
# مستند پروژه مبدل 128B to 5B با verilog

**گردآورنده: امین داودآبادی**
**استاد: دکتر فصحتی**

## مقدمه

در بسیاری از سیستم‌های ارتباطی، انتقال داده‌ها به صورت بیتی همراه با رمزگذاری‌های خاص (برای بهبود تعادل جریان و قابلیت اطمینان) انجام می‌شود. همچنین به‌منظور تشخیص خطاهای احتمالی، از الگوریتم‌های CRC (Cyclic Redundancy Check) استفاده می‌شود. این پروژه دقیقاً همین فرآیند را در محیط شبیه‌سازی Verilog مدل‌سازی کرده است.

---

## هدف پروژه

هدف اصلی این پروژه:

- **ایجاد یک داده‌ی اصلی 130 بیتی** با داشتن بیت‌های کنترلی مشخص (برای مثال، بیت‌های 0 و 1 در دو موقعیت اول).
- **انتقال داده** به صورت بیتی از یک فرستنده (TX1) به یک گیرنده (RX2) با امکان تزریق خطا (به عنوان مثال تغییر بیت در موقعیت 50).
- **محاسبه CRC** برای کل داده (130 بیت) جهت تشخیص خطا.
- **رمزگذاری داده‌های 130 بیتی** به قالب 5B/6B:
  - تقسیم داده به بلوک‌های 5 بیتی (26 بلوک)
  - تبدیل هر بلوک 5 بیتی به 6 بیت با استفاده از یک تابع look-up (مشابه جداول رمزگذاری در کد آردوینو).
- **انتقال بلوک‌های 6 بیتی** از فرستنده دوم (TX2) به گیرنده اول (RX1) با تزریق خطا در هر بلوک (یک بیت تصادفی تغییر داده می‌شود).
- **محاسبه CRC برای هر بلوک 6 بیتی** جهت تشخیص خطا در انتقال بلوکی.
- **رمزگشایی بلوک‌های 6 بیتی** به بلوک‌های 5 بیتی و بازترکیب آن‌ها به یک داده‌ی 130 بیتی.
- **مقایسه داده‌ی نهایی رمزگشایی شده** با داده‌ی اصلی جهت شمارش خطاهای موجود در زنجیره انتقال.

---

## شرح اجمالی عملکرد سیستم

1. **ایجاد داده‌ی اصلی:**
یک بردار 130 بیتی تولید می‌شود که در آن دو بیت اول به عنوان بیت‌های کنترلی (0 و 1) تنظیم شده و بقیه بیت‌ها به صورت تصادفی مقداردهی می‌شوند.
2. **انتقال مرحله اول:**
داده‌ی اصلی بیت به بیت از TX1 به RX2 ارسال می‌شود. در این مرحله، به عنوان نمونه، بیت موقعیت 50 معکوس (خطا تزریق می‌شود).
3. **محاسبه CRC برای داده‌های 130 بیتی:**
برای داده اصلی و داده دریافتی، یک مقدار CRC8 با استفاده از چندجمله‌ای 0x07 محاسبه می‌شود و سپس مقایسه می‌گردد تا از صحت انتقال اطمینان حاصل شود.

4. **رمزگذاری 5B/6B:**
داده دریافتی 130 بیتی به 26 بلوک 5 بیتی تقسیم شده و هر بلوک با استفاده از یک تابع رمزگذاری به 6 بیت تبدیل می‌شود. در این تبدیل از یک جدول look-up استفاده شده است.
5. **انتقال مرحله دوم:**
بلوک‌های 6 بیتی از TX2 به RX1 ارسال می‌شوند. در هر بلوک، یک بیت تصادفی تغییر داده می‌شود (تزریق خطا) و برای هر بلوک، CRC محاسبه و بررسی می‌شود.
6. **رمزگشایی:**
بلوک‌های 6 بیتی دریافتی به کمک یک تابع رمزگشایی به بلوک‌های 5 بیتی تبدیل می‌شوند و سپس داده‌های 5 بیتی به صورت مجدد ترکیب شده و یک بردار 130 بیتی به دست می‌آید.
7. **مقایسه نهایی:**
داده‌ی نهایی رمزگشایی شده با داده‌ی اصلی (به جز بیت‌های کنترلی) مقایسه شده تا تعداد خطاهای رخ داده مشخص شود.

