

[12] 发明专利申请公开说明书

[21] 申请号 01143595. X

[43] 公开日 2002 年 7 月 24 日

[11] 公开号 CN 1360404A

[22] 申请日 2001.12.13 [21] 申请号 01143595. X

[30] 优先权

[32] 2000.12.15 [33] US [31] 09/738,202

[71] 申请人 美国阿尔卡塔尔资源有限合伙公司

地址 美国得克萨斯州

[72] 发明人 迈克·雷

[74] 专利代理机构 中国国际贸易促进委员会专利商标事
务所

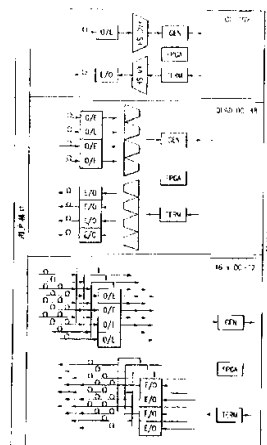
代理人 李德山

权利要求书 2 页 说明书 15 页 附图页数 29 页

[54] 发明名称 用于 FEC 解码器的优化并行输入并行输出 GF(2^M) 乘法器

[57] 摘要

本发明实现了作为乘法器电路的技术优势。乘法器电路乘以 GF(2¹³) 向量, 以实现 sigma 计算以及 Chien 搜索。该乘法器得到两个 将被相乘的输入, 并且仅在一个时钟周期中, 输出在 (2¹³) 向量中产生的乘积。乘法器电路被定制成可以利用 OC-48 以及 OC-192 数据工作, 并且满足 SONET 标准 T1X1.5/99-218R3 以及 SDH 标准 ITU-T G.707/Y.1322。



权 利 要 求 书

- 1.一种用于解码器中的乘法器电路, 包括:
一个时钟, 用于在第一频率上生成多个时钟周期; 以及
一个乘法器模块, 耦合到所述时钟并且接收 2 个输入, 所述乘法器电路利用 1 个所述的时钟周期, 相乘所述的 2 个输入, 以实现只有 1 个时钟周期的等待时间。
- 2.根据权利要求 1 的乘法器电路, 其中所述的 2 个输入为多项式。
- 3.根据权利要求 1 的乘法器电路, 其中所述的 2 个输入为伽罗瓦域向量。
- 4.根据权利要求 1 的乘法器电路, 其中所述的 2 个输入为 $GF(2^{13})$ 向量。
- 5.根据权利要求 1 的乘法器电路, 其中所述的时钟工作在 78MHz。
- 6.根据权利要求 1 的乘法器电路, 其中所述的输入与 OC-48 或者 OC-192 数据相关。
- 7.根据权利要求 1 的乘法器电路, 其中所述的乘法器电路遵守 SDH 标准 ITU-T.G.707/Y.1322。
- 8.根据权利要求 1 的乘法器电路, 其中所述的乘法器模块利用 XOR/AND 门以及锁存器来实现。
- 9.根据权利要求 2 的乘法器电路, 其中所述的乘法器仅利用 XOR/AND 门以及锁存器中实现。
- 10.根据权利要求 1 的乘法器电路, 其中所述的乘法器利用 2 个时钟周期来相乘所述的 2 个输入。
- 11.一种用于解码器中的乘法器电路, 包括:
一个时钟, 用以在第一频率上生成多个时钟周期;
一个乘法器模块, 耦合到所述时钟并且接收 2 个输入, 所述的乘法器电路利用一个流水线以及 2 个所述的时钟周期, 相乘所述的 2 个输入。
- 12.根据权利要求 11 的乘法器电路, 其中所述的 2 个输入为多项式。

13.根据权利要求 11 的乘法器电路,其中所述的 2 个输入为伽罗瓦域向量。

14.根据权利要求 11 的乘法器电路,其中所述的 2 个输入为 GF(2^{13}) 向量。

15.根据权利要求 11 的乘法器电路,其中所述的时钟工作在 78MHz。

16.根据权利要求 11 的乘法器电路,其中所述的输入与 OC-48 或者 OC-192 数据相关。

17.根据权利要求 11 的乘法器电路,其中所述的乘法器电路遵守 SDH 标准 ITU-T.G.707/Y.1322。

18.根据权利要求 11 的乘法器电路,其中所述的乘法器模块利用 XOR/AND 门以及锁存器来实现。

19.根据权利要求 12 的乘法器电路,其中所述的乘法器仅利用 XOR/AND 门以及锁存器中实现。

20.根据权利要求 11 的乘法器电路,其中所述的乘法器利用 2 个时钟周期来相乘所述的 2 个输入。

说明书

用于 FEC 解码器的优化并行输入并行输出 $GF(2^M)$ 乘法器

技术领域

本发明涉及光通信系统，尤其涉及在这些网络中的前向纠错（FEC）功能。

背景技术

由于 SONET 和 SDH 的数据率达到了 2.5Gbps 或更高，因此它们需要通过长光缆传输。这使得不仅检测传输中的误差，而且纠正这些误差以便防止要求保护倒换的频繁误差条件越来越重要。

在 1998 年，包括本发明的申请人在内的 ANSI T1X1.5 技术小组启动了一项开发一种带内前向纠错（FEC）算法的标准用于 SONET 传输设备的工作。（在本申请的下文中称这个标准为“标准”）。这个草案描述了在现有的 SONET 总开销内传输 FEC 校验位和状态/控制位的带内 FEC 标准。

SONET 标准的范围包括 OC-48 和 OC-192。对于一个 STS-48 分组，这个标准能提供每行每位片多达 3 个纠错。对于一个 STS-48 帧，能提供 $3 \times 9 \text{ 行} \times 8 \text{ 位} = 216 \text{ 位}$ 纠错。整形（conforming）电路也应能检测无法纠正的误差条件（即，当每行每位片超过 3 个误差时）。从概念上讲，FEC 层下降到低于线路层（line layer），而且为线路层提供“纠正服务”。详情参见这个申请中包含的标准。

目前存在适用于 SONET 和 SDH 的预期的带内 FEC 电路和方法，它能满足所提出的这些标准的要求，而且在电路区域和编解码时间上能提供最佳性能。这个电路和方法应能工作于 OC-48 和 OC-192 数据率，禁止使用 OC-12 数据，还应满足这些标准的等待时间要求，而且最好应用于高数据率。

发明内容

本发明实现了作为乘法器电路的技术优势。乘法器电路乘以 $GF(2^{13})$

向量，以实现 sigma 计算以及 Chien 搜索。该乘法器得到两个将被相乘的输入，并且仅在一个时钟周期中，输出在 (2^B) 向量中产生的乘积。乘法器电路被定制成可以利用 OC-48 以及 OC-192 数据工作，并且满足 SONET 标准 T1X1.5/99-218R3 以及 SDH 标准 ITU-T.G.707/Y.1322。

附图说明

图 1A 为 10Gbit/s SONET ADM (加/减 MUX) 的总体方框图，其中描述了本发明的 FEC 的存在；

图 1B 描述了 FEC 的生成；

图 1C 描述了 FEC 的终止；

图 1D 为根据本发明用于 SONET 的前向纠错 (FEC) 系统的总体方框图；

图 2 为 FEC 编码器的顶层方框图；

图 3 为 FEC 位片编码器的方框图；

图 4 为 FEC 编码器的方框图；

图 5 为 FEC 解码器的顶层方框图；

图 6 为位解码器的方框图；

图 7 为 FEC 位片解码器的方框图；

图 8 为校验位生成器电路的方框图；

图 9 为图 8 的校验位生成器的信号时序图；

图 10A 为校正子 1 计算电路的方框图；

图 10B 为校正子 3 计算电路的方框图；

图 10C 为校正子 5 计算电路的方框图；

图 11 示意了校正子 $G5(\alpha^5)$ 计算电路的计算过程；

图 12 为校正子 $G3(\alpha^3)$ 的计算过程；

图 13 示意了 $G3(x)$ 校正子计算的计算过程；

图 14 为 sigma3 计算电路的方框图；

图 15 为 sigma2 计算电路的另一方框图；

图 16A 为 Chien 搜索顶层纠错电路的方框图；

图 16B 为 CHIEN_SEARCH_1 部件的方框图；

图 16C 为 CHIEN_SEARCH_CB 部件的方框图;

图 18 描述了平方电路的公式;

图 17 描述了图 18 的平方电路的计算过程;

图 19 描述了乘法器电路的计算过程;

图 20 描述了为 $G1(x)$ 实现 4 位并行 LFSR 的电路;

图 21 描述了为 $G3(x)$ 实现 4 位并行 LFSR 的电路;

图 22 描述了为 $G5(x)$ 实现 4 位并行 LFSR 的电路;

图 23 描述了为 $G(x)$ 实现 4 位并行 LFSR 的电路;

图 24 描述了图 23 电路的计算过程;

图 25 描述了加 4 的伽罗瓦域向量生成器电路的计算过程; 以及

图 26 描述了用于 FEC 解码的反向 $GF(2^{13})$ 串行向量生成器。

具体实现方式

下面为根据本发明的 FEC 编码器的详细描述和算法。前面提供了 SONET 标准的概述, 因此, SONET 标准的理解非常类似于 SDH 标准, 而且本发明一般应用于虽然某些变量有别, 但公式和算法在本质上相似的标准。

FEC 编码的描述和算法

该标准规定了用于编码任何循环码的生成器多项式:

$G(x) = G1(x)G3(x)G5(x)$, 式中

$$G1(x) = x^{13} + x^4 + x^3 + x + 1$$

$$G3(x) = x^{13} + x^{10} + x^9 + x^7 + x^5 + x^4 + 1$$

$$G5(x) = x^{13} + x^{11} + x^8 + x^7 + x^4 + x + 1$$

码字: $C(x) = I(x) + R(x)$

信息位: $I(x) = a_{4358}x^{4358} + \dots + a_{39}x^{39}$

校验位: $R(x) = I(x) \bmod G(x) = a_{38}x^{38} + \dots + a_0$

缩短的 BCH 码源于 a(8191, 8152)母码。

该分组大小为 STS-48 的 1 行(位片)(每个分组 4320 个信息位加上 39 个校验位)。

最小码距 $d=7$, 可纠正误差数 $t=3$ 。

解码 FEC 描述和算法

BCH 码的 FEC 解码

接收码为: $r(x) = C(x) + e(x)$, 式中

$C(x)$ = 传输的码字

$e(x)$ = 误差模式

校正子计算

校正子值 $S_k = r(\alpha^k) = C(\alpha^k) + e(\alpha^k) = e(\alpha^k)$ 公式 2.1

由于 $C(\alpha^k)=0$ 因此 α^k 为 BCH 码的根。

$k=1, 3, 5, \dots, 2t-1$ (t 为接收码字中的误差数, $t \leq 3$)

因此, 校正子的每个元 S_k 只是在 $x = \alpha_k$ 估计的误差模式多项式 $e(x)$

我们称第 i 个误差定位为 X_i , 从公式 2.1 可得到:

$$S_1 = X_1 + X_2 + \dots + X_t$$

$$S_2 = (X_1)^2 + (X_2)^2 + (X_t)^2$$

公式 2.2

$$S_{2t} = (X_1)^{2t} + (X_2)^{2t} + (X_t)^{2t}$$

Peterson 直接求解方法

Peterson 表明, 利用误差定位子多项式 $\sigma(x)$ 可求解 S_k , 因为在每个误差定位子估计的 $\sigma(x)$ 等于 0。

公式 2.3
$$\sigma(X) = X_t + \sigma_1 X^{t-1} + \dots + \sigma_t$$

公式 2.4
$$\sigma(X_i) = X_i^t + \sigma_1 X_i^{t-1} + \dots + \sigma_t \quad i=1,2,\dots,t$$

利用二元码的牛顿识别 (Newton's identities), 可得到

公式 2.5
$$S_1 + \sigma_1 = 0$$

$$S_3 + S_2\sigma_1 + S_1\sigma_2 + \sigma_3 = 0$$

$$S_5 + S_4\sigma_1 + S_3\sigma_2 + S_2\sigma_3 + S_1\sigma_4 + \sigma_5 = 0, \text{ 等等。}$$

Chien 搜索

为找到误差定位子多项式 $\sigma(x)$ 的根，即误差定位子，以及纠正所指示的误差，使用 Chien 搜索。Chien 搜索顺序地通过所有可能的误差定位子值，并在发现误差定位子时纠正相应位。

通过公式 2.5 除以 X^t ，满足 $\sigma(x)=0$ 的 x 值满足公式

$$\sigma_1 x^{-1} + \sigma_2 x^{-2} + \dots + \sigma_t x^{-t} = 1$$

测试误差定位子 α^{n-j} 等效于寻求 σ^j 是否满足

$$\text{公式 2.6} \quad \sum \sigma_i \alpha^{ij} = \alpha^0 = 1, \quad j = 0, 1, 2, \dots, n-1$$

本发明中的 FEC 解码

从公式 2.2

$$S_1 = X_1 + X_2 + X_3$$

$$S_3 = (X_1)^3 + (X_2)^3 + (X_3)^3$$

$$S_5 = (X_1)^5 + (X_2)^5 + (X_3)^5$$

对于本发明：

$$S_1 = r(\alpha) \bmod G1(\alpha)$$

$$S_3 = r(\alpha^3) \bmod G3(\alpha^3)$$

$$S_5 = r(\alpha^5) \bmod G5(\alpha^5)$$

从公式 2.5 求解 $d1, d2, d3$

$$\sigma_1 = S_1$$

$$\sigma_2 = (S_1^2 S_3 + S_5) / (S_1^3 + S_3)$$

$$\sigma_3 = (S_1^3 + S_3) + S_1 \sigma_2$$

只使用 S_1, S_3, S_5 项是因为对二元码， $S_{2k} = S_k^2$ ，因此：

$$(S_1)^2 = S_2, (S_1)^4 = S_4$$

利用 2^m 元 $GF(2^m)$ 的伽罗瓦域特性。在本应用中 $m=13$ 。

从公式 2.6，Chien 搜索 $j, j = 3833, 3834, \dots, 8189$

$$\text{公式 2.7} \quad \sigma_1(\alpha^j) + \sigma_2(\alpha^j)^2 + \sigma_3(\alpha^j)^3 = ? 1$$

如果公式 2.7 = 1，则将位置 $8192-j$ 的位逆转。

FEC 系统顶层设计

参考图 1B, 一般在 10 示意一种根据本发明优选实施例的用于 SONET 的带内 FEC 系统和方法。系统 10 包括 FEC 编码器 12 和 FEC 解码器 11。编码器 12 还包括校验位生成器 14 和线性反馈移位寄存器 (LFSR) $G(x)$ 16。解码器 11 包括校正子生成器 20、sigma 计算 22 以及纠错电路 24。还提供无法校正误差的检测电路 26。校正子生成器 20 进一步包括 3 个 LFSR 用于 FEC 解码, 在图中表示为 27、28 和 30。乘法器 32 和平方器 34 可同时被 sigma 计数 22 和纠错电路 24 使用。纠错电路 24 包括 4 个加 4 LFSR 36 和一个串行 LFSR 38。

通过示意但并不局限于此, 在一个实施例中, 本发明 10 的电路控制 4 位并行电路工作于 78MHz。编码器 12 的电路降低了系统 10 的等待时间, 而且适合一般的 STS-48 和 STS-192 SONET 接收机/发射机系统。另外, 使用串行电路互补并行电路来完成编码和解码。

本发明相比现有技术具有多种优势。例如, FEC 解码器 11 中的解码电路使用离散数学器件计算 $GF(2^m)$, 而不是象现有技术一样使用 ROM 表。诸如平方器 34 和乘法器 32 的数学器件计算一个时钟周期而不是多个时钟周期。这样就提高了解码速度, 而且减轻了路由拥塞。这些功能块被分割以利用离散数学电路和并行电路提供短等待时间 (或小时延)。

在 STS-48 应用中, 如图 1A 所示, 使用 4 个独立的 FEC 编码块和 4 个 FEC 解码块处理字节中的每个位。这些功能块并行工作而且拥有自己的控制器和帧计数器, 这使得每个 STS-48 行能提供多达 24 个突发纠错。对解码器, 每个 STS-48 提供另一顶层控制块。

在 STS-192 应用中, 使用 16 个独立的 FEC 编码块和 16 个 FEC 解码块处理字节中的每个位。这些功能块并行工作而且拥有自己的控制器和帧计数器。对解码器, 每个 STS-192 提供另一顶层控制块。

对于更高的数据率, 如 STS-768, 使用与 STS-48 和 STS-192 相同的方法。

FEC 编码器顶层设计

参考图 2, FEC 编码器 12 由多个位 FEC 编码器 40 组成。每个位编码器 40 的设计相同。编码器 40 由 8 个位片编码器 42 构成。

参考图 3, 每个位片编码器 42 由校验位生成器 44、开销插入电路 46、行数据存储器 48 以及 FEC 开/关延迟电路 50 构成。

参考图 4, 每个位片编码器 42 还包括具有控制器状态机块 54 的控制器块 52, 校验位生成器块 44, FEC 状态指示 (FSI) 位插入块 55, 用于插入校验位的 2 个不同块 58, 以及选择块 60。这些功能块实现和满足了该标准的要求。通过编码系统 12 的总时延约为 14ms。控制器 52 发送信号到校验位生成器 44 以移入信息位, 移入 0, 移出校验位, 以及禁止编码某些位。控制器 52 还发送帧计数器信号和行尾标记符。校验位生成器 44 生成校验位以插入到段开销 (SOH) 和行开销 (LOH)。2 个不同插入块 58 插入具有最小时延和最小校验位存储空间的校验位。1104x4 RAM 59 被第二个校验位插入块用于延迟数据。控制器状态机 54 实现该标准中的状态机要求。数据选择块 60 实现该标准中的不同数据选择模式 (1 = 启动编码, 2 = 带编码时延关闭编码, 3 = 无编码时延关闭编码)。

FEC 解码器的顶层设计

现在参考图 5, FEC 解码器 11 由顶层控制器 70 和各个 FEC 位解码器 72 构成。顶层控制器 70 有状态机 74, 用于控制解码器 72 发送使能信号到各个位解码器 72。通过解码系统 14 的总时延约为 14.6ms。

参考图 6 和图 7, 每个位片解码器 78 有一个主控制器 76, 3 个校正子生成器块 79, 3 个校正子校验块 80, 计算 σ^2 的功能块 82, 计算 σ^3 的功能块 84, Chien 搜索功能块 86, 计数器 87, 存储器 (1154x4 RAM) 88, 纠错功能块 90, 误差计数功能块 92, 数据选择功能块 94, 以及解码器状态功能块 96。

主控制器 76 发送信号以启动/禁止除校正子校验功能块 80 外的其它功能块的功能。校正子校验功能块 80 的控制器 79 生成专用于启动/禁止校正子校验功能 80 的信号。校正子生成器 78 发送校正子到 σ 计算功能块 82 与 84, 这些功能块包含离散的乘法、平方和加法电路。Sigma 结

果被发送到 Chien 搜索 86，在此找到误差多项式的根。Chien 搜索功能块 86 发送误差 ID 位置到纠错功能块 90，块 90 同时纠正延迟的信息和校验位误差（多达 3 个误差）。误差计数功能块 92 统计纠错功能块 90 纠正的误差数。纠正的信息和校验位被送至校正子校验功能块 80，以检测无法校正的误差条件。解码状态功能块 96 将通知上一层这种条件。可能出现信息误差太大甚至无法由校正子校验器 80 检测的情况。这就要依靠 B2 计算来检测这些误差，如图 1A 所示。

