
CORDIC 알고리즘을 이용한 우주 통신용 BFSK 수신기의 FPGA 구현

하정우* · 이미진* · 허용원* · 윤미경* · 변건식*

*동아대학교

FPGA Implementation of a BFSK Receiver for Space Communication Using CORDIC Algorithm

Jeong-woo Ha* · Mi-jin Lee* · Yong-won Hur* · Mi-kyung Yoon* · Kun-sik Byon*

*Dong-a University

E-mail : saakam@nate.com hyukjin83@nate.com ksbyon@dau.ac.kr

본 연구보고서는 정보통신부 출연금으로 ETRI, SoC산업진흥센터에서 수행한 IT SoC 핵심설계인력양성사업의 연구결과입니다.

요약

본 논문은 Xilinx의 System Generator를 이용하여 저전력용 FSK 수신기를 구현하기 위한 논문이다. 심볼을 검출하기 위해서 16점 FFT를 사용하며, 저전력 효율을 증대하고 신뢰성을 높이기 위해 디지털로 설계한다. 수신기는 1 비트 데이터 처리를 하며 데이터 속도는 10kbps이다. 또한 FFT를 계산할 때 복소 승산을 피하기 위해 CORDIC 알고리듬을 사용하였으며 회전인자에 의한 승산을 회전기로 대체하였다. 수신기의 설계와 시뮬레이션은 먼저 Simulink로 수행하고, FPGA를 구현하기 위해 Xilinx의 System Generator를 사용하여 하드웨어 모델로 변환되며 성능이 확인된다.

ABSTRACT

This paper is to implement a low power Frequency Shift Keying(FSK) receiver using Xilinx System Generator. The receiver incorporates a 16 point Fast Fourier Transform(FFT) for symbol detection. The design units of the receiver are digital designs for better efficiency and reliability. The receiver functions on one bit data processing and supports data rates 10kbps. In addition CORDIC algorithm is used for avoiding complex multiplications while computing FFT, multiplication of twiddle factor is substituted by rotators. The design and simulation of the receiver is carried out in Simulink, then the simulink model is translated to a hardware model to implement FPGA using Xilinx System Generator and to verify performance.

키워드

BFSK, CORDIC, FFT, System Generator, FPGA

I. 서 론

최근 FPGA 설계 방법으로 HDL보다 효율적이며 직감적으로 이해하기 쉬운 설계 환경인 Simulink 모델로 구현하는 시스템 레벨 설계가 주목되고 있다. 이러한 시스템 레벨 설계의 실현 방법으로 Altera의 DSP Builder와 Xilinx의 System Generator의 전용 블록으로 작성한 Simulink 모델에서 HDL 코드를 자동 생성하여 FPGA를 구현하는 방법이 있다. 이와 같은 자동 코드 생성 기능은 HDL 코드를 모르고도 FPGA를 구현할 수 있으며 번잡한 코드의 디버그 작업을 대폭적으로 줄일 수 있다. 본 논문은 우주 통신에 응용할 수 있는 저전력 시스

템 구성이 목적이며 이를 구성하여 성능을 평가하고자 한다. 저전력 FSK에 기반한 수신기 구조는 Grayver[1] 가 제안하였으며, FFT 검출 기술을 사용하였다. 본 연구는 FSK 수신기 설계에 Grayver 구조를 이용하며, 수신기에서 FFT를 계산할 때 전력을 소비하는 승산기를 사용하여야 하므로, 승산을 없애기 위해 Bertazonni[2] 가 제안한 CORDIC 알고리듬을 사용한다. 즉, 본 연구에서는 전력 효율을 개선하기 위해 FFT 계산에 CORDIC 방법을 이용한다. 먼저 FSK 송신기를 설계하고 송신 신호에 부가 잡음과 도플러 천이를 추가하여 수신기를 구성한다. 수신기는 먼저 Matlab과 Simulink

로 설계하고 10kbps의 데이터 속도에서 이를 시험한다. 그 다음 FPGA를 구현하기 위해 Xilinx의 System Generator를 사용하여 하드웨어 모델로 변환된다. 이 하드웨어 모델은 Xilinx Spartan-3 FPGA로 합성, 구현되어 시스템 성능을 확인한다.

II. FSK

FSK는 신호원의 데이터에 따라 주파수가 변화하는 변조 방식으로 FSK 변조의 일반적인 해석적 표현은 식 (1)과 같다.

