Семинар на катедра “ Компютърни Системи и Технологии”
ТУ София – Филиал Пловдив
21 ноември 2008 г.

ТЕМА: Обсъждане на частични резултати от дисертационният труд на гл. ас. Иван Т. Кънев

“ИЗСЛЕДВАНЕ НА ЕДИН КЛАС СИСТЕМИ ЗА ГЛАСОВИ СЪОБЩЕНИЯ”

Актуалност на проблема

Системи за гласови съобщения използват компютърни методи за генериране на речеви съобщения по зададени параметри.

Тези системи са предназначени за потребители, които използват за комуникации гласови медии.

Развитието на FPGA технологиите позволява да се разширят сферите на тяхното приложение, като те станат част от по-големи системи, реализирани съвместно върху един програмируем чип (SoPC - System on a Programmable Chip).

FPGA базираните системи за гласови съобщения, предполагат нови подходи при тяхното изследване и проектиране: създаването на специализирани процесори, хардуерни декодери и интерфейсни устройства; развитие и реконфигуриране след въвеждане в експлоатация.

Предмет на дисертационния труд са изследвания свързани с един клас системи за гласови съобщения.

Терминът „клас” отразява ограниченията които са наложени на изследванията в дисертационния труд:

- системата трябва да е FPGA базирана, като заема част от ресурсите на платформата;
- да генерира висококачествени гласови съобщения с използването на ограничен речник състоящ се от числителни имена, думи, фрази и изречения;
- речника да е съставен от компресирани речеви образци.

Цели на дисертационния труд

Основната цел на дисертацията е: изследване, анализ и усъвършенстване на ADM алгоритми за компресиране на речеви сигнали, синтезиране на висококачествени гласови съобщения и изграждане на системи за тяхното реализиране с FPGA базирани платформи.

Задачи на дисертационния труд

- Да се изследват възможностите за компресиране на речеви сигнали с модифицирани ADM (*MADM(r)* - Modified Adaptive Delta Modulation) алгоритми с оглед реализирането им на микропроцесорни платформи и подобряване на адаптивните характеристики на алгоритмите.
- Да се представят автоматно и таблично на *MADM(r)* алгоритмите за да се оптимизират операциите при софтуерната и хардуерната им реализация.
- Да се предложат алгоритми за коригиране на грешката при декомпресиране с *MADM(r)* алгоритми, с оглед повишаване на качество на генерираните съобщения.
- Да се разработят алгоритми и синтезатор за висококачествен конкатенационен синтез на гласови съобщения, базиран на числителни имена, думи, фрази и изречения
- Да се проектира FPGA базиран микроконтролер за синтезиране на гласови съобщения с вграден нардуерен *MADM(r)* декодер

Методи на изследване

- За изследване поведението на предложените *MADM(r)* алгоритмите и тяхното автоматно и таблично представяне са използвани методите на математическата индукция и теория на автоматите. При оценяване качеството на генерираните гласови съобщения е използвана методика базирана на критерия MOS (Mean Opinion Score).
- Разработени са специализирани програми с които експериментално е доказано, че с използването на предложения алгоритъма за коригиране на грешката при декомпресиране се повишава качеството на генерираните гласовите съобщения.
- При проектирането и изследването на FPGA базирани системи за гласови съобщения са използвани вградените в инструменталните средства симулатори.

Приложимост и полезност

- Рзултатите от изследванията са намерили приложения при изпълнение на договори:

  I-274/10.06.2005 - *“Проектиране на FPGA базирани микроконтролери”* и I-468/ 14.06.2006- *“Проектиране и изследване на FPGA базирани самостоятелни системи за гласови съобщения”* по конкурсната програма „Научно изследователски проекти” на ТУ- София, Филиал Пловдив.

- Получените в дисертационния труд резултати, могат да се използват при проектирането на SoPC, FPGA базирани системи за гласови съобщения, които използват за комуникации гласови медии.

Апробация

- Резултатите от дисертационния труд са докладвани и обсъждани на:
- семинари на катедра „ Компютърни системи и технологии” към ТУ- София, Филиал Пловдив – март 2004 и октомври 2008.
- Международна конференция по компютърни системи и технологии CompSysTech Велико Търново, 2006 г.
- Международна конференция по компютърни системи и технологии CompSysTech, Варна, 2005 г.
- Национална юбилейна научна конференция на Русенски университет „Ангел Кънчев”, Русе, 2005 г.
- Национална научна конференция „ 10 години катедра Компютърни системи към ТУ-София, филиал Пловдив”, 2003 г.

Глава 1. Литературен обзор

Разгледани са и систематизирани системите за гласови съобщения. Направен е обзор в следните по-значими за тези системите направления:

- *Методи за компресиране на речеви сигнали*.

  Разгледани са методи, които се базират на спектралните характеристики на речевите сигнали в определен сегмент (vocoders) и методи които се базират на моментните характеристики на речевите сигнали (waveform).

- *Методи за синтезиране на гласови съобщения*.

  Разгледани са методи методи, които се базират на сегментната система на речта (форманти, фонеми, дифони) и конкатенационни методи (числителни имена, думи, фрази и изречения).

- *Платформи за изграждане на системи за синтезиране на гласови съобщения*.

  Разгледани са платформи базирани на конвенционални процесори, DSP и FPGA.

От извършения литературен обзор се налагат следните изводи:

- Методите за компресиране на речеви сигнали с ADM алгоритми не са директно приложими за изчисляване с микропроцесорни платформи, което налага тяхното модифициране.
- При изчисляване на ADM алгоритми се използват операции умножение, деление, сравняване по модул, което може да се избягне с разработване на методи за тяхното автоматно и таблично представяне.
- Съществен недостатък на ADM алгоритми е ниското качество на генерираните съобщения, което налага разработването на нови методи за коригиране на грешката при декомпресиране.
- Използването на конкатенационните синтезатори налага разработването на алгоритми за синтезиране количествените числителни имена в българския език и дефинирането на нови числови формати
- Системи за гласови съобщения базирани върху FPGA платформи са сравнително слабо са проучени. За да се използват преимуществата които предлага тази технология, трябва да се разработят нови специализирани процесори и контролери.

Глава 2. Компресиране на речеви сигнали с модифицирани ADM алгоритми

В глава 2. се поставят за решаване следните основни задачи:

2.1 Въвеждането на някои ограничения при изчисляване на адаптивната функция, и на тяхна база дефинирането на модифицирани ADM алгоритми с цел реализирането им на микропроцесорни платформи.

2.2 Автоматно и таблично представяне на модифицирани ADM алгоритми с цел:

а. Реализиране на алгоритмите без използване на операциите умножение, деление и сравняване по модул.

б. Реализиране на различните алгоритми с единни софтуерни и хардуерни средства .

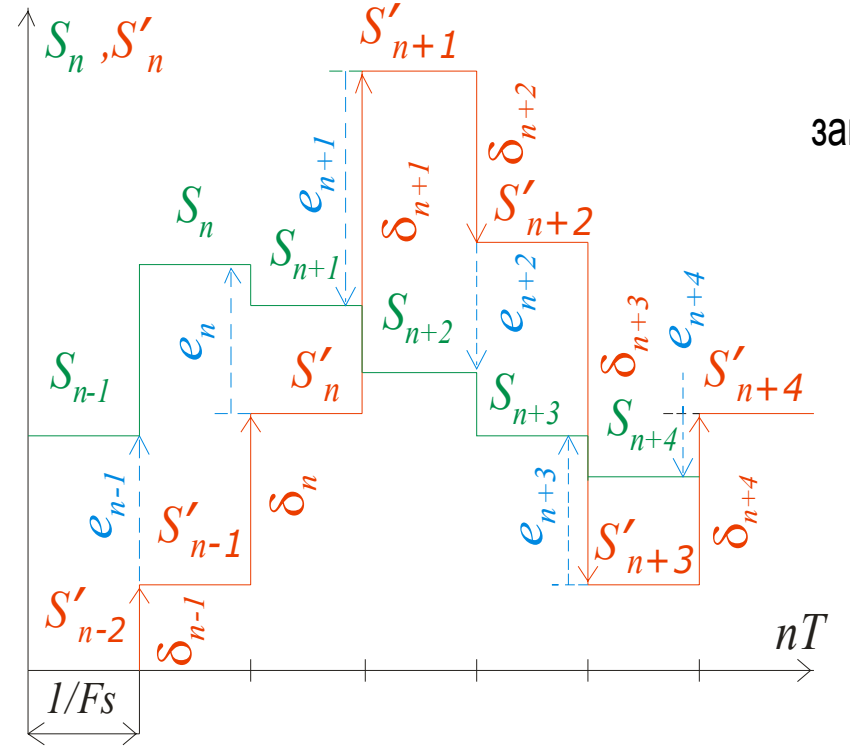
2.3. Създаване на алгоритми за намаляване на грешката при декомпресиране.

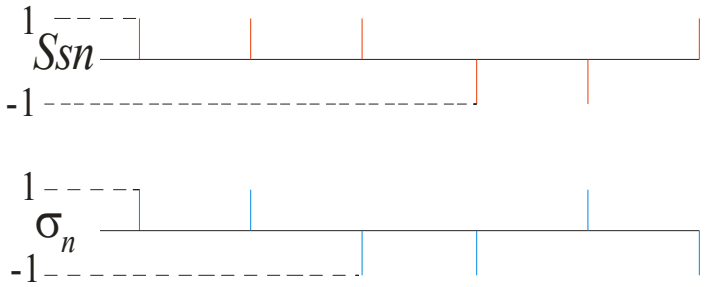
2.1 Модифицирани ADM алгоритми
2.1.1 Основни определения

● Вследствие дискретизирането на речев сигнал $S(t)$ с непрекъснат параметър t в дискретен момент от времето T, се получава дискретната редица $\{S_{(nT)}\}$ или само $\{S_n\}$, където $n = 1..N$ е целочислена променлива, а N е броя на дискретите, необходими за представяне на речевия сигнал.

● Отношението $Fs = 1/T$ определя честотата на дискретизиране. За апроксимирането на $S(t)$ при равномерно дискретизиране $(Fs = const.)$ са необходими L нива. Като следствие на дискретизирането се получава блоков код с дължина $B_m = \log_2 L$, където с m е означен броя на двоичните разряди за представяне на Sn.

● При прилагане на алгоритъма ADM (фиг. 2.1) върху дискретната редица $\{S_n\}$, се поражда нова дискретна редица $\{S'_n\}$.





Фиг. 2.1.1. Компресиране - декомпресиране с алгоритъма *ADM*

Алгоритъма ADM се базира на рекурентната зависимост:

$$S'_n = S'_{n-1} + \alpha\delta_{n-1}, \text{ където:} \qquad (2.1.1)$$

- δ_{n-1} е относителната промяна на S'_n:

$$\delta_{n-1} = S'_{n-1} - S'_{n-2}; \qquad (2.1.2)$$

- α е функция която определя адаптивния характер на алгоритъма.

