부호율 변경이 가능한 BCH Ecoder의 FPGA구현

제갈동* · 변건식*

*동아대학교

FPGA Implementation of BCH Encoder to change code rate

Dong Jegal* · Kun-sik Byon*

*Dong-A University

E-mail : zalove79@hanmail.net

요 약

본 논문에서는 블록 채널 부호 계열에서 다중 오류정정 능력을 갖는 BCH Encoder를 FPGA로 구현한 논문이다. 또한 부호율의 변경이 가능하게 하여 다양화 부호 율에 따른 부호를 생성할 수 있게 하였다.

본 논문에서는 FPGA 구현을 위해 Matlab을 이용하여 시뮬레이션을 하였고, 이를 HDL로 설계하고, 동시에 Xilinx사의 System Generator를 사용하여 구현하였고, Timming Analysis와 Resource estimation도 하였다.

ABSTRACT

The class of BCH codes is a large class of error correction codes. HDL implementation of BCH code generator to change code rate. and used System Generator, and implemented hardware to FPGA. Loaded bit stream to a FPGA board in order to verify this design to Hardware co-simulation from these results. Also, compared as investigated the maximum action frequency through timing analysis and resource of logic in order to evaluate performance of BCH code generator.

키워드

BCH Encoder, System generator

Ⅰ. 서 론

채널 부호(Channel code)는 데이터 전송에 있어서 채널 및 수신기로부터 야기되는 에러에 대해 어느 정도까지의 성능을 내도록 수정하는데 사용된다. 채널 부호 중 BCH 부호는 1959년에 Hocquenghem과 1960년 Bose와 Chaudhuri에 의해 발표되었으며, 단일 오류만을 정정할 수 있는 Hamming 부호를 확대시켜 무작위로 발생하는 산발적인 다중 오류에 대해 강력한 정정 능력을 갖는다. 본 논문에서는 BCH 부호 생성기를 임의의 모드 선택으로 (31,21)부호와 (31,16)부호를 선택할 수 있게 구현하였다. Xilinx사의 System Generator를 사용하여 FPGA를 구현하였으며 Taget device는 Xilinx사의 Spartan3 xc3s1000 fg676 -4를 사용하였다.

Ⅱ. BCH부호 생성 과정

Galios체 $GF(2^m)$의 원시원소(α)에서 얻어지는 2진 BCH 부호를 생각하자. $m_i(x)$를 α^i의 최소 다항식이라 하고 $c(x)=c_0+c_1x+c_2x^2+\cdots+c_{n-1}x^{n-1}$를 2원 기호 {0,1}를 계수로 하는 부호 다항식이라 하면 $\alpha,\alpha^2,\alpha^3,...,\alpha^{2t}$ 가 c(x)의 근일 때 c(x)는 근들에 대응되는 최소 다항식$m_1(x),m_2(x),\cdots,m_{2t}(x)$에 의해 나누어진다. 따라서 부호장이 $n=2^m-1$ 인 t중 오류 정정 BCH부호의 생성 다항식 g(x)는 의 그 근$\alpha,\alpha^2,\alpha^3,...,\alpha^{2t}$에 해당하는 최소 다항식들의 최소 공배 다항식이어야 한다. 즉 식(1)과 같다.

$$g(x)=LCM[m_1(x),m_2(x),...,m_{2t}(x)] \quad (1)$$

차수가 m=4 인 GF(2) 상의 원시 다항식 p(x)의 근이 α 일 때 Galios 확대체 $GF(2^4)$의 원소 $\{\alpha,\alpha^2,\alpha^3,...,\alpha^{14}\}$

에 대응되는 최소 다항식을 구해 보면 식(2)와 같다.

$$m_1(x)=m_2(x)=m_4(x)=m_8(x) \quad (2)$$
$$m_3(x)=m_6(x)=m_9(x)=m_{12}(x)$$
$$m_5(x)=m_{10}(x)$$
$$m_7(x)=m_{11}(x)=m_{13}(x)=m_{14}(x)$$

따라서$m_1(x),m_3(x),m_5(x),m_7(x)$ 4개만이 서로 다른 기약 다항식 임을 알 수 있다.