---

# توضیح بخش‌های مختلف کد

## ایجاد داده‌های اصلی (130 بیت)

در ابتدا یک بردار 130 بیتی به نام original_data ایجاد شده و دو بیت کنترلی (بیت 0 و بیت 1) به ترتیب برابر با 0 و 1 تنظیم می‌شوند. بقیه بیت‌ها به صورت تصادفی مقداردهی می‌شوند:

```
// 1. ایجاد داده 130 بیتی
original_data[0] = 0;
original_data[1] = 1;
for(i = 2; i < 130; i = i + 1) begin
original_data[i] = $random % 2;
end
```

## انتقال مرحله اول (TX1 -> RX2)

در این بخش، داده اصلی به صورت بیت به بیت "انتقال" داده می‌شود. در شبیه‌سازی، بیت شماره 50 معکوس (تزریق خطا) شده و سپس هر بیت ارسال و دریافت می‌شود:

```
verilog
CopyEdit
// 2. انتقال مرحله اول
for(i = 0; i < 130; i = i + 1) begin
if(i == 50) begin
$display("Flipping bit at position %0d", i);
received_first[i] = ~original_data[i]; // تزریق خطا در بیت 50
end else begin
```

```
;[received_first[i] = original_data[i
                                    end
;([display("Bit %0d: Sent %0d, Received %0d", i, original_data[i], received_first[i$
                                    end
```

### محاسبه و مقایسه CRC برای 130 بیت

برای اطمینان از صحت انتقال در مرحله اول، مقدار CRC8 بر روی داده اصلی و داده دریافتی محاسبه می‌شود. تابع calc_crc8_130 بر مبنای چندجمله‌ای 0x07 پیاده‌سازی شده است:

```
verilog
CopyEdit
// محاسبه CRC8 برای داده 130 بیتی
;(crc_original = calc_crc8_130(original_data
;(crc_received = calc_crc8_130(received_first
                    if(crc_original !== crc_received) begin
                ;first_error_count = first_error_count + 1
,"display("FIRST TRANSMISSION CRC MISMATCH: Original CRC = %h, Received CRC = %h$
                        ;(crc_original, crc_received
                                    end else
                ;(display("FIRST TRANSMISSION CRC MATCH: %h", crc_original$
```

تابع محاسبه CRC به صورت زیر تعریف شده است:

```
verilog
CopyEdit
;function [7:0] calc_crc8_130
                    ;input [129:0] data
                        ;integer i, j
                    ;reg [7:0] crc
                            begin
                        ;crc = 8'b0
                for(i = 0; i < 130; i = i + 1) begin
            ;(crc = crc ^ (data[i] ? 8'h80 : 8'h00
                for(j = 0; j < 8; j = j + 1) begin
                                ([if(crc[7
                    ;crc = (crc << 1) ^ 8'h07
                                    else
                        ;crc = crc << 1
                                    end
```

```
end
;calc_crc8_130 = crc
end
endfunction
```

### رمزگذاری 5B/6B

در این مرحله، داده 130 بیتی دریافتی به بلوک‌های 5 بیتی تقسیم می‌شود. سپس با استفاده از تابع encode5b6b هر بلوک 5 بیتی به 6 بیت تبدیل می‌شود. در کد زیر، برای هر بلوک 5 بیتی، قسمت مربوط به رمزگذاری نمایش داده شده است:

verilog
CopyEdit

```
// 4. رمزگذاری: تبدیل 130 بیت دریافتی به 156 بیت (26 بلوک 5بیتی به 6بیتی)
;received_first[0] = 0
;received_first[1] = 0
for(block = 0; block < 26; block = block + 1) begin
;reg [4:0] five_bits
;[five_bits = received_first[block*5 +: 5
;(encoded_data[block*6 +: 6] = encode5b6b(five_bits
;((display("Encoding Block %0d: 5-bit %b -> 6-bit %b", block, five_bits, encode5b6b(five_bits$
end
```

