References

1 L'ECUYER, P.: 'Uniform random number generation', *Ann. Oper. Res.*, 1994, **53**, pp. 77–120
2 KNUTH, D.E.: 'The art of computer programming: Seminumerical algorithms (Addison-Wesley, 1981), 2nd Edn., Vol. 2
3 RIDELLA, S., ROVETTA, S., and ZUNINO, R.: 'Plastic neural gas for adaptive vector quantisation', submitted to *IEEE Trans. Neural Netw.*
4 PARODI, G.C., RIDELLA, S., and ZUNINO, R.: 'Using chaos to generate keys for associative noise-like coding memories', *Neural Net.*, 1993, **6**, (4), pp. 559–572

Digit serial division algorithm

A.E. Bashagha and M.K. Ibrahim

Indexing term: Digital arithmetic

A new radix digit serial division algorithm is presented. The speed of the proposed algorithm is nearly twice that of an existing design. Moreover, the new algorithm requires less area for a digit size of up to 8bits (for a 32bit quotient).

Introduction: The digit serial approach has recently been developed as a compromise between the fast and expensive bit parallel approach and the slow and cheap bit serial realisation [1]. The digit serial structure processes a number of bits n, called a digit at one cycle, where n varies between 1 bit and the wordlength, N. Digit serial notation is also used in the context of redundant number based systems which are also known as on-line arithmetic systems [2]. The drawback of the on-line systems is an increased size of the computational elements and the overhead of data conversion.

New digit serial algorithms and architectures based on radix-2^n arithmetic have recently been developed [3, 4]. Each quotient bit is generated in m cycles, where m is the number of radix-2^n digits of the word. The throughput rate of these architectures can be increased by pipelining the architecture to the digit level [3] or even the bit level [4]. However, the pipelining of the architecture will increase the latency and the number of latches (i.e. the area). We propose a modified version of the original digit serial algorithm of [3, 4], where k quotient bits can be generated in $m+k$ cycles. This results in a reduction of the computation time per quotient bit and, moreover, a reduction in area since fewer latches are required.

Digit serial division: We consider the division process $Q = A/D$, where the dividend A, the divisor D, and the quotient Q, are $2N$ bit, N bit, and N bit fixed point fractions, respectively. It is assumed that $|A| < |D|$ and $|D| \neq 0$. The input operands, A and D, are divided into $2m$ digits and m digits, respectively, where each digit consists of n bits (i.e. $N = mn$). They are fed in a digit serial form (digit-by-digit) starting from the least significant digits (LSDs). The N bit quotient, Q, is generated in a bit parallel form and can be converted to a digit serial form. The m-digit partial remainder (hereafter referred to as the remainder) PR_i, for $i = 1, ..., N$ is generated in a digit serial form starting from the LSD. It is assumed that all the data, A, D, Q, and PR_i are in two's complement form.

In the original digit serial division algorithm [3, 4], m cycles are required to generate each quotient bit q_i, for $i = 1, ..., N$. This bit is used to control the $(i+1)$th step of the algorithm such that an addition (subtraction) is carried out if $q_i = 0$ (1). Therefore, the addition (subtraction) operation of the $(i+1)$th step cannot be started until the quotient bit q_i of the ith step is generated. The delay between feeding the LSDs of the inputs at the ith step and generating q_i is m cycles. Therefore, the LSD (and also the other digits) of the ith remainder has to be delayed by m cycles before being processed in the next step. The throughput rate of this algorithm is one quotient bit every m cycles, and the latency is mN cycles. The throughput rate can be increased by pipelining the architecture to the bit level [4], but the area will increase and the latency will become mnN (i.e. N^2) cycles.

Proposed algorithm: As indicated earlier, the value of q_i in the ith step is either 0 or 1, and hence the operation in the $(i+1)$th step is either addition or subtraction. In the new algorithm, instead of waiting m cycles until q_i is generated, we start the $(i+1)$th step and use the generated digits of the remainder PR_i without any additional delay. Since q_i is not yet known, we carry out both operations in the $(i+1)$th step, i.e. addition and subtraction. The addition and subtraction are carried out in parallel, and two possible values of the remainder, PR_{i+1} at the $(i+1)$th step, are generated. Once the quotient bit q_i is obtained, it is used as a control mode to an n bit multiplexer to select one of the two possible values of the remainder, PR_{i+1}, such that $PR_{i+1} = 2PR_i+D$ if $q_i = 0$ and $PR_{i+1} = 2PR_i-D$ if $q_i = 1$. Therefore, the ith and the $(i+1)$th steps are overlapped, and the two steps are carried out in $m+1$ cycles instead of $2m$ cycles as in the original algorithm. This procedure can be generalised such that k steps are overlapped together to generate k quotient bits in $m+k$ cycles.

