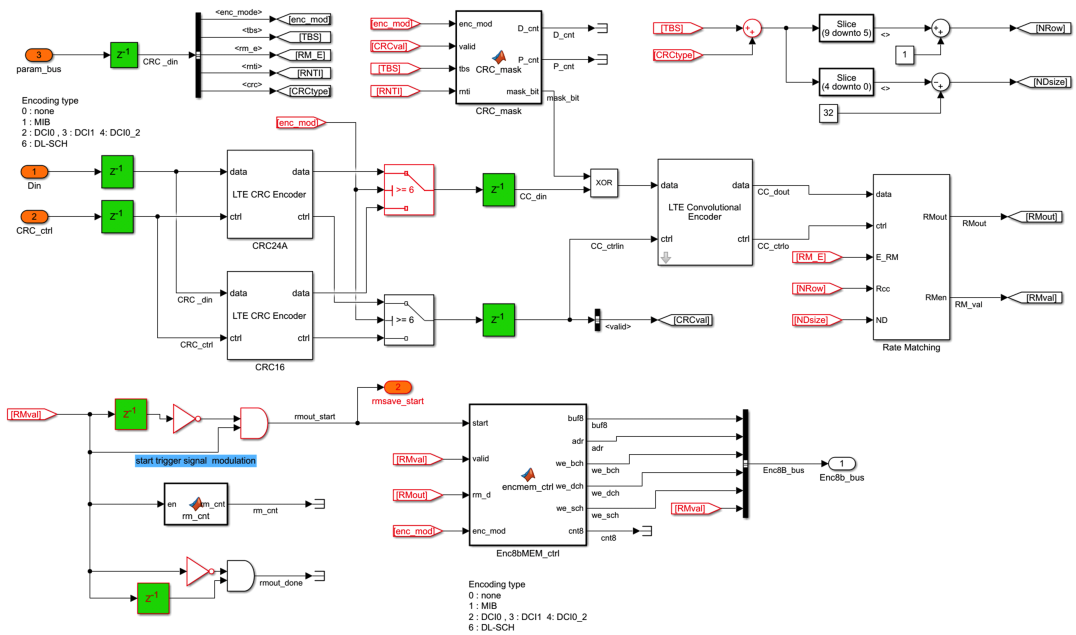


그림 5. HDL 생성을 위한 채널 부호화 블록의 Simulink 모델
Fig. 5. Channel coding block Simulink Model for HDL generation

2.2 채널 변조(Channel Modulation) 블록 설계

채널 변조 블록에서는 NPBCH, NPDSCH 그리고 NPDCCH 채널 변조 서브블록이 병렬로 설계되어 독립적으로 실행이 된다. 또한, NPDSCH 변조 서브블록은 SIB1(System Information Type 1), SIBs 및 DLSCH[14,15] 정보 채널을 subframe 단위로 혼재하여 전송할 수 있도록 3개의 독립적인 모듈로 구성이 된다. 채널 부호화 블록이 채널 코딩을 완료하면서 변조 서브블록을 정보 타입에 따라 선택적으로 활성화 시키고, 활성화된 서브블록은 데이터의 전송이 완료될 때까지 유한상태머신(Finite State Machine : FSM)으로 동작한다. 즉, 채널 부호화 블록은 각 물리 채널들의 FSM에 코드 데이터와 채널 파라미터들을 등록함으로써 해당 채널 변조 서브블록을 활성화 시킨다. 그림 6은 채널 변조 블록의 구조를 나타낸 것이다.

NPBCH, NPDSCH 및 NPDCCH 채널의 변조 과정은 공통적으로 Scrambling, QPSK 변조, Layer 할당 및 precoding 그리고 자원 할당 과정을 통하여 subframe grid를 생성한다. 자원 할당 및 반복전송을 위한 제어 모듈이 각 채널 변조 서브블록에 포함된다.

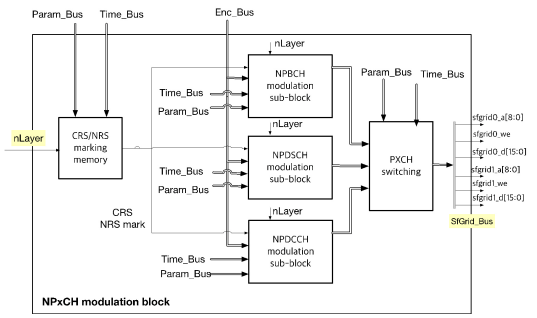


그림 6. Channel modulation block의 구조
Fig. 6. Channel modulation block diagram

2.2.1 NB-IoT 하향링크 채널의 Scrambling [10]

NPBCH는 64 frame 마다 Cell ID (N_{ID}^{Ncell})를 이용하여 scrambler를 초기화 시킨다. NPDCCH는 항상 1개의 subframe으로 구성이 되며, 반복 전송할 경우 전송 시작 및 4회 반복시점 마다 식(1)로 주어지는 c_{init} 값으로 초기화 시킨다. 여기서, n_s는 초기화 시점의 frame내의 slot 번호를 의미한다.

$$c_{init} = \lfloor n_s/2 \rfloor 2^9 + N_{ID}^{Ncell} \quad (1)$$

NPDSCH는 BCCH(Broad Control Channel)를 포함하는 SIB1과 SIBs의 경우, 식(2)를 이용하여 scrambler를 초기화하며, BCCH를 포함하지 않는 경우는 식(3)에 의하여 초기값이 주어진다. 여기서, n_f는 frame 번호, n_{RNTI}는 RNTI를 의미한다.

$$c_{init} = n_{RNTI} \cdot 2^{15} + (N_{ID}^{Ncell} + 1)((n_f \ mod 6) + 1) \quad (2)$$

$$c_{init} = n_{RNTI} \cdot 2^{14} + n_f \ mod 2 \cdot 2^{13} + \left\lfloor \frac{n_s}{2} \right\rfloor \cdot 2^9 + N_{ID}^{Ncell} \quad (3)$$