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos(\omega_i t + \phi), \quad 0 \leq t \leq T, i=1,2,\dots,M \quad (1)$$

여기서 $s_i(t)$ 는 변조된 캐리어, E는 정보의 심볼 폭 T에 대한 $s_i(t)$ 의 에너지, ω_i 는 M개의 이산치를 갖는 주파수 항이고 위상 항 ϕ 는 임의 상수이다. 반송파에서 정보 심볼을 추출하는 과정은 FSK 검출이라 하며, 검출 과정은 coherent 검출과 noncoherent 검출로 분류된다. coherent 검출의 수신기는 심볼을 검출하기 위해 반송파 위상을 알아야 하지만, noncoherent 검출의 수신기는 반송파 위상을 몰라도 된다. 따라서 coherent 수신기는 보통 BER 성능이 우수한 반면 설계가 복잡하고 소비 전력이 많다. 그러나 noncoherent 수신기는 BER 성능은 떨어지지만 설계가 간단하고, 전력 소비를 작아 소비 전력이 문제가 될 때는 noncoherent 수신기가 유리하다고 할 수 있다. 따라서 본 논문에서 추구하는 저전력 FSK 수신기에서는 저전력화가 중요하므로 noncoherent 수신기를 사용한다.

III. FFT 기반 검출기

noncoherent FSK 시스템의 에너지 검출 방법 중 DFT 기반 기술은 중요한 기술 중 하나이다. DFT는 이산 시간 신호의 전력 스펙트럼을 구하는 방법이며, 길이 N의 입력 $x(n)$ 에 대한 N점 DFT는 식 (2)와 같다.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \leq k \leq N-1 \quad (2)$$

식 (2)에서 $X(k)$ 는 N개의 상관기 뱅크로 볼 수 있으며, 각각의 상관기는 특정 주파수에서의 신호 에너지를 나타낸다. 신호 검출에 DFT를 사용하는 이유는 수신 신호의 도플러 주파수를 조절할 수 있기 때문이다. 그러나 DFT를 계산할 때 N^2 의 복소 승산과 N^2+N 의 복소 가산을 하여야 함으로 N이 클 때 DFT는 구현하기가 어려워진다. 이를 해결한 것이 FFT이며 계산량을 줄여 속도를 증가시킬 수 있으며 결과적으로 하드웨어 복잡도를 줄일 수 있다. 그러나 FFT에서도 복소 승산을 수행하여야 함으로서 전력 소비를 올리게 되며 저전력화에 문제가 발생한다. 본 논문에서는 CORDIC 알고리듬을 이용하여 전력 소비에 문제가 되는 복소 승산을 위상 회전으로 대체함으로서 승산에 의한 전력 소비를 줄이고자 한다.

IV. CORDIC

CORDIC 알고리듬은 주어진 각도의 sine 및 cosine을 계산하는데 사용하는 시공간적으로 효율적인 알고리듬이다. 이 알고리듬은 승산/제산을 천이 동작으로 처리함으로서 하드웨어를 쉽게 구현할 수 있다. FFT 계산을 할 때 복소 입력 $x(n)$ 은 twiddle factor W_N^i 으로 곱해지므로 승산기가 필요하지만 이는 $\exp(-j(2\pi/N))$ 와 같기 때문에 위상 회전으로 처리하면 됨으로 승산기를 생략할 수 있다. 예를 들어 $A = x + jy$ 에 $W_{16}^2 = \exp(-j(\pi/4))$ 를 곱한다는 것은 A를 $\theta = \pi/4$ 만큼 위상 회전하는 것과 같으며 다음과 같다.

$$AW_{16}^2 = (x + jy)[\cos(-\pi/4) + j\sin(-\pi/4)] \\ = \frac{\sqrt{2}}{2}[(x+y) + j(y-x)]$$

따라서

$$x' = \frac{\sqrt{2}}{2}(x+y) = (x+y)(2^{-1} + 2^{-3} + 2^{-4} + 2^{-6})$$

$$y' = \frac{\sqrt{2}}{2}(y-x) = (y-x)(2^{-1} + 2^{-3} + 2^{-4} + 2^{-6})$$

즉, $A = x + jy$ 에 $W_{16}^2 = \exp(-j(\pi/4))$ 를 곱한다는 것은 위와 같이 가산기, 감산기, 천이기로 구성할 수 있기 때문에 승산기를 사용할 필요가 없어지므로 전력 소비를 줄일 수 있고 디지털 하드웨어 구현을 가능하게 한다.

V. FSK 수신기

FSK 수신기 구성은 그림 1과 같고 수신기 규격은 표 1과 같다.