Функцията α се определя от условието:

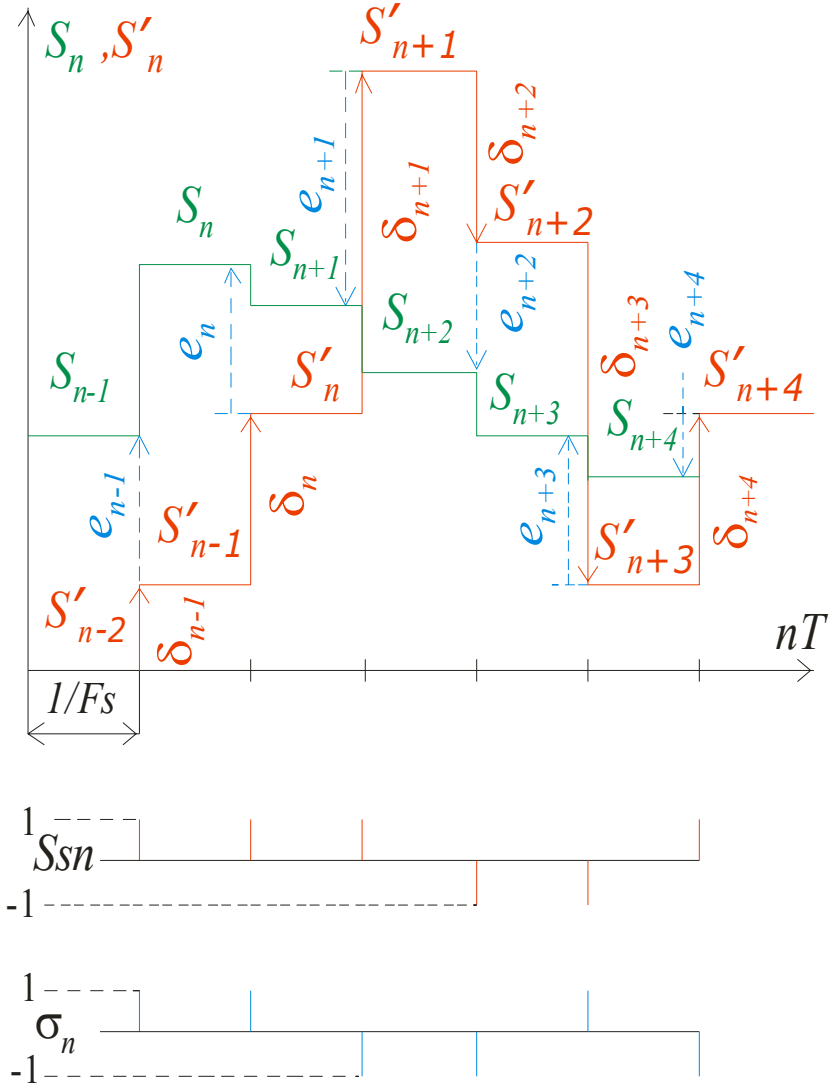
$$\alpha = \begin{Bmatrix} \alpha_+, \text{IF } \sigma_n = Ss_{n-1} \\ \alpha_-, \text{IF } \sigma_n \neq Ss_{n-1} \end{Bmatrix}, \text{ където:} \qquad (2.1.3)$$

- Ss_{n-1} е променлива определяща знака на δ_{n-1}:

$$Ss_{n-1} = sig\ \delta_{n-1}, \quad Ss_{n-1} = 1, -1; \qquad (2.1.4)$$

- α_+, α_- са коефициенти които определят нарастването на S'_n : $\alpha_+ = 2$, $\alpha_- = -\alpha_+^{-1}$.(2.1.5)

В зависимост от условието σ_n, при което се определя начина на получаване на адаптивната функция α са възможни две операции с алгоритъма (2.1.1)



Фиг. 2.1.1. Компресиране - декомпресиране с алгоритъма *ADM*

1. Компресиране

Операцията **C**, реализираща алгоритъма (2.1.1) върху дискретната редица $\{S_n\}$, при което се поражда нова дискретна редица $\{Ss_n\}$

$$\{S_n\} \overset{C}{\Longrightarrow} \{Ss_n\}, \qquad (2.1.6)$$

се нарича ***компресиране с алгоритъма ADM.***

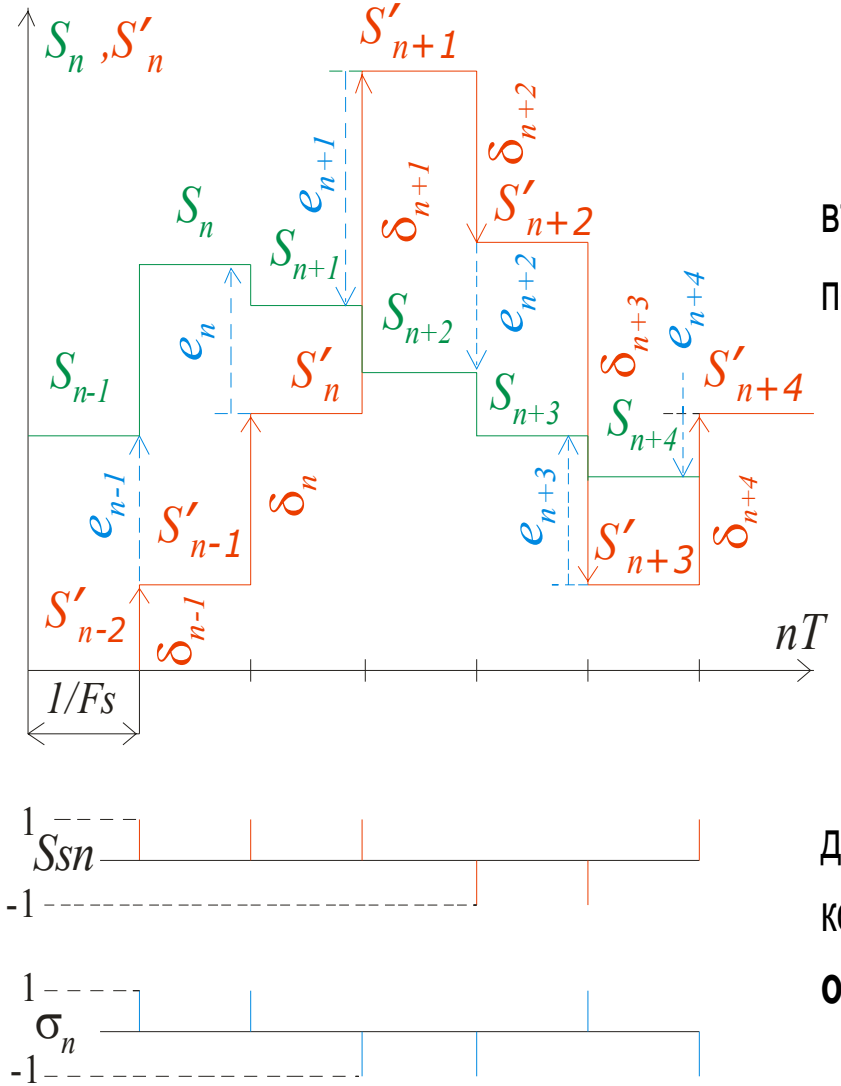
Ако с e_{n-1} означим грешката с която се извършва трансформацията (2.1.6)

$$e_{n-1} = S_{n-1} - S'_{n-1}, \qquad (2.1.7)$$

тогава σ_n се определя от знака на грешката e_{n-1}:

$$\sigma_n = \begin{cases} 1 \text{ IF } e_{n-1} > 0 \\ -1 \text{ IF } e_{n-1} \le 0 \end{cases}. \qquad (2.1.8)$$

Операцията **C** трансформира блоковия код S_n, с дължина B_m до блоков код Ss_n с дължина 1, при което се осъществява **компресиране в отношение** $B_m:1$.



Фиг. 2.1.1. Компресиране - декомпресиране с алгоритъма *ADM*

2. Декомпресиране

Операцията ***D***, реализираща алгоритъма (2.1.1) върху дискретната редица $\{Ss_n\}$, при което се поражда нова дискретна редица $\{S'_n\}$:

$$\{Ss_n\} \overset{D}{\Longrightarrow} \{S'_n\} \qquad (2.1.9)$$

се нарича **декомпресиране.**

Тогава σ_n се определя от равенството

$$\sigma_n = Ss_n \qquad (2.1.10)$$

Операцията ***D*** трансформира блоковия код Ss_n, с дължина 1 до блоков код S'_n с дължина B_m , при което се осъществява **декомпресиране в отношение** $1 : B_m$ и се възстановява S'_n .

2.1.4. Ограничаване на нарастването на функцията $\alpha\delta_{n-1}$. Определение 2.1.1

Да приемем, че оригиналния речев сигнал S_n е представен в допълнителен код $(S_n)_{2cc}$

$$(S_n)_{2cc} = -\mathrm{b}_{m-1}.2^{m-1} + \sum_{m=0}^{m-2} b_m 2^m , \qquad (2.1.11)$$

и $(S_n)_{cc}$ е ограничена в диапазона

$$-2^{m-1} \le (S_n)_{2cc} \le 2^{m-1} - 1. \qquad (2.1.12)$$

Да допуснем, че произведението $|\alpha\delta_{n-1}|$ е ограничено от условието

$$|\alpha\delta_{n-1}| \le \delta_{\max} \qquad (2.1.13)$$

където: $\delta_{\max}$ е коефициент ограничаващ нарастването на δ_{n-1},

$$\delta_{\max} = 2^k , \quad k \ge 1. \qquad (2.1.14)$$

Тогава при изчисление на алгоритъма (2.1.1), S'_n няма да надхвърли дефиниционната област (2.1.12) само ако е изпълнено неравенството:

$$\left|\left(S_n\right)_{2cc}\right| \le 2^{m-1} - 1 - \delta_{\max} . \qquad (2.1.15)$$

Следствие 2.1.1

Ако за адаптивната функция α *е изпълнено условието:*

$$\alpha = \begin{cases} \alpha_+ , \ \mathrm{IF}\,\sigma_{\mathrm{n}} = Ss_{n-1} \ \mathrm{and} \left|\delta_{\mathrm{n}-1}\right| < \delta_{\max} \\ 1 \quad , \ \mathrm{IF}\,\sigma_{\mathrm{n}} = Ss_{n-1} \ \mathrm{and} \left|\delta_{\mathrm{n}-1}\right| = \delta_{\max} \end{cases}, \qquad (2.1.16)$$

тогава δ_{n-1} *се ограничава в интервала*

$$1 < \left|\delta_{n-1}\right| \le \delta_{\max} . \qquad (2.1.17)$$

и се удоволетворява условие (2.1.13)

2.2.5. Ограничаване на намаляването на функцията $\alpha\delta_{n-1}$.

Определение 2.1.2

Компресиране с ADM алгоритъма е възможно само ако

$$\delta_{n-1} \neq 0 \tag{2.1.18}$$

за всички n = 2,..,N.

Следствие 2.1.2

Метод „А”

Ако за адаптивната функция α е изпълнено условието:

$$\alpha = \begin{cases} \alpha_{-}\ ,\ \mathrm{IF}\,\sigma_{\mathrm{n}} \neq Ss_{n-1}\ \mathrm{and}\left|\delta_{\mathrm{n-1}}\right| \neq 1 \\ \text{-}1\ ,\ \mathrm{IF}\,\sigma_{\mathrm{n}} \neq Ss_{n-1}\ \mathrm{and}\left|\delta_{\mathrm{n-1}}\right| = 1 \end{cases}, \tag{2.1.19}$$

тогава за δ_{n-1} е изпълнено неравенството

$$1 \leq \left|\delta_{n-1}\right|. \tag{2.1.20}$$

и се удоволетворява условие (2.1.18).

Метод „B”

Ако за адаптивната функция α *е изпълнено условието:*

$$\alpha = \left\{ -\frac{1}{\left|\delta_{\text{n-1}}\right|},\ \ \text{IF}\,\sigma_{\text{n}} \neq Ss_{n-1} \right\}, \tag{2.1.21}$$

тогава за всички δ_{n-1} *:*

$$\left|\alpha.\delta_{n-1}\right| = 1, \tag{2.1.22}$$

се променя посоката на адаптиране и се удоволетворява условие (2.1.18).

Метод „B” може да се използва за изследване на поведението на ADM алгоритъма при различни начини за образуване на адаптивната функция α.

2.1.6. Модифицирани ADM алгоритми

Определение 2.1.3

Всички ADM алгоритми за които при образуване на адаптивните функции α_r, $r = A, B, ..$ *са наложени допълнителни ограничения в сравнение с (2.1.3), се класифицират като "модифицирани ADM алгоритми – метод "r"(или съкратено MADM(r)).*

Модифициран ADM алгоритъм метод"А" (MADM(A))

От обединяване на условия (2.1.16) и (2.1.19), се получава ново условие за образуване на адаптивната функция α, класифицирана като α_A:

$$\alpha_A = \begin{Bmatrix} \alpha_+ \ , \ \text{IF}\, \sigma_{\text{n}} = Ss_{n-1} \ \text{and} \left|\delta_{\text{n}-1}\right| < \delta_{\max} \\ 1 \quad , \ \text{IF}\, \sigma_{\text{n}} = Ss_{n-1} \ \text{and} \left|\delta_{\text{n}-1}\right| = \delta_{\max} \\ \alpha_- \ , \ \text{IF}\, \sigma_{\text{n}} \neq Ss_{n-1} \ \text{and} \left|\delta_{\text{n}-1}\right| \neq 1 \\ \text{-}1 \ , \ \text{IF}\, \sigma_{\text{n}} \neq Ss_{n-1} \ \text{and} \left|\delta_{\text{n}-1}\right| = 1 \end{Bmatrix}. \qquad (2.1.23)$$

Функцията α_A не променя по същество действието на ADM алгоритъма (2.1.1), а само ограничава δ_{n-1} в интервала.

$$1 \le \left|\delta_{n-1}\right| \le \delta_{\max}. \qquad (2.1.24)$$

Това позволява при изчисляване на S'_n, да не се надхвърля диапазона (2.1.14) за изобразяване на числа представени в допълнителен код и появата на ситуация (2.1.18) при която $\delta_{n-1} = 0$.

