Area-Efficient On-the-Fly Code Generator for BDS B1C Receivers

Jiwoon Park
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
jwpark.cas@gmail.com

Gwanghee Jo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
j_ghee@cnu.ac.kr

Jinseok Kim
Navcours Co., Ltd
Daejeon, Korea
sythez@gmail.com

Hoyoung Yoo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
hyyoo@cnu.ac.kr

***Abstract*— Due to the remarkable progress, BeiDou Navigation Satellite System (BDS) has increased its influence in the area of GNSS. Although there is a lot of research on algorithms to improve the positional accuracy of BDS, research on its implementation is insufficient. For the first time, we present an area-efficient code generator, which plays an important role within the BDS receivers. Whereas the straightforward LUT-based implementation demands a tremendous hardware complexity, the proposed structure can save hardware complexity by generating B1C code in an on-the-fly manner using a small size of LUTs. The straightforward and proposed code generators are synthesized using a CMOS 180nm process. Compared to the straightforward code generator, the proposed code generator reduces hardware complexity by 94.23%. As a result, the proposed code generator can be a promising candidate to implement an optimized BDS B1C receiver with a small hardware complexity.**

Keywords— BDS; code generator; on-the-fly; area-efficient

I. INTRODUCTION

The Global Navigation Satellite System (GNSS) is a system that provides the user's location by receiving signals from satellites. It is used not only in military defense, but also in various fields such as communication, geodesy, and transportation. Starting with GPS in the United States, various countries are developing their own satellite navigation systems. BeiDou navigation satellite System (BDS) in China is later in development compared to the US, but it provides navigation services similar to that of GPS due to its advanced equipment and technology. It can even show higher positioning accuracy than GPS in some situations [1].

Although there is a lots of research on algorithms to improve the performance of BDS, research on hardware is hardly studied [2]. In this paper, we present an area-efficient structure of code generator, which plays an important role within the BDS B1C receivers. The signals from satellites are continuously correlated with various B1C codes in BDS B1C receivers for signal acquisition, tracking, and data extraction. In fact, the straightforward structure can be simply implemented using LUTs that store and provide B1C codes, but the practical problem is that the size of LUT increases in proportion to the number of code length, candidate satellites and signal components. In other words, the code length should be long enough to support high correlation property. All codes assigned to each of 63 candidate satellites should be unique, and all codes assigned to different signal component should be also unique [4]. To mitigate this problem, we propose an on-the-fly code generator with a small size of LUT compared to the straightforward structure. In section 2, code generation for BDS B1C is introduced, and the proposed code generator is described in Section 3. As experimental results, the synthesis results are presented in Section 4 and the conclusion in Section 5 is described.

II. BACKGROUND

The B1C signal $S_{PRN}(t)$ transmitted by the BDS satellite is modulated using a carrier $c(t)$ of 1575.42 MHz as

$$S_{PRN}(t) = c(t) \cdot s_{PRN}(t) \tag{1}$$

, where Pseudo Random Noise (PRN) represents the candidate satellites and BDS provides total 63 satellites $1 \le PRN \le 63$. The B1C signal before modulation $s_{PRN}(t)$ consists of two components of data component $s_{PRN,d}(t)$ and pilot component $s_{PRN,p}(t)$ with power ratio of 1:3 as

$$\begin{aligned} s_{PRN}(t) &= \frac{1}{2} s_{PRN,d}(t) + j\frac{\sqrt{3}}{2} s_{PRN,p}(t) \\ &= \frac{1}{2} D_{PRN}(t) \cdot C_{PRN,d}(t) \cdot sc_d(t) + j\frac{\sqrt{3}}{2} C_{PRN,p}(t) \cdot sc_p(t) \end{aligned} \tag{2}$$

The data component $s_{PRN,d}(t)$ is a signal including a navigation message. $s_{PRN,d}(t)$ consists of a navigation message $D_{PRN}(t)$, a code $C_{PRN,d}(t)$, and a subcarrier $sc_d(t)$. In addition, the pilot component $s_{PRN,p}(t)$ is a signal for signal acquisition and tracking. Furthermore, the pilot component $s_{PRN,p}(t)$ consists of a code $C_{PRN,p}(t)$ and a subcarrier $sc_p(t)$ for the pilot component. The receiver determines acquisition and tracking of the signal using the correlation value of the codes. It can be easily acquired and tracked by calculating the correlation value with the pilot component $s_{PRN,p}(t)$. It is important to note that a unique code $C_{PRN,d}(t)$ for each PRN is required to extract the navigation data, and a unique code $C_{PRN,p}(t)$ for each PRN is required to use the pilot component $s_{PRN,p}(t)$ for calculating the correlation value [3].