NPBCH는 채널 부호화된 1600 bit의 코드 워드에 대하여 scrambling을 실행한 후 메모리에 저장하고 반복 전송을 한다. NPDSCH와 NPDCCH는 코드 워드를 N_{rep}^{min}회 반복을 완료하면 다시 scrambler를 초기화 시킨다. 여기서 N_{rep}^{min}은 동일 subframe 반복 횟수이며 식(4)로 주어진다. 따라서, 초기화 시점에서 N_{sf}의 subframe 길이에 해당되는 Gold code를 저장하고 코드 워드가 N_{rep}^{min}회 전송될 때까지 반복적으로 사용한다.

$$N_{rep}^{min} = \begin{cases} 1 & NPDCCH \\ 1 & NPDSCH\ (\ with\ BCCH\ ) \\ \min(N_{rep}, 4) & NPDSCH\ (\ not\ BCCH) \end{cases} \quad (4)$$

2.2.2 QPSK 변조

모든 채널은 scrambling된 데이터의 연속 2비트 $\{b(i), b(i+1)\}$를 표 3에 따라 QPSK 심볼로 매핑한다.

표 3. QPSK symbol 매핑
Table 3. QPSK symbol mapping

$b(i), b(i+1)$	00	01	10	11
QPSK	$\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}} + \frac{-1}{\sqrt{2}}$	$\frac{-1}{\sqrt{2}} + \frac{1}{\sqrt{2}}$	$\frac{-1}{\sqrt{2}} + \frac{-1}{\sqrt{2}}$

2.2.3 Layer mapping과 Precoding

본 논문에선 두개의 안테나를 가정하여 설계를 하였으며, 송신 다이버시티(Transmit Diversity)를 위한 layer mapping과 precoding은 식(5)와 식(6)에 의하여 각각 정의된다[15].

여기서, $d(i)$, $x^{(p)}(i)$, $y^{(p)}(i)$는 각각 QPSK 변조, layer mapping 그리고 precoding 출력 시퀀스를

표 현하며, $p \in \{0,1\}$, $i = 0,1,\ldots, M_{symb}$이다. 단, M_{symb}은 QPSK 심볼 시퀀스의 길이이다.

$$x^{(0)}(i) = d(2i)\ ,\ x^{(1)}(i) = d(2i+1) \quad i = 0,1,\ldots,M_{symb}/2-1 \tag{5}$$

$$y(i) = \begin{bmatrix} y^{(0)}(i) & y^{(1)}(i) \end{bmatrix}^T\ , \quad i = 0,1,\ldots, M_{symb}-1$$

$$\begin{bmatrix} y^{(0)}(2i) \\ y^{(1)}(i) \\ y^{(0)}(2i+1) \\ y^{(1)}(2i+1) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & j & 0 \\ 0 & -1 & 0 & j \\ 0 & 1 & 0 & j \\ 1 & 0 & -j & 0 \end{bmatrix} \begin{bmatrix} Re(x^{(0)}(i)) \\ Re(x^{(1)}(i)) \\ Im(x^{(0)}(i)) \\ Im(x^{(1)}(i)) \end{bmatrix} \tag{6}$$

2.2.4 자원 할당(RE mapping) 및 반복 전송

NPBCH는 800 QPSK 심볼로 변조되어, 100 QPSK 심볼 씩 8개의 subframe 0에 mapping이 된다. 각 subframe은 8회 반복이 되며 총 64 frame에 걸쳐서 전송이 된다[14,15].

NPDSCH는 NRS (Narrowband Reference Signal)를 제외한 자원(RE)에 할당이 된다. 따라서, 16개의 NRS 위치를 제외한 152개의 RE에 심볼이 할당된다. 그림 8은 BCCH를 포함하지 않는 NPDSCH의 반복 전송 알고리즘을 나타낸 것이다. BCCH를 포함하지 않는 경우는 한개의 subframe을 식(4)로 주어지는 N_{rep}^{min}회 반복한 후 다음 subframe을 전송한다 (그림 8).

그러나 BCCH (SIB1, SIBs)를 포함하는 경우는 코드 워드 단위로 반복 전송을 한다. 반복 전송 시점에

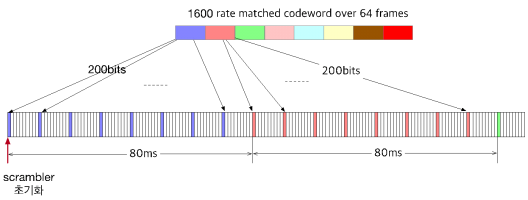


그림 7. NPBCH의 반복 전송
Fig. 7. NPBCH repetition

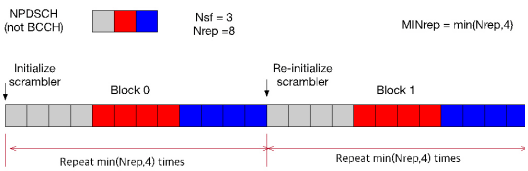
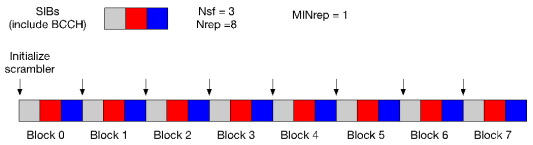
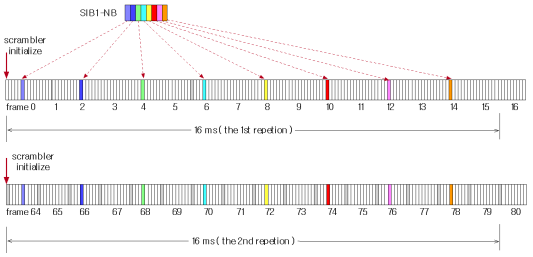


그림 8. BCCH를 포함하지 않는 NPDSCH
Fig. 8. NPDSCH not including BCCH



(a) SIBs 반복 전송
(a) SIBs repetition



(b) SIB1을 포함하는 NPDSCH
(b) SIB1 repetition including BCCH

그림 9. BCCH를 포함하는 NPDSCH
Fig. 9. NPDSCH including BCCH

서 scrambler는 다시 초기화를 시킨다. SIB1는 8개의 subframe으로 생성이 되며, 64 frame 마다 처음 8개의 짝수 frame의 subframe 4번에 할당이 된다. 따라서 최대 $N_{rep} = 16$의 반복을 할 경우, 256 frame 구간에 전송이 완료된다[14,15].

