一种基于频繁值和位变换感知的数据总线节能方法

张铭泉[1,2],古志民[2],张吉赞[2]
(1. 华北电力大学计算机系,河北保定 071003;2. 北京理工大学计算机学院,北京 100081)

摘 要: 深亚微米工艺下,片上数据总线能耗占嵌入式多核芯片能耗的比重越来越大. FV-MSB(Frequent Value-Most Significant Bits)方法降低了片外数据总线的能耗,但对于非频繁值和频繁高位值的低位部分未做处理,为进一步降低片上总线动态能耗,设计了一种基于频繁值和位变换感知的片上总线节能方法. 利用频繁值和对位变换数的感知选择编码方式,大幅减少了数据总线上的位变换,有效降低了总线动态能耗. 70nm 工艺下,仿真实验结果显示,本文的方法最大节能比例可达 17.76%,平均节能比例达 16.91%,较 FV-MSB 方法使节能比例提高了 6.28%. 并且节能比例随 λ 的变化趋势表明本方法在未来工艺尺寸缩小时仍能发挥作用.

关键词: 频繁交换值; 频繁值缓存; 总线节能; 翻转码; 耦合电容
中图分类号: TP302 **文献标识码:** A **文章编号:** 0372-2112 (2017)08-1810-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.08.002

An Energy Consumption Saving Method Based on Frequent Values and Bit Switching-Aware for On-chip Data Bus

ZHANG Ming-quan[1,2], GU Zhi-min[2], ZHANG Ji-zan[2]
(1. *Department of Computer, North China Electric Power University, Baoding, Hebei* 071003, *China*;
2. *School of Computer Science and Technology, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: In the deep sub-micron technology domain, the energy consumption of data bus accounts for the energy consumption proportion of embedded multi-core chip is increasing. FV-MSB method was used to reduce the off-chip data bus energy consumption. However, it didn't deal with the non-frequent values and the lower parts of the frequent high-ordered bits values. In order to further reduce the dynamic energy consumption of which is induced by these values on the bus, an energy saving method based on frequent values and bit switching-aware for on-chip data bus is designed. Using the frequent values and the perception of the switching activities number, automatically selecting the encoding mode and so switching activities on the bus are significantly reduced, and therefore the bus dynamic energy is effectively reduced. With 70 nm technology, the results of simulation experiments show our method can improve the ratio of energy saving by about 17.76% and average ratio about 16.91%, and which is better than FV-MSB over 6.28 %. And the trend of the energy saving ratio with λ shows that the method can still play the role when the technology size is further reduced in the future.

Key words: frequent exchanged value; frequent value cache; bus energy consumption saving; invert encoding; coupling capacitance

1 引言

在深亚微米(DSM)工艺下,工艺尺寸的缩小增加了总线串扰,使线间耦合电容远高于对地自身电容,从而使来自耦合电容的功耗远高于对地自身电容引起的功耗. 考虑耦合电容影响的数据总线动态能耗在芯片能耗中占的比重越来越大,现有研究表明片上总线的能耗已与来自处理器和缓存等主要部件的能耗相当并已占到片上系统能耗的近 30%[1]. 为发挥嵌入式多核芯片简单、高效、低功耗的特点,降低片上数据总线的能

收稿日期:2016-01-31;修回日期:2016-05-23;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61370062)

耗势在必行.

总线节能技术被应用到芯片设计的不同层次上.物理层的解决方案主要有:位线尺寸优化,屏蔽和中继器的插入,电压调节等[2,3].其他技术集中在数据链路层,通过减少位线上的变换数和总线上的数据通信量来实现,以编码技术为主.针对地址总线的编码主要有:T0 编码 、工作区编码和格雷码及它们的改进版本等[4],由于地址按顺序存放,这些编码措施可大幅降低总线上的变换数,实现地址总线的高效节能.针对数据总线的编码主要有:翻转码[5,6]和频繁值编码[7,8]及与它们相关的编码等,由于数据总线上的数据规律性较差,前后数据关联性差,适用于地址总线的编码并不能给数据总线带来大的能耗收益.其他的编码,如局部编码及其改进技术[9],避免串扰的数据编码等,对优化总线能耗也收到了不同的效果,但它们的结构复杂引入的负载较大.

现有数据总线节能技术中,最著名的一个是由 M. Stan 等人提出的总线翻转码[5].当待发送值与其前一个值相比有超过半数的数据位同时要翻转时,发送该值的反码,否则发送原值,并用指示线指示.这样可有效减少总线上的位变换数,减少电容量,降低总线能耗.在此基础上,相关学者又提出了奇/偶数位翻转码和部分翻转码等扩展形式[6].这些编码技术结构简单,引入负载低,对数据总线节能有一定的效果.频繁值编码利用值的局部性进行总线能耗优化是另一个热门方向.Yang Jun 等人在文献[7]中提出了利用线下的优化算法,动态检测频繁值的频繁值编码技术,用于片外总线能耗的优化.然而,此方法要求频繁值缓存具有高的命中率,较低的命中率会降低节能效果.

