Hardwired Paeth Codec for Portable Network Graphics (PNG)

E.A. Hakkennes and S. Vassiliadis
Electrical Engineering Dept.
Delft University of Technology
Mekelweg 4, 2628 CD Delft
The Netherlands

E-mail: E.Hakkennes@ET.TUDelft.NL

Abstract

This paper describes an execution unit capable of computing the Paeth Predictor, as used in the Portable Network Graphics (PNG) standard. PNG is a rather new, lossless compression method for real-world pictures. It features five prediction schemes, of which the modified Paeth predictor is the most computational intensive. This paper focuses on a hardware implementation of the Paeth predictor and a hardware Paeth codec capable of computing three different quantities: the Paeth predictor of three inputs, the difference of the current pixel and the Paeth predictor of the other inputs (Coding), and the sum of the coded input and the Paeth predictor of the other three inputs (Decoding). The proposed Paeth-codec takes two cycles, where a cycle is comparable to an general purpose ALU cycle. Depending on the mode of operation, the proposed mechanism produces the predictor or the (de/en)-coded pixel value.

1. Introduction

A standard that is gaining popularity in image coding and compression is the Portable Network Graphic (PNG) [1] standard. The PNG standard has been created as an alternative solution to Graphics Interchange Format (GIF) [2]. PNG features a lossless compression scheme, based on predictive coding and deflate compression. The standard is written so that its speed would be high and that it would benefit from the Multimedia extensions of general purpose processors [7]. One of the key-features of PNG is the ability to chose out of five predictors for the predictive coding, namely: none, up, left, average and Paeth. All five predictors can operate with the value of only three adjacent pixels. These pixels are positioned above, left and left-above the current pixel. After the prediction step, the coded pixel data is stored in a bit-stream which is subsequently deflated using the gzip algorithm [3].

The Paeth predictor is normally computed in software. The routine used for this is defined in the PNG standard and shown here as Figure 1.

```
int predict ( int a, b, c )
{
int p, pa, pb, pc
p = a + b − c                          /* this is the initial estimate */
pa = abs( p − a )          /* distance of each member to the */
pb = abs( p − b )                      /* initial estimate */
pc = abs( p − c )
if ( pa ≤ pb ) and ( pa ≤ pc ) return ( a )          /* return */
else if ( pb ≤ pc ) return ( b )       /* element nearest to p, */
return ( c )                           /* in a,b,c tie-break order */
}
```

Figure 1. The Paeth encoding routine according to the PNG specification [1].

To compute the Paeth predictor on a SunSparc 10 processor, 21 instructions are needed, including 6 branches. This code is shown in Figure 2 for reference. If the code would be scheduled for a Very Large Instruction Word (VLIW) [5] machine, the computation of pas, pbs and pcs could be parallelized, but the number of cycles would still be around 15 (assuming pipelined operation).

To improve the execution speed, we propose a hardwired Paeth prediction unit, which computes the Paeth predictor of a set of three input values in two machine cycles[1]. As the predictor is selected from the input values, and the critical path is the control of the output selectors, we can also precompute the difference or the sum of a fourth input (d) with each of the three inputs. This means that we are also able

[1] A machine cycle assumed here is comparable to the cycle time of a general purposed ALU.

```
_predict:
        add a,b,temp
        sub temp,c,p
        subcc p,a,pas
        bneg,a L2
        sub 0,pas,pa
L2:
        subcc p,b,pbs
        bneg,a L3
        sub 0,pbs,pb
L3:
        subcc p,c,pcs
        bneg,a L4
        sub 0,pcs,pc
L4:
        cmp pa,pb
        bg L9
        cmp pb,pc
        cmp pa,pc
        ble L8
        cmp pb,pc
L9:
        ble L8
        mov b,a
        mov c,a
L8:
        retl
        nop
```

Figure 2. Sparc-10 pseudo assembler code. All register names have been named to the original variable names for readability. Note that the caller expects to see the result in the register where it stored a.

to compute the coded or decoded value within the same two machine cycles. We compute the predictor by directly computing the distances of the initial estimator (p) to each input, and selecting the input which has the smallest distance.

The proposed scheme operates as follows:

- Direct computation of the distance of each input to the initial estimate.
- Compare these distances using Carry-generators.
- Select the input with the lowest distance.

