[[CRC] Bài 1 - Lý thuyết về CRC và mạch tính CRC nối tiếp](https://nguyenquanicd.blogspot.com/2017/10/basic-knowledgecrc-bai-1-ly-thuyet-ve.html)

Tác giả [Nguyễn Quân](https://www.blogger.com/profile/11739939602738380602) at 08:27  [background](https://nguyenquanicd.blogspot.com/search/label/background?&max-results=5), [CRC](https://nguyenquanicd.blogspot.com/search/label/CRC?&max-results=5), [IP core](https://nguyenquanicd.blogspot.com/search/label/IP%20core?&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), [System Verilog](https://nguyenquanicd.blogspot.com/search/label/System%20Verilog?&max-results=5), [Verilog](https://nguyenquanicd.blogspot.com/search/label/Verilog?&max-results=5)  [2 bình luận](https://nguyenquanicd.blogspot.com/2017/10/basic-knowledgecrc-bai-1-ly-thuyet-ve.html#comment-form)

*Nội dung bài viết này trình bày về lý thuyết tạo và và kiểm tra CRC. Phương pháp thực hiện mạch tính CRC nối tiếp. Đồng thời, bài viết đưa ra một RTL code tính CRC có thể lựa chọn độ dài đa thức sinh (generator polynomial) trước khi tổng hợp và gán giá trị đa thức sinh mong muốn khi sử dụng.*

**1. Tổng quan**

Việc truyền một dữ liệu trong một môi trường từ điểm này đến điểm khác, ví dụ như truyền dữ liệu giữa hay máy tính trong một mạng, luôn tiềm ẩn nhiều yếu tố làm dữ liệu truyền bị sai. Cơ chế phát hiện lỗi dữ liệu là không thể thiếu đối với các giao thức có độ tin cậy cao.

Phương pháp chung để kiểm tra lỗi dữ liệu là thêm các bit kiểm tra kèm theo dữ liệu được truyền theo một quy tắc đã được quy định trước. Bộ truyền dữ liệu sẽ tạo ra các bit kiểm tra từ giá trị dữ liệu cần truyền và gắn nó với dữ liệu truyền. Bộ nhận sẽ nhận dữ liệu và tính toán lại các bit kiểm tra để so sánh với các bit kiểm tra mà nó nhận được. Nếu hai kết quả khác nhau thì đây là một lỗi.

Một phương pháp kiểm tra đơn giản đó chính là sử dụng parity bit. Parity bit là phương pháp sử dụng 1 bit để kiểm tra số bit "1" hoặc "0" của chuỗi dữ liệu là "chẵn" hoặc "lẻ". Ví dụ sau đây minh họa việc kiểm tra parity chẵn, tổng số bit 1 trong chuỗi được truyền đi bao gồm cả dữ liệu và bit kiểm tra luôn là một số chẵn.

* Tạo bit parity: XOR tất cả các bit dữ liệu
* Kiểm tra parity: XOR tất cả các bit dữ liệu và bit parity. Nếu giá trị là 1 thì chuỗi dữ liệu nhận bị sai.

Một số điểm cần lưu ý:

* Bit sai có thể là bit dữ liệu hoặc bit parity
* Chỉ phát hiện được nếu số bit sai là số lẻ (sai 1, 3, 5, 7, ... bit)

|  |
| --- |
| [A diagram of a party  Description automatically generated with medium confidence](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjfw_xec0vBgOj4xXDCVNddSuBhxGX4lcOQ7WddUNpcdGHGgeX9rZV_YLOZHe8Gw2P9F3l_8HTeOILEuigq2hZyx76J-a0qt90TsYvFgDLq8XoWy5CsIl5XZTiPm2gVEQooaLHViA5som12/s1600/image.png) |
| Hình 1. Tạo và kiểm tra Parity chẵn cho chuỗi dữ liệu 8 bit |

Phương pháp kiểm tra parity đơn giản nhưng độ tin cậy kém nên được ứng dụng cho các giao thức có tốc độ truyền dữ liệu chậm hoặc số lượng bit dữ liệu cần kiểm tra ít ví dụ như giao thức UART.

