

基于关系逻辑的大数据处理研究

何 群

(南京工业大学 计算机科学与技术学院, 江苏 南京 210009)

摘 要: 提高系统运算速度是解决大数据问题的有效途径, 而从原始数值层进行高性能运算是解决这个问题最直接方法. 提出一种基于速度、功能、资源和连线的可编程部件优化设计方法, 将同类格式的数据运算归并为一类关系, 以函数的约束列出输入对应输出的运算关系逻辑表, 表的每列示一级位域; 表的行示由函数确定的运算数据集. 根据逻辑关系, 采用位域、邻域和全域运算的三个逻辑层次获得输出数值的逻辑表达式; 对数量或位长不同的数据变化情况, 具有差时可控功能. 给出了累加和判定两种关系的设计方法并且将其应用于大数据地址变换和挖掘分析.

关键词: 大数据; 关系逻辑; 可编程部件; 一体化设计; 粗糙集; 决策表

中图分类号: TP302

文献标识码: A

文章编号: 1000—7180(2017)08—0001—05

Research on Big Data Processing Based on Relational Logic

HE Qun

(College of Computer Science and Technology, NanJing University of Technology, Nanjing 210009, China)

Abstract: Improving the speed of system operation is an effective way to solve the problem of Big data, and the most direct method is to perform high performance computing from the original value layer. The method of optimization design is proposed based on the speed, function, resources and the connection of the programmable components, the operations in the same format are merged into a class of relation, the relation table on the input corresponding to the output data is listed as a function of the constraint, each column shows the a bit field and the data in the rows is determined by the function. According to the relationship, the logical expression of the output value is obtained using a domain, neighbourhood, global operations, differential time control can achieve for the number or bit length of different data changes. The design methods of two logic relations with accumulation and decision are given and it is applied to address transform and mining analysis on Big data.

Key words: big data; relational logic; PLD; integrated design; rough set; decision table

1 引言

大数据问题在于如何在当前大数据集中获得的有价值的数据且在所希望的时间内完成对该数据分析的任务, 其核心是加快系统的运作速度, 集中体现在运算部件的运行速度. 当前计算机的运算器是一种通用运算部件, 任一种复杂的算式是通过众多基本运算实现的, 用时量极大. 尽管在大数据应用层面已经提出了各种方法, 但最底层的运算速度始终是这些应用瓶颈, 随着可编程部件和和技术^[1-2]的发

展, 定制专用运算部件可以有效地解决速度问题. 将若干频繁处理的函数过程定制为专用部件从最底实现函数功能, 即实现一种由输入变量的原始代码通过部件直接生成输出函数值代码过程. 要实现这个目的, 首先要使设计部件的方法系统化, 即相同格式的数据运算都用同类逻辑关系; 运算逻辑采用统一层次设计; 根据参与运算的数据的数量及位长即时获得输出数据. 为此, 提出一种基于逻辑关系的一体化设计方法. 将同一类数据格式的函数归并为一类基本逻辑关系, 该关系是以输入变量值所产生的输

出函数值建立逻辑关系;根据部件的约束条件及优化连线、I/O 资源和速度的要求,依次采用本域、邻域、全域三个层次的逻辑运算,获得输出函数对应于入变量的最简逻辑表达式.分析各层次运算的时长,给出参与运算的数据的数量和位长的准确延时的计算方法,实现在同一关系下运算范围的可扩、缩性.

针对大数据存储、挖掘中常用的地址变换、分类判断运算,给出这两类运算的一体化设计方法.

2 理论与技术

2.1 大数据问题

大数据是指为决策问题提供服务的大数据集、大数据技术总称^[3].大数据集指该集中数据量大、类型多样;大数据技术指大数据资源获取、存储管理、挖掘分析等技术.从数据中及时挖掘出有意义的规律是当前大数据获得广泛应用的重要原因之一^[4].

由于一般运算部件的通用性,在频繁处理大数据方面效率不高;许多大数据挖掘算法可以表示为矩阵运算或表处理,程序计算量极大.而在这方面采用专用加速部件优于通用的程序运算.

