J.S. Lee and J.W. Kim (*Advanced Devices Group, LG Electronics Institute of Technology, 16 Woomyeon-Dong, Seocho-Gu, Seoul 137-724, Korea*)

J.H. Lee and J.H. Lee (*School of Electronic Engineering & Computer Science, Kyungpook National University, Daegue, 702-201, Korea*)

C.S. Kim and J.E. Oh (*School of Electrical and Computer Engineering, Hanyang University, Sa-1-Dong, Ansan-Si, Kyunggi-Do 425-791, Korea*)

M.W. Shin (*Department of Ceramics Engineering, Myongji University, Nam-Dong, Yongin-Si, Kyunggi-Do 449-728, Korea*)

References

1 AKASAKI, I., and AMANO, H.: 'Crystal growth and conductivity control of group III nitride semiconductors and their application to short wavelength light emitters', *Jpn. J. Appl. Phys.*, 1997, **36**, pp. 5393–5408

2 CHEN, Q.Y., YANG, J.W., GASKA, R., ASIF KHAN, M., SHUR, M.S., SULLIVAN, G.J., SAILOR, A.L., HIGGINGS, J.A., PING, A.T., and ADESID, I.: 'High-power microwave 0.25-μm gate doped-channel GaN/AlGaN heterostructure field effect transistor', *IEEE Electron Device Lett.*, 1998, **19**, pp. 44–46

3 KELLER, S., WU, Y.F., PARISH, G., ZIANG, N., XU, J.J., KELLER, B.P., DENBAARS, S.P., and MISHRA, U.K.: 'Gallium nitride based high power heterojunction field effect transistor process development and present status at UCSB', *IEEE Trans. Electron Devices*, 2001, **48**, p. 552

4 TILAK, V., GREEN, B., KAPER, V., KIM, H., PRUNTY, T., SMART, J., SHEALY, J., and EASTMAN, L.: 'Influence of barrier thickness on the high-power performance of AlGaN/GaN HEMTs', *IEEE Electron Device Lett.*, 2001, **22**, pp. 504–506

5 KIM, J.W., LEE, J.S., SHIN, J.H., LEE, J.H., HAHM, S.H., LEE, J.H., KIM, C.S., OH, J.E., and SHIN, M.W.: 'Influence of pinhole type defects in AlGaN on rf performance of AlGaN/GaN HFETs grown by MOCVD', *Phys. Status Solidi A*, 2001, **188**, pp. 267–270

6 GREEN, B.M., CHU, K.K., CHUMBES, E.M., and EASTMAN, L.F.: 'The effects of surface passivation on the microwave characteristics of undoped AlGaN/GaN HEMT's', *IEEE Electron. Device Lett.*, 2000, **21**, pp. 268–270

7 SMITH, K.V., YU, E.T., REDWING, J.M., and BOUTROS, K.S.: 'Local electronic properties of AlGaN/GaN heterostructures probed by scanning capacitance microscopy', *J. Electron. Mater.*, 2000, **29**, pp. 274–280

8 SHEN, L., HEIKMAN, S., MORAN, B., COFFIE, R., ZHANG, N.Q., BUTTARI, D., SMORCHKOVA, I.P., KELLER, S., DEN BAARS, S.P., and MISHRA, U.K.: 'AlGaN/AlN/GaN high-power microwave HEMT', *IEEE Electron Device Lett.*, 2001, **22**, pp. 457–459

9 LEE, J.H., KIM, J.H., BAE, S.B., LEE, K.S., LEE, J.S., KIM, J.W., HAHM, S.H., and LEE, J.H.: 'Improvement of electrical properties of MOCVD grown $Al_xGa_{1-x}N$/GaN heterostructure with isoelectronic Al-Doped channel', *Phys. Status Solidi C*, 2003, **1**, pp. 240–243

Algorithm for modulo (2^n+1) multiplication

L.A. Sousa

An algorithm for designing efficient modulo (2^n+1) multipliers based on Booth recoding is proposed. With this algorithm, Wallace-tree adders can be used to design the fastest among all known modulo (2^n+1) multipliers.

