Multiplierless hack for constant division by certain factors

I would like to present you with a formula for quotient calculation that works for any constant divisor of the form 2^n-1 (^ denotes exponentiation). For n=2,3,4,5,6,7, these divisors are 3,7,15,31,63,127. I have tested all divisors with n up to 16 and the approach works. My formula works only for dividends x that are in the range [0,(2^(2*n))-2].

The formula is as follows and it is both divisionless and multiplierless (as initially derived):

```
y = (((x>>n)+x+(1<<n)+1)>>n)-1;
```

which can also be written as:

```
y = (x + (x>>n) + 1) >> n;
```

This is a formula for software integer division, i.e. truncating division that calculates the quotient of the division. Please note that 1«n is the strength-reduced form for calculating 2^n. It does not require any double-word arithmetic (like with multiplying by the multiplicative inverse and then performing adjustment steps).

E.g. for n = 6, the divisor is 2^n – 1 = 63. The formula will work for x <= 4094 = (2^(2*n)) – 2 = (1«n)*(1«n)-2. It will produce one-off errors for certain x >= (2^(2*n))-2. But it will be correct for the specified "small" x values.

A testcase for the algorithm is here:

```
#include <stdio.h>

int main(void)
{
  int qapprox, qexact;
  int i, j, k;

  for (i = 2; i < 8; i++) {
    for (j = 0; j < (1<<i)*(1<<i)-1; j++) {
      k = (1<<i) - 1;
      qapprox = (((j>>i)+j+(1<<i)+1)>>i)-1;
      qexact  = j / k;
      if (qapprox != qexact) {
        fprintf(stderr,
          "qapprox = (%d/%d) = %d\tqexact = (%d/%d) = %d\n",
          j, k, qapprox, j, k, qexact);
```

```
        }
      }
    }
    return 0;
}
```

I had devised the hack via "algorithmic sketching". I mean that I had started from a basic formula with some “holes” in it. The holes where small constants or operators. There exist such techniques in academia, put more nicely; sometimes called algorithmic sketching, combinatorial sketching, and it is relevant to forms of program synthesis. Essentially it is about devising algorithmic sketches (incomplete algorithms with wildcards) and then using a mechanism to fill the gaps.

Seems I can claim the specific hack as back as May 16, 2011.

Special thanks to everyone who has participated in the following threads. The Taylor series connection is very interesting, as mentioned by Tim Wescott and Tauno Voipio.

- http://embdev.net/topic/346629
- http://groups.google.com/forum/#!topic/comp.arch.embedded/6Mb4efc1ofw
- http://www.reddit.com/r/programming/comments/2j18nt/multiplierless_hack_for_constant_division_by_2n1/