本文引用格式：代苏杰,杨定坤,阳江平,等. JESD204C 同步算法研究[J].自动化与信息工程,2025,46(2):18-24.
DAI Sujie, YANG Dingkun, YANG Jiangping, et al. Research on JESD204C synchronization algorithm[J]. Automation & Information Engineering, 2025,46(2):18-24.

JESD204C 同步算法研究

代苏杰　杨定坤　阳江平　耿建强
（成都振芯科技股份有限公司，四川 成都 610000）

摘要：JESD204B 接口协议的链路同步是在建链之初基于特殊码字完成，建链完成后，同步过程即结束；而 JESD204C 接口协议是依据链路实时传输的同步码来实现同步，且同步检测不仅在初始建链时起作用，在建链完成后也会实时跟踪，确认链路是否正常。首先，比较 JESD204C、JESD204B 接口协议的同步差异；然后，在解析 JESD204C 接口协议同步理论的基础上，提出一种并行同步头查找算法；最后，基于 MATLAB 编写算法仿真代码，仿真结果表明该算法可行有效。

关键词：JESD204C 接口协议；JESD204B 接口协议；同步算法；并行同步头查找；链路同步
中图分类号：TN914.3; TP302.1　　**文献标志码：**A　　**文章编号：**1674-2605(2025)02-0003-07
DOI：10.12475/aie.20250203　　开放获取

Research on JESD204C Synchronization Algorithm

DAI Sujie　YANG Dingkun　YANG Jiangping　GEN Jianqiang
(Chengdu Corpro Technology Co., Ltd., Chengdu 610000, China)

Abstract: The link synchronization of JESD204B interface protocol is completed based on special codewords at the beginning of link building, and the synchronization process ends after the link building is completed; The JESD204C interface protocol achieves synchronization based on the synchronization code transmitted in real-time by the link, and synchronization detection not only plays a role during the initial link establishment, but also tracks in real-time after the link establishment is completed to confirm whether the link is normal. Firstly, compare the synchronization differences between JESD204C and JESD204B interface protocols; Then, based on the analysis of JESD204C interface protocol synchronization theory, a parallel synchronization header lookup algorithm is proposed; Finally, algorithm simulation code was written based on MATLAB, and the simulation results showed that the algorithm is feasible and effective.

Keywords: JESD204C interface protocol; JESD204B interface protocol; synchronization algorithm; parallel synchronization header detection; link synchronization

0　引言

随着通信技术的高速发展，传统的互补金属氧化物半导体（complementary metal oxide semiconductor, CMOS）、低电压差分信号（low voltage differential signaling, LVDS）等并行接口因布线困难，通道串扰 大[1]等问题，已无法满足高速、高精度的模拟数字转换器（analog-to-digital converter, ADC）与数字模块之间的接口需求。2011 年，联合电子设备工程委员会（joint electron device engineering council, JEDEC）发布的串行接口协议 JESD204B[2]，串行模式简洁，最高传输速率可达 12.5 Gb/s。为了满足高宽带信号处理的需求，ADC 采样率已进入 GSPS 时代，这对接口速率提出了更高的要求。如 ADC 采样率为 1 Gs/s，精度为 16 bit 时，接口所需的传输速率为 16 Gb/s。此时，JESD204B 接口协议的单通道速率已无法满足数据传输的需求。JEDEC 于 2017 年发布了 JESD204C 接口协议，将接口传输速率提升至 32 Gb/s，且有效数据传输效率由 JESD204B 的 80.0%提高到 96.9%[3]。相比其他接口协

议，基于 JESD204C 接口协议的设计通道数更少，PCB 布局更紧凑，布线策略更简化。

目前，国内对 JESD204C 接口协议的研究相对较少，且主要针对某一特定层（传输层、链路层、物理层等）进行研究，对单个技术点的深入探讨较为有限。文献[4]主要对物理层进行分析；文献[5]主要对链路层进行分析，关于同步的分析仅限于状态机转移；文献[6]虽然对链路层和传输层均进行了分析，但并未对同步进行详细阐述；文献[7]主要对前向纠错（forward error correction, FEC）进行论述。