Модифициран ADM алгоритъм – метод "B" (MADM(B))

От обединяване на условия (2.1.16) и (2.1.21), се получава ново условие за образуване на адаптивната фукция α, класифицирано като α_B:

$$\alpha_B = \begin{cases} \alpha_+ \ , \ \text{IF} \ \sigma_{\text{n}} = Ss_{n-1} \ \text{and} \left|\delta_{\text{n}-1}\right| < \delta_{\max} \\ 1 \quad , \ \text{IF} \ \sigma_{\text{n}} = Ss_{n-1} \ \text{and} \left|\delta_{\text{n}-1}\right| = \delta_{\max} \\ -\dfrac{1}{\left|\delta_{\text{n-1}}\right|}, \ \ \text{IF} \, \sigma_{\text{n}} \neq Ss_{n-1} \end{cases}. \qquad (2.1.25)$$

Функцията α_B променя стратегията на адаптиране при смяна на посоката, като осигурява симетрично поведение на алгоритъма при нарастване, респективно намаляване на S_n'.

Функциите $\alpha_r \delta_{n-1}$ генерират множеството

$$\alpha_r \delta_{n-1} = \left\{\pm 1, \pm 2, .., \pm 2^{k-1}, \pm \delta_{\max}\right\}, \qquad (2.1.26)$$

с мощност $\left|Ar\right| = 2(k+1)$. (2.1.27)

2.2 Автоматно и таблично представяне на *MADM(r)* алгоритми

В раздел 2.2 са поставени за решаване следните задачи:

- Да се дефинират MADM(r) алгоритмите като крайни автомати които не използват операциите умножение, деление, сравняване по модул.
- Да се изследват възможностите за таблично представяне на *MADM(r)* крайни автомати. Целта на табличното представяне е автоматите да бъдат описани като масиви (едномерни или многомерни) и на тяхна база да се създадат алгоритми или хардуерни решения, за които структурата на кодера (декодера) не зависи от методите за образуване на адаптивната функция α_r.
- Да се изследват възможностите за хардуерно реализиране на таблично представените на *MADM(r)* крайни автомати върху FPGA базирани платформи. Целта на хардуерната реализация е декодирането на речеви сигнали да се извършва от вградени в FPGA контролери, като по този начин се освобождават изчислителни ресурси на микропроцесора и се повишава ефективността на системата.

1. Входна азбука V_n.

Нека заменим стойностите за които Ss_{n-1} (2.1.4) и σ_n (2.1.8) са равни на –1 с 0. В този случай Ss_n и σ_n се означават като $(Ss_{n-1})_{bin}$, респективно $(\sigma_n)_{bin}$. Тогава при

Компресиране

$$V_n = \begin{Bmatrix} 0, \text{ IF } (\sigma_n)_{bin} = (Ss_{n-1})_{bin} \\ 1, \text{ IF } (\sigma_n)_{bin} \neq (Ss_{n-1})_{bin} \end{Bmatrix}. \qquad (2.2.2)$$

Входната азбука може да се определи и с логическата функция “сума по модул 2”,

$$V_n = (\sigma_n)_{bin}\overline{(Ss_{n-1})_{bin}} \vee (\overline{\sigma}_n)_{bin}(Ss_n)_{bin} = (\sigma_n)_{bin} \oplus (Ss_{n-1})_{bin}. \qquad (2.2.3)$$

Декомпресиране

$$V_n = \begin{Bmatrix} 0, \text{ IF } (Ss_n)_{bin} = (Ss_{n-1})_{bin} \\ 1, \text{ IF } (Ss_n)_{bin} \neq (Ss_{n-1})_{bin} \end{Bmatrix}, \textit{ или} \qquad (2.2.4)$$

$$V_n = (Ss_n)_{bin} \oplus (Ss_{n-1})_{bin}. \qquad (2.2.5)$$

2. Изходна азбука W .

Изходната азбука се определя от елементите на множеството, което се генерира от функцията $\alpha_r \delta_{n-1}$ (2.1.26):

$$W = \left\{ \pm \left(2^i \right) \right\}, \quad i = 0,1,..,k \, . \tag{2.2.6}$$

3. Азбука на вътрешните състояния Q .

Ако с $q(+)$ означим състоянията за които е изпълнено условието $\delta_{n-1} > 0$, а с $q(-)$ означим състоянията за които $\delta_{n-1} < 0$ за азбуката на вътрешните състояния се получава:

$$Q = \left\{ q_i \left(\pm \right) \right\}, \quad i = 0,1,..,k \, . \tag{2.2.7}$$

4. Изходна функция λ

След дефинирането на $V, W \; u \; Q$ може да се определи изходната функция λ :

$$\lambda(q_i(\pm)) = \{ \pm 2^i \}, \; i = 0,1,..,k \, . \tag{2.2.8}$$

5. Начално състояние q_0.

Компресиране

Началното състояние на автомата q_0 при компресиране се определя от знака на грешката (2.1.11) на речевия сигнал $(\sigma_n)_{bin}$ и *n* = 1:

$$q_0 = \begin{Bmatrix} q_0(+), \text{ IF } \sigma_{(1)} = 1 \\ q_0(-), \text{ IF } \sigma_{(1)} = 0 \end{Bmatrix}. \qquad (2.2.9)$$

Декомпресиране

Началното състояние на автомата q_0 при декомпресиране се определя от знака на относителната промяна (2.1.4) на речевия сигнал $(Ss_n)_{bin}$ и *n* = 0:

$$q_0 = \begin{Bmatrix} q_0(+), \text{ IF } Ss_{(0)} = 1 \\ q_0(-), \text{ IF } Ss_{(0)} = 0 \end{Bmatrix} \qquad (2.2.10)$$

6. Функция на преходите d.

Функцията на преходите $d_{(r)}$, r = *A,B,..* може да се определи от условията за образуване на адаптивната функция α_r (2.1.23, 2.1.25):

$$d_A = \left\{ \begin{array}{l} q_i(\pm) \to q_{i+1}(\pm), \text{ IF } V = 0 \text{ and } 0 \le i < k \\ q_k(\pm) \to q_k(\pm) \text{ IF } V = 0 \text{ and } i = k \\ q_i(\pm) \to q_{i-1}(\mp) \text{ IF } V = 1 \text{ and } 1 \le i \le k \\ q_0(\pm) \to q_0(\mp) \text{ IF } V = 1 \text{ and } i = 0 \end{array} \right\}, \qquad (2.2.11)$$

$$d_B = \left\{ \begin{array}{l} q_i(\pm) \to q_{i+1}(\pm), \text{ IF } V = 0 \text{ and } 0 \le i < k \\ q_k(\pm) \to q_k(\pm) \text{ IF } V = 0 \text{ and } i = k \\ q_i(\pm) \to q_0(\mp) \text{ IF } V = 1 \text{ and } 0 \le i \le k \end{array} \right\}. \qquad (2.2.12)$$

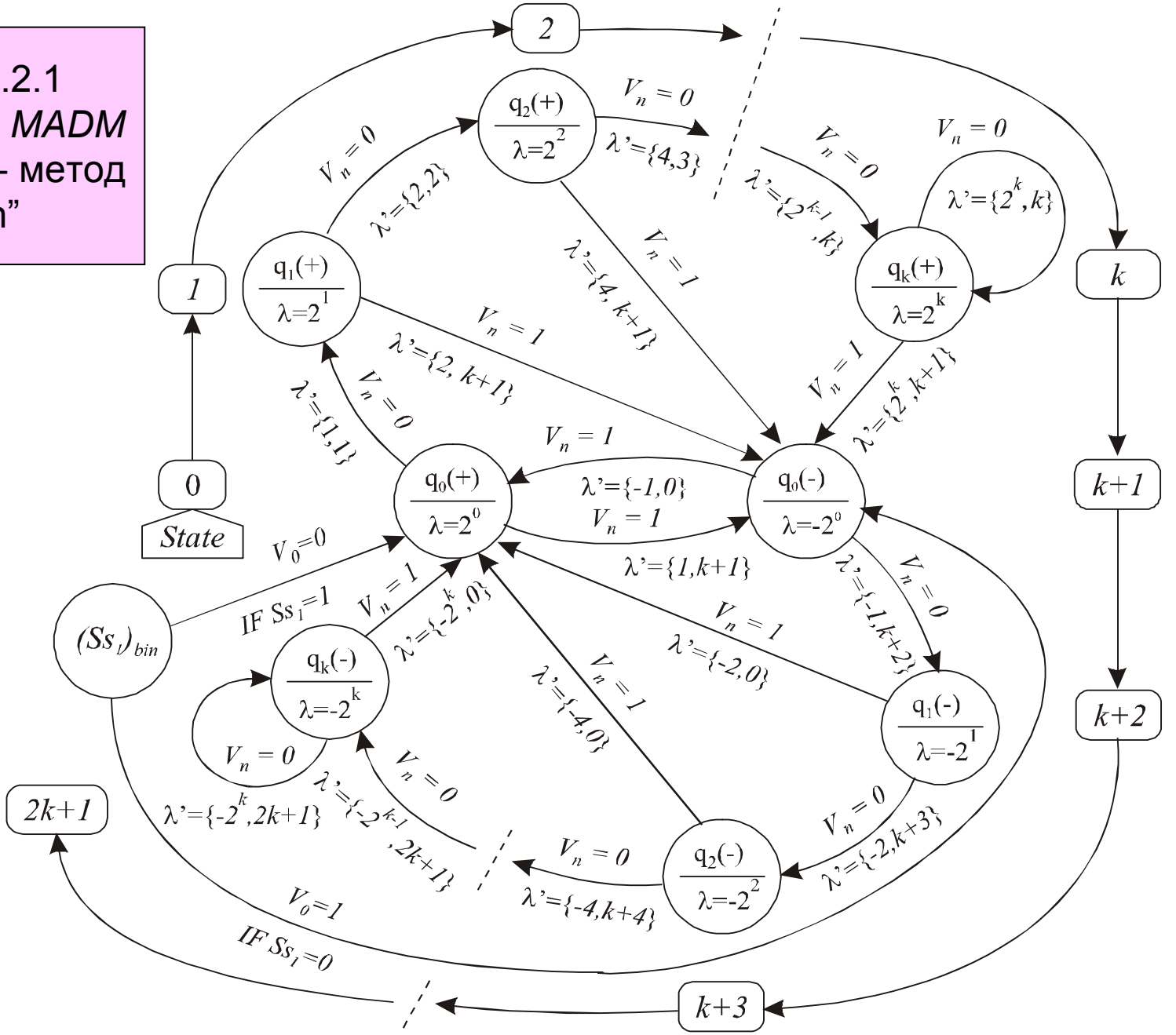
Графите с който се описват автоматите на Мур, реализиращи *MADM(r)* алгоритмите, могат да бъдат построени с използването на следните правила (фиг 2.2.1):

1. *При зададено k и за даден метод r, се построяват 2к+2 състояния на автомата (2.2.7)* $q_i(\pm)$ *,* $i = 0,1,.. k.$

2. *На всеки възел, се присвоява съответната стойност на изходната функция* λ_i *,* $0 \le i \le k$ *(2.2.8).*

3. *В съответствие с функцията на преходите, възлите на автомата се свързват с ребра, като на всяко ребро се присвоява съответната стойност на входната азбука* V_n *при която се извършва прехода* d_A *(2.2.11) или* d_B *(2.2.12).*

Фиг 2.2.1
Граф на *MADM* автомат - метод "Bh"



2.2.2 Таблично представяне на *MADM(r)* алгоритми

Дефинирането на *MADM(r)* алгоритмите като автомати на Мур може да бъде използвано за тяхното таблично представяне.

Различните конфигурации на *MADM(r)* автомати на Мур, могат да бъдат представени таблично с използването на следните правила:

1. *При зададено* ***k*** *и за даден метод* ***r*** *да построим MADM автомат на Мур .*
2. *Да преномерираме вътрешните състояния на автомата* $q_i(\pm),\ i = 0,1,..k$ *по неговия външен контур. Избора на първото състояние и посоката на преномериране са без значение. Да приемем, че за първо състояние е избран възел* $q_0(+)$*, а посоката е по часовниковата стрелка.*

3. *В зависимост от реализацията на таблично представения автомат (софтуерно или хардуерно) са възможни два метода за преномериране на новите вътрешните състояния:*

3.1. *Софтуерна реализация. Елементите на таблично представения автомат при софтуерна реализация се разполагат в паметта на микроконтролера. Следователно, при преномериране трябва да се осигури достатъчно разстояние в паметта за разполагане на елементите на дадено състояние. Нека дадено състояние на автомата се представя с два байта. Да означим новите вътрешни състояния на графа със* $State_j$, $j = 0,2,..,4k+2$, *а на първото състояние да присвоим номер 0.*

3.2. *Хардуерна реализация. Да означим новите вътрешни състояния на графа със* $State_j$, $j = 0,1,..,2k+1$, *а на първото състояние да присвоим номер 0.*

4. *Да съпоставим на всяко ребро на графа по една двойка числа* $\lambda' = \{\lambda_i, State_j\}$ *, като първото число се образува от изходната функция* λ_i *на възела* i *от което реброто излиза, а второто число се образува от състоянието* $State_j$ *в което реброто влиза.*

5. *Да съставим две таблици* TBL_V0, TBL_V1 *като:*

- *TBL_V0 съдържа всички двойки за които при дадено State,* $V_n = 0$.
- *TBL_V1 съдържа всички двойки за които при дадено State,* $V_n = 1$.