표 1. 수신기 사양

변조 형태	캐리어 주파수	데이터 속도	도풀러 천이
BFSK	473.1MHz	10kbps	± 3kHz

처음에 수신 신호는 전력 효율적인 저잡음 증폭기로 증폭되고, SAW 필터로 필터되고 서브샘플되며 1 비트 A/D 변환기를 사용하여 1 비트 precision으로 디지털ай즈된다. 수신 신호의 캐리어 주파수는 473.1MHz이고, 이 신호는 1.2MHz로 서브샘플된다[1]. 여기서 서브 샘플을 하는 이유는 전력 소비를 줄이기 위해서이다[3]. 일반적으로 전력 소비는 $P \approx CV^2f$ 이며 여기서 C는 회로 용량, f는 동작 주파수, V는 전원 전압이다. 따라서 전력 소비를 줄이려면 f를 줄여야 한다. 샘플링 주파수를 1.2MHz로 선택한 이유는 [1]에서 제안한대로 전력 소비 및 하드웨어의 크기를 줄이기 위해서이다. 또한 1 비트 A/D 변환 과정은 검출기 설계를 간단하게 하고 전력을 효율적으로 만든다. 1 비트 디지털ай즈된 신호는 처음에 복소

지수를 가진 신호를 곱함으로서 다운 변환되고 기저대역으로 전환된다. 서브 샘플링 주파수를 적절히 선택하면 출력을 $i, 1, -i, -1$ 중의 하나로 얻을 수 있으며 승산을 제거하여, 사실상 하드웨어 승산기의 사용을 피할 수 있다. 또한 복소 승산 과정은 'I'와 'Q' 신호를 생성하며, 이 경로의 각각에 별도의 처리를 요구한다[3]. 1.2MHz 신호는 50kbps의 데이터 속도를 얻기 위해 24로 데시메이션(1.2MHz/24=50kbps)되어 workspace에 저장되며 FFT 검출기에서 다시 10kbps를 얻는다.

FFT 검출기는 신호 중 5 샘플만을 취하며 나머지 11 샘플은 0을 패딩한다. 이 방법은 16점 FFT 계산 4 단 중 한 단을 제거하며 회로 복잡도를 줄일 수 있다.

VI. 검출과정

신호를 검출하기 위해 16점 DFT 검출기를 사용하여 이 과정은 그림 2와 같다. DFT를 계산하기 위해 FFT 알고리듬이 사용되며, 이는 작은 연산으로 처리함으로서 전력을 크게 절약할 수 있다.

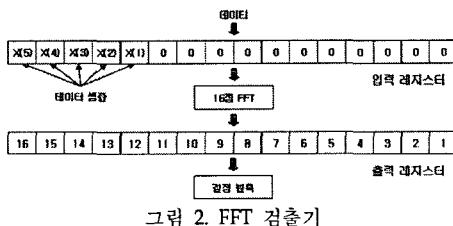


그림 2. FFT 검출기

FFT의 입력 레지스터는 5 샘플의 신호만을 취하고 11개의 0을 저장한다. 5개의 신호 샘플만을 취했을 때의 성능 손실을 0.25dB 이하로 유지하기에 충분하며 대신에 전력을 크게 줄일 수 있다[1]. FFT 출력은 16개이고 'bin'이라 하며 서로 다른 주파수 구간에서의 신호의 에너지를 나타낸다. 데이터 검출은 16개 bin을 해석함으로서 행해진다. 16개 bin 중에서 하나를 '결정(decision) bin'으로 취급하고, 결정 bin의 출력은 비교기를 이용하여 기지의 임계치와 비교된다. 결정 bin 값이 임계치보다 크면 심볼 1로 결정하고, 그렇지 않으면 심볼 0으로 결정한다.

VII. Simulink 모델

실험에 사용한 Simulink 모델은 그림 3과 같다.

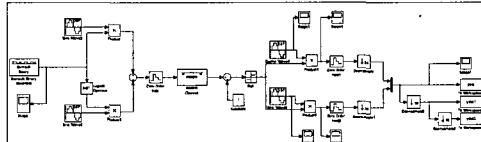


그림 3. FSK 수신기의 Simulink 모델

(1) Simulink 모델의 동작

송신기는 정보원으로 Bernoulli 랜덤 넘버 generator를 사용한다. 랜덤 generator는 특정 확률을 가지는 1 또는 0을 출력하며 속도는 10kbps로 설정한다. FSK를 구

성하기 위해 사용한 캐리어 주파수는 437.1MHz, 437.12MHz를 사용하였다. 즉, 톤 간격은 20kHz이다.

