基于共享 Cache 划分的电力芯片能耗优化技术

姚 浩[1,2],黄开天[1,2],余宏洲[3],王 轲[4]

(1. 南方电网数字电网研究院有限公司,广东 广州 510670; 2. 南方电网科学研究院有限责任公司,广东 广州 510663;
3. 浙江大学信息与电子学院,浙江 杭州 310000;4. 浙江大学电气工程学院,浙江 杭州 310000)

摘 要:提高电力终端芯片工作效率的同时降低其能耗,是优化智能电网系统的研究方向之一。首先针对 MPSoC 中高速缓存数据的高效管理问题,开展多处理器共享高速缓存划分(CP)技术研究,利用曲线拟合技术对高速缓存建模,通过数学方法求解 CP 问题;然后基于得到的缺失率曲线,根据共享高速缓存的缺失率与子系统能耗之间的数学关系,得出子系统能耗的数学表达;最后结合处理器能耗模型,综合全局求出最优的 CP 方案。实验验证表明使用求得的 CP 方法,处理器子系统能耗是进行优化前的子系统能耗的 27.9%。

关 键 词:共享高速缓存划分技术;缺失率;曲线拟合;子系统能耗

DOI:10.19781/j.issn.1673-9140.2021.05.004 **中图分类号**:TM763 **文章编号**:1673-9140(2021)05-0028-07

Shared Cache partition-based optimization technology for the power chip energy consumption

YAO Hao[1], HUANG Kaitian[1], YU Hongzhou[2], WANG Ke[3]

(1. Digital Grid Research Intitute, CSG, Guanzhou 510670, China; 2. Electric Power Research Institute, CSG, Guangzhou 510663, China;
3. College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310000, China;
4. College of Electrical Engineering, Zhejiang University, Hangzhou 310000, China)

Abstract: Improving the working efficiency of power terminal chips while reducing their energy consumption is one of the research direction for optimizing smart grid systems. Aiming at the efficient management of cache data in MPSoC, the multi-processor shared cache partitioning technology is studied. The curve fitting technology is utilized to model the cache, and mathematical methods is incorporated to solve the CP problem. The mathematical expression of the energy consumption in subsystem can be obtained according to the mathematical relationship between the obtained missing rate curve of the shared cache and the energy consumption of the subsystem. Combined with the energy consumption model of the processor, the comprehensive optimal CP solution is generated. Experimental verification shows that the processor subsystem energy consumption can be reduced to 27.9% of the subsystem before optimization using this CP method.

Key words: Cache partitioning; missing rate; curve fitting; subsystem energy consumption

智能变电站的低时延要求推动了芯片化保护,电力监控系统的物联网化也对芯片低功耗提出了相应的要求[1-3]。在智能电网中使用含有电力芯片的智能电力终端,是电力系统信息化进程中的重要一

收稿日期:2019-08-10;**修回日期**:2019-09-02

基金项目:国家重点研发计划(2018YFB0904900,2018YFB0904902)

通信作者:王 轲(1988-),男,博士后,主要从事多处理器 SoC 设计研究;E-mail:wangke@vlsi.zju.edu.cn

环。为了保证芯片的性能和功耗，智能电网中的电力终端芯片应用了学术界和工业界广泛认可的片上多处理器架构（chip multiprocessor architecture，CMP）。目前 CMP 产品的应用领域已经发展到电力系统中使用最广的嵌入式电子系统，电力终端中的嵌入式电子系统则大多通过电池供电，如智能电表等[4]。因此，找到一种能降低 CMP 能耗并保证电力芯片数据管理效率的技术具有非常重要的意义。

片上多处理器中有多个任务同步运行时，不同任务可能会将数据存在高速缓存的同一位置[5]，使得不同任务对某个 Cache line 反复读写，降低高速缓存读取效率的同时导致大量功耗损失。该文所研究的共享高速缓存划分（cachepartitioning，CP）技术将很有望解决这一难题。Cache 划分技术是按照一定规则对高速缓存进行适当地划分，以解决应用之间的冲突，从而提高系统中高速缓存的使用效率。应用 Cache 划分技术后，将高速缓存划分为不同组以供不同任务存取，并考虑对 Cache line 的填入逐出策略[6-7]，有效地减少高速缓存访问的冲突，达到提高数据读取效率的同时降低芯片功耗的目的。