For the codec unit, we precalculate three temporal results, which are the sum (decoding) or difference (encoding) of the current pixel and each of the inputs, and select one of these precalculated values. Using this scheme, the critical path is not affected, and yet the number of executed operations is increased.

The paper is organized as follows: first we provide some background information about the PNG standard and the Paeth Predictor, in Section 3 we describe the software routines used in the Paeth Predictor and the modifications needed to allow a hardware implementation. In Section 4 we describe an extension of the predictor, the Paeth Codec, which can code or decode a pixel with no additional cycle-time. In Section 5 we evaluate the unit with some hardware and time estimations. Section 6 concludes the discussion with some final remarks.

2. Background

In this section we will provide some information about compression of images, the type of compression schemes and in particular the compression method used by PNG.

We begin by indicating that pictures may contain information in a structured way, and this structure introduces redundancy. Redundancy means that all information may not be necessary to convey the message.

In order to diminish the size of the picture on the storage device (e.g. disk) or the transmission time over the Internet, we need a method to extract and describe the redundancy in pictures. There are two basic ways of compression, lossless and lossy. Lossy compression could be appropriate for photographic pictures. The decompression of a lossy compressed image results in a similar but not necessary a 100% identical picture. The legitimacy for such a scheme relies on the fact that "small" differences are not visible or distinguishable to the human eye, thus losing some information can be acceptable.

Lossless compression is mostly used for synthetic pictures, or computer generated graphics. When decompressing a lossless compressed image, the original image is restored, which is a 100% identical copy of the original.

PNG has been defined as an alternative solution to GIF and it has been developed with the following objectives [1]:

- Simple and portable: developers should be able to implement PNG easily.
- Legally unencumbered: to the best of knowledge of the PNG authors, no algorithms under legal challenge are used. (Some considerable effort has been spent to verify this.)
- Well compressed: both indexed color and true-color images are compressed as effectively as in many other widely used lossless formats, and in most cases more effectively.
- Interchangeable: any standard-conforming PNG decoder must read all conforming PNG files.

- Flexible: the format allows for future extensions and private add-ons, without compromising interchangeability of basic PNG.
- Robust: the design must support full file integrity checking as well as simple, quick detection of common transmission errors.

These objectives have resulted in quick adoption by both industry and the Open Source Software movement. PNG has become the native format for graphics in Microsoft Office 97 [8] and is also used more and more on the Internet.

In addition, PNG is now in the early stages of international standardization, thanks largely to its inclusion in the VRML97 standard [9]. It is expected to become a joint ISO/IEC standard (ISO/IEC 15948) [9] by early 2000.

The "core business" of PNG is the compression of graphical data. This is achieved using prediction coding (filtering) and standard deflate/inflate compression [3]. In order to improve the compressibility of the data, filtering is used. The purpose of filtering is the extraction of spatial redundancy by recording only the differences between the current pixel and one or more of its neighbors. PNG offers five filter types, namely: none, up, left, average and Paeth. In this paper, we develop a codec scheme for the Paeth predictor. The other four predictors are trivial to implement in hardware.

-	-	-	-
-	c	b	-
-	a	d	·
·	·	·	·

Figure 3. The definition of a, b, c and d according to the PNG specification [1].

Figure 3 gives the naming conventions of the Paeth predictor within the PNG standard. The pixels denoted with "-" are already transmitted and no longer of interest, the pixel denoted with "d" is the current pixel and the pixels denoted with "." will be transmitted in the future. Note that the naming, and as a result of the naming the tie-break-order, are not identical to the original Paeth predictor as defined in 1991 by Alan W. Paeth [6].

The Paeth predictor is used to achieve compression, as it is anticipated that the difference between the predictor and the actual pixel will be small. This is caused by the spatial redundancy in the picture. Pixels generally do not differ much from their neighbors. The difference between the predicted pixel value and the actual pixel value is transmitted, after compression using deflate techniques. The resulting differences are generally small numbers, which need fewer bits for transmission. In this way compression is achieved.

