ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 18-04-02

j.inst.Korean.electr.electron.eng.Vol.22,No.4,896~900,December 2018
http://dx.doi.org/10.7471/ikeee.2018.22.4.896
11

순환중복검사 부호용 하드웨어 HDL 코드 생성기
HDL Codes Generator for Cyclic Redundancy Check Codes

김 현 규*, 유 호 영*★

Hyeon-kyu Kim*, Ho-young Yoo*★

Abstract

Traditionally, Linear Shift Feedback Register (LFSR) has been widely employed to implement Cyclic Redundant Check (CRC) codes for a serial input. Since many applications including network and storage systems demand as high throughput as ever, various efforts have been made to implement CRC hardware to support parallel inputs. Among various parallel schemes, the look-ahead scheme is one of the most widely used schemes due to its short critical path. However, it is very cumbersome to design HDL codes for parallel CRC codes since the look-ahead scheme is inevitable to consider how register and input values move in the next cycles. Thus, this paper proposes a novel CRC hardware generator, which automatically produces HDL codes given a CRC polynomial and parallel factor. The experimental results verify the applicability to use the proposed generator by analyzing the synthesis results from the generated HDL code.

요 약

전통적으로 CRC 하드웨어는 선형 되먹임 시프트 레지스터를 이용하여 한 클럭 싸이클 당 하나의 비트를 처리하는 직렬 처리 방식을 사용하였다. 최근 다양한 응용 시스템에서 빠른 데이터 처리를 요구하면서 이를 만족시키기 위하여 다양한 병렬화 기법들이 제안되었고, Look-Ahead 병렬화 기법이 짧은 최대 경로 지연을 가지는 장점 덕분에 가장 널리 적용된다. 하지만 Look-Ahead 병렬 하드웨어의 경우 각 레지스터 값과 입력 데이터의 이동에 대하여 예측을 하여야 하기 때문에 직렬 하드웨어 대비 HDL 코드의 작성이 복잡하다. 따라서 본 논문에서는 다양한 CRC 다항식과 병렬화 계수를 지원할 수 있는 Look-Ahead 기반의 CRC 병렬화 하드웨어 생성기를 제안한다. 생성된 HDL 코드의 합성 결과를 분석함으로써 제안된 생성기의 활용 가능성을 판단한다.

Key words : Parallel Processing, CRC Codes, VLSI DSP, Error Detection Code, EDA

Ⅰ. 서론

순환 중복 검사(Cyclic Redundancy Check ; CRC)는 네트워크와 저장매체 등을 통하여 데이터를 전송할 때 전송된 데이터에 오류가 있는지 확인하기 위한 오류 검출 부호이다 [1]. 데이터를 전송하기 전에 주어진 데이터의 값에 따라 CRC 값을 붙여 전송하고, 데이터 전송이 끝난 후 받은 데이터의 값으로 다시 CRC를 계산한다. 이때, CRC 값이 0이 아닐 경우 데이터 전송 과정 중에서 오류가 발생한 것으로 판단한다. 이러한 CRC 부호는 전통적으로 선형 되먹임 시프트 레지스터(Linear Feedback Shift

* Dept. of Electronics Engineering, Chungnam National University
★ Corresponding author
Email : hyyoo@cnu.ac.kr, Tel : +82-42-821-6585
※ Acknowledgment
This study was supported by research fund of Chungnam National University in 2018
Manuscript received Dec. 10, 2018; revised Dec. 17, 2018; accepted Dec. 17, 2018

Register ; LFSR)를 사용하여 구현한다 [2]. 직렬 CRC 구조는 D플립플롭이 직렬로 연결된 단순한 형태이므로 다양한 종류의 CRC 다항식에 대응하는 직렬 CRC 구조 [2]를 하드웨어 기술 언어(Hardware Description Language ; HDL)로 작성하는 것이 간단하다. 그러나 고속의 CRC 코드 생성을 위한 여러 병렬화 기법들 [3][4]이 적용된 CRC 하드웨어는 HDL로의 코드 작성에 보다 많은 시간을 필요로 한다. 디지털 반도체 설계 과정에서 HDL 코드 작성 단계의 과정을 단축시키는 것으로 전체적인 설계 시간을 단축시킬 수 있다. 이 논문에서는 병렬화 기법 중 Look-Ahead CRC 하드웨어 [5]에 대한 합성 가능한 HDL 코드를 생성하는 CRC 생성기를 구현하고 생성된 HDL 코드의 합성 결과를 비교한다.