FEC 编码的校验位生成器

现在参考图 1，校验位生成器 14 从该标准中得到 $R(X)$ 的公式。

$$R(x) = I(x) \bmod G(x) = a_{38}x^{38} + \dots + a_0$$

$$G(x) = G1(x) * G3(x) * G5(x)$$

图 8 和图 9 详细示意了校验位生成器 14。校验位生成器 14 由 LFSR 100 和 102 组成。每个 LFSR 可同时工作于 4 位并行和 1 位串行 39 位模式。LFSR 100 和 102 协同工作使得数据能连续移入并生成校验位。第一 LFSR 100 移入信息位。在每行的结束，在信息位移入后，第一 LFSR 100 将其内容转储到第二 LFSR 102，LFSR 102 立即移入 39 个 0 位而且执行模操作。一旦移入 0 位，第二 LFSR 102 中的内容即包含 39 个校验位。接着，校验位每次移出 4 位到校验位寄存器 104，再馈入校验位插入功能块 106。

通过利用 $GF(2^{13})$ 和模 2 数学运算相乘和减去这 3 个较小的多项式函数，可导出 LFSR 100 和 102 的串行电路。LFSR 100 和 102 的并行电路与串行电路的函数相同。图 9 描述了校验位生成器 14 的信号时序。执行这些功能的 LFSR 为生成器 44。控制器 52 为编码器生成包括 SHIFT_12_ZERO 在内的信号。

用于 Fed 解码的 FEC 解码器

校正子生成器

现在参考图 10A、图 10B 和图 10C，图 1B 所示的校正子生成器 26、28 和 30 实现下述公式：

$$S_1 = r(\alpha) \bmod G1(\alpha)$$

$$S_3 = r(\alpha^3) \bmod G3(\alpha^3)$$

$$S_5 = r(\alpha^5) \bmod G5(\alpha^5)$$

校正子生成器 26、28 和 30 由计算校正子 1 (S1)、校正子 3 (S3) 和校正子 5 (S5) 的三个功能块构成。每个功能块由两个线性反馈移位寄存器 (LFSR) 110 和 112 构成。每个 LFSR 110 和 112 能同时工作于 4 位并行和 1 位串行 39 位模式。这两个 LFSR 110 和 112 协同工作, 使得数据能连续移入和生成校正子。第一 LFSR 110 移入信息位。在每行的结束, 在信息位移入后, 第一 LFSR 110 将其内容转储到第二 LFSR 112, LFSR 112 移入 39 个校验位而且执行模操作。一旦移入校验位, 第二 LFSR 112 中的内容即包含该校正子。接着, 校验位每次移出 4 位。

参考图 11 和 12, 为计算 S3 和 S5, 如下求解 α^3 和 α^5 中的校正子。图 9 描述了校验位生成器的信号定时。执行这些信号的 LFSR 在 79A、79B、79C 以及 80A、80B 和 80C 示出。控制器 76 和 77 生成这些定时信号, 在此使用信号 SHIFT_12_CB。

BCH-3 的 FEC Sigma 计算

参考图 14 和 15, 对 BCH-3 码的 FEC 解码的 sigma 计算是利用离散数学器件进行的, 而且利用并行结构使这个计算的等待时间很短。

这种计算实现了公式 σ_1 、 σ_2 和 σ_3 :

$$\sigma_1 = S_1$$

$$\sigma_2 = (S_1^2 S_3 + S_5) / (S_1^3 + S_3)$$

$$\sigma_3 = (S_1^3 + S_3) + S_1 \sigma_2$$

本发明实现了如图 16、17 和 18 详细示意做乘法、平方和立方计算的常规乘法器和平方器。立方运算是通过将平方器的输出与其一个输入相乘实现的。加法是利用 XOR 门实现的。除法电路基于 YuH-Tsuen Horng 和 Shyue-Win Wei 在“用于有限域 GF(2^m)的快速反向器和除法器” 1994 IEEE, 中提出的电路, 其内容在此作为参考。

仍参考图 15, sigma2 和 sigma3 计算电路 22 还包括 S1 立方电路 124。

S1*S3 乘法器 122 为 S5 加法器电路 126 提供乘积输出, 而 S1 立方电路 124 输出其立方结果到 S3 加法器电路 130。S1 立方电路 124 包括一个乘法器, 这个乘法器取其输入和平方电路 120 接收 sigma1 的结果。加法器 126 和 130 为 sigma2 除法器电路 132 提供输出。

参考图 15, 平方器 120 将 2 个输入相乘并在 1 个时钟周期输出结果。这个解决方案通常用于 $GF(2^{13})$ 。对其他幂次也可使用相同方法, 但得到的门将不同。参考图 17 和 18 对平方器电路、乘法器电路和立方电路的详细描述。

对 FEC 解码的纠错

现在参考图 16A, 本发明使用并行 Chien 搜索功能块每次搜索根 4 位, 如 140 所示。这种并行方案缩短了等待时间而且满足所使用的 4 位 78Mhz 数据率。串行 Chien 搜索纠正校验位误差。Chien 搜索功能块实现下述公式。

从公式 2.6, 利用 Chien 搜索 j , $j = 3833, 3834, \dots, 8189$

$$\text{公式 2.7} \quad \sigma_1(\alpha^j) + \sigma_2(\alpha^j)^2 + \sigma_3(\alpha^j)^3 = 1$$

如果公式 2.7=1, 那么位置 $8192-j$ 处的位被逆转。

仍参考图 16A, 在 140 示出了 Chien 搜索纠错电路。电路 140 包括 Sigma 同步器电路 142, 其有 3 个输出, 标记为 Sigma 1 sync, Sigma 2 sync 和 Sigma 3 sync, 它们的输出提供给 146 所示的 Chien 搜索电路。每个 Chien 搜索电路 146 提供的相应输出表示为误差 ID。

如图 16A 所示, CHIEN_SEARCH_1 功能块 146, CHIEN_SEARCH_2 功能块 147 CHIEN_SEARCH_3 功能块 149, 以及 CHIEN_SEARCH_1 功能块 151 为信息位误差和某些校验位误差 (行 3、5、6、7、8 和 9 的校验位) 生成 Error Id。CHIEN_SEARCH_1 功能块 146 纠正该数据比特位置 3 的误差。CHIEN_SEARCH_2 功能块 147 纠正该数据比特位置 2 的误差。CHIEN_SEARCH_3 功能块 149 纠正该数据比特位置 1 的误差。CHIEN_SEARCH_4 功能块 151 纠正该数据比特位置 0 的误差。

参考图 16B, 其示出了 CHIEN_SEARCH_1 功能块的低层。图中示

出了功能块 200 生成伽罗瓦域向量(为信息位和校验位)。POWER_GEN1 功能块 200 为信息位位置和行 3、5、6、7、8 和 9 的校验位位置生成伽罗瓦域向量,以发送到数学器件 SQUARER 功能块 204, CUBER 功能块 206、SIGMA1_MULT 功能块 210、SIGMA2_MULT 功能块 212 以及 SIGMA3_MULT 功能块 214。Sigma 乘法器功能块 210、212、和 214 的结果被送至 ADD_COMPARE 功能块 216,生成 error_id。CUBER 功能块 206 由一个乘法器功能块构成,其从 SQUARER 功能块 204 装入平方后的结果。数学器件 204、206、210、212、214 和 216 实现 Chien 搜索公式 2.7。每个 CHIEN_SEARCH_2 功能块 147 CHIEN_SEARCH_3 功能块 149 和 CHIEN_SEARCH_4 功能块 151 的低层都类似于图 16B 所示的 CHIEN_SEARCH_1 146,除了它们生成在数据中特定比特位置相关的不同向量位置。生成的误差 id 被纠错功能块 90 用于在通过 1152x4 RAM 88 移位时纠正延迟的数据。

图 16C 示出了 CHIEN_SEARCH_CB 功能块 153 的低层,153 为所有校验位误差生成 Error Ids。图 16C 示出了生成串行伽罗瓦域向量的 POWER_GEN_CB 功能块 218。SQUARER 功能块 220、CUBER 功能块 222、MULTIPLIER 功能块 226、228、230,以及 ADD_COMPARE 功能块 216 为生成的测试错误根位置的每个伽罗瓦域向量实现 Chien Search 公式。CUBER 功能块 222 由一个乘法器功能块构成,其从 SQUARER 功能块 220 装入平方后的结果。CHIEN_SEARCH_CB 功能块 153 提供的 Error Ids 用于纠正存储的校验位,以用于纠错故障检测校正子校验功能块 80。行 1、2 和 4 的校正存储的校验位在数据移过 88 时插入到开销中。

对 FEC 解码的纠错故障检测

“纠正的”信息位和校验位被馈入并行校正子 80 生成器以判断校正子是否为 0。如果校正子不为 0,则纠错宣布失败。这是确定是否存在 3 个以上误差(最大可纠错数)的一种可靠而简单的办法。尽管这个故障检测模块 26 无法捕获所有的纠错失败情况,但这个方案对预测何时纠错

失败很可靠。校验电路 80 与用于校正子生成器 79 的电路相同，因此这个方法被再次利用而且等待时间短。如果对于一个特定行，总误差 id 数（信息和校验位）大于 3，则校验器 26 宣称纠错失败，因为一个特定行生成的误差 id 应为 3 个或更少。由于 CHIEN_SEARCH_CB 功能块 153 为所有校验位生成 error_id，而其他 CHIEN_SEARCHs 功能块 146、147、149 和 151 为行 3、5、6、7、8 和 9 中的校验位生成 error_id。误差计数功能块 92 确保校验位 error_id 在同一行不被计数两次。因此只有 CHIEN_SEARCH_CB 功能块 153 生成的 error_id 被加入到行 1、2 和 4 的其他 error_id。

并入-并出 GF (2^{13}) 优化的平方器

现在参考图 15，为实现图 16B 的 sigma2 计算 22 和 Chien 搜索，GF (2^{13}) 向量被平方器 120 平方。

为比较：

（Jain, Surendra k. and Parhi, Keshab K. “基于低等待时间标准的 GF (2^M) 乘法器和平方器结构” IEEE 1995）

Jain 和 Parhi 对平方器的解决方案：

基本小区数： $m(m/2)=13(13/2)=84.50$ 。

等待时间= $m/2=6.5=7$ 个时钟周期。

每个小区取 3 个 2 输入 AND，3 个 2 输入 XOR 门和 4 个 1 位锁存器。GF (2^{13}) 的总 AND 门>253，XOR 门>253，338 个 1 位锁存器。

本发明：共：23 个 XOR 门，13 个锁存器。由于简化设计的门数少，因此等待时间=1 个时钟周期。参考图 18，图 18 示意了平方器 120 的计算。

为适用于 VLSI 装置，使用门优化的、等待时间平方电路。

并入并出 GF (2^{13}) 优化的乘法器

为实现 sigma 计算和 Chien 搜索，GF (2^{13}) 向量被乘法器 122 相乘。本发明有一个乘法器 122，其取 2 个输入相乘并在 1 个时钟周期输出 GF

(2^{13}) 的乘法结果。

为比较:

(Jain, Surendra k. and Parhi, Keshab K. “基于低等待时间标准的 GF (2^M) 乘法器和平方器结构” IEEE 1995)

Jain 和 Parhi 对乘法器的解决方案:

基本小区数: $M^2=169$ 。

等待时间= $m+1=14$ 个时钟周期。

基本小区有 2 个 2 输入 AND 门, 2 个 2 输入 XOR 门和 3 个 1 位锁存器。GF (2^{13}) 的总 AND 门数=338, XOR 门=338, 以及 507 个 1 位锁存器。

本发明: 共: 368 个 XOR/AND 门, 13 个锁存器。由于不使用流水线结构, 因此等待时间=1 个时钟周期。

该乘法器还有 2 级流水线连接以满足更严格的定时要求。完成 1 个乘法需 2 个时钟周期。参考图 19, 第一级为 TERMS (0-24), 而第二级为 RESULT_M (0-12)。

现在参考图 19, 其示意了乘法电路 122 的计算, 包括中间项和 GF(2^{13}) 的减法结果。

用于 FEC 解码的 $G1(x)$ LFSR

现在参考图 20, 图 20 描述了实现 4 位并行 LFSR 的电路 140:

$$G1(x) = x^{13} + x^4 + x^3 + x + 1$$

这种 LFSR 实现了根据标准的 $G1(x)$ 函数, 工作于 78Mhz, 而且在本例中有 4 位并行输入。该电路为校正子生成器电路实现了 4 位并行线性移位寄存器 (LFSR), 以根据本发明实现 FEC。

用于 FEC 解码的 $G3(x)$ LFSR

现在参考图 21, 图 21 描述了实现 4 位并行 LFSR 的电路 150:

$$G5(x) = x^{13} + x^{10} + x^9 + x^7 + x^5 + x^4 + 1$$

这种 LFSR 实现了根据标准的 $G3(x)$, 工作于 78Mhz, 而且有 4 位

并行输入。该电路为校正子生成器电路实现了 4 位并行线性移位寄存器 (LFSR)，以根据本发明实现 FEC。

用于 FEC 解码的 G5(x) LFSR

现在参考图 22，图 22 描述了实现 4 位并行 LFSR 的电路 160:

$$G5(x) = x^{13} + x^{11} + x^8 + x^7 + x^4 + x + 1$$

这种 LFSR 实现了根据标准的 G5 (x)，工作于 78Mhz，而且有 4 位并行输入。该电路为校验位生成器电路实现了 4 位并行线性移位寄存器 (LFSR)，以根据本发明实现 FEC。

用于 FEC 解码的 G(x) LFSR

现在参考图 23，图 23 描述了实现 4 位并行 LFSR 的电路 16:

$$G(X) = G1(X) * G3(X) * G5(X)$$

这种 LFSR 实现了根据标准的 GX (x)，工作于 78Mhz，而且有 4 位并行输入。该电路为校验位生成器电路实现了 4 位并行线性移位寄存器 (LFSR)，以根据本发明实现 FEC。图 24 描述了 4 位并行 LFSR 的计算。

用于 FEC 解码的伽罗瓦域 (GF 2¹³) “加 4” 向量生成器电路

现在参考图 25，一般在 170 有 4 个伽罗瓦域生成器。由于 FEC 解码器 11 在 78Mhz 工作于 4 位并行模式，因此生成 4 个 GF2¹³ 向量和馈送 4 个 Chien 搜索电路 (用于纠错) 的电路并行工作以搜索误差多项式的根。

这个电路 170 用于创建加 4 而不是加 1 的 4 个向量生成器。这个电路 170 为基本的 GF2¹³ 向量生成器的变型，由一个线性反馈移位寄存器构成。它加 4 而不是加 1。

用于 FEC 解码的反向 (GF 2¹³) 串行向量生成器

现在参考图 26，在 180 示出了一个生成串行 GF(2¹³) 向量用于校验位的纠错的电路。这个电路 180 为下述公式实现反向串行 LFSR:

$$G1(x) = x^{13} + x^4 + x^3 + x + 1。$$

参考的标准 T1X1.5/99-218R3 为本发明的使用和操作提供了附加的详细信息，包括描述带内 FEC 校验位的位置，状态/控制位的位置，码组定义和交织，STS-N 信号 (N=192)，FEC 状态指示 (FSI)，STS-48 信号的 FSI 编码，STS-N 信号 (N=192)，在编码器和解码器的 B1 计算，以及在编码器和解码器的 B2 计算。另外，提供了 FEC 激活和去活的工作状态，以及 FEC 状态指示 FSI，线 AIS 与 FEC 的相互作用，以及相关的图表。这个标准还综合本发明到总标准中，它满足这个标准的所有要求。

尽管是针对一个特定的优选实施例描述本发明的，但本领域的技术人员在读过本发明后能知道多种变化和改进。因此本发明的目的是，考虑现有技术尽可能广泛地解释所附权利要求书以包括所有这些变化和改进。

图 1A

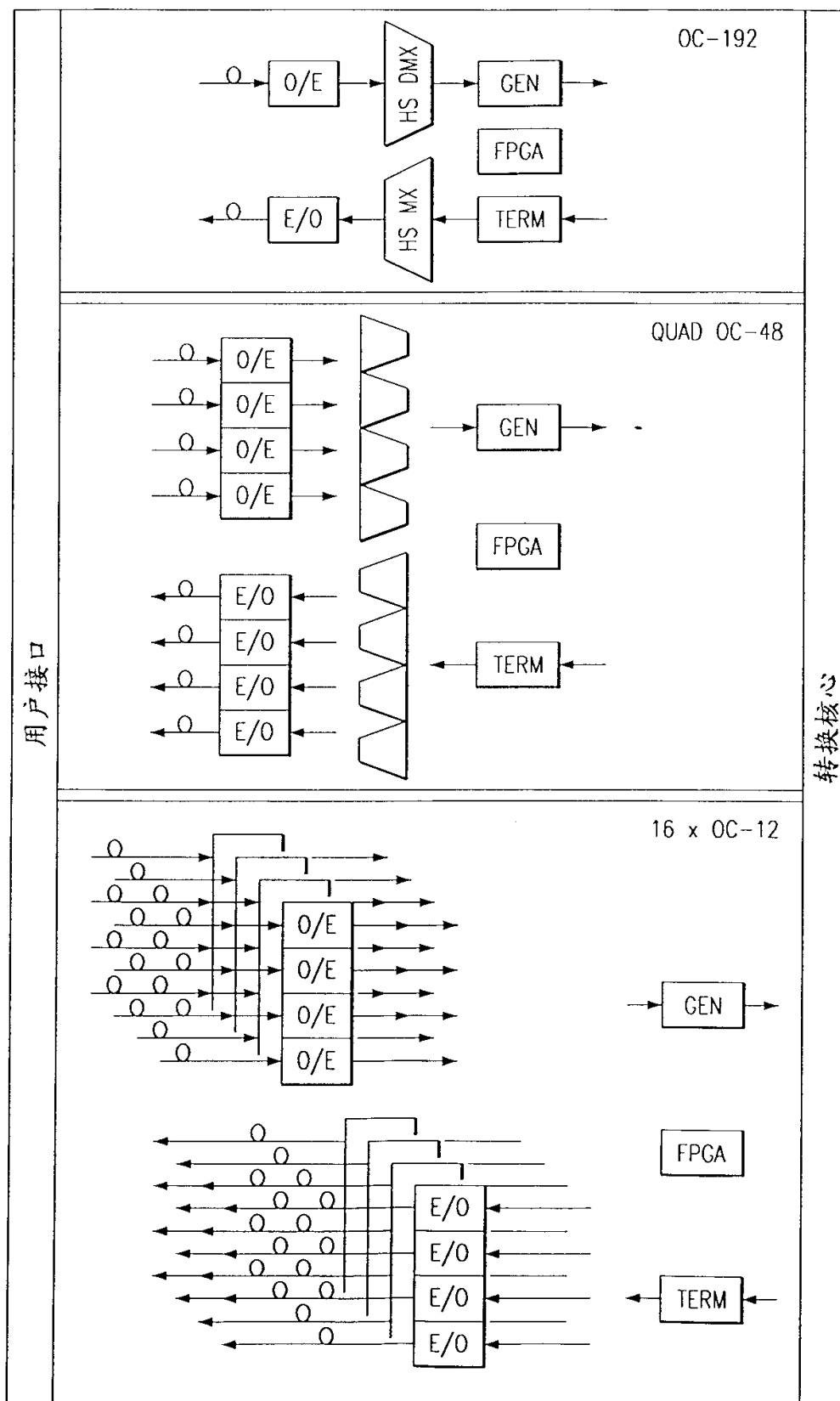
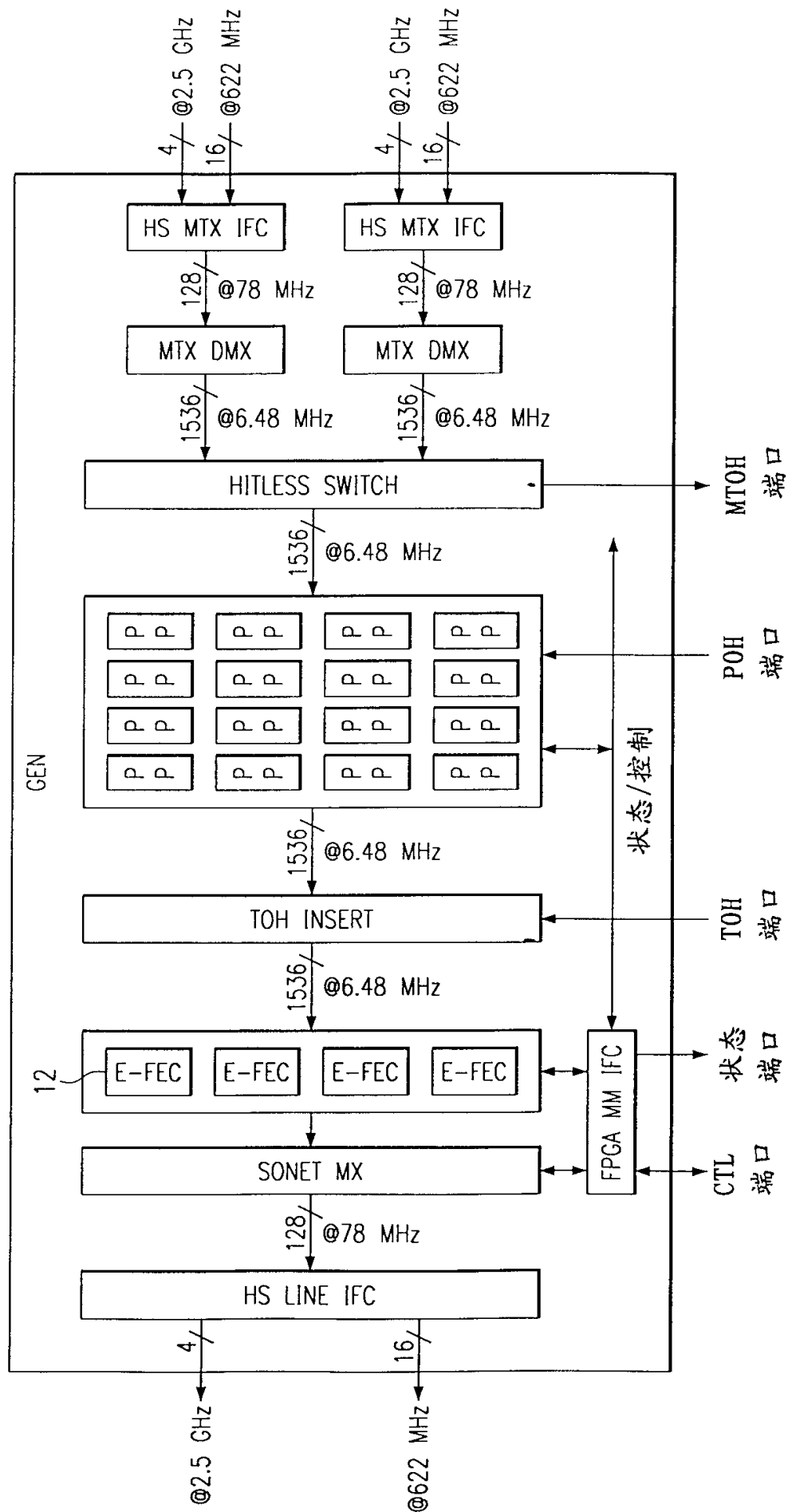


