پروژه پایانی طراحی سیستم های دیجیتال

نام و نام خانوادگی: فرزاد محمدی
شماره دانشجویی: ۹۶۳۰۱۴۳

عنوان پروژه : پیاده سازی مدار کدگذار و کدبردار CRC[1]

در این پروژه قصد داریم علاوه بر پیاده سازی مدار های کدگذار و کدبردار دو آزمایش هم انجام دهیم. آزمایش اول بدین صورت است که خروجی مدار کدگذار را بدون ایجاد خطا به ورودی مدار کدبردار وصل میکنیم تا از عملکرد مجموعه مطمئن شویم، سپس در آزمایش دوم تعدادی خطا در خروجی مدار کدگذار ایجاد میکنیم و سپس رشته بیت حاوی خطا را به ورودی مدار کدبردار وصل میکنیم که در این حالت نیز نتیجه باید همانند نتیجه آزمایش اول باشد.

روند گزارش به صورت زیر خواهد بود:

1. **شرح توضیحات مربوط به پیاده سازی مدار کدگذار**
2. **شرح توضیحات مربوط به پیاده سازی مدار کدبردار**
3. **توضیحات مربوط به انجام آزمایش اول**
4. **توضیحات مربوط به انجام آزمایش دوم**

پیاده سازی مدار کدگذار

به صورت خلاصه فرایند کدگذاری به صورت زیر است:

1. **انتخاب چندجمله ای مورد (هر چندجمله ای خواص خاص خود نظیر توان تشخیص تعداد بیت مشخص، توان تصحیح تعداد بیت مشخص و تعداد بیت اضافی مورد نیاز برای کدگذاری)**
2. **به رشته بیت پیام به تعداد یکی کمتر از رشته بیت چند جمله ای مولد صفر اضافه کنیم.**
3. **رشته بیت حاصل را تقسیم بر چند جمله ای مورد کنیم**
4. **باقی مانده تقسیم بیت های لازم برای کدگذاری هستند که باید به ابتدا یا انتهای رشته بیت پیام اضافه شود.**

نکات:

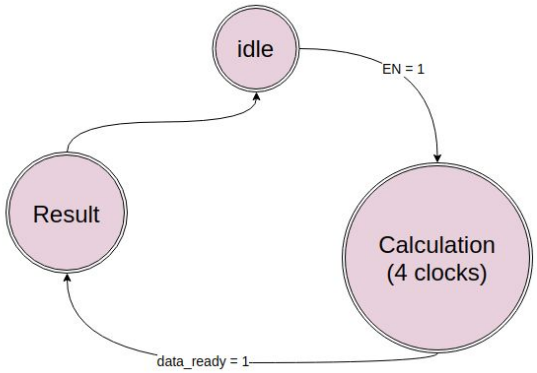
- **تقسیم مطرح شده در بند شماره ۳، یک تقسیم باینری است.**
- **در این پروژه چندجمله ای Hamming را به عنوان چندجمله ای مورد استفاده کرده ایم.**
  - **این چندجمله ای برابر $g(x) = x^3 + x + 1$ میباشد که آن را با رشته بیت ۱۰۱۱ در پیاده سازی نشان داده ایم (در واقع چندجمله ای های مورد دو طرز نمایش دارند یکی به شکل چندجمله ای و دیگری به صورت یک رشته بیت که همان ضرایب در چندجمله ای میباشند)**

[1] Cyclic Redundancy Code

- در این پروژه دنباله های پیام ۴ بیت هستند که با اضافه شدن ۳ بیت دیگر که توسط مدار کدگذار تعیین میشود، میتوانیم رشته بیت های پیام را کد کنیم
- این چندجمله ای توانایی شناسایی ۳ خطا یا تصحیح ۱ خطا را در رشته بیت دریافتی (که شامل ۷ بیت میباشد) دارد. (نحوه شناسایی و تصحیح در بخش کدبرداری توضیح داده میشود)

در اکثر پیاده سازی های انجام شده برای مدار کدگذار، مرحله شماره ۳ در فرایند تعریف شده بالا (یعنی همان تقسیم باینری) با استفاده از یک LFSR[2] انجام می شود ولی در این پروژه برای شفاف بودن عملیات کدگذاری، فرایند تقسیم را به همان صورتی که انسان ها روی کاغذ فرایند تقسیم را انجام میدهند، پیاده سازی را انجام داده ایم.