由上分析可知,从高层提高大数据处理方面空间有限;而从低层采用专用高性能计算部件可有效地解决大数据问题.

2.2 可编程技术

现场可编程门阵列(FPGA)^[1]和在系统可编程(iSP)^[2]是高性能的硬件产品,具有设计灵活、集成度高、可反复编程的特点.可将在实际环境达到充分检验的设计部件迅速转换为 ASIC 电路.

可编程设计的应采用逐层分解逻辑功能、进行最优化设计.分层是按实现功能的过程将其划分为若干个实现特定功能的层次;最优化则是充分利用器件的逻辑资源、连接资源和 I/O 资源.达到器件资源利用率高、运算速度快和连线简的目的.

设计步骤中关键步是确定逻辑算法,算法的合理与否直接影响系统结构的合理性.首先需要获得数据输入与输出变量的逻辑关系,通过该关系分层优化出其逻辑结构及算法.

2.3 逻辑关系

任何格式的二进制变量运算都可以用一两维逻辑关系表示.变量是一组有权重的二进制代码集合,当若干输入数据变量按函数要求进行运算时,输入数据与输出数据就产生了逻辑关系,该关系就是函数对输入数据的约束.

定义 1 设 $R=(U, \{In\}, Om, F)$. 其中, U 是

表的行号; Om 为输出数据的二进制位域, $\{Im\}$ 为输入数据的二进制数值变量的运算集; F 是函数,在 F 约束下生成的输入数据集相对于输出数据的逻辑关系集合;称 R 为逻辑关系表.

R 说明了输入生成输出的逻辑关系,每列位域中各变量的权重相同,在 F 约束下,行与行之间的运算共同映射出输出的逻辑表达.

如设输入 X 、 Y 、 Z 分别是 2 位整型变量, W 是 4 位整型变量,函数 $S(X,Y,Z,W) = X * Y * Z + W$, 其逻辑关系为表 1.

表 1 逻辑关系

U	S_1	S_4	S_3	S_2	S_1
1				$x_2y_1z_1$	$x_1y_1z_1$
2			$x_2y_2z_1$	$x_1y_2z_1$	
3			$x_2y_1z_2$	$x_1y_1z_2$	
4		$x_2y_2z_2$	$x_1y_2z_2$		
5		w_4	w_3	w_2	w_1

表中由 F 蕴含设定每列表示一级位域,列中每项的权重相同,列与列之间是进位关系、行与行之间是累加关系.

表 1 是对于整型格式的累加型运算基本关系结构,任何整型格式的四则运算(减、除采用补码)都可以表示为表 1 的形式.当变量的数量和长度变化时,其关系可按该结构进行扩展;对于其它格式和函数约束都可以按定义 1 进行推广.

2.4 关系设计

逻辑关系是可编程设计中确定逻辑算法的依据,一体化设计是将这种关系进行优化设计.以典型的累加逻辑关系为例进行分析和设计.

从优化速度要求,由于每个位域中各位值权重相同,所以各位域并发求和可使各域同时按 2 的指数量级压缩成各位域的有权码(权码),速度最快;从优化连线要求将各位域的权码按权重扩展至邻域形成新位值关系,连线最短;当各位域压缩至 2 位时,从优化速度和连线要求,进行全位域运算,可同时实现各位域进位和累加运算获得各位域为 1 位的结果.即用位域求和、邻域调正和全域运算 3 个层次运作实现这种逻辑关系可达最优化设计.

从功能上优化,要分别给出 2、4、8…、64、…个变量和变量为 8、16、…、64、…的位长运算的选择逻辑及各种运算时延的计算方法,使运算范围及其时延可扩、缩.