CRC (Cyclic Redundancy Code) là một phương pháp phổ biến có độ tin cậy cao hơn nhiều so với sử dụng bi parity. CRC được ứng dụng trong nhiều giao thức có khối lượng dữ liệu truyền lớn hoặc tốc độ truyền dữ liệu cao như CAN, Ethernet, giao tiếp RF 15693, ...

**2. Lý thuyết về tính toán CRC**

Giá trị chuỗi bit kiểm tra hay chuỗi CRC là số dư của phép chia của chuỗi bit dữ liệu cho một chuỗi bit đa thức sinh (Generator Polynomial). Đa thức sinh là số chia sẽ khác nhau tùy vào mỗi giao thức quy định. Phép chia trong tính toán CRC sử dụng cách tính modulo-2. Modulo-2 thực chất là XOR hai số hạng.  
Giả sử đa thức chuỗi dữ liệu cần truyền là M(x):

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEg7jFMyz8nT3EzSD0I8V4xHmIE9UpZLW9PfOODVCJeGVilPklyx8jZK4aObcdnCnJdibjlMJwQnOkYiJdI89XSOgYnsbsyHaJYMHcInRdCvomBkqtbO5fcCqY8H2Mn5ZM2iMLD1LOYk8yEy/s1600/image.png)

Đa thức sinh là G(x):

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjipqaKBIeeCpObj_9ZO_s8P2r_yl1TX6iNVoZLXj2ogiU80KHAOzaQQ9swhc90awCoXbfTCbydrRHmYnXRYMOEOPOYlh5Ydab5zRTOrZdWZ-zG3jXcVtfqlM9C5yTYuEwAdccJaXKR6Ap3/s1600/image.png)

Trong đó:

* am và an bằng 1 hoặc 0
* Độ dài chuỗi CRC bằng độ dài đa thức sinh trừ 1 và bằng số mũ lớn nhất của đa thức sinh và bằng n.

Để tạo CRC, chuỗi dữ liệu cần truyền sẽ được mở rộng thêm n bit về phía bên phải:

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjfZUiMIND1Amk7MCuwOLmeMqEWoz-5WWXkMBQe1n3iFtA_PXLv9OsKaSCBVWR3FR0oTyTPEMf2lmDvS4CDajERCxv6kAGAW04y6iBFkzePGK4QB8H5bMgnLd4WZVSMHgJ8R8Kgar5DkPg4/s1600/image.png)

Điều này, tương ứng với việc dịch trái n bit chuỗi dữ liệu M(x).

Cuối cùng, chia T(x) cho G(x) và lấy số dư. Số dư chính là chuỗi CRC n bit.

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEje9vuhkk1rEa82Y4ZxjiZoCL9k7FKy7ewk0oCQyg79-iYKNwKCU4UiCMqdJeLDLhrIqh8Rr0vLljpVFnoItnr3vh5lSrpwC1pRy1dICjChBphUHB8ODNlPqFJjkjnZ2eQRsB5FgpjWs_AB/s1600/image.png)

Kiểm tra CRC được thực hiện bằng 1 trong 2 cách sau:

* Lấy chuỗi dữ liệu có cả các bit kiểm tra CRC chia cho đa thức sinh. Nếu số dư khác "0" thì dữ liệu nhận bị lỗi.
* Tách chuỗi dữ liệu và chuỗi CRC riêng. Chỉ lấy chuỗi dữ liệu chia cho đa thức sinh rồi lấy số dư phép chia so sánh với chuỗi CRC. Nếu hai chuỗi khác nhau thì dữ liệu nhận bị lỗi.