ماشین حالت مدار کدگذار

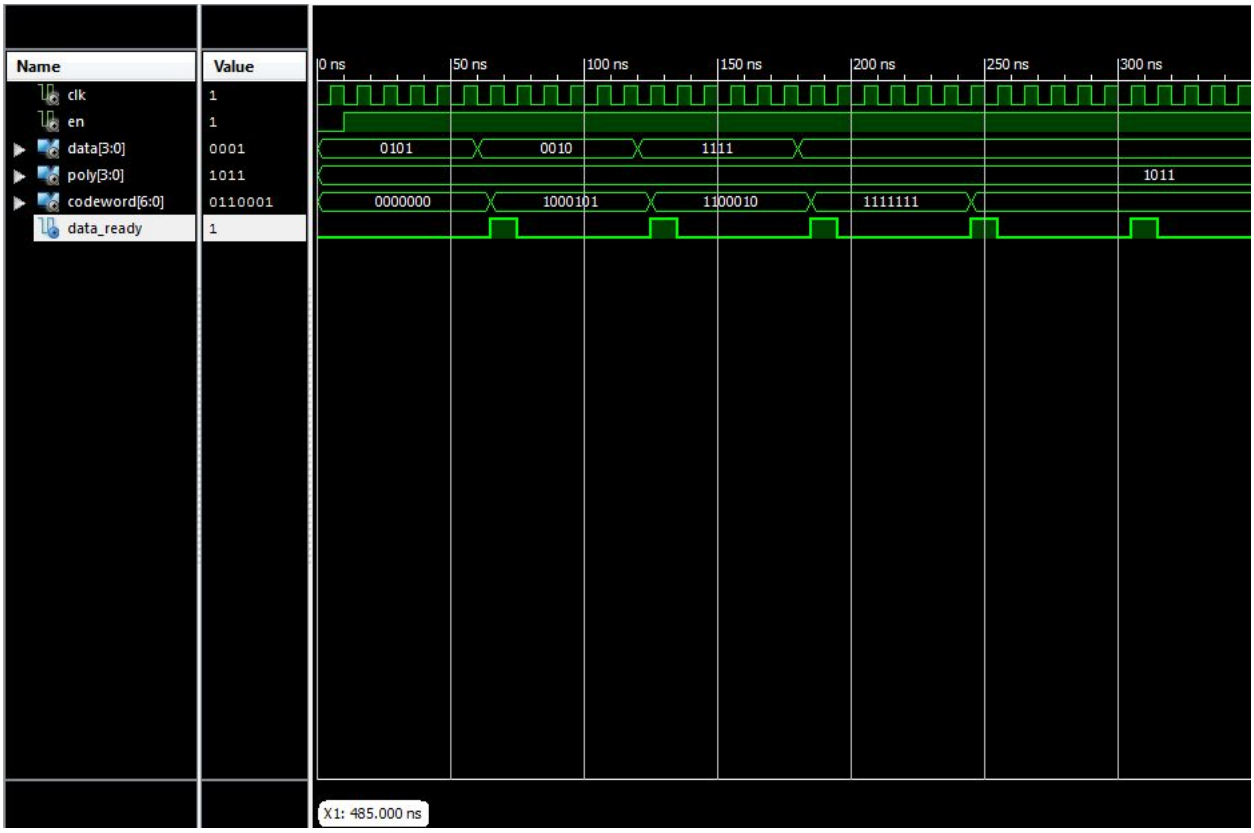


اجزای استخراج شده برای مدار کدگذار

```
Synthesizing Unit <m_encoder_2>.
    Related source file is "C:\ISE\FinalProject\m_encoder_2.vhd".
    Found 4-bit register for signal <data_int>.
    Found 4-bit register for signal <poly_int>.
    Found 7-bit register for signal <cal_reg>.
    Found 3-bit register for signal <msb>.
    Found 1-bit register for signal <data_ready_int>.
    Found 7-bit register for signal <output_int>.
    Found 2-bit register for signal <p_state>.
    Found finite state machine <FSM_0> for signal <p_state>.
    -----------------------------------------------------------------------
    | States             | 3                                              |
    | Transitions        | 5                                              |
    | Inputs             | 2                                              |
    | Outputs            | 3                                              |
    | Clock              | clk (rising_edge)                              |
    | Power Up State     | idle                                           |
    | Encoding           | auto                                           |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 4-bit subtractor for signal <n0086[3:0]> created at line 77.
    Found 3-bit subtractor for signal <lsb> created at line 45.
    Found 1-bit 7-to-1 multiplexer for signal <msb[2]_X_5_o_Mux_11_o> created at line 80.
