|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 00:00:00,200 --> 00:00:01,000 | 大家下午好 |  |
| 2 | 00:00:01,000 --> 00:00:01,790 | 我是胡进 |  |
| 3 | 00:00:01,790 --> 00:00:02,880 | 来自赛昉科技 |  |
| 4 | 00:00:02,881 --> 00:00:05,000 | 我负责CPU IP的技术支持 |  |
| 5 | 00:00:05,280 --> 00:00:07,146 | 今天我跟大家分享的是 |  |
| 6 | 00:00:07,146 --> 00:00:08,434 | 基于向量扩展的 |  |
| 7 | 00:00:08,434 --> 00:00:11,040 | x280 Core赋能AI和ML |  |
| 8 | 00:00:13,118 --> 00:00:14,981 | 了解RISC-V的 |  |
| 9 | 00:00:14,981 --> 00:00:16,900 | 指令集的朋友们或许知道 |  |
| 10 | 00:00:17,159 --> 00:00:19,410 | vector的扩展 |  |
| 11 | 00:00:19,568 --> 00:00:21,970 | 终于终于是public review了 |  |
| 12 | 00:00:21,971 --> 00:00:23,309 | 它定在6月25号 |  |
| 13 | 00:00:23,665 --> 00:00:26,750 | 这一组指令集被称为 |  |
| 14 | 00:00:26,750 --> 00:00:28,925 | RISC-V最重要的一组指令集 |  |
| 15 | 00:00:28,925 --> 00:00:31,209 | 那个SiFive是相应的 |  |
| 16 | 00:00:31,209 --> 00:00:32,350 | task group的leader |  |
| 17 | 00:00:32,406 --> 00:00:35,370 | Chris教授本人是担任主席的 |  |
| 18 | 00:00:35,581 --> 00:00:37,562 | 向量指令集其实是 |  |
| 19 | 00:00:37,562 --> 00:00:39,185 | 标量值邻近的一个扩充 |  |
| 20 | 00:00:39,186 --> 00:00:41,085 | 它是用来做并行计算的 |  |
| 21 | 00:00:41,085 --> 00:00:42,884 | 那RISC-V的向量指令集 |  |
| 22 | 00:00:42,884 --> 00:00:44,193 | 的特点就是它 |  |
| 23 | 00:00:44,487 --> 00:00:47,265 | 并行计算的长度可以动态的调整 |  |
| 24 | 00:00:47,634 --> 00:00:50,993 | 基于这个衍生的三个特点 |  |
| 25 | 00:00:51,065 --> 00:00:53,240 | 第一个就是code size会节省 |  |
| 26 | 00:00:53,630 --> 00:00:54,515 | 第二就是说 |  |
| 27 | 00:00:54,515 --> 00:00:59,090 | 比之前的那些并行计算 |  |
| 28 | 00:00:59,090 --> 00:01:02,130 | 固定长度的cmd的架构 |  |
| 29 | 00:01:02,131 --> 00:01:04,090 | 它的能效比会高 |  |
| 30 | 00:01:04,318 --> 00:01:06,303 | 第三个就是它的机器代码 |  |
| 31 | 00:01:06,450 --> 00:01:08,565 | 它其实是可以有最好的一致性 |  |
| 32 | 00:01:09,721 --> 00:01:11,512 | 说完向量扩展 |  |
| 33 | 00:01:11,512 --> 00:01:14,153 | 我们来看AI和ML |  |
| 34 | 00:01:14,387 --> 00:01:17,400 | 我们来看看在A极端部署 |  |
| 35 | 00:01:17,587 --> 00:01:21,020 | 低功耗推理计算的挑战 |  |
| 36 | 00:01:21,271 --> 00:01:22,756 | 那常见的这些 |  |
| 37 | 00:01:22,912 --> 00:01:24,940 | 嵌入式的控制CPU |  |
| 38 | 00:01:25,100 --> 00:01:29,100 | 它其实是缺乏去处理 |  |
| 39 | 00:01:29,131 --> 00:01:34,035 | 推理计算能力的 |  |
| 40 | 00:01:34,171 --> 00:01:35,412 | 那然后业界的 |  |