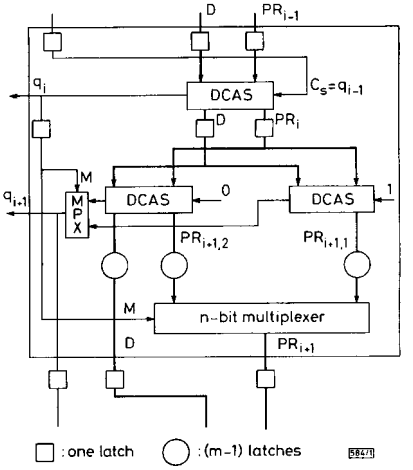

Fig. 1 *Basic digit serial cell*

Digit serial architecture: The basic cell used to implement this algorithm is shown in Fig. 1. For simplicity, we select $k = 2$ such that two quotient bits are generated every $m+1$ cycles. In this Figure, DCAS is an n bit digital controlled add/subtract cell [4], $PR_{i+1,1}$ and $PR_{i+1,2}$ are the possible values of the $(i+1)$th remainder, and PR_{i+1} is the correct value (i.e. PR_{i+1} equals either $PR_{i+1,1}$ or $PR_{i+1,2}$). The n bit multiplexer has two n bit inputs (one digit of $PR_{i+1,1}$ and the corresponding digit of $PR_{i+1,2}$) and one n bit output (one digit of PR_{i+1}) MPX is a 2 bit to 1 bit multiplexer used to select the quotient bit q_{i+1}. Both multiplexers are controlled by a control mode M (where $M = q_i$). The control signal C_s for the first DCAS cell equals q_{i-1}.

The area per quotient bit of the proposed structure A_p and the corresponding time T_p are now compared with the area A_0 and time T_0 of the original algorithm [3, 4] using the figures of merit in [5], with the area and time being given in units of two-input NAND gate. We also assume that the proposed architecture and that of [3, 4] are pipelined to the digit level. It should be noted that A_p and T_p are calculated as half the area and time, respectively, required to generate q_i and q_{i+1} as shown in the box of Fig. 1. The expressions for A_p, T_p, A_0, and T_0 and the corresponding values for $k = 2$ and $N = 32$ are as follows:

$$A_p = 1.5A_{DCAS} + (1.5mn + 0.5n + 1)A_{latch} + 0.5(n+1)A_{MPX} \qquad A_p = 355 + 23.5n$$

$$T_p = 0.5(m+1)T_{DCAS} \qquad T_p = 85 + 5m + 2.5n$$

$$A_o = A_{DCAS} + 2mnA_{latch} \qquad A_o = 455 + 12n$$

$$T_o = mT_{DCAS} \qquad T_o = 160 + 10m$$

where FA is a full adder, XOR is an exclusive-or gate, and MPX is a 2 bit to 1 bit multiplexer. It is assumed that a carry propagate adder is used in the DCAS cell, and hence the area of the DCAS cell is that of an n(FA+XOR)+latch, whereas the corresponding time is that of an nFA+XOR+latch. The area ratio A_p/A_0 and the

time ratios T_p/T_0 for different values of n (where $N = 32$ and $m = 32/n$) are given in Table 1.

Table 1: Area ratio A_p/A_0 and time ratio T_p/T_0

n	1	2	4	8	16
A_p/A_0, %	81	84	89	99	113
T_p/T_0, %	51	54	56	62	75

These results clearly illustrate the advantages of the new structure.

15 March 1995
Electronics Letters Online No: 19950648

A.E. Bashagha and M.K. Ibrahim (*Electrical and Electronic Engineering Department, University of Nottingham, Nottingham NG7 2RD, United Kingdom*)

References

1 PARHI, K.K.: 'A systematic approach for design of digit-serial signal processing architectures', *IEEE Trans.*, 1991, **CIS-38**, (4), pp. 358–375
2 IRWIN, M.J., and OWENS, R.M.: 'Digit pipelined arithmetic as illustrated by the paste-up system: A tutorial', *IEEE Computer*, 1987, **20**, pp. 61–73
3 BASHAGHA, A.E., and IBRAHIM, M.K.: 'A new digit-serial divider architecture', *Int. J. Electron.*, 1993, **75**, (1), pp. 133–140
4 BASHAGHA, A.E., and IBRAHIM, M.K.: 'A radix digit serial pipelined divider/square root architecture', *IEE Proc. E*, 1994, **141**, (6), pp. 375–380
5 HWANG, K.: 'Computer arithmetic: Principles, architecture, and design' (Wiley, New York, 1979)

Fuzzy uncertainty texture spectrum for texture analysis

Yih-Gong Lee, Jia-Hong Lee and Yuang-Cheh Hsueh

Indexing terms: Fuzzy set theory, Texture (image processing)

A new method using fuzzy uncertainty, which measures the uncertainty of the uniform surface in an image, is proposed for texture analysis. A grey-scale image can be transformed into a fuzzy image by the uncertainty definition. The distribution of the membership in a measured fuzzy image, denoted the fuzzy uncertainty texture spectrum (FUTS), is used as the texture feature for texture analysis.