图 1B



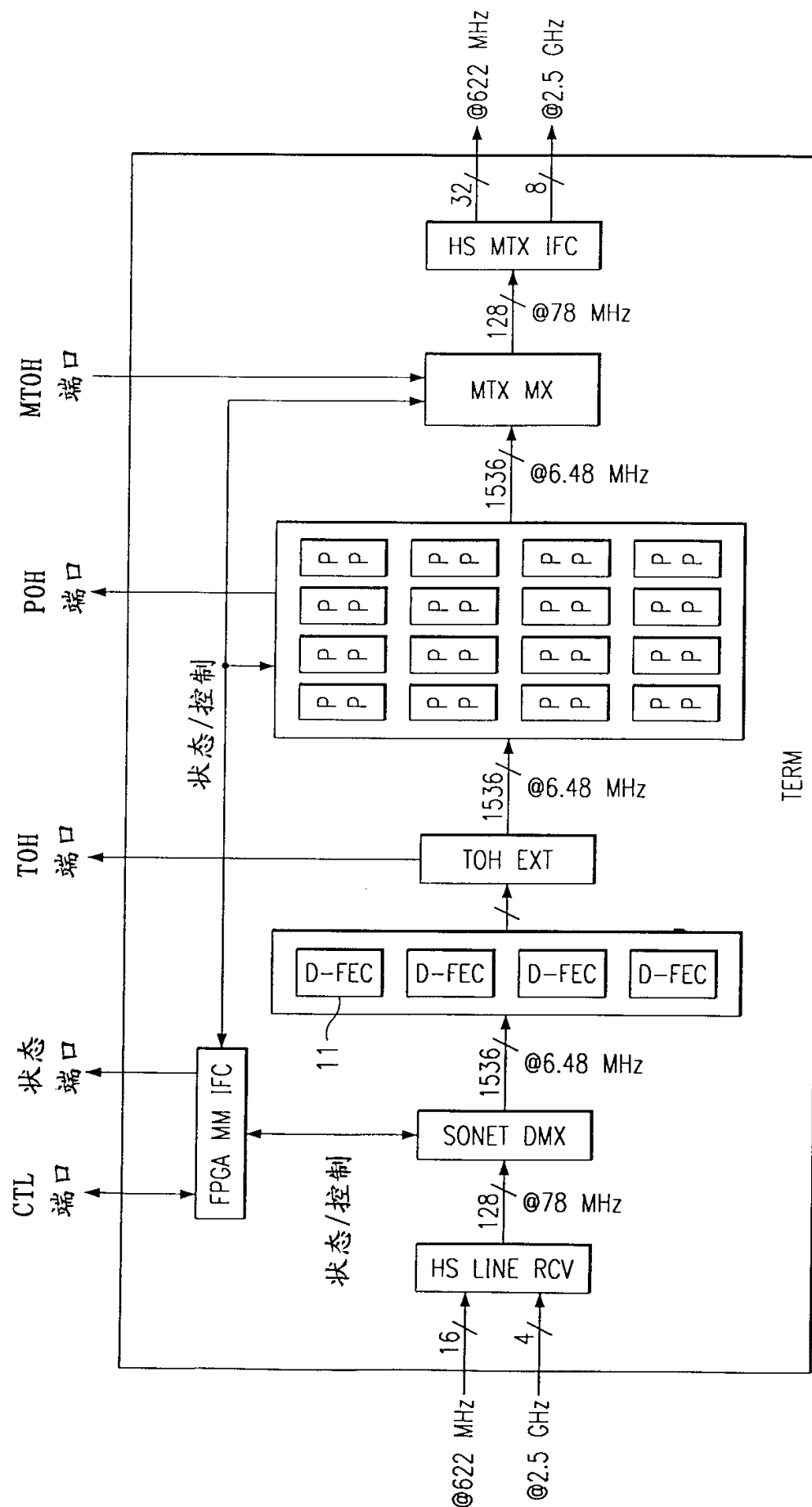
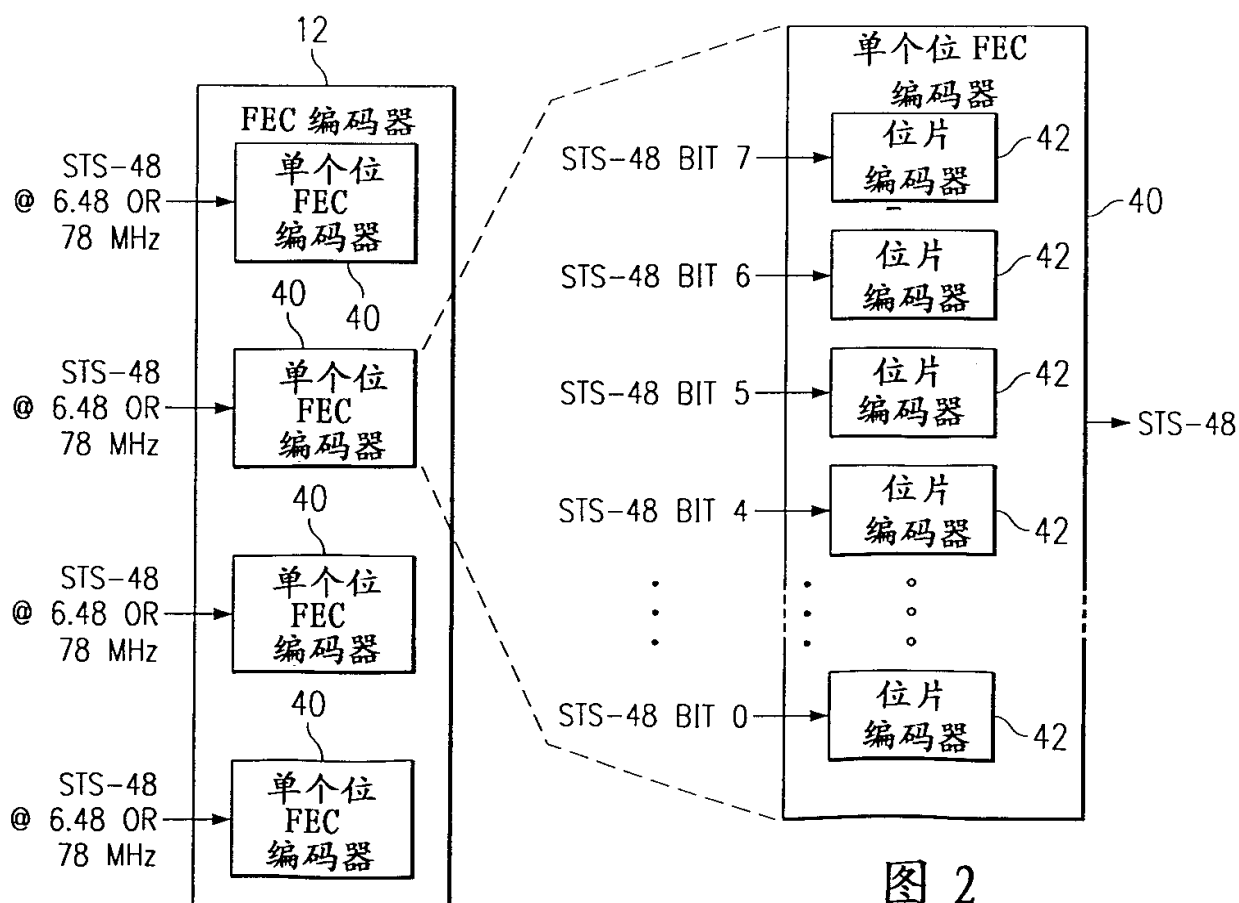
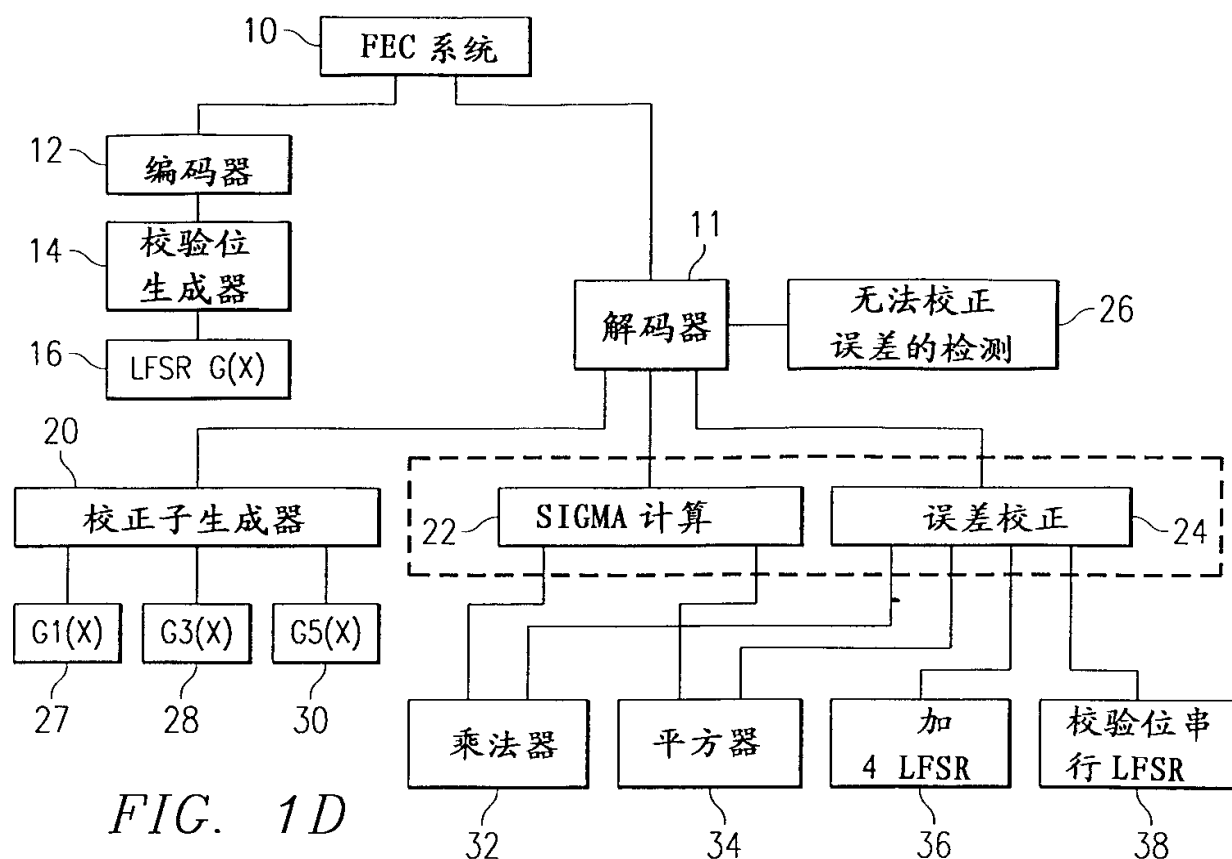