II. 본론

1. 순환 중복 검사(CRC)

CRC는 Modulo-2 정수에서 정의된 가환환(Communicative ring)의 나눗셈에 기반 하는 오류 검출 부호이다. 다시 말해서, 입력 데이터와 CRC 다항식은 한 자리 이진수를 계수로 가지는 다항식의 집합으로 표현된다. CRC부호는 입력 데이터 다항식을 CRC 다항식으로 나눈 나머지로 계산된다. 이 CRC부호를 입력 비트 열에 붙여 데이터를 전송하고 수신한 데이터를 다시 CRC 다항식으로 나누어 CRC 부호를 계산했을 때, 두 CRC부호가 다르면 전송 과정 중 오류가 발생하였음을 나타낸다. 예를 들어 x^3+x^2+1의 CRC 다항식으로 $x^7+x^3+x^2+x+1$의 8비트 데이터를 CRC 부호화하여 전송한다고 할 때 $x^7+x^3+x^2+x+1$에 대한 CRC부호는 그림 1과 같이 계산된다. 계산된 CRC부호 x는 전송하려는 비트 열의 LSB에 붙여 $x^{10}+x^6+x^5+x^4+x^3+x$로 전송된다. 그림 1은 수신된 데이터에 오류가 발생하지 않았다고 가정하고 CRC부호를 생성하는 과정과 검출하는 과정을 나타낸다.

2. CRC 하드웨어

가. LFSR 기반의 직렬 CRC 하드웨어

CRC 부호는 Modulo-2의 정수에 정의된 다항식의 나눗셈을 통하여 구할 수 있다. 이때 나눗셈을 위한 뺄셈 연산은 XOR 게이트로 구현되며 LFSR

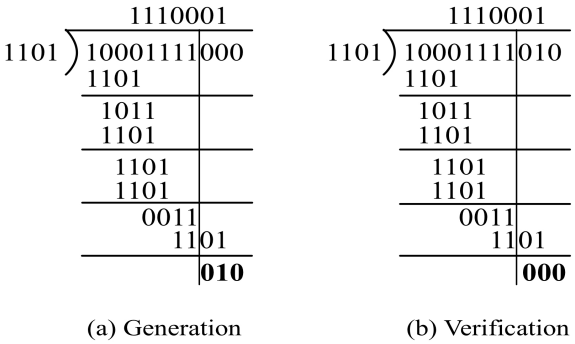


Fig. 1. Example of CRC code calculation.
그림 1. CRC 부호 계산 예제

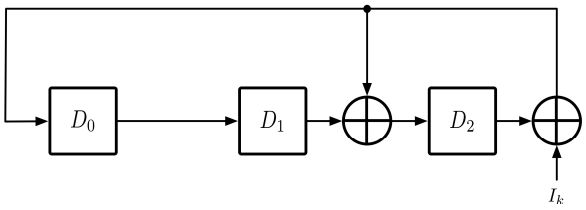


Fig. 2. Serial CRC hardware based on LFSR [2].
그림 2. LFSR 기반의 직렬 CRC 하드웨어 [2]

의 각 레지스터는 다항식의 항에 대응한다. 그림 2는 CRC 다항식이 x^3+x^2+1인 직렬 CRC 구조 [2]이며 입력 데이터를 MSB부터 한 클럭에 한 차수만큼씩 계산한다. 따라서 N 비트의 데이터를 처리하기 위해서 N 클럭이 필요하다. 고속의 신호 처리를 위해 1 클럭에 1비트 이상의 데이터를 처리하

$$\mathbf{A}^0=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\qquad \mathbf{B}^0=\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

$$\mathbf{A}^1=\begin{bmatrix}1&1&0\\0&0&1\\1&0&0\end{bmatrix}\qquad \mathbf{B}^1=\begin{bmatrix}1&0&0&0\\0&0&0&0\\1&0&0&0\end{bmatrix}$$

$$\mathbf{A}^2=\begin{bmatrix}1&1&1\\1&0&0\\1&1&0\end{bmatrix}\qquad \mathbf{B}^2=\begin{bmatrix}1&1&0&0\\1&0&0&0\\1&1&0&0\end{bmatrix}$$

$$\mathbf{A}^3=\begin{bmatrix}0&1&1\\1&1&0\\1&1&1\end{bmatrix}\qquad \mathbf{B}^3=\begin{bmatrix}0&1&1&0\\1&1&0&0\\1&1&1&0\end{bmatrix}$$

$$\mathbf{A}^4=\begin{bmatrix}1&0&1\\1&1&1\\0&1&1\end{bmatrix}\qquad \mathbf{B}^4=\begin{bmatrix}1&0&1&1\\1&1&1&0\\0&1&1&1\end{bmatrix}$$

Fig. 3. Update process of function A^4 and B^4.
그림 3. A^4, B^4의 업데이트 과정

는 병렬화 기법이 제안되었고 그 중 Look-Ahead CRC 하드웨어 [5]가 가장 널리 적용된다.