相较于 JESD204B 接口协议，JESD204C 接口协议的传输层基本保持不变，而链路层和物理层的变化较大。本文以 64B/66B 编码为例，对链路层的同步进行比较分析，并对同步头提取、扩展多块同步以及并行实现的难点进行探讨，以期为相关设计提供参考。

1　JESD204C 与 JESD204B 接口协议同步差异

JESD204C 接口协议是在 JESD204B 接口协议的基础上演进而来的，但这两个接口协议的同步方式并没有继承或兼容的关系，其差异如表 1 所示。

表 1　JESD204B、JESD204C 接口协议同步差异

对比项	JESD204B	JESD204C
同步信号	8 bit 特殊码	2 bit:01/10
同步信号发送条件	SYNC 拉低	无
同步信号发送间隔	0 bit	66 bit
同步信号发送时间	建链初	链路始终
边界判断	8 bit	66 bit
同步对应 1	符号同步	块同步
同步对应 2	多帧同步	扩展多块同步

JESD204B 接口协议的同步信号是一串特殊的 8 bit 码字。当接收端将 SYNC 信号拉低时，发送端会依次发出如图 1 所示的同步数据，其中的 K、R、A、Q 均为同步所需的特殊码字；接收端根据这些特殊码字进行符号同步和多帧同步。JESD204B 接口协议的同步流程在文献[8-9]中已有描述，此处不再赘述。

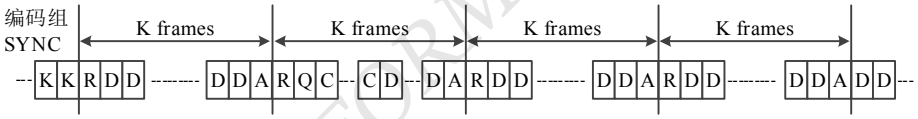

图 1　JESD204B 初始接入序列

JESD204C 接口协议的同步信号贯穿链路传输始末。发送端在每 66 bit 的周期内插入一个 2 bit 的同步头；接收端则根据同步头的跳变特性进行同步接入。

JESD204B 接口协议采用 8B/10B 编码，数据以字节为单位进行处理，因此在符号同步时需做到字节对齐，即从接收的 bit 串中识别出 8 bit 数据的最高有效位（MSB）或最低有效位（LSB）。符号同步后，再对齐多帧的首尾，从而实现多帧同步及链路接入。

JESD204C 接口协议采用 64B/66B 编码，数据以块为单位进行处理，块同步时对齐宽度为 66 bit。块同步后，系统判断扩展多块标志位是否有效，若有效，扩展多块同步完成，链路接入。

2　JESD204C 同步解析

JESD204C 接口协议用块同步和扩展多块同步分别替代 JESD204B 接口协议的符号同步和多帧同步。

将发送端加扰的 64 bit 数据与 2 bit 的同步头信息（SH）组成一个 64B/66B 编码码块。32 个编码码块组成一个多块，此多块包含 32 个同步头信息。数个多块组成一个扩展多块。块、多块、扩展多块的组成结构如图 2 所示。

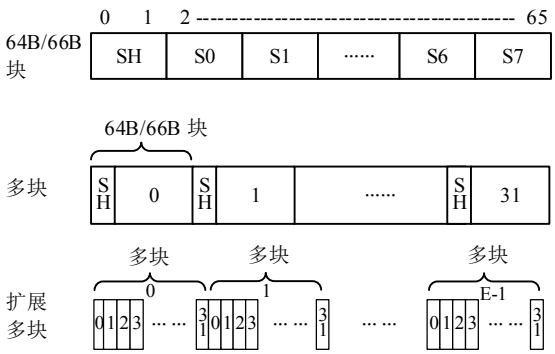

图 2　块、多块、扩展多块的组成结构图

2 bit 的同步头信息由 1 bit 编码而来，其编码规则如表 2 所示。

表 2　同步头编码规则

同步头待编码 bit	同步头编码 bit[1:0]
0	01
1	10

一个多块的32 个同步头信息除了用于块同步外，还可以向接收端传递用户信息，如 FEC 信息、命令提示符信息、其他自定义信息等，具体格式如表 3 所示。

表 3　多块同步头信息表

bit	function	bit	function	bit	function	bit	function
0	INFO[25]	8	INFO[17]	16	INFO[9]	24	INFO[2]
1	INFO[24]	9	INFO[16]	17	INFO[8]	25	INFO[1]
2	INFO[23]	10	INFO[15]	18	INFO[7]	26	INFO[0]
3	INFO[22]	11	INFO[14]	19	INFO[6]	27	0
4	INFO[21]	12	INFO[13]	20	INFO[5]	28	0
5	INFO[20]	13	INFO[12]	21	INFO[4]	29	0
6	INFO[19]	14	INFO[11]	22	EoEMB	30	0
7	INFO[18]	15	INFO[10]	23	INFO[3]	31	1