图 1C



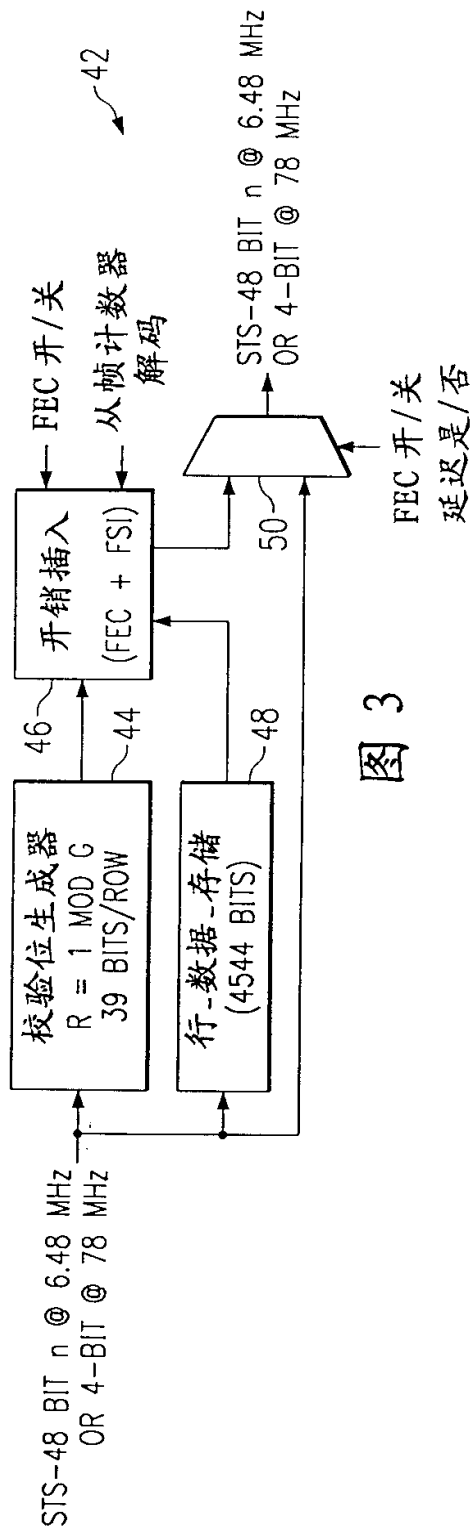


图 3

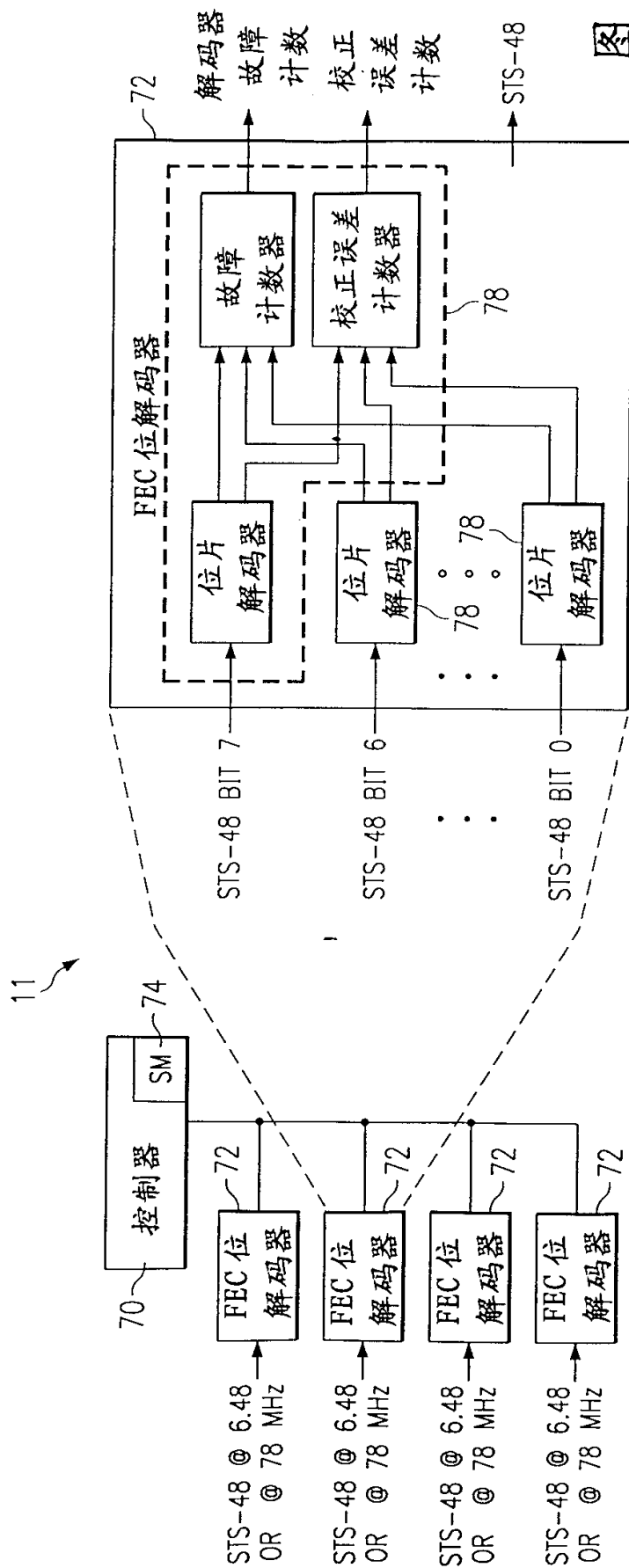


图 5

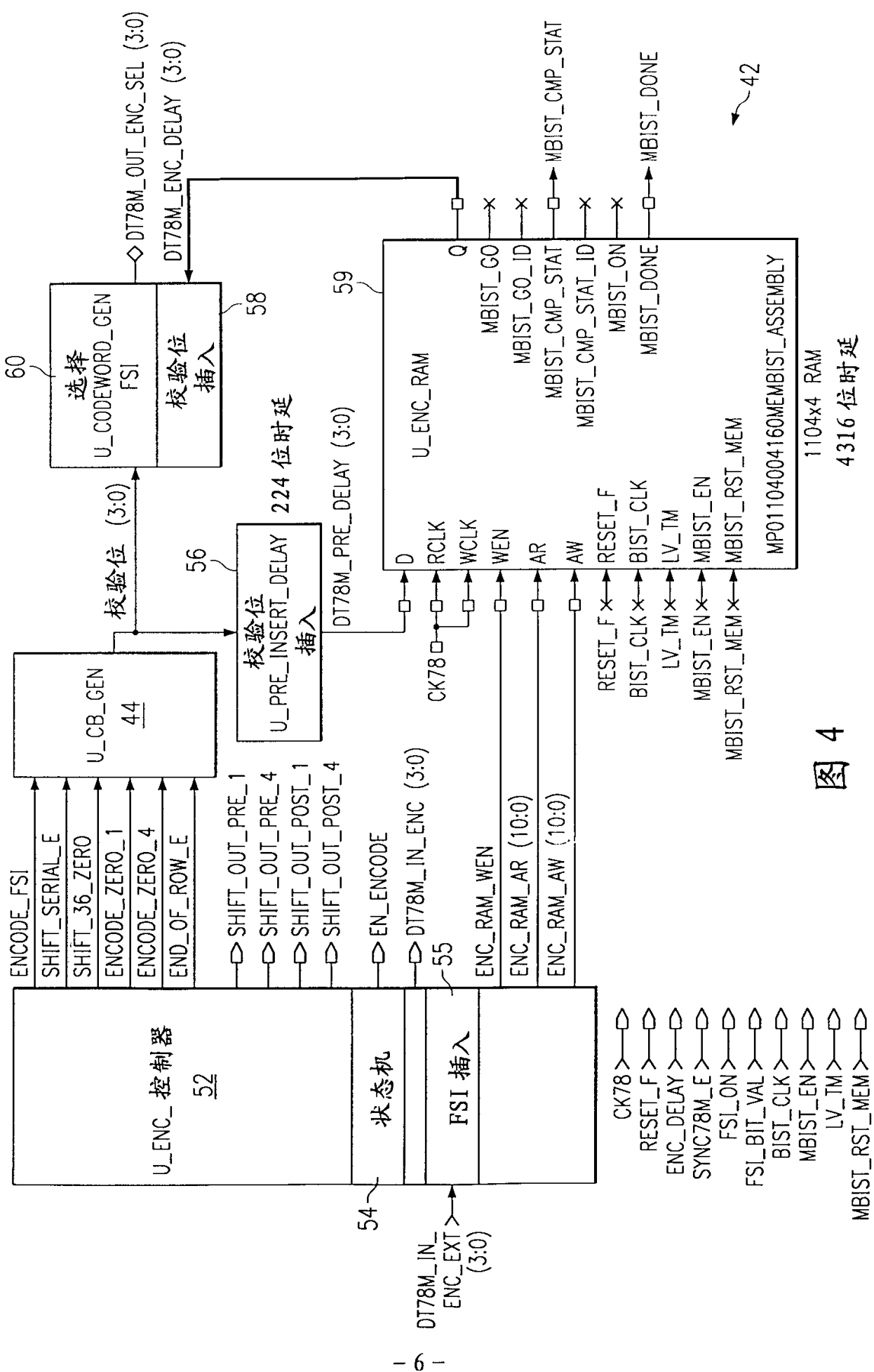


图 4

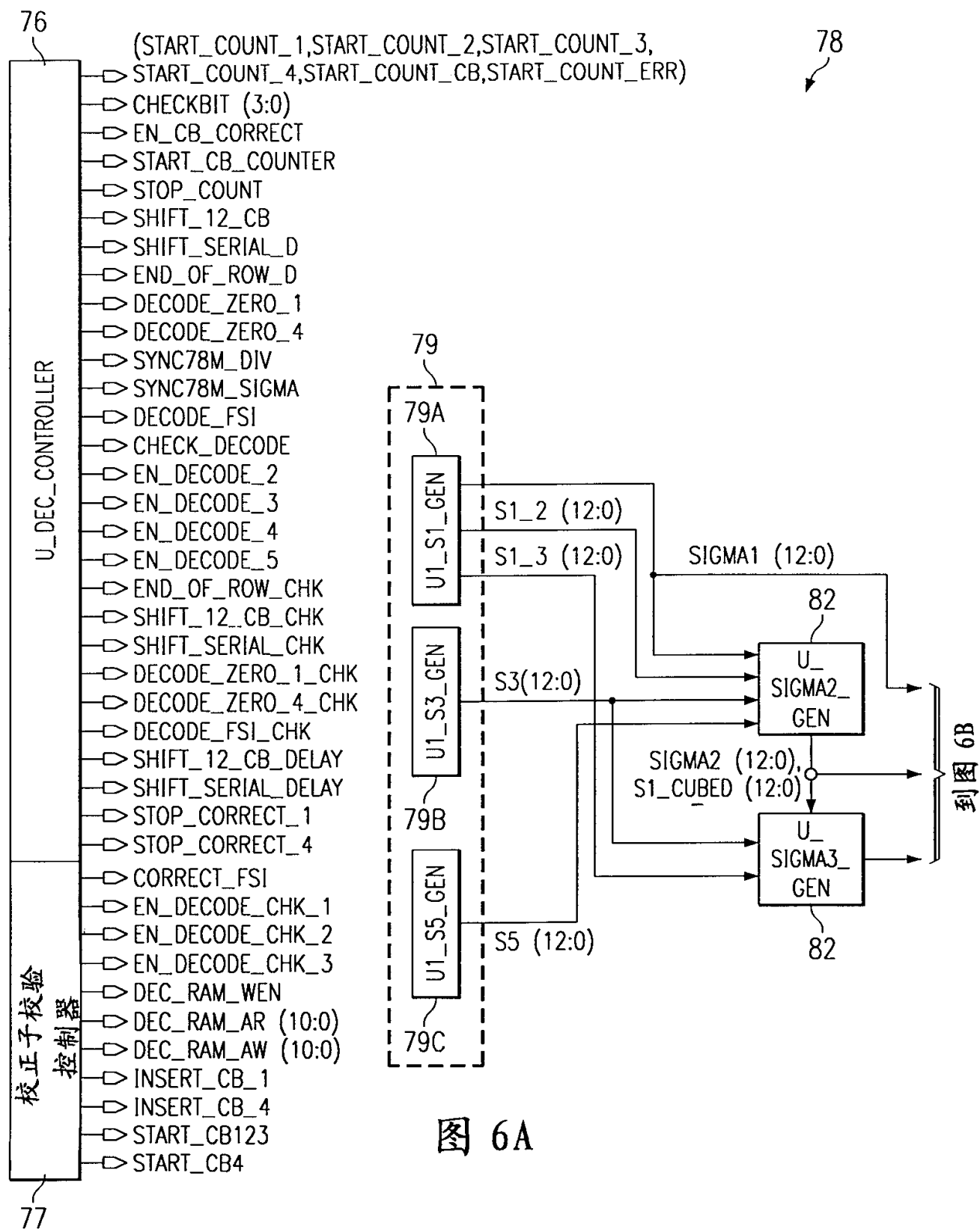
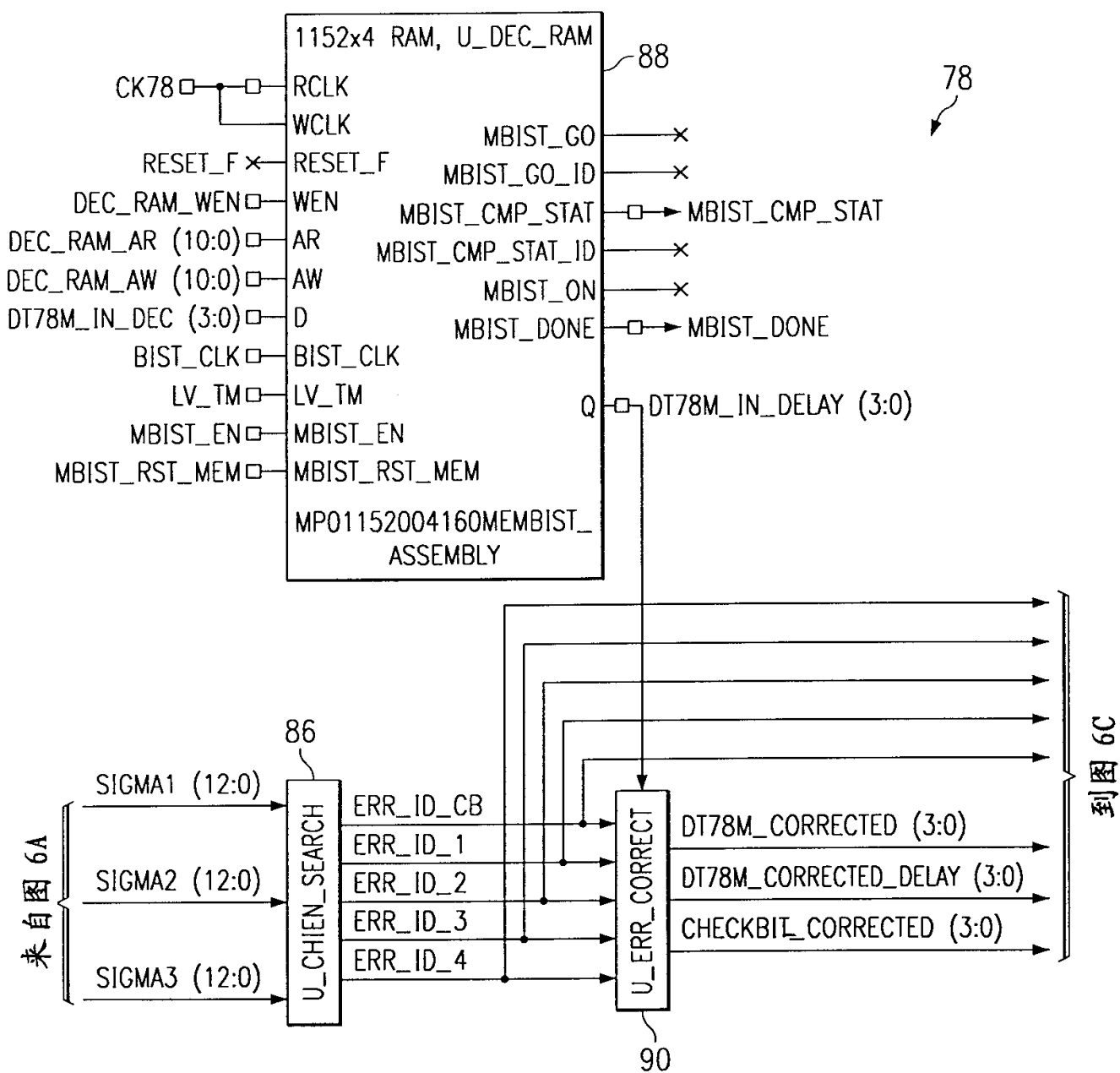


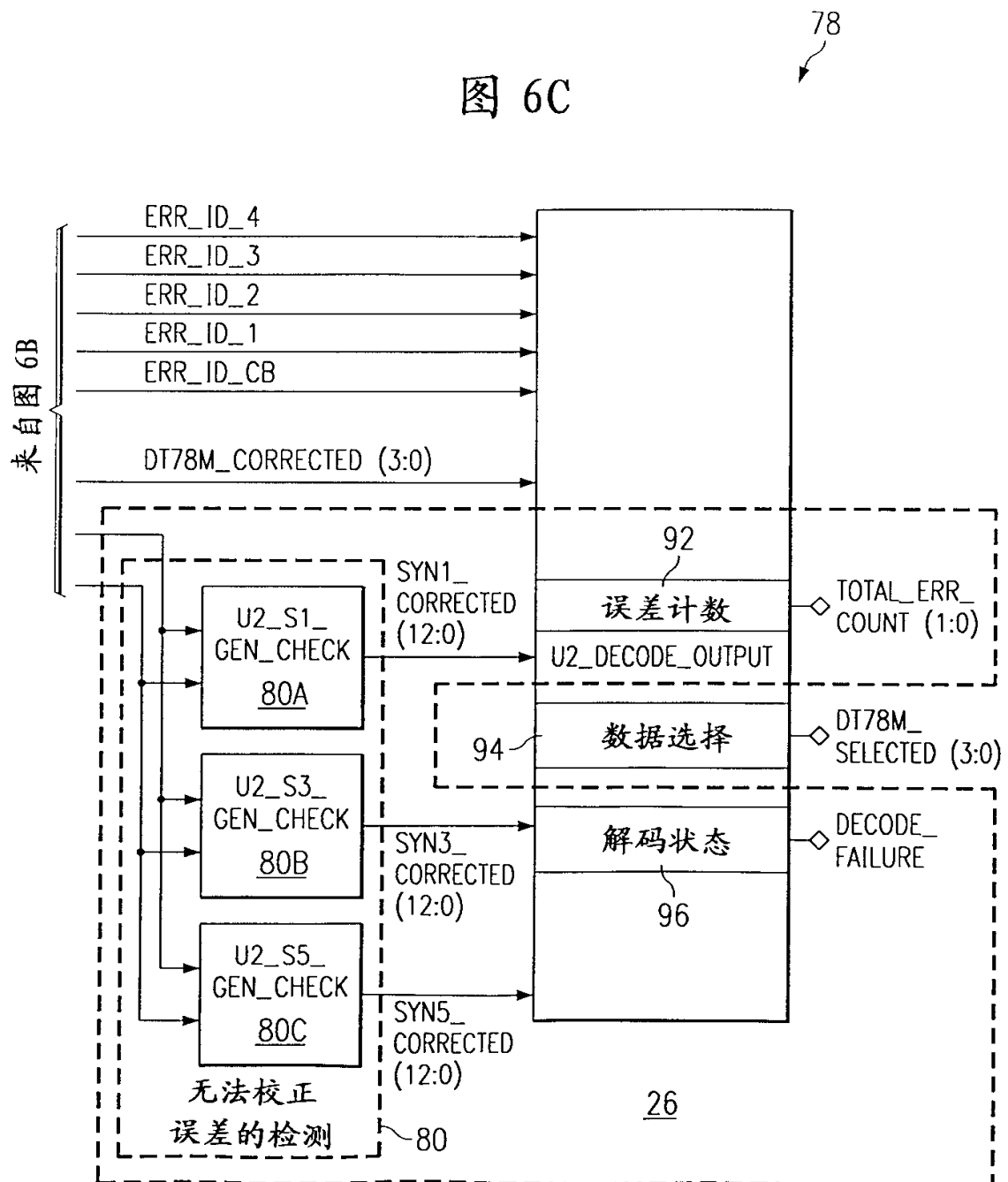
图 6A



DT78M_IN_DEC (3:0) >—>
 CK78 >—>
 RESET_F >—>
 SYNC78M_D >—>
 DEC_DELAY >—>
 EN_DECODE >—>
 BIST_CLK >—>
 MBIST_EN >—>
 LV_TM >—>
 MBIST_RST_MEM >—>