Introduction: Texture analysis is an important technique in image processing. The major problem of texture analysis is the extraction of texture features. The general methods for feature extraction are to estimate local features at each pixel in a texture image and then derive a set of statistics from the distributions of the local features. The surveys and comparisons of different methods for feature extraction can be found in [1, 2].

A new method for texture feature extraction based on fuzzy theory is presented. Fuzzy set theory [3, 4] is a mathematical tool for modelling ambiguity or uncertainty and has been applied to image processing [5]. In texture analysis, we define the 'uniform surface uncertainty', which ranges from 0 to 1, for a point p in the texture as the degree of p belonging to the uniform physical surface (as defined by the neighbourhood average intensity). Therefore, we can transform a grey-scale image into a fuzzy image by using the uncertainty definition. For a rougher texture, the intensity of pixels in its corresponding fuzzy image will cause a smaller value. The fuzzy image membership distribution, denoted the fuzzy uncertainty texture spectrum (FUTS), is then used as a distinguishing feature for texture analysis.

Fuzzy uncertainty texture spectrum: A grey-scale image f can be transformed into a fuzzy image by a fuzzification function ϕ. A variety of fuzzification functions can be used to reflect the degree to which a pixel intensity represents a uniform physical surface. However, textural properties need neighbourhood information about the pixel to adequately define membership functions. A

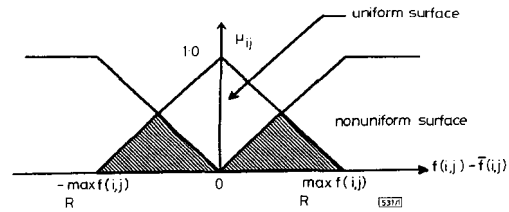

Fig. 1 *Fuzzy membership function for uniform surface*

simplified triangular membership function is used to define a uniform surface, as illustrated in Fig. 1. The uniform surface uncertainty is defined as

$$\mu_{ij} = 1 - \left[\frac{|f(i,j) - \bar{f}(i,j)|}{\max_R f(i,j)}\right] \quad (1)$$

where $\max_R f(i,j)$ is the maximum intensity within the $(\omega \times \omega)$ surface region R centred at point (i, j), and the average intensity is given by

$$\bar{f}(i,j) = \frac{1}{\omega \times \omega - 1} \sum_{\substack{R \\ m,n \neq i,j}} f(m,n) \quad (2)$$

Note that if $f(i, j)$ is equal to the average neighbourhood intensity $\bar{f}(i, j)$ then $f(i, j)$ possesses 'full membership' to the surface region R; i.e. $\mu_{ij} = 1$. Alternatively, if $f(i, j)$ is significantly different than the average neighbourhood intensity $\bar{f}(i, j)$, then $\mu_{ij} \to 0$.

To analyse a texture image, we can transform it into its corresponding fuzzy image by using eqn. 1. As the value in the fuzzy image represents the local aspect, the statistics of these values in the fuzzy image should reveal its texture surface information. The occurrence distribution of these values is called the fuzzy uncertainty texture spectrum (FUTS), with the abscissa indicating the belief degree and the ordinate representing its occurrence frequency. To evaluate the performance of the extracted feature by using the proposed method, we calculated and compared the FUTS for two Brodatz textures D77 and D90 [6], respectively. The two textures are shown in Figs. 3*g* and 3*h* and their corresponding FUTS are displayed in Fig. 2. From Fig. 2, we can find that the measured FUTS are distinguishable from each other so they can serve as a good discriminating tool in texture classification. The FUTS of D90 shows a higher frequency than D77 when the measured uncertainty is closed to unity, so that the texture image D90 is smoother than D77.

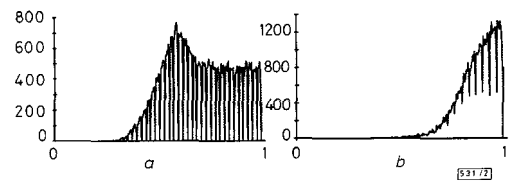

Fig. 2 *Fuzzy image and FUTS of textures D77 and D90*

a D77
b D90

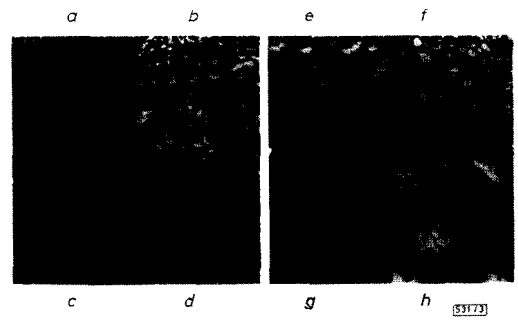

Fig 3 *Eight texture images extracted from Brodatz album*

a D4, *b* D5, *c* D9, *d* D15, *e* D18, *f* D54, *g* D77, *h* D90