[[CRC] Bài 3 - Bộ tính CRC song song toàn phần](https://nguyenquanicd.blogspot.com/2019/08/crc-bai-3-bo-tinh-crc-song-song.html)

Tác giả [Nguyễn Quân](https://www.blogger.com/profile/11739939602738380602) at 07:49  [background](https://nguyenquanicd.blogspot.com/search/label/background?&max-results=5), [CRC](https://nguyenquanicd.blogspot.com/search/label/CRC?&max-results=5), [Kiến Thức Cơ Bản](https://nguyenquanicd.blogspot.com/search/label/Ki%E1%BA%BFn%20Th%E1%BB%A9c%20C%C6%A1%20B%E1%BA%A3n?&max-results=5), [Synchronous Design](https://nguyenquanicd.blogspot.com/search/label/Synchronous%20Design?&max-results=5), [Verilog](https://nguyenquanicd.blogspot.com/search/label/Verilog?&max-results=5)  [0 bình luận](https://nguyenquanicd.blogspot.com/2019/08/crc-bai-3-bo-tinh-crc-song-song.html#comment-form)

*Bài viết này trình bày về cách thiết kế bộ tính CRC song song toàn phần. Với cách thiết kế này, giá trị CRC có thể có ngay lập tức trong cùng chu kỳ với ngõ vào hoặc chỉ delay một vài chu kỳ để giúp timing đường dữ liệu của bộ tính CRC tốt hơn.*

**1) Phân tích vấn đề và mô tả thiết kế**

[Bài 1](http://nguyenquanicd.blogspot.com/2017/10/basic-knowledgecrc-bai-1-ly-thuyet-ve.html) và [bài 2](http://nguyenquanicd.blogspot.com/2019/08/crc-bai-2-logic-tinh-crc-noi-tiep-voi.html" \t "_blank) trình bày về hai thiết kế CRC nối tiếp. Thiết kế CRC nối tiếp có ưu điểm là tài nguyên ít và mạch tổ hợp giữa các Flip-Flop không lớn. Tuy nhiên, vấn đề lớn của việc tính CRC nối tiếp chính là tốc độ tính toán càng chậm khi số bit dữ liệu càng dài.

Để giải quyết vấn đề trên, một cách thiết kế khác là tính CRC song song. Nguyên tắc cơ bản của việc tính CRC song song là xếp chồng nhiều mạch tổ hợp tính CRC lên nhau. Ngõ ra của mạch tổ hợp thứ nhất sẽ là ngõ vào của mạch tổ hợp thứ hai và cứ tiếp tục như vậy cho đến khi giá trị CRC được tính hoàn chỉnh.

Điểm khác biệt chính giữa tính CRC nối tiếp và CRC song song là ở số lượng mạch tổ hợp tính toán CRC. Quay lại thiết kế ở [bài 2](http://nguyenquanicd.blogspot.com/2019/08/crc-bai-2-logic-tinh-crc-noi-tiep-voi.html" \t "_blank), việc tính nối tiếp dùng một thanh ghi dịch để lưu dữ liệu trong suốt quá trình tính toán. Tại mỗi chu kỳ giá trị thanh ghi dịch được đưa qua một mạch tính cố định. Mạch này dựa trên bit MSB của thanh ghi dịch để tính giá trị kế tiếp nạp vào thanh ghi dịch. Cách thiết kế này chỉ cần 1 mạch tính CRC.

|  |
| --- |
| [A diagram of a computer system  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEiOjYjZXFSaRSEvj9qMFR9arXPlOBldzsgdM5tlOy-Nl-ciIwvYw8ob59gN_jtUYyvurrtNSni9iTa2HGtMNwXQAst0ystNh-2A0sJSUl0Q2jDsFGFD2upriXtixfWB0Buo1H16HBiThpYW/s1600/image.png) |
| Hình 1: Mạch nguyên lý tính CRC nối tiếp chỉ dùng một mạch tổ hợp cho việc tính giá trị CRC kế tiếp |

Phần mạch tổ hợp dùng cho việc tạo giá trị kế tiếp cho thanh ghi dịch tính CRC được đóng khung trong hình trên. Chú ý, các mạch tổ hợp khác là để điều khiển quá trình tính toán CRC. Với thiết kế ở bài 2, nếu dữ liệu cần tính là 16 bit thì bộ tính toán cần 16 chu kỳ để tính được CRC.