즉 α의 우 멱지수의 최소 다항식은 그에 선행되는 α의 임의의 기 멱지수의 최소 다항식과 같다. 이런 이유로 부호장 $n=2^m-1$인 t중 오류 정정 2진 BCH 부호의 생성 다항식은 일반적으로 식(3)와 같다.

$$g(x)=LCM[m_1(x),m_3(x),...,m_{2t-1}(x)] \quad (3)$$

이 식에서 각 mi(x)의 차수는 m 이하이며, 부호 다항식 c(x)가 $\alpha,\alpha^3,...,\alpha^{2t-1}$을 근으로 가질 때에 한해서 부호어가 된다. 각 최소 다항식의 차수가 m 이하이므로 g(x)의 차수는 mt 이하로 되며, 따라서 BCH부호의 검사 디지트 수는 $n-k\le mt$ 가 된다. GF(2) 상의 원시 다항식 p(x) = 1+x2+x5가 주어졌을 때 Galios체 $GF(2^5)$의 원소들은 다음 표 1과 같다.

표1. p(x)=1+x2+x5로 생성된 $GF(2^5)$

멱승 표현	다항식 표현	5-tuple 표현
0	0	00000
1	1	10000
α	α	01000
α^2	α^2	00100
α^3	α^3	00010
α^4	α^4	00001
α^5	$\alpha^5=1+\alpha^2$	10100
α^6	$\alpha^6=\alpha\cdot\alpha^5=\alpha(1+\alpha^2)=\alpha+\alpha^3$	01010
α^7	$\alpha^7=\alpha^5\cdot\alpha^1=\alpha^2+\alpha^4$	00101
α^8	$\alpha^8=\alpha(\alpha^2+\alpha^4)=\alpha^3+\alpha^5=1+\alpha^2+\alpha^3$	10110
α^9	$\alpha^9=\alpha(1+\alpha^2+\alpha^3)=\alpha+\alpha^3+\alpha^4$	01011
α^{10}	$\alpha^{10}=\alpha(\alpha+\alpha^3+\alpha^4)=\alpha^2+\alpha^4+1+\alpha^2=1+\alpha^4$	10001
α^{11}	$\alpha^{11}=\alpha(1+\alpha^4)=1+\alpha+\alpha^2$	11100
α^{12}	$\alpha^{12}=\alpha(1+\alpha+\alpha^2)=\alpha+\alpha^2+\alpha^3$	01110
α^{13}	$\alpha^{13}=\alpha(\alpha+\alpha^2+\alpha^3)=\alpha^2+\alpha^3+\alpha^4$	00111
α^{14}	$\alpha^{14}=\alpha(\alpha^2+\alpha^3+\alpha^4)=1+\alpha^2+\alpha^3+\alpha^4$	10111
α^{15}	$\alpha^{15}=\alpha(1+\alpha^2+\alpha^3+\alpha^4)=1+\alpha+\alpha^2+\alpha^3+\alpha^4$	11111
α^{16}	$\alpha^{16}=\alpha(1+\alpha+\alpha^2+\alpha^3+\alpha^4)=1+\alpha+\alpha^3+\alpha^4$	11011
α^{17}	$\alpha^{17}=\alpha(1+\alpha+\alpha^3+\alpha^4)=1+\alpha+\alpha^4$	11001
α^{18}	$\alpha^{18}=\alpha(1+\alpha+\alpha^4)=1+\alpha$	11000
α^{19}	$\alpha^{19}=\alpha(1+\alpha)=\alpha+\alpha^2$	01100
α^{20}	$\alpha^{20}=\alpha(\alpha+\alpha^2)=\alpha^2+\alpha^3$	00110
α^{21}	$\alpha^{21}=\alpha(\alpha^2+\alpha^3)=\alpha^3+\alpha^4$	00011
α^{22}	$\alpha^{22}=\alpha(\alpha^3+\alpha^4)=1+\alpha^2+\alpha^4$	10101
α^{23}	$\alpha^{23}=\alpha(1+\alpha^2+\alpha^4)=1+\alpha+\alpha^2+\alpha^3$	11110
α^{24}	$\alpha^{24}=\alpha(1+\alpha+\alpha^2+\alpha^3)=\alpha+\alpha^2+\alpha^3+\alpha^4$	01111
α^{25}	$\alpha^{25}=\alpha(\alpha+\alpha^2+\alpha^3+\alpha^4)=1+\alpha^3+\alpha^4$	10011
α^{26}	$\alpha^{26}=\alpha(1+\alpha^3+\alpha^4)=1+\alpha+\alpha^2+\alpha^4$	11101
α^{27}	$\alpha^{27}=\alpha(1+\alpha+\alpha^2+\alpha^4)=1+\alpha+\alpha^3$	11010
α^{28}	$\alpha^{28}=\alpha(1+\alpha+\alpha^3)=\alpha+\alpha^2+\alpha^4$	01101
α^{29}	$\alpha^{29}=\alpha(\alpha+\alpha^2+\alpha^4)=1+\alpha^3$	10010
α^{30}	$\alpha^{30}=\alpha(1+\alpha^3)=\alpha+\alpha^4$	01001