Introduction: Modulo (2^n+1) multiplication has been widely used in the Fermat number transform and in residue number systems (RNS), with application in signal processing. In recent years, several modulo (2^n+1) multipliers have been proposed [1–3] for the diminished-1 number representation [4]. These multipliers use Wallace-tree addition and Booth recoding speed-up techniques but are less efficient than ordinary integer multipliers owing to the additional operations required to perform modulo reduction and the 2^n-correction.

In this Letter we propose an improved algorithm and architecture for diminished-1 modulo (2^n+1) multiplication which removes all the drawbacks of the cited architectures. The algorithm is also based on Booth recoding but it modifies the Booth tables and manipulates the modulo reduction expression for simplifying both the global modulo (2^n+1) reduction and the 2^n-correction.

Modulo (2^n+1) multiplication: The product (P) of two numbers (X, Y) modulo (2^n+1) can be formulated in diminished-1 representation as:

$$P = \langle (y \times x)_{2^n+1} + \bar{y}_n \times x + \bar{x}_n \times y + (x_n \vee y_n) \times 2^n \rangle_{2^n+1} \quad (1)$$

where z_n represents the nth bit and z the remaining n least significant bits of a diminished-1 coded number Z and $\langle B \rangle_{2^n+1}$ is the residue of the binary number B.

Booth recoding (radix-4) can be applied to speed-up the computation of the partial product terms in (1) $(x_{n+1} = x_n = x_{-1} = 0)$:

$$\langle y \times x \rangle_{2^n+1} = \left\langle \sum_{i=0}^{\lfloor n/2 \rfloor} (x_{2i-1} + x_{2i} - 2x_{2i+1}) 2^{2i} \times y \right\rangle_{2^n+1} \quad (2)$$

Since the reduction modulo (2^n+1) of a number A with at most $2n$ bits can be computed with a single subtraction:

$$\langle A \rangle_{2^n+1} = \langle A \bmod 2^n - A \operatorname{div} 2^n \rangle_{2^n+1} \quad (3)$$

the two types of partial products found in (2) can be generated modulo (2^n+1):

$$\begin{aligned} \langle 2^k \times y \rangle_{2^n+1} &\equiv \langle y_{n-k-1:0} \# \bar{y}_{n-1:n-k} - 2^k + 1 \rangle_{2^n+1} \\ \langle -2^k \times y \rangle_{2^n+1} &\equiv \langle \bar{y}_{n-k-1:0} \# y_{n-1:n-k} + 2^k + 1 \rangle_{2^n+1} \end{aligned} \quad (4)$$

where $y_{w:v}$ represents bits of y originally located in positions from v (less significant) to w (more significant) and the symbol # is used to concatenate bits.

The $\lfloor n/2 \rfloor + 1$ partial products in (2) must be added modulo (2^n+1). The multiplication algorithm, proposed in this Letter, manipulates both the partial products p_k (the only dependent on y terms) and derives a simple expression for adding modulo (2^n+1) the only dependent on k terms in (4), designated by correction terms (ct_k: $2 \times 0 \leq k \leq 2 \times \lfloor n/2 \rfloor$). It assumes the usage of modulo (2^n+1) adders for diminished-1 representation $(A+B+1 = A+B+\overline{c_{out}})$ [1].

Algorithm for modulo multiplication: Partial products (p_{2i}) and correction terms (ct_{2i}) in Table 1 can be used for computing $\langle y \times x \rangle_{2^n+1}$ with Booth recoding. They are directly derived from (4). We consider at this time that $y_n = 0$. Although $p_{2i}(000)$ and $p_{2i}(111)$ should have the zero value non-null values can be assigned to them by adjusting the corresponding correction terms $-(2^n+1)$ complement. The correction terms in Table 1 are diminished by one because each modulo (2^n+1) adder used for adding p_{2i} also adds an extra '1', as referred to in the preceding Section. By adding all the correction terms, by considering a common term $(-2^{2i} - 2^{2i+1})$ and by applying the distributive property, the final correction term (CT) can be formulated as:

$$\begin{aligned} CT = \Bigg\langle \sum_{k=2\times(i=\lfloor n/2 \rfloor)}^{2\times(i=\lfloor n/2 \rfloor)} & [(x_{k-1}\bar{x}_k\bar{x}_{k+1} \vee \bar{x}_{k-1}x_k\bar{x}_{k+1} \\ & \times \vee \bar{x}_{k-1}\bar{x}_k\bar{x}_{k+1}) \times 2^{k+1} + (x_{k-1}x_k\bar{x}_{k+1} \\ & \times \vee \bar{x}_{k-1}\bar{x}_k x_{k+1}) \times 2^k + x_{k+1} \times 2^{k+2} \\ & - (2^k + 2^{k+1})] + ct_{2\times 0} \Bigg\rangle_{2^n+1} \end{aligned} \quad (5)$$

