4.4 Computing powers of x

or 64-bit generating polynomials was implemented by the authors in February-March 2007 and was used by a couple of Microsoft products. In 2009, the algorithm was generalized and these limitations were removed.

The fact that the CRC of a message followed by its CRC is a constant value

Proof:

$\mathrm{CRC}_u($

It is easy to see that

$$\mathrm{CRC}_u$$

[Has01]:

x)

$$\mathrm{CRC}_u\ \big(Q(x)\ x^{W\ \ D};CRC\ M(x);v(x)\big) =$$

$$= \mathrm{CRC}_u\ \big(Q(x)\ x^{W\ \ D};Q(x)\big) =$$

$$= \big(Q(x)\ u(x)\ x^{W} + Q(x)\ x^{W\ \ D}\ x^{D} + u(x)\big) \bmod P(x) \equiv$$

When D W,

CRC

```
1 Crc CrcByte(Byte value) f
```

Consequently,

MulByXpowD $v(x) = (x)$

```
1 // Processes N stripes of StripeWidth words each
2 // word by word, in an interleaved manner.
3 Crc CrcWordByWordBlocks(Word  data, Crc v, Crc u) f
4   assert(n % (N   StripeWidth) == 0);
```

The drawbacks of this approach are obvious: it does not work well with small inputs { the cost of CRC concatentation becomes a bottleneck, { and it may be susceptible to false cache collisions caused by cache line aliasing.

If the cost of CRC concatenation was not a problem, cache pressure could be mitigated with the use of very narrow stripes. The code in question, lines 33-37 of listing 5 which combine CRCs of individual stripes, iteratively computes

$$\mathrm{crc}_0(x) = \mathrm{crc}_k(x) + \quad \mathrm{crc}_0 \quad x^{8S} \bmod P(x)$$

for

mentally:

$v_{k+1;n}(x$

v

representation of $v(x)$, viewed as a W-normalized representation, is equal to $v(x)\ x^{W\ D}$.

Using the same technique as in formula (9), for $D\ W$ let

v

```
1  Crc CrcInterleavedWordByWord(
2      Word  data, int blocks, Crc v, Crc u) f
3    Crc crc[N+1] = f0g;
4    crc[0] = v ^ u;
5    for (int i = 0; i <
```

functions was better but still not as good as hand-written assember versions, which were provided for all compilers.

The fastest code for 128-bit CRC on X86 platform was generated by GCC 4.5.0.

5.3 Choice of interleave level

References

[Bla93] Richard Black. Fast CRC32 in software. `http://www.cl.cam.ac.uk/research/srg/bluebook/21/crc/crc.html`, 1993.

Availablehttp://mM979 w(anaarchPublications/)]TJ -94.9414 -11.955 Td [bBrau01f(as-.)-9(techrepors.pdf

GG01

ers-320(323405(,)-

int

Table 8: CRC-128 performance: Slicing CRC