나. Look-Ahead CRC 하드웨어

Look-Ahead CRC 하드웨어 [5]는 병렬화 계수 P 만큼의 연산 과정을 미리 예상하여 1클럭 사이클에 P 비트의 데이터를 동시에 처리한다. LFSR 값에 영향을 미치는 데이터는 이전 클럭 사이클의 레지스터 값과 새롭게 입력되는 P 비트의 데이터이다. 그림 4는 Look-Ahead 함수 블록에 의해 P 비트가 병렬 처리되는 하드웨어 구조를 나타낸다. A^P는 병렬화 계수가 P 일 때, 이전 사이클 레지스터 값에 대한 Look Ahead 함수이고, X^k는 k 클럭에서 $A^P \bullet D^{k-1}$이다. B^P는 병렬화 계수가 P 일 때 P 비트의 입력 데이터에 대한 함수이고 k 클럭에서 Y^k는 $B^P \bullet I^k$ 이다. k 사이클의 레지스터 값 D^k는 식 (1)과 같이 표현 가능하다. 각 Look-Ahead 함수에 대응하는 행렬 A^p와 B^p는 초기 값인 단위 행렬 A^0와 영 행렬 B^0로부터 식 (2)와 (3)을 p 번 수행하여 얻을 수 있으며, 이는 각각 레지스터의 초기 값과 입력 데이터가 p 번 시프트가 이루어졌을 때, 식 (2)와 (3)이 i 번째 레지스터 ($1 \le i \le M$) 값에 주는 영향을 나타낸다. p ($1 \le p \le P$)는 병렬화 인덱스를 나타내고 i ($1 \le i \le M$)는 LFSR의 레지스터 인덱스를 나타낸다.

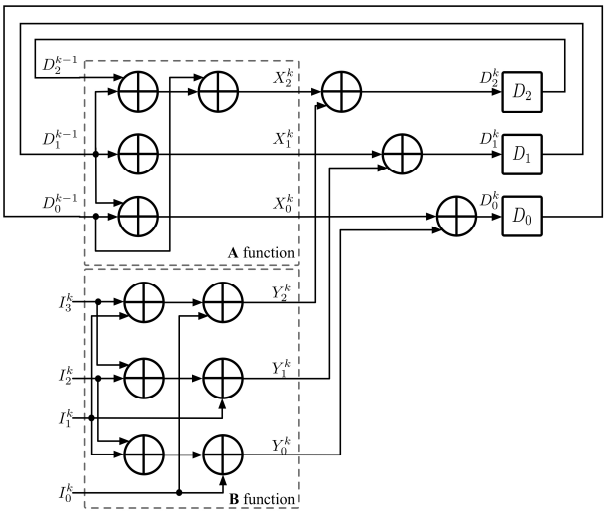


Fig. 4. 4bit-parallel Look-Ahead CRC Hardware.
그림 4. 4bit 병렬 Look-Ahead CRC 하드웨어

$$D^k = X^k \oplus Y^k = (A^p \bullet D^{k-1}) \oplus (B^p \bullet I^k) \quad (1)$$

$$A_i^p = \begin{cases} A_{i-1}^{p-1} \oplus (A_{M-1}^{p-1} \bullet c_i) & i \neq 0 \\ A_{M-1}^{p-1} & i = 0 \end{cases} \quad (2)$$

$$B_{ij}^p = \begin{cases} B_{i-1,j}^{p-1} \oplus (B_{M-1}^{p-1} \bullet c_i) & i \neq 0, j \neq p-1 \\ B_{M-1,j}^{p-1} & i = 0, j \neq p-1 \\ c_i & j = p-1 \end{cases} \quad (3)$$