2.4.1 位域求和

位域求和是将本位域各位按逻辑关系进行累加

获得该位域的权码,其推导公式如下:

(1) 设4位本位域的二进制变量为 $abcd$,得3位变量的权码 $E(E_2E_1E_0)$:

$$\begin{aligned} E_2 &= (ac)(bd); \\ E_1 &= (ac) \oplus (bd) + (a \oplus c)(b \oplus d); \\ E_0 &= (a \oplus b) \oplus (c \oplus d); \end{aligned}$$

(2) 设两个本位域的3位权码 $E(E_2E_1E_0)$, $F(F_2F_1F_0)$ 得8位变量的4位权码 $G(G_3G_2G_1G_0)$:

$$\begin{aligned} G_3 &= E_2F_2; \\ G_2 &= (E_1F_1) + (E_2 \oplus F_2); \\ G_1 &= (E_0F_0) + (E_1 \oplus F_1); \\ G_0 &= (E_0 \oplus F_0); \end{aligned}$$

根据(2)的方法,可递推出 $n = 2^{(i-1)}$ 位变量的 i 位权码. 设两个同位域的 $(i-1)$ 位权码 $I(I_{i-1}I_{i-2}\cdots I_1I_0)$ 、 $J(J_iJ_{i-1}\cdots J_1J_0)$,得

$$\begin{aligned} H(H_{i-1}H_{i-2}\cdots H_1H_0); \\ H_i &= I_{i-1}J_{i-1}; H_{i-1} = I_{i-2}J_{i-2} + (I_{i-1} \oplus J_{i-1}); \\ &\cdots; \\ H_1 &= I_0J_0 + (I_1 \oplus J_1); \\ H_0 &= (I_0 \oplus J_0) \end{aligned} \quad (1)$$

2.4.2 邻域调正

邻域调正是将各位域的变量权码按权重向相邻的位域调正,形成各位域新的位值.

设本次各位域都分别生成 $(k+1)$ 个权位码, jH_k 表示 j 域的权重为 k 的码.按权重调正有

$$(j+i)H_i \leftarrow jH_i \leftarrow (j-i)H_i \quad (i=1,2,\cdots,k).$$

邻域调正是为下一次的本位域求和作准备,逐次进行本位域求和及邻域调正,使各位域的代码的位数逐级下降至两位.

2.4.3 全域运算

全域运算是将两行的代码进行全域运算,得输出的位域值.实现的方法是并行进位^[5].将 M 个位域以4为单位分成 $M/4$ 个组域,组域内、组与组之间的进位并行产生.

设组域 J 为 $\{S_{i+3}, S_{i+2}, S_{i+1}, S_i\}$; S_k 域中的两个值为 (a_k, b_k) ; C_i 为后域向本组域的进位,计算后本位域的值为 S_k .有一般式

$$\begin{aligned} S_k &= (a_k \oplus b_k) \oplus C_k = p_k \oplus C_k; \\ C_{k+1} &= a_k b_k + C_k(a_k \oplus b_k) = g_k + C_k p_k. \end{aligned} \quad (2)$$

式中, $k=i, i+1, i+2, i+3$.

组与组之间并行的条件是由各组域发出的2个状态信号所决定的,以组域 J 为例,2个状态信号分别为:

$$\begin{aligned} G_j &= g_{i+3} + g_{i+2} p_{i+3} + g_{i+1} p_{i+2} p_{i+3} + \\ &g_i p_{i+1} p_{i+2} p_{i+3} \\ P_j &= p_i p_{i+1} p_{i+2} p_{i+3} \end{aligned} \quad (3)$$

式中, G_j 表示本组域内是否有向前进位; P_j 表示本组域内是否可接容后级进位.组控并行进位的方法是根据各后组域的2个状态信号综合判定产生的.有一般式:

$$\begin{aligned} C_k &= (G_k) + (P_k)C_{k-4} \\ (k &= 4, 8, 12, \cdots, 60, 64) \end{aligned} \quad (4)$$

式中, $(G_k) = G_k G_{k-1} G_{k-2} G_{k-3}$;

$$(P_k) = P_k P_{k-1} P_{k-2} P_{k-3}.$$

2.4.4 功能分析

由于输出的数据格式相同且被划分成域的逐层运算,可将表1的形式扩至 n 行 $\times m$ 列进行分析.

设计的输入数据、逻辑运算都从第一行开始,递序向 n 扩进,对于 $n=2,4,8,\cdots$,的运算都要给出其选择逻辑.