| 41 | 00:01:35,412 --> 00:01:38,084 | 这些专门的加速器 |  |
| 42 | 00:01:38,235 --> 00:01:41,993 | 它对于CNN layers 处理的很好 |  |
| 43 | 00:01:42,235 --> 00:01:45,535 | 但是它也有一些不足的地方 |  |
| 44 | 00:01:45,536 --> 00:01:48,231 | 比如说它对于新出来的 |  |
| 45 | 00:01:48,371 --> 00:01:50,775 | 神经网络它可能就不工作了 |  |
| 46 | 00:01:51,065 --> 00:01:54,487 | 然后它大多都比较难编程 |  |
| 47 | 00:01:54,978 --> 00:01:57,509 | 然后它的工具都不是很成熟 |  |
| 48 | 00:01:57,700 --> 00:01:58,743 | 第三个就是 |  |
| 49 | 00:01:58,749 --> 00:02:01,240 | 各家自己的这些 |  |
| 50 | 00:02:01,343 --> 00:02:05,218 | 加速器的方案都很难去维护 |  |
| 51 | 00:02:05,731 --> 00:02:06,909 | 基于这个挑战 |  |
| 52 | 00:02:07,065 --> 00:02:09,871 | 各大计算体系架构的 |  |
| 53 | 00:02:09,871 --> 00:02:11,121 | 回答也都比较统一 |  |
| 54 | 00:02:11,350 --> 00:02:12,900 | 就是在最通用的CPU上 |  |
| 55 | 00:02:13,140 --> 00:02:14,570 | 加大或者增强 |  |
| 56 | 00:02:14,731 --> 00:02:16,918 | 并行计算的能力 |  |
| 57 | 00:02:19,265 --> 00:02:20,890 | 那所以这个 |  |
| 58 | 00:02:20,893 --> 00:02:23,656 | RISC-V的CPU的IP的window们 |  |
| 59 | 00:02:23,656 --> 00:02:25,600 | 它们也都陆续的推出了 |  |
| 60 | 00:02:25,600 --> 00:02:27,337 | 支持vector的扩展的IP |  |
| 61 | 00:02:27,337 --> 00:02:28,275 | 比如说我们赛昉 |  |
| 62 | 00:02:28,415 --> 00:02:30,390 | 推出了最高性能的天枢 |  |
| 63 | 00:02:30,690 --> 00:02:32,540 | 然后SiFive它是推出了 |  |
| 64 | 00:02:32,540 --> 00:02:34,213 | Intelligence的系列 |  |
| 65 | 00:02:34,430 --> 00:02:36,190 | 这个系列它是两大特色 |  |
| 66 | 00:02:36,191 --> 00:02:37,570 | 第一个是在软件上 |  |
| 67 | 00:02:37,571 --> 00:02:40,580 | 它的策略是软件第一和优先 |  |
| 68 | 00:02:40,759 --> 00:02:42,500 | 这个后面会展开来讲一下 |  |
| 69 | 00:02:42,818 --> 00:02:43,915 | 第二就是在硬件上 |  |
| 70 | 00:02:43,915 --> 00:02:45,540 | 它的基础位架构 |  |
| 71 | 00:02:45,646 --> 00:02:48,620 | 是归验证过最成熟 |  |
| 72 | 00:02:48,800 --> 00:02:51,720 | 它同时支持的目前最新 |  |
| 73 | 00:02:51,721 --> 00:02:54,668 | 最标准的向量扩展的指令集 |  |
| 74 | 00:02:54,890 --> 00:02:55,800 | 除此之外 |  |
| 75 | 00:02:55,801 --> 00:02:58,124 | 它还特别针对于AI和ML |  |
| 76 | 00:02:58,124 --> 00:02:59,460 | 加了一些自定义指令集 |  |
| 77 | 00:03:01,468 --> 00:03:03,840 | 这些自定义指令它集中在 |  |
| 78 | 00:03:03,840 --> 00:03:05,469 | INT8的矩阵相乘 |  |
| 79 | 00:03:05,469 --> 00:03:07,681 | 和BF16的data type的处理上 |  |
| 80 | 00:03:07,778 --> 00:03:10,360 | 那第一个重量级的IP |  |
| 81 | 00:03:10,360 --> 00:03:11,695 | 就是x280 |  |