More precisely, Fig. 1 depicts a timing diagram of each signal components. The code $C_{PRN,d}(t)$ for the data component consists of the primary code $C_{PRN,d,1}(t)$. A primary code of 10230-chip has a code period of 10ms with a chip rate of 1.023Mcps. Unlike the data component with an only primary code, the code $C_{PRN,p}(t)$ for the pilot component is computed by combining the primary code $C_{PRN,p,1}(t)$ and the secondary code $C_{PRN,p,2}(t)$ using modulo-2 addition. The primary code $C_{PRN,p,1}(t)$ for the pilot component has the same length of 10230-chip, period of 10ms, and chip rate of 1023Mcps as those for the data component. A secondary code $C_{PRN,p,2}(t)$ for the pilot component of 1800-chip has a code period of 18s with a chip rate of 100cps. One code period of a primary code $C_{PRN,p,1}(t)$ is equal to the duration of one chip of a secondary code $C_{PRN,p,2}(t)$ [5].

Although there are differences in chip length or period, both primary and secondary codes are generated with the same manner based on Legendre code and Weil code. First, Legendre code is computed as follows

$$L(k)=\begin{cases}0, & k=0\\ 1, & k\neq 0 \; and \; k=x^2 \bmod N_L, \quad 1\le x\le N_L\\ 0, & else\end{cases} \tag{3}$$

The value of $L(0)$ is always 0, and the value of $L(k)$ is 1 in the case that any x within the range from 1 to N can compute k by computing square and modulo operation. Otherwise, the value of $L(k)$ is 0. Length of N_L varies depending on the type of code. $N_{L,1}$ is 10243 representing the length of the primary code, and $N_{L,2}$ is 3607 representing the length of the secondary code. When N_L sets to 10243 as $N_{L,1}$, two Legendre codes, $L_{d,1}(k)$ and $L_{p,1}(k)$, can be obtained for the primary code. When N_L sets to 3607 as $N_{L,2}$, one Legendre code $L_{p,2}(k)$ can be obtained for the secondary code.

Once a Legendre code of length N_L consisting of 1s and 0s is generated, the Weil code can be computed as

$$W_{PRN}(k)=L(k)\oplus L((k+w_{PRN})\bmod N_W), \quad 0\le k\le N_W-1 \tag{4}$$

Two Legendre codes are added by bit-wise modulo-2 addition. For the k-th bit in Weil code, one bit is obtained from k-th bit in Legendre code, $L(k)$, and the other bit is obtained from $((k+w_{PRN})\bmod N_W)$-th bit in Legendre code, $L((k+w_{PRN})\bmod N_W)$. Thus, the two bits are different as much as the phase difference w_{PRN}. Note that the phase difference w_{PRN} is described in the BDS B1C Interface Control Document(ICD) [4], and the phase difference w_{PRN} differs according to the number of PRNs, data/pilot component, and primary/secondary code. $w_{PRN,d,1}$ represents the phase difference for the primary code of the data component and $w_{PRN,p,1}$ represents the phase difference for the primary code of the pilot component with the range of $1\le w_{PRN,d,1}, w_{PRN,p,1}\le 5121$. $w_{PRN,p,2}$ represents the phase difference for the secondary code of the pilot component with the range of $1\le w_{PRN,p,2}\le 1803$. The length of the Weil code N_W has the same length as the length N_L of the Legendre code due to the bit-wise operation. Thus, two primary Weil codes, $W_{PRN,d,1}(k)$ and $W_{PRN,p,1}(k)$ have the same length as $L_{d,1}(k)$ and $L_{p,1}(k)$ as 10243, and one secondary Weil code, $W_{PRN,p,2}(k)$ is the same length as $L_{p,2}(k)$ as 3607. In other words, $N_{L,1}=N_{W,1}$ and $N_{L,2}=N_{W,2}$.