2.2.5 변조 제어 모듈

기지국 하향링크의 물리 채널들은 부호화된 후 독립적인 제어 모듈에서 변조 및 자원 할당 제어가 수행된다. 따라서 NPBCH, NPDCCH 서브블록에 각각 1개 그리고 NPDSCH 서브블록에 3개의 변조 및 제어 모듈이 설계되었다. 채널 코딩이 완료된 코드 워드는 버퍼에 저장된 후, 반복 전송을 위한 제어 모듈의 FSM의 의하여 subframe 단위로 변조와 자원 할당을 수행하며 그림 10은 반복 전송 제어 모듈의 기본 구조이며, 모든 채널 전송 제어에 반복적으로 사용할 수 있다. 그림 11은 3개의 제어 모듈이 사용된 NPDSCH

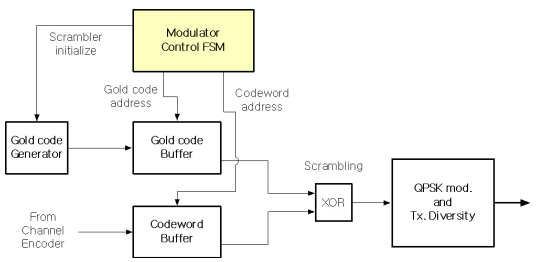


그림 10. 반복 제어 모듈의 구조
Fig. 10. The structure of repeat control module

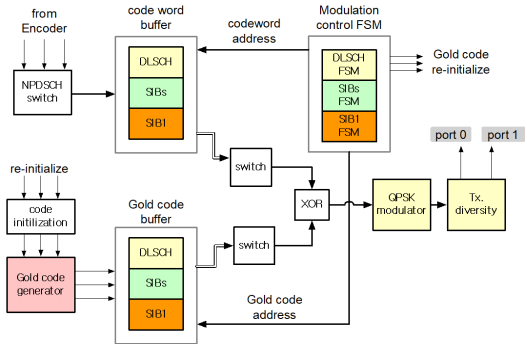


그림 11. NPDSCH 변조 서브 블록
Fig. 11. NPDSCH modulation sub-block

변조 서브 블록의 구조 보여준 것이다

반복 전송 제어모듈은 부호화된 코드 워드를 N_{rep}/N_{rep}^{min}의 전송 블록 (transmission block)으로 나누고 새로운 블록의 시작 시점에서 scrambler를 초기화하여 반복 횟수 (N_{rep})의 전송을 마칠 때 까지 표 4에 정의된 상태 변수를 관리한다

그림 12는 $N_{sf} = 2$, $N_{rep} = 8$인 NPDSCH의 반복 전송을 수행하는 FSM의 상태 변화를 보여준 것이다. 채널 부호화가 완료되면 FSM_start 신호가 발생되고 자원 할당을 위한 채널 파라미터 (block 수 = 2, $N_{rep}^{min} = 4$)가 저장되면서 FSM을 활성화시킨다. 활성화된 이후는 제어 모듈은 subframe 단위로 채널 타입이 일치할 경우에만 변조를 수행하고 FSM의 상태를 변경한다. 모든 반복전송이 완료되면 자동으로 제어 모듈은 비 활성화가 되고 초기상태로 돌아간다.

표 4. 반복 제어 모듈의 상태 변수
Table 4. State variables of repeat control module

state	descriptions
cntBlk	block count : 0 ~ N_{rep}/N_{rep}^{min} - 1
cntSf	subframe position : 0 ~ N_{sf} - 1
cntRep	subframe repeat count : 0 ~ N_{rep}^{min} -1
mod_rdy	modulator module ready

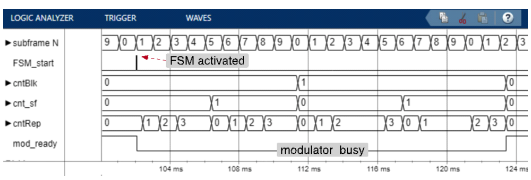


그림 12. 반복 제어 모듈의 상태 변수 파형의 예
Fig. 12. State variables of repeat control module

2.2.6 NRS generation block

NB-IoT는 하향 링크 subframe의 복조를 위하여 NRS (Narrowband Reference Signal)를 NPBCH, NPDSCH 및 NPDCCH subframe의 자원에 포함하여 전송한다[9]. NRS는 Cell ID(N_{ID}^{Ncell}와 subframe에서의 slot 위치(n_s)에 따라 다음과 같이 생성된다. 단, l은 symbol index를 의미한다.

2.2.6.1 Reference signal sequence 생성

Reference signal은 $l = 5,6,12,13$ 심볼의 시작에서 식(7)로 초기화되는 Gold code $c(m)$ 에 의하여 식(8)과 같이 복소 심볼로 생성된다[14].

$$c_{init} = 2^{10}(\ 7\ (\ n_s' + 1) + l + 1)(2 \cdot N_{ID}^{Ncell} + 1) + 2 \cdot N_{ID}^{Ncell} + N_{CP} \tag{7}$$

$$r_{l,n_s}(m) = \frac{1}{\sqrt{2}}\big(1 - 2 \cdot c(2m)\big) + j\frac{1}{\sqrt{2}}\big(1 - 2 \cdot c(2m+1)\big) \quad m = 0\ ,1 \tag{8}$$

2.2.6.2 NRS의 RE mapping

$r_{l,n_s}(m)$는 안테나 포트 p, subcarrier index k, symbol index l에 의하여 $a_{k,l}^{(p)}$에 할당을 한다[14].

