Journal of Sensor Technology and Application 传感器技术与应用, 2022, 10(2), 193-201
Published Online April 2022 in Hans. http://www.hanspub.org/journal/jsta
https://doi.org/10.12677/jsta.2022.102024

Hans汉斯

基于FPGA实现快速查找CAM缓存设计

陈重实

合肥工业大学，安徽 合肥

收稿日期：2022年3月8日；录用日期：2022年4月5日；发布日期：2022年4月14日

摘　要

随着当代通信的大力发展，通信场景更加广泛，对通信网络中的要求越来越高，对路由等设备的需求加大，CAM表高速查找的特性可以适用更多场景。本文阐述CAM缓存器(Content Addressable Memory)的应用场景，阐述了传统的CAM设计原理与性能优缺点，提供了新的CAM缓存器的设计方案，基于CRC32算法原理的基础上，将传统缓存RAM与CRC32原理相结合，将CRC32算法作为编码算法，将编码后的算法，做数据处理，将处理后的数据当作数据RAM和冲突RAM的地址，那么每一个数据和它随对应的地址都有一种算法关系，进行查找数据的时候，最快一个周期就可以实现查找，所以在此方案下可以实现高性能的CAM查找表。

关键词

Verilog，CAM表，CRC32，性能测试

Fast Search CAM Cache Design Based on FPGA

Zhongshi Chen

Hefei University of Technology, Hefei Anhui

Received: Mar. 8th, 2022; accepted: Apr. 5th, 2022; published: Apr. 14th, 2022

Abstract

With the development of contemporary communication, communication scenes are more and more extensive, and the requirements for communication networks are higher and higher. The demand for routing equipment increases. The feature of CAM table high-speed search can be applied to more scenes. In this paper, the application scenario of Content Addressable Memory (CAM) is described, the design principle of traditional CAM and its performance advantages and disadvantages are described, and the design scheme of a new CAM is provided. Based on the principle of CRC32

文章引用: 陈重实. 基于 FPGA 实现快速查找 CAM 缓存设计[J]. 传感器技术与应用, 2022, 10(2): 193-201.
DOI: 10.12677/jsta.2022.102024

algorithm, the traditional cache RAM is combined with the principle of CRC32. The CRC32 algorithm is used as the encoding algorithm, the encoding algorithm is used for data processing, and the processed data is regarded as the address of data RAM and conflicting RAM. Then each data has an algorithm relationship with its corresponding address. When searching for data, the search can be realized in the fastest cycle. Therefore, high-performance CAM lookup table can be realized under this scheme.

Keywords

Verilog, CAM Table, CRC32, Performance Test

1. 引言

随着互联网行业的快速发展，网络的不断升级，从 2G、3G、4G、到现如今的 5G，网络传输的速度越来越快，在现如今的 5G 通信网络中，因为性能的提高，所以其中包含的路由器等转发路由设备也需要发展，为了适配通信网络重数据量增大的趋势，既要保证数据链路的完整性，又要保证其流畅性，因此对路由器，交换机机等设备提出了更高的要求——在通信中如何快速路由。传统的网络查找技术都需要通过软件，但通过软件，就必然需要 CPU 的指令操作，那样就会浪费时间，降低效率，随着市面上 Altera 和 Xilinx 的 FPGA 芯片的性能越来越高，发展的技术越来越成熟，因此 FPGA 在网络中也被使用的频率越来越高。

CAM (Content Address Memory)是一种与 RAM 完全相反的缓存机制，当我们使用 RAM 的时候，每个数据存在某一个地址下，当我们想读取数据时，索引地址来读取数据，但 CAM 是相反的，想要查询某个地址是否存在，通过索引数据来查询这个地址是否存在。CRC32 是一种循环冗余校验码，在数据搜索和数据处理中经常会用到，经常用于保护数据的正确性和完整性时会采用，通常的 CRC32 用于计算一帧数据包的 CRC 数值，其 CRC 数值是通过对每个数据的 CRC 反复运算得到的，最终累计计算出结果，将 CAM 与 CRC32 相结合，这样就能保证，每一个存进 CAM 的数据都有一个属于自己的 CRC 值，如果把这个计算出的 CRC 值当作该数据的地址进行缓存，那么每个数据的自己的地址具有相关性，这样查找起来方便很多[1]。