Ví dụ về tính toán CRC-4, tương ứng với số bit kiểm tra là 4 bit, với đa thức sinh như sau:

x^4 + x + 1 (b10011)

Chuỗi dữ liệu cần truyền có 8 bit như sau:

x^7 + x^5 + x (b1010\_0010)

Chuỗi dữ liệu trước khi chia sẽ được mở rộng thêm 4 bit "0":

x^11 + x^9 + x^5 (b1010\_0010\_0000)

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhDzrFHMRdjHR09kF-RCj8I8U1I5s37dKDD0dpRzvxlpoA1BL2mnI7LpePSt8-UuxBR1EMRXhheFVhsVW0ngyprDlLuvQ-bYZ7ZSn7MuLxa-rhGuEnUVAlFGK6Rukat8Ode93OvsMebwFPv/s1600/image.png) |
| Hình 2. Tính chuỗi CRC |

Việc kiểm tra CRC được thực hiện trên chuỗi dữ liệu có đính kèm CRC như sau:

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEi9kyyVTe-ulRekZ2TBDz8vdjwYok92azwgKEU1qp1cwKKSNRRl0m-XcpMacN3cLYx6Br2j4-IqIPYYMTaXrFgV5XI3tnuapHTvlZJ2TRvng7XlDGqBB94nUB3xjCbTeBnyEotRhOe5FZ4D/s1600/image.png) |
| Hình 3. Kiểm tra CRC - trường hợp nhận đúng, số dư bằng 0 |

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjywpuNJlfA842G7ABEt3vosDeuSV_-cdOIzoEahst50UsxrW-m5bAfqqQail2IY_kNHHf2FsX8vxVBlapBKufE51EvKevxWMxPPf0Wn4mjlFc3ZhGwurhoKj_oOilThX0sFXIxIRMXxLa_/s1600/image.png) |
| Hình 4. Kiểm tra CRC bằng cách chia chuỗi dữ liệu có CRC với đa thức sinh - trường hợp sai 1 bit và trường hợp sai 2 bit, số dư khác 0 |

Bộ nhận sẽ không phát hiện được lỗi dữ liệu khi chuỗi dữ liệu bị sai và chuỗi CRC cũng sai trùng với giá trị CRC của chuỗi dữ liệu bị sai. Tuy nhiên, xác suất để xảy ra đúng trường hợp này là thấp. Xác suất này càng thấp khi chuỗi CRC càng dài.

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEh4-eMx_6Lm_WQcyB4EGlH-UB72SF2scMARKeIx4A8Ii0LuZLWEL8t3RrcsSwP2EJdSQOXXiIcMem1UYx67GHcQ7g4CBNftD6uLEJbA7_Q1fXYlFDdlOOGkqqUkSZmWxaYNBC2d-7MJFfZV/s1600/image.png) |
| Hình 6. Kiểm tra CRC không phát hiện được lỗi khi cả dữ liệu và CRC cùng sai đồng thời |

Xét đa thức sinh g(x) = x + 1, đây là đa thức CRC-1, tính CRC cho chuỗi 8 bit b10100010 và chuỗi b10011111.

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEiSb30QSFq061-W8Wk2qh3XBQHYpTTJSgImd831Rz0r7FMKFh88Yt87oMiL31led-JZgF8K1Tn_TGJbOmHNLfKBsthIjsxibdTuwhrEuYVg0Fdh17zK-CTulubbg7gynfDLsJuP8kVgIoch/s1600/image.png) |
| Hình 7. Tính CRC-1 |

So sánh kết quả với phương pháp tính parity chẵn đã trình bày phía trên chúng ta có thể nhận thấy sự tương đồng. CRC-1 chính là phương pháp kiểm tra parity.

**3. Mạch nguyên lý tính CRC**

Xem lại các ví dụ đã trình bày trên đây, CRC được tính theo nguyên tắc:

* 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/AVvXsEjmDoa8sW_HDyxuzFRYBGq-VPv6ucWKNJyuAo4ewQvh-jSsKWWd1iaSyvTZnm0du6Bk9FX61ypSGQ8IVf-xvSQskWcsl3WFkUXIWd5uIh4dNAMIBWubYHr7vd3_lw_0jRyutdctpVA7cKry/s1600/image.png) |
| Hình 8. Vị trí có MSB = 1 được XOR với đa thức sinh |