수신기는 기본적으로 1 비트 A/D 변환기와 같은 기능을 하는 sign 블록으로 시작하며, sign 블록은 1.2MHz의 샘플링율에서 동작한다. 1.2MHz를 사용한 이유는 전력 소비를 줄이기 위해서이다[1]. sign 블록의 출력은 복소 지수를 가진 신호를 곱함으로서 기저대역으로 다운 변환된다. 앞에서 논의한대로, 복소 곱은 적절한 샘플링 주파수를 선택함으로서(37.5kHz) $1, i, -1, -i$ 를 가진 꼽으로 줄어든다[3].

다운 변환된 신호는 데이터 속도를 50kbps로 줄이기 위해 24로 다운 샘플되며, FFT 검출기로 입력된다. FFT 블록은 DFT 계산을 하기 위해 단지 신호의 5 샘플만을 취득하고, 나머지 11개 입력은 0을 패딩한다. 이 과정은 5로 데이터 속도를 추가 다운 샘플하여, 최종 데이터 속도를 10kbps로 맞춘다. 다운 샘플된 신호는 workspace에 저장한 후 Matlab을 이용하여 처음 5비트를 취하고 11개의 0을 패딩한 후 FFT 처리한다. FFT 출력에서 결정 bin을 찾아 임계치와 비교함으로서 데이터를 추출할 수 있다.

(2) 시뮬레이션 결과

시뮬레이션 결과 파형은 아래와 같다. 그림 4는 FSK 송신기 출력 파형이며, 그림 5는 signum 출력 파형이다.