Таблиците TBL_V0 и TBL_V1 представят състоянията на *MADM(r)* автоматите като масиви от данни, образувани от двойките на новата изходна функция $\lambda' = \{\lambda_i, State_j\}$.

Таблиците предназначени за софтуерна реализация на даден метод r, $r = A, B, ..$ се обозначават като "$As, Bs, ..$", а тези предназначени за хардуерна реализация като $Ah, Bh, ..$

q_i	$q_0(+)$	$q_1(+)$	….	$q_k(+)$	$q_k(-)$	….	$q_1(-)$	$q_0(-)$
State	*0,1*	*2,3*	….	*2k,2k+1*	*2k+2,2k+3*	….	*4k,4k+1*	*4k+2,4k+3*
TBL_V0	*1,2*	*2,4*	….	2^k*,2k*	-2^k*,2k+2*	….	*-2,4k-2*	*-1,4k*
TBL_V1	*1,4k+2*	*2,4k+2*	….	2^k*,2k+4*	-2^k*,2k-2*	….	*-2,0*	*-1,0*

Таблица 2.2.1. Таблично представен *MADM* краен автомат – метод "*As*".

q_i	$q_0(+)$	$q_1(+)$	….	$q_k(+)$	$q_k(-)$	….	$q_1(-)$	$q_0(-)$
State	0	1	….	k	k+1	….	2k	2k+1
TBL_V0	1,1	2,2	….	2^k,k	-2^k,k+1	….	-2,2k-1	-1,2k
TBL_V1	1,2k+1	2,2k+1	….	2^k,k+2	-2^k,k-1	….	-2,0	-1,0

Таблица 2.2.2. Таблично представен *MADM* краен автомат – метод "*Ah*".

q_i	$q_0(+)$	$q_1(+)$	….	$q_k(+)$	$q_0(-)$	$q_1(-)$	….	$q_k(-)$
State	*0,1*	*2,3*	….	*2k,2k+1*	*2k+2,2k+3*	*2k+4,2k+5*	….	*4k+2,4k+3*
TBL_V0	*1,2*	*2,4*	….	2^k*,2k*	*-1,2k+4*	*-2,2k+6*	….	-2^k*,4k+2*
TBL_V1	*1,2k+2*	*2,2k+2*	….	2^k*,2k+2*	*-1,0*	*-2,0*	….	-2^k*,0*

Таблица 2.2.3. Таблично представен *MADM* краен автомат – метод "*Bs*".

q_i	$q_0(+)$	$q_1(+)$	….	$q_k(+)$	$q_0(-)$	$q_1(-)$	….	$q_k(-)$
State	*0*	*1*	….	*k*	*k+1*	*k+2*	….	*2k+1*
TBL_V0	*1,1*	*2,2*	….	2^k*,k*	*-1,k+2*	*-2,k+3*	….	-2^k*,2k+1*
TBL_V1	*1,k+1*	*2,k+1*	….	2^k*,k+1*	*-1,0*	*-2,0*	….	-2^k*,0*

Таблица 2.2.4. Таблично представен *MADM* краен автомат – метод "*Bh*".

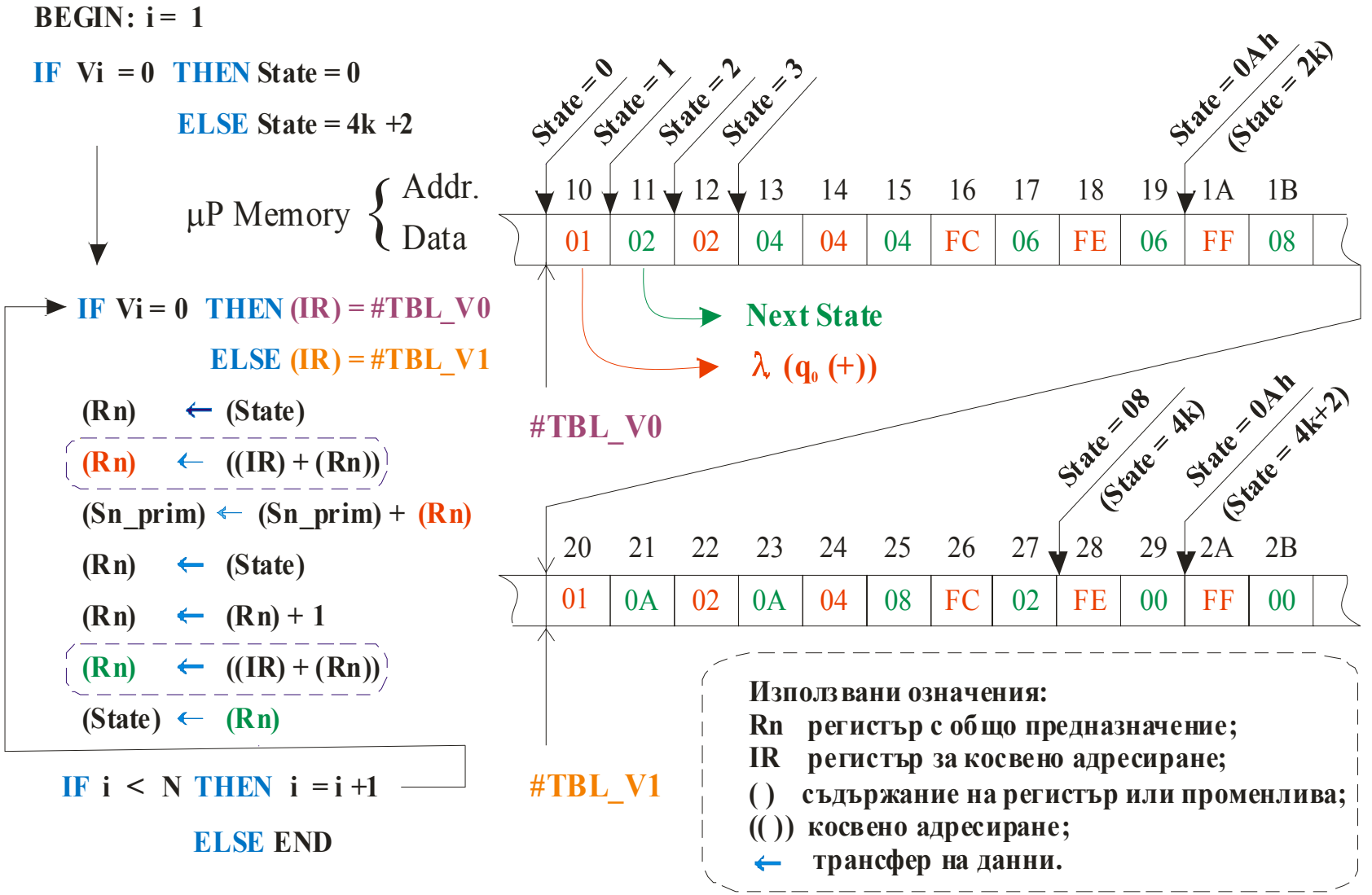
Софтуерно декомпресиране с таблично зададени $MADM(r)$, $(r = As, Bs,..)$ автомати

При софтуерно декомпресиране елементите на таблиците *TBL_V0* и *TBL_V1* се разполагат последователно в паметта за данни на микроконтролера, започвайки от произволно избрани начални адреси *#TBL_V0* и *#TBL_V1* (фиг 2.2.2)

В зависимост от текущата стоиност на входната азбука Vi се определя в кой от двата масива се намира изходната функция и следващото състояние на автомата.

За осъществаване на достъп до елементите на масивите е достатъчно микроконтролера да притежава една инструкция за косвено адресирана с индекс.

По този начин, софтуера за декомпресиране не зависи от вида на автомата $(r = As, Bs,..)$, което позволява експериментиране с различни методи за образуване на адаптивната функция.



Фиг. 2.2.2. Разполагане в паметта на микроконтролера на таблично представен *MADM* автомат – метод “As”, *k* = 2. Източник - таблица 2.2.1. Примерна програма за декомпресиране.

Хардуерно декомпресиране с таблично зададени *MADM(r)* , *(r = Ah, Bh,..)* автомати

При хардуерно декомпресиране елементите на таблиците *TBL_V0* и *TBL_V1*, могат да се кодират с използването на комбинационна логическа схема (таблица 2.2.7):

- A_addr_n - входна функция. Тази функция се образува от конкатенацията на *State* и входната азбука *Vn*.

- *Aout_n* - изходна функция. При зададено State и Vn, тази функция се образува от първите елементи на таблиците *TBL_V0* и *TBL_V1*.

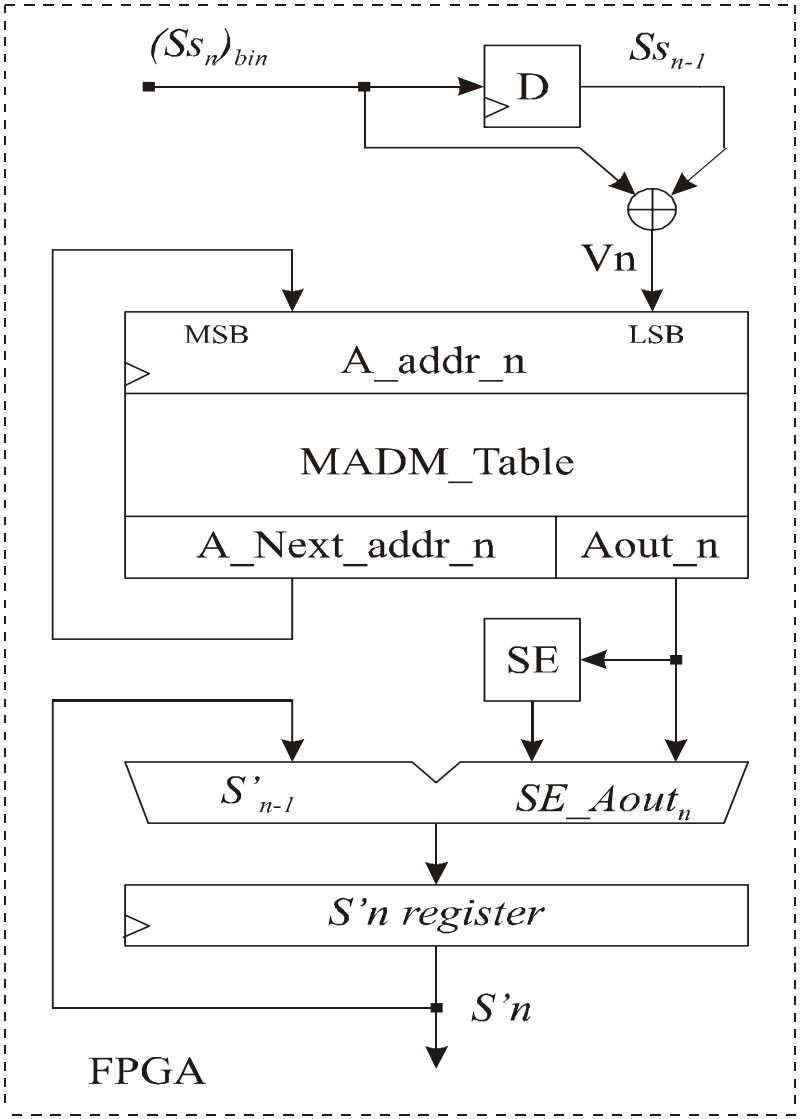
- A_next_addr_n - изходна функция. При зададено State и Vn, тази функция се образува от вторите елементи на таблиците *TBL_V0* и *TBL_V1*.

Една примерна схема на FPGA базиран, хардуерен декодер, описан на VHDL е показана в следващата фигура.

A_addr_n				
State	Vn	Aout_n	A_next_addr_n	Източник: Таблица 2.2.3.
000	0	0001	001	$q_0(+)$: TBL_V0 = 1,1
000	1	0001	101	$q_0(+)$: TBL_V1 = 1,2k+1
001	0	0010	010	$q_1(+)$: TBL_V0 = 2,2
001	1	0010	101	$q_1(+)$: TBL_V1 = 2,2k+1
010	0	0100	010	$q_2(+)$: TBL_V0 = 2^k,k
010	1	0100	100	$q_2(+)$: TBL_V1 = 2^k,k+2
011	0	1100	011	$q_2(-)$: TBL_V0 = -2^k,k+1
011	1	1100	001	$q_2(-)$: TBL_V1 = -2^k,k-1
100	0	1110	011	$q_1(-)$: TBL_V0 = -2,2k-1
100	1	1110	000	$q_1(-)$: TBL_V1 = -2,0
101	0	1111	100	$q_0(-)$: TBL_V0 = -1,2k
101	1	1111	000	$q_0(-)$: TBL_V1 = -1,0

Таблица 2.2.7. Таблично задаване на състоянията на MADM краен автомат като комбинационна логическа схема. Метод "Ah"; $k = 2$.