It is easy to prove the correctness of the following equations, where $n_0 = 1$ for n odd:

$$\begin{aligned} \left\langle -\sum_{i=1}^{\lfloor n/2 \rfloor} (2^{2i} + 2^{2i+1}) \right\rangle_{2^n+1} &= 6 + 2 \times \bar{n}_0 \\ \sum_{i=1}^{\lfloor n/2 \rfloor} x_{2i+1} \times 2^{2i+2} &= \sum_{i=1}^{\lfloor n/2 \rfloor} (x_{2i-1} \times 2^{2i}) - x_1 \times 2^2 \end{aligned} \quad (6)$$

Therefore, (5) can be rewritten as:

$$CT = \Bigg\langle \sum_{k=2\times(i=\lfloor n/2 \rfloor)}^{2\times(i=\lfloor n/2 \rfloor)} \overbrace{[(\bar{x}_k x_{k+1} \vee x_{k-1}x_{k+1} \vee x_{k-1}\bar{x}_k) \times 2^k + \bar{x}_{k+1} \times 2^{k+1}]}^{\psi} + \underbrace{ct_{2\times 0} - 4 \times x_1 + 6 + 2 \times \bar{n}_0}_{\phi} \Bigg\rangle_{2^n+1} \quad (7)$$

For diminished-1 multiplication, modulo (2^n+1) addition of y and x has to be performed whenever x_n and y_n are equal to zero, respectively (see (1)). The y term can be added to p_0 whenever $x_n=0$, which leads to the first four rows in Table 2 and the following expression for the ϕ term in (7):

$$\phi = 2 \times \bar{x}_1\bar{x}_0 + \bar{x}_1 x_0 - 2 \times x_1 x_0 + 3 + 2 \times \bar{n}_0 \quad (8)$$

In Table 2, the x_n bit takes the place of the x_{-1} bit, which is always equal to zero, and the zero value is assigned to p_0 when $x_n=1$. The x_n bit is not present in (8) since the value assigned to $ct_{2\times 0}$ is not dependent on the value of x_n (see Table 2). For obtaining a simple logic equation for CT the term $-2 \times x_1 x_0$ has to be eliminated. For $y_n=1$, the value $-2 \times x_1 x_0$ can be directly added modulo (2^n+1) to p_0 by changing the value of $p_0(011)$ from '0' to '2^n-1' (see Table 2). For $y_n=0$, we can take advantage of the fact that $2 \times x_1 + x_0$ (see (1)) must be added to ϕ. Thus ϕ' takes the place of ϕ in (7) for both values of y_n:

$$\phi' = 2^1 \times (\bar{y}_n\bar{x}_1 \vee \bar{y}_n x_1\bar{x}_0 \vee y_n\bar{x}_1\bar{x}_0) + 2^0 \times (\bar{y}_n x_1 x_0 \vee y_n\bar{x}_1 x_0) + 3 + 2 \times \bar{n}_0 \quad (9)$$

To add the remaining bits of x when $y_n=0$, the terms $x_{2i} \times 2^{2i}$ and $x_{2i+1} \times 2^{2i+1}$ must be added to ψ in (7) for all $1 \le i \le \lfloor n/2 \rfloor$ $(k=2\times i)$:

$$(\bar{x}_k x_{k+1} \vee x_{k-1}\bar{x}_k \vee x_k) \times 2^k + (\bar{x}_{k-1} \vee \bar{x}_k \vee \bar{x}_{k+1})2^{k+1} + x_{k-1}x_k x_{k+1}(2^{k+1}+2^k) \quad (10)$$

