Final Exam (Embedded System)

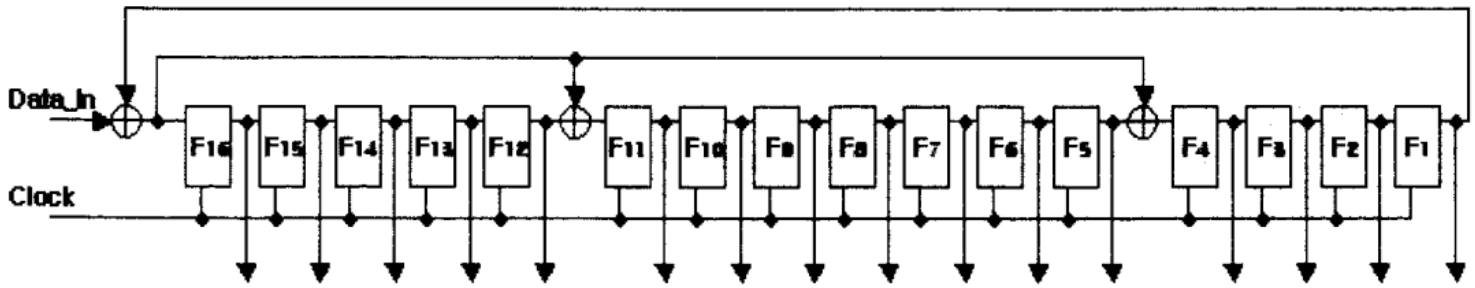
이진서

Dept. of Computer Science

Sookmyung Women’s University

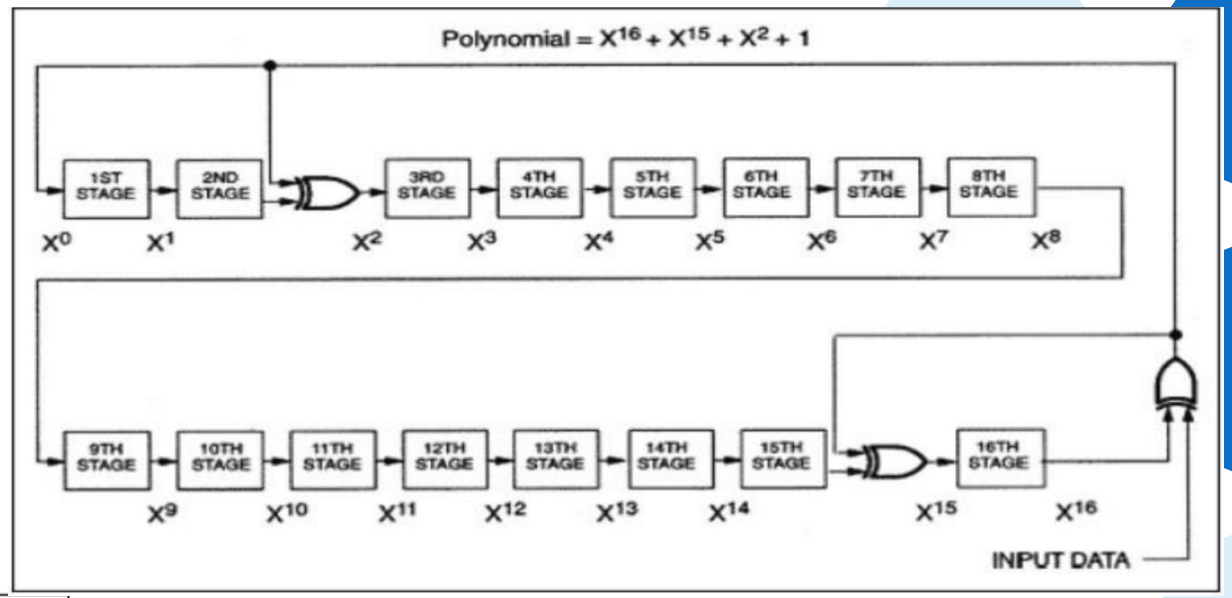
About My Selected Algorithm

- Explain what algorithm I select
  - ❖CRC
    - ❖A Cyclic Redundancy Check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.
    - ❖In this project, I selected CRC-16.
- Explain why I select the algorithm
  - ❖ CRC : 네트워크 보안 시간에 배운 후, 에러 검출은 매우 중요한 단계라고 생각했고 알고리즘으로 구현해보고 싶었다
  - ❖CRC-16 : 보통 CRC-16과 CRC-32 두가지가 사용되는데 One chip micro-processor에서는 CRC-16이 보통 사용된다고 하였기 때문이다.