文献[2]基于 FPGA 的低功耗预比较策略实现 CAM，实现 CAM 的读写功能，CAM 的写功能将缓存设置成 M*N 的坐标轴，按照坐标写入，读取的时候依次比较进行读取，虽然查找功能实现，但是查找过程需要一次比较，查找周期过长。文献[3]一种基于 FPGA 的内容可寻址存储器的设计，对 CAM 的实现做了具体的阐述，但是其中编解码的算法没有具体说明，因此无法判断该 CAM 的读写性能是否优良。

2. 功能介绍

FPGA 中有 BRAM，即 block ram，一种存储器。可以给数据地址写入数据，并且给地址读出对应的数据。相反地，给数据和地址写入数据之后，如果给数据来读出对应的地址，这样的存储结构就是 CAM 表，CAM 表既可以用硬件电路做出也可以用 FPGA 来实现。在 FPGA 没有被广泛应用的时期，CAM 在早些年在软件中是一种昂贵的设备，主要用途是 CPU 的 cache 中。最近这些年，互联网和物联网的高速

发展，早期使用软件匹配算法实现的路由器、防火墙、入侵检测等技术，已经不能适用于当代这个网路技术，在 FPGA 的基础上实现 CAM 的功能是目前解决路由问题的主要方法。

CRC 校验经常使用的方法是多项式。一个 n 阶 2 进制的多项式子可以一次被处理：(an-1)*(n-1) + (an-2)*(n-2) + … + a0。比如目前有一个 8bit 的进制数字表示为 10110100：1*7 + 0*6 + 1*5 + 1*4 + 0*3 + 1*2 + 0*1 + 0*0。多项式乘除法运算过程与普通代数式的乘除法相同。多项式的加减法运算以 2 为模，加减时不进位、错位，和逻辑异或运算一致[4]。

3. CAM 的一般方法

目前，实现 CAM 的方法有很多种，但都有利有弊，对应不同场景选用不同的方法，像最初的顺序查找法，这种方法应用的原理与 ram 基本一致，想找到其中一个数据对应的地址，就要遍历所有的数据，随着 CAM 的深度增加，遍历的时间越长，此方法仅使用于一些对时间性能要求不高的场景。因为顺序查找法的延迟很大，也伴随着 Hash 查找法的产生，对原有数据进行 Hash 算法，经过 Hash 后，将数据存放到固定的位置上，下此查找时一次就能找到，虽然此方法查找速度较快，但是此方法容易产生 Hash 冲突，并且 Hash 算法在 FPGA 中实现较复杂，容易出现时序违例，布局布线复杂，从而有时会选择降低频率，从而影响整个模块。

3.1. 顺序查找法

顺序查找又称为线性查找，就是先将一些数据依次写入一张表中，每一个数据会有相应的地址对应，当我们想查找这个表中是否存在某一个数据的时候，就要从头开始依次比较，第一个比较不对，就比较第二个数据，以此类推到第 n 个数据，如果在查找的过程中，比对成功了，要么返回该数据所对应的地址，要么显示查找成功[5]。

3.2. Hash 查找法

Hash 方法的主要原理是查找 Hash 表，Hash 表中的数据，都是经过各种 Hash 算法后，每一个数据都有一个固定的位置存放，当我们进行 Hash 查找时，只需要将数据再通过 Hash 函数的计算后，便可求得数据的位置，即可进行一次的数据的比较即找到欲查找数据。在 Hash 结构中，我们称输入数据的值为键值(Key) [6]。

在 Hash 查找中必须要选择 Hash 函数，常见的几种 Hash 函数有：随机数运算法、旋转运算法、中间平均运算法、直接运算法、余数运算法、折迭运算法，数值抽出运算法。

在 Hash 查找中，利用 Hash 函数所产生的数据的位置可能会发生数据位置已存在有一笔数据的情形，此时我们称为 Hash 碰撞。发生 Hash 碰撞时，我们必须提供一些解决方法，才能让数据有对应的存储位置，否则会造成数据流失的错误。常见的 Hash 碰撞解决法有下列几种：

1) 线性开放寻址法；
2) 差值解决法；
3) 链表解决法；
4) 分桶 Hash 法[7]。

3.3. 优化 CAM 查找方法

在我们进行 CAM 表查找的过程，我们怎样能将顺序查找法的实现简单，与 Hash 查找的速度相结合，经过长时间的研究，想到一种既实现简单，查找速度又快的方法，准备将顺序查找法与 CRC32 算法相融合。