As indicated earlier the five filters of PNG are None, Up, Left, Average, and Paeth. The first is very simple, each pixel is predicted as zero. The Up and Left predictor depend on one of the neighbor pixels. The Average filter depends on both these neighbors. The Paeth predictor is the most complicated predictor. It depends on three neighboring pixels. The naming of the neighboring pixels is shown in Figure 3. Summarizing the five predictors and their function, the following holds true:

None The None filter transmits (d)

Up The Up filter transmits $(d - b)$

Left The Left filter transmits $(d - a)$

Average The Average filter transmits $(d - (a + b)/2)$

Paeth The Paeth filter transmits $(d - Paeth(a, b, c))$

The Paeth Predictor makes an initial prediction of d with the following formula: $Pd = a + b - c$, where a, b and c are defined according to Figure 3. If the intensity of the picture is gradually increasing of decreasing in that area, this prediction is perfect. However, this predictor doesn't fulfill all four criteria which Paeth defined in his article. These criteria are summarized in Table 1.

Identity	$P(a, a, a)$	$=$	a	(1)
Transposition	$P(a, b, c)$	$=$	$P(a, c, b)$	(2)
Complementation	$P(a, b, c)'$	$=$	$P(a', b', c')$	(3)
Membership	$P(a, b, c)$	$\in$	$\{a, b, c\}$	(4)

Table 1. The criteria for the Paeth Predictor.

The first criterion is trivial, if all pixels in some neighborhood have the same value, it is safe to predict this value.

The second criterion ensures that if rows and columns are interchanged, the result doesn't change. This seems contradictory to the tie-break order. However, in the original Paeth predictor one can't construct a case where this is of importance. In the PNG Paeth predictor, the second criterion doesn't hold anymore. However, as column and row interchanging is not used in PNG, it is not important.

The third criterion yields that the inversion of the raster (like a negative of a photo) results in an inversion of the predictor.

The fourth criterion ensures that the predictor is always within the bounds of the pixel-value range for each input combination. (No need for clipping/saturation.) This fourth criterion is implemented by selecting the element closest to (a+b-c) as the predictor. This may yield a less optimal predictor, but there are no worries about bounds. It means that

all computations and comparisons within the Paeth predictor should be done using enough precision in order to produce a correct result.

Figure 1 gives the software routine to compute the Paeth Predictor. The PNG standard [1] deviates slightly from the original Paeth Predictor in that it operates as follows:

1. It defines a new naming scheme for the incoming pixels, which essentially results in a different tie-breaking order.

2. It defines that all operations are done on 8-bit (byte) quantities. This simplifies the computations and encourages the use of MMX [7] instructions.

3. It defines the pixel left to the current pixel to be the last transmitted pixel. In case of interlacing, there can be quite a number of pixels "in between". The same holds for the vertical direction. The previous line is interpreted as the previous transmitted line. This technique reduces the amount of bufferspace needed. The different interlacing schemes ensure that the a, b, c, and d pixels are still the corners of a rectangle.

All these modifications are made in order to make a software implementation "straight forward" and possibly fast, maybe at the expense of a slightly less optimal compression. They also make the implementation of a hardware accelerator for the Paeth predictor or even a Paeth codec feasible.

In this paper, we will assume the definition of the Paeth predictor as given in the PNG definition [1]. Under this definition, all operations are performed on 8-bit quantities (bytes). These bytes are interpreted as unsigned binary numbers. If the image is composed of 16-bit deep RGB values, (48 bits per pixel), the operations are performed 6 times on 6 bytes independently of each other. If the data is only black-and-white, 1 bit per pixel, this 1 bit is packed as a byte and consequently treated as a byte.

Due to the possible interlacing schemes defined in the PNG standard, the previous received pixel is not necessarily adjacent to the current pixel. This holds also for the vertical direction. The Paeth predictor selects the previous pixel as the last transmitted pixel, and the previous line as the last transmitted line. This might result in a slightly less optimal compression, but decreases the memory-requirements of the (de)coding process.

3. Computing the Paeth Predictor

The routine displayed as Figure 1 is the Paeth predictor as defined in the PNG standard [1]. The inputs a, b and c are 8-bit, treated as unsigned binary numbers. However, the variables internal to the routine are not to be truncated to 8 bits.

In order to propose a hardware implementation of this routine, we rewrite it. In Figure 4 we can distinguish three steps. These steps will be exposed in the hardware implementation. In the first step, we compute pas, pbs and pcs. These are the signed variants of pa, pb and pc, denoted as 10 bit, two's complement intermediate numbers. In the second step, we compute Test_1,Test_2 and Test_3. The third step is the selection of the right input as the output.