Once again, to avoid undesirable arithmetic operations for CT computation, the term $(2^{2i}+2^{2i+1})$ is added to the corresponding partial product for $y_n=0$, leading to $p_{2i}(111)=2^{2i+1}$ (see Table 1). So, the term ψ in (7) is replaced by ψ' for both values of y_n:

$$\psi' = (\bar{x}_k x_{k+1} \vee x_{k-1}\bar{x}_k \vee y_n x_{k-1}x_{k+1} \vee \bar{y}_n x_k) \times 2^k + (\bar{x}_{k+1} \vee \bar{y}_n\bar{x}_k \vee \bar{y}_n\bar{x}_{k-1}) \times 2^{k+1} \quad (11)$$

By isolating the terms of ψ' for $k=2\times(i=\lfloor n/2 \rfloor)$ in (7) and by operating logic simplifications of them together with the ϕ' (9), the simple logic equations (12) and (13) are obtained to compute CT for n odd and n even, respectively. These equations already include the value of the nth bit of the result, with $\theta = 2^n \times (x_n \vee y_n)$.

$$CT = \theta + \left\langle \left[\sum_{k=2\times(i=1)}^{2\times(i=(n-3)/2)} \psi' \right] + 2^{n-1} \times (x_{n-2}\bar{x}_{n-1} \vee \bar{y}_n x_{n-1}) + 2^1 \times (\bar{y}_n\bar{x}_1 \vee \bar{y}_n x_1\bar{x}_0 \vee y_n\bar{x}_1\bar{x}_0) + 2^0 \times (\bar{y}_n x_1 x_0 \vee y_n\bar{x}_1 x_0) + 2 \right\rangle_{2^n+1} \quad (12)$$

$$CT = \theta + \left\langle \left[\sum_{k=2\times(i=1)}^{2\times(i=(n-2)/2)} \psi' \right] + 2^1 \times (\bar{y}_n\bar{x}_{n-1} \vee \bar{x}_1\bar{x}_0 \vee \bar{y}_n\bar{x}_1 \vee \bar{y}_n\bar{x}_0 \vee \bar{x}_{n-1}\bar{x}_1) + 2^0 \times (\bar{x}_{n-1}\bar{x}_0 \vee \bar{y}_n\bar{x}_{n-1}\bar{x}_1 \vee \bar{y}_n x_{n-1}x_1 x_0 \vee y_n x_{n-1}\bar{x}_1 x_0 \vee y_n\bar{x}_{n-1}x_1) + 2 \right\rangle_{2^n+1} \quad (13)$$

Thereby, modulo (2^n+1) multiplication can be computed using (14), with the values of p_0, p_{2i} and CT found in Table 2, Table 1 and (12) or (13), respectively. It is important to note that the correction term depends on x and y_n by simple logic equations.

$$P = \left\langle \left(\sum_{i=0}^{\lfloor n/2 \rfloor} (p_{2i}+1) + CT + 1 \right) + 1 \right\rangle_{2^n+1} \quad (14)$$

The fast modulo (2^n+1) multiplier proposed in Fig. 1 was modelled in structural VHDL for a general value of n and its operation was exhaustively verified. It is composed of a set of logic gates for generating CT, a Wallace-tree of modulo (2^n+1) carry-save adders (to add p_{2i} and CT), and a final modulo (2^n+1) carry-propagate adder. The unitary terms in (14) are automatically added, because diminished-1 adders are used. In the proposed architecture, the number of stages of the Wallace-tree is at most one more than the number required for ordinary integer multiplication, whenever $(\lfloor n/2 \rfloor + 1)$ is the upper limit value of the number of operands accommodated by a given number of stages [1]. This is a significant improvement regarding the fastest known modulo (2^n+1) architectures [1–3]. The multiplier in [1] does not apply Booth recoding and requires the precomputation of a data-dependent correction term, which introduces an additional delay of some full-adders to implement an $(n-1)$-bit counter. The introduction of Booth recoding in [2] implies the addition of a supplementary value which depends on n and requires two additional carry-save adder stages in front of the carry-save adder tree for modulo reduction. The modulo (2^n+1) multiplier proposed in [3], that also applies bit-pair recoding and a Wallace-tree adder, can be adapted for diminished number representation but explicitely requires the addition of the terms $\bar{x}_n \times y$ and $\bar{y}_n \times x$, and the special case of X, $Y=0$ has to be treated separately.