$$\begin{aligned} a_{k,l}^{(p)} &= r_{l,n_s}(m') \\ m' &= m + N_{RB}^{max,DL} - 1 \ :\ m = 0\ ,\ 1 \\ k &= 6m + \big(v + v_{shift}\big) mod 6 \\ l &= 5,\ 6 \\ v_{shift} &= N_{ID}^{Ncell}\ mode\ 6 \\ v &= 0\ (l = 5)\ ,\ 3\ (l = 6\ )\ :\ p = 0 \\ v &= 3\ (l = 5)\ ,\ 0\ (l = 6\ )\ :\ p = 1 \end{aligned} \tag{9}$$

그림 13은 NRS 심볼 생성 블록의 구조를 나타낸 것이다. Gold code 시퀀스, $c(m)$ 은 subframe 시작 시

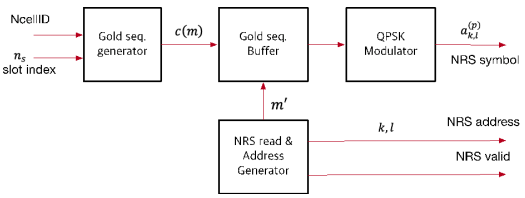


그림 13. NRS 심볼 생성 블록의 구조
Fig. 13. NRS symbol generation block diagram

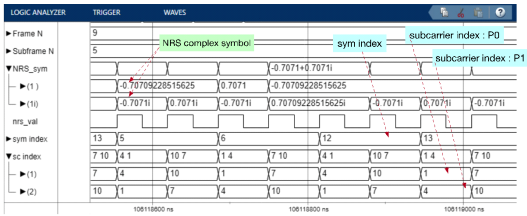


그림 14. NRS와 index 출력 파형
Fig. 14. NRS and index output waveforms

점에서 버퍼에 저장되며, 식(8)과 식(9)를 이용하여 심볼을 NRS 심볼들을 생성한다.

NRS 심볼은 subcarrier index k와 symbol index l와 같이 출력이 되어 subframe grid에 기록이 될 수 있도록 한다. 그림 14는 NRS 출력 파형을 보여준 것이다. 이 신호에 의하여 NPSS와 NSSS를제외한 모든 subframe grid에 NRS를 삽입한다.

그림 15는 채널 변조 블록의 Simulink HDL 생성 모델이며, NPBCH, NPDSCH 및 NPDCCH 변조 모듈과 subframe 채널 타입에 따라 선택하여 subframe grid buffer에 저장하는 PXCH switching 블록으로 구성이 된다. NPxCH 채널 변조 모듈에는 그림 10의 전송 제어 모듈이 포함된다.

2.3 동기신호 발생 블록 설계

기지국 하향 링크는 NPSS와 NSSS를 동기 신호로 송신한다. NPSS는 모든 frame의 4번 subframe에서, 그리고 NSSS는 짝수 frame의 9번 subframe에서 전송한다.

2.3.1 NPSS 신호

NPSS는 subcarrier index $k = 0{\sim}10$의 주파수 영역에서 Zadoff-Chu 시퀀스를 식(10)에 의하여 계산한 후, LUT에 저장하고 반복적으로 사용할 수 있다.

$s(3), \ldots, s(13) = \quad 1,1,1,1,-1,-1,1,1,1,-1,1$ 이고, $u = 5$로 주어진다[14].

$$d_l(k) = S(l)\, e^{-j\pi uk(k+1)},\ k = 0,\ 1,\ \ldots 10 \quad (10)$$

2.3.2 NSSS 신호

Standalone 모드에서 NSSS는 $l = 0,1,2$ 를 제외한 132개의 RE에 식(11)로 주어지는 Zadoff-Chu 시퀀스를 이용하여 할당한다. 여기서 n_f는 frame index이며, $b_q(m)$는 문헌[14]의 표 10.2.7.2.1-1로 주어진다

$$d(n) = b_q(m) \cdot \ e^{\{-j2\pi\theta_f n\}} \ \cdot \quad e^{-\frac{j\pi un'(n'+1)}{131}} \quad (11)$$

단, $n = 0,1,\ldots,131 \quad , n' = n \ mod \ 131$

$$m = n \ mod \ 128,\ q = floor\left(\frac{N_{ID}^{NCell}}{126}\right)$$

$$u = N_{ID}^{NCell} \ mod 126 + 3$$

$$\theta_f = \frac{1}{4}\ (n_f/2) mod\ 4 = \left\{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}\right\}$$

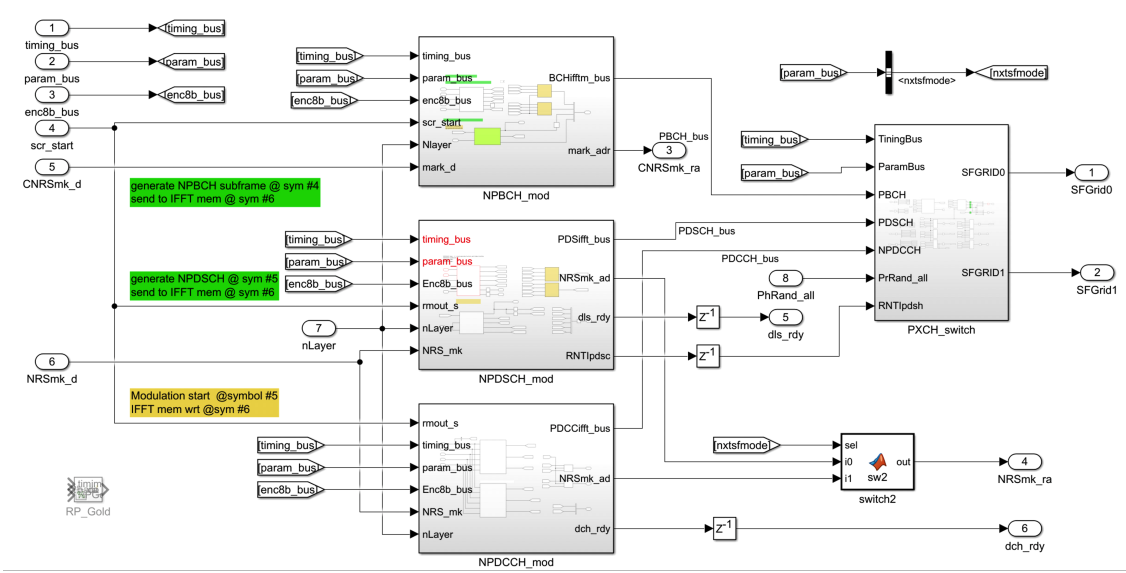


그림 15. 채널 변조 블록의 Simulink HDL 생성 모델
Fig. 15. Simulink HDL generation model of channel modulation block