多块同步头 32 个同步头信息中的 bit[27:31]={0，0，0，0，1}为固定导频值，用于多块边界的判断；bit22 为扩展多块指示标志，用于扩展多块边界的判断；其余 26 个 bit 为用户根据需要自定义的信息 INFO[0:25]，如为提升链路质量，定义 INFO=FEC 信息，FEC 详情可参考文献[7]，此处略。

接收端在收到数据后，需准确提取块、多块、扩展多块的边界，其关键在于同步头查找。同步头查找流程如图 3 所示。

同步头查找的具体步骤如下：

1） 链路复位，接收计数器 rcv_bit_cnt、对齐计数器 align_bit_cnt、循环计数器 loop_cnt 清零；

2） 启动同步头搜索，接收计数器 rcv_bit_cnt 从 0~65 循环计数；

3） 当接收计数器 rcv_bit_cnt 等于对齐计数器 align_bit_cnt 时，判断当前 bit 与上一个 bit 是否存在翻转，若存在，循环计数器 loop_cnt 加 1；否则对齐计数器 align_bit_cnt 加 1，循环计数器 loop_cnt 清零；

4） 判断对齐计数器 align_bit_cnt 是否超过 65，若超过，表明无法锁定同步头，返回搜索失败指示；若在搜索期间循环计数器 loop_cnt 超过 64，则同步头锁定成功；

5） 同步头译码和扩展多块同步。

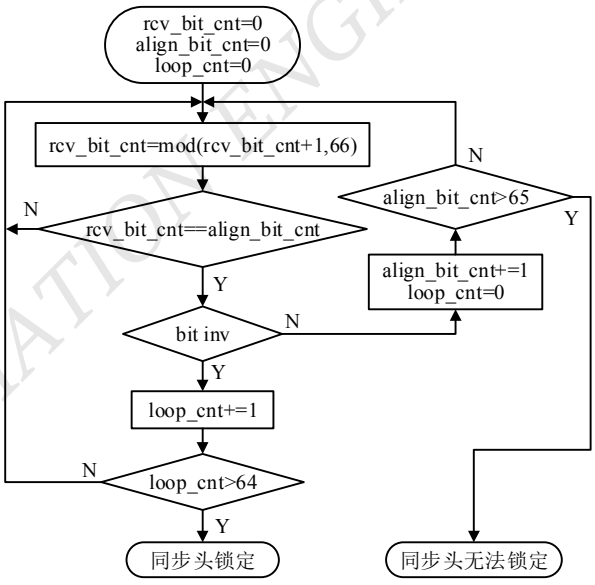

图 3　同步头查找流程图

同步头译码规则如表 4 所示。若同步头位置接收的数据为 00 或 11，则上报接收链路异常，接收的数据无效。

表 4　同步头译码规则表

收到同步头 bit[1:0]	译码同步头 bit
00	无效头
01	0
10	1
11	无效头

扩展多块同步的流程如图 4 所示，其具体步骤如下：

1） 将同步头计数器 rcv_sh_cnt、同步头缓存器 rcv_sh_buf、扩展多块计数器 emb_cnt、错误计数器 err_blk_cnt 清零；

2） 每译码出一个新的同步头 sh_din，循环左移到 32 bit 的同步头缓存器 rcv_sh_buf，同步头计数器 rcv_sh_cnt 循环累加，最大值为 32×E（E 为扩展多块包含的多块个数）；