从运算延时上分析,域运算包括本域求和、邻域调正,其中邻域调正是连线,无延时;本域求和是并发的,延时最长为(1)式 H_{i-1} 中的最长的逻辑项,为一级两变量异或和一级或运算,总延时为 $3t_y$.所以域位数量为 n 的求和至 i 位的延时为 $3it_y$ ($n=2^i$).

全域运算包括进位生成、各位域求和,其中进位生成是以4为基本单位分组生成,4位域为一小组;4小组(16位)为一中组;4中组(64位)为一大组; $\cdots$ 组内、组间并发是通过 p 和 g 、 P 和 G 信号传递.由式(2)、(3),选 p (或 P)和 g (或 G)中项延时最长项,都为一级两变量异或其延时为 $2t_y$;由式(4),最高组生成组控并行进位信号延时为 $1.5t_y$,而由式(2),最高域求和延时为 $2.5t_y$.所以域长为 m 位的全域运算延时有 $2jt_y + 1.5t_y + 2.5t_y$ ($m=4^j$).

根据上述算式可计算出逻辑关系为 n 行、 m 列的总延时,按时获得计算结果.反之,当 m 大于一个芯片的容量,可通过 G 和 P 信号联级扩展;而当 n 大于一个芯片的容量 k 时,可分 n/k 批分次处理.

3 大数据应用

3.1 指针变换实例

大数据存储技术是分布式文件系统,其对地址的大量计算是整型格式的累加型运算. GFS^[6]的主控机在获得客户端的请求[文件名、块索引]后,主控机首先通过该信息在文件名空间得一个文件分类指针: I (8位,一级索引)、 j (16位,二索)、 k (32位,三

索)、 p (32 位, 偏移). 然后将其变换为一个 4 变量控制范围信息. 算式为

$$F(r, s, b, l) = k * j * I + p$$

式中, $r(F_{56 \sim 53})$ 示路由区间; $s(F_{52 \sim 41})$ 示服务器区间; $b(F_{25 \sim 40})$ 示存储体区间; $l(F_{24 \sim 1})$ 示位移量.

根据算式可得类似表 1 的 $m = 64, n = 97$ 的整型累加逻辑关系 $k(32) * j(16) * I(8) + p(32)$.

设芯片可实现的容量为 $n = 64, m = 32$. 所以要分 4 批实现这个函数关系, 前 3 批完成 1 ~ 96 行运算, 每批输出 1 个中间结果. 根据 3.1, 对 j 位域的运算有

(1) j 位域经对至多 32 位长的求和得至多 5 位权码 $jH_i (i = 0, 1, \dots, 4)$;

(2) 经邻域调正 5 域的位值为

$\{jH_0, (j-1)H_1, (j-2)H_2, (j-3)H_3, (j-4)H_4\}$;

(3) j 位域经对 5 位长的求和得 3 位权码 $jH_i (i = 0, 1, 2)$;

(4) 经邻域调正 3 域的位值为 $\{jH_0, (j-1)H_1, (j-2)H_2\}$;

(5) j 位域经对 3 位长的求和得 2 位权码 $jH_i (i = 0, 1)$;

(6) 经邻域调正 2 域的位值为 $\{jH_0, (j-1)H_1\}$;

(7) 经全域运算由各位域这两位值获得该轮计算的一中间值.

再将 3 个中间值和 p 计算得 56 位的函数值 $F(r, s, b, l)$. 前 3 次分别按 $n = 64, m = 32$ 的用时获得 3 个中间值; 而第 4 次则是按 $n = 64, m = 4$ 的用时获得函数值.

3.2 挖掘分析实例

大数据挖掘则是通过分析不同来源的数据形成决策依据. 其中对决策表进行属性约简和确定分裂属性的都是运算量大的环节. 粗糙集^[7-8]在处理这两类问题的基本思路是通过计算出各条件属性对决策属性的分辨能力的强弱进而确定条件属性的重要性, 是一种判别逻辑.

判别型逻辑关系是累加型逻辑关系的推广, 将关系表划分为若干块, 每块并发判别得该块判别值, 再以块域为单位进行累加型的 3 个层次的运算.