Để thực hiện mạch CRC-1, ngoài cách XOR tất cả các bit dữ liệu đầu vào như đã trình bày ở phần trên, chúng ta có thể thực hiện dựa trên nguyên lý của việc chia đa thức như hình trên. Mạch cần 2 FF để lưu giá trị sau mỗi lần XOR và mạch sẽ dịch 1 bit sau mỗi lần XOR để lấy 1 bit dữ liệu mới như hình sau:

|  |
| --- |
| [A diagram of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEi-LQEDjesheFJHEj8hY_V2pr1eNQr9qkKE-sPQfFZSsdYM4lilxbxWjcJO1l_xpwol9lfR2ZHsUMh9XD6tF1SHHUtgiT9cs806g2_84fA31ejfEkbcZXcXc8qE_zKbNzkfjEY2a0UAxAyu/s1600/image.png) |
| Hình 9. Mạch nguyên lý của CRC-1 |

Ở hình trên, bit MSB sẽ điều khiển MUX chọn có XOR với đa thức sinh x+1 hay không? Tuy nhiên, sau mỗi chu kỳ tính, bit MSB luôn bị loại bỏ nên mạch MUX và XOR của bit MSB là không cần thiết. Mạch được rút gọn như hình sau:

|  |
| --- |
| [A diagram of a number  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgAYujA_jRH2GTRaV1JksKomaGQfU-uCBBC1rNtdNK3TWl-F6I9E2b4iYPNbdgSCF5x0_qUuaItWrD9-tZBdxkd8uoOC8MadSer6ILjH54BgcJ8mZJ57vwzBw2LIw21ZPOfbOAwGoY-j-bH/s1600/image.png) |
| Hình 10. Mạch nguyên lý CRC-1 (bỏ mạch tính ở bit MSB) |

Xét mạch MUX, nếu bit MSB bằng 1 thì bit 0 XOR với 1, nếu bit MSB bằng 0 thì tương ứng với việc bit 0 XOR với 0 nên mạch MUX được loại bỏ để thay bằng bit 1 XOR bit 0.

|  |
| --- |
| [A diagram of a number  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhPZV51YvU0uipXx4zMYCVlshuUwzzf93jeY7EHZAAoNMM_WczLPL5gjeA6ennNaVAQVB1EE873JdUn4STJqmFgq6iQM6rlXLuuxHG9UlGLMHeYARkqBRDa7E5I99xNa57Fc5EUSgxBv6F6/s1600/image.png) |
| Hình 11. Mạch nguyên lý CRC-1 (bỏ mạch MUX) |

Bit 0 chỉ dùng để lưu giá trị bit dịch vào nên cũng có thể loại bỏ.

|  |
| --- |
| [A close-up of a number  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEi7byNfzk9aKWd9pTCPGZeZYg-Pj2HpJ-6SZnDKpF5OrUnQM3o6LKXDVnVZ2NSMKCPEjbaABRyX7BtANgI8PAM5B9V-2vdrY4y_FHlFv6ngSJjlBZ2Nf2fjAR9PJYGgVbIqwSVsQTHyIRzw/s1600/image.png) |
| Hình 12. Mạch nguyên lý CRC-1 (bỏ FF đầu vào) |

Ở đây, bit CRC chỉ có 1 bit nên việc thêm 1 bit 0 ở chuỗi dữ liệu đầu vào để tính CRC cũng không cần thiết vì giá trị nào XOR với 0 cũng bằng chính nó.

|  |
| --- |
| [A close-up of a number  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhlIQFs5BsNZrGCinIShyphenhyphenUV3QLXQW1TRWje_k3QsOnE2JHJEK7tuQfV1a-SLD_H4i-T5CWKOJaXxmWp9PlsA5CqTwVfCrOhulPoXTbidFr_oI8IYZm6hWV4fpUT7ELmRGOzGOL14DqXG7Tx/s1600/image.png) |
| Hình 12. Mạch nguyên lý CRC-1 |