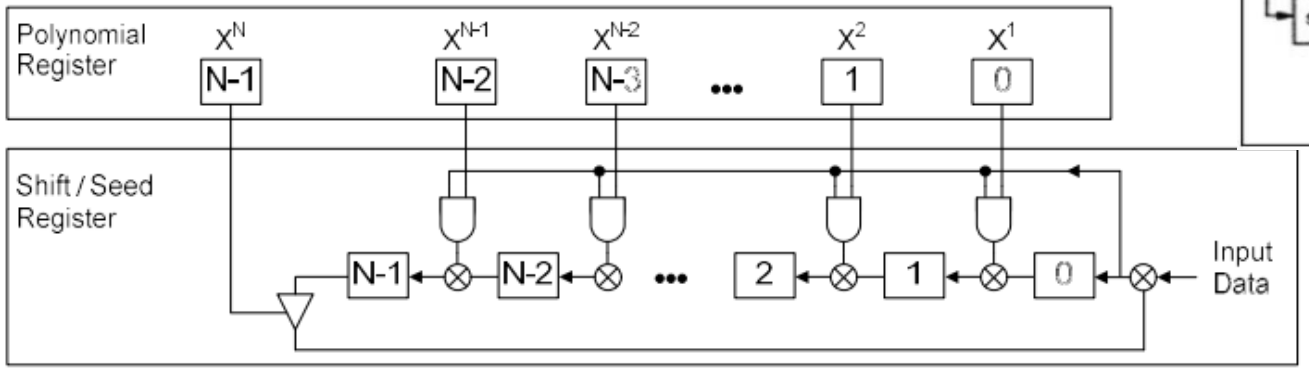
EQUATION 1: THE CRC-16 POLYNOMIAL

$$P(x) = x^{16}+x^{15}+x^{2}+1$$

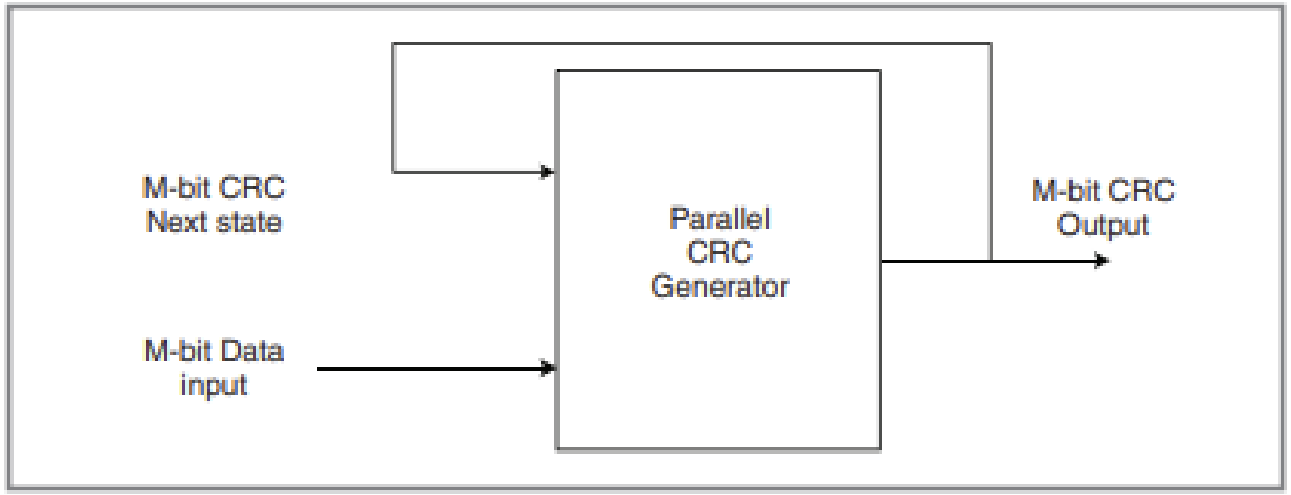


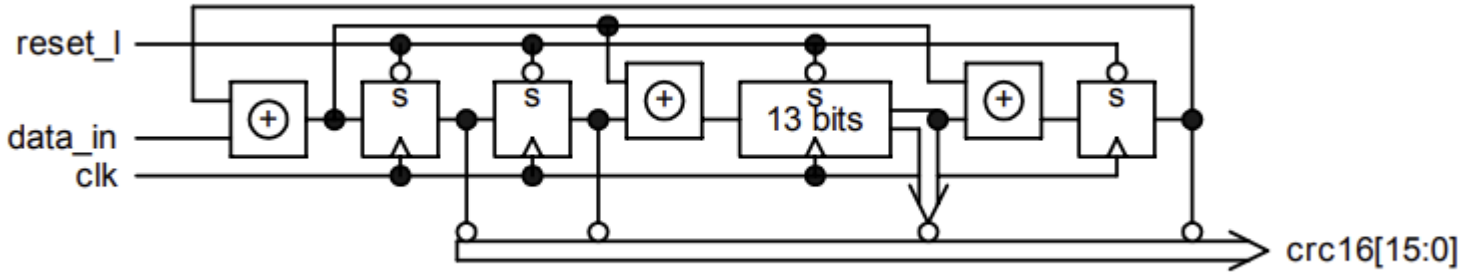
Block Diagram and Configuration

N = 16



Block Diagram and HLSM for RTL Design of the Algorithm





$$
\begin{array}{r|l}
 & x^{22} + x^{19} + x^{18} + x^{15} + x^{13} + x^{11} + x^{9} + x^{6} + x^{4} + x^{3} + x^{2} + 1 \\
\hline
x^{16}+x^{12}+x^{5}+1 & x^{38} + x^{35} + x^{30} + x^{29} + x^{27} + x^{24} + x^{21} + x^{16} \\
 & -\ x^{38} + x^{34} + x^{27} + x^{22} \\
\hline
 & +x^{35} + x^{34} + x^{30} + x^{29} + x^{24} + x^{22} + x^{21} + x^{16} \\
 & -\ x^{35} + x^{31} + x^{24} + x^{19} \\
\hline
 & +x^{34} + x^{31} + x^{30} + x^{29} + x^{22} + x^{21} + x^{19} + x^{16} \\
 & -\ x^{34} + x^{30} + x^{23} + x^{18} \\
\hline
 & x^{31} + x^{29} + x^{23} + x^{22} + x^{21} + x^{19} + x^{18} + x^{16} \\
 & -\ x^{31} + x^{27} + x^{20} + x^{15} \\
\hline
 & x^{29} + x^{27} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{16} + x^{15} \\
 & -\ x^{29} + x^{25} + x^{18} + x^{13} \\
\hline
 & x^{27} + x^{25} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{16} + x^{15} + x^{13} \\
 & x^{27} + x^{23} + x^{16} + x^{11} \\
\hline
 & x^{25} + x^{22} + x^{21} + x^{20} + x^{19} + x^{15} + x^{13} + x^{11} \\
 & x^{25} + x^{21} + x^{14} + x^{9} \\
\hline
 & x^{22} + x^{20} + x^{19} + x^{15} + x^{14} + x^{13} + x^{11} + x^{9} \\
 & x^{22} + x^{18} + x^{11} + x^{6} \\
\hline
 & x^{20} + x^{19} + x^{18} + x^{15} + x^{14} + x^{13} + x^{9} + x^{6} \\
 & x^{20} + x^{16} + x^{9} + x^{4} \\
\hline
 & x^{19} + x^{18} + x^{16} + x^{15} + x^{14} + x^{13} + x^{6} + x^{4} \\
 & x^{19} + x^{15} + x^{8} + x^{3} \\
\hline
 & x^{18} + x^{16} + x^{14} + x^{13} + x^{8} + x^{6} + x^{4} + x^{3} \\
 & x^{18} + x^{14} + x^{7} + x^{2} \\
\hline
 & x^{16} + x^{13} + x^{8} + x^{7} + x^{6} + x^{4} + x^{3} + x^{2} \\
 & x^{16} + x^{12} + x^{5} + 1 \\
\hline
 & x^{13} + x^{12} + x^{8} + x^{7} + x^{6} + x^{5} + x^{4} + x^{3} + x^{2} + 1
\end{array}
$$

0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1

One-Procedure RTL Description of the Algorithm in Verilog

C:/Users/SM-PC/Desktop/finalcrc/CRC_16_parallel.v (/CRC_16_parallel_test/u1) - Default

```
Ln#
1  module CRC_16_parallel(clk,rst,load,d_finish,crc_in,crc_out);
2
3  input clk;
4  input rst;
5  input load;
6  input d_finish;
7  input [7:0] crc_in;
8  output [7:0] crc_out;
9  reg [7:0] crc_out;
10 reg [15:0] crc_reg;
11 reg [1:0] count;
12 reg [1:0] state;
13 wire [15:0] next_crc_reg; //
14 parameter idle = 2'b00; //
15 parameter compute = 2'b01;//
16 parameter finish = 2'b10; //
17 //
18
19 assign next_crc_reg[0] = (^crc_in[7:0]) ^ (^crc_reg[15:8]);
20 assign next_crc_reg[1] = (^crc_in[6:0]) ^ (^crc_reg[15:9]);
21 assign next_crc_reg[2] = crc_in[7] ^ crc_in[6] ^ crc_reg[9] ^ crc_reg[8];
22 assign next_crc_reg[3] = crc_in[6] ^ crc_in[5] ^ crc_reg[10] ^
23 crc_reg[9];
24 assign next_crc_reg[4] = crc_in[5] ^ crc_in[4] ^ crc_reg[11] ^
25 crc_reg[10];
26 assign next_crc_reg[5] = crc_in[4] ^ crc_in[3] ^ crc_reg[12] ^
```