当前国内外对 Cache 划分技术的研究方向和目标不尽相同，一般有降低能耗[8-9]、保证可靠性[9]、降低缺失率[10]等。文献[11]提出对高速缓存利用率进行监控，并结合实时的监控情况对共享高速缓存进行在线划分，以达到降低缺失率的目的；文献[12]以提高系统高速缓存使用效率为目的，提出了一种基于不同任务对共享高速缓存不同的带宽要求来进行划分的方法；文献[13]基于 Cache 划分的公平性与其吞吐量间的关系，分别提出了动态与静态的 Cache 划分方法，以确保不同应用之间的公平性。上述研究从各个角度对 Cache 划分问题进行了研究，但缺少一种从整体分析以达到最优的划分方法。

实际上，共享高速缓存的缺失率与其功耗并不一定成严格的正相关，即降低缺失率，子系统的功耗不一定下降。该文首先分析共享高速缓存的容量与缺失率间的关联关系，并将离散的缺失率拟合成曲线，得到其表达函数；然后利用高速缓存子系统的能耗与缺失率的数学关系得出子系统能耗的函数表达，使对抽象的 Cache 划分问题的研究转变为对函数极值的求解；最后得到整体最优并且快速有效的分配方案。

1　Cache 硬件模型与子系统能耗模型

1.1　多核处理器高速缓存硬件架构

高速缓存是主存和处理器之间的一级存储器，由静态随机存储器（static random access memory，SRAM）组成，是最靠近处理器的存储层次，高速缓存利用了时间和空间的局部性原理来提高数据读取速度。主流的多核处理器 Cache 架构通常包含私有的第 1 级 L1 Cache 和共享的第 2 级 L2 Cache。私有的 L1 Cache 是只能由一个指定 CPU 访问的高速缓存，而 L2 Cache 则可由多个处理器访问。私有高速缓存虽然能避免不同应用之间的竞争，但容量较小，缺失率较高。共享高速缓存的容量较大，但应用之间可能会发生冲突，而 Cache 划分技术能有效解决这个问题。该文采用的 CMP 硬件架构如图 1 所示。

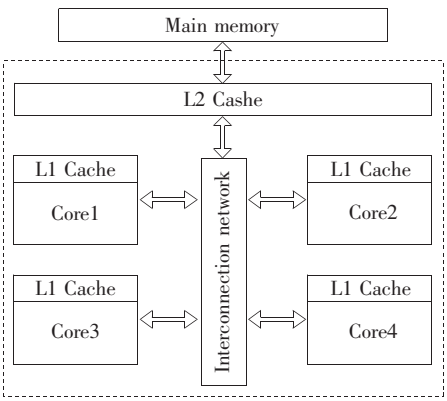

图 1　有 2 级 Cache 的多核处理器架构

Figure 1　Architecture of multi-core processor with two levels of cache

图 1 中每个核都有一个私有 L1 Cache，各核之间又通过互联网络共享 L2 Cache。这些 Cache 都在多核处理器架构内，架构内的 L2 Cache 再与主存储器互联。因为该文研究的是共享高速缓存，为了表述的简洁，文中出现的共享高速缓存均用 Cache 代替。

1.2　高速缓存子系统能耗模型

该文采用以下 Cache 能耗模型来表示 Cache 的子系统的能耗。在此能耗模型中，子系统的能耗为

$$E = E_d + E_s \tag{1}$$

式中 E_s、E_d 分别为子系统的静态能耗和动态能耗，其中静态功耗表达较简单。由于静态功耗是始终存在，则子系统在一个任务执行期间的静态能耗为

$$E_s = E_{sc} \cdot N_c \tag{2}$$

式中 E_{sc} 为 Cache 的子系统每个周期的静态能耗；N_c 为执行一个任务所需要的周期数。动态能耗可表达为 Cache 命中时和缺失时的耗能之和，Cache 每次缺失过程中所消耗的能量为

$$E_m = E_a + E_w + E_f \tag{3}$$

式中 E_a 为访问主存过程中的耗能，若 Cache 缺失，则访问主存以读取缺失的块；E_w 为等待主存数据返回过程中的能耗；E_f 为数据从主存返回后还要填入 Cache line 过程中的耗能。