Biểu diễn thường thấy cho mạch tính CRC như sau:

|  |
| --- |
| [A diagram of a connection  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEg9yz_zUrqR4aWuIf4u8dlQYzkDA-L5k4s-nktS9IYqENronr6RFW6yZtt4Ys0ASnBd_La0C5PikyOjAdqUOo-Icn7jAK0Iu1hx5VPieRJhhNi4GEEp8zUHtINWDx0iCXsCCYjTLtPrOj2I/s1600/image.png) |
| Hình 13. Mạch nguyên lý CRC-1 với biểu diễn thông thường |

Tương tự, xét lại mạch CRC-4 có đa thức sinh x^4 + x + 1, mạch nguyên lý tính CRC-4 như sau (lưu ý, vị trí XOR với "0" thì loại bỏ cả MUX và cổng XOR):

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEihCnlvmreSqrSzjD3h5DnDBcPgCx3tMBS6ryOPvjF6dJL29nuxE9xlMobG5EYR475j4X-t5OT7y994qnd9jiDgoVme0ErJ_7DqL7NBnFyzATytjxMhgfEWLTfIJT7Oj8B83sw5426A0_yg/s1600/image.png) |
| Hình 14. Mạch nguyên lý tính CRC-4 |

**5. RTL code tính CRC nối tiếp**

**5.1 Nhận xét**

Qua hai ví dụ trên đây, nhận xét chung như sau:

* Tại vị trí mà bit đa thức sinh bằng "0" thì chỉ là phép dịch bit
* Tại vị trí mà bit đa thức sinh bằng "1" thì được chèn cổng XOR
* Dữ liệu nối tiếp để tính CRC dịch từ MSB đến LSB với số lần dịch bằng độ dài dữ liệu cộng độ dài giá trị CRC. Ví dụ, dữ liệu 8 bit dùng CRC-4 thì số lần dịch là 12 lần với 4 bit cuối là 4 bit 0 được thêm vào chuỗi dữ liệu.

**5.2 Phân tích module tạo và kiểm tra CRC**

Căn cứ vào những nhận xét trên, một thiết kế thực hiện tính CRC tổng quát được thực hiện như sau:

1. Sử dụng một define CRC\_CTRL\_POLY để cho phép tạo tín hiệu input điều khiển giá trị của đa thức sinh nếu muốn. Chú ý, độ rộng tín hiệu điều khiển bằng số bit CRC vằ bằng số mũ lớn nhất của đa thức sinh. Ví dụ, nếu đa thức sinh là x^4 + x + 1 thì độ rộng tín hiệu là 4 bit và giá trị gán cho tín hiệu điều khiển là 4'b0011 (bỏ bit 1 của x^4)
2. Sử dụng một define CRC\_CHECKER để cho phép tạo chức năng kiểm tra CRC
3. Sử dụng một parameter CRC\_GPW\_MAX cho phép cấu hình độ rộng đa thức sinh. Độ rộng đa thức sinh bằng số mũ lớn nhất của đa thức sinh. Ví dụ, nếu đa thức sinh là x^4 + x + 1 thì CRC\_GPW\_MAX = 4
4. Sử dụng một parameter CRC\_POLY\_VALUE cho phép gán giá trị đa thức sinh sẽ sử dụng nếu không sử dụng tín hiệu điều khiển được tạo ra bởi định nghĩa CRC\_CTRL\_POLY. Ví dụ, nếu không định nghĩa CRC\_CTRL\_POLY, đa thức sinh là x^4 + x + 1 thì CRC\_GPW\_MAX = 4 và giá trị CRC\_POLY\_VALUE = 4'b0011

Sơ đồ tín hiệu giao tiếp của module CRC như sau:

|  |
| --- |
| [A diagram of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEj4gsS0zpwkV9kr5KTsGjbFALUaSRidbJpUodmgIwYdBjSqyC5-g0mNdTlNmRlhvTG8JKfigeVFGEcjqyhS8qiJoQGGQQZccC1V26loHJ4pTu2M23KFDD3OI_kLSsQ3davtrlD2QayybcX3/s1600/image.png) |
| Hình 15. Sơ đồ tín hiệu của module tính CRC |

Hai tín hiệu ctrl\_en và chk\_en sẽ điều khiển chức năng tạo và kiểm tra CRC như sau, khi tín hiệu ctrl\_en tích cực, dữ liệu dùng để tạo CRC hoặc được kiểm tra CRC sẽ bắt đầu dịch vào data\_in. ctrl\_en sẽ tích cực bằng số bit cần dịch trên data\_in.