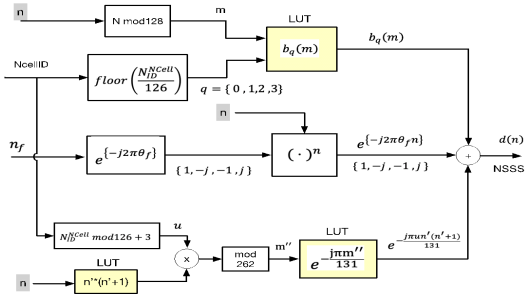


그림 16. NSSS 발생 블록의 구조
Fig. 16. NSSS generation block diagram

NSSS는 그림 16과 같이 3개의 독립적인 항목을 계산한 후 곱하는 구조로 구현한다. $b_q(m)$, $n'(n'+1)$ 및 $e^{-j\pi m''/131}$는 LUT를 이용하여 구현을 하였다.

Ⅲ. 모델 블록의 검증 및 FPGA 시험

Simulink의 HDL 생성 블록과 스크립트로 작성한 블록을 이용하여 설계한 기지국 모델은 HDL 컴파일러를 이용하여 Verilog HDL 모듈로 직접 변환이 가능하다. 본 논문에서는 FPGA 시험의 전 단계로 FILS를 이용한 검증을 하였다. 또한 FPGA에서의 동작 검증을 위하여 MATLAB 5G Toolbox를 이용한 기준 파형을 생성하고 ILA (Integrated Logic Analyzer)로 측정된 파형과 비교하는 방법을 사용하였다. 그리고 Amarisoft의 UE emulator를 이용하여 하향 링크 송신기의 성능을 검증하였다.

3.1 기준 파형 생성

MATLAB 5G Toolbox를 이용하여 기준 파형을 생성하고 기지국 Simulink 모델의 파형 및 FPGA ILA 측정 파형의 검증에 이용한다. 표5는 기준 신호의 예로서 100 subframe 구간에 전송한 물리 채널을 나타낸 것이다.

표 5의 전송 스케줄에 따라 생성한 안테나 0 및 안테나 1의 기저 대역 신호의 파형을 그림 17에 보여 주

표 5. 테스트 채널 전송 스케줄
Table 5. Transmission schedule for test

frame index

subframe index	0	1	2	3	4	5	6	7	9	10
0	NPBCH	NPBCH	NPBCH	NPBCH	NPBCH	NPBCH	NPBCH	NPBCH	NPBCH	NPBCH
1	NPDCCH	Unused	NPDSCH	NPDCCH	SIBs	NPDSCH	Unused	NPDCCH	Unused	Unused
2	NPDCCH	NPDCCH	NPDSCH	NPDCCH	SIBs	NPDSCH	NPDCCH	NPDCCH	NPDSCH	NPDSCH
3	Unused	NPDCCH	NPDSCH	NPDCCH	SIBs	NPDSCH	NPDCCH	NPDCCH	NPDSCH	NPDSCH
4	SIB1-NB	NPDCCH	SIB1-NB	NPDCCH	SIB1-NB	NPDSCH	SIB1-NB	Unused	SIB1-NB	NPDSCH
5	NPSS	NPSS	NPSS	NPSS	NPSS	NPSS	NPSS	NPSS	NPSS	NPSS
6	Unused	NPDCCH	NPDSCH	Unused	SIBs	Unused	NPDCCH	NPDSCH	NPDSCH	NPDSCH
7	Unused	NPDCCH	Unused	Unused	Unused	Unused	NPDCCH	NPDSCH	NPDSCH	NPDSCH
8	Unused	Unused	Unused	SIBs	Unused	Unused	NPDCCH	NPDSCH	Unused	NPDSCH
9	NSSS	Unused	NSSS	SIBs	NSSS	Unused	NSSS	Unused	NSSS	NPDSCH

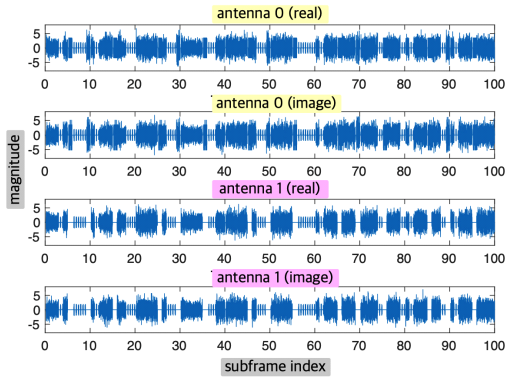


그림 17. 기저 대역 기준 파형
Fig. 17. Reference baseband waveform

었다.

3.2 FILS (FPGA In the Loop Simulation)

HDL로 변환된 송신기 블록을 FPGA에서 테스트를 하기 위해서는 대부분의 경우 입출력 신호들의 인터페이스 을 제어하기 위한 추가적인 HDL testbench 설계가 필요하지만, FILS는 FPGA에 구현된 블록에 Simulink 환경에서 입력을 인가하고 출력을 비교할 수 있는 Simulink testbench와 FIL DUT (Device Under test) 블록을 제공한다[10]. 본 논문에서는 그림 18과 같이 NB-IoT 송신기 FIL DUT 블록과 Simulink 블록을 FILS 환경에서 동시에 실행하고 결과 비교를 위한 모델을 작성하였다.