تابع رمزگذاری encode5b6b به صورت زیر پیاده‌سازی شده است:

verilog
CopyEdit

```
;function [5:0] encode5b6b
;input [4:0] in
begin
(case(in
;b00000: encode5b6b = 6'b100111'5
;b00001: encode5b6b = 6'b011101'5
;b00010: encode5b6b = 6'b101101'5
;b00011: encode5b6b = 6'b110001'5
;b00100: encode5b6b = 6'b110101'5
;b00101: encode5b6b = 6'b101001'5
;b00110: encode5b6b = 6'b011001'5
;b00111: encode5b6b = 6'b111000'5
;b01000: encode5b6b = 6'b111001'5
;b01001: encode5b6b = 6'b100101'5
```

```
;b01010: encode5b6b = 6'b010101'5
;b01011: encode5b6b = 6'b110100'5
;b01100: encode5b6b = 6'b001101'5
;b01101: encode5b6b = 6'b101100'5
;b01110: encode5b6b = 6'b011100'5
;b01111: encode5b6b = 6'b010111'5
;b10000: encode5b6b = 6'b011011'5
;b10001: encode5b6b = 6'b100011'5
;b10010: encode5b6b = 6'b010011'5
;b10011: encode5b6b = 6'b110010'5
;b10100: encode5b6b = 6'b001011'5
;b10101: encode5b6b = 6'b101010'5
;b10110: encode5b6b = 6'b011010'5
;b10111: encode5b6b = 6'b111010'5
;b11000: encode5b6b = 6'b110011'5
;b11001: encode5b6b = 6'b100110'5
;b11010: encode5b6b = 6'b010110'5
;b11011: encode5b6b = 6'b110110'5
;b11100: encode5b6b = 6'b001110'5
;b11101: encode5b6b = 6'b101110'5
;b11110: encode5b6b = 6'b011110'5
;b11111: encode5b6b = 6'b101011'5
 ;default:   encode5b6b = 6'b000000
                endcase
                    end
endfunction
```

### انتقال مرحله دوم (RX1 <- TX2) و تشخیص خطا در بلوک‌های 6 بیتی

پس از رمزگذاری، بلوک‌های 6 بیتی به عنوان داده رمزگذاری شده (encoded_data) جهت انتقال آماده می‌شوند. در این بخش، هر بلوک 6 بیتی ارسال می‌شود و در هر بلوک یک بیت به صورت تصادفی تغییر می‌کند (تزریق خطا). سپس برای هر بلوک، CRC محاسبه و با مقدار اصلی آن مقایسه می‌شود:

verilog

CopyEdit

```
// 5. انتقال مرحله دوم و تشخیص خطا برای هر بلوک 6 بیتی
for(block = 0; block < 26; block = block + 1) begin
    // انتخاب تصادفی یک بیت برای تغییر در بلوک
    rand_flip = $random % 6;
$display("Block %0d: Flipping bit position %0d in the 6-bit block", block, rand_flip);
    for(j = 0; j < 6; j = j + 1) begin
```

```
(if(j == rand_flip
;[received_second[block*6 + j] = ~encoded_data[block*6 + j
else
;[received_second[block*6 + j] = encoded_data[block*6 + j
,display("  Block %0d, Bit %0d: Sent %0d, Received %0d", block, j$
;([encoded_data[block*6 + j], received_second[block*6 + j
end

// محاسبه CRC برای بلوک 6 بیتی
;([crc_block_original = calc_crc8_6(encoded_data[block*6 +: 6
;([crc_block_received = calc_crc8_6(received_second[block*6 +: 6
if(crc_block_original !== crc_block_received) begin
;second_error_count = second_error_count + 1
,"display("  Block %0d CRC MISMATCH: Original CRC = %h, Received CRC = %h$
;(block, crc_block_original, crc_block_received
end else
;(display("  Block %0d CRC MATCH: %h", block, crc_block_original$
end
```