그림 4. 송신기 출력



그림 5. signum 출력

	0.0000	0.0001	0.0002	0.0003	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.0010	0.0011	0.0012	0.0013	0.0014	0.0015	0.0016	0.0017	0.0018	0.0019	0.0020	0.0021	0.0022	0.0023	0.0024	0.0025	0.0026	0.0027	0.0028	0.0029	0.0030	0.0031	0.0032	0.0033	0.0034	0.0035	0.0036	0.0037	0.0038	0.0039	0.0040	0.0041	0.0042	0.0043	0.0044	0.0045	0.0046	0.0047	0.0048	0.0049	0.0050	0.0051	0.0052	0.0053	0.0054	0.0055	0.0056	0.0057	0.0058	0.0059	0.0060	0.0061	0.0062	0.0063	0.0064	0.0065	0.0066	0.0067	0.0068	0.0069	0.0070	0.0071	0.0072	0.0073	0.0074	0.0075	0.0076	0.0077	0.0078	0.0079	0.0080	0.0081	0.0082	0.0083	0.0084	0.0085	0.0086	0.0087	0.0088	0.0089	0.0090	0.0091	0.0092	0.0093	0.0094	0.0095	0.0096	0.0097	0.0098	0.0099	0.0100	0.0101	0.0102	0.0103	0.0104	0.0105	0.0106	0.0107	0.0108	0.0109	0.0110	0.0111	0.0112	0.0113	0.0114	0.0115	0.0116	0.0117	0.0118	0.0119	0.0120	0.0121	0.0122	0.0123	0.0124	0.0125	0.0126	0.0127	0.0128	0.0129	0.0130	0.0131	0.0132	0.0133	0.0134	0.0135	0.0136	0.0137	0.0138	0.0139	0.0140	0.0141	0.0142	0.0143	0.0144	0.0145	0.0146	0.0147	0.0148	0.0149	0.0150	0.0151	0.0152	0.0153	0.0154	0.0155	0.0156	0.0157	0.0158	0.0159	0.0160	0.0161	0.0162	0.0163	0.0164	0.0165	0.0166	0.0167	0.0168	0.0169	0.0170	0.0171	0.0172	0.0173	0.0174	0.0175	0.0176	0.0177	0.0178	0.0179	0.0180	0.0181	0.0182	0.0183	0.0184	0.0185	0.0186	0.0187	0.0188	0.0189	0.0190	0.0191	0.0192	0.0193	0.0194	0.0195	0.0196	0.0197	0.0198	0.0199	0.0200	0.0201	0.0202	0.0203	0.0204	0.0205	0.0206	0.0207	0.0208	0.0209	0.0210	0.0211	0.0212	0.0213	0.0214	0.0215	0.0216	0.0217	0.0218	0.0219	0.0220	0.0221	0.0222	0.0223	0.0224	0.0225	0.0226	0.0227	0.0228	0.0229	0.0230	0.0231	0.0232	0.0233	0.0234	0.0235	0.0236	0.0237	0.0238	0.0239	0.0240	0.0241	0.0242	0.0243	0.0244	0.0245	0.0246	0.0247	0.0248	0.0249	0.0250	0.0251	0.0252	0.0253	0.0254	0.0255	0.0256	0.0257	0.0258	0.0259	0.0260	0.0261	0.0262	0.0263	0.0264	0.0265	0.0266	0.0267	0.0268	0.0269	0.0270	0.0271	0.0272	0.0273	0.0274	0.0275	0.0276	0.0277	0.0278	0.0279	0.0280	0.0281	0.0282	0.0283	0.0284	0.0285	0.0286	0.0287	0.0288	0.0289	0.0290	0.0291	0.0292	0.0293	0.0294	0.0295	0.0296	0.0297	0.0298	0.0299	0.0300	0.0301	0.0302	0.0303	0.0304	0.0305	0.0306	0.0307	0.0308	0.0309	0.0310	0.0311	0.0312	0.0313	0.0314	0.0315	0.0316	0.0317	0.0318	0.0319	0.0320	0.0321	0.0322	0.0323	0.0324	0.0325	0.0326	0.0327	0.0328	0.0329	0.0330	0.0331	0.0332	0.0333	0.0334	0.0335	0.0336	0.0337	0.0338	0.0339	0.0340	0.0341	0.0342	0.0343	0.0344	0.0345	0.0346	0.0347	0.0348	0.0349	0.0350	0.0351	0.0352	0.0353	0.0354	0.0355	0.0356	0.0357	0.0358	0.0359	0.0360	0.0361	0.0362	0.0363	0.0364	0.0365	0.0366	0.0367	0.0368	0.0369	0.0370	0.0371	0.0372	0.0373	0.0374	0.0375	0.0376	0.0377	0.0378	0.0379	0.0380	0.0381	0.0382	0.0383	0.0384	0.0385	0.0386	0.0387	0.0388	0.0389	0.0390	0.0391	0.0392	0.0393	0.0394	0.0395	0.0396	0.0397	0.0398	0.0399	0.0400	0.0401	0.0402	0.0403	0.0404	0.0405	0.0406	0.0407	0.0408	0.0409	0.0410	0.0411	0.0412	0.0413	0.0414	0.0415	0.0416	0.0417	0.0418	0.0419	0.0420	0.0421	0.0422	0.0423	0.0424	0.0425	0.0426	0.0427	0.0428	0.0429	0.0430	0.0431	0.0432	0.0433	0.0434	0.0435	0.0436	0.0437	0.0438	0.0439	0.0440	0.0441	0.0442	0.0443	0.0444	0.0445	0.0446	0.0447	0.0448	0.0449	0.0450	0.0451	0.0452	0.0453	0.0454	0.0455	0.0456	0.0457	0.0458	0.0459	0.0460	