文献[10]进一步开发了值的局部性,增加了频繁值最高有效位(MSB)缓存,与频繁值缓存一起构成 FV-MSB 编码,以减少片外总线的动态能耗.它在总线两端分别增加两部分缓存,一个存放频繁值,另一个存放值的频繁高位,工作原理同频繁值编码,当发送端同时命中两个缓存时,存放频繁值的缓存具有更高的优先级,该方法进一步降低了位变换,优化了片外数据总线动态能耗.然而,FV-MSB 编码在优化片上数据总线能耗时具有如下不足:(1)片上受限的面积决定了频繁值的数量不宜太大,降低了频繁值缓存的命中率,削弱了总线节能效果;(2)对存放于 MSB 中值的低位部分未做处理;(3)对总线上传输的非频繁值未做处理.这些因素使 FV-MSB 方法在优化片上数据总线能耗时的效果不明显.文献[11]提出了频繁交换值缓存(Frequent Exchanged Value Cache,FEVC)的方法来解决交叉开关的动态能耗优化问题,结果表明在不考虑耦合电容影响时,该方法可大幅降低交叉开关的动态能耗.FEVC 方法也可用于优化数据总线的动态能耗,但是,当考虑耦合电容对数据总线动态能耗带来的影响时,FEVC 的效果大打折扣.

本文研究的对象是 70nm 工艺下片上数据总线,此时耦合电容的影响不可忽略,实现总线动态能耗的优化采用以上方法均不能达到理想效果.为实现总线节能最大化,结合 FEVC 方法,本文设计了一种基于频繁值和位变换感知的片上数据总线节能方法(Multi-Code Frequent Value Cache,MCFVC),该方法可自动感知未被缓存的频繁值低位部分和非频繁值的位变换数大小,自动选择编码方式,使传输值的总变换距离(自身变换数和耦合变换数之和)最小,以实现最大程度的总线节能.

2　基于总线的 MCFVC 多核结构和能耗模型

2.1　基于总线的 MCFVC 嵌入式多核结构

本文采用的基于总线的多核结构,如图 1 所示.四核心 CMP 结构,每个核心有 4 路组相连一级私有指令缓存(IL1 cache)和数据缓存(DL1 cache)各 32KB,16 路 1MB 共享二级缓存(L2 cache).各级缓存行大小均为 64B,且为包含式缓存,每个核心和 L2 端各增加 1 个 MCFVC 模块.片上处理核之间及核与缓存间采用总线互连结构,这种共享带宽的总线交换方式在内核不多时,比其他交换方式更有优势.数据总线被不同核的 L1 cache 交替使用,从而达到访问共享 L2 cache 的目的.各处理核分别挂接在数据总线上,使每个核都能够和 L2 cache 相连,当有不同核需要同时访问 L2 cache 时会产生竞争,这时需要相应的仲裁机制来选择访问顺序以保证数据的一致性.

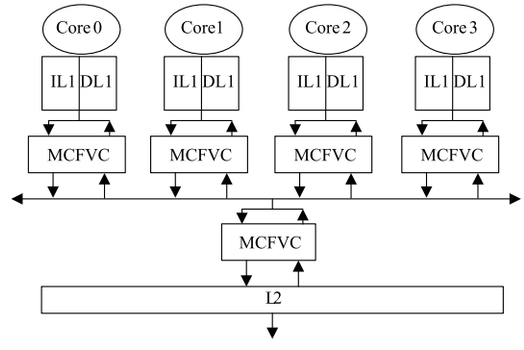

图1　MCFVC措施下的嵌入式片上多核结构

2.2　能耗模型

DSM 工艺下数据总线电容模型如图 2 所示.其中,C_L 表示对地自身电容,C_I 表示位线间耦合电容,线上的数字(1),(2)等表示位线编号,即图示数据总线共有 n 条并行位线.令 $\lambda = C_I/C_L$,对于不同的工艺尺寸,λ 的取值不同,工艺尺寸越小,λ 的值越大.本文研究的

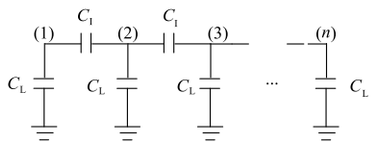

图2 DSM下的数据总线电容模型

70nm 工艺取 $\lambda = 5$.