실시간으로 실행된 FIL DUT의 출력 파형은 Simulink 블록의 파형과 비교하여 FPGA 구현의 오류 여부를 검증하였다. 그림 19는 안테나 0의 Simulink 모델의 파형과 FILS 파형을 중첩하여 비교한 파형과 오차를 계산하여 보여준 것이다

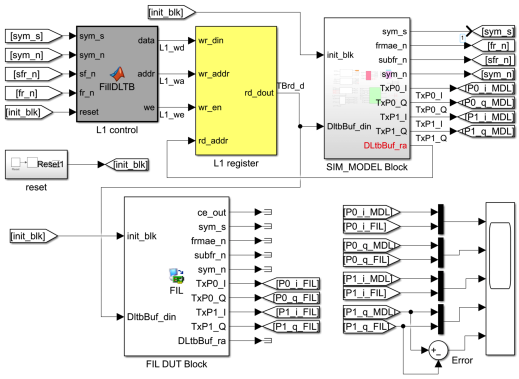


그림 18. FIL 블록을 이용한 Simulink testbench
Fig. 18. Simulink FILS testbench using FIL block

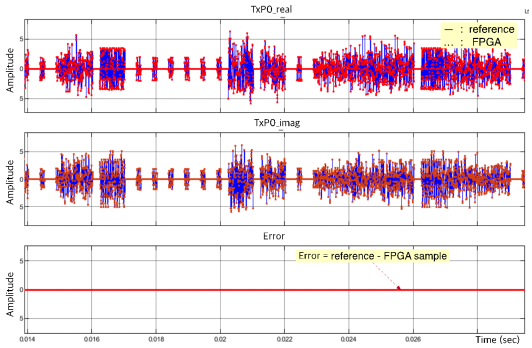


그림 19. 모델 블록과 FIL 블록 파형 및 오차
Fig. 19. Waveform comparison and error

3.3 송신 블록의 FPGA 시험 및 결과 검증

송신 블록은 Xilinx ZCU-104 Zynq Ultrascale+ 개발 보드[16]에 구현되어 FPGA 시험을 하였다. 송신 블록 제어 코드는 Zynq PS (Processing System)에서 실행되며, 송신 제어를 위하여 FPGA DPRAM (Dual Port RAM)으로 구현된 L1 레지스터에 subframe 주기로 채널 부호화 및 채널 변조 파라미터를 기록한다.

송신 블록은 L1 레지스터를 확인하여 부호화 명령이 확인 될 경우, 채널 부호화 블록을 실행하고 변조 제어 모듈의 FSM을 선택하여 활성화 시킨다. 활성화 된 FSM들은 출력해야 할 subframe 채널 타입을 확인하여 해당 채널을 변조하여 출력하고 FSM의 상태를 갱신한다. DUT 블록의 출력 파형을 MATLAB 생성 기준 파형과 비교 검증하기 위하여 ILA로 획득하였다. 그림 20은 FPGA에서 기지국 송신 블록의 기저 대역 파형을 검증하기위한 시험 환경을 나타낸 것이다.

ILA에서는 최대 2 subframe 길이의 파형을 관측하여 CSV(comma separated values) 파일로 저장하고, 테스트 시나리오에 따라 미리 생성한 기준 파형과 비교를 하여 송신 블록의 출력 파형을 검증하였다. 그림 21은 SIB1과 NPSS subframe을 테스트한 예이며, 그림 22는 SIB1 subframe 구간을 확대하여 관측한 파형

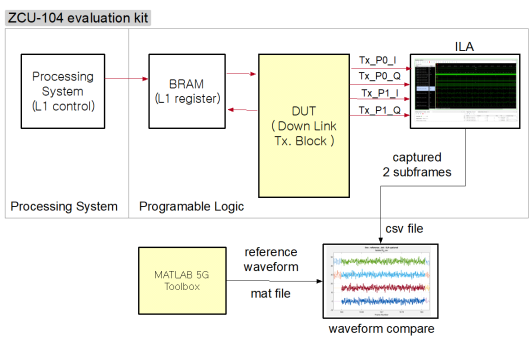


그림 20. 송신 블록의 기저 대역 파형을 시험 환경
Fig. 20. Test setup for downlink baseband waveform

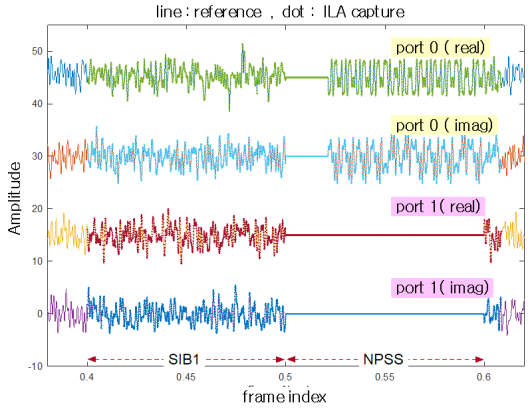


그림 21. Frame 0 : subframe 4,5 파형 검증
Fig. 21. Verification for Frame 0 : subframe 4,5

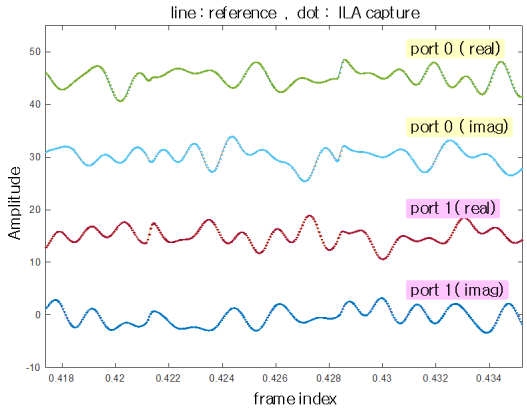


그림 22. Frame 0 : subframe 4의 구간 파형
Fig. 22. Verification for Frame 0 : subframe 4