تابع محاسبه CRC برای بلوک 6 بیتی به صورت زیر تعریف شده است:

```
verilog
CopyEdit
;function [7:0] calc_crc8_6
;input [5:0] data
;integer j, k
;reg [7:0] crc
begin
;crc = 8'b0
for(j = 0; j < 6; j = j + 1) begin
;(crc = crc ^ (data[j] ? 8'h80 : 8'h00
for(k = 0; k < 8; k = k + 1) begin
([if(crc[7
;crc = (crc << 1) ^ 8'h07
else
;crc = crc << 1
end
end
;calc_crc8_6 = crc
end
endfunction
```

### رمزگشایی 6B/5B و بازترکیب داده‌ها

پس از دریافت بلوک‌های 6 بیتی، هر بلوک توسط تابع decode6b5b به بلوک 5 بیتی تبدیل می‌شود. سپس این بلوک‌های 5 بیتی به صورت پشت سرهم در بردار decoded_data قرار می‌گیرند:

verilog
CopyEdit
```
// 6. رمزگشایی: تبدیل بلوک‌های 6 بیتی دریافت‌شده به 5 بیتی
for(block = 0; block < 26; block = block + 1) begin
    reg [5:0] six_bits;
    reg [4:0] five_bits_decoded;
    six_bits = received_second[block*6 +: 6];
    five_bits_decoded = decode6b5b(six_bits);
    decoded_data[block*5 +: 5] = five_bits_decoded;
    $display("Block %0d: 6-bit %b -> Decoded 5-bit %b", block, six_bits, five_bits_decoded);
end
```

تابع رمزگشایی decode6b5b به صورت زیر تعریف شده است:

verilog
CopyEdit
```
function [4:0] decode6b5b;
    input [5:0] in;
    begin
        case(in)
            6'b100111: decode6b5b = 5'b00000;
            6'b011101: decode6b5b = 5'b00001;
            6'b101101: decode6b5b = 5'b00010;
            6'b110001: decode6b5b = 5'b00011;
            6'b110101: decode6b5b = 5'b00100;
            6'b101001: decode6b5b = 5'b00101;
            6'b011001: decode6b5b = 5'b00110;
            6'b111000: decode6b5b = 5'b00111;
            6'b111001: decode6b5b = 5'b01000;
            6'b100101: decode6b5b = 5'b01001;
            6'b010101: decode6b5b = 5'b01010;
            6'b110100: decode6b5b = 5'b01011;
            6'b001101: decode6b5b = 5'b01100;
            6'b101100: decode6b5b = 5'b01101;
```

```
;b011100: decode6b5b = 5'b01110'6
;b010111: decode6b5b = 5'b01111'6
;b011011: decode6b5b = 5'b10000'6
;b100011: decode6b5b = 5'b10001'6
;b010011: decode6b5b = 5'b10010'6
;b110010: decode6b5b = 5'b10011'6
;b001011: decode6b5b = 5'b10100'6
;b101010: decode6b5b = 5'b10101'6
;b011010: decode6b5b = 5'b10110'6
;b111010: decode6b5b = 5'b10111'6
;b110011: decode6b5b = 5'b11000'6
;b100110: decode6b5b = 5'b11001'6
;b010110: decode6b5b = 5'b11010'6
;b110110: decode6b5b = 5'b11011'6
;b001110: decode6b5b = 5'b11100'6
;b101110: decode6b5b = 5'b11101'6
;b011110: decode6b5b = 5'b11110'6
;b101011: decode6b5b = 5'b11111'6
default:   decode6b5b = 5'bx; // نشان‌دهنده خطا
endcase
end
endfunction
```

### مقایسه داده‌های نهایی با داده‌های اصلی

در انتها، داده‌ی بازترکیب‌شده (decoded_data) با داده‌ی اصلی (original_data) – به جز بیت‌های کنترلی – مقایسه می‌شود تا تعداد خطاهای رخداده در فرآیند انتقال مشخص گردد:

```
verilog
CopyEdit
// 7. مقایسه داده نهایی با داده اصلی (از بیت 2 به بعد)
for(i = 2; i < 130; i = i + 1) begin
if(original_data[i] !== decoded_data[i]) begin
error_count = error_count + 1;
"display("Mismatch at bit %0d: Original %0d, Decoded %0d$,
([i, original_data[i], decoded_data[i];
end
end

("=== display("\n=== Transmission Complete$;
(display("Data bit errors: %0d", error_count$;
```

```
$display("First transmission CRC errors: %0d", first_error_count);
$display("Second transmission CRC errors: %0d", second_error_count);
```