Đối với CRC song song, thay vì chờ kết quả tính toán của chu kỳ trước nạp vào *crcSeq*rồi với tiếp tục tính cho chu kỳ sau thì nhiều mạch tổ hợp sẽ được xếp chồng nối tiếp với nhau để tạo ra giá trị CRC cuối cùng. Nếu dữ liệu đầu vào là 16 bit thì số lượng mạch tổ hợp cần dùng là 16 mạch, có cấu trúc như mạch tổ hợp dùng để tính CRC nối tiếp.

|  |
| --- |
| [A computer screen shot of a diagram  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjwkodC9J24_H43xgO-guJlmKMVNC_NE4R6uQKWBc3QBL6f-JEjd0BTJeDyRDR5bVZp_IM4bS9BEiLX5puvwa_3iqN7DLeZj4CpWTMurxWVMbbdtj81Y51_TnHN6gV3orF0iboGHlqFjX0b/s1600/image.png) |
| Hình 2: Mạch nguyên lý tính CRC song song, minh họa cho trường hợp số bit dữ liệu cần tính là 16 và số bit của chuỗi CRC là 8 |

Việc tính CRC song song sẽ gồm các phần sau:

1. Thanh ghi lưu lưu dữ liệu ngõ vào (Store input data): thanh ghi này lưu lại dữ liệu cần tính mỗi khi có yêu cầu tính CRC. Nếu có thanh ghi này làm cho việc tính CRC trễ 1 chu kỳ. Nếu không dùng thanh ghi này, dữ liệu ngõ vào sẽ đưa trực tiếp đến các mạch tổ hợp dùng để tính CRC và không bị trễ 1 chu kỳ.
2. Tạo ngõ vào cho mạch tổ hợp tính CRC (Create input): Ngõ vào này gồm các bit dữ liệu và các bit 0 được thêm vào. Số lượng bit 0 thêm vào bằng độ dài chuỗi CRC.
3. Mạch tổ hợp tính CRC song song (Parallel calculation): Ngõ ra của mạch tính toán trước, ngõ ra của MUX, là ngõ vào của mạch tính toán kế tiếp, MUX kế tiếp. Chú ý, vòng tròn "XOR GP" trong hình là mạch XOR giữa kết quả tính của 1 tầng cộng 1 bit thêm với đa thưc sinh.
4. Thanh ghi lưu kết quả CRC (Latch CRC result): Thanh ghi này lấy giá trị của mạch tính cuối cùng (mạch này không bao gồm MUX điều khiển bởi *crcReady*). Nếu không dùng thanh ghi này, kết quả tính CRC có thể lấy trực tiếp từ mạch tổ hợp cuối cùng bằng cách loại bỏ MUX điều khiển bởi *crcReady*.

Như vậy, nếu không sử dụng thanh ghi để chốt dữ liệu ngõ vào và chốt giá trị CRC ngõ ra thì bộ tính CRC sẽ cho kết quả ngay lập tức, cùng chu kỳ với giá trị ngõ vào. Tuy nhiên điều này có thể làm timing của bộ tính không tốt vì mạch tổ hợp lớn nên delay trên mạch cao. Mạch tính này càng lớn khi số lượng bit đầu vào càng cao.

Nếu chỉ sử dụng một thanh ghi để chốt dữ liệu đầu vào hoặc chốt giá trị CRC đầu ra thì giá trị CRC sẽ xuất hiện sau ngõ vào 1 chu kỳ.

Nếu sử dụng cả thanh ghi chốt ngõ vào và ngõ ra thì bộ tính CRC cần 2 chu kỳ để tính toán nhưng điều này sẽ giúp timing bộ tính tốt hơn. Tùy vào vị trí mà bộ tính CRC được tích hợp trong hệ thống mà bạn có cách lựa chọn thiết kế phù hợp.

Trong bài viết này, tác giả chọn cách thiết kế có cả FF chốt ngõ vào và FF chốt ngõ ra.

**3) Mô tả chi tiết thiết kế**

Thiết kế gồm các tín hiệu giao tiếp như bài 2:

* *clk*: clock đồng bộ
* *rstN*: reset tích cực mức thấp
* *ctrlEn*: Enable bộ tính CRC. Tích cực trong 1 chu kỳ để báo có dữ liệu cần tính CRC. Dữ liệu sẽ được đưa vào trên dataIn.
* *dataIn*: Dữ liệu cần tính CRC
* *genPoly*: Giá trị đa thức sinh sử dụng để tính CRC
* *crcReady*: Tích cực mức 1 khi bộ tính CRC rảnh hoặc sau khi tính xong CRC
* *crcSeq*: Kết quả chuỗi CRC

Ngoài mạch nguyên lý như đã trình bày ở phần tổng quan, thiết kế còn có 2 mạch khác là thanh ghi lưu lại giá trị đa thức sinh và bit lái ngõ ra *crcReady*.

|  |
| --- |
| [A diagram of a computer generated structure  Description automatically generated with medium confidence](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgLXLVaEVs2zmYsh1aUqVBdt8C64RWRq_nWmUaRiabEFy_pCEdvUWXbw5Wasv5S8oQ1maKSDVNf221kqv2Ap1puPDSUXVaITL7cIKo0STKorWLEmiFSG-tFxIBQP07NWTMgF-jDnwaVgIW5/s1600/image.png) |
| Hình 3: Thanh ghi lưu giá trị đa thức sinh khi ctrlEn=1 |

Thanh ghi đa thức sinh sẽ là ngõ vào của các bộ "XOR GP" dùng ở mạch tính CRC.

|  |
| --- |
| [A diagram of a block diagram  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgzER8tF8rdc5NLoz4cXCkqlhmCw_H_IGAZJ6d_2cL6F14lgG5UbZZ4pjgEtyshyphenhyphenOrKDQ0DhxGN6R6lfPAoykFXjOD3FH4DIKsZglcg_Q44e6EjjeU5sfmJ7mTSB82y2RRhhyphenhyphenR9OmQKGG_u/s1600/image.png) |
| Hình 4: Mạch nguyên lý của tín hiệu crcReady |

Vì việc tính CRC trong 2 chu kỳ nên mỗi lần tính, *ctrlEn=1* trong 1 chu kỳ, thì *crcReady*chỉ bằng 0 trong 1 chu kỳ.