CRC (Cyclic Redundancy Check)纠错校验经常使用在数据缓存和 5G 通信领域，因为要确保数据的完整性，流畅性和正确性为，所以我们必须对每一帧数据都要采取纠错的手段。在目前校验检错的方法中，CRC 被用的次数是最多的、最频繁的，因为它是最稳定的，最可靠的[8]。CRC32 虽经常用于数据校验，但是我们也乐意用在 CAM 查找中，假设 64 bit 的数据进行 CRC32 后就变成了 32 bit，每一个 64 bit 数字进行编码后都会有一个独特的 32 bit 数字，但是这里肯定有人认为 64 bit 变 32 bit 那肯定会产生冲突的，没错，理论上是这样的，只要超过有 2 的 32 次方的数次，那么就会产生冲突，但是 2 的 32 次方是一个多么大的数字，FPGA 的缓存容量本来就不多，所以我们基本可以忽略冲突，为什么要选择 CRC32 呢，目前 CRC32 技术非常成熟，它的算法都是异或来实现，要比 Hash 简单的多。

以下是我基于 CRC32 算法的基础上进行优化的 CAM 表查找架构图，如图 1 所示。

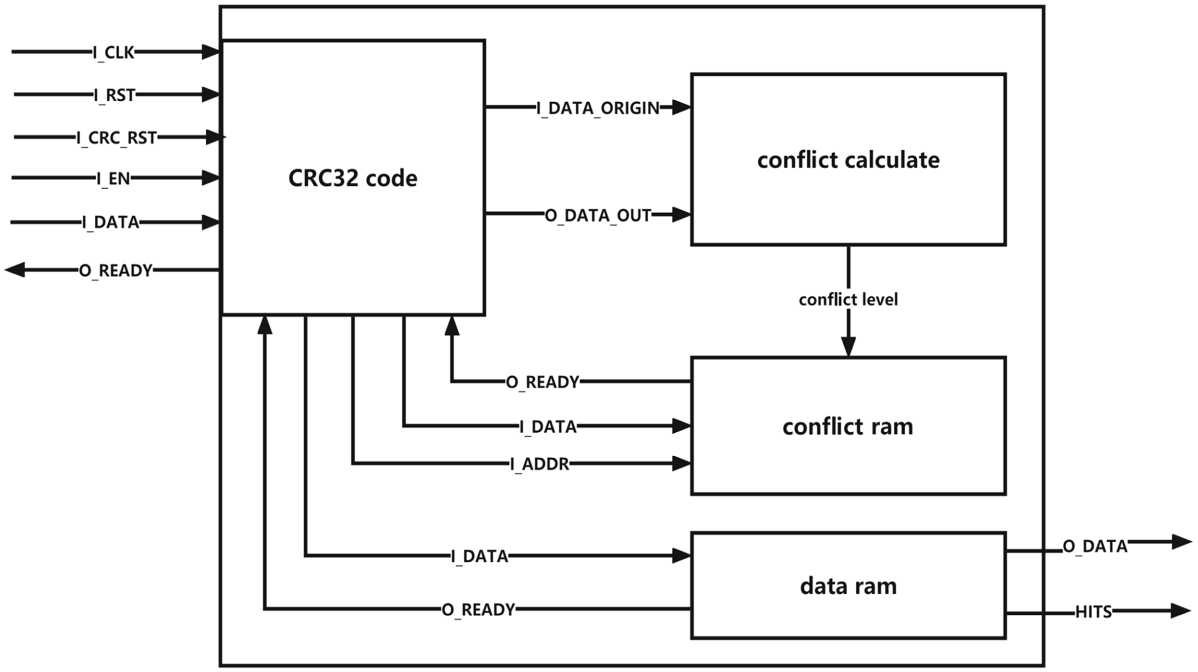

Figure 1. Architecture diagram
图 1. 架构图

具体功能是这样实现的：

写入过程：首先我要么将里面的缓存进行初始化，全部清 0，然后将输入进来的数据进行 CRC32 运算，得到一个 32 bit 的数据，但是我们只取其中低 12 bit，为什么只取低 12 bit 呢，因为 12 bit 位宽也可以表达 2 的 12 次方个数字了，在实际应用中已经是一个很大的数字，这 12 bit 的数据将作为冲突计算模块的输入，在 conflict calculate 中具有一个检测功能，会将这个 12 bit 的数据的最高位拼接上一个 00，达到一个 14 位宽 bit 的数据。