| 82 | 00:03:11,695 --> 00:03:15,190 | x280这个左边是它的架构框图 |  |
| 83 | 00:03:15,443 --> 00:03:17,806 | 了解这个SiFive的IP |  |
| 84 | 00:03:17,806 --> 00:03:19,534 | 的朋友们或许知道 |  |
| 85 | 00:03:19,678 --> 00:03:20,637 | 噢 这个没错 |  |
| 86 | 00:03:20,638 --> 00:03:22,175 | x280它的微架构 |  |
| 87 | 00:03:22,175 --> 00:03:24,317 | 是基于U7开发而来 |  |
| 88 | 00:03:24,497 --> 00:03:27,257 | 它增加了vector的处理单元 |  |
| 89 | 00:03:27,496 --> 00:03:28,758 | 这个vector处理单元 |  |
| 90 | 00:03:28,758 --> 00:03:31,672 | vector的位宽可以达到512-bit |  |
| 91 | 00:03:31,672 --> 00:03:32,537 | 这个非常快 |  |
| 92 | 00:03:32,750 --> 00:03:35,237 | 它同时它的计算能力也可以达到 |  |
| 93 | 00:03:35,277 --> 00:03:37,317 | 每cycle处理512-bit |  |
| 94 | 00:03:37,587 --> 00:03:40,615 | 所以它的性能 |  |
| 95 | 00:03:40,615 --> 00:03:41,990 | 比如说标量的性能 |  |
| 96 | 00:03:41,990 --> 00:03:45,003 | 可以达到5.17 CoreMarks/MHz |  |
| 97 | 00:03:45,003 --> 00:03:49,295 | 它的矢量的性能如果以INT8的 |  |
| 98 | 00:03:49,295 --> 00:03:51,075 | 矩阵乘来去看的话 |  |
| 99 | 00:03:51,295 --> 00:03:54,256 | 它可以deliver 2.2 TOPS |  |
| 100 | 00:03:54,668 --> 00:03:57,968 | 正是因为它增加了 |  |
| 101 | 00:03:57,968 --> 00:03:59,741 | 非常宽的vector unit |  |
| 102 | 00:03:59,830 --> 00:04:02,062 | 为了满足vector unit的 |  |
| 103 | 00:04:02,062 --> 00:04:03,110 | 仿真的需要 |  |
| 104 | 00:04:03,111 --> 00:04:04,759 | 它在memory系统上面 |  |
| 105 | 00:04:04,759 --> 00:04:05,710 | 做了很多的【】 |  |
| 106 | 00:04:05,711 --> 00:04:06,590 | 那第一个来讲 |  |
| 107 | 00:04:06,591 --> 00:04:08,978 | 它又增加了【】 |  |
| 108 | 00:04:09,237 --> 00:04:12,010 | 这个【】单元的【】的话 |  |
| 109 | 00:04:12,010 --> 00:04:15,570 | 它主要是以【】作为主存的 |  |
| 110 | 00:04:15,570 --> 00:04:16,934 | 第二来讲就是 |  |
| 111 | 00:04:16,982 --> 00:04:18,071 | 它的三级cache |  |
| 112 | 00:04:18,071 --> 00:04:19,430 | 后面两级cache |  |
| 113 | 00:04:19,437 --> 00:04:23,081 | 它最后增加了Stride Pre-fetcher的功能 |  |
| 114 | 00:04:23,784 --> 00:04:24,909 | 再然后就是 |  |
| 115 | 00:04:24,910 --> 00:04:27,606 | 它支持了SV48的规格 |  |
| 116 | 00:04:27,609 --> 00:04:31,287 | 以满足AI/ML SoC |  |
| 117 | 00:04:31,287 --> 00:04:33,568 | 对于物理地址和虚拟地址的 |  |
| 118 | 00:04:33,653 --> 00:04:36,509 | 更大空间的需求 |  |
| 119 | 00:04:37,481 --> 00:04:40,837 | 然后x280在性能 |  |
| 120 | 00:04:40,837 --> 00:04:42,730 | 特别优秀的情况前提下 |  |
| 121 | 00:04:42,731 --> 00:04:44,618 | 它的物理的PPA优化的也非常好 |  |