Về RTL code, phần hơi rắc rối là phần code cho mạch tổ hợp tính CRC. Phần mạch này chỉ khó ở việc lựa chọn đúng vị trí bit cho vòng lặp tạo mạch tổ hợp tính CRC. Các bạn hãy tải code về và so sánh với mạch nguyên lý hình 2 để hiểu rõ hơn.

*Ví dụ 1: RTL code mạch tổ hợp tính CRC*

generate

genvar i;

assign subCrc[TMP\_WIDTH-1:(TMP\_WIDTH-1)-(CRC\_WIDTH-1)]

= dataInCal[DWIDTH+CRC\_WIDTH-1]?

dataInCal[DWIDTH+CRC\_WIDTH-2:

(DWIDTH+CRC\_WIDTH-1)-(CRC\_WIDTH-1)-1]

^ GenPolyReg[CRC\_WIDTH-1:0]

:dataInCal[DWIDTH+CRC\_WIDTH-2:

(DWIDTH+CRC\_WIDTH-1)-(CRC\_WIDTH-1)-1];

for (i=1; i < DWIDTH; i=i+1) begin: CrcCal

assign subCrc[TMP\_WIDTH-1-(i\*CRC\_WIDTH):

(TMP\_WIDTH-1)-(i\*CRC\_WIDTH)-(CRC\_WIDTH-1)]

= subCrc[TMP\_WIDTH-1-(i-1)\*CRC\_WIDTH]?

{subCrc[(TMP\_WIDTH-1)-((i-1)\*CRC\_WIDTH)-1:

(TMP\_WIDTH-1)-(i\*CRC\_WIDTH-1)],

dataInCal[(DWIDTH+CRC\_WIDTH-1)-CRC\_WIDTH-i]}

^ GenPolyReg[CRC\_WIDTH-1:0]

:{subCrc[(TMP\_WIDTH-1)-((i-1)\*CRC\_WIDTH)-1:

(TMP\_WIDTH-1)-(i\*CRC\_WIDTH-1)],

dataInCal[(DWIDTH+CRC\_WIDTH-1)-CRC\_WIDTH-i]};

end

endgenerate

Trong đoạn code trên:

* CRC\_WIDTH: Số bit chuỗi CRC
* DWIDTH: Số bit dữ liệu cần tính CRC
* TMP\_WIDTH = DWIDTH\*CRC\_WIDTH

**4) Kết quả mô phỏng**  
Source có kèm 1 testbench đơn giản minh họa việc tính CRC. Khi chạy testbench này, kết quả tính toán sẽ như sau:

|  |
| --- |
| [A close-up of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEg4P-819pG9dR7HycuTbRg8nSNGOY9qX-05YFSFBEwEfUMYvb79a9HTI3x2ahG4919YXIgvLjzmzAd47vL2pp08sP7OTet177mz6xg0a3byLR4q7g7zPzYxx9_2DFJVGHxvfSiWDRR7M2WL/s1600/crcParallel_nguyenquanicd_3.png) |
| Hình 5: Kết quả tính toán trên cửa sổ Transcript của QuestaSim |

|  |
| --- |
| [A diagram of a cell phone  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEg83R3bEd3aL7ckR_jaMV7MjaM3EjylOdmORbvRGDTai6HxHyp7lJ_E-P_F6diFXxOcy8aysphbEdv88C6cickQXF9uaPUs_B0cP_LXH2gDXGSPT7veE1UzHrnEHn084Tup94y0hIIJOlAB/s1600/crcParallel_nguyenquanicd_2.png) |
| Hình 6: Waveform của testbench |

Ở đây, một đa thức sinh 8 bit được sử dụng là: 0x07  
Ba chuỗi dữ liệu, mỗi chuỗi 16 bit có giá trị lần lượt là:

* 0x0102 -> CRC là 0x1b
* 0xa522 -> CRC là 0xb7
* 0xf0e5 -> CRC là 0xa1

**5) Nhận xét:**  
Cách tính CRC song song toàn phần cho kết quả nhanh nhất nhưng tốn nhiều tài nguyên và mạch tổ hợp lớn. Chú ý, việc delay 0, 1, 2 hay nhiều chu kỳ hơn hoàn toàn cho bạn có thể quyết định. Nó tùy vào việc bạn chia đường dữ liệu của mạch tính CRC thành bào nhiêu đoạn để chèn FlipFlop.

*Web tính CRC onlline dùng để kiểm chứng kết quả:*  
<https://crccalc.com/>  
  
*Dữ liệu có thể download:*  
[Source code trên Github](https://github.com/nguyenquanicd/crcCalculator)