식 (1)에서 I^k는 k 클럭에서 입력 P 비트를 열의 형태로 나타낸 것이고 식 (2)와 (3)에서 c_i는 CRC 다항식을 이진수의 열로 표현했을 때, i 번째 행의 원소를 나타낸다. 그림 3은 CRC 다항식이 $x^3 + x^2 + 1$ 이고 병렬화 계수가 4일 때, A^4와 B^4를 구하는 과정을 예시로 나타낸다. 구해진 A^4와 B^4로 $x^3 + x^2 + 1$에 대응하는 Look-Ahead CRC 하드웨어 [5]는 그림 4와 같다. 하지만 Look-Ahead 함수 A와 B를 구하는 과정으로 인해서 병렬화 계수가 커질수록 LFSR 구조와 비교하여 HDL 코드의 작성이 어렵고, 유사한 코드의 반복으로 불필요한 시간을 소요하게 된다. 따라서 이 논문에서는 Look-Ahead 함수의 업데이트와 이에 대응하는 HDL 코드 생성을 자동화하는 HDL 코드 생성기를 제안한다.

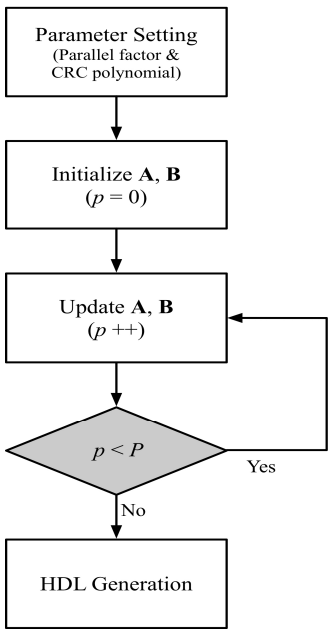


Fig. 5. HDL Code generation algorithm.
그림 5. HDL 코드 생성 알고리즘

3. CRC 하드웨어 HDL 코드 생성기

반도체 설계는 HDL 코드 작성, Function 시뮬레이션, 합성, Pre-Layout 시뮬레이션, 레이아웃, Post-Layout 시뮬레이션, DB 생성의 순서로 이루어진

다. 각 설계 단계는 많은 시간을 소요하고 결과에 따라 이전 단계를 반복적으로 수행해야하는 경우가 많다. 이때, 합성 가능하고 신뢰도 높은 HDL 코드를 자동 생성할 수 있다면 코드 작성 단계와 Function 시뮬레이션 단계를 건너뛰거나 시간 소요를 단축하여 개발비용을 줄일 수 있다. 특히 Look-Ahead CRC 하드웨어의 경우에는 병렬화 계수에 따라 그림 4의 기능 블록의 구성이 달라지므로 다른 병렬화 계수를 가지는 Look-Ahead CRC 하드웨어의 HDL 코드를 작성하는데 있어 많은 시간을 소요하게 된다. 이를 사전 컴퓨팅을 통해 계산하고 이에 맞는 HDL 코드를 자동 생성함으로써 반도체 설계 과정에서의 시간 소요를 단축한다. 그림 5에는 HDL 코드 생성기에서 사용자가 입력한 병렬화 계수와 CRC 다항식을 지원하는 HDL 코드를 생성하는 과정을 블록 다이어그램으로 나타냈다. 먼저 병렬화 계수와 CRC 다항식을 설정한다. 다음으로 입력받은 병렬화 계수 P의 A^P와 B^P를 구하기 위해서 A^0와 B^0를 설정한다. 이후 P 회만큼 식 (2)와 (3)을 수행하여 행렬을 업데이트하고 업데이트된 행렬을 기반으로 CRC 하드웨어를 HDL(.v) 파일로 생성한다.

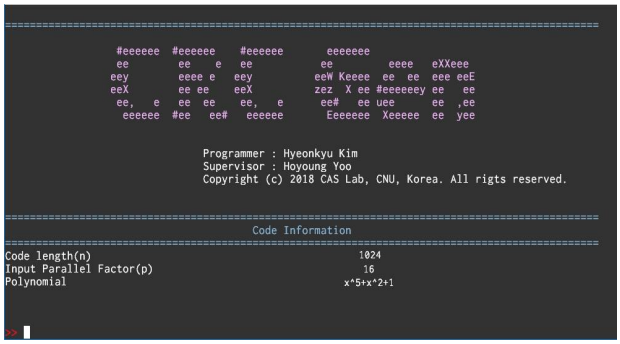


Fig. 6. CRC Hardware HDL code generator.
그림 6. CRC 하드웨어 HDL 코드 생성기

Table 1. Equivalent gate counts of CRC hardware.
표 1. CRC 하드웨어의 등가 2 입력 NAND 게이트 수