| 122 | 00:04:44,831 --> 00:04:48,651 | 左下角PPA的图大家可以看到 |  |
| 123 | 00:04:48,651 --> 00:04:49,950 | 如果在7nm下的话 |  |
| 124 | 00:04:49,951 --> 00:04:54,115 | 它的typical频率可以达到2.2GHz |  |
| 125 | 00:04:54,350 --> 00:04:58,284 | 它的Core Logic Area也是0.161平方毫米 |  |
| 126 | 00:04:58,560 --> 00:05:01,038 | 如果说把它的vector单元 |  |
| 127 | 00:05:01,071 --> 00:05:05,229 | 把它配置到128b的宽度的时候 |  |
| 128 | 00:05:05,359 --> 00:05:08,680 | 它的Logic Area也是不到0.1平方毫米的 |  |
| 129 | 00:05:10,293 --> 00:05:13,945 | x280它的SNP的多核的架构 |  |
| 130 | 00:05:13,947 --> 00:05:15,726 | 它是目前也是最成熟的 |  |
| 131 | 00:05:15,726 --> 00:05:17,053 | 所以它可以去 |  |
| 132 | 00:05:17,053 --> 00:05:18,785 | 对应到不同的使用场景 |  |
| 133 | 00:05:18,945 --> 00:05:19,825 | 比如说 |  |
| 134 | 00:05:19,825 --> 00:05:22,168 | 穿戴式需要的算力比较小 |  |
| 135 | 00:05:22,353 --> 00:05:24,196 | 单核x280就可以满足 |  |
| 136 | 00:05:24,425 --> 00:05:26,381 | 那么像智能家居这样的 |  |
| 137 | 00:05:26,421 --> 00:05:29,065 | 它就是一个Class 4个核可以满足 |  |
| 138 | 00:05:29,225 --> 00:05:30,537 | 如果像mobile |  |
| 139 | 00:05:30,700 --> 00:05:31,987 | 工业控制机器人 |  |
| 140 | 00:05:31,987 --> 00:05:33,445 | 这样的应用来讲 |  |
| 141 | 00:05:33,487 --> 00:05:36,445 | 它是一个Class的4个核的280 |  |
| 142 | 00:05:36,446 --> 00:05:38,062 | 加上一个专有的NN的加速器 |  |
| 143 | 00:05:38,062 --> 00:05:38,845 | 就可以满足 |  |
| 144 | 00:05:39,300 --> 00:05:41,140 | 对于算力要求最高的 |  |
| 145 | 00:05:41,141 --> 00:05:43,381 | 比如说像数据中心 |  |
| 146 | 00:05:43,721 --> 00:05:47,921 | 它是多个class |  |
| 147 | 00:05:47,953 --> 00:05:50,350 | 然后每个class多个Core的 |  |
| 148 | 00:05:50,428 --> 00:05:52,120 | x280这样的族群 |  |
| 149 | 00:05:52,435 --> 00:05:55,315 | 加上一个更大算力的NN可以去满足 |  |
| 150 | 00:05:56,806 --> 00:05:58,481 | 说到数据中心 |  |
| 151 | 00:05:58,481 --> 00:06:01,218 | 业界有一家公司叫Tenstorrent |  |
| 152 | 00:06:01,218 --> 00:06:02,455 | 它比较出名 |  |
| 153 | 00:06:02,457 --> 00:06:04,459 | 它已经推出了好几代针对于 |  |
| 154 | 00:06:04,459 --> 00:06:07,006 | 数据中心的加速的芯片 |  |
| 155 | 00:06:07,006 --> 00:06:08,555 | 然后出货量也是非常可观 |  |
| 156 | 00:06:09,031 --> 00:06:11,310 | 今年他们发生了两件事情 |  |
| 157 | 00:06:11,350 --> 00:06:12,730 | 引起了更多人的注意力 |  |
| 158 | 00:06:13,010 --> 00:06:16,110 | 第一个就是Jim Keller加入他们 |  |
| 159 | 00:06:16,271 --> 00:06:17,310 | 并担任CTO |  |
| 160 | 00:06:17,590 --> 00:06:18,810 | 第二个就是Jim Keller |  |