FPGA базиран $MADM_{(r)}$ декодер




```
--    VHDL program
--    MADM Automat - method "Ah"; k=2
--    Component MADM_TABLE
ARCHITECTURE rtl OF MADM_TABLE IS
BEGIN
 PROCESS(A_addr_n) BEGIN
  CASE A_addr_n IS
   WHEN "0000" => Aout_n <= "0001";
                  A_Next_addr_n <= "001";
    WHEN "0001" => Aout_n <= "0001";
                   A_Next_addr_n <= "101";
    ---------------------------------
    WHEN "1011" => Aout_n <= "1111";
                   A_Next_addr_n <= "000";
   END CASE;
  END PROCESS;
END rtl;
```

2. 3. Коригиране на грешката при декомпресиране с $MADM_{(r)}$ алгоритми

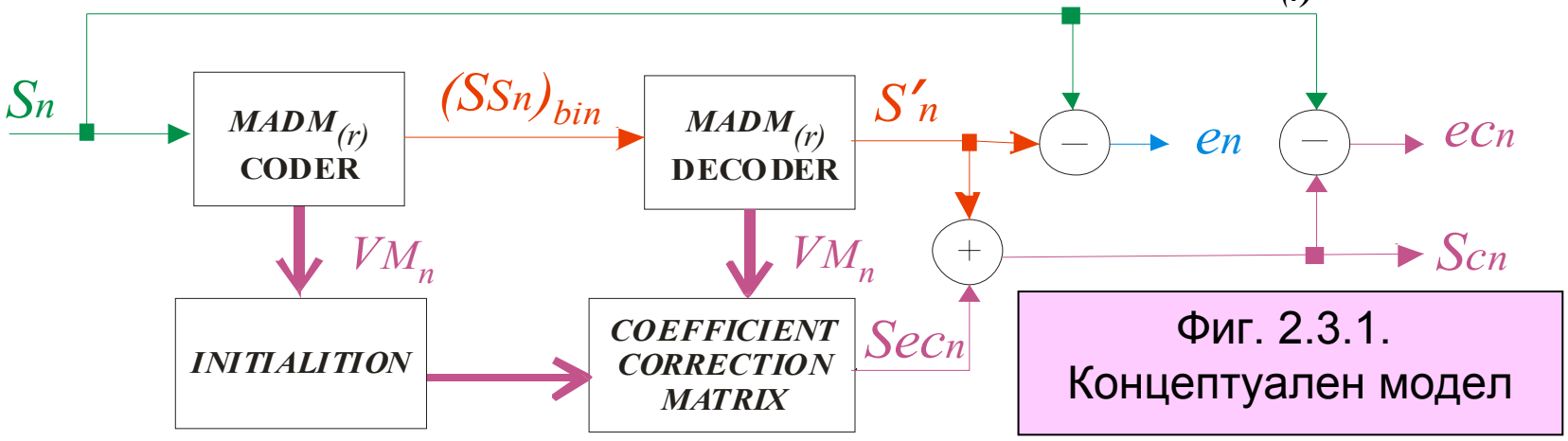
Съществен недостатък на ADM (*MADM(r)*) алгоритмите е относително голямата грешка с която се възстановява оригиналния речев сигнал при декомпресиране.

Разработването и изследването на алгоритми с които да се коригира грешката при декомпресиране е цел на настоящето изследване.

За изпълнение на поставената цел, трябва да се решат следните задачи:

- Да се дефинира концептуален модел на алгоритъма за коригиране на грешката при декомпресиране.
- Да се определи критерий за оценка на ефективността от извършената корекция.
- Да се разработят алгоритми за коригиране на грешката при декомпресиране.
- Да се предложи схема за реализиране на *MADM(r)* декодер с коригиране на грешката при декомпресиране с използването на FPGA базирани платформи.
- Да се докажат експериментално резултатите от извършената корекция.

2. 3. Коригиране на грешката при декомпресиране с $MADM_{(r)}$ алгоритми



Фиг. 2.3.1.
Концептуален модел

При компресиране с операцията **C** (2.1.6), реализираща алгоритъма (2.1.1) се получава грешка e_n (2.1.7).

Идеята за коригиране на грешката при декомпресиране (фиг.2.3.1), се базира на предположението, че за всяко S'_n на операцията **D** (2.1.9) съществува коефициент за корекция Sec_n.

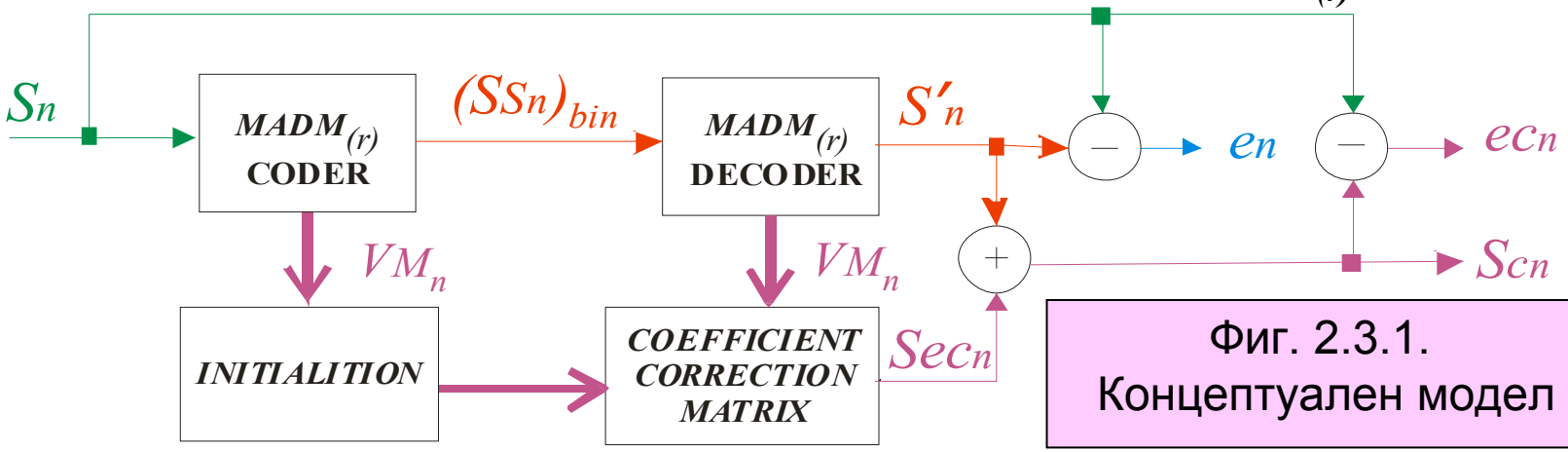
След сумиране на S'_n и Sec_n:

$$Sc_n = S'_n + Sec_n, \qquad (2.3.1)$$

се получава нова стойност на декомпресирания сигнал Sc_n и грешка

$$ec_n = S_n - Sc_n. \qquad (2.3.2)$$

2. 3. Коригиране на грешката при декомпресиране с $MADM_{(r)}$ алгоритми

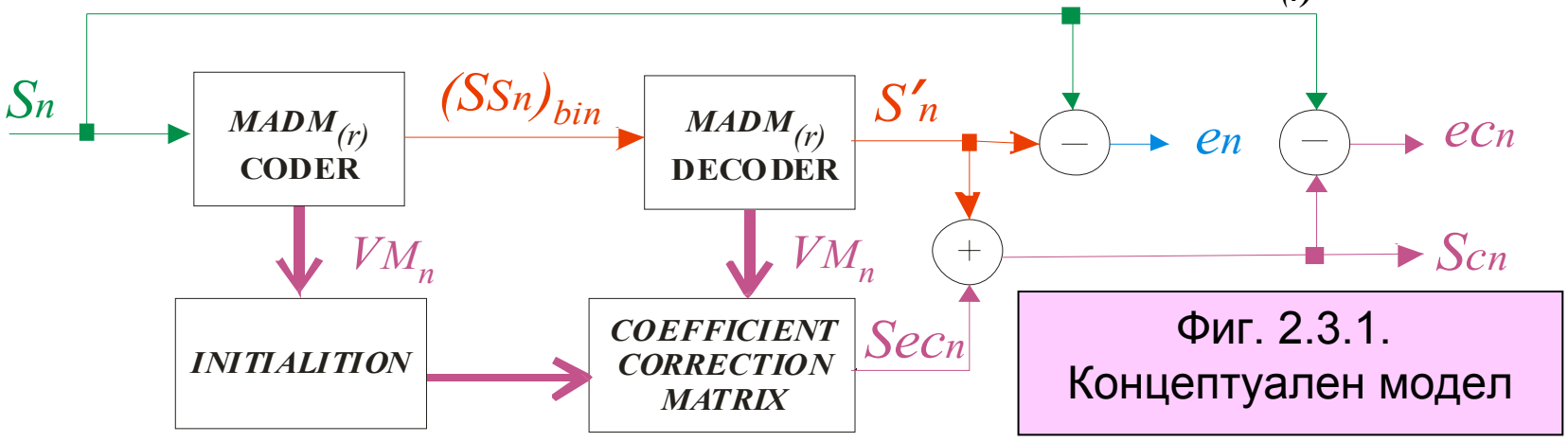


Фиг. 2.3.1.
Концептуален модел

Коефициентите на корекция Sec_n могат да се определят така, че да минимизират грешката ec_n. Така формулирана задачата може да се разшири, като се търси решение за определянето на Sec_n, което след корекция да минимизира средноквадратичната грешка $\overline{ec_n}^2$ за всички $n,\ n=1,..,N$:

$$\overline{ec_n}^2 = \frac{1}{N}\sum_{n=1}^{N} ec_n^2 \ . \qquad (2.3.3)$$

2. 3. Коригиране на грешката при декомпресиране с $MADM_{(r)}$ алгоритми



Фиг. 2.3.1. Концептуален модел

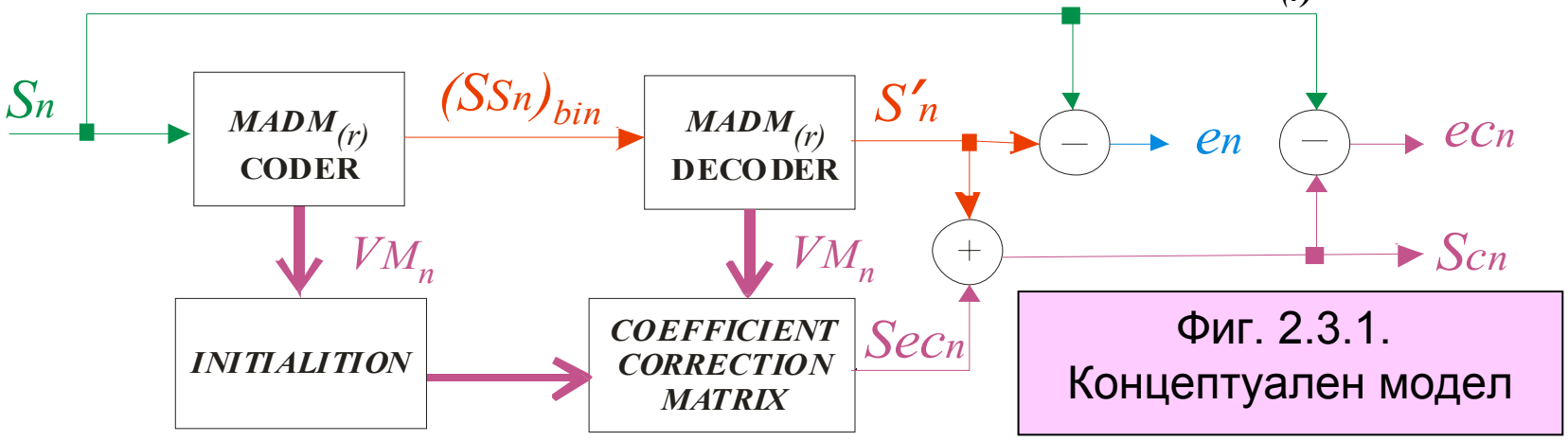
Нека с $\bar{e}_n^2$ означим средноквадратичната грешка получена след компресиране(2.1.6):

$$\bar{e}_n^2 = \frac{1}{N}\sum_{n=1}^{N} e_n^2 , \qquad (2.3.4)$$

а с Ec означим относителната средноквадратична грешка:

$$Ec = \frac{\bar{ec}_{cn}^2}{\bar{e}_n^2}. \qquad (2.3.5)$$

2. 3. Коригиране на грешката при декомпресиране с $MADM_{(r)}$ алгоритми

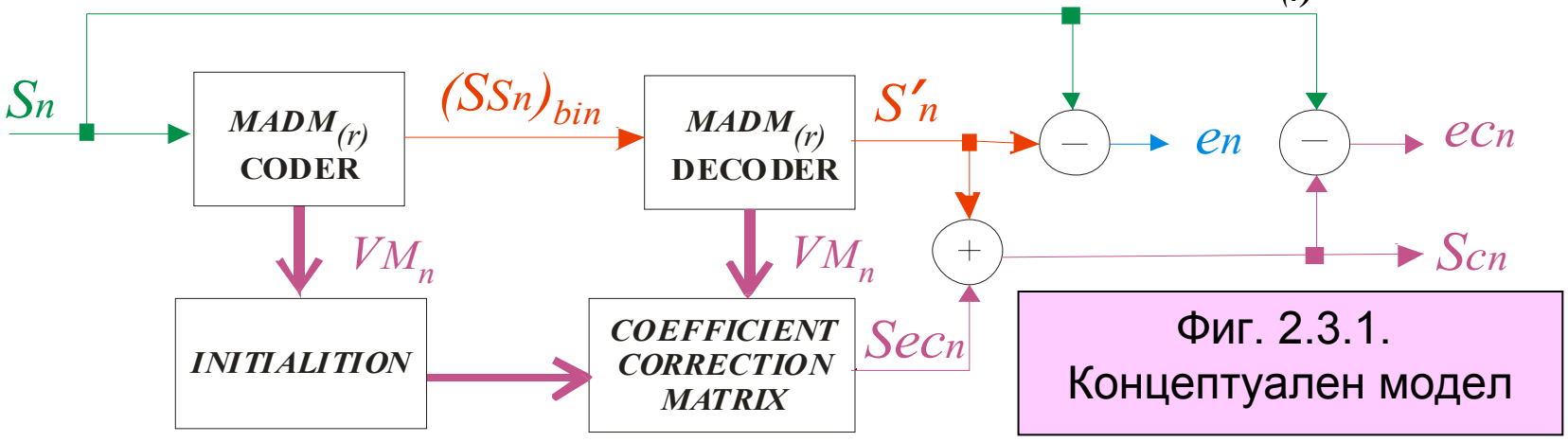


Фиг. 2.3.1.
Концептуален модел

Тогава относителната средноквадратична грешка Ec може да се използва като показател за ефективността на извършената корекция. При компресиране на статични речеви образци, с каквито се характеризират системите за гласови съобщения, средноквадратичната грешка $\overline{e}_n^2$ в отношението (2.3.5), за даден метод *r* е константа.

Следователно, минимизирането на средноквадратичната грешка $\overline{ec}_n^2$ е еквивалентно на минимизирането на относителната средноквадратична грешка Ec.

2. 3. Коригиране на грешката при декомпресиране с $MADM_{(r)}$ алгоритми



Фиг. 2.3.1.
Концептуален модел

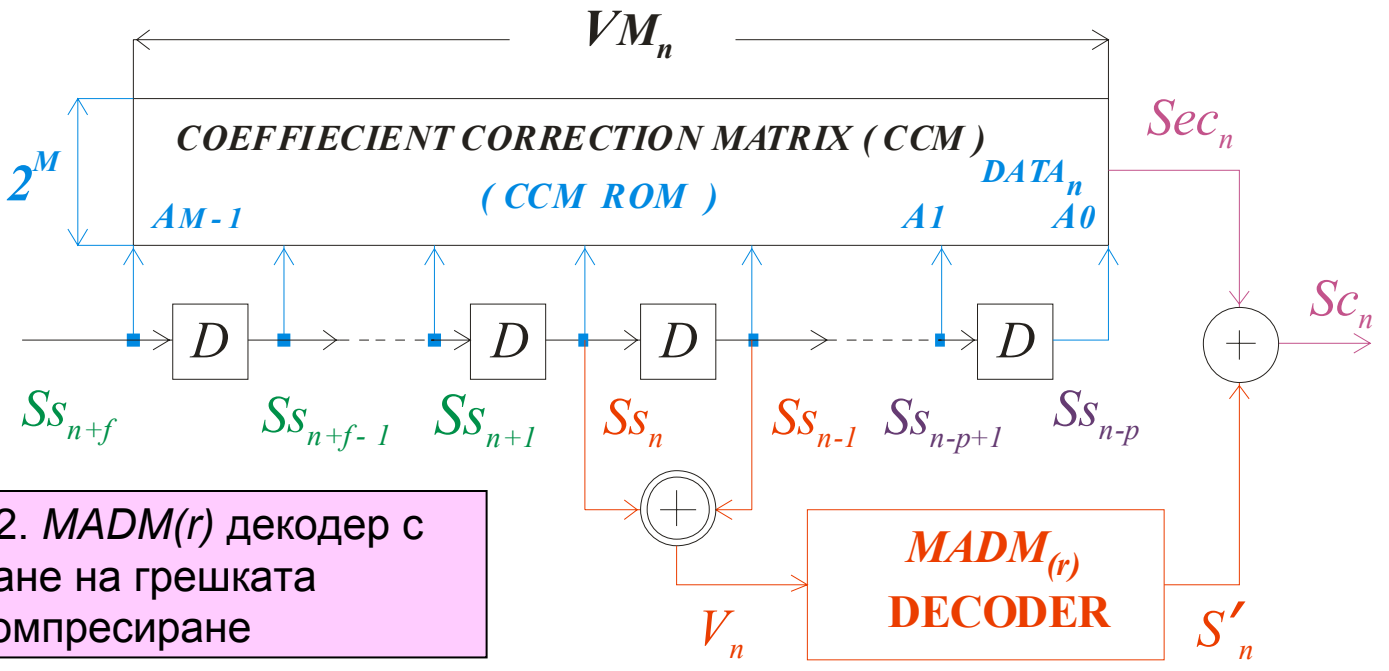
Ако приемем, че за всяко ec_n е изпълнено неравенството:

$$|ec_n| \le |e_n|, \qquad (2.3.6)$$

относителната средноквадратична грешка Ec е ограничена в интервала

$$0 \le Ec \le 1 \qquad (2.3.7)$$

и се стреми към лявата си граница с увеличаване на броя на успешно коригираните стойности на S'_n.



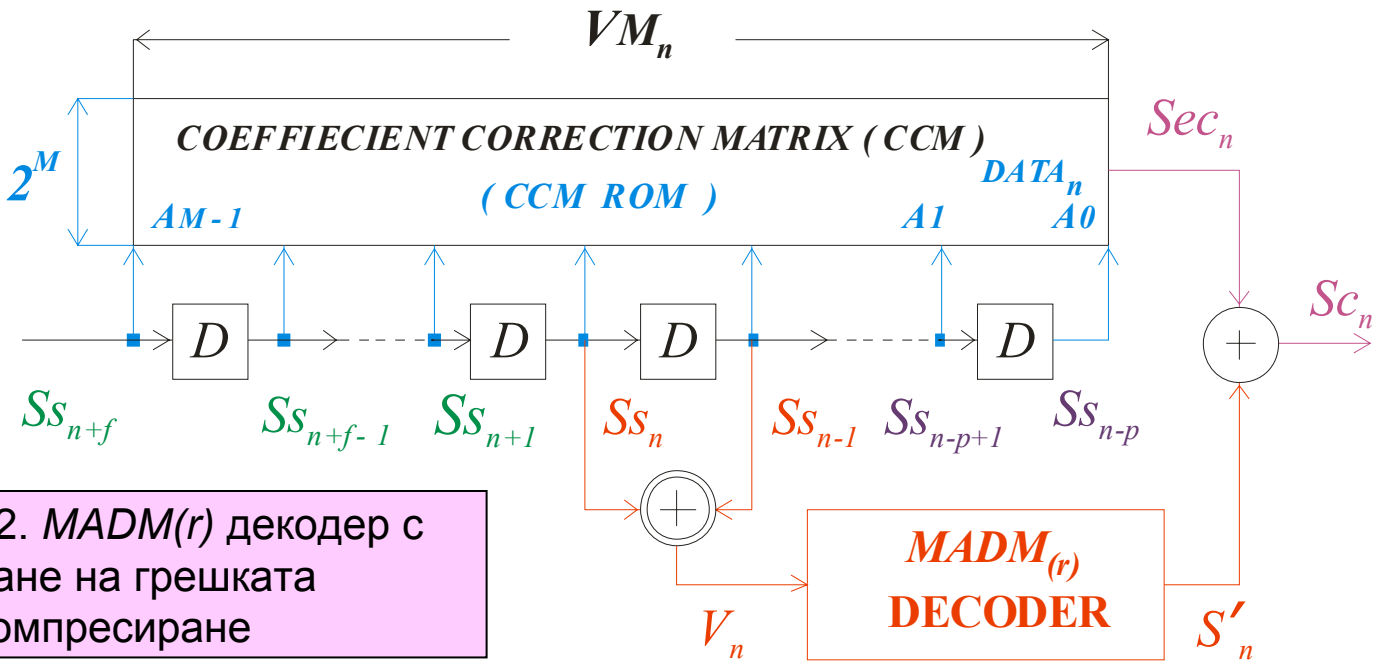
Фиг. 2.3.2. *MADM(r)* декодер с коригиране на грешката при декомпресиране

Коефициентите за коригиране на грешката могат да се определят при компресиране, като функция на двоичния вектор-предсказател VM_n състоящ се от M елемента на редицата $\{Ss_n\}$ (фиг. 2.3.2).

Текущата стойност на вектора VM_n се формира от *p* предходни и *f* следващи стойности на Ss_n. Тогава за VM_n е валидна следната зависимост:

$$VM_n = \sum_{i=0}^{M-1} Ss_{n-p+i}.2^i, \quad VM_n \in \left\{0, 2^M - 1\right\}, \qquad (2.3.8)$$

където: $M = p + f + 1$. (2.3.9)



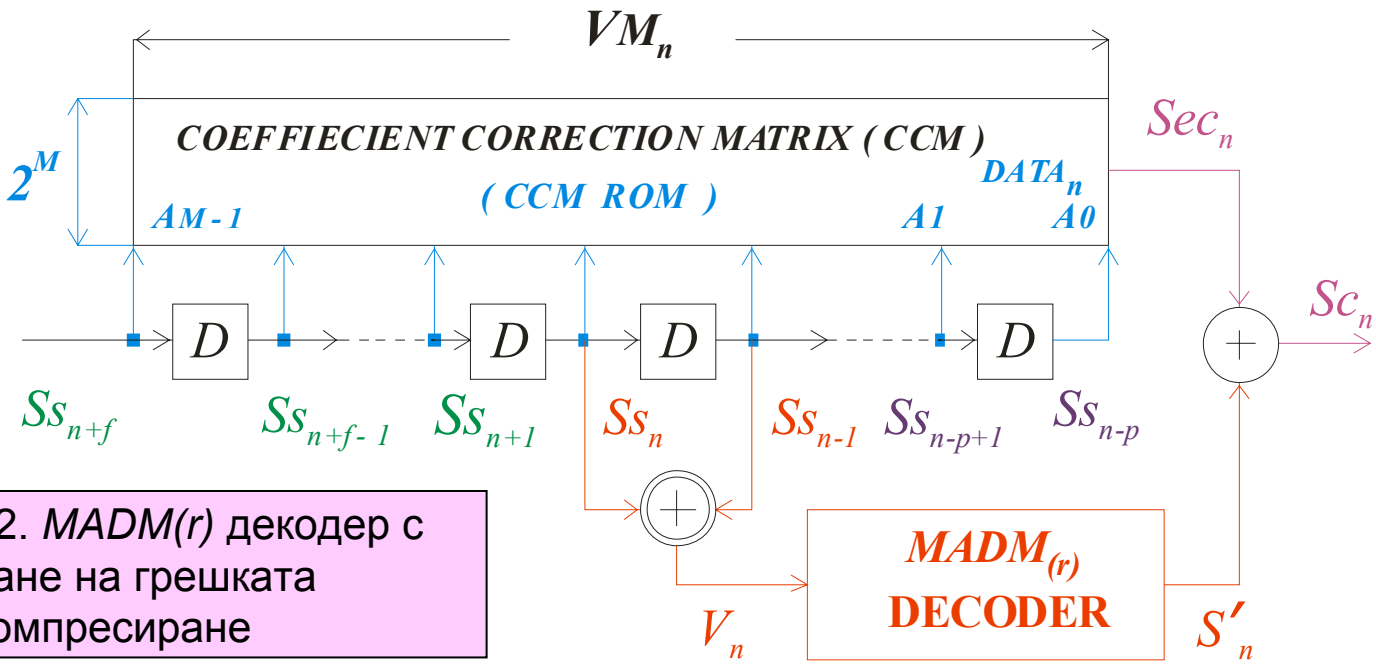
Фиг. 2.3.2. *MADM(r)* декодер с коригиране на грешката при декомпресиране

Коефициентите за коригиране на грешката, съответстващи на всички възможни комбинации на вектора VM_n образуват матрица на коефициентите за коригиране на грешката $\left\| \left[\bar{e}_{VM_n} \right] \right\|$ с размер 2^M.