Cache 命中时的能耗为每次命中所消耗的能量 E_h 与命中次数 N_h 的乘积，Cache 缺失时的能耗为每次缺失所消耗的能量 E_m 与缺失次数 N_m 的乘积。因此在能耗模型中，Cache 的动态能耗为

$$E_d = E_h \cdot N_h + E_m \cdot N_m \tag{4}$$

2 基于曲线拟合的 CP 技术

该文将曲线拟合技术用于对 Cache 的建模[14]。首先统计在不同的 Cache 容量下的缺失率，然后将容量和缺失率之间的关系用函数进行曲线拟合，并结合前述的能耗模型，最后用数学方法求解这个 Cache 的划分及子系统能耗优化问题。

2.1 对缺失率的曲线拟合

为了对 Cache 的缺失率进行曲线拟合，首先统计给应用划分的 Cache way 数量不同时所对应的 Cache 缺失次数、命中次数以及总指令条数，然后根据统计结果计算出不同的 Cache way 对应的缺失率。

为了将离散的 Cache 缺失率数据拟合为连续的曲线，需要选取适合拟合 Cache 缺失率的函数类型。当 Cache 的容量增大时，缺失率按照“sqrt2 rule”[15]变化，可以将 Cache 的缺失率拟合成幂函数；而在 Cache 处于饱和区的情况下，又可用线性函数来进行拟合。此外，在缺失率的不同子区域，可能适用不同的曲线来进行拟合，所以在此要判断在 Cache 缺失率的拟合曲线中是否存在分界点，并找到该点。一组 Cache way 数量和 Cache 缺失率的实例，如图 2 所示。

使用表 1 中的算法来寻找该曲线中的分界点。根据该算法，可得到拟合曲线的函数形式可以是指数为$(1-\sqrt{2})$(但实际结果为$-0.7\sim-0.3$)的幂函数或线性曲线。

上述算法输出的是分界点处的 Cache way 数。该算法中，$N_m(n)$ 为 Cache 在应用划分到 n 个 Cache way 时缺失的次数；A_{ve_dec} 为随着 Cache way 的增加，缺失次数的增量的平均值。

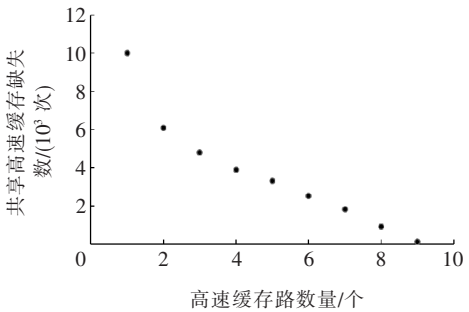

图 2 Cache 缺失实例

Figure 2 An instance of Cache miss

表 1 寻找分界点算法

Table 1 Algorithm to find inflexions

Input:	$N_m(n)(k=1,2,\cdots,W)$
1	W = the total number of ways of the shared cache;
2	K = the ratio of diversity;
3	$A_{ve\_dec} = (N_m(1) - N_m(W))/W$;
4	$I_{NF} = W - 2$;
5	while($I_{NF} > 2$){
6	$i=0$; $j=0$;
7	for($n = 2$; $n \leqslant W-1$; $n++$){
8	dec0 = $N_m(n-1) - N_m(n)$;
9	dec1 = $N_m(n) - N_m(n+1)$;
10	thre = $K \cdot A_{ve\_dec}$;
11	if((dec0>thre) && (dec1<thre)){
12	infl(i)= n ; $i++$;}
13	else if((dec0<thre) && (dec1>thre)){
14	inf2(j)= n ; $j++$;}}
15	$I_{NF} = i + j$;
16	increase K ;}

以下再使用另一算法来对分界点分割开的不同的子区域进行曲线拟合。

表 2 中的算法对不同子区域分别进行曲线拟合，图 3 是通过算法找到分界点后对子区域分别拟合得到的拟合曲线实例。

上述算法中输入 Cache way 的个数及与之对应的 Cache 缺失次数，计算后输出 Cache 缺失率函数。在算法输出时，线性函数的优先级要高于幂函数，这是由于幂函数要比线性函数多进行一次转化，拟合曲线的精确度会因此下降。