片上数据总线动态能耗 E_{bus} 采用式(1)计算,E_L 表示自身对地电容引起的动态能耗,E_I 表示相邻位线间耦合电容引起的动态能耗. 引起数据总线动态能耗的主要因素是传输数据时位线上的变换数. 记总线位线上相邻周期前后数据的自变换数之和为垂直距离 X,同一周期内位线间的耦合电容引起的耦合变换数之和为水平距离 Y. 其中 $X_{i,j}$ 表示位线 i 上从第 j 个周期到第 $j+1$ 个周期时自身位变换数,传输数据由 0 变到 1 或从 1 变到 0 时,$X_{i,j}$ 为 1,否则为 0. n 表示传输数据的位线数,t 表示传输数据的周期数. $Y_{(i,i+1),j}$ 表示从第 j 个周期到第 $j+1$ 个周期时相邻位线 i 和 $i+1$ 之间的耦合变换数.

$$E_{bus} = E_L + E_I \tag{1}$$

$$E_L = C_L \cdot V_{DD}^2 \cdot \sum_{i=1}^{n} \sum_{j=1}^{t} X_{i,j} \tag{2}$$

$$E_I = C_I \cdot V_{DD}^2 \cdot \sum_{i=1}^{n-1} \sum_{j=1}^{t-1} Y_{(i,i+1),j} \tag{3}$$

而 $X = \sum_{i=1}^{n} \sum_{j=1}^{t} X_{i,j}$, $Y = \sum_{i=1}^{n-1} \sum_{j=1}^{t-1} Y_{(i,i+1),j}$ 则由式(1 ~ 3)可得

$$E_{bus} = (X \cdot C_L + Y \cdot C_I) \cdot V_{DD}^2 \tag{4}$$

把 $\lambda = C_I / C_L$ 代入式(4)整理得到式(5)

$$E_{bus} = (X + \lambda \cdot Y) \cdot C_L \cdot V_{DD}^2 \tag{5}$$

记数据总线上与能耗相关的总变换距离为 D_{enc},采用式(6)计算,

$$D_{enc} = X + \lambda \cdot Y \tag{6}$$

由式(5)(6)可知,为取得最小的 E_{bus} 只需使总变换距离 D_{enc} 最小.

3 MCFVC 总线节能方法

3.1 MCFVC 编码描述

MCFVC 多核结构如图 1 所示,每个核增加 1 个 MCFVC 模块,L2 端增加 1 个 MCFVC 模块,该模块按功能主要分为编码和解码两个部分,它们的结构分别如图 3 和图 4 所示. MCFVC 模块按结构分为频繁值缓存 FVC 和多编码模块 MC 两部分.

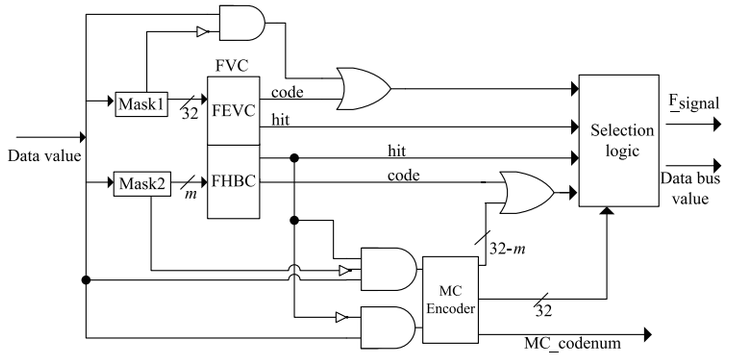

图3 MCFVC编码示意图

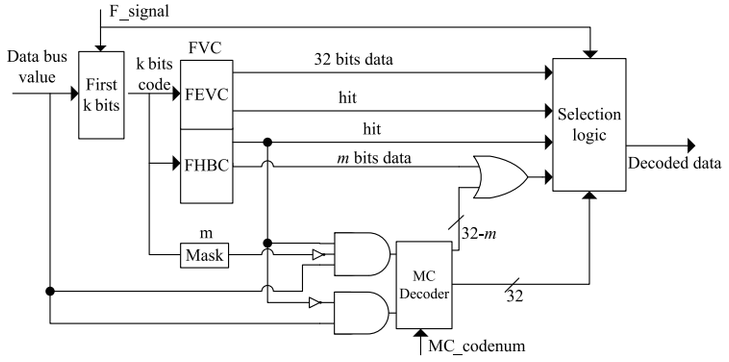

图4 MCFVC解码示意图

FVC 由频繁交换值缓存 FEVC(用于存放频繁交换值,完整值)和频繁高位值缓存 FHBC(用于存放频繁高位值,部分值)构成,两者都采用内容地址存储器(CAM). 为便于搜索,需要对待发送的值先使用掩码处理,得到与 FVC 中值长度相等的待搜索值,具体为 32 位完整值需要 32 位掩码,m 位高位值需要 m 位掩码. 为减少硬件复杂性和措施本身引入的能耗代价,在 MCFVC 结构中,每个 FVC 存放的值和值在其中的位置(索引)都相同,并在程序运行期间保持不变. 如果 FVC 中的值保持动态变化需要引入大量硬件电路,会抵消掉编码措施带来的收益. 每个 FVC 中存放 4 个频繁完整值(FEVC)和 4 个频繁值高位部分值(FHBC). 存放值的个数是由片上受限的面积和频繁值的覆盖率来确定的.