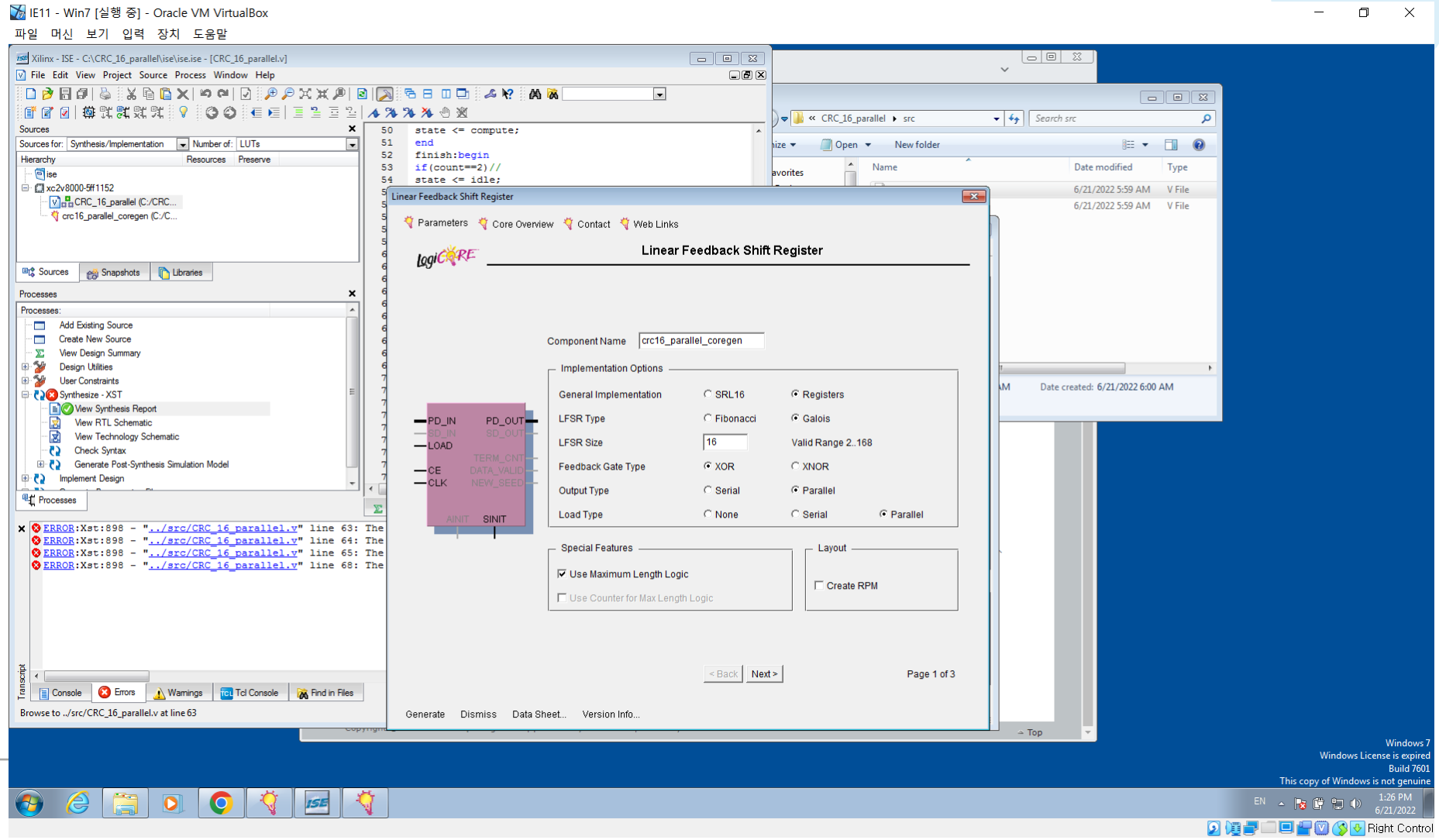
```
27 crc_reg[11];
28 assign next_crc_reg[6] = crc_in[3] ^ crc_in[2] ^ crc_reg[13] ^
29 crc_reg[12];
30 assign next_crc_reg[7] = crc_in[2] ^ crc_in[1] ^ crc_reg[14] ^
31 crc_reg[13];
32 assign next_crc_reg[8] = crc_in[1] ^ crc_in[0] ^ crc_reg[15] ^ crc_reg[14]
33 ^ crc_reg[0];
34 assign next_crc_reg[9] = crc_in[0] ^ crc_reg[15] ^ crc_reg[1];
35 assign next_crc_reg[14:10] = crc_reg[6:2];
36 assign next_crc_reg[15] = (^crc_in[7:0]) ^ (^crc_reg[15:7]);
37 always@(posedge clk) //
38 begin
39 case(state) //
40  idle:begin //
41  if(load) //
42  state <= compute;
43  else
44  state <= idle;
45  end
46  compute:begin
47  if(d_finish)//
48  state <= finish;
49  else
50  state <= compute;
51  end
```

One-Procedure RTL Description of the Algorithm in Verilog

C:/Users/SM-PC/Desktop/finalcrc/CRC_16_parallel.v (/CRC_16_parallel_test/u1) - Default *

```
Ln#
52   finish:begin
53   if(count==2)//
54   state <= idle;
55   else
56   state <= finish;
57   end
58 endcase
59 end
60 always@(posedge clk or negedge rst)//
61   if(rst)
62   begin
63   crc_reg[15:0] <= 16'b0000_0000_0000_0000;//
64   state <= idle;
65   count <= 2'b00;
66   end
67   else
68   case(state)
69   idle:begin //
70   crc_reg[15:0] <= 16'b0000_0000_0000_0000;
71   end
72   compute:begin //
73   crc_reg[15:0]<= next_crc_reg[15:0];
74   crc_out[7:0] <= crc_in[7:0];
75   end
76   finish:begin //
77   crc_reg[15:0] <= {crc_reg[7:0],8'b0000_0000};
78   crc_out[7:0] <= crc_reg[15:8];
79   end
80   endcase
81 endmodule
82
```

Specification of Components from Xilinx CORE Generator



IE11 - Win7 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

Xilinx - ISE - C:\CRC_16_parallel\ise\ise.ise - [CRC_16_parallel.v]

File Edit View Project Source Process Window Help

Sources

Sources for: Synthesis/Implementation Number of: LUTs

Hierarchy | Resources | Preserve

- ise
- xc2v8000-5ff1152
  - CRC_16_parallel (C:/CRC...
  - crc16_parallel_coregen (C:/C...

Sources | Snapshots | Libraries

Processes

- Add Existing Source
- Create New Source
- View Design Summary
- Design Utilities
- User Constraints
- Synthesize - XST
  - View Synthesis Report
  - View RTL Schematic
  - View Technology Schematic
  - Check Syntax
  - Generate Post-Synthesis Simulation Model
- Implement Design

```
50  state <= compute;
51  end
52  finish:begin
53  if(count==2)//
54  state <= idle;
```

```
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 63: The
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 64: The
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 65: The
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 68: The
```

Console | Errors | Warnings | Tcl Console | Find in Files

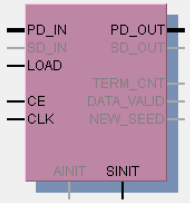
Browse to ../src/CRC_16_parallel.v at line 63

Linear Feedback Shift Register

Parameters | Core Overview | Contact | Web Links

LogiCORE

Linear Feedback Shift Register



Optional Ports

- [x] Asynchronous Reset
  - Asynchronous Reset Value: FFFF
- [ ] Synchronous Reset
  - Synchronous Reset Value: FFFF

Optional Pins

- [x] Clock Enable
- [ ] Data Valid Output
- [ ] New Seed Output
- [ ] Terminal Count Output

< Back | Next > | Page 2 of 3

Generate | Dismiss | Data Sheet... | Version Info...