Table 1: Partial products $(p_{2i};\ 1 \le i \le \lfloor n/2 \rfloor)$ for diminished-1 representation

x_{2i+1}	x_{2i}	x_{2i-1}	$p_{2i}(x_{2i+1}x_{2i}x_{2i-1})$	ct_{2i}
0	0	0	'$0\cdots01\cdots1$' $\equiv 2^{2i}-1$	2^n+1-2^{2i}
0	0	1	$y_{n-2i-1:0}\#\bar{y}_{n-1:n-2i}$	-2^{2i}
0	1	0	$y_{n-2i-1:0}\#\bar{y}_{n-1:n-2i}$	-2^{2i}
0	1	1	$y_{n-2i-2:0}\#\bar{y}_{n-1:n-2i-1}$	-2^{2i+1}
1	0	0	$\bar{y}_{n-2i-2:0}\#y_{n-1:n-2i-1}$	2^{2i+1}
1	0	1	$\bar{y}_{n-2i-1:0}\#y_{n-1:n-2i}$	2^{2i}
1	1	0	$\bar{y}_{n-2i-1:0}\#y_{n-1:n-2i}$	2^{2i}
1	1	1	'$y_n\cdots y_n$]$\bar{y}_n\cdots\bar{y}_n$' $\equiv 2^n-2^{2i}\vert_{y_n=1}\vee 2^{2i+1}-1\vert_{y_n=0}$	2^{2i}

Table 2: Partial product (p_0) for diminished-1 representation

x_n	x_1	x_0	$p_{2\times 0}(x_n x_1 x_0)$	$ct_{2\times 0}$
0	0	0	$y_{n-1:0}$	-1
0	0	1	$y_{n-2:0}\#\bar{y}_{n-1}$	-2
0	1	0	$\bar{y}_{n-1:0}$	$+1$
0	1	1	'$y_n\cdots y_n$' $\equiv 2^n-1\vert_{y_n=1}\vee 0\vert_{y_n=0}$	-1
1	0	0	'$0\cdots0$'	-1

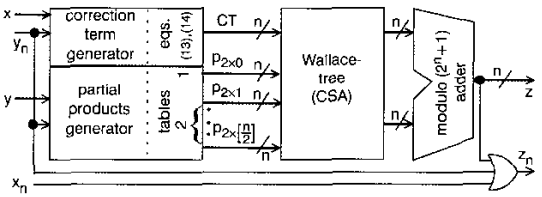

Fig. 1 *Proposed architecture for modulo (2^n+1) multiplication*

The cost of the proposed modulo (2^n+1) multipliers is only slightly greater than the cost of the binary multipliers. In spite of being faster than the other modulo (2^n+1) architectures, the new architecture does not increase the hardware requirements.

Conclusions: We have proposed an improved algorithm for designing efficient modulo (2^n+1) multipliers. By manipulating the Booth tables and by applying a simple correction term, the proposed multiplier is the most efficient among all the known modulo (2^n+1) multipliers and is almost as efficient as those for ordinary integer multiplication.

31 January 2003
Electronics Letters Online No: 20030467
DOI: 10.1049/el:20030467

L.A. Sousa (*Department of Electrical Engineering, Instituto Superior Técnico/INESC-ID, R. Alves Redol, 9, 1000-029 Lisboa, Portugal*)

E-mail: las@inesc-id.pt

References

1 WANG, Z., JULLIEN, G.A., and MILLER, W.C.: 'An efficient tree architecture for modulo (2^n+1) multiplication', *J. VLSI Signal Process. Syst.*, 1996, **14**, (3), pp. 241–248

2 MA, Y.: 'A simplified architecture for modulo (2^n+1) multiplication', *IEEE Trans. Comput.*, 1998, **47**, (3), pp. 333–337
3 ZIMMERMANN, R.: 'Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication' . Proc. IEEE Symp. on Computer Arithmetic, Adelaide, Australia, April 1999, pp. 158–167
4 LEIBOWITZ, L.: 'A simplified binary arithmetic for the Fermat number transform', *IEEE Trans. Acoust. Speech Signal Process.*, 1976, **24**, (5), pp. 356–359

Adaptive noise estimation algorithm for speech enhancement

L. Lin, W.H. Holmes and E. Ambikairajah

An efficient noise estimation algorithm for speech enhancement is proposed. The noisy speech is decomposed into subband signals and the subband noise estimate is updated by adaptively smoothing the noisy signal power. The smoothing parameter is chosen as a function of the estimated signal-to-noise ratio (SNR). This noise estimation technique gives reliable results even at very low SNRs.