FVC 受限的容量空间决定了其中存放值的个数很少,传输值中必然还有很多值(非频繁值),无法在 FVC 中命中,这些非频繁值和只在 FHBC 中命中的值的低位部分,如果不加处理,也会引起大量的位变换,造成可观的总线能耗,为此,本文引入由四种编码方式,即原码,翻转码,奇数位翻转码和偶数位翻转码构成的多编码模块 MC,对未存储在 FVC 中的部分值和非频繁值进行处理,进一步减少位变换数. 对于进入 MC 编码器的数值,编码器根据式(6)计算总变换距离的大小,自动确定编码方式对数值进行编码,同时生成编码代号发送至接收端.

在设计中还要确定指示线的条数和 FHBC 中存放值的长度 m,下面分别论述这两个问题.

(1)指示线. 使用 1 位指示线区分频繁值和非频繁值及频繁值的来源. 当 F_signal 置 1 时表示频繁值,置 0 时表示非频繁值,并且利用总线高位位线传输的索引值来确定频繁值是来自 FEVC 还是 FHBC. 而在 MC 编码器中需要指示频繁值低位部分或非频繁值采用的编码方式,本文采用了四种编码方式,需要 2 位编码指示线. 这样,MCFVC 方法以数据总线节能最大化为目标,共增加了 3 位指示线.

(2)m 值的确定. 直观上看,m 越小 FHBC 的命中率越高,但是更高的命中率不一定总能带来更少的位变换数和更低的能耗. 虽然较大的 m,将导致命中率的下降,但是每次命中带来的收益将可能增加. 选择 m 的值需要以能耗收益为依据,通过实验我们发现当 m 取 4 时可使收益最大化. 因此,FHBC 中存储值的长度 m 取 4.

3.2　MCFVC 编码结构与算法

用 B^j 表示位宽为 n 的数据总线在第 j 个周期待传输的数据,表示为 $B^j=(b_n^j,b_{n-1}^j,b_{n-2}^j,\cdots,b_1^j)$,$B^{(j-1)\text{enc}}$ 表示第 $j-1$ 个周期在数据总线上传输的编码数据. MC 采用四种编码方式:记 $B^{(\text{org})}$ 为值的原始编码,编码代号为 00;原始值反码 $B^{(\text{inv})}$ 为翻转码编码,编码代号 11;原始值奇数位取反 $B^{(\text{odd})}$ 为奇数位翻转码,编码代号 01;原始值偶数位取反 $B^{(\text{evn})}$ 为偶数位翻转码,编码代号 10. 两位编码指示线用于传递编码代号,指示解码器正确解码. 采用函数 $\text{ST}\__n(d1,d2)$ 计算两个数据位数为 n 的数据值间的垂直距离 X,采用函数 $\text{CT}\__n(data)$ 计算位数为 n 的数据值的水平距离 Y,用 M_d 表示本次传输编码数据的最小总变换距离.

算法 1 为 MCFVC 编码算法,图 3 为 MCFVC 编码示意图. 发送端待传输数据经过掩码处理后进入 FVC 中进行搜索,为减少延迟我们在两个缓存中并行搜索. 如果值在 FEVC 和 FHBC 中同时命中,则 FEVC 有更高的优先级,因为它能减少更多的位变换,此时用值在 FEVC 中的索引代替原值发送. 当值仅在 FHBC 中命中时,由于 FHBC 中存放的是值的高位部分,此时将得到值的高位在 FHBC 中的索引,其低位部分进入 MC 编码器(MC Encoder),编码器调用算法 2 自动选择编码方式,把得到的低位编码值与其在 FHBC 中的高位编码一起送入选择器,同时生成采用的编码代号 MC_codenum. 如果在两个缓存中都没有命中,待发送值将直接进入 MC 编码器,由编码器调用算法 2,自动选择使总距离最小的编码方式,把得到的编码值,送入选择器,同时生成采用的编码代号. 最后选择逻辑根据编码方式的优先级,确定在总线上传输的编码值和指示信号 F_signal,同时发送的还有 MC 编码器产生的编码代号 MC_codenum.

算法 1　MCFVC 编码算法

```
1. FOR each data value DO
2.    search for data value in FVC
3.    IF hit in FVC THEN
4.        F_signal ← 1;
5.        upper code ← code for hit entry in FVC;
6.        IF upper code < =3 THEN
7.           current data bus value ← upper code;
8.         ELSE
9 .       lower data value←Mask & data value;
10.        lower code ← MC(lower data value,32 - m);
11.          mc_code ← MCcodenum;
12.          current data bus value ← upper code OR lower code;
13.       END IF
14.    ELSE
15.   current data bus value ← MC(data value,32);
16.   F_signal ← 0;
17.   END IF
18. END FOR
```

算法 2 为 MC 编码函数算法,根据式(6),选择总变换距离最小的编码方式,返回编码值和编码代号.

