[[CRC] Bài 2 - Bộ tính CRC nối tiếp với một giá trị khởi tạo](https://nguyenquanicd.blogspot.com/2019/08/crc-bai-2-logic-tinh-crc-noi-tiep-voi.html)

Tác giả [Nguyễn Quân](https://www.blogger.com/profile/11739939602738380602) at 08:29  [CRC](https://nguyenquanicd.blogspot.com/search/label/CRC?&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-2-logic-tinh-crc-noi-tiep-voi.html#comment-form)

*Trong [bài 1](http://nguyenquanicd.blogspot.com/2017/10/basic-knowledgecrc-bai-1-ly-thuyet-ve.html), đã được viết cách đây rất lâu, tác giả có trình bày ngắn gọn về cách tính CRC (Cyclic Redundancy Code), nguyên lý tính CRC nối tiếp và trình bày một cách thiết kế bộ tính CRC nối tiếp. Bài viết này tiếp tục mô tả một cách thiết kế khác của bộ tính CRC nối tiếp. Các thiết kế này có chút ưu điểm hơn so với cách đã thực hiện ở [bài 1](http://nguyenquanicd.blogspot.com/2017/10/basic-knowledgecrc-bai-1-ly-thuyet-ve.html). Chuỗi bài viết này được tiếp tục vì có bạn hỏi về CRC làm tác giả chợt nhớ là bỏ bê hơi lâu.*

**1) Phân tích vấn đề**

Vấn đề chính của bộ tính CRC nối tiếp chính là "thời gian tính toán lớn". Thời gian tính toán của bộ tính CRC nối tiếp trình bày ở [bài 1](http://nguyenquanicd.blogspot.com/2017/10/basic-knowledgecrc-bai-1-ly-thuyet-ve.html" \t "_blank) là số chu kỳ xung clock tính bằng số bit chuỗi dữ liệu cần tính cộng số bit chuỗi CRC. Như vậy, nếu chuỗi dữ liệu càng dài và chuỗi CRC càng dài thì số chu kỳ cần để tính được CRC càng nhiều.

Nhược điểm của bộ tính CRC nối tiếp ở bài 1 là việc tính CRC bắt đầu khi thanh ghi dịch CRC,*crc\_seq[CRC\_GPW\_MAX-1:0]*, bằng 0. Khi *ctrl\_en=1*, từng bit dữ liệu mới bắt đầu được dịch vào thanh ghi *crc\_seq[CRC\_GPW\_MAX-1:0]*. Điều này làm cho việc tính toán mất một số chu kỳ đầu không cần thiết, số chu kỳ này bằng đúng độ dài đa thức sinh, bằng *CRC\_GPW\_MAX* là số bit đa thức sinh. Xét lại nguyên tắc tính CRC:

* Nếu bit MSB của lần tính hiện tại bằng 1 thì nó sẽ được XOR (modulo-2) với đa thức sinh
* Nếu bit MSB của lần tính hiện tại bằng 0 thì nó sẽ không đổi

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEiz-a_xwO7GVgiNLuKUzJ717Ti2QDSuEqC_XazoVUPItOnDTyCg2SyQL9oP0Ar6-DaJOfJZBBm8Ctelr4gUP1GvsioeXZ21FJzuKFb_NrrfScG0XPkZz68GqQktGLWQI4l534q-f-mTK72-/s1600/image.png) |
| Hình 1: Minh họa cách tính CRC |

Theo nguyên tắc trên, khi bắt đầu tính CRC, thanh ghi dịch *crc\_seq[CRC\_GPW\_MAX-1:0]* có giá trị 0 nên trong CRC\_GPW\_MAX chu kỳ đầu tiên, bit MSB luôn bằng 0 và giá trị *crc\_seq* không XOR với giá trị đa thức sinh.

Để giảm số chu kỳ không cần thiết như đã nêu. Khi bắt đầu tính CRC, thanh ghi dịch tính CRC sẽ được nạp ngay các bit cao nhất của chuỗi dữ liệu cần tính. Số bit nạp bằng số bit của thanh ghi dịch và bằng số bit của chuỗi CRC.