    Summary:
        inferred   2 Adder/Subtractor(s).
        inferred  26 D-type flip-flop(s).
        inferred 105 Multiplexer(s).
        inferred   1 Finite State Machine(s).
Unit <m_encoder_2> synthesized.
```

[2] Linear-Feedback Shift Register

شبیه سازی مدار کدگذار به صورت مجزا



پیاده سازی مدار کدبردار

به صورت خلاصه فرایند کدبرداری به صورت زیر است:

1. **رشته بیت دریافتی (که در این پروژه ۷ بیت میباشد) را بر چند جمله ای مولد تقسیم میکنیم. در اینجا با توجه به وضعیت باقیمانده (که به آن کد سندروم[3] میگوییم) دو حالت می تواند رخ دهد:**
   a. **اول اینکه سندروم تمام صفر باشد، که این به معنای آن است که در رشته بیت دریافتی خطایی وجود نداشته و به راحتی با حذف بیت های اضافی میتوان به بیت های پیام رسید.**
   b. **دوم اینکه سندروم غیر صفر باشد، که این به معنای آن است که خطایی در رشته بیت دریافتی وجود دارد**
2. **این مرحله فقط برای حالت سندروم غیر صفر است، در اینجا باید با استفاده از سندروم بدست آمده و جدول lookup که از قبل در مدار کدبردار محاسبه شده (در ادامه در این رابطه توضیحات ارائه خواهد شد)، بردار خطا متناظر با سندروم را با رشته بیت دریافتی XOR کنیم.**

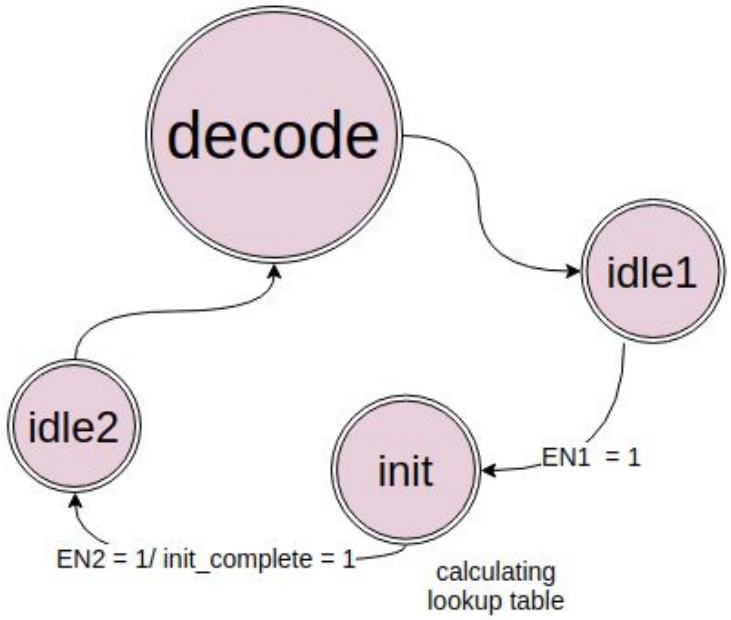
[3] Syndrom

3. با انجام عملیات مرحله قبل در صورتی که ۱ خطا در رشته بیت دریافتی وجود داشته باشد، آن خطا تصحیح شده و اکنون با حذف بیت های اضافی می توانیم به رشته بیت پیام برسیم. (اگر خطا بیشتر از ۱ مورد باشد با استفاده از این چندجمله ای مورد انتخاب شده نمی توانیم آن را تصحیح کنیم)