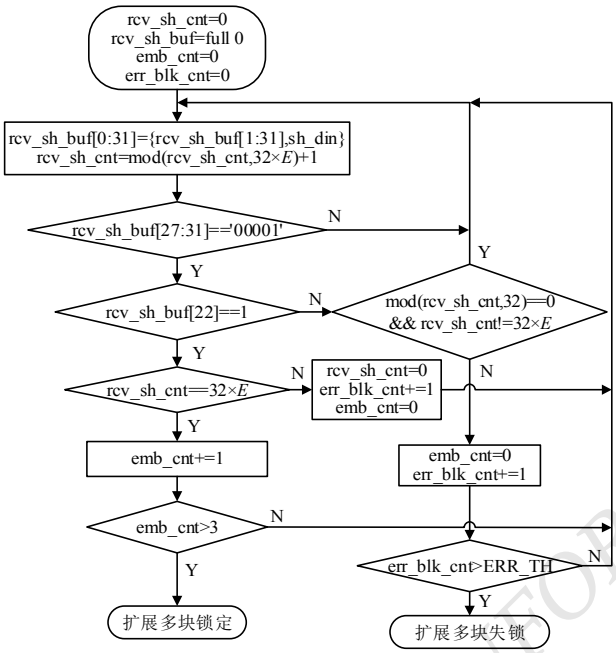

图 4 扩展多块同步流程图

3） 当同步头缓存器的 rcv_sh_buf[27:31]={0,0,0,0,1}，且 rcv_sh_buf[22]=0 时，表明非扩展多块结束，需进一步判断是否为多块结束位置；

4） 若同步头计数器 rcv_sh_cnt 不等于 32×E，但为 32 的倍数，则继续接收同步头；否则，扩展多块计数器 emb_cnt 清零，错误计数器 err_blk_cnt 累加；

5） 若错误计数器 err_blk_cnt 大于设置门限 ERR_TH，表明扩展多块失锁，返回搜索失败信息；否则，继续接收同步头；

6） 若 rcv_sh_buf[22]=1，且同步头计数器 rcv_sh_cnt 等于 32×E，表明找到一个扩展多块，扩展多块计数器 emb_cnt 累加；

7） 当扩展多块计数器 emb_cnt 大于 3 时，则扩展多块锁定，返回扩展多块同步信息。

传统同步算法[3]采用串行处理方式，一次仅处理一个数据，这在实际运用中几乎不可行。本文的技术难点和核心为数据并行处理。数据并行度过大会增加系统面积，过小则会增加系统功耗，并在一定程度上给后端时序约束带来困难。考虑到同步并行实现的面积和功耗等因素，本文选取 66 bit 并行来处理数据。此外，66 bit 还是一个码块的宽度，方便与物理层及后续模块的接口衔接。

3 并行同步头查找算法

物理层将接收的串行数据转换为 66 bit 并行数据后，系统时钟频率降为串行时钟的 1/66，从而减少了时序限制和功耗[10-11]。但物理层仅负责串并数据转换，不进行数据对齐[12]，因此 66 bit 的任意位置都可能是同步头。快速准确地找出同步信息，以达到收发同步，是 JESD204C 接口协议设计者努力实现的目标。

本文提出一种快速可靠的并行同步头查找算法。该算法设计简单，仅有位异或、位与两级操作，且资源开销少，仅需 1 个 67 bit 寄存器 $b(n)$，3 个 66 bit 寄存器 $x(n)$、$a(n)$和 $dly(n)$。并行同步头查找算法结构如图 5 所示。

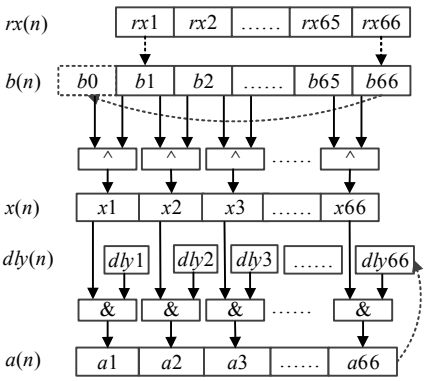

图 5 并行同步头查找算法结构图

物理层接收的 66 bit 数据 $rx(n)$被送入寄存器 $b(n)$的 $b1$~$b66$，$b0$ 等于上一拍的 $b66$。对 $b(n)$的相邻两位进行异或处理，记为 $x(n)=b(n\text{-}1)\wedge b(n)$，即

$$x1=b0 \wedge b1,$$

$x2=b1 \wedge b2$,

.....