---

int predict (int a, b, c)
{
int pas, pbs, pcs
bool $Test\_1$ $Test\_2$ $Test\_3$

pas = ($b - c$)
pbs = ($a - c$)
pcs = ($a + b - 2c$)

$Test\_1$ = ($|pas| \leq |pbs|$)
$Test\_2$ = ($|pas| \leq |pcs|$)
$Test\_3$ = ($|pbs| \leq |pcs|$)

If $Test\_1$ and $Test\_2$ return (a)
else if $Test\_3$ return (b)
return (c)
}

Figure 4. The Paeth Algorithm simplified, so that it can be mapped to hardware. Note that the result of this routine and the original (Figure 1) is the same.

The computation of pas, pbs and pcs is done by an adder circuit [12, 10]. As the range of the unsigned bytes a, b and c is from 0 to $(2^8 - 1)$, the variable pcs can range from $0 + 0 - 2 * (2^8 - 1)$ to $2^8 - 1 + 2^8 - 1 - 2 * 0$, which is from $-2^9 + 2$ to $2^9 - 2$. This range is just covered by a 10-bit two's complement number, which ranges from -2^9 to $2^9 - 1$. Although pas and pbs are representable as 9-bit two's complement numbers, we also represent them as 10-bit two's complement numbers to facilitate the subsequent comparisons and to preserve the regularity of the unit. In binary notation, this leads to the following additions:

$$pas = \quad (b - c) \quad = (00b + 11\overline{c} + 1) \tag{1}$$

$$pbs = \quad (a - c) \quad = (00a + 11\overline{c} + 1) \tag{2}$$

$$pcs = \quad (a + b - 2c) \quad = (00a + 00b + 1\overline{c}1 + 1) \tag{3}$$

As can be concluded from the previous three formulas, a sign-extension takes place to make all the input numbers 10 bits long. The negative value of $-c$ is computed using an inversion and the addition of a hot-one.

The computation of pcs involves a 3 to 1 addition, where one of the operands is shifted one position to the left (multiplied by 2). This is accommodated by using an extra level of Full-Adders, which performs a carry-save addition of the three operands, resulting in a sum and a carry word. These are then added in a 2-1 binary adder. A graphical representation is shown in Figure 5.

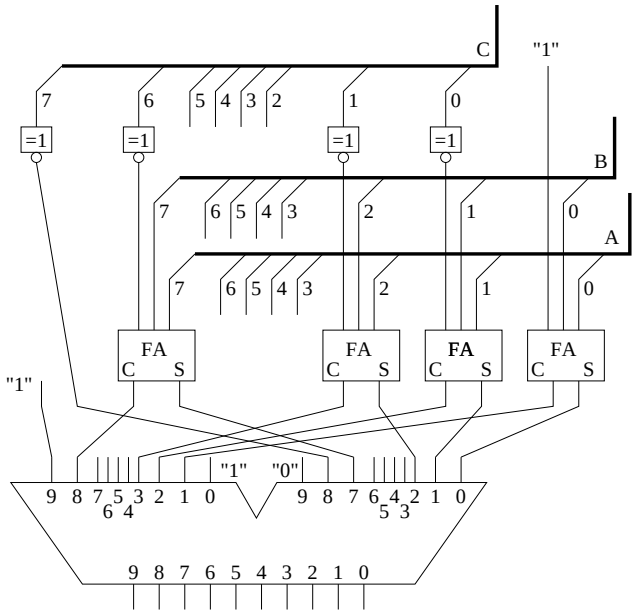

Figure 5. The adder used to compute pcs from a, b and c.

After the computation of pas, pbs and pcs, there are several ways to compute $Test\_1$, $Test\_2$ and $Test\_3$. To compute $Test\_1$ we have to find out whether $|pas| \leq |pbs|$. We can use a carry-based comparison of pas and pbs. This means that we add them in some form and that the resulting carry reflects whether the inequality was true or false.

We first have to adjust the signs of pas and pbs. In order to compare them, we check whether $|pbs| - |pas| \geq 0$. In order to facilitate this, we have to make sure that pbs has a positive sign and pas has a negative sign. If the sign is opposite, we invert the operand and add a hot-one to the result. This hot one is taken care off in the addition. If both pas and pbs are inverted, there are two hot ones. This means the carry-generator needs a special structure to accommodate this. This is implemented as a layer of half-adders, as shown in Figure 6.