| 161 | 00:06:18,862 --> 00:06:22,190 | 参与并主导了他们下一代 |  |
| 162 | 00:06:22,390 --> 00:06:26,303 | 下一代SOC的芯片的设计 |  |
| 163 | 00:06:26,521 --> 00:06:29,201 | 并选择了x280作为他们的主控CPU |  |
| 164 | 00:06:29,806 --> 00:06:34,634 | 最近关于他有一篇采访 |  |
| 165 | 00:06:34,784 --> 00:06:36,345 | 我读了一下 |  |
| 166 | 00:06:36,345 --> 00:06:39,153 | 我觉得里面他也谈到 |  |
| 167 | 00:06:39,153 --> 00:06:40,525 | 为什么选择x280 |  |
| 168 | 00:06:40,753 --> 00:06:42,825 | 我读下来的话有三个原因 |  |
| 169 | 00:06:42,925 --> 00:06:45,821 | 第一个就是他喜欢RISC-V |  |
| 170 | 00:06:45,880 --> 00:06:46,984 | 跟在座的各位一样 |  |
| 171 | 00:06:47,178 --> 00:06:50,640 | 他推崇RISC-V的精简 |  |
| 172 | 00:06:50,934 --> 00:06:51,728 | 模块化 |  |
| 173 | 00:06:52,009 --> 00:06:54,120 | 那然后就是没有历史包袱 |  |
| 174 | 00:06:54,160 --> 00:06:55,220 | 也没有额外的 |  |
| 175 | 00:06:55,318 --> 00:06:57,740 | 一些历史的原因加进来的指令 |  |
| 176 | 00:06:57,884 --> 00:07:01,571 | 他也同样喜欢RISC-V的开放性 |  |
| 177 | 00:07:01,778 --> 00:07:04,927 | 第二点来讲就是x280的标量 |  |
| 178 | 00:07:05,153 --> 00:07:06,605 | 矢量的计算能力 |  |
| 179 | 00:07:06,662 --> 00:07:08,343 | 还有SNP的多核 |  |
| 180 | 00:07:08,578 --> 00:07:09,853 | PPA这边 |  |
| 181 | 00:07:09,853 --> 00:07:13,662 | 都非常符合他们下一代SoC的需要 |  |
| 182 | 00:07:13,878 --> 00:07:15,945 | 第三点就是 |  |
| 183 | 00:07:15,945 --> 00:07:17,585 | 软件 软件 软件 |  |
| 184 | 00:07:17,731 --> 00:07:19,840 | Jim Keller在这个专访里面 |  |
| 185 | 00:07:19,962 --> 00:07:22,165 | 称Chris Lattner为 |  |
| 186 | 00:07:22,206 --> 00:07:23,703 | 这个星球上最好的 |  |
| 187 | 00:07:23,703 --> 00:07:25,165 | 编译器工程师之一 |  |
| 188 | 00:07:25,393 --> 00:07:30,471 | SiFive在Jim Keller的带领下 |  |
| 189 | 00:07:30,471 --> 00:07:33,221 | 他在AI/ML |  |
| 190 | 00:07:33,221 --> 00:07:35,131 | 或者说在vector这边 |  |
| 191 | 00:07:35,131 --> 00:07:37,068 | 他不论是编译器 |  |
| 192 | 00:07:37,068 --> 00:07:38,732 | os中间库 |  |
| 193 | 00:07:38,732 --> 00:07:40,330 | 还是AI的框架 |  |
| 194 | 00:07:40,610 --> 00:07:42,170 | 他们都是最先推行 |  |
| 195 | 00:07:42,330 --> 00:07:45,168 | 也是最完整的 |  |
| 196 | 00:07:45,340 --> 00:07:46,765 | 那下面我们来看几个例子 |  |
| 197 | 00:07:47,406 --> 00:07:49,765 | 第一个就是我们提供一个工具 |  |
| 198 | 00:07:49,965 --> 00:07:51,577 | 让有些用户 |  |
| 199 | 00:07:51,753 --> 00:07:55,512 | 在传统的cmd的架构上面的 |  |
| 200 | 00:07:55,512 --> 00:07:57,715 | 投入的这些算法 |  |
| 201 | 00:07:57,956 --> 00:08:00,728 | 它可以直接的移到 |  |