$x66=b65 \wedge b66$。

定义 $a(n)$为 66 bit 数据，$dly(n)$为 $a(n)$的上一拍数据，其初值全为 1。记为 $a(n)=x(n)\ \&dly(n)$，即

$a1=x1\ \&dly1$,

$a2=x2\ \&dly2$,

.....

$a66=x66\ \&dly66$。

基于上述两级流水运算，只需 65 个时钟周期即可根据 $a(n)$值判断同步头位置。若 $a(n)$中有且仅有一个 1，则表示找到同步头，且该 1 的位置即为同步头位置；否则，表示同步失败。若因数据随机导致 $a(n)$中有多个 1 出现，可通过增加运算次数来保证只有一个 1，从而避免混淆，以便找出正确的同步头。

块同步后，进行扩展多块同步。定义一个 32 bit 的同步头缓存器 rcv_sh_buf(n)，每译码出一个同步头 sh_din，便将其循环左移到该缓存器。同步头缓存示意图如图 6 所示。

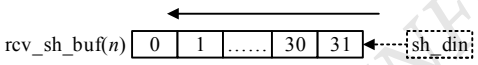

图 6　同步头缓存示意图

扩展多块同步状态转移图如图 7 所示。

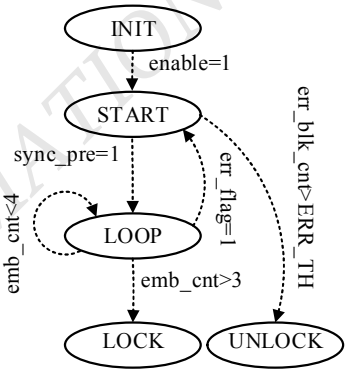

图 7　扩展多块同步状态转移图

扩展多块同步状态转移过程如下：

1） 初始上电或复位后，同步模块进入 INIT 初始化状态，复位清零各缓存器和寄存器；

2）当使能 enable=1 启动，同步模块进入 START 状态，同步头缓存器 rcv_sh_buf 开始接收译码同步头 sh_din；

3） 当同步头缓存器 rcv_sh_buf[27:31]={0,0,0,0,1}，且 rcv_sh_buf[22]=1 时，即扩展多块的所有导频信号均已找到，初始同步标识置位 sync_pre=1，同步头计数器 rcv_sh_cnt 清零并开始累加，错误标志 err_flag 清零，同步模块进入 LOOP 状态；

4） 当同步头计数器 rcv_sh_cnt=32×E，且同步头缓存器 rcv_sh_buf[27:31]={0,0,0,0,1}，rcv_sh_buf[22]=1 时，扩展多块计数器 emb_cnt 加 1，rcv_sh_cnt 清零重新累加；

5） 当 emb_cnt>3 时，扩展多块同步成功，同步模块进入 LOCK 状态；

6） 在 LOOP 状态下，当 rcv_sh_cnt=32×E 时，若 rcv_sh_buf[27:31]≠{0,0,0,0,1}或 rcv_sh_buf[22]≠1，则置错误标志 err_flag=1，错误计数器 err_blk_cnt 加 1，同步模块重新进入 START 状态；

7） 在 LOOP 状态下，当同步头计数器 rcv_sh_cnt=32，但不等于 32×E 时，需满足同步头缓存器 rcv_sh_buf[27:31]={0,0,0,0,1}，且 rcv_sh_buf[22]=0，否则置错误标志 err_flag=1，错误计数器 err_blk_cnt 加 1，同步模块重新进入 START 状态；

8）在 START 状态下，当错误计数器 err_blk_cnt 大于设置门限 ERR_TH 时，表明扩展多块无法同步，同步模块进入 UNLOCK 状态；

9） 当同步模块处于 LOCK 状态时，系统已进入接入同步状态，此时仍可依据同步头信息进行实时链路跟踪，并实时上报链路状态。

4　并行同步头查找算法仿真验证

为了验证并行同步头查找算法的正确性，本文基于 MATLAB 平台，随机产生一组数据，并在起始位置加入随机冗余数据，相关参数配置如表 5 所示。

表 5　同步源参数配置表

参数	配置值	描述
F	8	一块包含的字节数
L	1	通道数
M	4	转换器个数
N	16	转换器位宽
N’	16	加入控制位后的位宽
E	1	扩展多块包含的多块个数
S	1	过采样次数