The test $|pbs| - |pas| \geq 0$ is now modified to $pb_{pos} + pa_{neg} \geq 0$. The test for carry_out is basically the test for result $\geq 2^{10}$. We have to keep in mind that the sign-bit of pas is interpreted as a positive number here, with value 2^9 in stead of -2^9. We are therefore essentially adding 2^{10}. The binary summation is therefore: $pa_{neg} + 2^{10} + pb_{pos} \geq 2^{10}$ So if $pb_{pos} + pa_{neg} \geq 0$ the binary addition $pb_{pos} + pa_{neg}$ generates a carry_out. Figure 6 gives a graphical representation of the unit which computes Test_1 from pas and pbs.

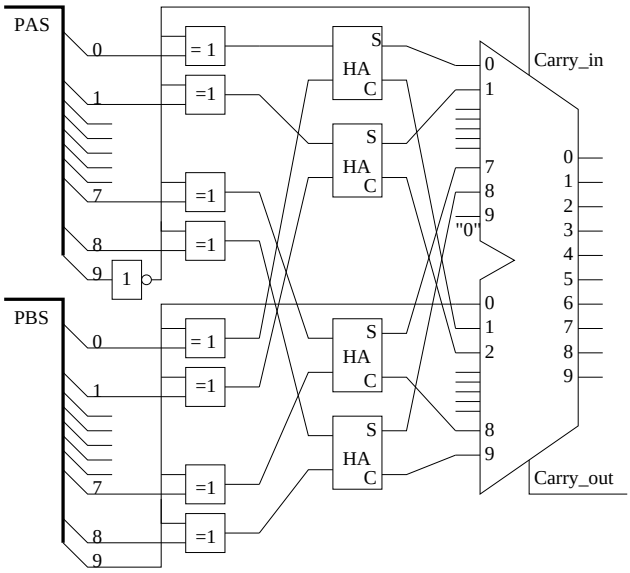

Figure 6. The computation of Test_1. Bit number 9 is the Most Significant bit, the Sign-bit. The outputs 0 to 9 are not used, and need not be computed.

Test_2 and Test_3 are computed using similar logic. The control of two mux-boxes is trivial. Figure 7 gives a graphical representation of the entire unit.

4. Implementation of a Paeth codec

A possible extension of this unit is the extension to a Paeth codec, which has not only the a, b and c input, but also uses the to be encoded or decoded pixel, d. When the prediction is known, the coding is simply subtracting the prediction from the actual data.

$$\begin{aligned} Coded\_Data &= Actual\_Data - Prediction \\ &= d - pred \end{aligned}$$

Decoding is done by adding the prediction and the received coded data.

$$\begin{aligned} Original\_Data &= Coded\_Data + Prediction \\ &= d + pred \end{aligned}$$

These operations are done modulo 256, as defined in the standard [1].

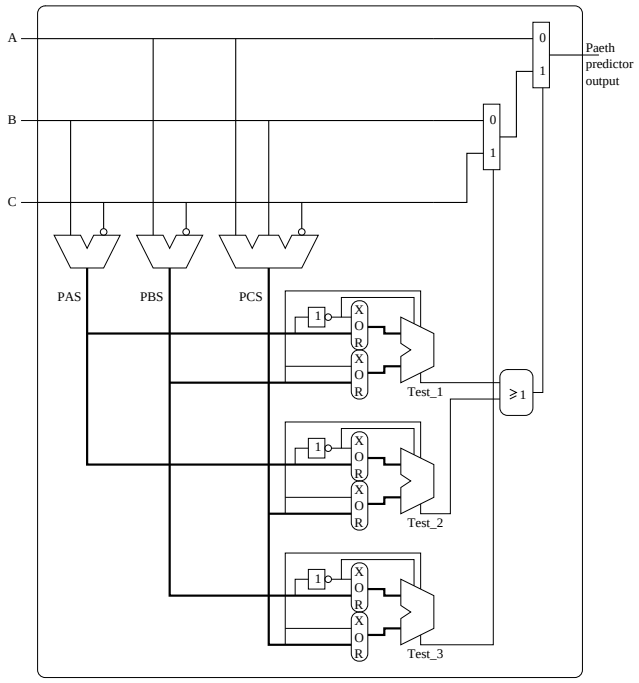

Figure 7. Proposed implementation of the PNG Paeth Predictor.