将这个数据作为 conflict ram 的基础地址进行查询，如果查询到该地址是有数据对应的，那么证明产生了冲突，那么我就在这个 12 bit 数据前面补上 01，然后去查绚这个地址下面是否有数据，如果该地址下面依然存在数据，那么表示存在了二级冲突，那么就要在 12 bit 数据前面补上 10，以此类推，根据前面补位数不同，所以支持的冲突级数也不相同，12 bit 可以表示很多数字，所以当一个缓存器如果实时更新里面的数据内容，那么在实际情况下产生的冲突数量几乎为 0，并且这样做做法的好处就是每一个数据与它的地址是有相互关系的，查找的过程速度很快，以下为写入状态机，如图 2。

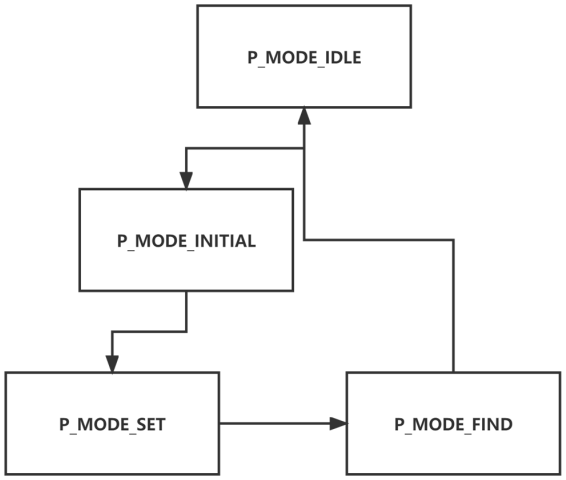

Figure 2. Write state machine
图 2. 写入状态机

查找过程：如果我们想查找 CAM 表中是否存在某个数据，要将输入进来数据做 CRC32 得到一个 64 bit 的数据，将 64 bit 的数据截断，取它的低 12 bit，将这 12 bit 数据发送给 conflict calculate 检查是否有多个数据经过 CRC32 后会产生这 12 bit 数据，就是检查是否存在冲突，将这 12 bit 数据前面补上 00，然后当作 conflict ram 的地址去进行查询，如果该地址下对应的数据为 1，证明已经存在了一级冲突，那么我们就将 12 bit 数据前面补上 01 去查询，如果该地址下显示为 0，那就证明没有存在 2 级冲突，所以就要取 data ram 中对应的这两个地址位去比较，两个数据中有一个就是我们想要查询的数据，最多比较两次得到结果，以下为读出状态机，如图 3。

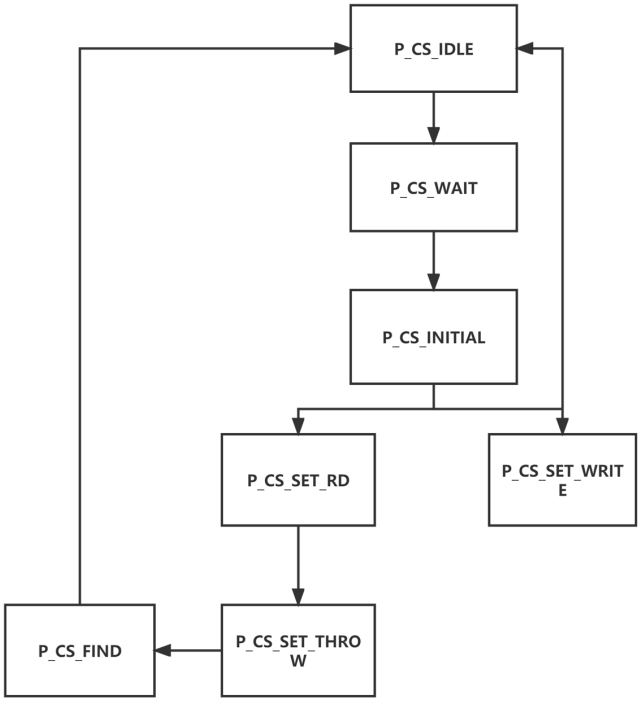

Figure 3. Read state machine
图 3. 读出状态机

4. 功能验证

4.1. FPGA 验证

在我们将 design 设计完成后，一般都要进行仿真测试，主要分为前仿真和后仿真，前仿真指的就是不考虑时序和布局布线等问题，基于仿真工具，通过仿真波形和测试用例来比对结果是否正确，功能是否达到，常用的仿真工具有 modelsim、VCS 等，后仿真就是看我们的设计是否有时序违规，布局布线是否合理，建立时间和保持时间是否达到要求，有时一些毛刺也是在后仿真时候发现的，后仿真相对来说是和真实情况符合的，最反应实际工作情况的仿真，一般当我们前方真没有问题后再进行后仿真。