---

## خروجی

### ۱. نمایش داده اصلی (Original Data (130 bits))

در ابتدای شبیه‌سازی، یک بردار 130 بیتی تولید می‌شود.
مثال خروجی:

```
java
CopyEdit
Original Data (130 bits):
01011111101101101010110100010111100001011111001110000010011000001111100111011001101110110000110011111011100000001111101100110111010
```

این رشته نشان‌دهنده داده‌ای است که در ابتدا توسط بخش تولید داده (کد زیر) ایجاد شده است:

```
verilog
CopyEdit
original_data[0] = 0;
original_data[1] = 1;
for(i = 2; i < 130; i = i + 1) begin
    original_data[i] = $random % 2;
end
```

دو بیت اول ثابت (0 و 1) تنظیم شده و بقیه بیت‌ها به صورت تصادفی تولید شده‌اند.

---

### ۲. مرحله اول انتقال (First Transmission)

در این مرحله، هر بیت از داده اصلی به صورت تک تک "انتقال" داده می‌شود. برای شبیه‌سازی خطا، بیت شماره 50 تغییر می‌کند:

```
python-repl
CopyEdit
...
Bit 49: Sent 0, Received 0
```

```
Flipping bit at position 50
 Bit 50: Sent 0, Received 1
 Bit 51: Sent 0, Received 0
 ...
```

- **توضیح:**
  - برای بیت‌های 0 تا 49، مقدار ارسال شده و دریافت شده یکسان است.
  - در بیت 50، پیام «Flipping bit at position 50» نشان می‌دهد که به عمد بیت 50 معکوس شده است (از 0 به 1) تا خطا را شبیه‌سازی کند.
  - بیت‌های بعد از 50 نیز همانند بیت‌های قبل انتقال پیدا می‌کنند.

پس از انتقال تک تک بیت‌ها، برای کل 130 بیت مقدار CRC محاسبه می‌شود. در این خروجی مشاهده می‌کنیم:

java
CopyEdit

```
FIRST TRANSMISSION CRC MISMATCH: Original CRC = bf, Received CRC = cf
```

- **توضیح:**
  - به دلیل تغییر بیت 50، مقدار CRC محاسبه‌شده روی داده اصلی (bf) با CRC داده دریافتی (cf) تفاوت دارد. این موضوع نشان‌دهنده وجود خطا در انتقال است.

---

### ۳. مرحله رمزگذاری (Encoding) 5B/6B

پس از انتقال مرحله اول، داده دریافت‌شده (که شامل خطا در بیت 50 شده) به بلوک‌های 5 بیتی تقسیم می‌شود. سپس هر بلوک 5 بیتی توسط تابع `encode5b6b` به یک بلوک 6 بیتی تبدیل می‌شود. خروجی مانند زیر نشان می‌دهد:

arduino
CopyEdit

```
Encoding Block 0: 5-bit 11000 -> 6-bit 110011
Encoding Block 1: 5-bit 01111 -> 6-bit 010111
Encoding Block 2: 5-bit 11011 -> 6-bit 110110
...
Encoding Block 25: 5-bit 01111 -> 6-bit 010111
```

- **توضیح:**
  - برای هر بلوک، ابتدا ۵ بیت استخراج می‌شود (مثلاً برای Block 0، بیت‌های مربوط به 11000).
  - سپس تابع رمزگذاری مقدار معادل ۶ بیتی را برمی‌گرداند. به عنوان نمونه، بلوک 0 با ورودی "5b 11000" به "6b 110011" تبدیل می‌شود.
  - این مراحل به‌طور پی‌درپی برای تمام 26 بلوک انجام می‌شود.