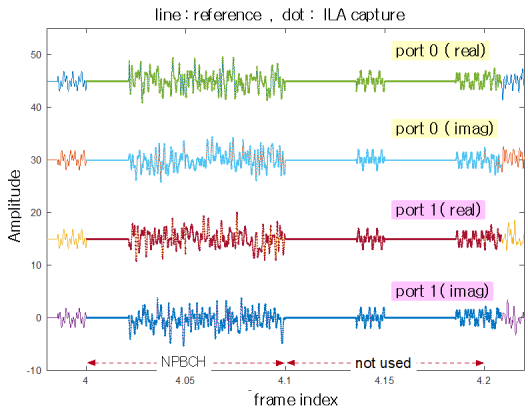


그림 23. Frame 4 : subframe 0,1 파형 검증
Fig. 23. Verification for Frame 4 : subframe 0,1

이다. 그리고 그림 23은 NPBCH와 전송에 사용하지 않는 subframe의 파형을 테스트한 결과 이다. 기준 신호는 실선, ILA 파형은 dot 로 중첩하여 표현하였다.

3.4 UE emulator 수신 시험

NB IoT 하향 링크 수신 성능은 Amarisoft의 NB-IoT UE emulator[12]를 이용하여 확인하였다. 그림 24와 같이 송신기의 기저 대역 신호는 AD9361에 의하여 RF 신호로 변환된 후, UE emulator에서 수신 및 복조를 한다.

UE emulator의 수신기는 NPSS, NSSS를 이용하여 동기를 획득하고, NPBCH에서 MIB를 성공적으로 수신하였다. 그림 25는 NPBCH 수신 성공 결과를 보여준 것이고, 그림 26은 NPDSCH (SIB1 및 SIBs) 의 수신 결과와 constellation을 보여준 것이다.

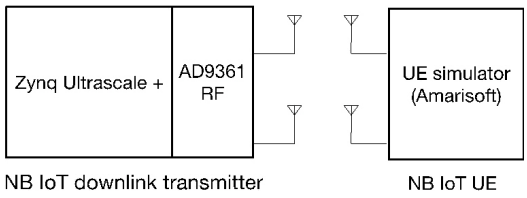


그림 24. UE emulator를 이용한 하향링크 시험
Fig. 24. Downlink test using UE emulator

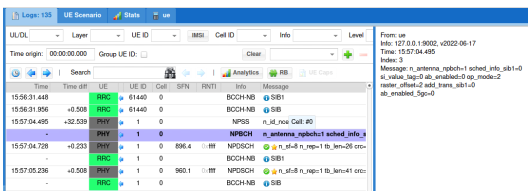


그림 25. NPBCH 수신 시험
Fig. 25. NPBCH receive test

3.5 자원 사용량 분석

HDL compiler를 이용하는 모델 기반의 설계는 항상 자원 사용량에 대한 우려가 따르게 된다. 본 논문에서는 동일한 FPGA(Xilinx xczu19eg-ffvc1760)에 HDL code로 구현한 NB-IoT 하향 링크 송신기의 자원 사용량(그림 27)과 SIMULINK 모델을 기반으로 HDL compiler를 이용하여 구현한 송신기의 자원 사용량 (그림 28)을 비교하였다.

표 6에는 주요 자원 사용량을 요약하여 보여 주었다. 알고리즘의 구체적인 구현 방법에 따라 자원의 사용 형태가 달라진다는 것을 인정하더라도, 모델 기반의 설계의 경우, DSP의 사용량이 다소 증가하였지만

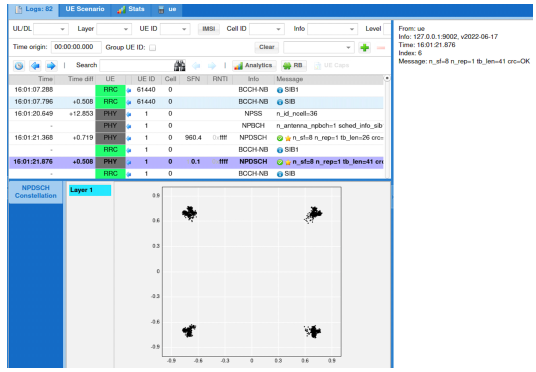


그림 26. NPDSCH 수신 시험
Fig. 26. NPDSCH receive test

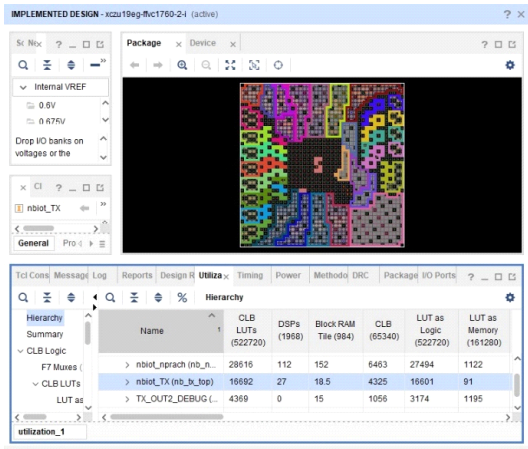


그림 27. HDL 코드로 설계한 송신기의 자원 사용
Fig. 27. Resource utilization of the transmitter implemented by HDL coding

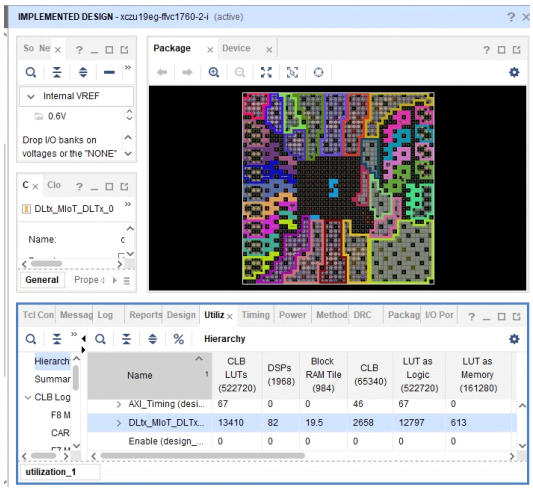


그림 28. SIMULINK HDL compiler로 설계한 송신기의 자원 사용
Fig. 28. Resource utilization of the transmitter designed by SIMULINK HDL compiler