نحوه محاسبه جدول lookup شامل بردار های خطا

به صورت کلی در یک رشته بیت ۷ تایی در ۷ جایگاه میتواند خطا رخ دهد، که اگر محل خطا را با ۱ و محل هایی که خطا وجود ندارد را با ۰ نمایش دهیم بردار خطای مربوط به آن جایگاه بدست می آید. به عنوان مثال بردار ۰۱۰۰۰۰۰ بردار خطای مربوط به خطا در بیت در جایگاه ۶ میباشد.
با تقسیم هر یک از این بردار های خطا بر چند جمله ای مولد سندروم مربوط به آن محاسبه میشود، که پس از محاسبه می توانیم آن ها را در یک جدول lookup ذخیره کنیم.

ماشین حالت مدار کدبردار



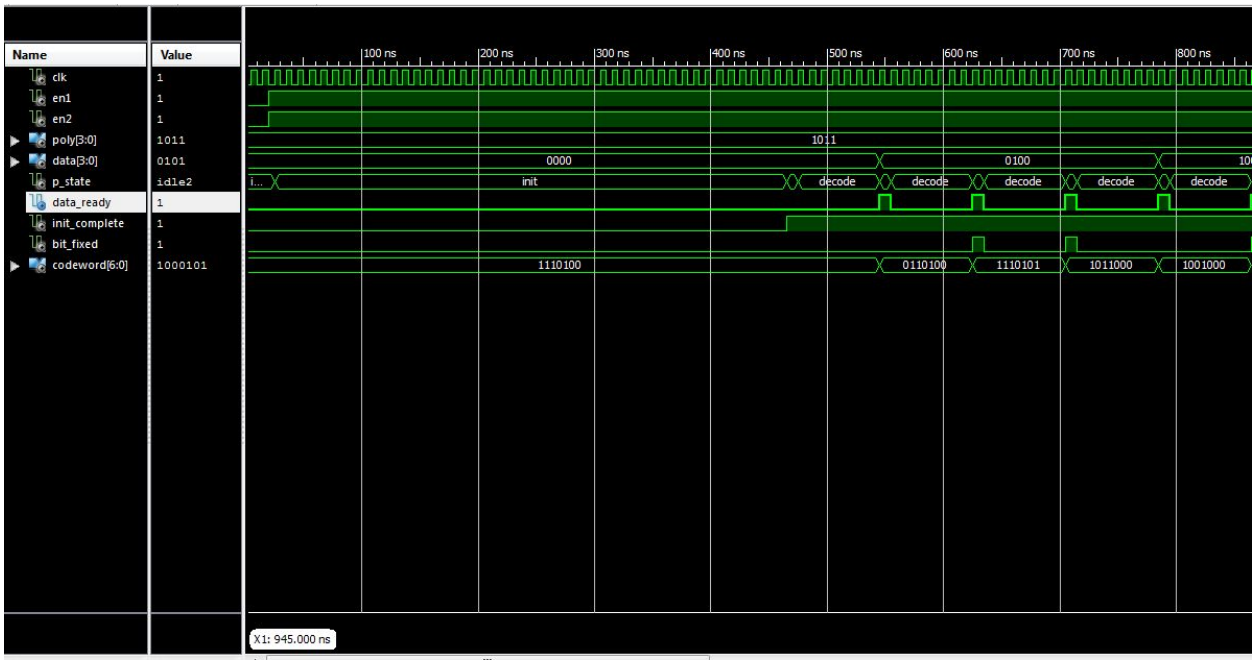
دلیل اینکه در ماشین حالت مدار کدبردار ۲ حالت idle وجود دارد این است که یکی منتظر فعال شدن سیگنال لازم برای محاسبه lookup می ماند (چون امکان دارد چند جمله ای مولد تغییر کند) و دیگری منتظر دریافت سیگنال لازم برای شروع فرایند کدبرداری میماند.