图 6B

图 6C



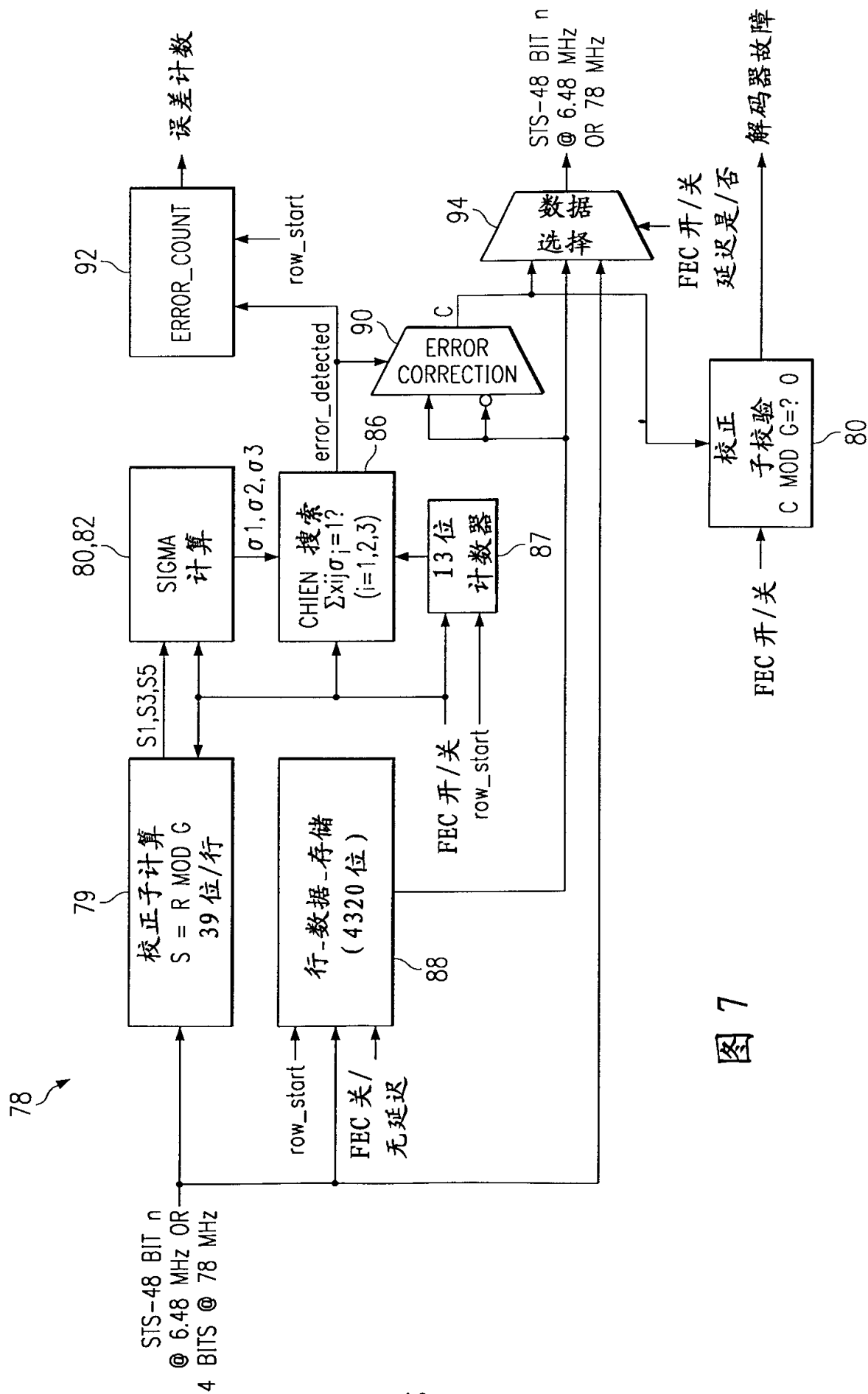


图 7

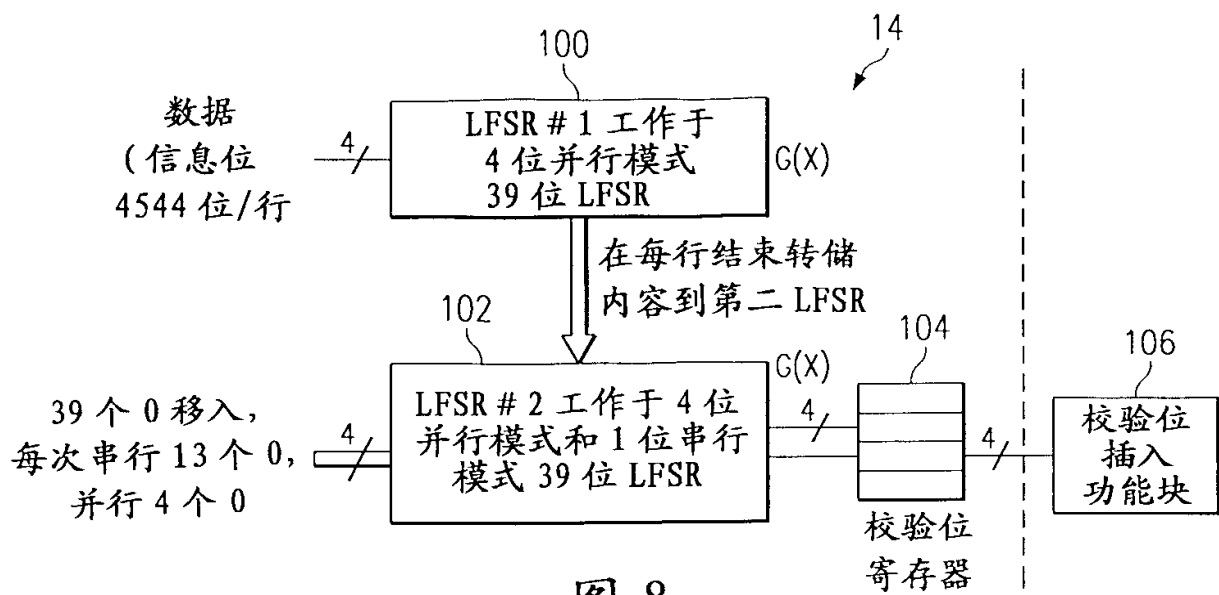


图 8

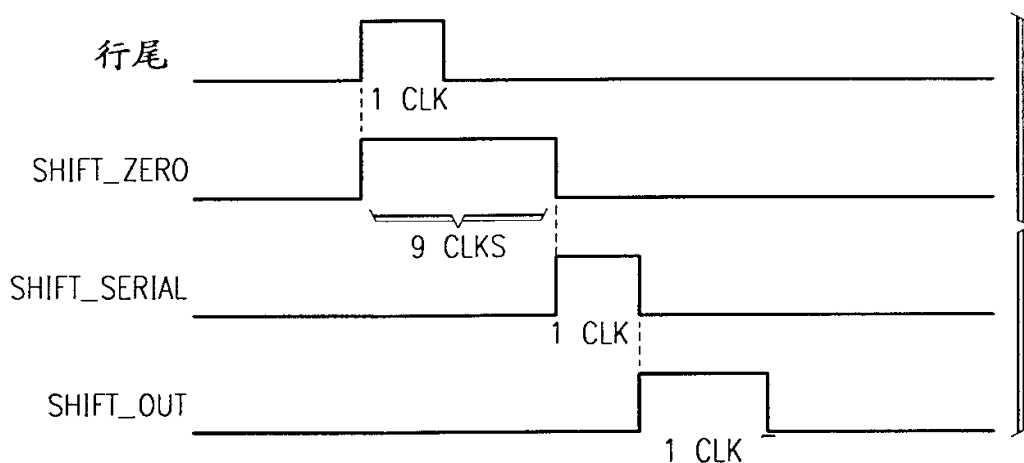


图 9A

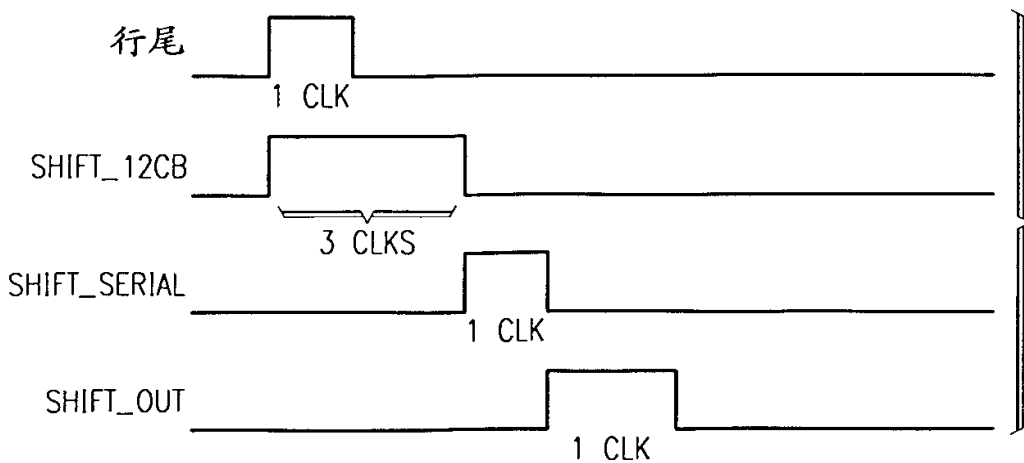


图 9B

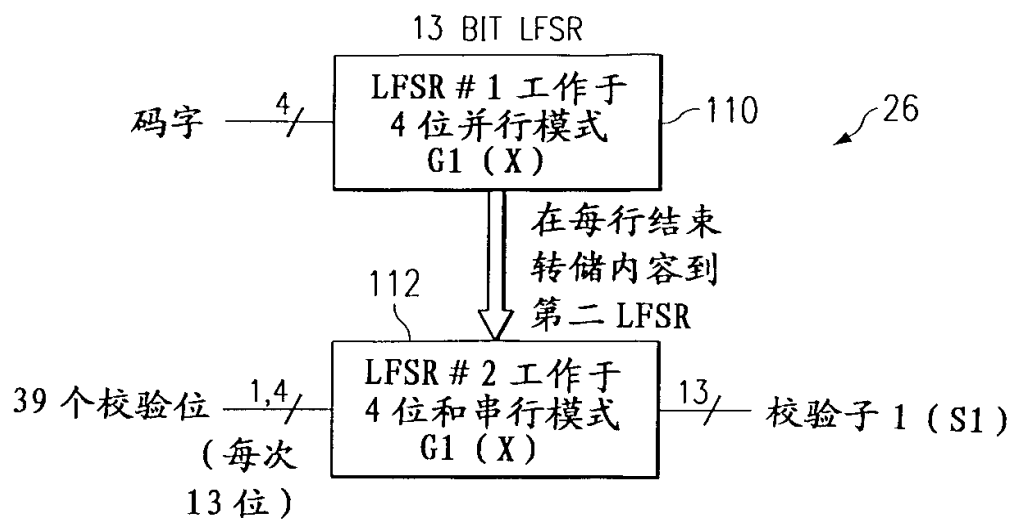


图 10A

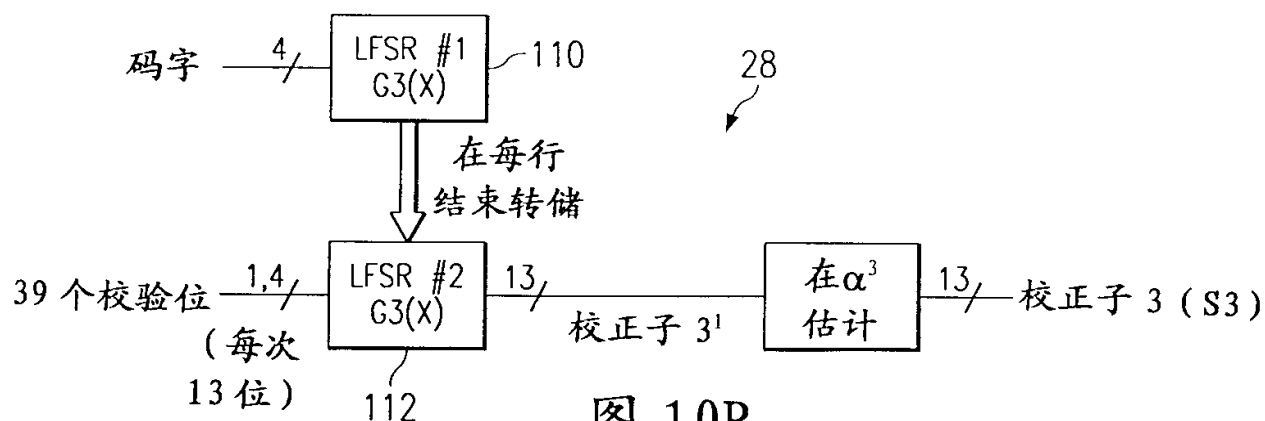


图 10B

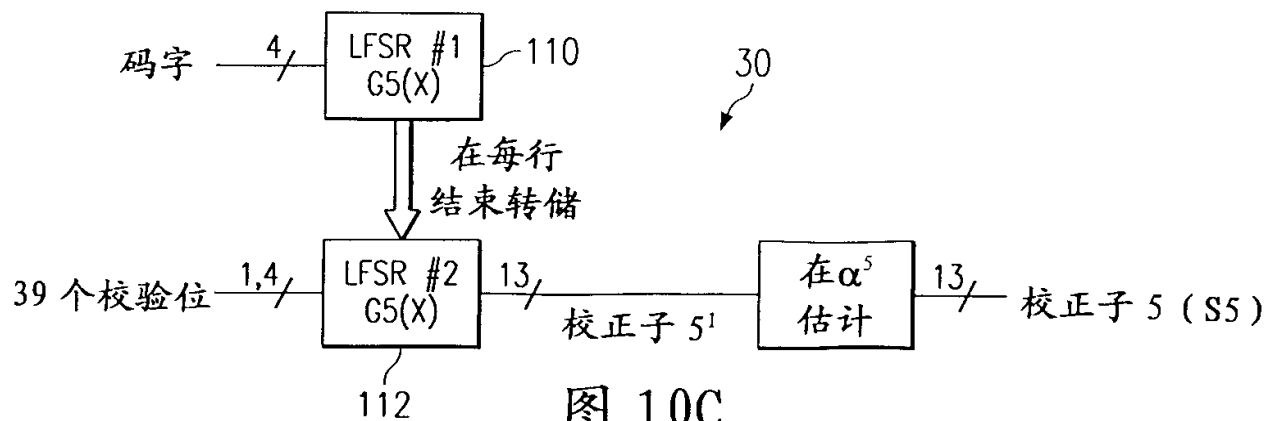


图 10C

$$\alpha_{12}x^{60}+\alpha_{11}x^{55}+\alpha_{10}x^{50}+\alpha_9x^{45}+\alpha_8x^{40}+\alpha_7x^{35}+\alpha_6x^{30} \\ +\alpha_5x^{25}+\alpha_4x^{20}+\alpha_3x^{15}+\alpha_2x^{10}+\alpha_1x^5+\alpha_0x^0$$

$$\alpha^{60} = \begin{array}{cccccccccccccc} 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \alpha^{10} + \alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha \end{array}$$

$$\alpha^{55} = \begin{array}{cccccccccccccc} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \alpha^{10} + \alpha^9 + \alpha^5 + \alpha^2 \end{array}$$

$$\alpha^{50} = \begin{array}{cccccccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ \alpha^{12} + \alpha^{11} + \alpha^{10} + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^0 \end{array}$$

$$\alpha^{45} = \begin{array}{cccccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \alpha^{11} + \alpha^{10} + \alpha^9 + \alpha^8 + \alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + \alpha^0 \end{array}$$

$$\alpha^{40} = \begin{array}{cccccccccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ \alpha^{12} + \alpha^{11} + \alpha^9 + \alpha^8 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha^0 \end{array}$$

$$\alpha^{35} = \begin{array}{cccccccccccccc} 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ \alpha^{11} + \alpha^9 + \alpha^8 + \alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 \end{array}$$

$$\alpha^{30} = \begin{array}{cccccccccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \alpha^{12} + \alpha^{10} + \alpha^6 + \alpha^4 \end{array}$$

$$\alpha^{25} = \alpha^0 + \alpha^1 + \alpha^2 + \alpha^3 + \alpha^5 + \alpha^7 + \alpha^{12}$$

$$\alpha^{20} = \alpha^{11} + \alpha^{10} + \alpha^8 + \alpha^7$$

$$\alpha^{15} = \alpha^0 + \alpha^5 + \alpha^3 + \alpha^2$$

$$\alpha^{10}$$

$$\alpha_0 (b_0 + b_5 + b_8 + b_9 + b_{10})$$

$$\alpha_1 (b_5 + b_{12})$$

$$\alpha_2 (b_3 + b_5 + b_7 + b_{11} + b_8 + b_9)$$

$$\alpha_3 (b_3 + b_5 + b_7 + b_{10} + b_{12})$$

$$\alpha_4 (b_6 + b_7 + b_{10} + b_{12} + b_8)$$

$$\alpha_5 (b_1 + b_3 + b_5 + b_8 + b_{10} + b_{11} + b_9)$$

$$\alpha_6 (b_3 + b_6 + b_7 + b_8 + b_9 + b_{12})$$

$$\alpha_7 (b_4 + b_5 + b_7 + b_9 + b_{12})$$

$$\alpha_8 (b_4 + b_7 + b_8 + b_9)$$

$$\alpha_9 (b_7 + b_8 + b_9 + b_{11})$$

$$\alpha_{10} (b_2 + b_4 + b_6 + b_9 + b_{10} + b_{11} + b_{12})$$

$$\alpha_{11} (b_4 + b_7 + b_8 + b_9 + b_{10})$$

$$\alpha_{12} (b_5 + b_6 + b_8 + b_{10})$$

图 11

$$a_{12}x^{12}+a_{11}x^{11}+a_{10}x^{10}+a_9x^9+a_8x^8+a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x^1+a_0x^0$$

$$a_{12}x^{36}+a_{11}x^{33}+a_{10}x^{30}+a_9x^{27}+a_8x^{24}+a_7x^{21}+a_6x^{18}+a_5x^{15}+a_4x^{12}+a_3x^9+a_2x^6+a_1x^3+a_0x^0$$

$$a_{12} (\alpha^3+\alpha^4+\alpha^5+\alpha^7+\alpha^8+\alpha^9+\alpha^{10}+\alpha^{12})$$

$$a_{11} (\alpha^0+\alpha^1+\alpha^2+\alpha^4+\alpha^5+\alpha^6+\alpha^7+\alpha^9)$$

$$a_{10} (\alpha^4+\alpha^6+\alpha^{10}+\alpha^{12})$$

$$a_9 (\alpha^1+\alpha^3+\alpha^7+\alpha^9)$$

$$a_8 (\alpha^{12}+\alpha^{11}+\alpha^6+\alpha^4+\alpha^3+\alpha)$$

$$a_7 (\alpha^{12}+\alpha^{11}+\alpha^9+\alpha^8)$$

$$a_6 (\alpha^9+\alpha^8+\alpha^6+\alpha^5)$$

$$a_5 (\alpha^6+\alpha^5+\alpha^3+\alpha^2)$$

$$a_4 \alpha^{12}$$

$$a_3 \alpha^9$$

$$a_2 \alpha^6$$

$$a_1 \alpha^3$$

$$a_0 \alpha^0$$

α^{12}	α^{11}	α^{10}	α^9	α^8	α^7	α^6	α^5	α^4	α^3	α^2	α^1	α^0
a_4	a_7	a_{10}	a_3	a_6	a_9	a_2	a_5	a_8	a_1	a_5	a_8	a_0
a_7	a_8	a_{12}	a_6	a_7	a_{11}	a_5	a_6	a_{10}	a_5	a_{11}	a_9	a_{11}
a_8			a_7	a_{12}	a_{12}	a_6	a_{11}	a_{11}	a_8		a_{11}	
a_{10}			a_9			a_8	a_{12}	a_{12}	a_9			
a_{12}			a_{11}			a_{10}			a_{12}			
			a_{12}			a_{11}						

图 12

$$\begin{aligned}
G_3(X) &= x^{13} + x^{10} + x^9 + x^7 + x^5 + x^4 + 1 \\
&= a_{10}(x^{10})^3 + a_9(x^9)^3 + a_7(x^7)^3 + a_5(x^5)^3 + a_4(x^4)^3 + a_0(x^0)^3 \\
&= a_{10}x^{30} + a_9x^{27} + a_7x^{21} + a_5x^{15} + a_4x^{12} + a_0x^0 \\
&= a_{10}(\alpha^4 + \alpha^6 + \alpha^{10} + \alpha^{12}) \\
&\quad + a_9(\alpha^1 + \alpha^3 + \alpha^7 + \alpha^9) \\
&\quad + a_7(\alpha^{12} + \alpha^{11} + \alpha^9 + \alpha^8) \\
&\quad + a_5(\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2) \\
&\quad + a_4(\alpha^{12}) + a_0\alpha^0 \\
&= a_0\alpha^0 + a_9\alpha^1 + a_5\alpha^2 + \alpha^3 + \alpha^4 + \alpha^5
\end{aligned}$$

α^{12}	α^{11}	α^{10}	α^9	α^8	α^7	α^6	α^5	α^4	α^3	α^2	α^1	α^0
a_{10}	a_7	a_{10}	a_9	a_7	a_9	a_{10}	a_5	a_{10}	a_9	a_5	a_9	a_0
a_7			a_7			a_5			a_5			
a_4												