随机生成 64 个多块数据，放大其中的 850~1 050 bit 如图 8 所示，可看出数据完全随机。

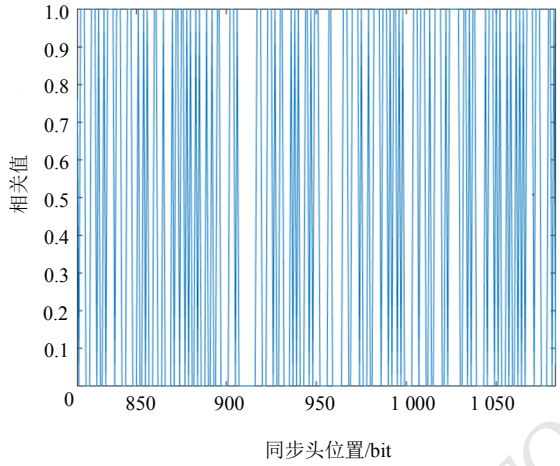

图 8　随机多块数据

并行同步头查找算法仿真找到的多块同步头位置如图 9 所示。峰值 1 对应的横坐标值为同步头位置，即在 66 bit 并行数据的 bit[6]位置。

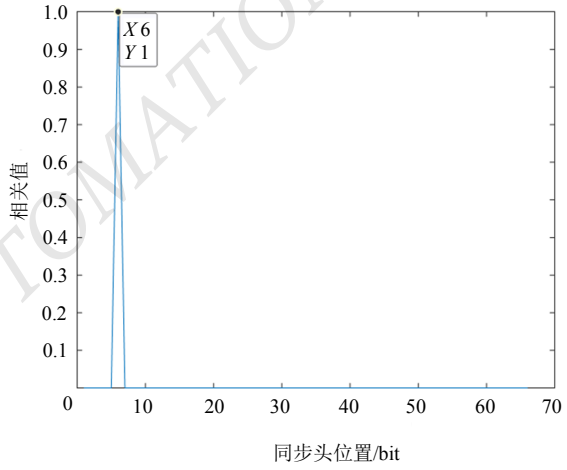

图 9　多块同步头位置

并行同步头查找算法的 MATLAB 代码如图 10 所示。为编码方便，将 bn 定义为 66 bit，图 5 中的 *b*0 单独定义为 bn_last66。

扩展多块同步仿真代码如图 11 所示。

由图 11 的命令窗（Command Window）可看出，扩展多块计数器 emb_cnt 的仿真结果为 4，表明同步模块处于 LOCK 状态，链路成功接入。

```
30      %% find header.
31 —    bn        = zeros(1,66);
32 —    xn        = zeros(1,66);
33 —    an        = zeros(1,66);
34 —    dlyn      = ones(1,66);
35 —    bn_last66 = 0;
36 —    for rep_i=1:64
37          % step 1: prepare xor data.
38 —        bn(1,:) = rxn(rep_i,:);
39          % step 2: xor process.
40 —        xn = xor(bn,[bn_last66,bn(1:65)]);
41          % step 3: and process.
42 —        an = and(dlyn,xn);
43          % ready for next loop.
44 —        bn_last66 = bn(1,66);
45 —        dlyn      = an;
46 —    end
47      % step 4: find if there is 1.
48 —    idx = find(an==1);
```

图 10　同步头查找算法的 MATLAB 代码

```
%% find extended multiblock header.
rcv_sh_buf = zeros(1,32); % INIT STATUS.
emb_cnt = 0;
rcv_sh_cnt = 0;
sync_pre = 0;
err_blk_cnt = 0;
for rep_i= 1:length(rxn) % START STATUS.
  rcv_sh_buf(1:31)= rcv_sh_buf(2:32);
  rcv_sh_buf(32)=rxn(rep_i, idx); % Detect synchronization header condition
  if((sync_pre == 0)&& (rcv_sh_buf(23)== 1)&& all(rcv_sh_buf(28:32) == [0 0 0 0 1]))
      emb_cnt = 0;
      rcv_sh_cnt= 0;
      sync_pre = 1;
  end
  if(err_blk_cnt>7)   % ERR_TH=7
      fprintf("err_blk_cnt=%d. EMB UNLOCK!\n",err_blk_cnt);
```

```
        break;
    end
    if (sync_pre == 1)% LOOP STATUS:
        if (mod(rcv_sh_cnt, 32*1)== 0)% Shift 32*E times
            if((rcv_sh_buf(23)==1)&& all(rcv_sh_buf(28:32)== [0 0 0 0 11]))
                emb_cnt=emb_cnt + 1;
            else
                sync_pre = 0;
                err_blk_cnt= err_blk_cnt + 1;
            end
        end
    end
    rcv_sh_cnt= mod(rcv_sh_cnt, 32*1) + 1;
    if(emb;cnt >3)   % Locking condition check
        fprintf("emb_cnt=%d. EMB LOCK!\n",emb_cnt);
        break;
    end
end
```