Introduction: In most applications involving speech enhancement, we have only a noisy speech signal available. The noise may be non-stationary and coloured, and its power is unknown. Its properties must be extracted from the noisy speech signal alone. Noise power estimation is crucial to effective speech enhancement. Inaccurate noise power used in the suppression rule can result in musical noise and speech distortion. In recent years, noise estimation techniques have been developed based on tracking the spectral minima in each frequency band [1–3]. In this Letter, we propose a new simple and reliable noise estimation technique for speech enhancement. The noise estimate is updated adaptively and continuously, with a smoothing parameter that depends on the estimated subband SNR.

Proposed noise estimation technique: Noisy speech $x(n)$, consisting of clean speech $s(n)$ with additive noise $w(n)$, is first decomposed into M bandpass signals $x_i(n)$ using a filterbank. In the implementation reported here we use the auditory filters proposed Lin *et al.* [4] to split the signal into critical band signals, although other bandpass filterbanks could be used (including the common Fourier decomposition).

The output of the ith critical band filter (i.e. the ith noisy subband signal) is given by

$$x_i(n) = h_i(n)*x(n) \equiv s_i(n) + w_i(n) \tag{1}$$

where $s_i(n) = h_i(n)*s(n)$ is the output from the ith critical band filter when the input to the filterbank is clean speech only, and $w_i(n) = h_i(n)*w(n)$ is the corresponding output when the input is noise only.

We assume that noise and speech are independent non-stationary signals, but that the noise power changes relatively slowly. The subband noisy signal power, $\sigma_{x_i}^2 = E\{x_i^2(n)\}$, is estimated on a frame-by-frame basis using

$$\hat{\sigma}_{x_i}^2(p) = \frac{1}{N}\sum_{n=0}^{N-1} x_i^2(pN+n) \tag{2}$$

where $\hat{\sigma}_x^2(p)$ is the estimated noisy signal power calculated using frame p, and N is the frame size.

The subband noise power, $\sigma_{w_i}^2 = \mathrm{E}\{w_i^2(n)\}$, is estimated using the one-pole smoothing filter

$$\hat{\sigma}_{w_i}^2(p) = \alpha_i(p)\hat{\sigma}_{w_i}^2(p-1) + (1-\alpha_i(p))\hat{\sigma}_{x_i}^2(p) \tag{3}$$

where $\hat{\sigma}_{w_i}^2(p)$ is the estimate of subband noise power in frame p.

The smoothing parameter $\alpha_i(p)$ at frame p is chosen as a Sigmoid function changing with $SNR_i(p)$

$$\alpha_i(p) = \frac{1}{1+e^{-a(SNR_i(p)-T)}} \tag{4}$$

where $SNR_i(p) = \hat{\sigma}_{x_i}^2(p)/\bar{\sigma}_{w_i}^2(p-1)$, and $\bar{\sigma}_{w_i}^2(p-1)$ is the average or median of the noise estimates of the previous 5 to 10 frames, e.g. $\bar{\sigma}_{w_i}^2(p-1) = 1/10\sum_{k=1}^{10} \hat{\sigma}_{w_i}^2(p-k)$. The variable $SNR_i(p)$ can be considered as an approximation to the *a posteriori* signal-to-noise ratio $\sigma_{x_i}^2/\sigma_{w_i}^2 = (\sigma_{s_i}^2+\sigma_{w_i}^2)\sigma_{w_i}^2$. Theoretically the *a posteriori* SNR should always be 1 when only noise is present and greater than 1 when both speech and noise are present. The parameter a in (4) is a constant with a value between 15 to 30, and the value of T is around 1.5. A plot of α_i against the *a posteriori* signal-to-noise ratio $\sigma_{x_i}^2/\sigma_{w_i}^2$ at different values of a is shown in Fig. 1.