图 13

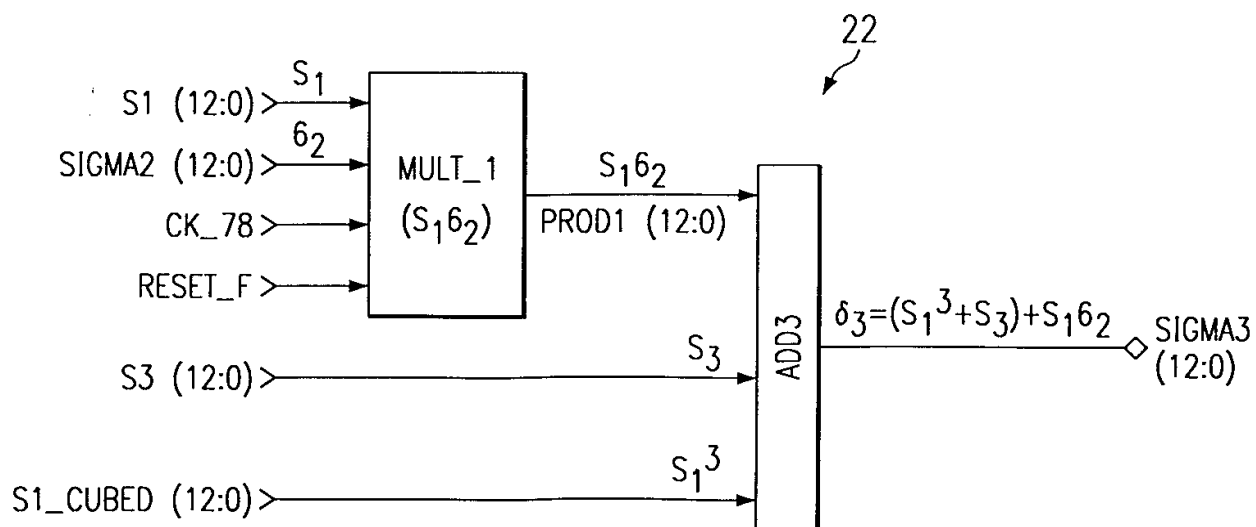


图 14

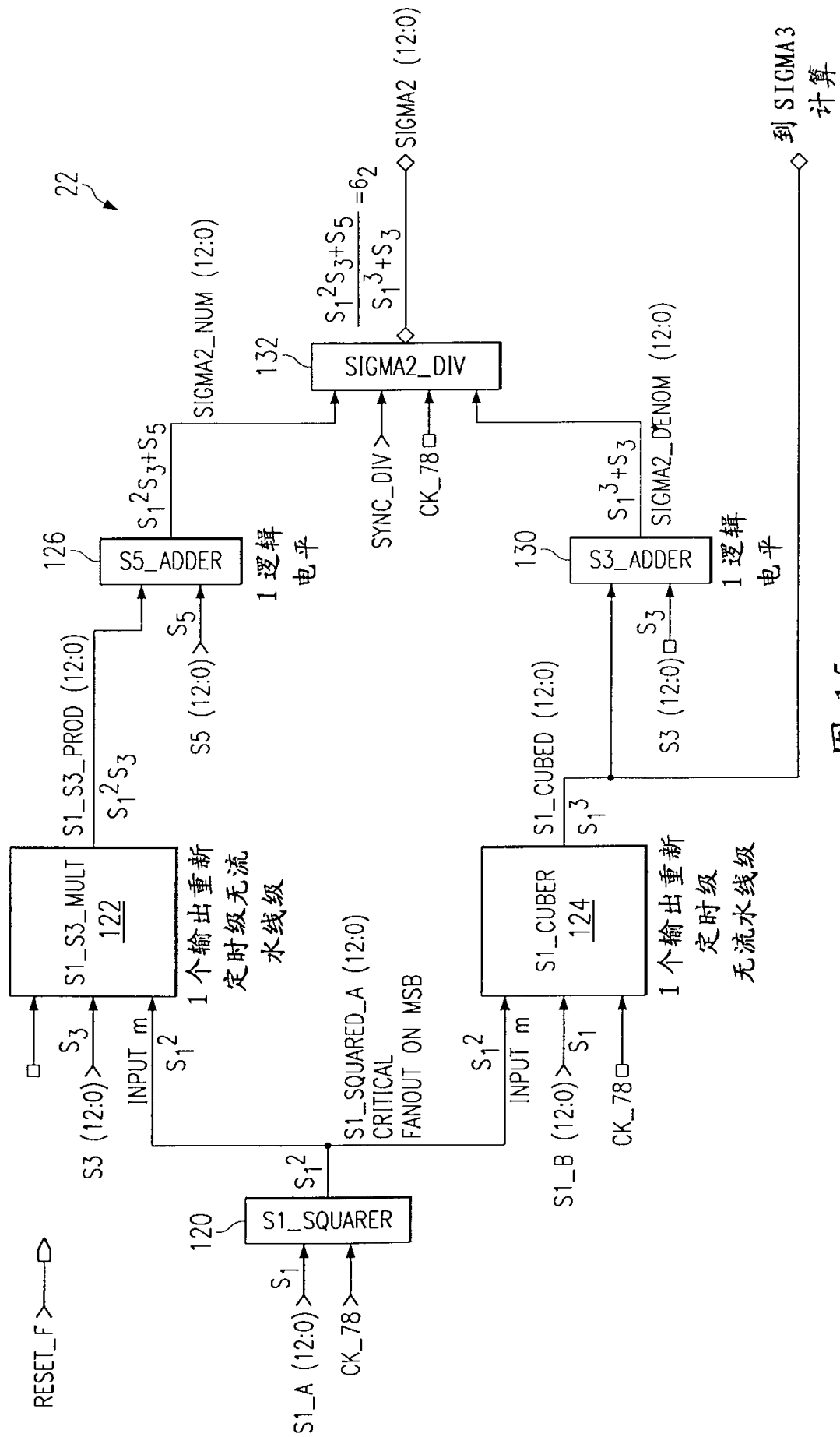


图 15

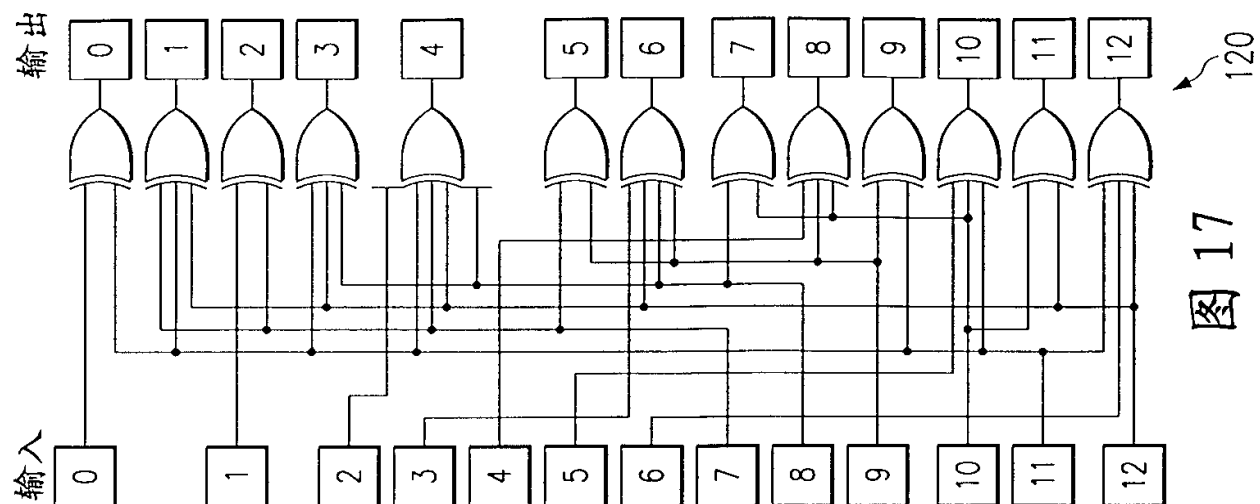


图 17

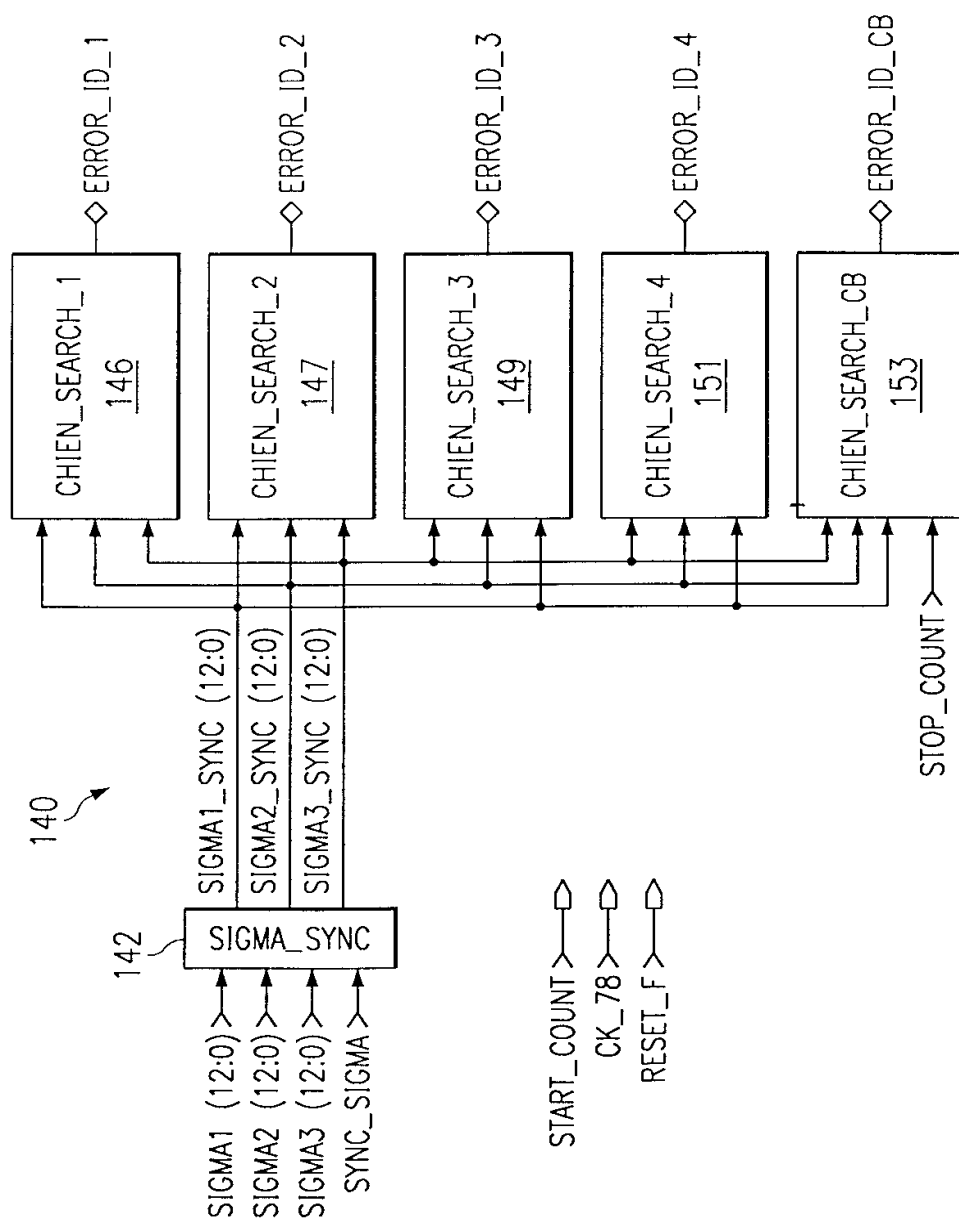


图 16A

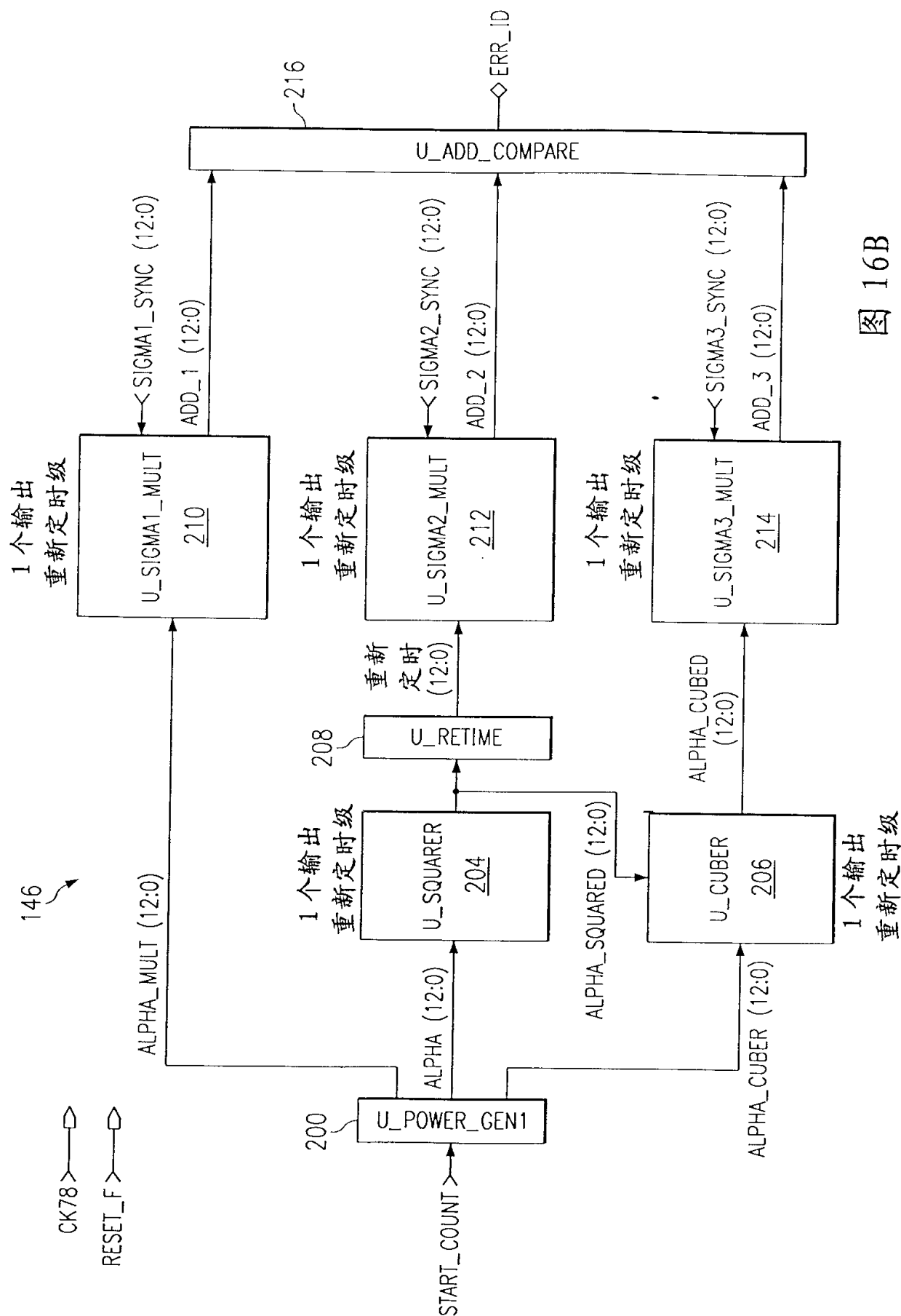


图 16B

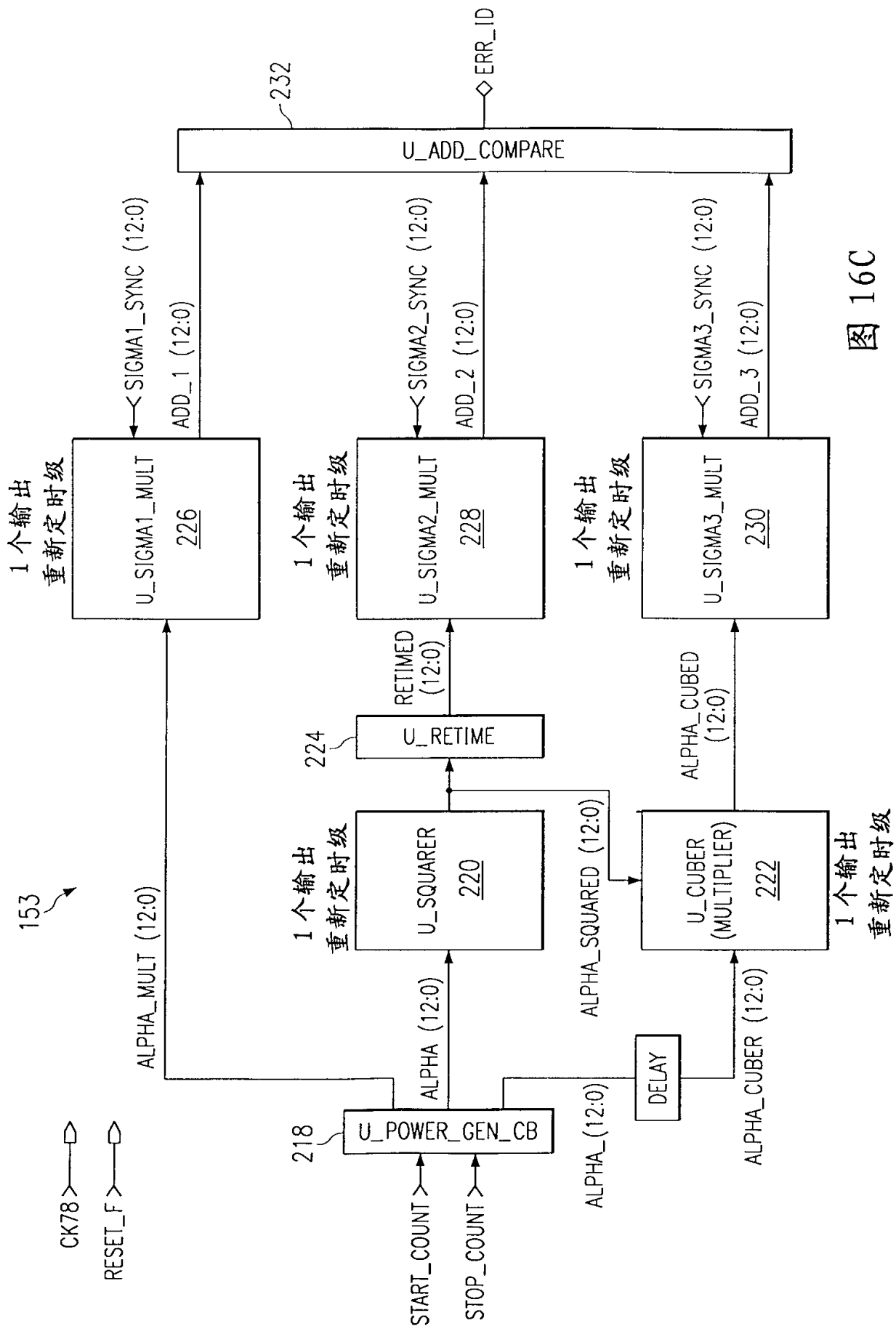


图 16C

GF (2¹³) 向量可写为

$$A = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4 + a_5\alpha^5 + a_6\alpha^6 + a_7\alpha^7 + a_8\alpha^8 + a_9\alpha^9 + a_{10}\alpha^{10} + a_{11}\alpha^{11} + a_{12}\alpha^{12}$$

$$A^2 = a_0 + a_1\alpha^2 + a_2\alpha^4 + a_3\alpha^6 + a_4\alpha^8 + a_5\alpha^{10} + a_6\alpha^{12} + a_7\alpha^{14} + a_8\alpha^{16} + a_9\alpha^{18} + a_{10}\alpha^{20} + a_{11}\alpha^{22} + a_{12}\alpha^{24}$$

需扩展 α^{14} , α^{16} , α^{18} , α^{20} , α^{22} , α^{24} 项,
将公式分为 C, D, E, F, G, H, I, 可得到

$$C = a_0 + a_1\alpha^2 + a_2\alpha^4 + a_3\alpha^6 + a_4\alpha^8 + a_5\alpha^{10} + a_6\alpha^{12}$$

$$D = a_7\alpha^{14} = a_7(\alpha + \alpha^2 + \alpha^4 + \alpha^5)$$

$$E = a_8\alpha^{16} = a_8(\alpha^3 + \alpha^4 + \alpha^6 + \alpha^7)$$

$$F = a_9\alpha^{18} = a_9(\alpha^5 + \alpha^6 + \alpha^8 + \alpha^9)$$

$$G = a_{10}\alpha^{20} = a_{10}(\alpha^7 + \alpha^8 + \alpha^{10} + \alpha^{11})$$

$$H = a_{11}\alpha^{22} = a_{11}(\alpha^0 + \alpha^1 + \alpha^3 + \alpha^4 + \alpha^9 + \alpha^{10} + \alpha^{11})$$

$$I = a_{12}\alpha^{24} = a_{12}(\alpha^1 + \alpha^3 + \alpha^4 + \alpha^6 + \alpha^{11} + \alpha^{12})$$

利用 GF (2¹³) 的特性进行扩展

通知已知 $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_{12}$, 为要平方的数的输入位,
以及 $\alpha^1, \alpha^2, \alpha^3, \dots, \alpha^{12}$, 为输出的平方结果的位位置,