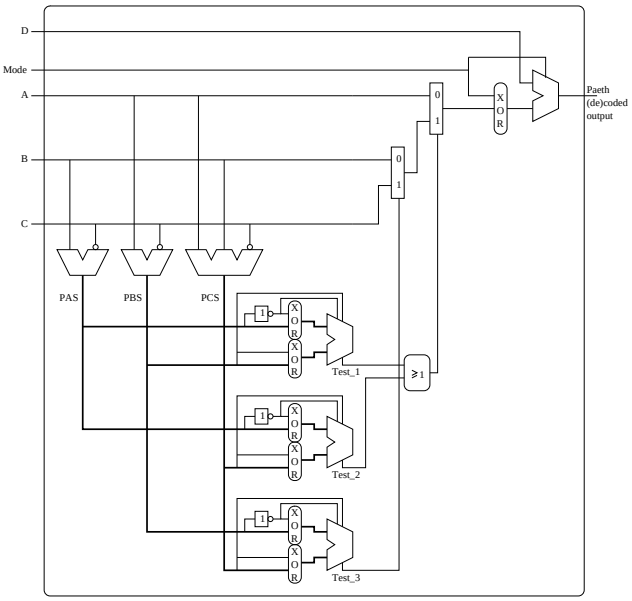

Figure 8. Direct implementation of the PNG Paeth Codec.

The most obvious accommodation of this addition/subtraction step is after the multiplexers, but this causes an increase of the latency of the unit. This is shown in Figure 8.

As the predictor is always equal to one of the input values, and the path from the input to the data-input of the multiplexers is empty, (as opposed to the path from the input to the control-input of the multiplexers) the adding/subtraction step can be accommodated there. Thereby we effectively make a 2-cycle, 4-input Paeth codec, with three operation modes:

Code In this mode, the d input is set to the value of the to be coded pixel. The result is the coded pixel. (Mode=1)

Decode In this mode, the d input is set to the received coded input and the output is the reconstructed pixel. (Mode=0)

Predict In this mode, the d input is set to zero and the operation is set to decoding. This results in the output reflecting the normal Paeth predictor. (Mode=0)

The program-notation for this is given in Figure 9. A graphical representation of the implementation of this optimized execution unit is shown in Figure 10.

5. Hardware and Time estimations

We have presented a sample implementation of a 4-input Paeth codec, capable of coding and decoding images using the Paeth Predictor as described in the PNG standard. It computes which of the inputs to use as Paeth predictor and in parallel (de)codes the fourth input with all inputs of the predictor. After this step, the right result is chosen.

The critical path of the codec unit is the control of the output multiplexers. It is basically two levels of binary adders long. The path to the data-input of the multiplexers is only one level of adders long. We estimate that the speed of this unit equals that of a standard two-cycle multiply unit or requires two ALU cycles. This estimation is based on the fact that the critical path is basically two times an adder delay, which is bounded by an ALU cycle time.

The hardware requirements for the proposed unit are relative modest for a 4-input unit. We need:

- 2 2-input 10-bit adders
- 1 3-input 10-bit adder
- 3 3-input 10-bit carry-generators
- 2 2-input 8-bit selectors
- 3 2-input 8-bit adders for the codec unit.

The scheme we have presented here is incorporated in the execution unit for multimedia of the embedded system

int codec (int a, b, c, d, $mode$)
{
int pas, pbs, pcs
unsigned int Res_a, Res_b, Res_c
bool $Test\_1$ $Test\_2$ $Test\_3$

pas = ($b - c$)
pbs = ($a - c$)
pcs = ($a + b - 2c$)
Res_a = $(d + (1 - 2 * mode) * a)$
Res_b = $(d + (1 - 2 * mode) * b)$
Res_c = $(d + (1 - 2 * mode) * c)$

$Test\_1$ = ($|pas| \leq |pbs|$)
$Test\_2$ = ($|pas| \leq |pcs|$)
$Test\_3$ = ($|pbs| \leq |pcs|$)

If $Test\_1$ and $Test\_2$ return (Res_a)
else if $Test\_3$ return (Res_b)
return (Res_c)
}

Figure 9. The Paeth Codec Algorithm, with the subtraction before the selection. This results in a codec which is as fast as a predictor in hardware.