(1) 分类原理

根据粗糙集的分辨关系^[7-8]原理, 对决策表中条件属性 C_i 重要程度的判定, 可以用 C_i 的每类属性值 v_j 将决策表划分为若干子行集, 每个行集所对应

的决策属性值 D, d_j 的比率可直接反应 C_i, v_j 对 D 的分辨能力, 该比率值越接近分辨能力越差, 反之亦然. 将 C_i 各子集分辨能力累加即为 C_i 分辨能力.

(2) 分辨算法

设 N 为条件属性 $C = a$ 的个数, 在这 N 行内, 决策属性 D 的属性值 $\{d_i\}$ 也有 N 个, 在 $\{d_i\}$ 中, 对各类数值统计可得一个比率值, 该比率值直接反应出 $C = a$ 对 $D = d_j$ 分辨能力, 由其确定比率权重 K , 表 2 的左表列出了 D 为 2 类值时 N 、比率与 K 三者之间的关系. 分辨算法定义为:

$$\begin{aligned} D(C_i) &= \sum D(C_i, v_j) \\ &= \sum (N_j * K_j) \\ (i &= 1, \dots, n) (j = 1, \dots, m) \end{aligned} \quad (5)$$

$$C_k = \text{Max}(D(C_i)) \quad (6)$$

式中, N_j 为 $C_i = v_j$ 的行数量值; K_j 是 v_j 所对应 D 中属性值的类别比率的权重; C_k 为最重条件属性.

求 K_j 是针对 D 的属性值数量而定. 对于 D 具有两类属性值的计算方法如下: 设 D 的属性值每个字长为 2 位, 共 4 组 $\{w_2 w_1, x_2 x_1, y_2 y_1, z_2 z_1\}$. 这 4 组数值比较式为

$$p_1 = (w_2 \odot x_2)(w_1 \odot x_1);$$

$$p_2 = (w_2 \odot y_2)(w_1 \odot y_1);$$

$$p_3 = (w_2 \odot z_2)(w_1 \odot z_1).$$

其 N 、比率、 K 值关系及算法如表 2.

表 2 K 值的算法

$N \backslash K$	1	2	4	算式
1			1:0	只有一类数据, 所以 $k = 4$
2	1:1		2:0	若 $p_1 = 0, 1$, 则 $K = 1, 4$
3		2:1	3:0	$q_1 = p_1 p_2 = 0, 1$, 则 $k = 2, 4$
4	2:2	3:1	4:0	$q_2 = q_1 p_3, r_1 = (p_1 \oplus p_2 \oplus p_3)$ $q_2 = 1, k = 4; r_1 = 1, 0, k = 1, 2$

表 2 是根据粗糙集的分辨关系^[7-8]原理及逻辑设计的可操作性给出的. 只给出决策属性 2 类且 4 行的情况, 根据要求可推广出多类、多行的比率权重表.

(3) 属性分类

文献[7]中表 1 中决策属性 d 的值是 2 类, 即 $\{00, 01\}$. 得其逻辑关系如表 3.

其中, 表中的二进制数为有效信息, 其它信息是为方便说明而设; $U = 1, 7$ 示条件属性值, $U = 2, 8$ 示该块中决策属性值的个数; 而 d_i 是决策属性值, i 示表 1^[7] 的行号. 其判别运算过程如下:

对 $a, 1, n = 3, d_2$ 与 d_3, d_5 的比率为 2:1,由表 2 得 $k = 2$,由式(5) 得 $a = 1$ 的本域值

$$D(a, 1) = n * k = 3 * 2 = 6 = (110)_2;$$

表 3 决策逻辑关系

U	$a(x_8 \sim x_1)$	$b(y_8 \sim y_1)$	$c(z_8 \sim z_1)$
1	$a, 1$	$b, 0$	$c, 0$
2	$n = 11$	$n = 11$	$n = 100$
3	$d_2 = 00$	$d_4 = 00$	$d_1 = 01$
4	$d_3 = 01$	$d_5 = 00$	$d_2 = 00$
5	$d_5 = 00$	$d_6 = 01$	$d_3 = 01$
6			$d_4 = 00$
7	$a, 2$	$b, 2$	$c, 1$
8	$n = 10$	$n = 11$	$n = 10$
9	$d_1 = 01$	$d_1 = 01$	$d_5 = 00$
10	$d_6 = 01$	$d_2 = 00$	$d_6 = 01$
11	$a, 0; n = 01$	$d_3 = 01$	