表 2　曲线在子区域的拟合算法

Table 2　Algorithm of curve fitting in a sub-region

Input	(x,y)
1	$(a_1,b_1,r_1)=\text{linear\_fit}(x_1,\cdots,x_n,y_1,\cdots,y_n)$;
2	if($\lvert r_1\rvert\geqslant T$){
3	$Y=a_1\cdot X+b_1$;}
4	else{
5	for($i=1$; $i\leqslant n$; $i++$){
6	$x_i'=\log_2(x_i)$; $y_i'=\log_2(y_i)$;}
7	$(a_2,b_2,r_2)=\text{linear\_fit}(x_1',\cdots,x_n',y_1',\cdots,y_n')$;
8	if(($-0.7<a_2<-0.3$) && ($\lvert r_2\rvert\geqslant T$)){
9	$Y=2^{b_2}\times X^{a_2}$;}
10	else{
11	$(a_3,b_3,r_3)=\text{linear\_fit}(x_1,\cdots,x_n,y_1,\cdots,y_n)$;
12	$Y=a_3\cdot X+b_3$;}
13	linear_fit($x_1,\cdots,x_n,y_1,\cdots,y_n$){
14	$a=\dfrac{\overline{x\cdot y}-\overline{x}\cdot\overline{y}}{\overline{x^2}-\overline{x}^2}$;
15	$b=\overline{y}-a\overline{x}$;

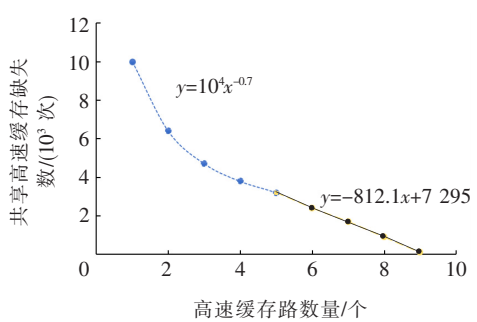

图 3　Cache 缺失率的拟合结果

Figure 3　Curve fitting result of cache miss rate

2.2　CP 问题与能耗优化问题

得到缺失率的拟合曲线后，可以将此函数应用于 Cache 子系统的能耗模型中。该文在对子系统的能耗计算时，使用了文献[16]中的能耗模型。

根据文献[16-18]可知，在此假设 Cache 的动态、静态能耗与 Cache 被使能的容量成正比。设 r 为 Cache 的缺失率，设总共访问 Cache 的次数为 N_a，结合 1.2 节所述的能耗模型，可以得到 Cache 子系统的动态能耗和静态能耗为

$$E_d=E_h\cdot N_a\cdot(1-r)+E_m\cdot N_a\cdot r \quad (5)$$

$$E_s=E_{sc}\cdot(N_{c\_a}+N_{c\_n}) \quad (6)$$

式(5)、(6)中　N_{c_a}、N_{c_n} 分别为访问存储器和不访问存储器的周期数。

根据前文描述可知，Cache 的动态功耗和静态功耗都与 Cache 被使能的容量的大小成正比，即

$$E_h=(S_{en}/S_t)\cdot E_{hit\_en} \quad (7)$$

$$E_{sc}=(S_{en}/S_t)\cdot E_{sc\_en} \quad (8)$$

式(7)、(8)中　E_{hit_en} 为当 Cache 全部被使能时命中一次的能耗；E_{sc_en} 为所有 Cache 都被使能时一个周期内的静态能耗；S_t 为 Cache 的总大小；S_{en} 为 Cache 中被使能部分的大小。