输出结果为 $C + D + E + F + G + H + I$,
简化逻辑后, 得到:

$$\begin{aligned} O(0) &= I(0) + I(11) \\ O(1) &= I(7) + I(11) + I(12) \\ O(2) &= I(1) + I(7) \\ O(3) &= I(8) + I(11) + I(12) \\ O(4) &= I(2) + I(7) + I(8) + I(11) + I(12) \\ O(5) &= I(7) + I(9) \\ O(6) &= I(3) + I(8) + I(9) + I(12) \\ O(7) &= I(8) + I(10) \\ O(8) &= I(4) + I(9) + I(10) \\ O(9) &= I(9) + I(11) \\ O(10) &= I(5) + I(10) + I(11) \\ O(11) &= I(10) + I(12) \\ O(12) &= I(6) + I(11) + I(12) \end{aligned}$$

图 18

```

TERMS(0) <= (A(0) AND B(0));
TERMS(1) <= (A(0) AND B(1)) XOR (A(1) AND B(0));
TERMS(2) <= (A(0) AND B(2)) XOR (A(1) AND B(1)) XOR (A(2) AND B(0));
TERMS(3) <= (A(0) AND B(3)) XOR (A(1) AND B(2)) XOR (A(2) AND B(1)) XOR (A(3) AND B(0));
TERMS(4) <= (A(0) AND B(4)) XOR (A(1) AND B(3)) XOR (A(2) AND B(2)) XOR (A(3) AND B(1))
XOR (A(4) AND B(0));
TERMS(5) <= (A(0) AND B(5)) XOR (A(1) AND B(4)) XOR (A(2) AND B(3)) XOR (A(3) AND B(2))
XOR (A(4) AND B(1)) XOR (A(5) AND B(0));
TERMS(6) <= (A(0) AND B(6)) XOR (A(1) AND B(5)) XOR (A(2) AND B(4)) XOR (A(3) AND B(3))
XOR (A(4) AND B(2)) XOR (A(5) AND B(1)) XOR (A(6) AND B(0));
TERMS(7) <= (A(0) AND B(7)) XOR (A(1) AND B(6)) XOR (A(2) AND B(5)) XOR (A(3) AND B(4))
XOR (A(4) AND B(3)) XOR (A(5) AND B(2)) XOR (A(6) AND B(1)) XOR (A(7) AND B(0));
TERMS(8) <= (A(0) AND B(8)) XOR (A(1) AND B(7)) XOR (A(2) AND B(6)) XOR (A(3) AND B(5))
XOR (A(4) AND B(4)) XOR (A(5) AND B(3)) XOR (A(6) AND B(2)) XOR (A(7) AND B(1))
XOR (A(8) AND B(0));
TERMS(9) <= (A(0) AND B(9)) XOR (A(1) AND B(8)) XOR (A(2) AND B(7)) XOR (A(3) AND B(6))
XOR (A(4) AND B(5)) XOR (A(5) AND B(4)) XOR (A(6) AND B(3)) XOR (A(7) AND B(2))
XOR (A(8) AND B(1)) XOR (A(9) AND B(0));
TERMS(10) <= (A(0) AND B(10)) XOR (A(1) AND B(9)) XOR (A(2) AND B(8)) XOR (A(3) AND B(7))
XOR (A(4) AND B(6)) XOR (A(5) AND B(5)) XOR (A(6) AND B(4)) XOR (A(7) AND B(3))
XOR (A(8) AND B(2)) XOR (A(9) AND B(1)) XOR (A(10) AND B(0));
TERMS(11) <= (A(0) AND B(11)) XOR (A(1) AND B(10)) XOR (A(2) AND B(9)) XOR (A(3) AND B(8))
XOR (A(4) AND B(7)) XOR (A(5) AND B(6)) XOR (A(6) AND B(5)) XOR (A(7) AND B(4))
XOR (A(8) AND B(3)) XOR (A(9) AND B(2)) XOR (A(10) AND B(1)) XOR (A(11) AND B(0));
TERMS(12) <= (A(0) AND B(12)) XOR (A(1) AND B(11)) XOR (A(2) AND B(10)) XOR (A(3) AND B(9))
XOR (A(4) AND B(8)) XOR (A(5) AND B(7)) XOR (A(6) AND B(6)) XOR (A(7) AND B(5))
XOR (A(8) AND B(4)) XOR (A(9) AND B(3)) XOR (A(10) AND B(2)) XOR (A(11) AND B(1))
XOR (A(12) AND B(0));
TERMS(13) <= (A(1) AND B(12)) XOR (A(2) AND B(11)) XOR (A(3) AND B(10)) XOR (A(4) AND B(9))
XOR (A(5) AND B(8)) XOR (A(6) AND B(7)) XOR (A(7) AND B(6)) XOR (A(8) AND B(5))
XOR (A(9) AND B(4)) XOR (A(10) AND B(3)) XOR (A(11) AND B(2)) XOR (A(12) AND B(1));
TERMS(14) <= (A(2) AND B(12)) XOR (A(3) AND B(11)) XOR (A(4) AND B(10)) XOR (A(5) AND B(9))
XOR (A(6) AND B(8)) XOR (A(7) AND B(7)) XOR (A(8) AND B(6)) XOR (A(9) AND B(5))
XOR (A(10) AND B(4)) XOR (A(11) AND B(3)) XOR (A(12) AND B(2));
TERMS(15) <= (A(3) AND B(12)) XOR (A(4) AND B(11)) XOR (A(5) AND B(10)) XOR (A(6) AND B(9))
XOR (A(7) AND B(8)) XOR (A(8) AND B(7)) XOR (A(9) AND B(6)) XOR (A(10) AND B(5))
XOR (A(11) AND B(4)) XOR (A(12) AND B(3));
TERMS(16) <= (A(4) AND B(12)) XOR (A(5) AND B(11)) XOR (A(6) AND B(10)) XOR (A(7) AND B(9))
XOR (A(8) AND B(8)) XOR (A(9) AND B(7)) XOR (A(10) AND B(6)) XOR (A(11) AND B(5))
XOR (A(12) AND B(4));
TERMS(17) <= (A(5) AND B(12)) XOR (A(6) AND B(11)) XOR (A(7) AND B(10)) XOR (A(8) AND B(9))
XOR (A(9) AND B(8)) XOR (A(10) AND B(7)) XOR (A(11) AND B(6)) XOR (A(12) AND B(5));
TERMS(18) <= (A(6) AND B(12)) XOR (A(7) AND B(11)) XOR (A(8) AND B(10)) XOR (A(9) AND B(9))
XOR (A(10) AND B(8)) XOR (A(11) AND B(7)) XOR (A(12) AND B(6));
TERMS(19) <= (A(7) AND B(12)) XOR (A(8) AND B(11)) XOR (A(9) AND B(10)) XOR (A(10) AND B(9))
XOR (A(11) AND B(8)) XOR (A(12) AND B(7));
TERMS(20) <= (A(8) AND B(12)) XOR (A(9) AND B(11)) XOR (A(10) AND B(10)) XOR (A(11) AND B(9))
XOR (A(12) AND B(8));
TERMS(21) <= (A(9) AND B(12)) XOR (A(10) AND B(11)) XOR (A(11) AND B(10)) XOR (A(12) AND B(9));
TERMS(22) <= (A(10) AND B(12)) XOR (A(11) AND B(11)) XOR (A(12) AND B(10));
TERMS(23) <= (A(11) AND B(12)) XOR (A(12) AND B(11));
TERMS(24) <= (A(12) AND B(12));

```

```

RESULT_M(0) <= TERMS(0) XOR TERMS(13) XOR TERMS(22) XOR TERMS(23);
RESULT_M(1) <= TERMS(1) XOR TERMS(13) XOR TERMS(14) XOR TERMS(22) XOR TERMS(24);
RESULT_M(2) <= TERMS(2) XOR TERMS(14) XOR TERMS(15) XOR TERMS(23);
RESULT_M(3) <= TERMS(3) XOR TERMS(13) XOR TERMS(15) XOR TERMS(16) XOR TERMS(22) XOR TERMS(23) XOR TERMS(24);
RESULT_M(4) <= TERMS(4) XOR TERMS(13) XOR TERMS(14) XOR TERMS(16) XOR TERMS(17) XOR TERMS(22) XOR TERMS(24);
RESULT_M(5) <= TERMS(5) XOR TERMS(14) XOR TERMS(15) XOR TERMS(17) XOR TERMS(18) XOR TERMS(23);
RESULT_M(6) <= TERMS(6) XOR TERMS(15) XOR TERMS(16) XOR TERMS(18) XOR TERMS(19) XOR TERMS(24);
RESULT_M(7) <= TERMS(7) XOR TERMS(16) XOR TERMS(17) XOR TERMS(19) XOR TERMS(20);
RESULT_M(8) <= TERMS(8) XOR TERMS(17) XOR TERMS(18) XOR TERMS(20) XOR TERMS(21);
RESULT_M(9) <= TERMS(9) XOR TERMS(18) XOR TERMS(19) XOR TERMS(21) XOR TERMS(22);
RESULT_M(10) <= TERMS(10) XOR TERMS(19) XOR TERMS(20) XOR TERMS(22) XOR TERMS(23);
RESULT_M(11) <= TERMS(11) XOR TERMS(20) XOR TERMS(21) XOR TERMS(23) XOR TERMS(24);
RESULT_M(12) <= TERMS(12) XOR TERMS(21) XOR TERMS(22) XOR TERMS(24);

```

$$\begin{aligned}
x^0 &= x^9 + I_0 \\
x^1 &= x^{10} + x^9 + I_1 \\
x^2 &= x^{10} + x^{11} + I_2 \\
x^3 &= x^{12} + x^{11} + x^9 + I_3 \\
x^4 &= x^{12} + x^{10} + x^9 + x^0 \\
x^5 &= x^{10} + x^{11} + x \\
x^6 &= x^{12} + x^{11} + x^2 \\
x^7 &= x^{12} + x^3 \\
x^8 &= x^4 \\
x^9 &= x^5 \\
x^{10} &= x^6 \\
x^{11} &= x^7 \\
x^{12} &= x^8
\end{aligned}$$

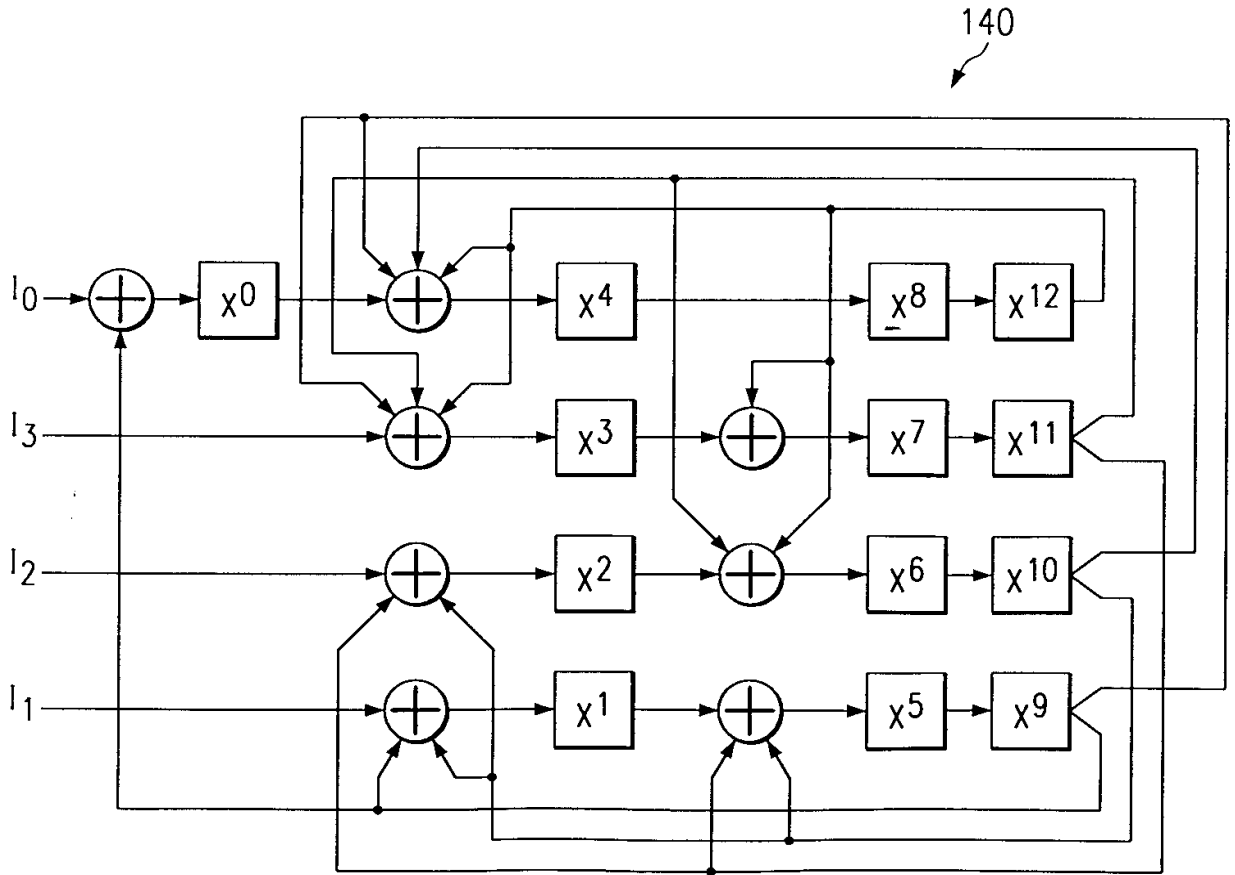


图 20

$$x^0 = x^9 + x^{12} + I_0$$

$$x^1 = x^{10} + I_1$$

$$x^2 = x^{11} + I_2$$

$$x^3 = x^{12} + I_3$$

$$x^4 = x^9 + x^{12} + x^0$$

$$x^5 = x^9 + x^{10} + x^{12} + x^1$$

$$x^6 = x^{10} + x^{11} + x^2$$

$$x^7 = x^9 + x^{11} + x^3$$

$$x^8 = x^{10} + x^{12} + x^4$$

$$x^9 = x^9 + x^{11} + x^{12} + x^5$$

$$x^{10} = x^9 + x^{10} + x^6$$

$$x^{11} = x^{10} + x^{11} + x^7$$

$$x^{12} = x^{11} + x^{12} + x^8$$

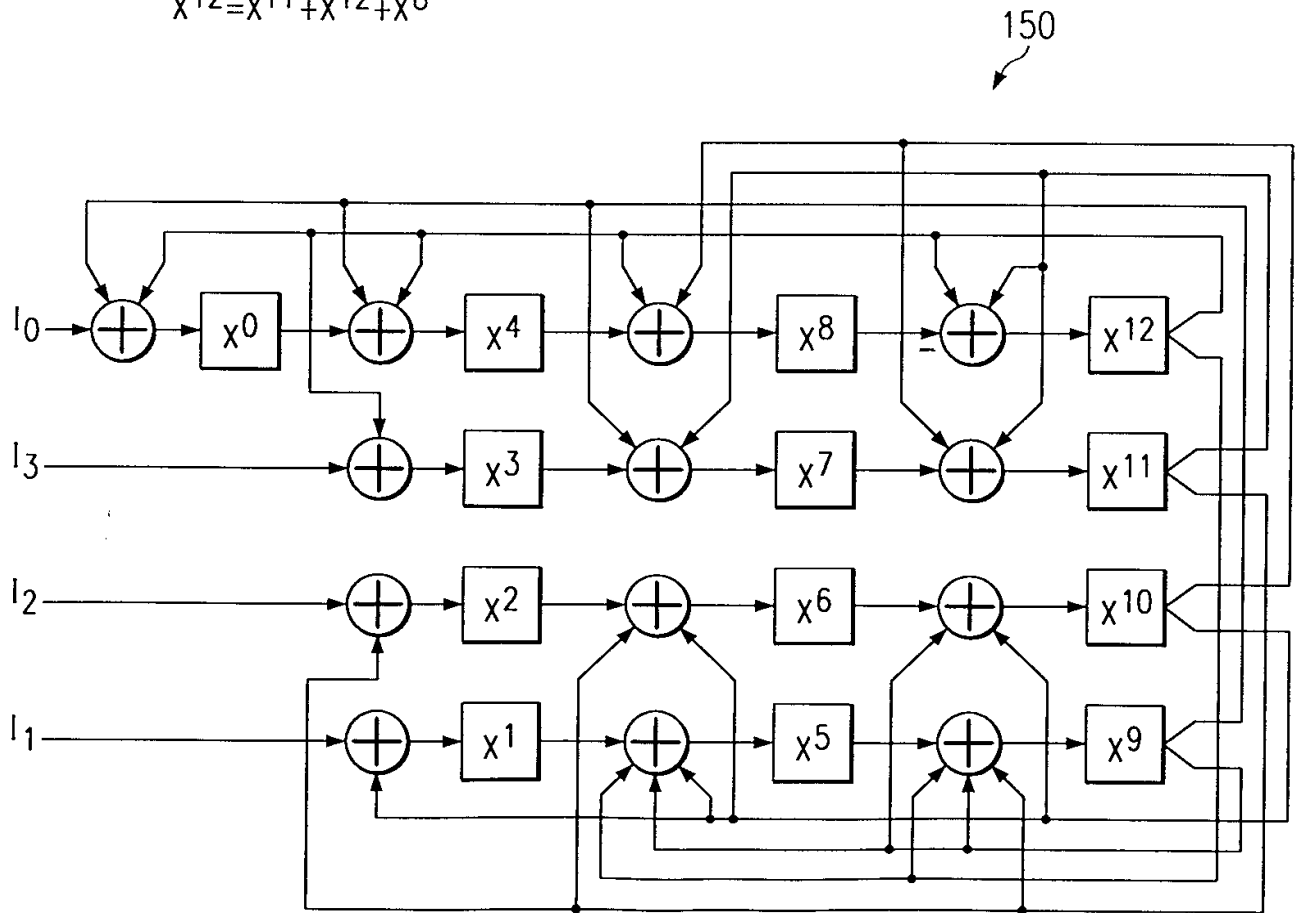


图 21

$$\begin{aligned}
x^0 &= x^9 + x^{11} + l_0 \\
x^1 &= x^9 + x^{10} + x^{11} + x^{12} + l_1 \\
x^2 &= x^{10} + x^{11} + x^{12} + l_2 \\
x^3 &= x^{11} + x^{12} + l_3 \\
x^4 &= x^9 + x^{11} + x^{12} + x^0 \\
x^5 &= x^{10} + x^{12} + x^1 \\
x^6 &= x^{11} + x^2 \\
x^7 &= x^9 + x^{11} + x^{12} + x^3 \\
x^8 &= x^9 + x^{10} + x^{11} + x^{12} + x^4 \\
x^9 &= x^{10} + x^{11} + x^{12} + x^5 \\
x^{10} &= x^{11} + x^{12} + x^6 \\
x^{11} &= x^9 + x^{11} + x^{12} + x^7 \\
x^{12} &= x^{10} + x^{12} + x^8
\end{aligned}$$