P	CRC-5	CRC-16	CRC-32
4	678.59	1670.60	3793.93
8	901.45	1973.42	4675.93
16	1310.60	2542.42	6157.17
32	1973.56	3585.86	9288.20

4. 실험 결과

본 절에서는 C언어를 기반으로 작성한 그림 6의 HDL 코드 생성기를 통해 4, 8, 16, 32의 병렬화 계수로 USB2.0에서 사용되는 CRC-5, CRC-16 표준 그리고 이더넷에서 사용되는 CRC-32용 하드웨어를 생성하였다. HDL 코드 생성기를 사용함으로써 많은 시간이 걸리는 HDL 코드 작성을 수 초 이내에 완료하였고, Synopsys Design Compiler를 통해 180 nm 공정에서 합성한 결과를 제시한다. 모든 CRC 하드웨어는 200 MHz로 합성되었으며, 2-입력 NAND 게이트로 정규화한 게이트 수를 표 1에 나타내었다. 하드웨어 복잡도는 CRC 다항식의 차수, 가중치, 병렬화 계수에 선형적으로 비례함을 확인하였다.

III. 결론

디지털 반도체 설계 과정은 여러 설계 단계와 각 단계에 위치하는 검증 단계로 인해서 많은 시간을 소요한다. 제안하는 CRC 하드웨어 HDL 코드 생성기를 통해 개발시간을 단축하고 신뢰도 높은 하드웨어의 확보가 가능할 것으로 기대된다.

References

[1] W. W. Peterson and D. T. Brown, “Cyclic Codes for Error Detection,” *Proceedings of the IRE*, vol.49, 228-235, 1961. DOI:10.1109/JRPROC.1961.287814

[2] G. Campobello *et al*. “Parallel CRC realization,” *IEEE Trans. on Computers* vol.52, 1312-1319, 2003. DOI:10.1109/TC.2003.1234528

[3] J. Jung *et al,* “Efficient Parallel Architecture for Linear Feedback Shift Regsiters,” *IEEE Trans. on Circuits and Sys. II - Express Brief*, vol.62, no.11, 2015. DOI:10.1109/TCSII.2015.2456294

[4] G. Albertango and R. Sisto, “Parallel CRC Generation,” *IEEE Micro*, vol.10, no.5, 1990. DOI:10.1109/40.60527

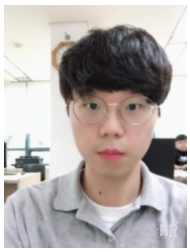
[5] E. Stavinov, “A Parallel CRC Generation Method,” *Circuits Celler Magzines for Computers*, 2010.

[6] T. B. Pei, C. Zukowski, “High-speed parallel CRC circuits in VLSI,” *IEEE Trans. on Communications*,

vol.40, 653-657, 1992. DOI:10.1109/26.141415
[7] M. Walma, "Pipelined Cyclic Redundancy Check (CRC) Calculation," *2007 16th International Conference on Computer Communications and Networks*, pp. 365-370, 2007. DOI:10.1109/ICCCN.2007.4317846
[8] C. Condo, M. Martina, G. Piccinini and G. Masera, "Variable Parallelism Cyclic Redundancy Check Circuit for 3GPP-LTE/LTE-Advanced," *IEEE Signal Processing Letters*, vol.21, no.11, pp.1380-1384, 2014. DOI:10.1109/LSP.2014.2334393
[9] C. Cheng and K. K. Parhi, "High-Speed Parallel CRC Implementation Based on Unfolding, Pipelining, and Retiming," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol.53, no.10, pp.1017-1021, 2006.
DOI:10.1109/TCSII.2006.882213
[10] M. Ayinala and K. K. Parhi, "High-Speed Parallel Architectures for Linear Feedback Shift Registers," *IEEE Transactions on Signal Processing*, vol.59, no.9, pp.4459-4469, 2011.
DOI:10.1109/TSP.2011.2159495

BIOGRAPHY

Hyeonkyu Kim (Student member)



2018 : BS degree in Electronics Engineering, Chungnam National University.
2018~ : M.S degree in Electronics Engineering, Chungnam National University.

Hoyoung Yoo (Member)

2010 : BS degree in Eletrical & Electronic Engineering, Yonsei University
2012 : MS degree in Electronics Engineering, KAIST
2016 : Ph.D. degree in Electronics Engineering, KAIST
2016~ : Assistant Professor, Chungnam National University