类似地,

$$D(a, 2) = n * k = 2 * 4 = 8 = (1\ 000)_2;$$

$$D(a, 0) = n * k = 1 * 4 = (100)_2.$$

进行本域和、邻域调正和全域运算,则

$$\begin{aligned} D(a) &= \sum D(a, v_j) = D(a, 1) + D(a, 2) + D(a, 0) \\ &= 18 = (x_5x_4x_3x_2x_1) = (10\ 010)_2. \end{aligned}$$

同理,

$$\begin{aligned} D(b) &= \sum D(b, v_j) = D(b, 0) + D(b, 2) \\ &= 3 * 2 + 3 * 2 = 12 = (y_5y_4y_3y_2y_1) \\ &= (01\ 100)_2; \end{aligned}$$

$$\begin{aligned} D(c) &= \sum D(c, v_j) = D(b, 0) + D(b, 2) \\ &= 4 * 1 + 2 * 1 = 6 = (z_5z_4z_3z_2z_1) \\ &= (00\ 110)_2. \end{aligned}$$

这就按式(5) 得到各属性的分辩值. 式(6) 是将其进行比较,得各属性的分辩强度的排序. 表 3 中 $a(x_8 \sim x_1), b(y_8 \sim y_1), c(z_8 \sim z_1)$ 是 24 位($3 * 8$) 位长一并输出,经对 3 个 8 位值的比较有 $a > b > c$. 这就是输出的分辩结论. 如要建决策树,则首选 a 为分裂属性;如要进行属性约简,则约简 c ,因为对于 6 行的决策表,按式(5) 算法, $D(C_i) = 6$ 是最低的分

辩率,即对决策 D 没有任何分辩能力. 分辩能力由强至弱为 $a > b > c$ 且约简 c 的结论与文献[7] 的结论相同.

算法以 2 的指数递增形式确定分辩率,具有高分辩率且易于逻辑实现.

4 结束语

认为解决大数据处理的速度问题最有效的方法是采用可编程的专用运算部件,因为其具有底层直接处理原码的功能,而如何设计该部件使其适于这种应用就是一个关键问题. 基于由函数所约束的输入与输出数据的二元逻辑关系提出一种运算部件的一体化设计方法,方法从逻辑运算速度、连线、器件资源利用率和运算范围可扩展性几个方面对可编程部件进行综合优化. 通过对两个实际的大数据问题的处理证明了该方法的有效性.

参考文献:

[1] 朱欣欣,李树国. 基于 FPGA 的高性能 3DES 算法实现[J]. 微电子学与计算机,2015,32(9):54-59.

[2] 张志友,胡松. 在系统编程技术在煤矿井下通信中的应用[J]. 信息化研究, 2014,39(10):73-75.

[3] 朱扬勇,熊赞. 大数据是数据、技术,还是应用[J]. 大数据,2015,1(1):71-81.

[4] 杨刚,杨凯. 大数据关键处理技术综述[J]. 计算机与数字工程,2016,44(4):694-699.

[5] 袁浩,唐建,方毅. 超前进位加法器的优化设计[J]. 通信技术,2014,47(3):339-342.

[6] 杨斐,艾晓燕. 大数据精准挖掘处理架构及预测模型研究[J]. 电子设计工程 2016,24(12):29-32.

[7] 唐定勇,景运革. 一种决策表属性值细化的正域约简算法[J]. 微电子学与计算机,2015,32(3):23-27.

[8] 黄治国,刘罡. 代数与信息论观点的分辩矩阵属性约简研究[J]. 微电子学与计算机,2014,31(7):88-92.

作者简介:

何 群 男,(1955-),高级工程师,研究生导师. 研究方向为粗糙集、粒计算和可编程部件.
E-mail:hequn20161@126. com.