CRC_16_parallel ▸ src Search src

Open | New folder

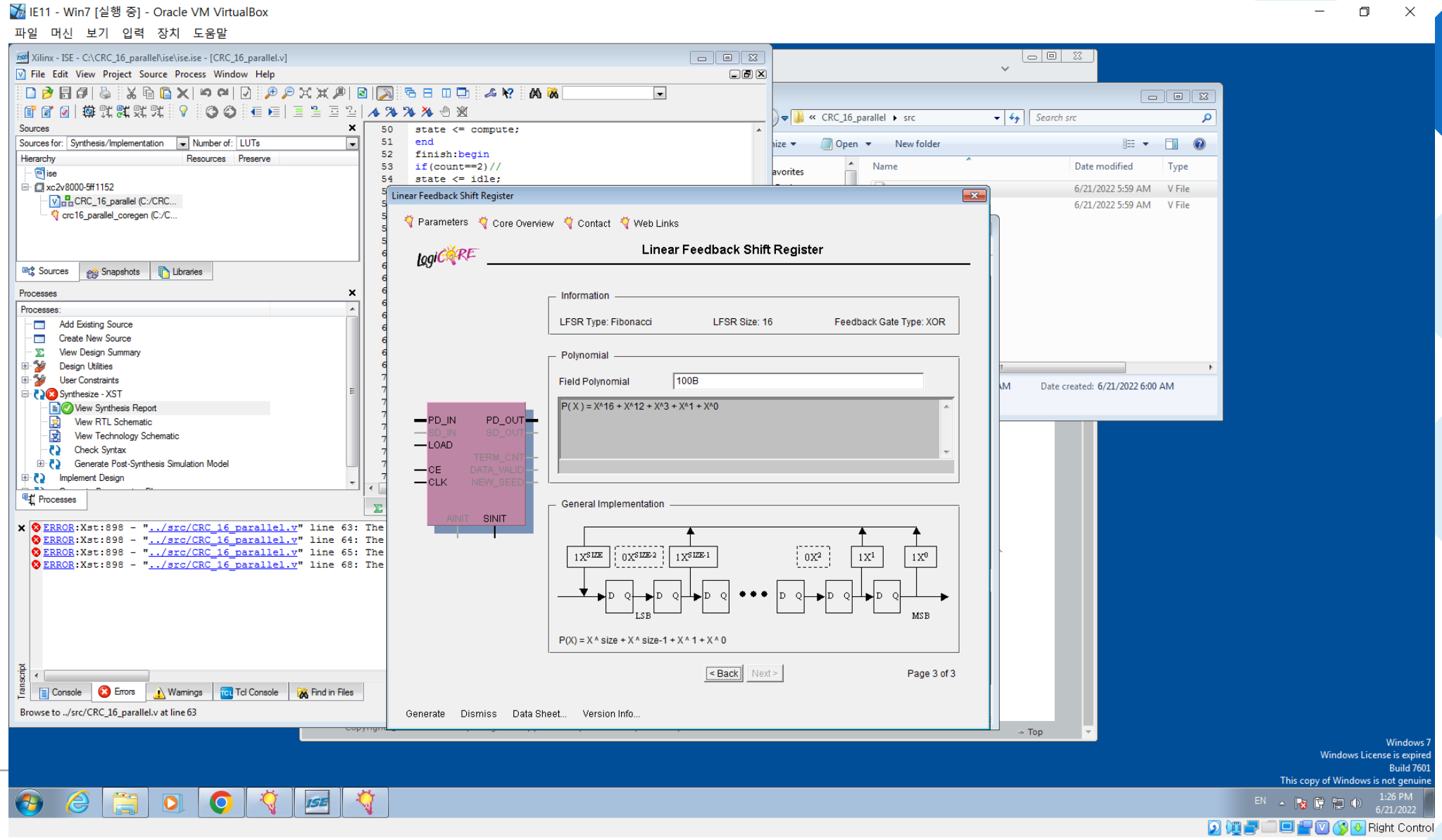
Name	Date modified	Type
	6/21/2022 5:59 AM	V File
	6/21/2022 5:59 AM	V File

Date created: 6/21/2022 6:00 AM

Windows 7
Windows License is expired
Build 7601
This copy of Windows is not genuine

EN 1:26 PM 6/21/2022

Right Control


IE11 - Win7 [실행 중] - Oracle VM VirtualBox
파일 머신 보기 입력 장치 도움말
Xilinx - ISE - C:\CRC_16_parallel\ise\ise.ise - [CRC_16_parallel.v]
File Edit View Project Source Process Window Help
Sources
Sources for: Synthesis/Implementation
Number of: LUTs
Hierarchy
Resources
Preserve
ise
xc2v8000-5ff1152
CRC_16_parallel (C:/CRC...
crc16_parallel_coregen (C:/C...
Sources
Snapshots
Libraries
Processes
Processes:
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
View Synthesis Report
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Processes
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 63: The
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 64: The
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 65: The
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 68: The
Console
Errors
Warnings
Tcl Console
Find in Files
Browse to ../src/CRC_16_parallel.v at line 63
Transcript
50 state <= compute;
51 end
52 finish:begin
53 if(count==2)//
54 state <= idle;
Linear Feedback Shift Register
Parameters
Core Overview
Contact
Web Links
LogiCORE
Linear Feedback Shift Register
Information
LFSR Type: Fibonacci
LFSR Size: 16
Feedback Gate Type: XOR
Polynomial
Field Polynomial
100B
P(X) = X^16 + X^12 + X^3 + X^1 + X^0
PD_IN
PD_OUT
SD_IN
SD_OUT
LOAD
TERM_CNT
CE
DATA_VALID
CLK
NEW_SEED
AINIT
SINIT
General Implementation
LSB
MSB
P(X) = X ^ size + X ^ size-1 + X ^ 1 + X ^ 0
< Back
Next >
Page 3 of 3
Generate
Dismiss
Data Sheet...
Version Info...
CRC_16_parallel ▸ src
Search src
Open
New folder
Name
Date modified
Type
6/21/2022 5:59 AM
V File
Date created: 6/21/2022 6:00 AM
Top
Windows 7
Windows License is expired
Build 7601
This copy of Windows is not genuine
EN
1:26 PM
6/21/2022
Right Control

IE11 - Win7 [실행 중] - Oracle VM VirtualBox

파일 머신 보기 입력 장치 도움말

Xilinx - ISE - C:\CRC_16_parallel\ise\ise.ise - [CRC_16_parallel.v]

File Edit View Project Source Process Window Help

```
50   state <= compute;
51   end
52   finish:begin
53   if(count==2)//
54   state <= idle;
55   else
56   state <= finish;
57   end
58  endcase
59  end
60  always@(posedge clk or negedge rst)//
61   if(rst)
62   begin
63   crc_reg[15:0] <= 16'b0000_0000_0000_0000;//
64   state <= idle;
65   count <= 2'b00;
66   end
67   else
68   case(state)
69   idle:begin //
70   crc_reg[15:0] <= 16'b0000_0000_0000_0000;
71   end
72   compute:begin //
73   crc_reg[15:0]<= next_crc_reg[15:0];
74   crc_out[7:0] <= crc_in[7:0];
75   end
76   finish:begin //
77   crc_reg[15:0] <= {crc_reg[7:0],8'b0000_0000};
78   crc_out[7:0] <= crc_reg[15:8];
```

```
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 63: The reset or set test condition for <crc_reg> is incompatible
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 64: The reset or set test condition for <state> is incompatible wi
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 65: The reset or set test condition for <count> is incompatible wi
ERROR:Xst:898 - "../src/CRC_16_parallel.v" line 68: The reset or set test condition for <crc_out> is incompatible
```