표 6. 자원 사용량 요약
Table 6. The summary of resource utilization

Resources in the FPGA	CLB LUT	DSP	BRAM
	552,720	1,968	984
Utilization (HDL coding)	16,692	27	18.5
Utilization (HDL compiler)	13,410	82	19.5

CLB, BRAM의 사용량은 유사 하거나 오히려 적게 사용한 것을 알 수 있다. 따라서, 알고리즘의 모델링에 노력을 집중하면서 FPGA에 최적화된 HDL 코드를 쉽게 생성할 수 있다는 것을 알 수 있었다.

Ⅳ. 결 론

모델 기반 설계의 장점은 HDL 코딩에 소요되는 노력과 시간을 단축하고 알고리즘 설계와 결과 검증에 집중하여 빠르게 개발할 수 있다는 것이다. 이러한 장점을 활용하여 , 본 논문에서는 3GPP TS36 V.15를 적용한 NB-IoT 하향 링크의 기지국 송신기 물리 계층을 FPGA에 Simulink model을 기반으로 설계 및 구현을 하였다.

주요 자원 사용량은 약 13,000 CLB LUT와 80 DSP slice 그리고 20 BRAM tile로 추정이 되며, 이는 동일한 조건에서 HDL coding으로 설계한 경우와 비교한 결과 DSP 자원만 조금 더 사용하기 때문에 HDL compiler의 효율이 매우 높다고 판단이 되었다.

MATLAB Simulink를 이용하여 설계한 모델은 논리 시뮬레이션, Verilog HDL 생성 및 FILS (FPGA In the Loop) 검증 과정을 모두 동일한 Simulink 환경에서 실행을 한 후, Xilinx Zynq Ultrascale 개발 보드에 FPGA 구현을 하고 설계 검증을 하였다. 마지막으로 Amarisoft UE emulator를 이용하여 하향 링크 송신 신호의 오류 여부를 확인하였다.

본 논문에서 시도한 모델 기반의 설계 및 다양한 검증 방법은 HDL 코드 작성에 의한 설계 방법에 비하여 개발 기간이 단축될 뿐 아니라 편리한 기능 검증 방법을 이용할 수 있다는 것을 확인할 수 있었다.

References

[1] H. Wu, "Recent advances in technology and application of internet of things," in *Proc. Internet Things Conf.*, p. 1, 2017.

[2] I. B. F. de Almeida, L. L. Mendes, J. J. P. C. Rodrigues, and M. A. A. da Cruz, "5G waveforms for IoT applications," in *IEEE Commun. Surv. & Tuts.*, vol. 21, no. 3, pp. 2554-2567, 2019.

[3] D. Wang, D. Chen, B. Song, N. Guizani, X. Yu, and X. Du, "From IoT to 5g I-IoT: The next generation IoT-based intelligent algorithms and 5g technologies," *IEEE Commun. Mag.*, vol. 56, pp. 114-120, 2018.

[4] Y. Zhang, F. Ren, A. Wu, T. Zhang, J. Cao, and D. Zheng, "Certificateless multi-party authenticated encryption for nb-iot terminals in 5G networks," in *IEEE Access*, vol. 7, pp. 114721-114730, 2019. (https://doi.org/10.1109/ACCESS.2019.2936123)

[5] M. Chernyshev, et al., "Internet of things (IoT): Research simulators and testbeds," *IEEE Internet of Things J.*, vol. 5, no. 3, pp. 1637-1647, 2017.

[6] Y. Miao, W. Li, D. Tian, M. S. Hossain, and M. F. Alhamid, "Narrowband internet of things: Simulation and modeling," in *IEEE Internet of Things J.*, vol. 5, no. 4, pp. 2304-2314, Aug. 2018.

[7] W. Liu, J. Dong, and N. Liu, "NB-IoT key technology and design simulation method," *Telecommun. Sci.*, pp. 144-148, Jul. 2016.

[8] S. K. Jung and W. Y. Yeo, "Repetition patterns and scheduling timing for NB-IoT downlink channels," in *Proc. Symp. KICS*, pp. 348-349, Jun. 2018.

[9] Mathworks Inc., "*HDL Code User's Guide*," https://kr.mathworks.com/help/pdf_doc/hdlcoder/hdlcoder_ug.pd

[10] Mathworks Inc., https://kr.mathworks.com/help/supportpkg/xilinxfpgaboards/fpga-in-the-loop-simulation.html

[11] Mathworks Inc., "*5G Toolbox User's Guide*," https://kr.mathworks.com/help/pdf_doc/5g/5g_ug.pdf

[12] https://www.amarisoft.com/products/testmeasurements/amari-ue-simbox/

[13] 3GPP TS 36.212, "*E-UTRA Multiplexing and*

channel coding - Chap.6 Narrowband IoT," 2018.

[14] 3GPP TS 36.211, "*E-UTRA Physical channels and modulation- Chap.10 Narrowband IoT*," 2018.

[15] M. Kanj, V. Savaux, and M. Le Guen, "A tutorial on NB-IoT physical layer design," in *IEEE Commun. Surv. & Tuts.*, vol. 22, no. 4, pp. 2408-2446, Fourth quarter 2020. (https://doi.org/10.1109/COMST.2020.3022751)

[16] Xilinx Inc., https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

김 재 형 (Jae Hyung Kim)

1983년 2월 : 고려대학교 전자공학과 졸업

1985년 2월 : 고려대학교 전자공학과 석사

1989년 8월 : 고려대학교 전자공학 박사

1991년~현재 : 창원대학교 전기전자제어공학부 교수

<관심분야> 전자공학, 무선통신, FPGA 설계

[ORCID:0000-0001-9327-1173]