При зададено M матрицата на коефициентите за коригиране на грешката се инициализира с цялата част от усреднената стойност на грешката

$$\left\| \left[\bar{e}_{VM_n} \right] \right\| = \frac{1}{QVM} \sum_{n=1}^{QVM} e_n , \qquad (2.3.10)$$

където: QVM е броя на еднаквите стойности на вектора VM_n в редицата $\{Ss_n\}$.



Фиг. 2.3.2. *MADM(r)* декодер с коригиране на грешката при декомпресиране

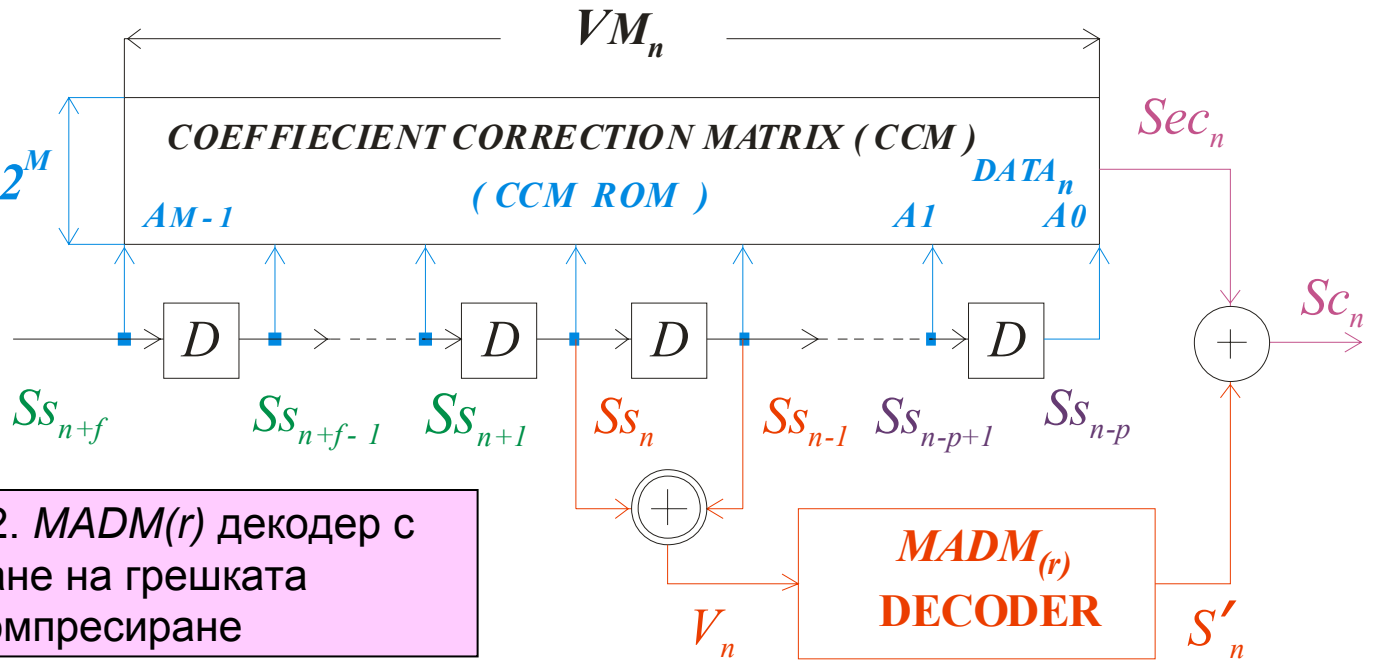
След инициализиране на матрицата на коефициентите за коригиране на грешката може да се извърши корекция при декомпресиране с коефициента за корекция Sec_n (2.3.1).

$$Sec_n = \left\| \left[\bar{e}_{VM_n} \right] \right\|. \qquad (2.3.11)$$

Минималната стойност на относителната средноквадратична грешка Ec_{min},

$$Ec_{min} = min\{ Ec \mid r, k, M, f \}, \qquad (2.3.12)$$

за даден метод за образуване на адаптивната функция *r*, *r* = *A*,*B*,.. може да се определи с промяната на k, M, f.



Фиг. 2.3.2. *MADM(r)* декодер с коригиране на грешката при декомпресиране

MADM(r) декодерът с коригиране на грешката при декомпресиране е подходящ за реализиране с FPGA базирани платформи. Текущата стойност на вектора V_{M_n} може да се формира с използването на един преместващ регистър. За съхраняване на елементите на матрица на коефициентите за коригиране на грешката, може да се използва някоя от вградените в FPGA памети или външен за системата за гласови съобщения ROM.

2.3.4. Резултати от изследванията

Минималната стойност на относителната средноквадратична грешка Ec_{min} ,

$$Ec_{min} = min\left\{ Ec \mid r, k, M, f \right\}, \qquad (2.3.12)$$

за даден метод за образуване на адаптивната функция *r, r = A,B,..* може да се определи с промяната на k, M , f .

- коефициента *k* (2.1.18) *,* ограничаващ нарастването на δ_n (2.1.2):

$$k_L \le k \le k_H, \qquad (2.3.13)$$

където k_L , k_H са коефициенти които ограничават промяната на *k.*

- броя на елементите *M* (2.3.9) от които се формира вектора V_{M_n} (2.3.8):

$$M_L \le M \le M_H, \qquad (2.3.14)$$

където M_L , M_H са коефициенти които ограничават промяната на *M.*

- отношението на *p*-те предходните и *f*-те следващи стойности на Ss_n от които се формира текущата стойност на вектора V_{M_n} :

$$f = 0..M\text{-}1; \qquad (2.3.15)$$

$$p = M - 1 - f. \qquad (2.3.16)$$

Определянето на минималната стойност на относителната средноквадратична грешка Ec_{min} се извършва в следващата последователност:

2.3.4. Резултати от изследванията

*1. Създава се тестов файл от речеви образци с която ще се реализира системата за гласови отговори. При даден метод за образуване на адаптивната функция r се задават коефициенти които ограничават промяната на **k** (2.3.13) и M (2.3.14).*

2. При зададен метод за образуване на адаптивната функция r и за дадено k се извършва компресиране на тестовия файл и се определя средноквадратичната грешка $\bar{e}_n^2$ (2.3.4).

*3. С промяната на **f** (2.3.15) се конструират M конфигурации на вектора V_{M_n}.*

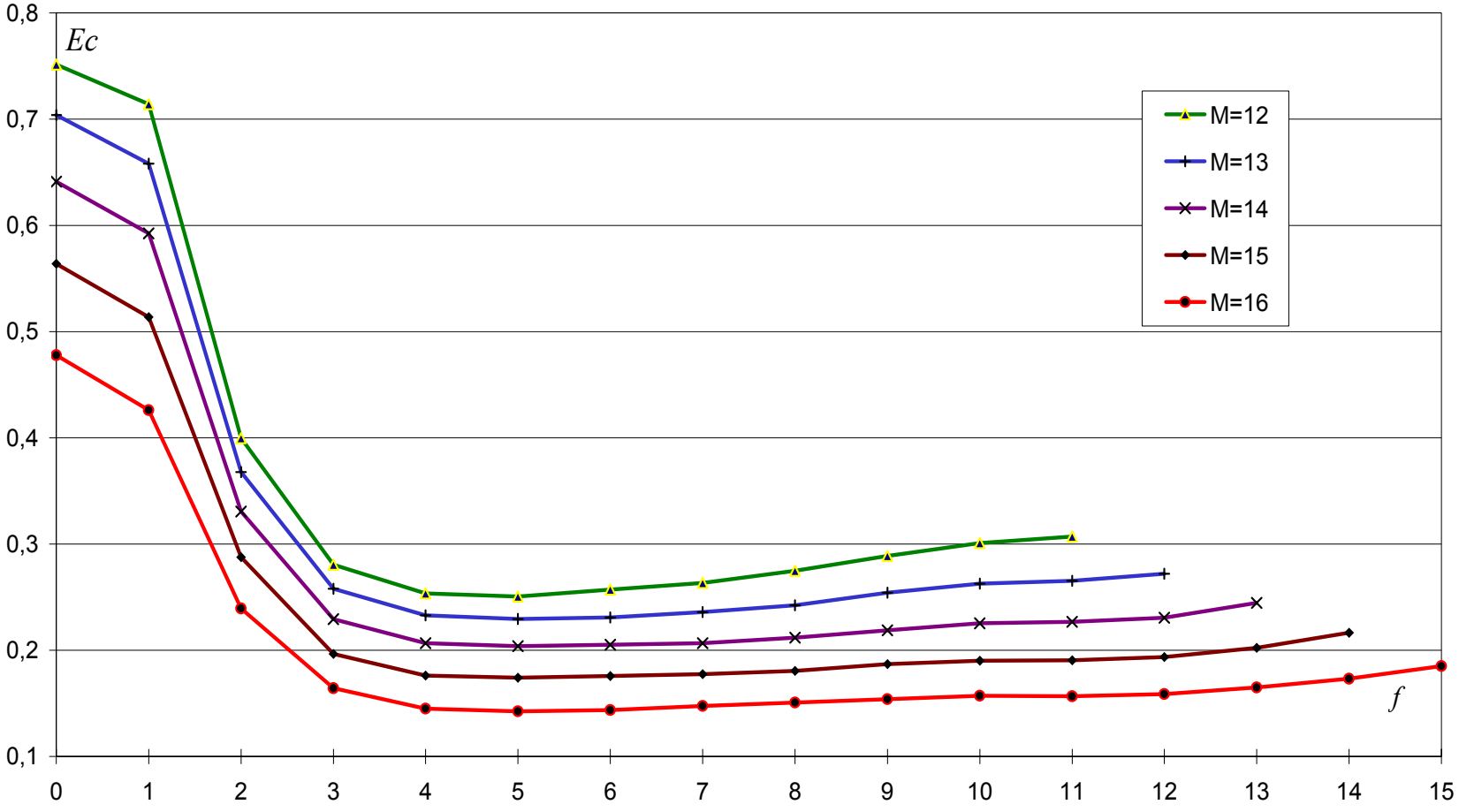
4. С всяка отделна конфигурация на вектора V_{M_n} се инициализира матрицата на коефициентите за коригиране на грешката (2.3.10).

5. С всяка отделна матрица се извършва декомпресиране с коригиране на грешката и се определя средноквадратичната грешка $\overline{ec}_n^2$ (2.3.3) и относителната средноквадратична грешка Ec (2.3.12).

6. Действия 2 до 5 се повтарят до определяне на относителната средноквадратична грешка Ec за всички методи за образуване на адаптивната функция r и за всички k.