算法 2 MC 编码函数算法 MC(*value*,*n*)

1. $B^{(org)}$←org(*value*);//The true code of value;
2. $B^{(inv)}$←invert(*value*);//Invert code;
3. $B^{(odd)}$←oddinvert(*value*);// Odd invert code;
4. $B^{(evn)}$←eveninvert(*value*);//Even invert code;
5. D_{org}←$ST_{-n}(B^{(org)},B^{(j-1)enc})+\lambda\cdot CT_{-n}(B^{(org)})$;
6. D_{inv}←$ST_{-n}(B^{(inv)},B^{(j-1)enc})+\lambda\cdot CT_{-n}(B^{(inv)})$;
7. D_{odd}←$ST_{-n}(B^{(odd)},B^{(j-1)enc})+\lambda\cdot CT_{-n}(B^{(odd)})$;
8. D_{evn}←$ST_{-n}(B^{(evn)},B^{(j-1)enc})+\lambda\cdot CT_{-n}(B^{(evn)})$;
9. M_d←Min (D_{org},D_{inv},D_{odd},D_{evn});
10. IF $M_d==D_{org}$ THEN
11. *Thiscode* = $B^{(org)}$;
12. *MCcodenum* ← 00;
13. END IF
14 IF $M_d==D_{odd}$ THEN
15. *Thiscode* = $B^{(odd)}$;
16. *MCcodenum*←01;
17. END IF
18. IF $M_d==D_{evn}$ THEN
19. *Thiscode* = $B^{(evn)}$;
20. *MCcodenum*←10;
21. END IF
22. IF $M_d==D_{inv}$ THEN
23. *Thiscode* = $B^{(inv)}$;
24. *MCcodenum* ←11;
25. END IF
26. RETURN *Thiscode*,*MCcodenum*;

3.3 MCFVC 解码结构与算法

算法 3 为 MCFVC 解码算法,图 4 为 MCFVC 解码示意图. 当传输频繁值时总线上传输编码值的高 k 位(本文中 $k=3$)表示频繁值在 FVC 中的索引. 当传输非频繁值时,总线所有位线上传输的都是编码数据位. 解码器接收到编码值后,首先判断频繁值指示信号 F_signal,当 F_signal 为 1 时,表明接收到的值为频繁值,解码器根据总线前 k 位位线上的值来判断值在缓存中的位置. 如果前 k 位值对应频繁完整值索引,则从 FEVC 中得到原值. 如果前 k 位值对应频繁高位部分值索引,则从 FHBC 中取出原值的高位部分,同时由 MC 解码器调用算法 4,根据编码代号 MC_codenum,得到原值的低位部分,最后由两部分一起恢复出原值,完成解码. 当 F_signal 为 0 时,则接收到的值为非频繁值,直接由 MC 解码器调用算法 4,根据编码代号 MC_codenum 解码获得原值.

算法 3 MCFVC 解码算法

1. FOR *each data bus value* DO
2. IF *F_signal* = =1 THEN
3. *upper code* ← upper_code_index[*data bus value*];
4. IF *upper code* < =3 THEN
5. *data value* ← data[*upper code*];
6. ELSE
7. *lower data* ← UNMC(*lower data bus value*,*mc_code*);
8. *upper data* ← data[*upper code*];
9. *data value* ←*upper data* OR *lower data*;
10. END IF
11. ELSE
12. *current data value* ← UNMC(*data bus value*,*mc_code*);
13. END IF
14. END FOR

算法 4 为 MC 解码函数算法,根据编码代号解码出原传输值返回.

算法 4 MC 解码函数算法 UNMC(*value*,*S*)

1. IF *S* = =00 THEN
2. *Thisdata*← org(*value*);
3. END IF
4. IF *S* = =01 THEN
5. *Thisdata* ← odd invert(*value*);
6. END IF
7. IF *S* = =10 THEN
8. *Thisdata*←even invert(*value*);
9. END IF
10. IF *S* = =11 THEN
11. *Thisdata*← invert(*value*);
12. END *IF*
13. RETURN *Thisdata*;

3.4 MC 编解码结构

图 5 为 MC 编码结构示意图. n 位输入数据同时进入 4 个不同的编码处理部件,由这些部件生成 4 种编码数据,其中附加的后两位为编码指示位,接着各 $n+2$ 位编码数据被送入距离评估计算器,得到总变换距离,然后送入距离比较器,由比较器选出总距离最小的编码数据 $B^{j(enc)}$,同时生成编码代号. 距离评估计算器主要由异或门和锁存器构成. 图 6 为 MC 解码结构示意图,主要由 1 个 2-4 译码器和 4 个解码线路构成. 2-4 译码器的输入是编码指示线 a 和 b 传来的编码代号,译码后确定相应部件接收的数据有效,有效的解码部件解码后输出 n 位原始值. 其中,buffer 部件缓存 n 位数据值,Inverter 对数据值取反,Odd inverter 对数据值奇数位取反,同理 Even inverter 对数据值偶数位取反,它们主要由反相器和异或门构成.