여기서 최소다항식을 구해보면 식(4)와 같다.

$$m_1(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^4) \quad (4)$$
$$(x+\alpha^8)(x+\alpha^{16})=1+x^2+x^5$$
$$m_3(x)=(x+\alpha^3)(x+\alpha^6)$$
$$(x+\alpha^{12})(x+\alpha^{24})(x+\alpha^{17})$$
$$=1+x^2+x^3+x^4+x^5$$
$$m_5(x)=(x+\alpha^5)(x+\alpha^{10})(x+\alpha^{20})$$
$$(x+\alpha^9)(x+\alpha^{18})$$
$$=1+x+x^2+x^4+x^5$$

식(5)은 t=2일 때 2중 오류 정정 BCH부호의 생성 다항식이며 이 부호는 $d_{\min}\ge 2t+1=5$ 인 (31,21) 순회 부호이다.

$$g(x)=m_1(x)m_3(x)$$
$$=(1+x^2+x^5)$$
$$(1+x^2+x^3+x^4+x^5)$$
$$=1+x^3+x^5+x^6+x^8+x^9+x^{10} \quad (5)$$

식(6)은 t=3일 때 3중 오류 정정 BCH부호의 생성 다항식이다. 즉 $d_{\min}\ge 2t+1=7$ 인 3중 오류 정정 (31,16) 순회 부호이다.

$$g(x)=m_1(x)m_3(x)m_5(x) \quad (6)$$
$$=(1+x^3+x^5+x^6+$$
$$x^8+x^9+x^{10})$$
$$(1+x+x^2+x^4+x^5)$$
$$=1+x+x^2+x^3+x^5+x^7$$
$$+x^8+x^9+x^{10}+x^{11}+x^{15}$$

따라서 위의 생성 다항식을 이용하여 BCH 부호 생성기를 하드웨어로 구현한다.