由于该文通过对 Cache way 进行划分来完成 Cache 的分配，因此以被用到的 Cache way 个数与总共的 Cache way 个数之比作为 Cache 被使能的容量与总容量之比。Cache 的缺失次数与分配的 Cache way 数量关系可由上文得到的曲线拟合的结果得到，Cache 的缺失率 r 就等于缺失次数 N_m 与访问 Cache 的次数 N_a 之比。得出 Cache 子系统能耗与 Cache way 之间的函数关系为

$$E=(K_1\cdot N_m+K_2)\cdot(S_{en}/S_t)+K_3\cdot N_m \quad (9)$$

式中　K_1、K_2、K_3 均为常系数，其数值为

$$K_1=E_{sc\_en}\cdot(N_{c\_m}-N_{c\_h})-E_{hit\_en} \quad (10)$$

$$K_2=N_a\cdot E_{hit\_en}+E_{sc\_en}\cdot N_a\cdot[N_{c\_h}+N_{c\_na}\cdot N_{na}/N_a] \quad (11)$$

$$K_3=E_m=E_a+E_w+E_f \quad (12)$$

式中　N_{c_h}、N_{c_m} 分别为访问 Cache 时命中和缺失消耗的周期数；N_{na} 为非访问 Cache 的指令的数量，N_{c_na} 为非访问 Cache 的指令的平均执行周期数。

得到了子系统能耗与 Cache way 的函数关系后，在给应用划分 Cache way 时，需设定一定的约束条件，即

$$\sum S_i \leqslant S_t \tag{13}$$

式中 S_i 为每个应用划分到的 Cache way 数量。在这个约束条件下，求解子系统能耗的数学表达式的极值，就能得到子系统能耗最优情况下的 Cache 划分方案。

3 实验验证分析

3.1 应用场景分析

该文主要研究的是针对电力终端芯片的能耗优化。实验所采用的电力终端芯片架构由 MCU 及外围控制部分、电源部分、显示部分、计量部分和通信部分构成；挂载有包括继电器、各类采样器等多种电力工控相关的外设。

在电力终端的工作场景中，终端芯片主要执行的任务可以分为 2 类，控制类和计算类。控制类的任务如 ADC 采样、数据通信等主要是控制外设，而外设的地址为 non-cacheable，对该类任务分配 Cache 不会提高工作效率。而对于计算类的任务，可以通过分配 Cache 提高 CPU 的工作效率。该文实验使用测试应用程序来模拟计算类任务，对 Cache 划分进行实验研究。

3.2 实验平台

为了验证该文所提出的 Cache 划分方案的有效性，采用 GEM5 模拟器作为硬件平台，SPEC CPU 2017 作为应用。SPEC CPU 2017 标准测试程序集是业界广泛使用的由 SPEC 推出的最新一代标准测试程序集。这个程序集可以为处理器与存储器的子系统提供可靠的评估结果。

GEM5 模拟器是一款高度可配置、集成多种 ISA、CPU 模型的体系结构模拟器，是用于研究计算机体系结构的模块化平台。GEM5 为使用者提供了包括有序 CPU 的详细模型和无序 CPU 的详细模型在内的 4 种级别的 CPU 模型。该文对 GEM5 模拟器的实验配置中，每个处理器都包含一个 16 KB 的 L1 Cache 和一个 2 MB 的 L2 Cache，另有一个 512 MB 的主存。该文将使用能耗统计工具 CACTI 对 Cache 子系统的能耗进行分析。

3.3 实验分析

在实验中，将该文所提出的 Cache 划分方案与另外 2 种 Cache 划分方法进行对比，分别将 3 种划分方法用于 GEM5 模拟器的多核处理模型上，同时运行 SPEC CPU 2017 标准测试程序集中的多个应用，以此验证该文提出的分配方法的有效性。

该文采用拟合曲线来表示 Cache 的缺失率，首先要对曲线拟合出的缺失率与仿真实际得到的缺失率进行比较。在分配 1～16 个 Cache way 的情况下，上述 2 种方法求得的 Cache 缺失率的平均差值如表 3 所示。

表 3 拟合曲线的 Cache 缺失率偏差

Table 3 Cache miss rate deviation of fittedcurve %

测试应用编号	平均偏差	测试应用编号	平均偏差
401	1.68	456	0.15
403	10.87	464	0.39
429	2.93	471	0.70
445	0.53	483	13.87