4 仿真实验及结果分析

4.1 仿真实验环境及测试基准程序

为验证 MCFVC 方法的有效性,本文在图 1 结构下

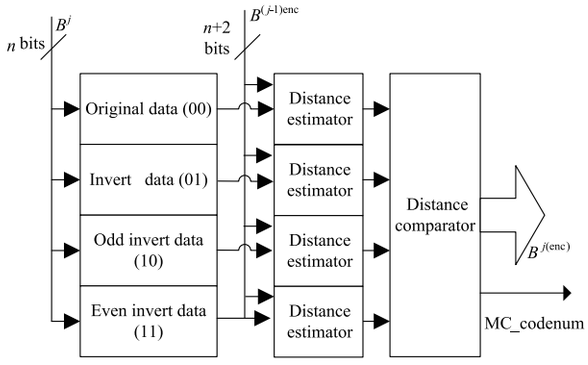

图5　MC编码结构示意图

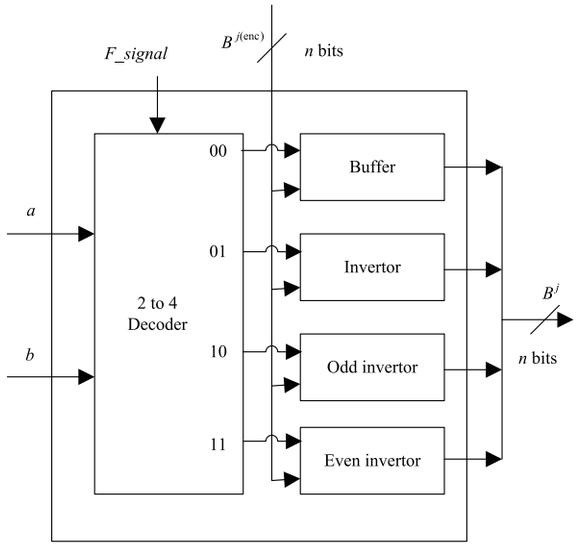

图6　MC解码结构示意图

进行了仿真实验. 表 1 列出了初始参数,为考察 MCFVC 在不同情况下的节能效果,实验中对部分参数做了调整.

表 1　初始实验参数

参数	值
时钟频率	500MHz
处理核数	4
IL1,DL1 大小	32KB
缓存行大小	64B
L1 相连度	4 路
FVC 每次访问能耗	18.6pJ
总线宽度	32
总线每线能耗	11.6pJ
耦合因子 λ	5

本文选用了 Olden[12] 和 CPU2006[13] 性能评测基准程序集中的三个带帮助线程的存储密集型程序:mst_ht、em3d_ht 和 429.mcf_ht. 所有程序均用 GCC 交叉编译为 MIPSII 可执行文件,GCC 优化选项为实现最佳优化的 -O3.

4.2　仿真实验结果分析

4.2.1　节能效果分析

通过仿真实验比较了不同措施下的总线节能效果,并给出了不同工艺下 MCFVC 的节能效果随 λ 变化的趋势. 各措施分别为:单独采用多编码技术(MC),单独采用 FEVC 存放 1 个频繁值(F1),单独采用 FEVC 存放 4 个频繁值(F4),联合使用 MC 和 F1(MCF1),联合使用 MC 和 F4(MCF4),FV-MSB 方法(MSB),本文方法 MCFVC 其中 FEVC 和 FHBC 中均存放 4 个值(MCHF4). 70nm 下,当 λ=5 时,对比了 3 个测试程序和平均情况下,各种措施的节能比例,结果如图 7 所示.

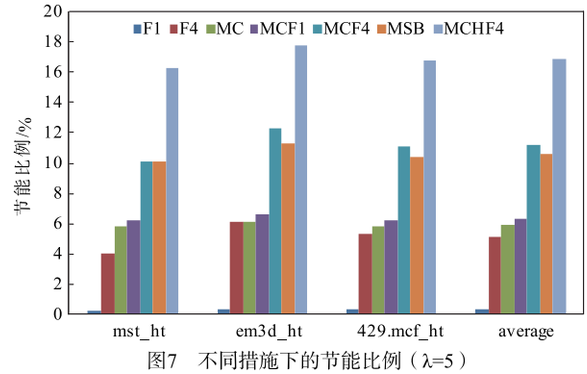

图7　不同措施下的节能比例(λ=5)