Xilinx CORE Generator

Date created: 6/21/2022 6:00 AM

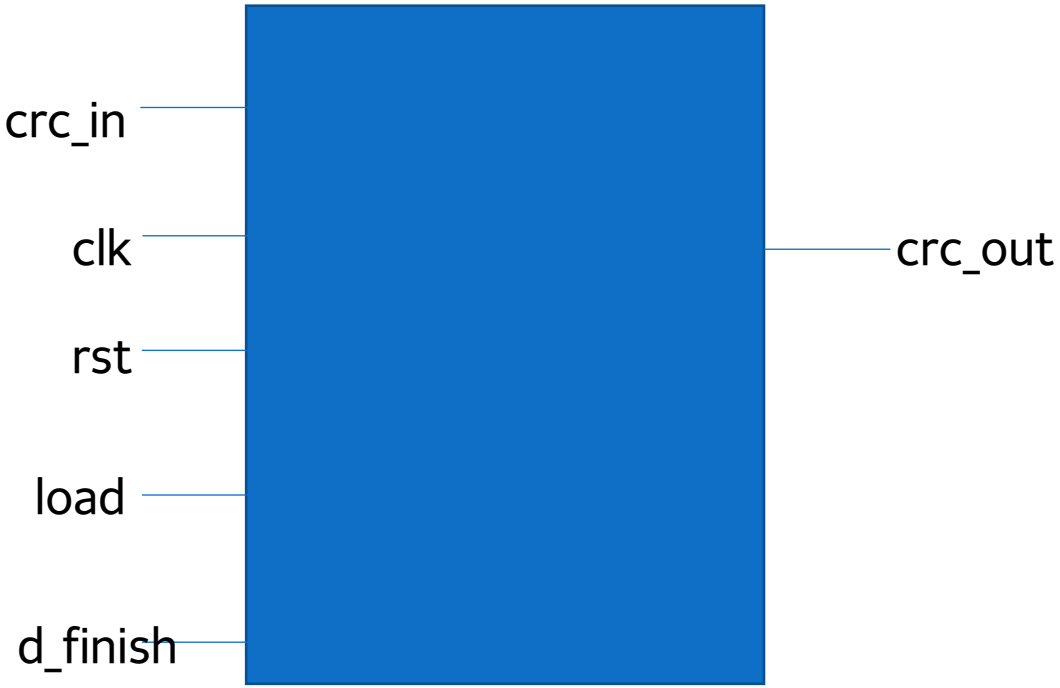
Windows 7
Windows License is expired
Build 7601
This copy of Windows is not genuine

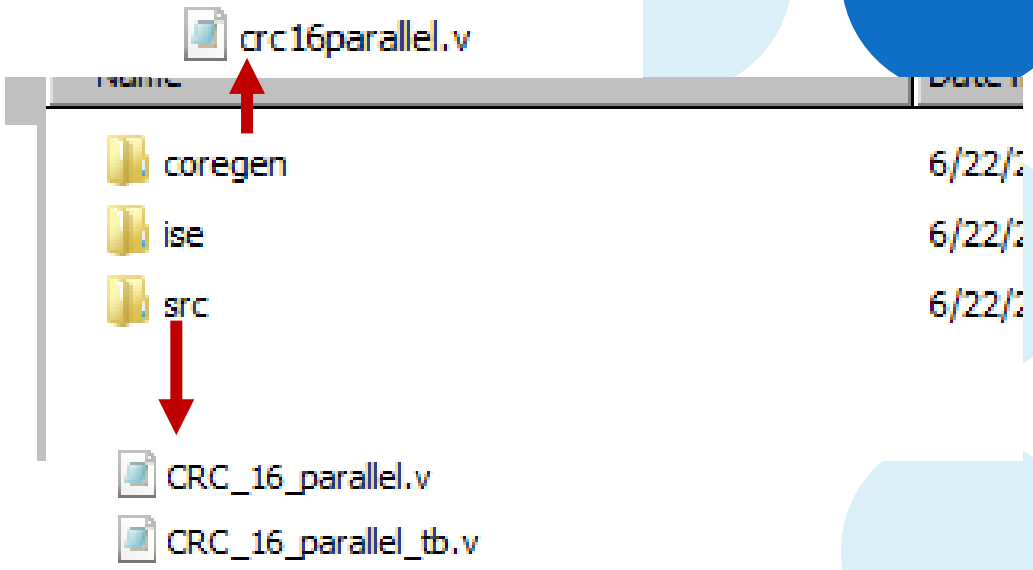
Specification of Components from Xilinx CORE Generator

C:/Users/SM-PC/Desktop/crc16/crc16.v (/CRC_TEST/ul) - Default

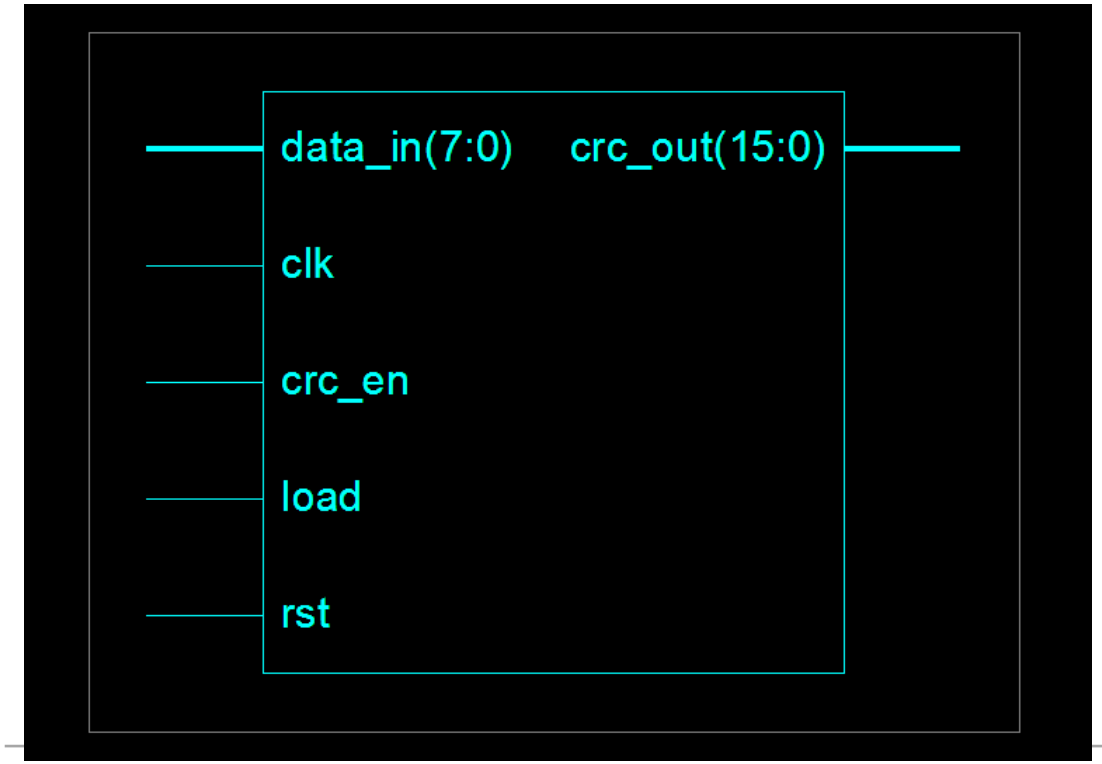
```
module CRC(
  input [7:0] data_in,
  input load,
  input crc_en,
  output [15:0] crc_out,
  input rst,
  input clk);

  reg [15:0] lfsr_q,lfsr_c;

  assign crc_out = lfsr_q;

  always @(*) begin
    $display(data_in);
    lfsr_c[0] = lfsr_q[8] ^ lfsr_q[9] ^ lfsr_q[10] ^ lfsr_q[11] ^ lfsr_q[12] ^ lfsr_q[13] ^ lfsr_q[14]
    lfsr_c[1] = lfsr_q[9] ^ lfsr_q[10] ^ lfsr_q[11] ^ lfsr_q[12] ^ lfsr_q[13] ^ lfsr_q[14] ^ lfsr_q[15]
    lfsr_c[2] = lfsr_q[8] ^ lfsr_q[9] ^ data_in[0] ^ data_in[1];
    lfsr_c[3] = lfsr_q[9] ^ lfsr_q[10] ^ data_in[1] ^ data_in[2];
    lfsr_c[4] = lfsr_q[10] ^ lfsr_q[11] ^ data_in[2] ^ data_in[3];
    lfsr_c[5] = lfsr_q[11] ^ lfsr_q[12] ^ data_in[3] ^ data_in[4];
    lfsr_c[6] = lfsr_q[12] ^ lfsr_q[13] ^ data_in[4] ^ data_in[5];
    lfsr_c[7] = lfsr_q[13] ^ lfsr_q[14] ^ data_in[5] ^ data_in[6];
    lfsr_c[8] = lfsr_q[0] ^ lfsr_q[14] ^ lfsr_q[15] ^ data_in[6] ^ data_in[7];
    lfsr_c[9] = lfsr_q[1] ^ lfsr_q[15] ^ data_in[7];
    lfsr_c[10] = lfsr_q[2];
    lfsr_c[11] = lfsr_q[3];
    lfsr_c[12] = lfsr_q[4];
    lfsr_c[13] = lfsr_q[5];
    lfsr_c[14] = lfsr_q[6];
    lfsr_c[15] = lfsr_q[7] ^ lfsr_q[8] ^ lfsr_q[9] ^ lfsr_q[10] ^ lfsr_q[11] ^ lfsr_q[12] ^ lfsr_q[13]

  end // always

  always @(posedge clk, posedge rst) begin
    if(rst) begin
      lfsr_q <= {16{1'b1}};
    end
    else begin
      if(load) begin
      lfsr_q <= crc_en ? lfsr_c : lfsr_q;
      end
    end
  end // always

endmodule // crc
```

