# Squirt: K Ring Compression Codec

By Simon Jackson, BEng.

## Fundamental Principals

The codec principal described takes a new approach to data compression. Forget the idea of source coding, as this does not work for all source data streams. Think more of it as source modulation onto a carrier sequence which has entropic redundancy in it. This redundancy can absorb some of the source stream, while allowing the carrier to always occupy a fixed number of bits. As a completely random carrier sequence can not store any information this is unsuitable for use in this example, and a totally ordered carrier sequence can not be easily be composed from a finite 'generator' using a finite number of bits of state machine information, as it will not hold modulation of zero to one, as there are no ones in the stream to sequence on from.

A middle ground has to be found. A modulation coding called Jaxon Modulation is capable of storing any source data onto a carrier with greater than two zeros for every one in the carrier stream. This sets a limit to aim for. This limit can be further reduced by using parallel carrier or so called bias element streams, where the usual rules of statistics of the binomial distribution increase the bias toward and over 2:1 as the number of parallel bias elements is increased. If any finite generator with ANY amount of bit bias is used, then given enough parallel bias elements used, data absorption of the source is possible.

## A Pseudo Code Example Of A Bias Element

| | |
|---|---|
| function forward<br>q=p(2); r=p(1);<br>switch(q)<br>/* asymmetric process */<br>case(00): if(n==0 and s==0) sample of k valid;<br>if(n==1 and s==1) k^=r;<br>/* odd or even exchange */<br>case(00): if(n!=s and r) begin n=!n; s=!s; end;<br>/* two to one motion bias */<br>case(01): if(k==s) m++ carry into n;<br>case(1x): if(k==!s) m++ carry into n;<br>end switch;<br>end forward; | function reverse<br>r=p(1);q=p(2);<br>switch(q)<br>/* asymmetric process */<br>case(00): if(n==0 and s==0) sample of k valid;<br>if(n==1 and s==1) k^=r;<br>/* odd or even exchange */<br>case(00): if(n!=s and r) begin n=!n; s=!s; end;<br>/* two to one motion bias */<br>case(01): if(k==s) m-- carry into n;<br>case(1x): if(k==!s) m-- carry into n;<br>end switch;<br>end reverse; |

Where p(n) gives up n bits of a reversible pseudo random number sequence, with x implying a don't care which state bit is in.

## Statistics At The So Called Injection Or Carry Points

In the randomization plane at the injection point, when the k bit is at the point of q motion into m++ carry into n, ¼ of the time no motion will happen and the k bit shall remain at the injection point with 50:50 chance of being zero or one [case 00]. This state has no effect on the bias. 1/8 of the time [n=0, s=1, case 01] motion of a one over the injection point will happen. ¼ of the time [n=0, s=1, case 1x] motion of a zero over the injection point will happen. The rest of the time nothing will happen. This gives the conditional probability ratio of zeros to ones given that something happens as 2 zeros for every one. Now the transit time before re-randomization would be twice as fast for zeros as for ones, so giving balance if it were not for the even or odd exchange.

At the second injection point, exiting the exchange zone, the speeds of zeros and ones are reversed. This has the effect of cancelling the bias between the bit states of 2 zeros for every one, but also makes zeros move slower by a factor of two through the sampling zone. This gives two sampled zeros for every sampled one bit. QED. With three bias elements used this gives the ratio of (all three bias elements are zero):(all three bias elements are one) = $2^3:1^3 = 8:1$, which exceeds the required 2:1 bias element probability for Jaxon Modulation to be successfully applied. For successful modulation also, a sub-sampling of k bits to allow at most one modulation per pass through the bias element sampling plane is required.

## Jaxon Modulation Of This 8:1 Bias

Jaxon Modulation is a reductive input to output coding. All input comes from the source, and all output is placed back into the source buffer. The sampled k bit can also be modulated from a zero to a one.

| Carrier State | Source In | Source Out Re-buffer | Modulate Carrier 0->1 |
|---|---|---|---|
| 0 | x0 | x | No |
| 0 | x1 | x0 | Yes |
| 1 | x | x1 | No |

The limit of two to one can be seen to result from each carrier one needing two zeros to reduce the 1 added to the source. This code is reversible.