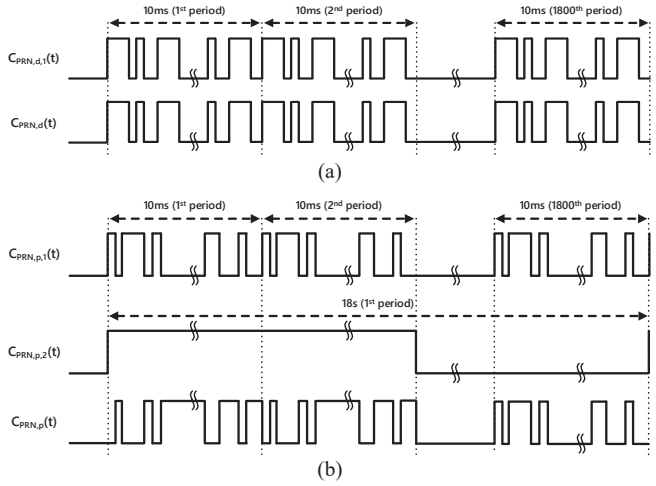

Fig. 1. Timing diagram for code period of each signal components

Lastly, the B1C code $C_{PRN}(n)$ can be obtained by applying the truncation circularly as

$$C_{PRN}(n)=W((n+p_{PRN}-1)\bmod N_W), \quad 0\le n\le N_C-1 \tag{5}$$

For the n-th bit in B1C code, one bit is obtained from $((n+p_{PRN}-1)\bmod N_W)$-th Weil code, $W((n+p_{PRN}-1)\bmod N_W)$. Similar to phase difference w_{PRN}, truncation points p_{PRN} also is described in the BDS B1C ICD[4], and truncation points p_{PRN} differs according to the number of PRNs, data/pilot component, and primary/secondary code. $p_{PRN,d,1}$ represents the truncation point for the primary code of the data component and $p_{PRN,p,1}$ represents the truncation point for the primary code of the pilot component with the range of $1\le p_{PRN,d,1}, p_{PRN,p,1}\le 10243$. $p_{PRN,p,2}$ represents the truncation point for the secondary code of the pilot component with the range of $1\le p_{PRN,p,2}\le 3607$. The length of B1C code N_C is shorter than the length of Weil code N_W due to the truncation. Two primary B1C codes, $C_{PRN,d,1}(k)$ and $C_{PRN,p,1}(k)$, have the length of 10230, and one