由图 7 可知,当采用 F1 时,基本没有节能效果,这是因为引入的 FEVC 自身能耗与其带来的能耗收益基本相抵. 但当采用 MCF1 时,可使平均节能比例达到 6.35%. 采用 F4 和采用 MC 时的效果相当,MC 效果稍好,平均可实现节能 5.92%. 以上措施的节能效果都不显著,而当采用 MCF4 时,节能比例得到提升,最高可达 12.30%(em3d_ht),平均节能比例为 11.16%. 这是因为当采用 MCF4 时 FEVC 中存放 4 个值,较存放 1 个值时,增加了频繁值的覆盖率,降低了位线上的总变换距离,使节能效果更显著. 而当采用 MCHF4 时进一步提升了节能比例,最高可达 17.76%(em3d_ht),测试程序的平均节能比例达 16.91%. MCFVC 方法较 FV-MSB 方法可使平均节能比例提高 6.28%. 本方法节能效果显著的原因是,多编码结构自动选择编码方式对频繁高位值的低位部分和非频繁值的处理,进一步减少了总线上的总变换距离. 而如果 FVC 中存放更多的值则需要增加额外硬件单元,使其自身能耗增加,这将抵消掉部分节能收益,增大片上面积负载,故 FVC 中存放的值不宜过多. 具体节能数据如表 2 所示.

表 2 $\lambda=5$ 时不同措施下的节能比例(%)

	F1	F4	MC	MCF1	MCF4	MSB	MCHF4
1	0.24	4.01	5.86	6.19	10.09	10.15	16.24
2	0.34	6.16	6.14	6.65	12.30	11.32	17.76
3	0.35	5.31	5.78	6.22	11.09	10.41	16.73
4	0.31	5.16	5.92	6.35	11.16	10.63	16.91

(最左列 1:mst_ht,2:em3d_ht,3:429.mcf_ht,4:average)

MCHF4 措施下,λ 取不同值时的节能比例见表 3,这些数据中包含了引入指示线和 FVC 带来的负载能耗. average 表示平均节能比例,当 $\lambda=0$,即不考虑耦合电容时,最高可节能 13.82%(em3d_ht),平均节能比例为 12.42%,此时的节能比例并不高,因为此时引入措施带来的负载能耗占比较大,抵消了由其带来的能耗收益. 当 $\lambda>0$ 时,随着耦合电容的出现并逐渐增加,措施带来的能耗收益不断增加,$\lambda=5$ 时平均节能比例为 16.91%,此后随 λ 增大节能比例略有上升,增加幅度逐渐减小并趋于稳定,当 $\lambda=10$ 时,最大节能比例可达 18.6%,平均节能比例为17.76%. MCFVC方法实现的总线节能比例随 λ 变化的趋势如图 8 所示,平均节能比例随 λ(λ 取 0 到 10)增加而逐渐增加并趋于稳定. 这也表明本文的方法适合工艺尺寸缩小,λ 不断增大的趋势,对未来工艺下的嵌入式片上多核数据总线节能依然有效.

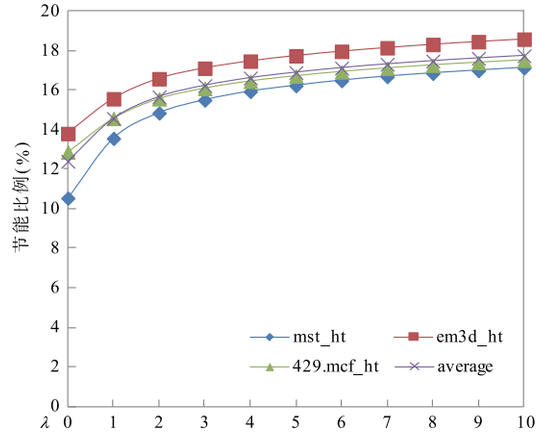

图8 不同λ下的节能效果趋势

表 3 MCHF4 措施下 λ 取不同值时的节能比例(%)

λ	0	1	2	3	4	5	6	7	8	9	10
mst_ht	10.55	13.57	14.86	15.51	15.95	16.24	16.50	16.70	16.87	17.01	17.14
em3d_ht	13.82	15.58	16.59	17.13	17.49	17.76	17.98	18.16	18.33	18.47	18.60
429.mcf_ht	12.88	14.58	15.58	16.10	16.46	16.73	16.94	17.12	17.28	17.41	17.53
average	12.42	14.58	15.68	16.25	16.63	16.91	17.14	17.33	17.49	17.63	17.76

4.2.2 负载分析

采用 MCFVC 方法实现总线有效节能的同时也引入了负载. 在能耗方面,上述节能比例已包含指示线和 MCFEVC 模块自身能耗;从时间上看,发送数据前搜索 FVC 时,采用了流水线的方式,对发送的数据块采用边搜索边发送,即当在 FVC 中搜索第二个数据值时,第一个数据值已在总线上传输,如果对整个数据块搜索完再一起发送,将带来很高的延迟,采用这种流水线的发送方式,对一个待发送的数据块,引入的延迟只发生在发送端第一个值的搜索和接收端最后一个值的搜索上,因此,只是引入了两个时钟周期的延迟,对于尺寸大的数据块带来的延迟可以忽略;在芯片面积方面,引入了 MC 编解码模块(主要由 2~4 译码器,异或门和反相器构成),3 条指示线和 5 个包含 8 条记录的 FVC 内容地址存储器,这些硬件的引入对于片上门电路影响较小,在此没有量化. 总之,MCFVC 方法在引入有限负载的情况下,实现了嵌入式片上多核数据总线动态能耗的有效降低.