**2) Mô tả tổng quan thiết kế**

Thiết kế này gồm các tín hiệu sau:

* 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

Các parameter của thiết kế:

* CRC\_WIDTH: Độ rộng chuỗi CRC
* DWIDTH: Độ rộng chuỗi dữ liệu cần tính CRC
* DWIDTH\_LSB: Số bit dữ liệu còn lại của chuỗi dữ liệu sau khi trừ đi số bit MSB được nạp vào thanh ghi dịch tính CRC.
* COUNTERW: Độ rộng bộ đếm số chu kỳ tính CRC

|  |
| --- |
| [A diagram of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEh5OERJjQ1QJMuah0BC1TewTB9Tshu4ZONp0R6HDylF2ZLwmVLr8Ag2aIBkQnwq2FlwzswMClnjvLHAPezeZkKJdiTnzTyx9spyECAOw8RFdCBuU3XYTHim-JUxaDI13jv0uY5vsKCqHvPr/s1600/image.png) |
| Hình 2: Sơ đồ tín hiệu giao tiếp của bộ tính CRC nối tiếp crcInitialValue |

Thiết kế này gồm các mạch logic sau:

* Một bit trạng thái điều khiển quá trình tính CRC, cho biết khi nào bộ đếm CRC đang hoạt động, *calStatus*.
* Một thanh ghi lưu lại giá trị đa thức sinh của mỗi lần tính, *genPolyReg*.
* Một thanh ghi lưu lại giá trị các bit thấp của *dataIn*, *dataInReg*. Các bit cao sẽ được nạp trực tiếp vào thanh ghi dịch CRC khi bắt đầu tính toán nên không cần thanh ghi để lưu.
* Một bộ đếm số chu kỳ tính toán CRC, *mCounter*.
* Logic tạo ngõ ra *crcReady*.
* Thanh ghi dịch tính CRC, *crcSeq*.

**3) Phân tích chi tiết**

**3.1) Bit trạng thái và logic của crcReady**

Bit trạng thái *calStatus=1*sau khi *ctrlEn=1*, nó báo hiệu quá trình tính CRC đang diễn ra. Bit này tự động xóa cùng với bộ đếm chu kỳ tính CRC khi quá trình CRC đã kết thúc, *clrmCounter=1*.

|  |
| --- |
| [A diagram of a block diagram  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEh4Yuh95Bpqc-9egerWHglMQtQf9vce3Gc6QPwE423MfQvvFcVf2Tht2wENE9kO2J9bH8jSO8EVTaTLgR_0pWqCmVGNRjHbw2xcQirxeQEliAk6BXqXtGz8RkSBgks6TuvVsqvE2CGsNdw3/s1600/image.png) |
| Hình 3: Logic của bit trạng thái và ngõ ra crcReady |

Ngõ ra *crcReady=1* khi bộ đếm CRC không hoạt động, tức *calStatus=0*.

**3.2) Thanh ghi lưu giá trị đa thức sinh**

Thanh ghi cập nhật giá trị khi *ctrlEn=1*.

|  |
| --- |
| [A diagram of a block diagram  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEiRCWY7xwajwnEYvJLym7lypZqF71rJORpkZNgbBjQA6IhaUDnPbCjKzs9gjZrTMAEu9vm-2UHPrM9hDYNQbdx9V2o1iVUblUQ33rlz8F4Vn7WhyVquPzXzi5fGkbwcLaR214k3PLuXYRUq/s1600/image.png) |
| Hình 4: Thanh ghi lưu lại giá trị đa thức sinh |

**3.3) Thanh ghi lưu các bit LSB của chuỗi dữ liệu**

Thanh ghi này lưu lại các bit LSB của chuỗi dữ liệu khi *ctrlEn=1*. Sau đó, nó sẽ dịch trái 1 bit trên mỗi chu kỳ tính CRC. Bit MSB, *dataInReg[DWIDTH\_LSB-1]*, sẽ được nối đến ngõ vào thanh ghi dịch tính CRC.