4.2. 验证方法

要测试 CAM 的功能是否实现，以及测试写入功能需要的始终周期，以及发生冲突时写入时钟周期，测试查找功能的时钟周期，以及发生冲突的情况下查找需要的时钟周期。因为当数据量足够大的时候，才会产生冲突，所以要不断增加发包数量直到产生冲突后，再测试冲突模式下写入和查找需要的时钟周期，根据测试结果与顺序查找法和 Hash 查找法进行比较结果，以此可以直观的看出优化的好处。

基于以上考虑，要逐渐增加发包数量，产生冲突，来判断在常规时候和极限状态下，写入和查找模式下的 CAM 性能，因此我做了以下三次发包实验来做数据分析：

连续写入 100 个 64 bit 的数据，这些数据的内容是随机的，是不相同的，然后依次将数据读出来与输入数据对比是否存在丢包，然后开启查找模式，查找第 100 个数据；

连续写入 1000 个 64 bit 的数据，这些数据的内容是随机的，是不相同的，然后依次将数据读出来与输入数据对比是否存在丢包，然后开启查找模式，查找第 1000 个数据；

连续写入 10000 个 64 bit 的数据，这些数据的内容是随机的，是不相同的，然后依次将数据读出来与输入数据对比是否存在丢包，然后开启查找模式，查找第 1000 个数据；

测试部分代码如下：

```
    task    set1;
    input [31:0] times;
    begin
                for(i =0 ;i<=times;i=i+1)begin
s={$random}%999999;
$fwrite(fo1,“%6h\n”,s);
q[i]<=s;
@(posedge I_clk);
  I_req=1;
  I_mode=P_MODE_SET;
  I_valid =1;
  I_din=s;
  I_addr=I;
@(posedge I_clk);
  I_req=0;
  I_valid=0;
I_din=0;
I_addr=0;
@(negedge O_busy);
@(posedge I_clk);
                end
end
endtask
```

```
initial begin
        I_req=0;
        I_mode=P_MODE_NOP;
        I_valid =0;
        I_din=0;
        I_addr=0;
        Lostpkg=0;
        Rstpulse;
        Init
        Wait(O_done == 1);
        #1000
        fo1=$fopen(“datain1.txt”);
        fo2=$fopen(“datain2.txt”);
        fo3=$fopen(“datacompare.txt”);
        set(10000);
        find1(10000);
        $fclose(fo1);
        $fclose(fo2);
        $fclose(fo3);
        $display(“lostpkg is %d\n”,lostpkg-1);    #1000
end
```

4.3. 验证结果

验证工具 Questasim 10.6c 上，通过 Verilog 建立的验证环境对 CAM 控制器的功能进行了完整测试。并通过写测试 tb，对 CAM 表产生大量随机数，进行数据写入，从仿真波形中可以看出 CAM 控制器状态机跳转过程，从初始化——数据算法——冲突计算——数据写入——数据读取——数据比对，可以看到各个过程，最后我们我在 tb 中将比较结果产生了输出文件，以及发包数量由少变多的情况下，冲突数量的变化的记录，如图 4。

发包数量	冲突结果
100	0
1000	0
10000	4

Figure 4. Test statistic
图 4. 测试统计

我们将测试结果与顺序查找法和 Hash 查找法从几个维度做了清晰直观的比较，包括写入，查找所用时钟周期对比，从而反应性能的好坏，通过对内存资源的使用进行对比，从而反应资源情况的好坏。