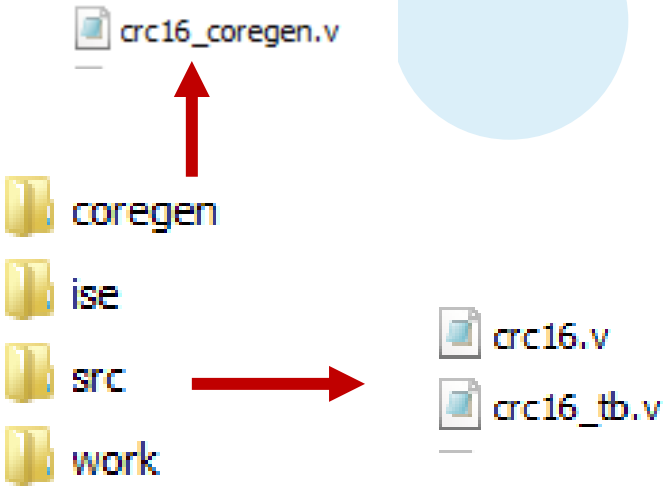
Hierarchy of Verilog Files





Hierarchy of Verilog Files





Test-Vector Generator

Chapters - Re

https://replit.com/@wlstj9964/NuttyNiceChapters#main.c

lstj9964 / NuttyNiceChapters

Run

main.c

```c
#include <stdio.h>
#define POLYNOMIAL  0x8005
//#define USE_CRC_TABLE
#ifndef USE_CRC_TABLE

static unsigned short crc_table[256];

#else

static unsigned short crc_table[256] = { 0x0000,
0x8005, 0x800F, 0x000A, 0x801B, 0x001E, 0x0014, 0x8011,
0x8033, 0x0036, 0x003C, 0x8039, 0x0028, 0x802D, 0x8027,
0x0022, 0x8063, 0x0066, 0x006C, 0x8069, 0x0078, 0x807D,
0x8077, 0x0072, 0x0050, 0x8055, 0x805F, 0x005A, 0x804B,
0x004E, 0x0044, 0x8041, 0x80C3, 0x00C6, 0x00CC, 0x80C9,
0x00D8, 0x80DD, 0x80D7, 0x00D2, 0x00F0, 0x80F5, 0x80FF,
0x00FA, 0x80EB, 0x00EE, 0x00E4, 0x80E1, 0x00A0, 0x80A5,
0x80AF, 0x00AA, 0x80BB, 0x00BE, 0x00B4, 0x80B1, 0x8093,
0x0096, 0x009C, 0x8099, 0x0088, 0x808D, 0x8087, 0x0082,
0x8183, 0x0186, 0x018C, 0x8189, 0x0198, 0x819D, 0x8197,
0x0192, 0x01B0, 0x81B5, 0x81BF, 0x01BA, 0x81AB, 0x01AE,
0x01A4, 0x81A1, 0x01E0, 0x81E5, 0x81EF, 0x01EA, 0x81FB,
0x01FE, 0x01F4, 0x81F1, 0x81D3, 0x01D6, 0x01DC, 0x81D9,
0x01C8, 0x81CD, 0x81C7, 0x01C2, 0x0140, 0x8145, 0x814F,
0x014A, 0x815B, 0x015E, 0x0154, 0x8151, 0x8173, 0x0176,
0x017C, 0x8179, 0x0168, 0x816D, 0x8167, 0x0162, 0x8123,
0x0126, 0x012C, 0x8129, 0x0138, 0x813D, 0x8137, 0x0132,
0x0110, 0x8115, 0x811F, 0x011A, 0x810B, 0x010E, 0x0104,
0x8101, 0x8303, 0x0306, 0x030C, 0x8309, 0x0318, 0x831D,
0x8317, 0x0312, 0x0330, 0x8335, 0x833F, 0x033A, 0x832B,
0x032E, 0x0324, 0x8321, 0x0360, 0x8365, 0x836F, 0x036A,
0x837B, 0x037E, 0x0374, 0x8371, 0x8353, 0x0356, 0x035C,
```

lstj9964/NuttyNiceChapters#main.c

Chapters

Run

main.c

```c
0x83CF, 0x03CA, 0x83DB, 0x03DE, 0x03D4, 0x83D1, 0x83F3,
0x03F6, 0x03FC, 0x83F9, 0x03E8, 0x83ED, 0x83E7, 0x03E2,
0x83A3, 0x03A6, 0x03AC, 0x83A9, 0x03B8, 0x83BD, 0x83B7,
0x03B2, 0x0390, 0x8395, 0x839F, 0x039A, 0x838B, 0x038E,
0x0384, 0x8381, 0x0280, 0x8285, 0x828F, 0x028A, 0x829B,
0x029E, 0x0294, 0x8291, 0x82B3, 0x02B6, 0x02BC, 0x82B9,
0x02A8, 0x82AD, 0x82A7, 0x02A2, 0x82E3, 0x02E6, 0x02EC,
0x82E9, 0x02F8, 0x82FD, 0x82F7, 0x02F2, 0x02D0, 0x82D5,
0x82DF, 0x02DA, 0x82CB, 0x02CE, 0x02C4, 0x82C1, 0x8243,
0x0246, 0x024C, 0x8249, 0x0258, 0x825D, 0x8257, 0x0252,
0x0270, 0x8275, 0x827F, 0x027A, 0x826B, 0x026E, 0x0264,
0x8261, 0x0220, 0x8225, 0x822F, 0x022A, 0x823B, 0x023E,
0x0234, 0x8231, 0x8213, 0x0216, 0x021C, 0x8219, 0x0208,
0x820D, 0x8207, 0x0202 };
#endif

void gen_crc_table();
unsigned short update_crc(unsigned short crc_accum, unsigned
char* data_blk_ptr,
  unsigned short data_blk_size);


void gen_crc_table()
{
#ifndef USE_CRC_TABLE
  register unsigned short i, j;
  register unsigned short crc_accum;

  /* 0 ~ 255 까지의 CRC를 미리 계산 */
  /* ex) i=2 이면 0x020000 을 polynomial로 CRC 나눗셈 하는 것임
*/
  for (i = 0; i < 256; i++)
  {
```