III. BCH부호 생성기의 HDL 구현

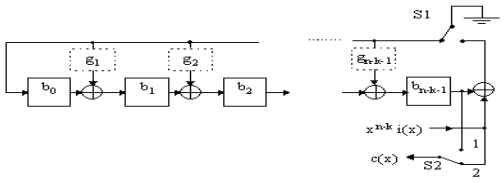

그림 1. (n,k) BCH 부호 생성기

(n,k) BCH 부호 생성기는 생성 다항식 g(x)에 의해 쉬프트 되는 단 수가 결정되며 위의 그림1 은 일반화된 하드웨어의 구조이다. s2가 2에 연결되어 c(x)의 코드를 입력 받고, s1은 피드백 라인에 연결되어 동시에 생성 다항식 블록인 쉬프트 레지스터를 통과하게 된다. c(x)의 코드가 완전히 입력되는 순간 s2는 1에 연결되어 패리티 비트를 c(x)에 추가하게 되어 BCH 부호가 생성된다. 이 때에 s1은 GND에 연결되어 피드백 라인을 0으로 유지시켜 패리티 비트가 완전히 빠져 나갈 때까지 패리티 비트에 변경 없이 유지 시킨다. 이 구조에 모드 선택에 따른 부호율의 변경이 가능 하게하여 생성 다항식을 선택할 수 있게 하였다. 먼저 HDL로 구현하여 검증하였으며, 그 소스 중 일부는 그림 3과 같다. 소스에서 입력되는 코드는 그대로 1bit 씩 차례로 출력되며, 동시에 생성 다항식 g(x)의 쉬프트 레지스터를 차례로 통과하고, 다시 피드백 되는 신호와 XOR 연산하여 결과적으로 입력되는 코드를 생성 다항식으로 나누는 연산을 하게 된다. 입력되는 코드가 온전히 출력되고 난 후에 MUX의 select로 쉬프트 레지스터의 출력부분에서 1bit 씩 차례로 출력되며, 이는 원래 입력되는 코드에 패리티 부분으로 더해지게 된다. 이로써 원래 코드는 BCH 코드로 부호화 되어 출력되게 된다. mode0,1을

선택하여, 부호 율 (31,21)(mode=0) 과 부호 율 (31,16)(mode=1)을 선택 할 수 있게 하였다. mode의 선택에 따라 부호 율이 변경 되어 정상 출력됨을 확인 하였다. 이의 Simulation 파형은 그림 2와 같다.

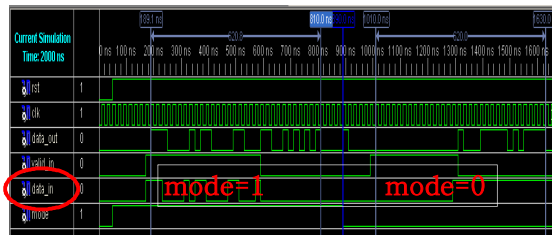

그림 2. HDL코드의 SIMULATION

```
module bch_enc_multiver1(
  clk, rst, valid_in,
  data_in,
  data_out,
  mode
    );

input clk,rst,valid_in;
input data_in;
input mode;
output reg data_out;

reg lfsr1, lfsr2, lfsr3, lfsr4, lfsr5;
reg lfsr6, lfsr7, lfsr8, lfsr9, lfsr10;
reg lfsr11, lfsr12, lfsr13, lfsr14, lfsr15;
wire fdback;

assign fdback = valid_in && mode == 1'b1 ? lfsr10 ^ data_in :
          valid_in && mode == 1'b0 ? lfsr15 ^ data_in :
          1'b0;


always @(posedge clk or negedge rst) begin
 if(!rst)
   lfsr1 <= 0;
 else begin
   lfsr1 <= fdback;
 end
end

always @(posedge clk or negedge rst) begin
 if(!rst)
   lfsr2 <= 0;
 else begin
  case(mode)
    1'b1 : lfsr2 <= lfsr1;
    1'b0 : lfsr2 <= fdback ^ lfsr1;
  default: ;
  endcase
 end
end

always @(posedge clk or negedge rst) begin
 if(!rst)
   lfsr13 <= 0;
 else
   lfsr13 <= lfsr12;
end

always @(posedge clk or negedge rst) begin
 if(!rst)
   lfsr14 <= 0;
 else
   lfsr14 <= lfsr13;
end

always @(posedge clk or negedge rst) begin
 if(!rst)
   lfsr15 <= 0;
 else
   lfsr15 <= lfsr14;
end


always@(posedge clk or negedge rst)
begin
   if(!rst)
     data_out <= 0;
  else begin
     if(valid_in == 1'b0) begin
       case(mode)
         1'b1 : data_out <= lfsr10;
         1'b0 : data_out <= lfsr15;
       default: ;
       endcase
     end
     else
       data_out <= data_in;
  end
end


endmodule
```