图 11　扩展多块同步 MATLAB 代码

5　结论

本文对 JESD204B、JESD204C 接口协议的同步算法进行了分析比较，并基于 JESD204C 接口协议的串行同步算法，提出一种并行同步头查找算法。仿真结果表明，本文所提的并行同步头查找算法符合 JESD204C 接口协议规定，实现简单，在一定程度上降低了同步模块的复杂度和功耗，可作为 IP 应用于 JESD204C 接收端电路，对推动高速接口芯片的国产化进程具有一定的借鉴意义。

然而，对同步的研究远未结束。根据同步头进行实时链路质量跟踪；将同步头作为载体向接收端传递时间戳、命令、FEC 等信息；以及实现这些信息的无损实时切换，均是未来的研究方向。

参考文献

[1] 刘安,禹卫东,马小兵,等.基于 FPGA 的高速串行数据收发接口设计[J].电子技术应用,2017,43(6):48-51.

[2] JEDEC.JEDEC Standard JESD204B[S].U.S.A. JEDEC Solid State Technology Association, 2011:11-12.

[3] JEDEC.JEDEC Standard JESD204C[S].U.S.A. JEDEC Solid State Technology Association, 2017:17,133-138.

[4] 李士杰,马瑞昌,邓明兴,等.JESD204C 高速串行接口电路设计技术[J].微纳电子与智能制造,2023,5(3):14-21.

[5] 张春茗,杨添,王一平.JESD204C 协议接收端 64B/66B 链路层电路设计[J].西安邮电大学学报,2021,26(1):60-66.

[6] 张春茗,杨添,严展科,等.基于 6 Gsample/s 12bit ADC 接口控制层电路设计与实现[J].电子器件,2020,43(5):1142-1147.

[7] 赵文飞,王永禄,陈刚.一种符合 JESD204C 协议的并行 FEC 译码器[J].微电子学,2023,53(1):50-54.

[8] 欧阳靖,姚亚峰,霍兴华,等.JESD204B 协议中发送端同步电路设计与实现[J].电子器件,2017,40(1):118-124.

[9] 孔玉礼,陈婷婷,万书芹,等.基于 JESD204B 协议的接收端电路设计[J].电子与封装,2022,22(12):77-83.

[10] 陈思浩,吴黎明,彭克锦,等.基于 ZYNQ 平台的卷积神经网络加速器设计与实现[J].自动化与信息工程,2024,45(1):30-34.

[11] 胡建华,王亚伟,厉宝山,等.一种新型串行传输协议研究及实现[J].机电工程技术,2022,51(8):193-198.

[12] YIN Peng, SHU Zhou, XIA Yingjun, et al. A low-area and low-power comma detection and word alignment circuits for JESD204B/C controller[J]. IEEE Transactions on Circuits and Systems-I: Regular Papers, 2021,68(7):2925-2934.

作者简介：

代苏杰，男，1979 年生，硕士研究生，工程师，主要研究方向：数字集成电路设计。E-mail: pcalap@163.com

杨定坤，男，1995 年生，硕士研究生，工程师，主要研究方向：射频模拟集成电路设计。E-mail: iwasrick@qq.com

阳江平，男，1985 年生，硕士研究生，高级工程师，主要研究方向：数字集成电路设计。E-mail: 452727408@qq.com

耿建强，男，1975 年生，硕士研究生，高级工程师，主要研究方向：射频模拟集成电路设计。E-mail: 13730636101@163.com