|  |
| --- |
| [A diagram of a data storage system  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhh6Q35wehnAgsbf7Wk4fNs5L7_AGFgoNqZVL8FJVhCXG7Ug2bfPEhWJpYsSWci-UYX-59q_EdsdDSEZNc27YzHj1xcnlbJAsLTa_dBG5XFkc3o8lNIeFHJqmlYIAumWWMcHobbunr1HvCU/s1600/image.png) |
| Hình 5: Thanh ghi lưu lại các bit LSB của chuỗi dữ liệu |

**3.4) Bộ đếm số chu kỳ tính toán CRC**

Bộ đếm tăng 1 đơn vị từ 0 đến DWIDTH-1 trong suốt quá trình tính toán CRC (calStatus=1). Bộ đếm sẽ xóa lại về 0 khi quá trình tính CRC kết thúc. Quá trình tính CRC kết thúc khi số chu kỳ tính toán bằng số bit dữ liệu cần tính.

|  |
| --- |
| [A diagram of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgaWITOMqbMV2S-kwz_ZlUomki1gmyDIC8qcqTD91n9kCd7Fq3VGr75FnAQWwZ2q8WdglaH1BcgYisRrvzTotQhTriDoNZfynHZTbb5wsL2EeQdCejlI-GIdk6NtbF-07d3KtCVFBe8nnNA/s1600/image.png) |
| Hình 6: Bộ đếm chu kỳ tính toán CRC |

**3.5) Thanh ghi dịch lưu giá trị CRC**

Thanh ghi crcSeq vừa là thanh ghi dịch tính CRC vừa là thanh ghi lưu lại giá trị CRC. Khi *ctrlEn=1*, thanh ghi nạp các bit cao của chuỗi dữ liệu từ *dataIn* với số bit bằng số bit chuỗi CRC.

Sau đó, thanh ghi bắt đầu quá trình tính toán (*calStatus=1*), tùy vào bit MSB của thanh ghi, *crcSeq[CRC\_WIDTH-1]*, mà giá trị mới của thanh ghi dịch có được XOR với đa thức sinh *genPolyReg* hay không. Chú ý, giá trị dùng để XOR là giá trị thanh ghi đã được dịch trái 1 bit và kèm với bit MSB của thanh ghi dữ liệu *dataInReg*.

|  |
| --- |
| [A diagram of a computer program  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgyCRqt43kWIZm7EFM_iXpxsjJicimourlHaGh5gTWMB8wJv-6O-OzY5v6XlT_E3bn25rUNImLohD69XF1PBlAEVspqx6EpbnoYIjG8bJi6RaD_kTjzN0PweuEJU_4ZbOu8bOuPftBjKvX3/s1600/image.png) |
| Hình 7: Thanh ghi CRC |

**4) Minh họa một kết quả mô phỏng**

Trong waveform này, *dataIn=16'h0a55* và đa thức sinh *genPoly=8'hD5*.

Kết quả tính toán được là *crcSeq=8'haa* khi *crcReady=1*.

|  |
| --- |
| [A graph with lines and text  Description automatically generated with medium confidence](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgzRHA6r2lHOkdMY9pBXn7Rv452JhD5tyZzvIau37J2or37T4lnsXZWLpQ9UrzEFY6rGNSyPf_Df4NYLjQt_Sc02RPt_0oJgwIyxVu5q1DT_OpdGWE4hywbglOzS_rsoqq_giwZ2_JqqSCV/s1600/crc_waveform.png) |
| Hình 8: Một waveform mô phỏng tính toán CRC |

RTL code cũng được tổng hợp trên phần mềm Quartus II.

|  |
| --- |
| [A diagram of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjyHeprzu7gmTFl18nwdSRTxWal_L7OvepO-8N7i91RIT5W6U6_-vx-hFZ56aiEL782R7ug0Ku9qYNdH858NBu68S4_RSabQphq5n8joZFxs5m8c2MUxwv6tx_mJRTId7KrQtaRx5rX8btu/s1600/crc_netlist.png) |
| Hình 9: Kết quả tổng hợp trên Quartus II |

**5) Nhận xét**

Tuy cách thiết kế này giảm được CRC\_WIDTH chu kỳ tính toán nhưng vẫn chưa phải là cách tính CRC nhanh nhất. Một số cách khác sẽ được trình bày trong những bài tiếp theo.