---

## ۴. مرحله دوم انتقال (Second Transmission)

در این مرحله، بلوک‌های 6 بیتی تولیدشده ارسال می‌شوند. در هر بلوک، به صورت تصادفی یک بیت تغییر داده می‌شود (تزریق خطا) و پس از انتقال، CRC هر بلوک محاسبه و مقایسه می‌شود.

برای هر بلوک پیام‌های مربوط به انتقال به‌صورت زیر نمایش داده می‌شود. به عنوان مثال:

yaml
CopyEdit

```
Block 0: Flipping bit position 0 in the 6-bit block
                Block 0, Bit 0: Sent 1, Received 0
                Block 0, Bit 1: Sent 1, Received 1
                Block 0, Bit 2: Sent 0, Received 0
                Block 0, Bit 3: Sent 0, Received 0
                Block 0, Bit 4: Sent 1, Received 1
                Block 0, Bit 5: Sent 1, Received 1
Block 0 CRC MISMATCH: Original CRC = 44, Received CRC = a8
```

- **توضیح بلوک 0:**
  - پیام "Flipping bit position 0" نشان می‌دهد که بیت 0 در بلوک معکوس شده است.
  - سپس برای بیت‌های هر بلوک مقدار ارسال‌شده و دریافت‌شده نمایش داده می‌شود.
  - در نهایت، CRC محاسبه‌شده روی بلوک اصلی (44) با CRC بلوک دریافتی (a8) مطابقت ندارد؛ به همین دلیل پیام "CRC MISMATCH" نشان داده می‌شود.

برای بلوک‌های دیگر نیز مشابه همین روند اتفاق می‌افتد:

- در برخی بلوک‌ها CRC مطابقت دارد (مثلاً Block 7، Block 6، Block 5 و غیره) و پیام "CRC MATCH" نمایش داده می‌شود.
- در بلوک‌های مختلف، بیت تصادفی معکوس شده و CRC محاسبه‌شده تفاوت دارد؛ در این موارد پیام "CRC MISMATCH" همراه با مقادیر CRC اصلی و دریافت‌شده نمایش داده می‌شود.

توجه کنید که در برخی پیام‌ها مانند "Flipping bit position -2" یا "-3" یا "-1"، این مقادیر ممکن است به دلیل نحوه‌ی انتخاب بیت تصادفی یا چاپ نامناسب اندیس رخ داده باشند؛ اما مفهوم اصلی این است که یک بیت در آن بلوک تغییر کرده است.

---

## ۵. مرحله رمزگشایی (Decoding)

پس از انتقال مرحله دوم، بلوک‌های 6 بیتی دریافت‌شده به کمک تابع `decode6b5b` به بلوک‌های 5 بیتی تبدیل می‌شوند. خروجی مربوط به رمزگشایی به صورت زیر نمایش داده می‌شود:

arduino
CopyEdit

```
=== Decoding Stage ===
Block 0: 6-bit 110010 -> Decoded 5-bit 10011
Block 1: 6-bit 010011 -> Decoded 5-bit 10010
Block 2: 6-bit 111110 -> Decoded 5-bit xxxxx
Block 3: 6-bit 010100 -> Decoded 5-bit xxxxx
...
```

- **توضیح:**
  - برای هر بلوک، ابتدا 6 بیت دریافت‌شده نمایش داده می‌شود.
  - سپس تابع رمزگشایی مقدار 5 بیتی متناظر را بازمی‌گرداند.
  - اگر مقدار دریافتی با هیچ یک از کدهای معتبر موجود در جدول تطابق نداشته باشد، مقدار بازگشتی به صورت `xxxxx` (نامعتبر) نمایش داده می‌شود.
  - به عنوان مثال، در Block 2 و Block 3، خروجی به دلیل خطا در انتقال (تزریق خطا و تغییر بیت‌ها) قابل رمزگشایی صحیح نیست؛ بنابراین مقدار `xxxxx` نمایش داده شده است.