类别	顺序查找法	Hash查找法	优化查找法
资源（位）	15224K	40113K	18991K

Figure 5. Resources compare
图 5. 资源比较

类别	顺序查找法	Hash查找法	优化查找法
写入周期	1 CLK	1 CLK	1 CLK
查找周期	100 CLK	1 CLK	1 CLK

Figure 6. Test 1
图 6. 测试 1

类别	顺序查找法	Hash查找法	优化查找法
写入周期	1 CLK	1 CLK	1 CLK
查找周期	1000 CLK	1 CLK	1 CLK

Figure 7. Test 2
图 7. 测试 2

类别	顺序查找法	Hash查找法	优化查找法
写入周期	1 CLK	1 CLK	1 CLK
查找周期	10000 CLK	1 CLK	3 CLK

Figure 8. Test 3
图 8. 测试 3

我们通过测试结果可以发现，在资源使用上，如图 5，优化查找法，相比于顺序查找法的使用资源会相对多一些，但是明显少于 Hash 查找法的资源，在写入和查找时钟周期上，顺序查找法需要一一比较，当查找后面数据的时候，时钟周期明显变多，与 Hash 查找法相比，如果没有存在冲突的情况下，如图 6 和如图 7，没有产生冲突，写入和查找时钟周期相同，如图 8 中，在极限情况下，因为连续写入大量数据，产生了冲突，所以查找周期会比 Hash 查找法多一些。

通过实验我们可以得出结论，优化查找法是一种相对折中的办法，虽相比于顺序查找法多了一些资源，但是性能明显提升，相比于 Hash 查找法，只有在极限情况下，性能会稍微降低，但是会少了很多资源，因此优化查找法完全具备可行性。

5. 结束语

用 FPGA 设计 CAM 存储器，方法简便、使用灵活、匹配速度快，本文基于 FPGA 设计 CAM 存储器，在传统的查找算法上，基于 CRC32 算法的原理，将其融入在 CAM 查找算法的机制里，实现了查找速度快，实现简单的原理，将设计代码，基于 xilinx 的开发板，使用 modelsim 进行仿真，使用 verilog 和 systemverilog 写 testbench 进行性能测试，测试性能完全达到预期效果。

参考文献

[1] 郭军, 尾笹勤. 基于 FPGA 的在线可写 CAM 存储器设计[J]. 微电子学与计算机, 2007(6): 97-99. https://doi.org/10.19304/j.cnki.issn1000-7180.2007.06.029

[2] Rehman, N.U., Mujahid, O., Irfan, M., Hafeez, A. and Ullah, Z. (2019) Low Power Pre-Comparison Configuration Strategy for a Logic-Based Binary CAM on FPGA. 2019 *Second International Conference on Latest trends in Electrical Engineering and Computing Technologies* (*INTELLECT*), Karachi, 13-14 November 2019, 1-5. https://doi.org/10.1109/INTELLECT47034.2019.8955476

[3] 李训根, 罗霁. 一种基于 FPGA 的内容可寻址存储器的设计[J]. 杭州电子科技大学学报, 2014, 34(4): 65-69. https://doi.org/10.13954/j.cnki.hdu.2014.04.016

[4] 张正龙, 张小华, 李冀明, 段怡. 基于 CRC32 的数据校验的研究和应用[J]. 科学咨询(科技·管理), 2011(4): 62-63.

[5] Qazi, A., Ullah, Z. and Hafeez, A. (2021) Fast Mapping and Updating Algorithms for a Binary CAM on FPGA. *IEEE Canadian Journal of Electrical and Computer Engineering*, **44**, 156-164. https://doi.org/10.1109/ICJECE.2020.3025198

[6] Mujahid, O. and Ullah, Z. (2020) High Speed Partial Pattern Classification System Using a CAM-Based LBP Histogram on FPGA. *IEEE Embedded Systems Letters*, **12**, 87-90. https://doi.org/10.1109/LES.2019.2956154

[7] Irfan, M., Ullah, Z. and Cheung, R.C.C. (2019) High Performance Power-Efficient Gate-Based CAM for Reconfigurable Computing. 2019 15*th International Conference on Mobile Ad-Hoc and Sensor Networks* (*MSN*), Shenzhen, 11-13 December 2019, 327-331. https://doi.org/10.1109/MSN48538.2019.00068

[8] Irfan, M., Ullah, Z., Sanka, A.I. and Cheung, R.C.C. (2021) Accelerated Updating Mechanisms for FPGA-Based Ternary Content-Addressable Memory. *IEEE Embedded Systems Letters*, **13**, 37-40. https://doi.org/10.1109/LES.2020.2999471