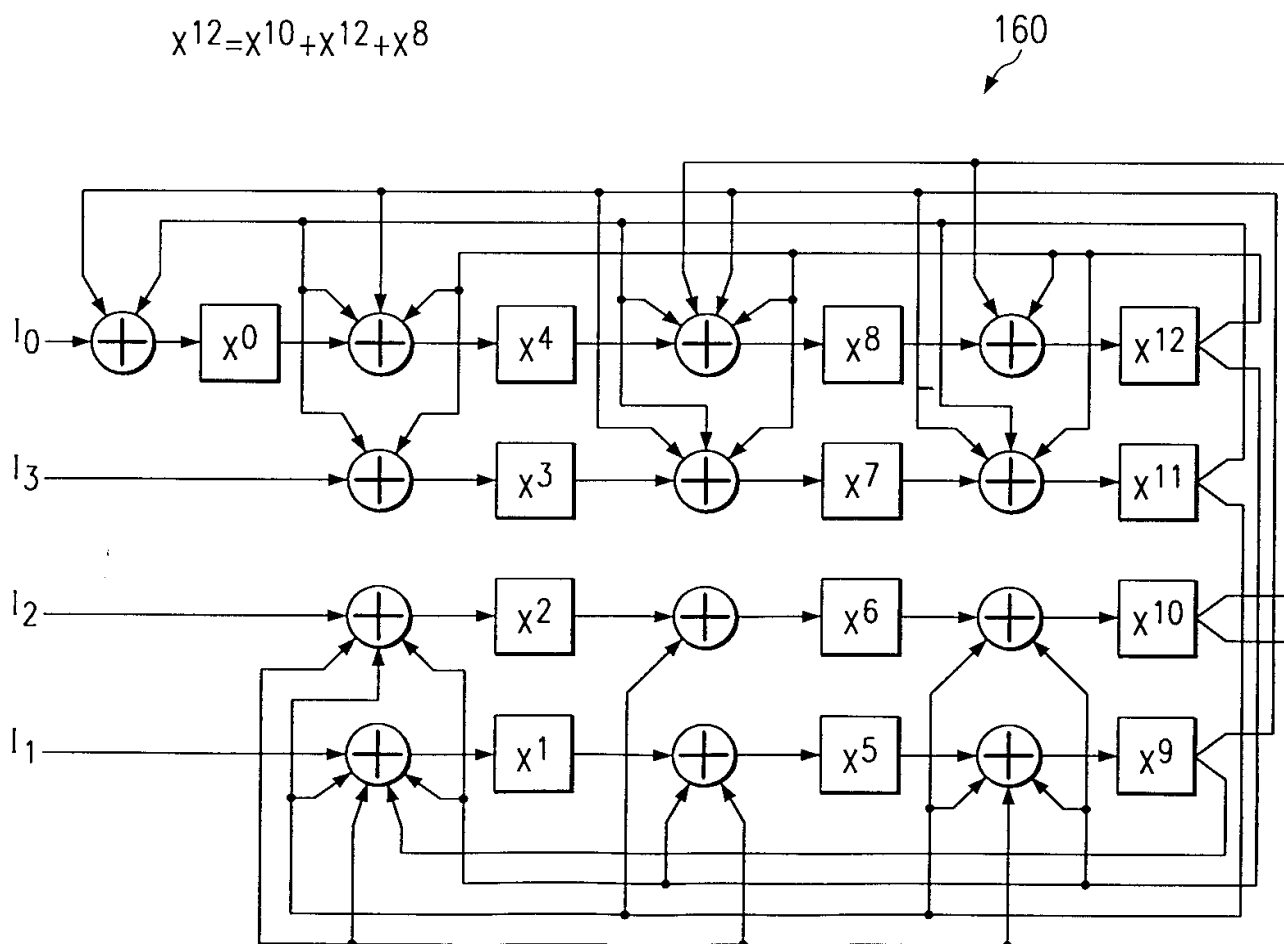


图 22

到图 23B

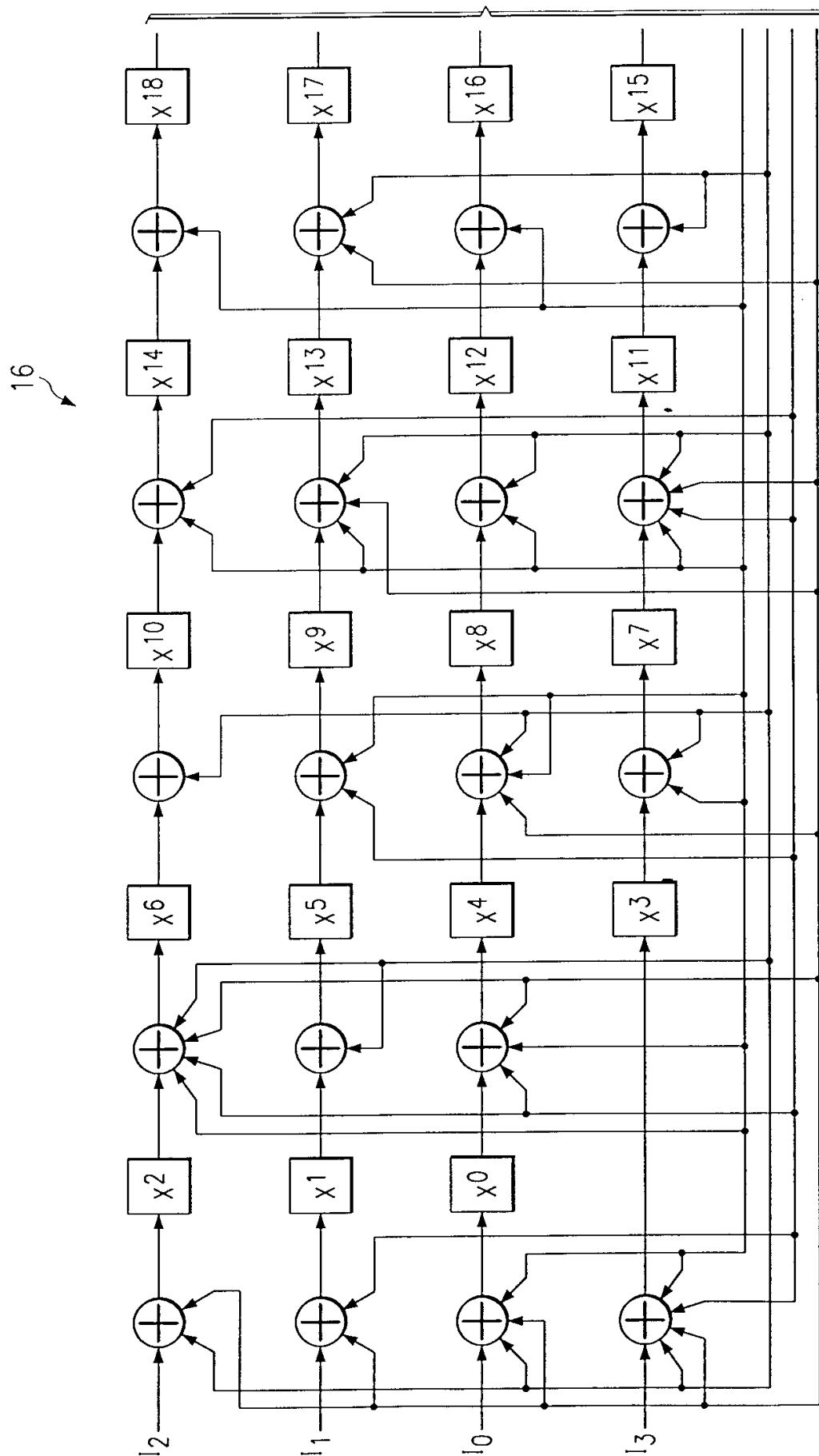


图 23A

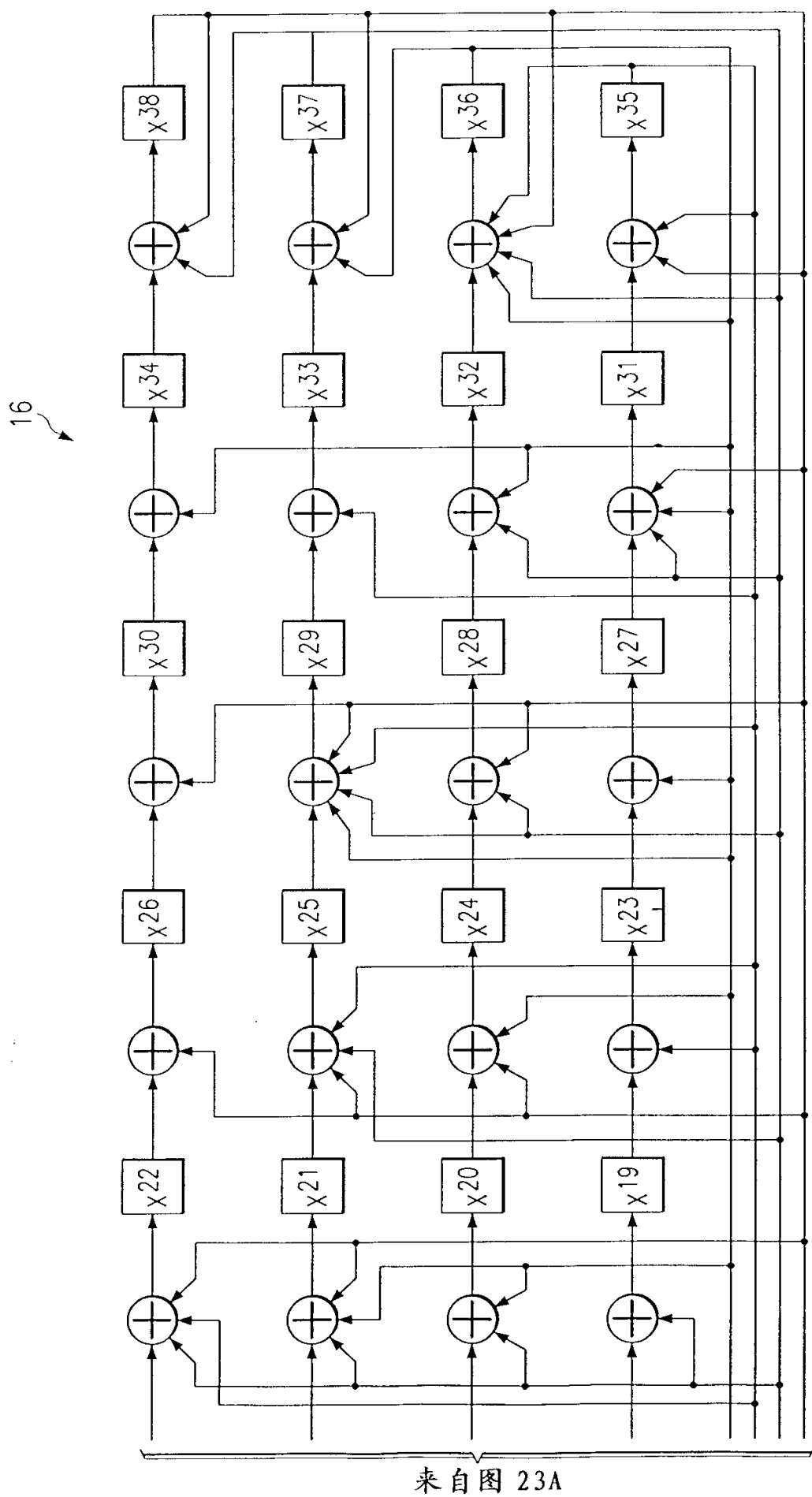
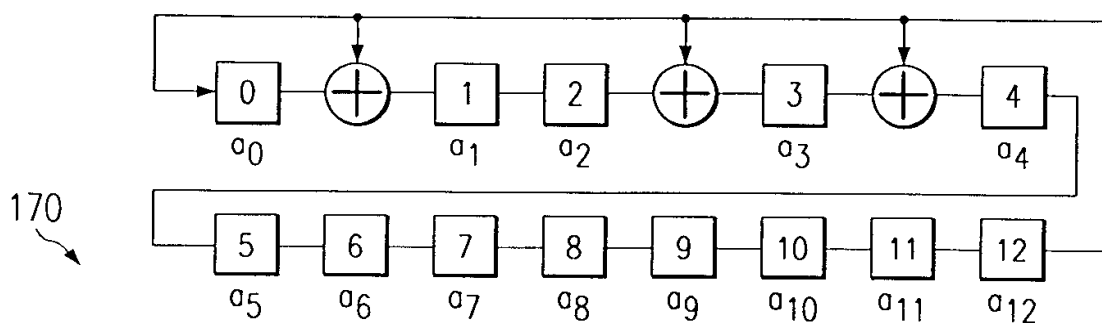


图 23B

$$G(X)=G1(X) \cdot G3(X) \cdot G5(X)$$

$$\begin{aligned} x^0 &= x^{35} + x^{37} + x^{38} + I_0 & x^{20} &= x^{35} + x^{37} + x^{16} \\ x^1 &= x^{36} + x^{38} + I_1 & x^{21} &= x^{35} + x^{36} + x^{37} + x^{17} \\ x^2 &= x^{35} + x^{38} + I_2 & x^{22} &= x^{36} + x^{37} + x^{38} + x^{18} \\ x^3 &= x^{35} + x^{36} + x^{37} + x^{38} + I_3 & x^{23} &= x^{35} + x^{19} \\ x^4 &= x^{36} + x^{37} + x^{38} + x^0 & x^{24} &= x^{35} + x^{36} + x^{37} + x^{38} + x^{20} \\ x^5 &= x^{35} + x & x^{25} &= x^{36} + x^{37} + x^{38} + x^{21} \\ x^6 &= x^{35} + x^{36} + x^{37} + x^{38} + x^2 & x^{26} &= x^{35} + x^{22} \\ x^7 &= x^{35} + x^{36} + x^3 & x^{27} &= x^{36} + x^{23} \\ x^8 &= x^{35} + x^{36} + x^{38} + x^4 & x^{28} &= x^{35} + x^{38} + x^{24} \\ x^9 &= x^{36} + x^{37} + x^5 & x^{29} &= x^{35} + x^{36} + x^{37} + x^{38} + x^{25} \\ x^{10} &= x^{35} + x^6 & x^{30} &= x^{35} + x^{36} + x^{26} \\ x^{11} &= x^{35} + x^{36} + x^{37} + x^{38} + x^7 & x^{31} &= x^{35} + x^{36} + x^{38} + x^{27} \\ x^{12} &= x^{35} + x^{36} + x^8 & x^{32} &= x^{36} + x^{37} + x^{28} \\ x^{13} &= x^{35} + x^{36} + x^{38} + x^9 & x^{33} &= x^{35} + x^{29} \\ x^{14} &= x^{36} + x^{37} + x^{10} & x^{34} &= x^{36} + x^{30} \\ x^{15} &= x^{35} + x^{11} & x^{35} &= x^{35} + x^{38} + x^{31} \\ x^{16} &= x^{36} + x^{12} & x^{36} &= x^{35} + x^{36} + x^{37} + x^{38} + x^{32} \\ x^{17} &= x^{35} + x^{38} + x^{13} & x^{37} &= x^{35} + x^{36} + x^{33} \\ x^{18} &= x^{36} + x^{14} & x^{38} &= x^{36} + x^{37} + x^{34} \\ x^{19} &= x^{37} + x^{15} \end{aligned}$$

图 24



$$\begin{aligned}
 t_1 &= a_0 = a_{12} & a_5 &= a_4 & a_{10} &= a_9 \\
 a_1 &= a_{12} + a_0 & a_6 &= a_5 & a_{11} &= a_{10} \\
 a_2 &= a_1 & a_7 &= a_6 & a_{12} &= a_{11} \\
 a_3 &= a_{12} + a_2 & a_8 &= a_7 \\
 a_4 &= a_{12} + a_3 & a_9 &= a_8
 \end{aligned}$$

$$\begin{aligned}
 t_2 &= a_0 = a_{11} & a_5 &= a_{12} + a_3 & a_{10} &= a_8 \\
 a_1 &= a_{11} + a_{12} & a_6 &= a_4 & a_{11} &= a_9 \\
 a_2 &= a_{12} + a_0 & a_7 &= a_5 & a_{12} &= a_{10} \\
 a_3 &= a_{11} + a_1 & a_8 &= a_6 \\
 a_4 &= a_{11} + a_{12} + a_2 & a_9 &= a_7
 \end{aligned}$$

$$\begin{aligned}
 t_3 &= a_0 = a_{10} & a_5 &= a_{11} + a_{12} + a_2 & a_{10} &= a_7 \\
 a_1 &= a_{10} + a_{11} & a_6 &= a_{12} + a_3 & a_{11} &= a_8 \\
 a_2 &= a_{11} + a_{12} & a_7 &= a_4 & a_{12} &= a_9 \\
 a_3 &= a_{10} + a_{12} + a_0 & a_8 &= a_5 \\
 a_4 &= a_{10} + a_{11} + a_1 & a_9 &= a_6
 \end{aligned}$$

$$\begin{aligned}
 t_4 &= a_0 = a_9 & a_5 &= a_{10} + a_{11} + a_1 & a_{10} &= a_6 \\
 a_1 &= a_9 + a_{10} & a_6 &= a_{11} + a_{12} + a_2 & a_{11} &= a_7 \\
 a_2 &= a_{10} + a_{11} & a_7 &= a_{12} + a_{13} & a_{12} &= a_8 \\
 a_3 &= a_9 + a_{11} + a_{12} & a_8 &= a_4 \\
 a_4 &= a_9 + a_{10} + a_{12} + a_0 & a_9 &= a_5
 \end{aligned}$$

注意, $a_8, a_{10}, a_{11}, a_{12}$ 被延迟

GEN 1 COUNTS	0, 4, 8, 12 ...	START	1 0 0 0 0 0 0 0 0 0 0 0 0 0
GEN 2 COUNTS	1, 5, 9, 13 ...	START	0 0 0 0 0 0 0 0 0 0 0 0 0 1
GEN 3 COUNTS	2, 6, 10, 14 ...	START	0 0 0 0 0 0 0 0 0 0 0 0 1 0
GEN 4 COUNTS	3, 7, 11, 15 ...	START	0 0 0 0 0 0 0 0 0 0 0 1 0 0

向后走, 相同方式

计数 1 开始

8191-4320=3871 START

图 25

architecture BEHAVIORAL of POWER_GEN is

signal LFSR_REVERSE: std_logic_vector(12 downto 0);

signal ALPHA_RET1: std_logic_vector(12 downto 0);

signal ALPHA_RET2: std_logic_vector(12 downto 0);

begin

ALPHA <= LFSR_REVERSE after 1 ns;

-- LFSR counting up from 8153 to 8191

process (RESET_F, CK_78)

begin

if RESET_F = '0' then

LFSR_REVERSE <= "0000000000001";

elsif (CK_78' event and CK_78='1') then

if START_COUNT='1' then

LFSR_REVERSE <= "0000000000001"; --8153 for 39 checkbits

elsif STOP_COUNT='1' then

LFSR_REVERSE <= (others => '0');

else

LFSR_REVERSE(12) <= LFSR_REVERSE(0);

LFSR_REVERSE(11) <= LFSR_REVERSE(12);

LFSR_REVERSE(10) <= LFSR_REVERSE(11);

LFSR_REVERSE(9) <= LFSR_REVERSE(10);

LFSR_REVERSE(8) <= LFSR_REVERSE(9);

LFSR_REVERSE(7) <= LFSR_REVERSE(8);

LFSR_REVERSE(6) <= LFSR_REVERSE(7);

LFSR_REVERSE(5) <= LFSR_REVERSE(6);

LFSR_REVERSE(4) <= LFSR_REVERSE(5);

LFSR_REVERSE(3) <= LFSR_REVERSE(4) xor LFSR_REVERSE(0);

LFSR_REVERSE(2) <= LFSR_REVERSE(3) xor LFSR_REVERSE(0);

LFSR_REVERSE(1) <= LFSR_REVERSE(2);

LFSR_REVERSE(0) <= LFSR_REVERSE(1) xor LFSR_REVERSE(0);

end if;

end if;

end process;

-- Retime stages to sync up with squarer and cuber

process (RESET_F, CK_78)

begin

if RESET_F='0' then

ALPHA_RET1 <= (others => '0');

ALPHA_RET2 <= (others => '0');

elsif CK_78' EVENT AND ck_78='1' then

ALPHA_RET1 <= ALPHA;

ALPHA_RET2 <= ALPHA_RET1;

end if;

end process;

-- Selecting appropriate bits for cuber and squarer.

ALPHA_CUBER <= ALPHA_RET1;

ALPHA_MULT <= ALPHA_RET2;

end;

180

图 26