0.0461	0.0462	0.0463	0.0464	0.0465	0.0466	0.0467	0.0468	0.0469	0.0470	0.0471	0.0472	0.0473	0.0474	0.0475	0.0476	0.0477	0.0478	0.0479	0.0480	0.0481	0.0482	0.0483	0.0484	0.0485	0.0486	0.0487	0.0488	0.0489	0.0490	0.0491	0.0492	0.0493	0.0494	0.0495	0.0496	0.0497	0.0498	0.0499	0.0500	0.0501	0.0502	0.0503	0.0504	0.0505	0.0506	0.0507	0.0508	0.0509	0.0510	0.0511	0.0512	0.0513	0.0514	0.0515	0.0516	0.0517	0.0518	0.0519	0.0520	0.0521	0.0522	0.0523	0.0524	0.0525	0.0526	0.0527	0.0528	0.0529	0.0530	0.0531	0.0532	0.0533	0.0534	0.0535	0.0536	0.0537	0.0538	0.0539	0.0540	0.0541	0.0542	0.0543	0.0544	0.0545	0.0546	0.0547	0.0548	0.0549	0.0550	0.0551	0.0552	0.0553	0.0554	0.0555	0.0556	0.0557	0.0558	0.0559	0.0560	0.0561	0.0562	0.0563	0.0564	0.0565	0.0566	0.0567	0.0568	0.0569	0.0570	0.0571	0.0572	0.0573	0.0574	0.0575	0.0576	0.0577	0.0578	0.0579	0.0580	0.0581	0.0582	0.0583	0.0584	0.0585	0.0586	0.0587	0.0588	0.0589	0.0590	0.0591	0.0592	0.0593	0.0594	0.0595	0.0596	0.0597	0.0598	0.0599	0.0600	0.0601	0.0602	0.0603	0.0604	0.0605	0.0606	0.0607	0.0608	0.0609	0.0610	0.0611	0.0612	0.0613	0.0614	0.0615	0.0616	0.0617	0.0618	0.0619	0.0620	0.0621	0.0622	0.0623	0.0624	0.0625	0.0626	0.0627	0.0628	0.0629	0.0630	0.0631	0.0632	0.0633	0.0634	0.0635	0.0636	0.0637	0.0638	0.0639	0.0640	0.0641	0.0642	0.0643	0.0644	0.0645	0.0646	0.0647	0.0648	0.0649	0.0650	0.0651	0.0652	0.0653	0.0654	0.0655	0.0656	0.0657	0.0658	0.0659	0.0660	0.0661	0.0662	0.0663	0.0664	0.0665	0.0666	0.0667	0.0668	0.0669	0.0670	0.0671	0.0672	0.0673	0.0674	0.0675	0.0676	0.0677	0.0678	0.0679	0.0680	0.0681	0.0682	0.0683	0.0684	0.0685	0.0686	0.0687	0.0688	0.0689	0.0690	0.0691	0.0692	0.0693	0.0694	0.0695	0.0696	0.0697	0.0698	0.0699	0.0700	0.0701	0.0702	0.0703	0.0704	0.0705	0.0706	0.0707	0.0708	0.0709	0.0710	0.0711	0.0712	0.0713	0.0714	0.0715	0.0716	0.0717	0.0718	0.0719	0.0720	0.0721	0.0722	0.0723	0.0724	0.0725	0.0726	0.0727	0.0728	0.0729	0.0730	0.0731	0.0732	0.0733	0.0734	0.0735	0.0736	0.0737	0.0738	0.0739	0.0740	0.0741	0.0742	0.0743	0.0744	0.0745	0.0746	0.0747	0.0748	0.0749	0.0750	0.0751	0.0752	0.0753	0.0754	0.0755	0.0756	0.0757	0.0758	0.0759	0.0760	0.0761	0.0762	0.0763	0.0764	0.0765	0.0766	0.0767	0.0768	0.0769	0.0770	0.0771	0.0772	0.0773	0.0774	0.0775	0.0776	0.0777	0.0778	0.0779	0.0780	0.0781	0.0782	0.0783	0.0784	0.0785	0.0786	0.0787	0.0788	0.0789	0.0790	0.0791	0.0792	0.0793	0.0794	0.0795	0.0796	0.0797	0.0798	0.0799	0.0800	0.0801	0.0802	0.0803	0.0804	0.0805	0.0806	0.0807	0.0808	0.0809	0.0810	0.0811	0.0812	0.0813	0.0814	0.0815	0.0816	0.0817	0.0818	0.0819	0.0820	0.0821	0.0822	0.0823	0.0824	0.0825	0.0826	0.0827	0.0828	0.0829	0.0830	0.0831	0.0832	0.0833	0.0834	0.0835	0.0836	0.0837	0.0838	0.0839	0.0840	0.0841	0.0842	0.0843	0.0844	0.0845	0.0846	0.0847	0.0848	0.0849	0.0850	0.0851	0.0852	0.0853	0.0854	0.0855	0.0856	0.0857	0.0858	0.0859	0.0860	0.0861	0.0862	0.0863	0.0864	0.0865	0.0866	0.0867	0.0868	0.0869	0.0870	0.0871	0.0872	0.0873	0.0874	0.0875	0.0876	0.0877	0.0878	0.0879	0.0880	0.0881	0.0882	0.0883	0.0884	0.0885	0.0886	0.0887	0.0888	0.0889	0.0890	0.0891	0.0892	0.0893	0.0894	0.0895	0.0896	0.0897	0.0898	0.0899	0.0900	0.0901	0.0902	0.0903	0.0904	0.0905	0.0906	0.0907	0.0908	0.0909	0.0910	0.0911	0.0912	0.0913	0.0914	0.0915	0.0916	0.0917	0.0918	0.0919	0.0920	0.0921	0.0922	0.0923	0.0924	0.0925	0.0926	0.09

