Doing the math

M. Simon

For all you random number lovers there is some excellent documentation on the www about linear feed back shift registers. LFSRs are a way to produce quasi random numbers without too much effort. Why quasi random? Well one number is excluded (all ones or all zeroes) depending in whether you use the XNOR or XOR function for computing your random number. You can start from the beginning (all ones or all zeros) or use a seed number so your results are less decipherable. Now you might think this would be good for encoding (making a key) but that is not so. LFSRs are in some ways very predictable. But if you need a noise generator they are pretty good. And if you start with the same seed every time the noise is the same every time - which can be useful for debugging.

My favorite introduction is this Xilinx application note [1]. It covers the feedback terms for LFSRs up to 168 bits long. Note that the 167 bit register has only two feedback terms (and the input). A number of them described have that property. The 127 bit register is particularly convenient. It uses terms 126 and 127 and the input to compute the function. Lets see. Two to the 127th power is about 1.7E38 (don't forget the minus one). If you clocked that register at 1 GHZ it would take about 5.39E21 years before it repeated (if I did the calculation correctly). Anyway it is a very long time - much longer that the projected likely maximum age of the universe. All from 127 register bits and two bits of feedback.

If you want to get more into the math look up The Galois Field [2] and The Finite Field [3]. That is staring to get way above my pay grade. Fortunately smarter people than me have been thinking about this.

Did you know you can use an LFSR as a counter [4]? Me either. But a professor (or a grad student - I'm not sure) explains how to do it with Verilog tools. From 2010 so it is fairly recent. [5]

RSA Labs [5] gets into the crypto aspects. [6]

New Wave Instruments [6] has an explanation of the properties of the LFSR that is not too math heavy. I can understand it. And my math has deteriorated

considerably in the last 40 years. [7]

Texas Instruments has a 12 pager [7] on how to do signal analysis with an LFSR. They discuss fault coverage and other things useful to chip makers.

This all got triggered off by a friend of mine who was looking for a good hashing function to speed dictionary searches for a Forth compiler. An LFSR does a good job of that. But my idea was quicker and probably good enough. You just take the characters in the entry and add them two at a time (modulo arithmetic). i.e you add "AB" to "CD" and you get a tolerably good hash function with minimal computation. It differentiates "ACBD" from "ABCD" - and the short words are the most troublesome cases.

Well that got me and another buddy who is working on the project to discussing how computers do math [8]. Which got us discussing flooring [9]. Or to put it more mathematically "Division and Modulus for Computer Scientists". Because of the way computers do math they have more negative numbers than positive numbers. One more to be exact. This leads to some interesting results when the signs of the two numbers used in a division are different. The results are not symmetric. [10]

This page [10] shows how the different division algorithms can differ in different Forths. There is supposed to be a standard that eliminates this problem. But not every compiler writer follows the math standard.

And it is not just Forth. The Python people [11] are discussing the problem. Forth Inc [12] also gets into it and discusses why what is mandated was different from what was done. [13]

This Google group discusses [13] how to make a symmetric system perform floored division.

And finally this old Dr. Dobbs article [14] discusses why all this arithmetic can be a problem - it all has to do with remainders. It is probably the easiest explanation of the lot. So what have I decided to do about all this in the Forth I'm designing? I'll tell you when I decide. More study may be required.

M. Simon's e-mail can be found on the sidebar at Space-Time Productions [15].

Source URL (retrieved on *03/08/2014 - 1:13am*):
http://www.ecnmag.com/blogs/2012/09/doing-math

Links:
[1] http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf
[2] http://en.wikipedia.org/wiki/Galois_field
[3] http://mathworld.wolfram.com/FiniteField.html
[4] http://www.egr.uh.edu/courses/ece/ECE5440/ece5440_LFSR_Counters.pdf
[5] http://www.rsa.com/rsalabs/node.asp?id=2175

[6] http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm
[7] http://www.ti.com/lit/an/scta036a/scta036a.pdf
[8] http://www.diycalculator.com/downloads.shtml
[9] http://research.microsoft.com/pubs/151917/divmodnote-letter.pdf
[10] http://ccreweb.org/software/kforth/kforth5.html
[11] http://docs.python.org/release/2.2.3/whatsnew/node7.html
[12] http://www.forth.com/resources/evolution/evolve_5.html
[13] https://groups.google.com/forum/?fromgroups=#!topic/comp.lang.forth/QBtluXNh7J0
[14] http://wiki.strotmann.de/wiki/Wiki.jsp?page=Signed%20Integer%20Division&skin=raw
[15] http://spacetimepro.blogspot.com/