5 结论

本文考察了深亚微米工艺下,考虑耦合电容影响的数据总线动态能耗节能问题. 设计了附带总线节能措施的嵌入式片上多核结构,提出了 MCFVC 数据总线节能方法,通过频繁值缓存 FVC 及对非频繁值和频繁值的低位部分进行的基于位变换数感知的多编码处理,大幅降低了总变换距离,有效提高了总线节能比例. 仿真实验结果显示,70nm 工艺下,MCFVC 方法可大幅降低数据总线动态能耗,与其他措施相比本方法均可取得更好的节能效果. 此外,节能比例随 λ 的增大而增加的变化趋势表明,MCFVC 方法在未来工艺尺寸进一步缩小时,仍可有效减少嵌入式片上多核数据总线的动态能耗.

参考文献

[1] Yang X,et al. A high-performance on-chip bus (MSBUS) design and verification[J]. IEEE Trans on VLSI Systems, 2015,23(7):1350-1354.

[2] 钟琥,齐勇,等. 基于 DVS 的多核实时系统节能调度[J]. 电子学报,2006,(S1):2481-2484.
Zhong Xiao,Qi Yong,et al. Tasks scheduling with dynamic voltage scalingon multi-core real-time systems [J]. Acta

Electronica Sinica,2006,(S1):2481 – 2484. (in Chinese)
[3] Wang Shinan, et al. Application configuration selection for energy-efficient execution on multicore systems[J]. Journal of Parallel and Distributed Computing, 2016, 87(1): 43 – 54.
[4] 林焕,马志峰,等. 基于格雷码-相移的双目三维测量方法研究[J]. 电子学报,2013,41(1):24 – 28.
Lin Huan, Ma Zhifeng, et al. 3D measurement technology based on binocular vision using a combination of gray code and phase-shift structured light[J]. Acta Electronica Sinica,2013,41(1):24 – 28. (in Chinese)
[5] Stan M R, Burleson W P. Bus-invert coding for low-power I/O[J]. Trans on VLSI Systems,1995,3(1):49 – 58.
[6] 刘星成,叶远生. 系统 RA 码的基于 WBF 策略的改进 BP 译码算法[J]. 电子学报,2010,38(7):1541 – 1546.
Liu Xingcheng, Ye Yuansheng. An improved BP decoding algorithm based on WBF scheme for systematic RA codes [J]. Acta Electronica Sinica,2010,38(7):1541-1546. (in Chinese)
[7] Yang, et al. Frequent value encoding for low power data buses[J]. ACM Trans on DAE Systems,2004,9(3):354 – 384.
[8] Mohammad, Khader, et al. On-chip power minimization using serialization-widening with frequent value encoding [J]. VLSI Design,2014,v2014(1):1 – 14.
[9] Halak, Basel. Partial coding algorithm for area and energy efficient crosstalk avoidance codes implementation[J]. IET Computers and Digital Techniques,2014,8(2):97-107.
[10] Suresh, Dinesh C, et al. FV-MSB: A scheme for reducing transition activity on data buses[A]. High Performance Computing[C]. Berlin, Heidelberg: Springer-Verlag, 2003. 44 – 54.
[11] 张铭泉,古志民,等. 基于频繁交换值的多核交叉开关节能方法研究[J]. 北京理工大学学报,2015,35(11):1146-1151.
Zhang Mingquan, Gu Zhimin, et al. The research of crossbar energy saving of CMP based on frequent exchanged value[J], Journal of Beijing Institute of Technology, 2015,35(11):1146 – 1151. (in Chinese)
[12] Rogers A, et al. Supporting dynamic data structures on distributed memory machines[J]. ACM Transactions on Programming Languages and Systems, 1995, 17(2): 233 – 263.
[13] Henning, John L. SPEC CPU2006 benchmark descriptions [J]. SIGARCH Comput Archit News, 2006, 34(4): 1 – 17.

作者简介

张铭泉　男,1980 年生于山东莘县. 北京理工大学计算机学院博士生. 研究方向为嵌入式多核节能设计.
E-mail:zmqcherish@163.com

古志民　男,1964 年生于山西运城. 北京理工大学教授,博士生导师. 研究方向为嵌入式多核优化设计.