FFT 출력 중에서 결정 bin을 찾아 임계치를 결정하여 데이터를 판별한다. 이 경우는 결정 bin을 6으로 하고 임계치를 2로 하여 판정하면 데이터는 1, 0, 1, 1, 0, 1, 1, 1, 0이 되어 정확히 복구되었음을 확인할 수 있다.



그림 8. Bernoulli 송신기 출력

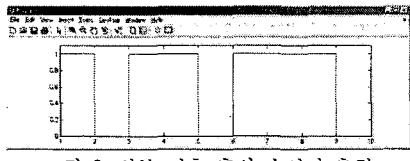


그림 9. 심볼 검출 후의 수신기 출력

그림 8은 입력파형을 보이며, 그림 9는 Matlab으로 그린 최종 검출 파형을 나타낸다. 송신 파형과 검출된 파형은 일치함을 알 수 있다.

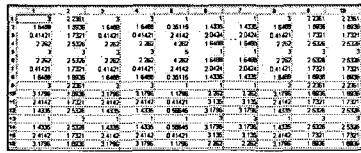


그림 10. 5kHz의 도플러 천이 후의 FFT 검출기 출력

도플러 불변에 대해 수신기를 시험하기 위해, 송신기의 캐리어 주파수를 5kHz 증가시키고, 모델을 실행하였다. 그럼 10은 처음 11개 시간 순간에서의 FFT 검출기 출력을 표시한다. 그럼에서 알 수 있듯이 결정 bin은 더 이상 bin 6이 아니며 다른 bin으로 변위되었다.

이 경우에는 bin 8을 선택하였다. 주파수 추적 알고리듬은 결정 bin의 크기에서의 변화를 추적하는데 도움이 된다. 결정 bin의 크기에서 감소를 볼 수 있듯이, 또한 결정 bin에 인접한 bin의 크기에서 비례적인 증가를 볼 수 있다. 즉, 주파수 추적 알고리듬의 기능은 결정 bin의 크기가 특정 레벨 이하로 떨어질 때 bin 수를 증가 /감소하는 것이다.

(3) 도플러 천이에 대한 수신기 성능

송신 신호 주파수를 여러 가지로 변화시킴으로서 도플러 주파수 변화에 대해 수신기를 실험하였다. 수신기는 3kHz의 도플러 천이에 대해 만족스럽게 동작함을 확인하였다. 표 2는 시뮬레이션 결과를 보여준다.

표 2. 도플러 청이에 대한 수신기 성능

주파수변화	100Hz	200Hz	500Hz	1kHz	1.5kHz	3kHz
결정 bin	7	7	7	12	12	10

(4) 수신기 BER 성능

AWGN 채널에서 송신된 신호의 SNR을 바꿈으로서

접음 조건의 변화에 대해 모델을 실험하였다. 입력 신호 전력은 상수 100mW로 설정하였다. 표 3은 여러 가지 SNR에 대해 얻은 결과를 보인다. BER은 송신기에서 1000개 정보 심볼을 송신하여, 오류적으로 검출된 심볼 수를 측정함으로서 측정하였다.

표 3. 수신기의 BER 성능

SNR (dB)	100bps		1kbps		10kbps	
	Error	BER	Error	BER	Error	BER
1	13	0.13	127	0.127	1238	0.1238
5	11	0.11	103	0.103	1047	0.1047
10	6	0.06	65	0.065	695	0.0695
15	4	0.04	22	0.022	192	0.0192
20	0	0	2	0.002	8	0.0008
25	0	0	0	0	0	0

(5) A/D 클럭 변화에 대한 수신기 성능

클럭 주파수 1.2MHz, 데이터 속도 10kbps, SNR 15dB일 때의 A/D 변환기의 클럭 변화에 대한 수신기 성능은 표 4와 같다.

표 4. A/D 변환기의 클럭 변화에 대한 수신기 성능

주파수 변화(%)	0.1	0.2	0.3	0.4	0.5
BER	0.008	0.012	0.017	0.024	0.036

A/D 변환기에서 클럭 주파수를 작은 량만큼 바꾸어서 수신기를 시험하였다. 표 4는 시뮬레이션 결과이다. 이러한 주파수 변화는 원 클럭 속도의 $\pm 0.5\%$ 로 제한하였다. 표 4에서 알 수 있듯이, 수신기 성능은 클럭 주파수에서의 변화가 증가할 때 열화됨을 알 수 있다.

VIII. 수신기의 Xilinx 구현

Simulink로 완성한 시스템을 FPGA로 구현하기 위해 Xilinx사의 System Generator를 사용하였다.

(1) System Generator 설계 모델

그림 11과 같이 송신기와 채널 블록은 Simulink 블록을 사용하여 모델화하고, 수신기는 Xilinx 블록셋을 사용하여 모델화하여 서브시스템 'Subsystem1'로 구성하였다.

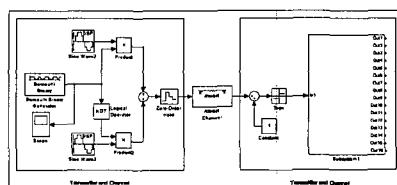


그림 11. System Generator 모델

그림 11에서 본 Subsystem1은 두 개의 서브시스템으로 구성되며, 그림 12의 첨 서브시스템은 디지타이즈된

신호의 다운 변환과 데시메이션을 하고, 두 번째 서브 시스템은 FFT 검출을 한다.