由表 3 中的比较结果可知，大多数测试应用通过曲线拟合得到的 Cache 缺失率与仿真得到的差别很小，说明了该文采用的曲线拟合方法是可靠的。

实验中，将 GEM5 模拟器配置成双核、四核、八核架构，并将测试应用也分成相应的组数。实验中用到的另外 2 种 Cache 划分方法，方法 1 不对 Cache way 进行分配，Cache way 可被任何处理器访问；方法 2 对 Cache way 平均划分，每个处理器得到相同数量的 Cache way 且不能访问其余的 Cache way。方法 3 则是使用该文提出的基于曲线拟合的 Cache 划分方法求得的 Cache 划分方案。为保证实验结果的有效性和可靠性，实验中的测试应用都要执行 3×10^8 条指令。在实验中，2.1 节提到的寻找分界点算法中的变量 K 设置为 1，曲线拟合算法中的线性相关性阈值 T 设置为 0.7。

不同的 Cache 划分方法在运行不同的测试应用时，子系统能耗的对比情况，如图 4 所示。由于实验结果中用方法 1 时，Cache 子系统耗能最高，将其耗能归一化为 1，其余 2 种方法的结果也相应地进行归一化处理。

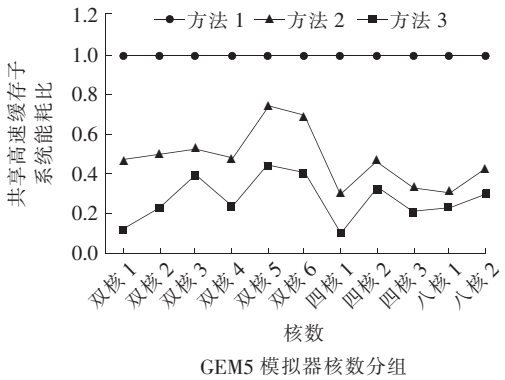

图 4　3 种划分方案的能耗对比

Figure 4　Comparison of energy consumption between three distribution schemes

由图 4 的实验结果可知，使用方法 2 运行测试应用平均消耗的能量是方法 1 的 48.3%，而用该文提出的方法 3 运行测试应用得出的平均耗能是方法 1 的 27.9%。由此可见，该文提出的基于曲线拟合的 Cache 划分方案优于传统的划分方法。

下面分别分析以上 3 种划分方法。方法 1 中没有对 Cache way 进行划分，Cache way 可以被任一处理器访问，因此 Cache way 都将被使能；方法 2 中将 Cache way 平均分配给处理器核，在运行测试应用时，Cache 的缺失率就已经比较低了，不需要将所有的 Cache way 使能，因此耗能相较于方法一有所下降；而该文提出的方法 3 会在应用运行时，根据各自的需求将 Cache way 激活，使被使能的 Cache way 数量更少，而测试应用运行时用不到的 Cache way 就会被关掉以减少耗能，所以方法 3 的能耗要少于前 2 种方法。

从图 4 中还可以观察到，GEM5 模拟器为四核和八核架构时，方法 2 和方法 3 之间的能耗差距比双核的更小，这是由于此时可以被关掉的 Cache way 的数量更少了。

4　结语

该文针对多核处理器同时运行多个应用时，应用对 Cache 访问时会发生冲突的问题，提出了一种基于曲线拟合的，用数学方法求解的 Cache 划分技术。利用曲线拟合得到缺失率曲线，并对 Cache 子系统能耗进行数学建模，将 Cache 的划分问题转化为对函数极值的求解问题，大大提高了求解问题的效率。

该文通过实验，首先将曲线拟合得到的缺失率与实际仿真得到的缺失率进行对比，证明拟合得到的缺失率是准确的；然后将提出的 Cache 划分方法与传统的划分方法进行对比；最后实验结果表明此方法能得到一种全局最优、子系统能量消耗最少的 Cache 划分方案，验证了所述方法的有效性。将此方法用于多核处理器架构的电力芯片中，能够有效地在降低处理器能耗的同时提高 Cache 的效率。

参考文献：

[1] 陈家璘，贺易，李磊，等. 泛在电力物联网传输网优化关键技术研究[J]. 中国电力，2019，52(12)：20-26+38.
CHEN Jialin，HE Yi，LI Lei，et al. Research on key optimization technologies for transmission network of ubiquitous power internet of things[J]. Electric Power，2019，52(12)：20-26+38.

[2] 刘喜梅，马俊杰. 泛在电力物联网在电力设备状态监测中的应用[J]. 电力系统保护与控制，2020，48(14)：69-75.
LIU Ximei，MA Junjie. Application of the ubiquitous power internet of things in state monitoring of power equipment[J]. Power System Protection and Control，2020，48(14)：69-75.