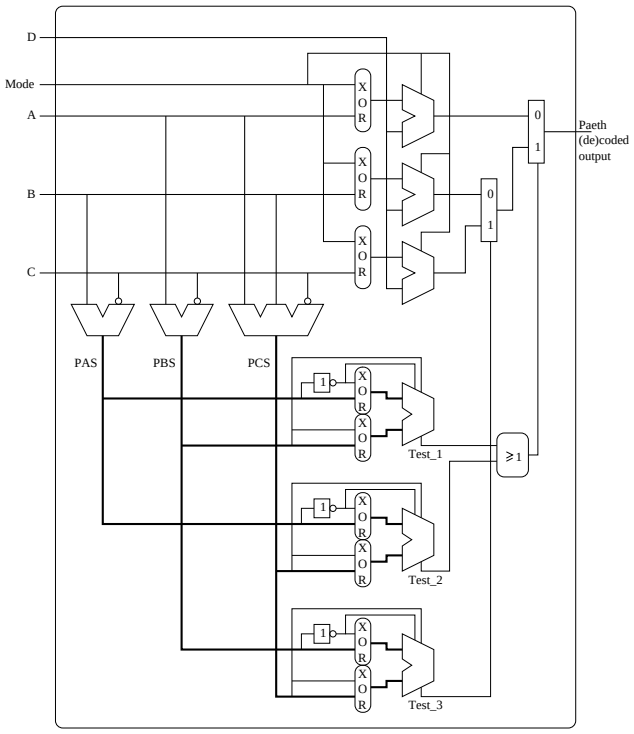

Figure 10. Optimized implementation of the PNG Paeth Codec.

project Molen (Dutch for windmill). The project aims at developing a special purpose multimedia coprocessor, in addition to other embedded system architectures. A number of special function units are currently being developed, among which the Add-Multiply-Add unit [4] and the Sum Absolute Difference accelerator [11].

6. Conclusions

We presented a Paeth codec, which computes either the Paeth predictor, the Paeth-coded or the Paeth-decoded pixel with only a two cycle delay. Compared to a 21 machine instruction SPARC implementation, this is a ten-fold speedup. The unit is developed for PNG coding, but can also be useful in other graphic schemes. The hardware requirements for the unit are modest, in the order of 9 10-bit adders.

We compute the Paeth Predictor using the following steps:

- Direct computation of the distance of each input to the initial estimate.
- Compare these distances using Carry-generators.
- Select the input with the lowest distance.

The Codec variant computes the difference or sum of the value to be encoded/decoded in parallel to the first step, it does not alter the last two steps. The final step remains the selection of the right output.

Acknowledgements

The authors would like to thank Sorin Cotofana and Lyonne Zonneveld for their careful proofreading and helpful suggestions.

References

[1] T. Boutell and T. Lane. PNG (Portable Network Graphics) Specification, version 1.0. ftp://ftp.uu.net/graphics/png/documents/png-1.0-w3c.ps.gz.

[2] C. W. Brown and B. J. Shepherd. *Graphic File Formats; reference and guide*. Manning Publications Co., 1995.

[3] P. Deutch, J.-L. Gailly, and M. Adler. *GZip*. http://www.gzip.org.

[4] E. Hakkennes, S. Vassiliadis, and S. Cotofana. Fast computation of compound expressions in two's complement notation. In A. Sydow, editor, *15th Imacs World Congress on Scientific Computation, Modelling and Applied Mathematics*, volume 4, Artificial Intelligence and Computer Science,

pages 689–694, Berlin, August 1997. Wissenschaft & Technik Verlag.

[5] J. L. Hennesy and D. A. Patterson. *Computer Architecture, a Quantitative Approach*. Morgan Kaufmann Publishers Inc., 1990.

[6] A. W. Paeth. Image file compression made easy. In J. Arvo, editor, *Graphic GEMS II*. Academic Press Inc., 1991.

[7] A. Peleg and U. Weiser. MMX technology extension to the intel architecture. *IEEE Micro*, 16(4):42–50, August 1996.

[8] G. Roelofs. http://www.cdrom.com/pub/png/pnghist.html. Here is stated that Office 97 is using PNG as the native image format.

[9] G. Roelofs. News and history of the png development group. http://www.cdrom.com/pub/png/pngnews.html.

[10] N. R. Scott. *Computer number systems and arithmetic*. Prentice Hall, Englewood Cliffs, New Jersey, 1985.

[11] S. Vassiliadis, E. Hakkennes, J. Wong, and G. Pechanek. The sum-absolute-difference motion estimation accelerator. In *Euromicro '98*, volume II, pages 559–566. IEEE Computer Society, IEEE, August 1998.

[12] S. Waser and M. J. Flynn. *Introduction to arithmetic for digital systems*. CBS College Publishing, 1982.