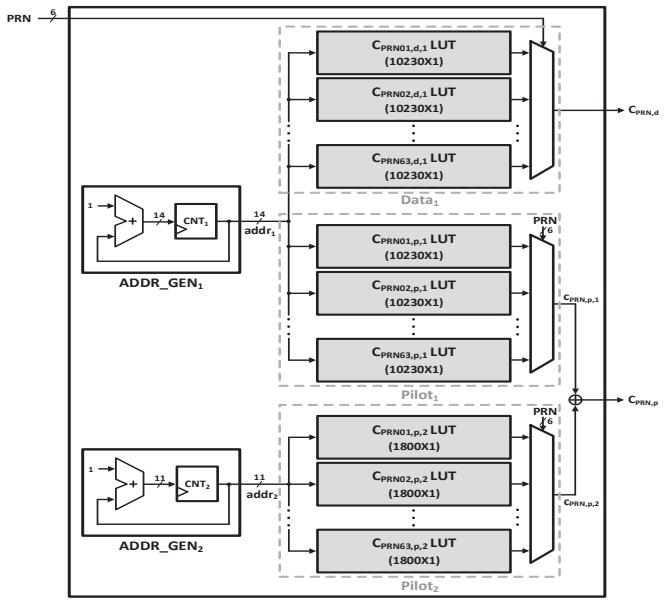

Fig. 2. Hardware Configuration of straight-forward code generator

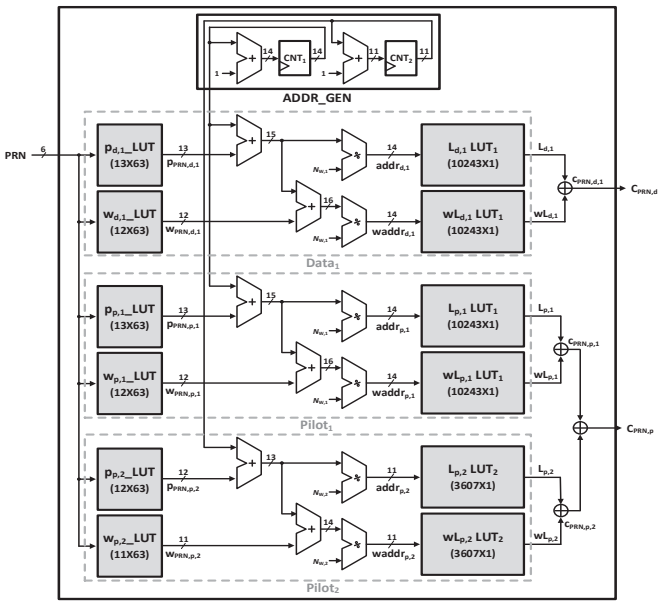

Fig. 3. Hardware Configuration of proposed code generator

secondary B1C code, $C_{PRN,p,2}(k)$, has the length of 3607. In other words, $N_{C,1}$ =10230 and $N_{C,2}$ =1800. To sum up, B1C code $C_{PRN}(n)$ can be generated based on Leg endre code and Weil code using a phase difference w_{PRN} and a truncation point p_{PRN} .

III. Proposed Code Generator

In order to acquire and track the pilot component and extract navigation data from data component in BDS B1C signals, all component B1C codes of $C_{PRN,d,1}(t)$, $C_{PRN,p,1}(t)$, $C_{PRN,p,2}(t)$ should be generated for each PRN. Thus, each PRN demands $N_{C,1}$ =10230 bits of the primary code for data component $C_{PRN,d,1}(t)$, $N_{C,1}$ =10230 bits of the primary code for pilot component $C_{PRN,p,1}(t)$, and $N_{C,2}$ =1800 bits of the secondary code for pilot component $C_{PRN,p,2}(t)$ resulting in total 22,260 bits as ($N_{C,1}+N_{C,1}+N_{C,2}$). Moreover, total bits are increased as multiple as the number of PRNs since each B1C codes for all PRN should be unique. Total code bits in a receiver becomes 1,402,380 that is 63 × 22,260 as #PRN× ($N_{C,1}+N_{C,1}+N_{C,2}$).

Fig. 2 shows the straightforward code generator that stores all the primary and secondary codes for both the pilot component and the data component with the total amount of 1,402,380. As shown in Fig. 2, given the PRN number, the straightforward code generator computes addresses and access associated LUTs to generate all types of B1C codes simultaneously. It is clear that the straightforward code generator suffers from a huge hardware complexity resulting from a large size of LUTs. There are 63 LUTs for the primary code of data component $C_{PRN,d,1}(t)$ with the length of $N_{C,1}$ =10230 bits, 63 LUTs for the primary code of pilot component $C_{PRN,p,1}(t)$ with the length of $N_{C,1}$ =10230 bits, and 63 LUTs for the secondary code of pilot component $C_{PRN,p,2}(t)$ with the length of $N_{C,2}$ =1800 bits.

To alleviate the huge size of LUTs, we propose an area-efficient code generator using small hardware complexity. Rather than storing all the codes, the proposed code generator can compute B1C codes in an on-the-fly manner using Legendre code. To compute B1C codes in on-the-fly manner, (4) is substituted into (5) resulting in

$$C_{PRN}(n) = L((n+p_{PRN}-1) \bmod N_W) \oplus L((n+w_{PRN}+p_{PRN}-1) \bmod N_W), 0 \le n \le N_C - 1 \quad (6)$$

Fig. 3 depicts the proposed code generator that computes B1C codes that calculates B1C codes in the on-the-fly manner. Whereas the straightforward code generator demands 192 LUTs with the total amount of 1,402,380 bits, the proposed code generator demands 12 LUTs with the total amount of 52,785 bits.

More precisely, the primary code of data component needs 2 LUTs for Legendre code $L_{d,1}(k)$ with the length of $N_{L,1}$ =10243 bits, 1 LUT for phase difference $w_{PRN,d,1}$ with the length of 12 bits for each PRN, and 1 LUT for truncation point $p_{PRN,d,1}$ with the length of 13 bits for each PRN. Similarly to the primary code of data component, the primary code of pilot component needs 2 LUTs for Legendre code $L_{p,1}(k)$ with the length of $N_{L,1}$ = 10243 bits, 1 LUT for phase difference $w_{PRN,p,1}$ with the length of 12 bits for each PRN, and 1 LUT for truncation point $p_{PRN,p,1}$ with the length of 13 bits for each PRN. Lastly, the secondary code of pilot component needs 2 LUTs for Legendre code $L_{p,2}(k)$ with the length of $N_{L,2}$ = 3607 bits, 1 LUT for phase difference $w_{PRN,p,2}$ with the length of 11 bits for each PRN, and 1 LUT for truncation point $p_{PRN,p,2}$ with the length of 12 bits for each PRN.