그림 3. BCH 부호 생성기 HDL구현

VI. System Generator 구현 및 Timming & Resource량 분석

HDL로 구현하고, Simulation에서 검증한 모델을 기반으로 이를 xilinx사의 System generator로 구현하였다. 전체적인 설계는 그림 4와 같다. 이 때 target board는 spartan-3를 사용하였으며 샘플링 클럭은 50MHz를 사용하였다.

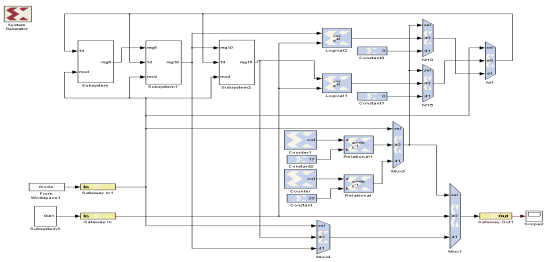

그림 4. System generator구현

코드를 입력 받는 블록은 그림 5와 같다. 전체 입력 블록 중에 일부분을 simulation 한 것인데, 여기서 입력 bit는 병렬로 저장되어 차례대로 직렬 쉬프트 되어 출력된다. 입력이 완전히 생성 다항식 블록 에 입력되는 순간에 온전한 패리티를 생성하는데, 이 패리티가 1bit 씩 모두 출력되는 시간 동안 입력 코드를 저장하는 레지스터는 소비되는 시간이 길어지게 되며, 이를 해소 하고자 병렬로 입력 받아서 직렬로 쉬프트 하는 과정을 통해 대기 시간을 감소 시켰다. 그림 6은 생성 다항식 블록의 일부분이며, 입력되는 코드를 생성 다항식으로 나누어 주는 연산을 하면서 패리티를 생성하게 된다. mode의 0,1에 따라 부호 율 (31,21)과 (31,16)의 선택이 가능하게 한다. 그림 7에서 보는 바와 같이 총 31bit/clk 이 소요 되며 그 결과로써 mode선택에 따른 부호 율이 변경되어 정상적으로 출력됨을 확인 할 수 있다.