The foregoing algorithm can be explained as follows. If speech is absent in frame p, the new noisy signal power calculation $\hat{\sigma}_{x_i}^2(p)$ should be very close to the average noise estimate $\bar{\sigma}_{w_i}^2(p-1)$, so that, from (4) and Fig. 1, $\alpha_i(p) \simeq 0$. Also, from (3) we have $\hat{\sigma}_{w_i}^2(p) \simeq \hat{\sigma}_{x_i}^2(p)$, because of the small value of $\alpha_i(p)$, i.e. the estimate of noise power in frame p rapidly follows the power of the noisy signal in the absence of speech—there is minimal smoothing.

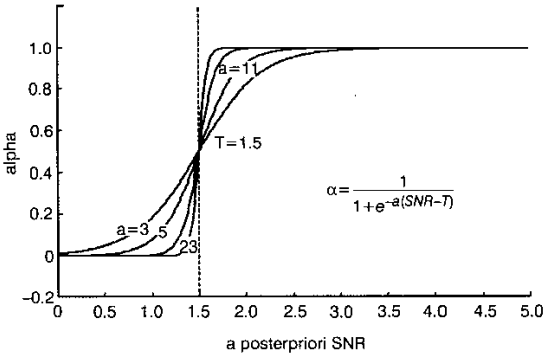

Fig. 1 *Plot of α against a posteriori SNR*

Conversely, if speech is present, the new noisy signal power $\hat{\sigma}_{x_i}^2(p)$ is much larger than the previous noise estimate $\bar{\sigma}_{w_i}^2(p-1)$, i.e. $SNR_i(p) > T$, so that, from (4) and Fig. 1, $\alpha_i(p) > 0.5$. Hence the noise update in (3) is slower because of the large value of $\alpha_i(p)$. The value of $\alpha_i(p)$ increases rapidly with increasing $\hat{\sigma}_{x_i}^2(p)/\bar{\sigma}_{w_i}^2(p-1)$. During voiced frames we have $\hat{\sigma}_{x_i}^2(p) \gg \bar{\sigma}_{w_i}^2(p-1)$, $\alpha_i(p) \simeq 1$, and $\hat{\sigma}_{w_i}^2(p) \simeq \hat{\sigma}_{w_i}^2(p-1)$, i.e. the noise update process almost stops and the noise estimate approximately equals that of the previous frame because the value of $\alpha_i(p)$ is almost 1.

The parameter a in (4) controls the way in which $\alpha_i(p)$ changes with $\hat{\sigma}_{x_i}^2(p)/\bar{\sigma}_{w_i}^2(p-1)$. Generally, larger values of a lead to larger values of $\alpha_i(p)$ and slower noise updates, whereas smaller values of a give faster noise updates, at the risk of possible over-estimation during long voiced intervals.

To further smooth the noise estimate, the following filtering operation is used to obtain the final noise estimate $\hat{\sigma}_{w_i,final}(p)$, where α_{final} is chosen to be around 0.5–0.95:

$$\hat{\sigma}_{w_i,final}^2(p) = \alpha_{final}\hat{\sigma}_{w_i,final}^2(p-1) + (1-\alpha_{final})\hat{\sigma}_{w_i}^2(p) \tag{5}$$

This operation reduces fluctuation in the noise power estimate but introduces a small amount of delay in the response to noise power changes. The estimate for the subband clean signal power, $\sigma_{s_i}^2 = E\{s_i^2(n)\}$, is then calculated using

$$\hat{\sigma}_{s_i}^2(p) = \max[\hat{\sigma}_{x_i}^2(p) - \hat{\sigma}_{w_i,final}^2(p), 0] \tag{6}$$

where $\hat{\sigma}_{s_i}^2(p)$ is the estimate of subband clean speech power at frame p. The operation $\max\{\cdot\}$ is used to avoid negative power estimates.

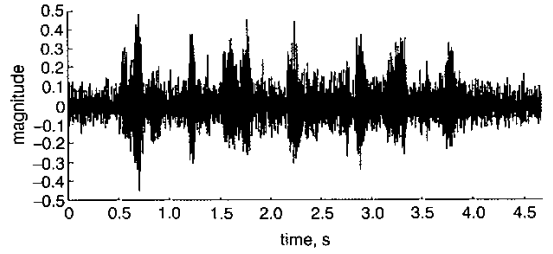

Fig. 2 *Noisy speech*