| 202 | 00:08:01,237 --> 00:08:03,090 | RISC-V的vector上来 |  |
| 203 | 00:08:03,262 --> 00:08:07,000 | 这些代码都不用再重新去写 |  |
| 204 | 00:08:07,220 --> 00:08:08,620 | 只要加一个编译选项 |  |
| 205 | 00:08:11,031 --> 00:08:13,915 | 第二来讲对于ML |  |
| 206 | 00:08:13,915 --> 00:08:16,606 | 它比较核心的就是 |  |
| 207 | 00:08:16,606 --> 00:08:18,365 | INT8的矩阵相乘 |  |
| 208 | 00:08:18,628 --> 00:08:23,300 | 这个图里面可以很清楚的看得到 |  |
| 209 | 00:08:23,462 --> 00:08:27,805 | SiFive的x280有Intelligence指令扩展 |  |
| 210 | 00:08:27,981 --> 00:08:31,585 | 只是实现了RISC-V Vectors的话 |  |
| 211 | 00:08:31,775 --> 00:08:33,345 | 提升了多少倍呢提升了12倍 |  |
| 212 | 00:08:33,545 --> 00:08:35,790 | 那这个12倍我相信不仅仅是说 |  |
| 213 | 00:08:35,790 --> 00:08:38,245 | 你的硬件上要支持这些自定义指令 |  |
| 214 | 00:08:38,496 --> 00:08:40,125 | 而且一定是 |  |
| 215 | 00:08:40,125 --> 00:08:41,462 | 配套的编译器 |  |
| 216 | 00:08:41,462 --> 00:08:43,200 | 和配套的计算库 |  |
| 217 | 00:08:43,362 --> 00:08:45,655 | 都要配套起来 |  |
| 218 | 00:08:45,771 --> 00:08:47,534 | 才可以达到这么高的提升 |  |
| 219 | 00:08:47,534 --> 00:08:48,695 | 12倍的提升 |  |
| 220 | 00:08:50,462 --> 00:08:55,255 | 在移动端也非常popular的就是 |  |
| 221 | 00:08:55,255 --> 00:08:57,246 | 有一个model就是MobileNet v1 |  |
| 222 | 00:08:57,490 --> 00:09:00,384 | 这个图里面也可以看得到 |  |
| 223 | 00:09:00,384 --> 00:09:02,896 | 带矢量扩展 |  |
| 224 | 00:09:02,896 --> 00:09:04,785 | 比不带矢量的扩展 |  |
| 225 | 00:09:04,785 --> 00:09:06,778 | 标量的CPU提升多少倍 |  |
| 226 | 00:09:06,778 --> 00:09:07,565 | 提升24倍 |  |
| 227 | 00:09:08,068 --> 00:09:12,428 | x280在带有 |  |
| 228 | 00:09:12,505 --> 00:09:13,403 | 自定义扩展 |  |
| 229 | 00:09:13,403 --> 00:09:17,426 | 和仅仅是带矢量扩展的话 |  |
| 230 | 00:09:17,509 --> 00:09:18,589 | 提升多少倍 提升4倍 |  |
| 231 | 00:09:19,040 --> 00:09:20,155 | 那么total的来讲 |  |
| 232 | 00:09:20,215 --> 00:09:22,215 | x280比同系列的 |  |
| 233 | 00:09:22,415 --> 00:09:24,035 | 比同样的微架构的 |  |
| 234 | 00:09:24,036 --> 00:09:26,195 | 标量的Core提升了接近100倍 |  |
| 235 | 00:09:26,535 --> 00:09:27,406 | 那么这个100倍 |  |
| 236 | 00:09:27,406 --> 00:09:28,709 | 就是两个数量级 |  |
| 237 | 00:09:28,896 --> 00:09:30,075 | 它就可以改变 |  |
| 238 | 00:09:30,340 --> 00:09:33,895 | AI/ML SoC架构的玩法 |  |
| 239 | 00:09:36,296 --> 00:09:40,990 | 那在做AI/ML |  |
| 240 | 00:09:40,990 --> 00:09:42,605 | 像这样的框架 |  |
| 241 | 00:09:42,606 --> 00:09:43,545 | 就Tensorflow |  |