بخش های استخراج شده از مدار کدبردار

```
Synthesizing Unit <m_decoder>.
    Related source file is "C:\ISE\FinalProject\m_decoder.vhd".
    Found 8x7-bit single-port RAM <Mram_lookup> for signal <lookup>.
    Found 2-bit register for signal <p_state>.
    Found 4-bit register for signal <poly_int>.
    Found 1-bit register for signal <en_int>.
    Found 7-bit register for signal <codeword_int>.
    Found 7-bit register for signal <error_vec>.
    Found 7-bit register for signal <error_vec2>.
    Found 4-bit register for signal <count>.
    Found 3-bit register for signal <count2>.
    Found 1-bit register for signal <dec_ready>.
    Found 1-bit register for signal <bit_fix_int>.
    Found 4-bit register for signal <dec_out>.
    Found 1-bit register for signal <init_com_int>.
    Found finite state machine <FSM_0> for signal <p_state>.
    -----------------------------------------------------------------------
    | States             | 4                                              |
    | Transitions        | 9                                              |
    | Inputs             | 5                                              |
    | Outputs            | 4                                              |
    | Clock              | clk (rising_edge)                              |
    | Power Up State     | idle1                                          |
    | Encoding           | auto                                           |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 3-bit subtractor for signal <GND_5_o_GND_5_o_sub_7_OUT<2:0>> created at line 111.
    Found 4-bit subtractor for signal <GND_5_o_GND_5_o_sub_13_OUT<3:0>> created at line 123.
    Found 1-bit 4-to-1 multiplexer for signal <p_state[1]_GND_5_o_Mux_39_o> created at line 84.
    Summary:
        inferred   1 RAM(s).
        inferred   2 Adder/Subtractor(s).
        inferred  40 D-type flip-flop(s).
        inferred  18 Multiplexer(s).
        inferred   1 Finite State Machine(s).
Unit <m_decoder> synthesized.
```

```
Synthesizing Unit <m_syndrom_generator>.
    Related source file is "C:\ISE\FinalProject\m_syndrom_generator.vhd".
    Found 2-bit register for signal <p_state>.
    Found 4-bit register for signal <poly_int>.
    Found 7-bit register for signal <cal_reg>.
    Found 3-bit register for signal <msb>.
    Found 3-bit register for signal <syndrom_int>.
    Found 1-bit register for signal <data_ready_int>.
    Found finite state machine <FSM_1> for signal <p_state>.
    -----------------------------------------------------------------------
    | States             | 3                                              |
    | Transitions        | 5                                              |
    | Inputs             | 2                                              |
    | Outputs            | 3                                              |
    | Clock              | clk (rising_edge)                              |
    | Power Up State     | idle                                           |
    | Encoding           | auto                                           |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 4-bit subtractor for signal <n0080[3:0]> created at line 76.
    Found 3-bit subtractor for signal <lsb> created at line 44.
    Found 1-bit 7-to-1 multiplexer for signal <msb[2]_X_6_o_Mux_10_o> created at line 79.
    Summary:
        inferred   2 Adder/Subtractor(s).
        inferred  18 D-type flip-flop(s).
        inferred 105 Multiplexer(s).
        inferred   1 Finite State Machine(s).
Unit <m_syndrom_generator> synthesized.
```

شبیه سازی مدار کدبردار به صورت مجزا



آزمایش اول

نتایج این آزمایش در جدول زیر قابل مشاهده است:

ورودی کدگذار	خروجی کدگذار و ورودی کد بردار	خروجی کد بردار
1000	1011000	1000
0100	1110100	0100
1100	0101100	1100
1001	1101001	1001
1011	0001011	1011
0111	0100111	0111
0001	0110001	0001

فایل های انجام این آزمایش در پیوست ارسال میشود.

آزمایش دوم

نتایج این آزمایش در جدول زیر قابل مشاهده است:

خروجی کدبردار	ورودی کدبردار	خروجی کدگذار	ورودی کدگذار
1000	1011001	1011000	1000
0100	1010100	1110100	0100
1100	0111100	0101100	1100
1001	0101001	1101001	1001
1011	1001011	0001011	1011
0111	0101111	0100111	0111
0001	0111001	0110001	0001

در ستون سوم بیت های با رنگ قرمز نسبت به ستون دوم تغییر کرده اند.

فایل های انجام این آزمایش در پیوست ارسال میشود.