Given the number of PRN as an input, the proposed code generator calculates two addresses using LUTs of phase

difference w_{PRN} and the truncation point p_{PRN} according to (6). For the n-th iteration time, the first address and second address become $L((n+p_{PRN}\text{-}1)\text{mod}N_W)$ and $L((n+w_{PRN}+p_{PRN}\text{-}1)\text{mod}N_W)$, respectively. Next, LUTs associated with Legendre codes are accessed using the computed addresses. Lastly, B1C codes are achieved by applying modulo-2 addition between two bits from LUTs of Legendre codes. The proposed code generator can compute addresses and access associated LUTs to generate all types of B1C codes simultaneously as shown in Fig. 3.

IV. Experimental Results

To verify the advantages of the proposed structure, two types of code generators, straightforward and proposed ones, were synthesized using a CMOS 180nm process. Table 1 summarizes the synthesis results. As shown in Fig. 2 and Fig. 3, the straightforward structure demands a huge amount of 1.4M bits, but the proposed structure saves an only small amount of 5.2K bits by generating code in on-the-fly manner. Due to the difference stored in LUTs, the proposed code generator can save hardware complexity significantly compared to the straightforward code generator in terms of equivalent gate counts. In other words, the straightforward structure demands 432K equivalent gate counts, but the proposed structure demands 24K equivalent gate counts, which saves 94.23% compared to the straightforward structure.

Although the proposed code generator saves hardware complexity remarkably, the critical path delay increases slightly due to the fact that the proposed code generator computes internal addresses for Legendre code. However, the degradation for the throughput is completely negligible because the proposed code generator can generate B1C codes at the fast enough speed. In other words, the throughput of the proposed code generator can achieve up to 133.33Mcps whereas B1C primary codes demands 1.023Mcps and secondary codes demands 100cps. As a result, the proposed code generator can save hardware complexity significantly by computing B1C codes in the on-the-fly manner without affecting the system performance.

TABLE I. Synthesis Results

Metrics	Straightforward	Proposed
Total Code Bits	1,402,380 bits	52,785 bits
Equivalent Gate Count	432,861.03	24,936.32
Normilized EGC	100%	5.76%
EGC for LUTs	417,051.15	20,239.64
Critical-Path Delay	4.55 ns	7.07 ns
Operating frequency	156.25 MHz	133.33 MHz
Throughput (chip/sec)	156.25 Mcps	133.33 Mcps

V. Conclusion

In order to acquire and trace satellite signals and extract data using correlation values, the BDS receiver should provide all types of BDS B1C codes. Unlike the straightforward code generator that stores all types of BDS B1C codes using LUTs, this paper presents an area-efficient code generator that computes necessary code in an on-the-fly manner using tiny size of LUTs. According to experimental results, the proposed code generator shows 94.23% reduction in hardware complexity compared to the straightforward code generator with a negligible increase of critical path delay. As a result, the proposed on-the-fly code generator can be a promising candidate to implement an efficient BDS B1C receiver with less hardware complexity.

Acknowledgment

This work was supported by Navcours Co., Ltd.

References

[1] Ma, Xiaping, et al. "Positioning Performance Comparison Between GPS and BDS With Data Recorded at Four MGEX Stations." IEEE Access 8, 2020.

[2] L. Yafeng, Nagaraj C. Shivaramaiah, and D.M. Akos, "Design and implementation of an open-source BDS-3 B1C/B2a SDR receiver." GPS Solutions 23.3, 2019.

[3] Lu, Mingquan, et al. "Overview of BDS III new signals." Navigation, 66.1, pp. 19-35, 2019

[4] China Satellite Navigation Office, "BeiDou navigation satellite system signal in space interface control document open service signal B1C (Version 1.0).", 2017.

[5] Yan, Shulin, Shenghong Zhou, and Yuguo Zhang, "An efficient two-stage B1C signal acquisition technique for engineering implementation of the modern beidou receiver." 2018 4th International Conference on Computer and Technology Applications (ICCTA), IEEE, 2018.