| 242 | 00:09:43,545 --> 00:09:44,987 | TensorFlow Lite这样的框架 |  |
| 243 | 00:09:44,987 --> 00:09:46,478 | 是非常非常popular |  |
| 244 | 00:09:46,703 --> 00:09:51,505 | SiFive在推出x280这个Core的时候 |  |
| 245 | 00:09:51,506 --> 00:09:55,000 | 已经打通了整个TensorFlow Lite的框架 |  |
| 246 | 00:09:55,156 --> 00:09:57,705 | 完全的支持了TensorFlow Lite |  |
| 247 | 00:09:57,785 --> 00:09:59,905 | 那所以我们的用户可以 |  |
| 248 | 00:10:00,037 --> 00:10:02,156 | 直接在TensorFlow TensorFlow Lite上 |  |
| 249 | 00:10:02,156 --> 00:10:04,009 | 去开发和部署他们的 |  |
| 250 | 00:10:04,009 --> 00:10:05,421 | AI/ML的应用 |  |
| 251 | 00:10:05,696 --> 00:10:08,145 | 支持TensorFlow Lite的话 |  |
| 252 | 00:10:08,145 --> 00:10:12,756 | 就需要支持【b16】这样的data type |  |
| 253 | 00:10:14,543 --> 00:10:16,078 | 最后做一个总结 |  |
| 254 | 00:10:16,243 --> 00:10:20,085 | x280是我们StarFive和SiFive一起 |  |
| 255 | 00:10:20,085 --> 00:10:23,545 | 给用户针对于AI/ML |  |
| 256 | 00:10:23,545 --> 00:10:25,965 | 这样的应用推出来的新的CPU IP |  |
| 257 | 00:10:26,109 --> 00:10:30,675 | 它不仅实现了最新最标准的 |  |
| 258 | 00:10:30,937 --> 00:10:32,995 | RISC-V的矢量扩展 |  |
| 259 | 00:10:33,175 --> 00:10:34,415 | 而且还加入了 |  |
| 260 | 00:10:34,535 --> 00:10:36,431 | 特别针对于AI/ML |  |
| 261 | 00:10:36,431 --> 00:10:38,815 | 做优化的自定指令 |  |
| 262 | 00:10:39,056 --> 00:10:41,303 | 它可以增加performance |  |
| 263 | 00:10:41,303 --> 00:10:42,312 | 同时减少功耗 |  |
| 264 | 00:10:42,628 --> 00:10:46,390 | 那然后的话 |  |
| 265 | 00:10:46,600 --> 00:10:50,240 | x280特别适合目前正在演进的 |  |
| 266 | 00:10:50,280 --> 00:10:53,280 | ML这样的不同的框架 |  |
| 267 | 00:10:53,525 --> 00:10:57,175 | 同时x280非常易于大家 |  |
| 268 | 00:10:57,175 --> 00:10:59,140 | 在SoC里面直接去使用它 |  |
| 269 | 00:10:59,246 --> 00:11:02,640 | 最后一点说这个软件 |  |
| 270 | 00:11:02,793 --> 00:11:06,583 | 那就是软件第一软件优先的策略 |  |
| 271 | 00:11:06,812 --> 00:11:08,071 | 也是AI的需要 |  |
| 272 | 00:11:08,331 --> 00:11:10,178 | 也是我们StarFive和SiFive |  |
| 273 | 00:11:10,178 --> 00:11:12,153 | 一起所秉持的策略 |  |
| 274 | 00:11:12,415 --> 00:11:14,846 | 最后x280现在已经是 |  |
| 275 | 00:11:14,846 --> 00:11:16,237 | 公开可以授权的状态 |  |
| 276 | 00:11:16,415 --> 00:11:17,975 | 如果大家有兴趣可以 |  |
| 277 | 00:11:17,975 --> 00:11:20,311 | 到我们赛昉的展台上来洽谈 |  |
| 278 | 00:11:20,906 --> 00:11:22,240 | 今天我的分享到这里 |  |
| 279 | 00:11:22,240 --> 00:11:22,751 | 谢谢大家 |  |