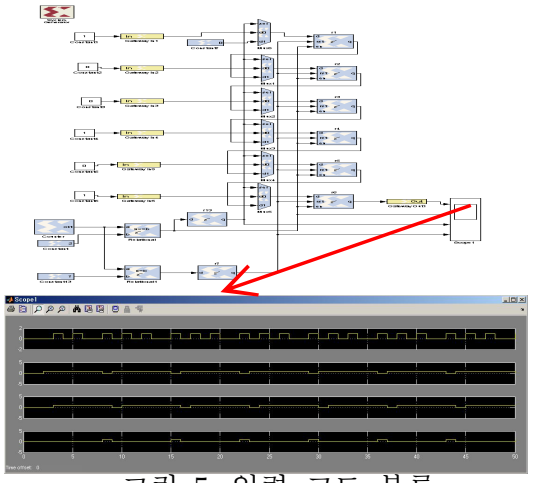

그림 5. 입력 코드 블록

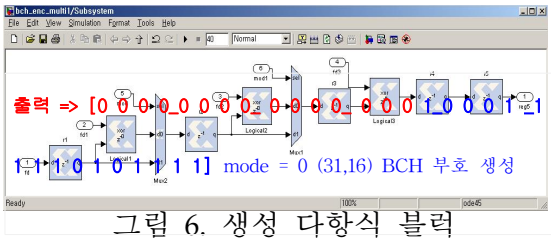

그림 6. 생성 다항식 블럭

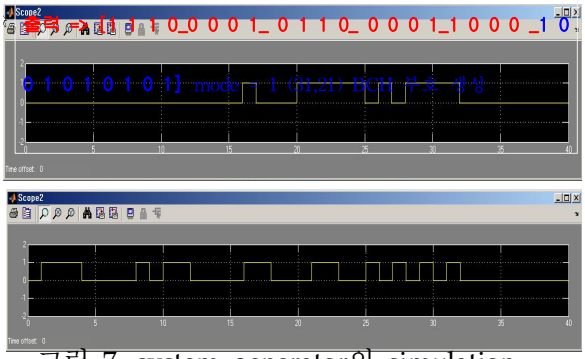

그림 7. system generator의 simulation

Ⅴ. Hardware co-simulation

설계한 BCH 부호 생성기를 spartan-3 board에 연결하여 hardware co-simulation을 하기 위해, 먼저 System generator 토큰의 compilation을 등록한 spartan-3 보드로 설정한 후 JTAG 케이블을 이용하여 하드웨어에 연결한다. 연결하여 동작 시키면, hardware co-simulation 블록이 그림 8과 같이 생성되며, 이는 설계한 블록을 실제 합성하여 칩에 다운로드하고, hardware in the loop를 형성하여 온 칩에서의 동작을 검증하게 된다.

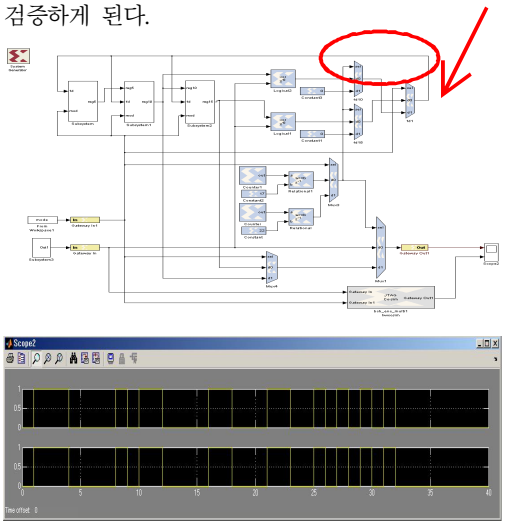

그림 8. Hardware co-simulation

hardware-co-simulation결과,system generator의 출력과 일치함을 볼 수 있다.

```
Device Utilization Summary:

   Number of BUFGMUXs                        1 out of 8      12%
   Number of External IOBs                   4 out of 391     1%
      Number of LOCed IOBs                   0 out of 4       0%

   Number of Slices                         21 out of 7680    1%
      Number of SLICEMs                      0 out of 3840    0%
```

```
Timing constraint: TS_clk_799d4c18 = PERIOD TIMEGRP "clk_799d4c18" 20 ns HIGH 50%;
2' 3 path's analyzed, 74 endpoints analyzed, 0 failing endpoints
0 timing errors detected. (0 setup errors, 0 hold errors)
Minimum period is 5.351ns.
```

그림 9. Timming & Resource 분석

그림 9에서 Resource량과 50MHz CLK에서Timming error없이 동작함을 알 수 있다.

Ⅵ. 결 론

본 논문에서는 mode선택에 따른 부호율의 변경이 가능한 BCH 부호 생성기를 FPGA로 구현하였다. BCH부호의 생성 다항식을 하드웨어적으로 선택하여 이를 통해 부호 율(31,21)와 (31,16)의 변경을 가능하게 하고, 이를 위해 HDL로 설계하여 검증 후, Xilinx사의 Sytem generator로 구현하여 hardware co-simulation과 Timming Analzer 와 Resource량을 평가하였다.

참고문헌

[1] 이문호, "갈루이스필드.리드솔로몬.비터비.터보 부호기의 설계", 도서출판 영일,2000
[2] Hank Wallace, "Error Detection and Correction Using the BCH Code", 2001
[3] Rhee, Man Young, " Error correcting coding theory", McGraw-Hill Publishing Company
[4] http://www.xilinx.com
[5] Xilinx System generator for DSP user's guide