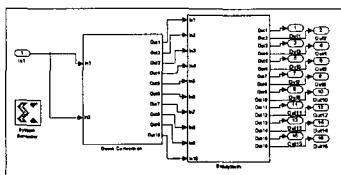


그림 12. System Generator block으로 구현한 수신기 모델

(2) Xilinx Model : Down conversion

그림 13은 수신기의 다운 변환 서브시스템의 하드웨어 설계이다. 설계에 사용된 모든 블록은 Xilinx 블록셋에서 제공되며, Xilinx 하드웨어로 바로 합성될 수 있다. Simulink 블록과 같이 24 분주된 출력은 S/P 블록을 이용하여 병렬화되고 실수 5 비트, 헤수 5 비트를 출력한다.

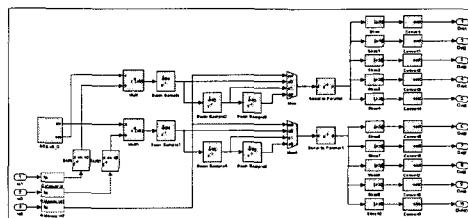


그림 13. Down Conversion의 System Generator 모델

(3) Xilinx Model : FFT 검출기

FFT 검출기는 Bertazzoni[2]가 제안한 FFT 알고리듬을 기반으로 CORDIC 알고리듬을 이용하여 구성한다.

(4) 시뮬레이션 결과

Xilinx 모델은 AWGN 채널에서 데이터 속도 10kbps에 대해 Simulink로 시험하였다. 그림 14는 검출된 파형의 결과를 보인다. 위 파형은 Bernoulli 랜덤 생성기에에서 송신된 데이터이며, 아래의 파형은 수신기에 의해 검출된 파형이다. 송수신 파형은 서로 잘 일치되며, 수신기의 Xilinx 하드웨어 모델이 잘 동작함을 나타낸다.

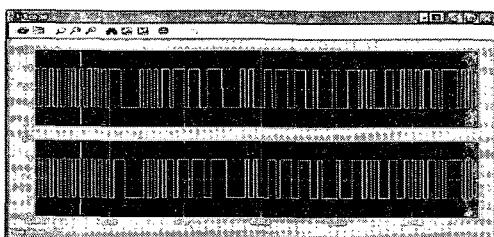


그림 14. 심볼 검출 후 표시된 파형

(5) Xilinx FPGA로의 합성

System Generator 블록을 사용하여 Xilinx 하드웨어 모델을 Xilinx FPGA로 합성한다. Product family는 Spartan-3이며, 사용한 디바이스는 xc3s200-4ft256이고, Xilinx system clock 주기는 20ns로 설정하였다. 이 모델은 ISE로 합성 및 구현될 수 있으며, Timing Analysis, Hardware co-simulation 등을 실현할 수 있다.

IX. 결론 및 향후 계획

미래의 심우주 통신을 예상하여 저전력 BFSK 수신기를 성공적으로 설계하고 시뮬레이션하였다. 수신기는 도플러 불변을 위해 16점 FFT 검출기로 구현하였고, 실험을 통해, 10kHz까지의 속도에서 정상적인 통신을 할 수 있음을 확인하였다. 또한 Xilinx System Generator를 이용하여 성공적인 하드웨어 구현을 하였으며 성능을 확인하였다.

향후 기존의 radix-2 알고리듬 대신에 radix-4 FFT 알고리듬을 사용하여 구현할 예정이며 다운 변환단에서의 송신기와 디지털 주파수 합성기를 다른 블록으로 대체하여 구성을 간단히 할 예정이다. 또한 ISE와 연동하여 FPGA 합성 및 구현을 할 예정이며 Hardware co-simulation도 할 예정이다.

참고문헌

- [1] E.Grayver and B.daneshrad, "A low power all digital FSK receiver for space applications", IEEE Trans. Communications, Vol.49, Issue : 5, May 2001 pages : 911-921
- [2] S.Bertazzoni et al, "16-point high speed (I)FFT for OFDM modulation", Proceedings of the 1998 IEEE Inter. symposium on Circuits and system, Vol.5, 31 May-3 June 1998, pages: 210-212
- [3] J.G.Proakis and D.G.Manolakis, Digital Signal Processing, Englewood Cliffs, NJ:Prentice-Hall, 1997
- [4] E.Grayver and B.daneshrad, "VLSI implementation of a 100uW multirate FSK receiver", IEEE J. Solid-State Circuits, Vol.36 Issue : 11, Nov. 2001, pages : 1821-1828
- [5] Bernard Sklar, Digital Communication, Fundamental and Applications, Englewood Cliffs, NJ : Prentice-Hall, 2001