[3] 何奉禄，陈佳琦，李钦豪，等. 智能电网中的物联网技术应用与发展[J]. 电力系统保护与控制，2020，48(3)：58-69.
HE Fenglu；CHEN Jiaqi；LI Qinhao，et al. Application and development of internet of things in smart grid[J]. Power System Protection and Control，2020，48(3)：58-69.

[4] 侯兴哲，刘型志，郑可，等. 泛在电力物联网环境下新一代智能电能表技术展望[J]. 电测与仪表，2020，57(9)：128-131.
HOU Xingzhe，LIU Xingzhi，ZHENG Ke，et al. Technical prospect of a new generation smart meter in the ubiquitous power internet of things environment[J]. Electrical Measurement & Instrumentation，2020，57(9)：128-131.

[5] Chen G，Huang K，Huang J，et al. Cache partitioning and scheduling for energy optimization of real-time MPSoCs[C]//2013 IEEE 24th International Conference on Ap-

plication-Specific Systems, Architectures and Processors, Washington, USA: IEEE, 2013.

[6] Haque M S, Easwaran A. Predictability and performance aware replacement policy PVISAM for unified shared caches in real-time multicores[J]. IEEE Transactions on Computer-Aided of Integrated Circuits and Systems, 2018, 37(11): 2720-2731.

[7] Sridharan A, Seznec A. Discrete cache insertion policies for shared last level cache management on large multicores[C]//IEEE International Parallel and Distributed Processing Symposium, Chicago, USA: IEEE, 2016.

[8] Kedar G, Mendelson A, Cidon I. SPACE: semi-partitioned Cache for energy efficient, hard real-time systems[J]. IEEE Transactions on Computers, 2017, 66(4): 717-730.

[9] Xiang Y C, Wang X L, Huang Z H, et al. DCAPS: dynamic cache allocation with partial sharing[C]//Proceedings of the Thirteenth EuroSys Conference, New York, USA: ACM, 2018.

[10] Ahmad M, Dogan H, Checconi F, et al. Software-hardware managed last-level cache allocation scheme for large-scale NVRAM-based multicores executing parallel data analytics applications[C]//IEEE International Parallel and Distributed Processing Symposium, Vancouver, Canada: IEEE, 2018.

[11] Qureshi M K, Patt Y N. Utility-based cache partitioning: a low-overhead, high-performance, runtime mechanism to partition shared caches[C]//39th Annual IEEE/ACM International Symposium on Microarchitecture, Orlando, USA: ACM, 2006.

[12] Yu C J, Petrov P. Off-chip memory bandwidth minimization through cache partitioning for multi-core platforms[C]//Proceedings of the 47th Design Automation Conference, Anatheim, USA, 2010.

[13] Kim S, Chandra D, Solihin Y. Fair cache sharing and partitioning in a chip multiprocessor architecture[C]//Proceedings of 13th International Conference on Parallel Architecture and Compilation Techniques, Antibes, France: IEEE, 2004.

[14] Venkatesan V, Tay Y C, Zhang Y I, et al. A 3-level cache miss model for a nonvolatile extension to transcendent memory[C]//IEEE International Conference on Cloud Computing Technology and Science, Singapore: IEEE, 2015.

[15] Hartstein A, Srinivasan V, Puzak T R, et al. Cache miss behavior: is it $\sqrt{2}$?[C]//Conference on Computing Frontiers, Yorktown Heights, USA: ACM, 2006.

[16] Zhang C, Vahid F, Najjar W A. A highly configurable cache for low energy embedded systems[J]. ACM Transactions on Embedded Computing Systems, 2005, 4(2): 363-387.

[17] Wang W, Mishra P, Gordon-Ross A. Dynamic cache reconfiguration for soft real-time systems[J]. ACM Transactions on Embedded Computing Systems, 2012, 11(2): 1-31.

[18] Krisztián Flautner, Kim N S, Martin S M, et al. Drowsy caches: simple techniques for reducing leakage power[C]//29th International Symposium on Computer Architecture, Anchorage, USA: IEEE, 2002.