wlstj9964/NuttyNiceChapters#main.c

ceChapters

Run

main.c

```c
  /* ex) i=2 이면 0x020000 을 polynomial로 CRC 나눗셈 하는 것임
*/
  for (i = 0; i < 256; i++)
  {
    crc_accum = ((unsigned short)i << 8);
    /* CRC나눗셈이 가능하도록 8-bit shift */
    for (j = 0; j < 8; j++)
    {
      /* 나머지항의 MSB가 1인지 검사 */
      if (crc_accum & 0x8000L)
        crc_accum = (crc_accum << 1) ^ POLYNOMIAL;
      /* 참이면 1-bit shift하고 polynomial을 빼줌(==XOR) */
      else
        crc_accum = (crc_accum << 1);
      /* 거짓이면 1-bit shift만 함*/
    }
    /* 결과적으로 총 16-bit가 shift 되었고 */
    /* 나머지만 남게 됨 */
    crc_table[i] = crc_accum;

    //printf("%03d=%04x, ", i, crc_accum);
    //if(i%7 == 0)
    //  printf("\n");
  }
#endif
  return;
}


/*
이전에 계산한 CRC 값을 추가된 데이터에 맞춰 갱신함
이전:
```

https://replit.com/@wlstj9964/NuttyNiceChapters#main.c

wlstj9964 / NuttyNiceChapters

Run

main.c

```c
이전:          ___________
    8005 | XX 00 00
           yy yy ---> CRC

갱신:          _____________
    8005 | XX ZZ 00 00
           yy yy 00
              ww ww ---> CRC'
*/
unsigned short update_crc(unsigned short crc_accum, uns
char* data_blk_ptr, unsigned short data_blk_size)
{
  register unsigned short i, j;
  for (j = 0; j < data_blk_size; j++)
  {
    /* 추가된 데이터 ZZ는                          */
    /* 이전 계산시 나머지항의 상위 바이트와       */
    /* 자리수가 맞으므로 그 둘을 합하고 다시      */
    /* 그 자리까지의 나머지를 구함                */
    i = ((unsigned short)(crc_accum >> 8) ^ *data_blk_p
& 0xff;
    crc_accum = (crc_accum << 8) ^ crc_table[i];
    /* 나머지항의 하위 바이트는 뒤에 0x00 이      */
    /* 추가되어 자리수가 올라가고 앞자리의        */
    /* 나머지와 더해짐 */

    printf("crc: %05d\n",crc_accum);

  }
  printf("\n");
  return crc_accum;
}
```

tj9964/NuttyNiceChapters#main.c

hapters

Run

Publish

Invite

main.c

```c
    /* 나머지와 더해짐 */

    printf("crc: %0x\n", crc_accum);
  }
  printf("\n");
  return crc_accum;
}

/*********************************************************
  CRC-16 test sample
*********************************************************/
int main() {
  unsigned short crc, check;
  unsigned char in_frame[20];

  gen_crc_table();

  /* 임의 데이터 생성 */
  in_frame[0] = 0x55;
  in_frame[1] = 0xA1;
  in_frame[2] = 0x12;
  in_frame[3] = 0x34;
  for (int i = 0; i< 4; i++){
    crc = update_crc(0,&in_frame[i],1);
  }
  crc = update_crc(0, in_frame, 4);
  printf("in_frame crc: %04x", crc);
  return crc;
}
```

Console Shell

```
> clang-7 -pthread -lm -o main main.c
> ./main
crc: 1fe

crc: 83c5

crc: 6c

crc: 80bb

crc: 1fe
crc: fdc0
crc: 4261
crc: e037

in_frame crc: e037exit status 55
>
```

Verilog-Testbench for Simulating RTL Design of the Algorithm

C:/Users/SM-PC/Desktop/finalcrc/CRC_16_parallel_tb.v (/CRC_16_parallel_test) - Default

```
//
module CRC_16_parallel_test;
reg clk;
reg rst;
reg load;
reg d_finish;
reg [7:0] crc_in;
wire [7:0] crc_out;
parameter clk_period = 40;

initial
begin
 #clk_period clk = 1;
 #clk_period rst = 1;
 #clk_period rst = 0;
 #clk_period crc_in[7:0] = 8'b1010_1010; //
 #clk_period load = 1;
 #clk_period load = 0;
 #clk_period d_finish = 0;
 #(10*clk_period) d_finish = 1;
 #clk_period d_finish = 0;
end
always #(clk_period/2) clk = ~clk;
always #(clk_period) crc_in[7:0] = ~crc_in[7:0]; //
//
CRC_16_parallel u1(.clk(clk), .rst(rst), .load(load), .d_finish(d_finish), .crc_in(crc_in), .crc_out(crc_out));
endmodule
```