---

### ۶. مقایسه نهایی (Mismatch Report)

پس از رمزگشایی، داده‌ی بازترکیب‌شده (130 بیت) با داده اصلی (به جز بیت‌های کنترلی) مقایسه می‌شود. در خروجی، هر خطا به تفصیل نمایش داده شده است:

arduino
CopyEdit

```
Mismatch at bit 3: Original 1, Decoded 0
Mismatch at bit 5: Original 1, Decoded 0
...
Mismatch at bit 129: Original 0, Decoded 1
```

- **توضیح:**
  - هر خط "Mismatch" نشان می‌دهد که در آن بیت از داده اصلی، مقدار دریافتی پس از رمزگشایی با مقدار اصلی متفاوت است.
  - به عنوان مثال، در بیت 3، مقدار اصلی 1 بوده ولی مقدار رمزگشایی شده 0 است.
  - تعدادی از بیت‌ها به دلیل خطاهای انباشته (ابتدا در انتقال مرحله اول، سپس در رمزگذاری و انتقال بلوکی) دچار اختلاف شده‌اند.
  - همچنین بیت‌هایی که در خروجی به عنوان `x` نمایش داده شده‌اند، به دلیل عدم تطابق کدهای 6 بیتی (نتیجه تزریق خطا) قابل تفسیر صحیح نیستند.

در انتها، تعداد خطاهای شناسایی‌شده نمایش داده می‌شود:

yaml
CopyEdit

```
=== Transmission Complete ===
Data bit errors: 71
First transmission CRC errors: 1
Second transmission CRC errors: 18
```

- **توضیح:**
  - **Data bit errors:** تعداد کل بیت‌هایی است که پس از رمزگشایی با داده اصلی مطابقت ندارند (در این مورد 71 بیت).
  - **First transmission CRC errors:** تعداد خطاهایی که در مرحله‌ی اول (انتقال 130 بیتی) شناسایی شده است (به دلیل تغییر بیت 50، مقدار 1).
  - **Second transmission CRC errors:** تعداد بلوک‌های 6 بیتی که در انتقال دوم خطا داشته‌اند (18 بلوک از 26 بلوک).

---

### جمع‌بندی توضیحات خروجی

1. **داده اصلی تولید شده** به صورت یک رشته 130 بیتی نمایش داده می‌شود.
2. **انتقال اولیه:**
   - بیت به بیت انتقال داده می‌شود.
   - بیت 50 معکوس شده که باعث ایجاد تفاوت در CRC کل 130 بیت می‌شود.
3. **محاسبه CRC در انتقال اول:**
   - تفاوت CRC اصلی و دریافتی نشان‌دهنده وجود خطا (نتیجه تغییر بیت 50) است.
4. **رمزگذاری 5B/6B:**
   - داده به بلوک‌های 5 بیتی تقسیم شده و هر بلوک به 6 بیت تبدیل می‌شود.
5. **انتقال دوم:**
   - برای هر بلوک 6 بیتی، یک بیت به صورت تصادفی معکوس می‌شود.
   - برای هر بلوک، CRC محاسبه شده و در بسیاری از بلوک‌ها CRC دریافتی با CRC اصلی تفاوت دارد (ماده‌ی پیام‌های CRC MISMATCH).
6. **رمزگشایی:**
   - بلوک‌های 6 بیتی دریافت‌شده به 5 بیت تبدیل می‌شوند.
   - در برخی بلوک‌ها به دلیل خطاهای ناشی از تزریق خطا، رمزگشایی صحیح انجام نشده و نتیجه `xxxxx` به عنوان خروجی نمایش داده می‌شود.
7. **مقایسه نهایی:**
   - در مقایسه بیت به بیت، تعداد زیادی اختلاف (Mismatch) بین داده اصلی و داده رمزگشایی شده مشاهده می‌شود.
   - در نهایت، مجموع خطاهای شناسایی شده (Data bit errors: 71) به همراه خطاهای CRC در هر دو مرحله گزارش می‌شود.

این خروجی نشان می‌دهد که خطاهای تزریق‌شده (هم در انتقال اولیه و هم در بلوک‌های رمزگذاری‌شده) تأثیر قابل توجهی بر روی صحت داده نهایی داشته‌اند. این امر اهمیت استفاده از روش‌های تصحیح خطا و یا انتقال‌های مطمئن‌تر را در سیستم‌های واقعی نشان می‌دهد.