* Nếu chk\_en = 0 thì khi ctrl\_en = 0, crc\_seq sẽ giữ giá trị chuỗi CRC trong 1 chu kỳ xung clock
* Nếu chk\_en = 1 thì khi ctrl\_en = 0, crc\_error sẽ báo lỗi CRC
  + crc\_error = 1 thì chuỗi kiểm tra bị lỗi CRC
  + crc\_error = 0 thì chuỗi kiểm tra không bị lỗi

Mạch tổng quát của từng bit trong thanh ghi chứa giá trị CRC như sau:

|  |
| --- |
| [A diagram of a computer program  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhYgMsP5Dbny8EEa41ly_El_NRFYKBLyVVdhk6frC8Dgwm1JQWKqKsZDYFDYxO_aKGtvxt3Xna8kaIsFGjyG4fsGu8WixWHjF7ORB_nvV7cbick8dn9wal3rApwmIHIv0UkN1Cc6dVMmJ69/s1600/image.png) |
| Hình 16. Mạch tổng quát tính giá trị bit CRC thứ i |

Riêng bit 0 có đầu vào là data\_in có mạch như sau:

|  |
| --- |
| [A diagram of a computer program  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjIrPdyPf_K9tYaMucSHHZgd_72d23V8fVXJttsvGc3_DZPmmVOzkphMcJpAB9rDW7Vki9klRHBCyOBkYCB-bMP1_7yT_WDf0ArEPY07x0IPcQmUBXHIeVdfv_EIBp0aB-Ix9dJdAeic6hU/s1600/image.png) |
| Hình 17. Mạch tổng quát tính giá trị bit CRC 0 |

Khi có định nghĩa CRC\_CHECKER, mạch kiểm tra lỗi CRC sẽ được tạo ra như sau:

|  |
| --- |
| [A diagram of a block diagram  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEjjcEdKYEPcWXrzVDvTLiASJOWMCNkw7yLboY7NdNWhSf64kAsVlcEX4Uzcae3SF2ctU2vCFaxkjodk4_U0AIyEbQhiqHqjydtsRVpLJ-dWNQhB2e-aUOdmqOF7GIRy-qbdo02FthIzrhGW/s1600/image.png) |
| Hình 18. Mạch kiểm tra lỗi CRC |

**5.3 RTL code**

Link download RTL code và testbench: [CRC RTL code](http://www.mediafire.com/file/74plyk772ukvjad/crc_nguyenquanicd.zip)  
pass (nếu có): nguyenquanicd  
 **5.4 Kết quả mô phỏng**

|  |
| --- |
| [A screenshot of a computer  Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEiRSGubBt0OVUpjeHTCSogNC9h7DOUdEH6VYxUw_6jfPnGzS7iYZddBeE5eUPhIswfATbv3NTcgCFV4D3Qwcb0EKDOyurS1NBR9xvzr7NLKzsodtOzS9cu5h9-cvFFe3b3o5EU5m4CvYfd9/s1600/crc.PNG) |
| Hình 19. Mô phỏng tạo và kiểm tra CRC |

Đa thức sinh: x^4 + x + 1 tương ứng với việc gán ctrl\_poly\_en = 4'b0011

Dữ liệu dùng để tạo CRC: 1010\_0110 sau khi thêm 4 bit "0" là 1010\_0110\_0000 => Kết quả tính CRC là 1110

Dữ liệu dùng để kiểm tra CRC: 1010\_0110\_1110. Trong đó, 4 bit LSB 1110 là chuỗi CRC => Kết quả kiểm tra CRC là crc\_error = 0

*Lưu ý, đa thức sinh có nhiều cách biểu diễn khác nhau. Trong bài viết này chỉ dùng cách biểu diễn thông thường (normal representation) từ MSB đến LSB.*