Xilinx ISE

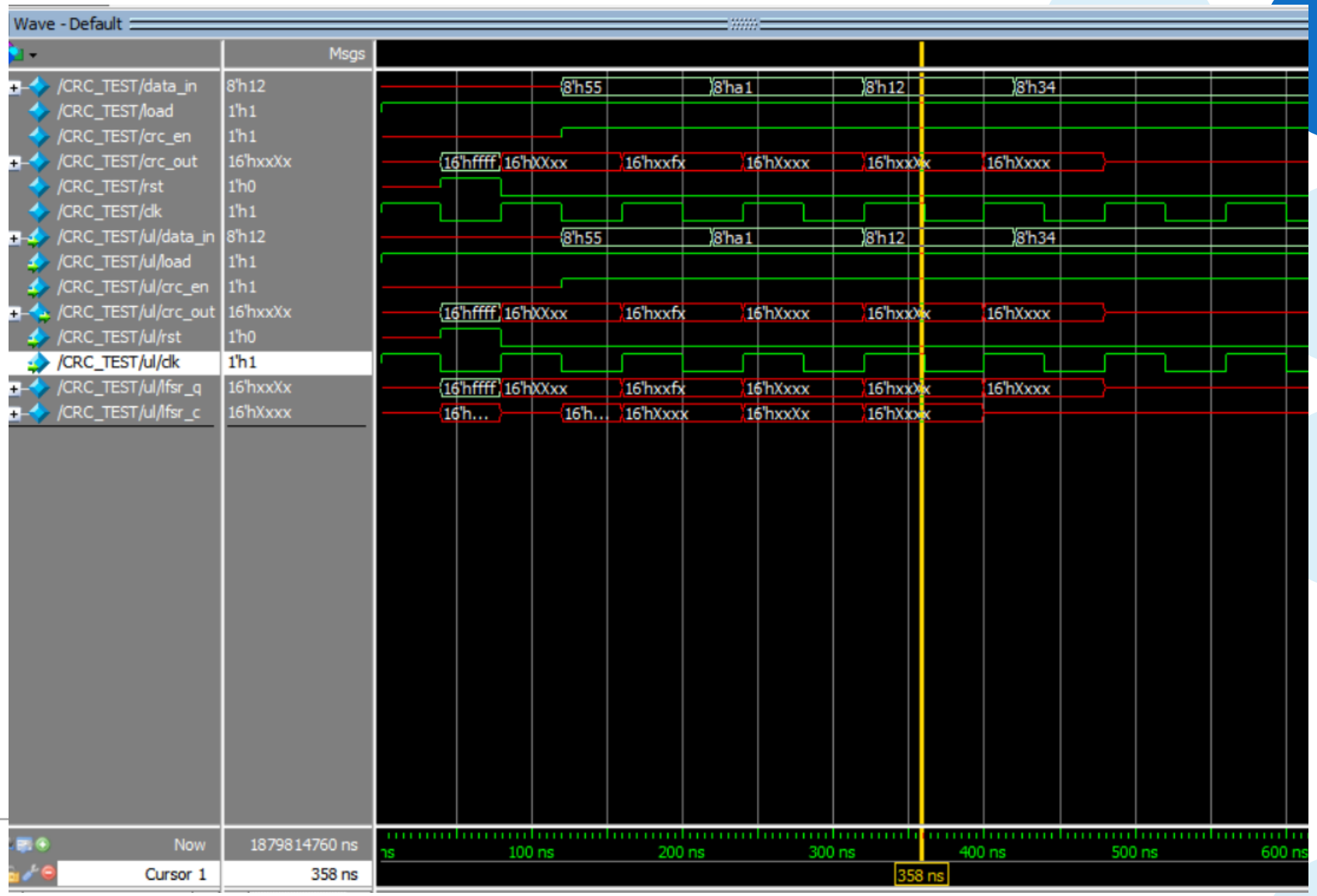

Msgs
/CRC_16_parallel_test/clk 1'h1
/CRC_16_parallel_test/rst 1'h0
/CRC_16_parallel_test/load 1'h0
/CRC_16_parallel_test/d_finish 1'h0
/CRC_16_parallel_test/crc_in 8'h55
[7] 0
[6] 1
[5] 0
[4] 1
[3] 0
[2] 1
[1] 0
[0] 1
/CRC_16_parallel_test/crc_out 8'h00
[7] 1'h0
[6] 1'h0
[5] 1'h0
[4] 1'h0
[3] 1'h0
[2] 1'h0
[1] 1'h0
[0] 1'h0
8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa
8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'h55 8'haa 8'hf4 8'hec 8'h00
Now 900 ns
100 ns 200 ns 300 ns 400 ns 500 ns 600 ns 700 ns 800 ns 900 ns

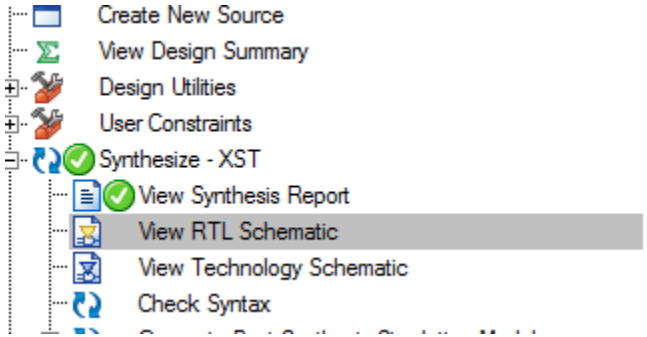
Verilog-Testbench for Simulating RTL Design of the Algorithm

C:/Users/SM-PC/Desktop/crc16/crc16_tb.v (/CRC_TEST) - Default

```
module CRC_TEST;
  reg [7:0] data_in;
  reg load;
  reg crc_en;
  wire [15:0] crc_out;
  reg rst;
  reg clk;
  parameter clk_period = 100;

  CRC u1(.data_in(data_in), .load(load), .crc_en(crc_en), .crc_out(crc_out), .rst(rst), .clk(clk));

  initial begin
    clk = 1;
    load = 1;
    #clk_period rst = 1;
    #clk_period rst = 0;
    #clk_period crc_en = 1;
    data_in[7:0] = 8'b01010101;
    #4000 data_in[7:0] = 8'b10100001;
    #4000 data_in[7:0] = 8'b00010010;
    #4000 data_in[7:0] = 8'b00110100;
  end
  always #(clk_period) clk = ~clk;

endmodule
```

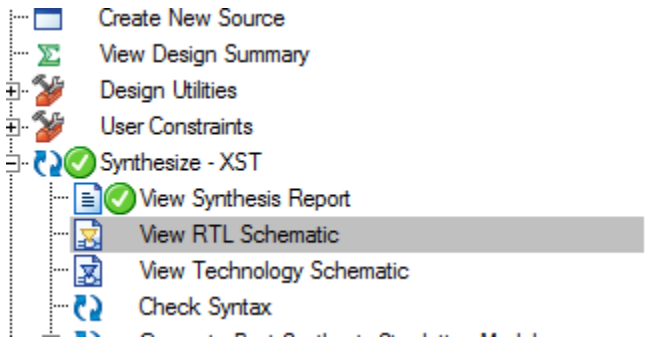
Xilinx ISE


Wave - Default
Msgs
/CRC_TEST/data_in 8'h12
/CRC_TEST/load 1'h1
/CRC_TEST/crc_en 1'h1
/CRC_TEST/crc_out 16'hxxXx
/CRC_TEST/rst 1'h0
/CRC_TEST/clk 1'h1
/CRC_TEST/ul/data_in 8'h12
/CRC_TEST/ul/load 1'h1
/CRC_TEST/ul/crc_en 1'h1
/CRC_TEST/ul/crc_out 16'hxxXx
/CRC_TEST/ul/rst 1'h0
/CRC_TEST/ul/clk 1'h1
/CRC_TEST/ul/lfsr_q 16'hxxXx
/CRC_TEST/ul/lfsr_c 16'hXxxx
Now 1879814760 ns
Cursor 1 358 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
358 ns




```
=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Registers                                            : 16
 Flip-Flops                                            : 16
# Xors                                                 : 19
 1-bit xor2                                            : 16
 1-bit xor3                                            : 1
 1-bit xor7                                            : 1
 1-bit xor9                                            : 1

=========================================================================
```

```
Device utilization summary:
---------------------------

Selected Device : 2v8000ff1152-5

 Number of Slices:                    12  out of  46592     0%
 Number of Slice Flip Flops:          16  out of  93184     0%
 Number of 4 input LUTs:              21  out of  93184     0%
 Number of IOs:                       28
 Number of bonded IOBs:               28  out of    824     3%
 Number of GCLKs:                      1  out of     16     6%
```




```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 3.118ns (Maximum Frequency: 320.718MHz)
   Minimum input arrival time before clock: 5.294ns
   Maximum output required time after clock: 5.080ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 3.118ns (frequency: 320.718MHz)
  Total number of paths / destination ports: 46 / 16
-------------------------------------------------------------------------
Delay:               3.118ns (Levels of Logic = 2)
  Source:            lfsr_q_9 (FF)
  Destination:       lfsr_q_1 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: lfsr_q_9 to lfsr_q_1
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDPE:C->Q             4   0.494   0.898  lfsr_q_9 (lfsr_q_9)
     LUT4_D:I0->O          2   0.382   0.640  lfsr_c<0>253 (lfsr_c<0>2_map38)
     LUT4:I2->O            1   0.382   0.000  lfsr_c<15> (lfsr_c<15>)
     FDPE:D                    0.322          lfsr_q_15
    ----------------------------------------
    Total                      3.118ns (1.580ns logic, 1.538ns route)
                                       (50.7% logic, 49.3% route)
```

Hardware Performance Evaluation

Analysis / Type	Total Cycle Counts	Critical Path Delay /Operating Frequency	Execution Time (Cycle Count x Critical Path Delay)
Hardware (*One-Procedure RTL)*	6	Minimum period: 3.118ns (Maximum Frequency: 320.718MHz)	6 X 3.118 ns = 18.708 ns