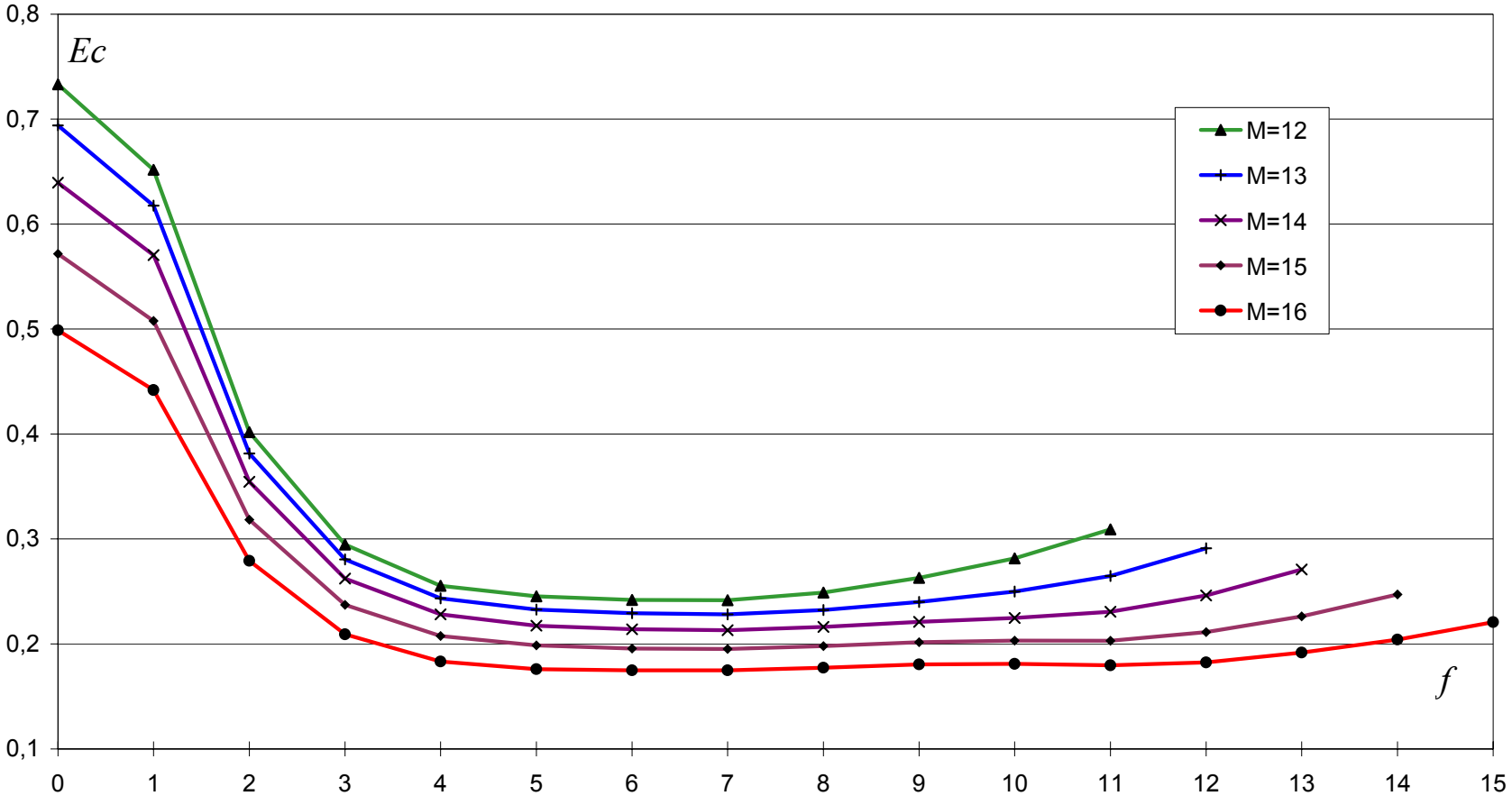
7. За така определените Ec се търси минимум на относителната средноквадратична грешка Ec_{min}.

2.3.4. Резултати от изследванията



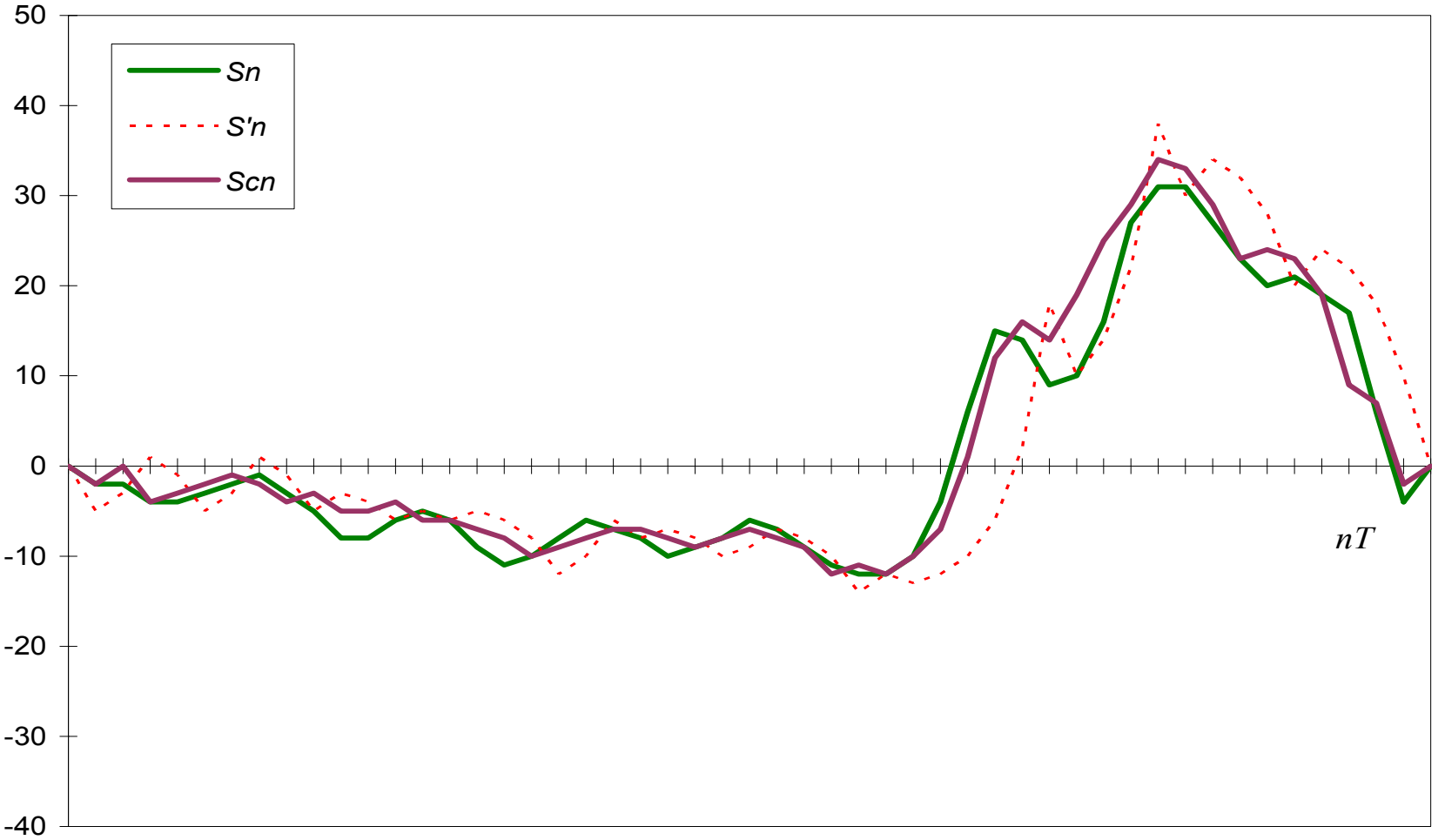
Фиг. 2.3.3 Зависимост на относителната средно-квадратична грешка Ec от конфигурацията на вектора V_{M_n} - мъжки говор: $r = A; k = 3$

2.3.4. Резултати от изследванията



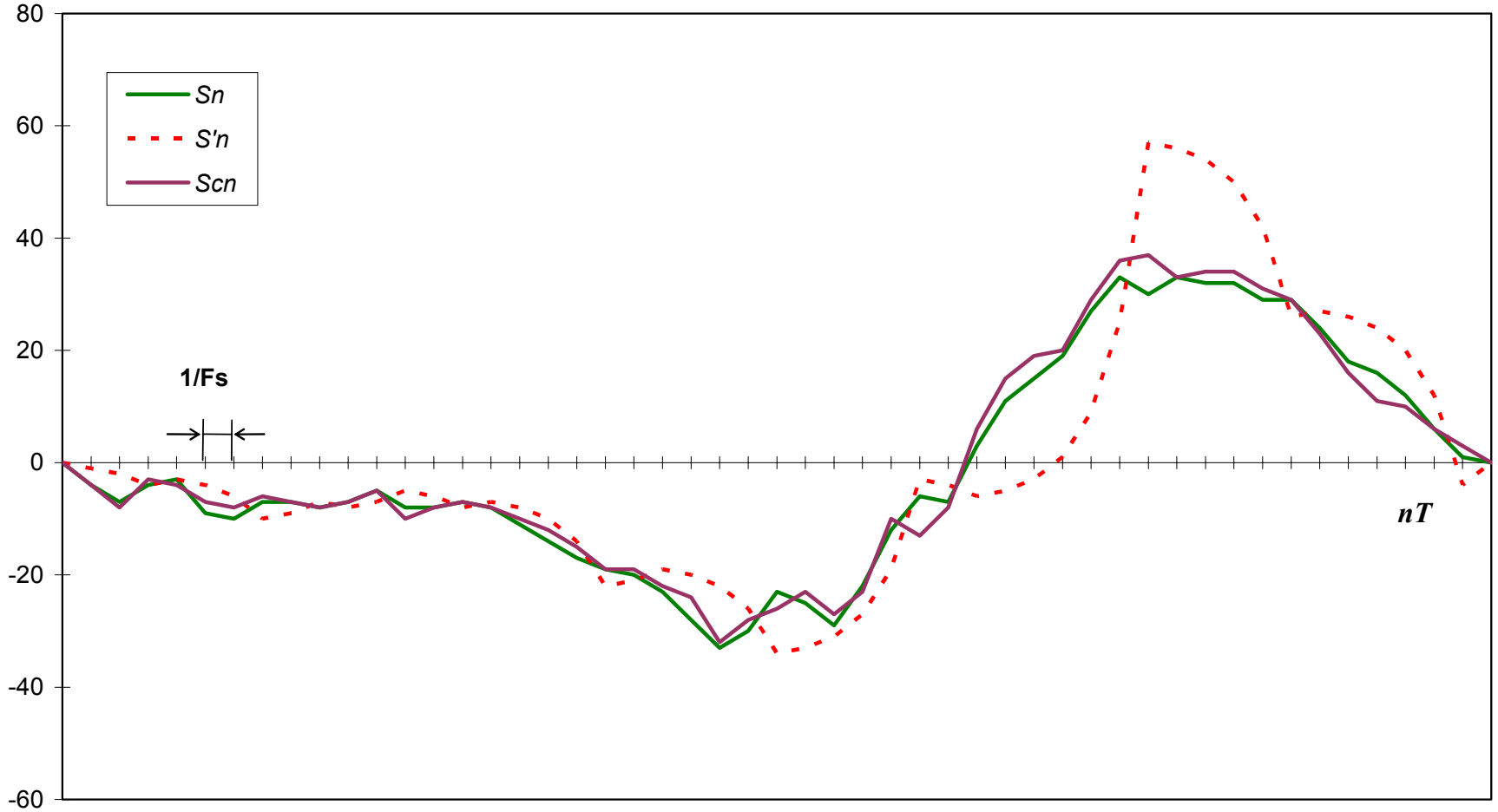
Фиг. 2.3.4. Зависимост на относителната средно-квадратична грешка Ec от конфигурацията на вектора V_{M_n} - женски говор: $r=B; k=4$

2.3.4. Резултати от изследванията



Фиг. 2.3.5. Коригиране на грешката при декомпресиране – метод *“A”*:
Fs = 11025 Nz ; m = 8 bit ; k = 4 ; M = 16.

2.3.4. Резултати от изследванията



Фиг. 2.3.6. Коригиране на грешката при декомпресиране – метод *“B”*:
Fs = 11025 Nz ; m = 8 bit ; k = 4 ; M = 16.

2.4. Изводи

- Аналитично са обосновани два модифицирани метода за компресиране на речеви сигнали с ADM алгоритми с цел реализирането им на микропроцесорни платформи. В класифицирания като *MADM(B)* метод е предложен нов алгоритъм за образуване на адаптивната функция.

- Аналитично са обосновани методи за автоматно представяне на *MADM(r)* алгоритми с което се избягват операции умножение, деление и сравняване по модул. С използването на методика, базирана на правила, *MADM(r)* алгоритмите се описват таблично, което позволява тяхното софтуерно или хардуерно реализиране на FPGA базиран микроконтролер.

- Изследван е алгоритъм за коригиране на грешката при декомпресиране с *MADM(r)* алгоритми. Експериментално е доказано, че с предложения алгоритъм се повишава качеството на генерираните съобщения.

Съдържание на следващия семинар

Глава 3. Синтезиране на гласови съобщения, базирани на числителни имена, думи, фрази и изречения

3.1 Гласов синтез на количествени числителни имена

3.2 Организация на базата от фонетични примитиви

3.3 Структура на синтезатор за конкатенацоонен синтез на гласови съобщения

3.4 Резултати от изследванията

3.5 Обобщение

Глава 4. Проектиране на FPGA базиран микроконтролер за синтезиране на гласови съобщения „VMCore”

4.1. Методика за развитие на проекта „VMCore”

4.2. Анализ на основните процеси в проекта „VMCore”

4.3. Дефиниране на архитектурният и програмен модел на микроконтролера

4.4. Система инструкции

4.5. Проектиране на хардуерен *MADM(r)* декодер

4.6. Обобщение

Постигнати резултати в дисертационния труд

Научни и научно-приложни приноси

Публикации по дисертационния труд

Насоки за бъдещи изследвания