通过程序变换对数字系统进行优化*

胡振江 孙永强

(上海交通大学计算机系)

OPTIMIZING A DIGITAL SYSTEM BY PROGRAM TRANSFORMATION

Hu Zhenjiang and Sun Yongqiang

(Department of Computer Science, Shanghai Jiaotong University)

ABSTRACT

In paper [10], we have given a detailed discussion on how to describe, synthesize and simulate a digital system using functional hardware description language. In this paper a further study of the algebraic properties of the language will be presented and the digital system will be optimized by program transformation. In this way, we can get a correct and optimized logic structure which has no redundancy and reuses each common component to the great extent. Some rules and algorithms of transformation and some examples are given to illustrate this new method.

摘 要

在文章[10]中，我们详细地讨论了如何用函数式硬件描述语言对数字电路进行描述、综合和模拟验证的方法。本文将在此基础上，进一步研究函数式硬件描述语言的代数性质并通过程序变换对所描述的电路进行综合和优化，从而设计出正确的优化逻辑结构，在这种逻辑结构中去除了冗余，最大限度地重复使用各个子部件。文章中给出了变换规则和变换的算法及若干示例。

* 1989年6月17日收到。

§1. 引 言

随着 VLSI 的迅猛发展，自动的逻辑综合和优化系统显得相当重要，它不但减小了 VLSI 设计的复杂性，而且缩短了设计的轮回时间．但是，功能行为级的正确性验证是 VLSI 设计系统的瓶颈问题．从经济的角度来看，VLSI 电路的设计在制造之前，必须得到充分的验证，目前常用的方法是通过各种模拟和测试．现在人们试图采用形式化的设计和验证方法[2][3][4]，来保证设计出来的电路的正确性。

人们把很多精力集中在开发形式化的硬件描述语言，如 DDL，VHDL 上以及形式化的硬件验证方法，但是对形式化的逻辑综合和优化方法却讨论得不多．人们把注意力主要集中在如何从布尔方程出发进行逻辑化简和优化[5]，但这种方法比较适合小规模的电路，且综合和优化的层次较低——在逻辑级上，另外象时间逻辑等物理特性在优化过程中很难得到体现．为了解决上述问题，人们利用元件的电路模型进行逻辑综合和优化[6][7][8]，这种系统用已进行过总体优化并考虑过电路物理特性的元件来替换初始电路中部分元件，但是元件的电路模型很难表示且实现起来相当困难，同时很难进行公用部件的提取和重用。

本文所介绍的数字电路的优化方法，是基于函数式硬件描述语言 FP-1 的程序代数性质，通过程序变换的方法，对初始的功能行为级的描述进行变换，得到一个优化的结构表达，它有效地解决了公用部件的提取和重用问题．并且对物理特性和电路子部件的优化策略都进行了讨论。

§2. FP-CAD 系统的概述

在文章[10]中，我们详细地讨论了自动逻辑综合系统 FP-CAD 系统．其系统结构图如图 1 所示．FP-CAD 系统采用类似于巴科斯提出的 FP 语言[1]，并参照 μFP[2] 和 γFP[3] 而形成的一种新型函数式硬件描述语言 FP-1[10]，FP-1 语言适合于描述同步时序电路和组合电路．图 1 中用一个半加器的自动综合和优化过程来说明整个系统的体系结构．文章[10]中详细地讨论了 FP-1 语言的编译、逻辑综合及模拟的算法和利用函数式硬件描述语言的特点。

用函数式硬件描述语言 FP-1 描述数字电路具有以下四个特点：

(i) FP-1 把对数字系统中抽象的行为描述和具体的逻辑结构的描述统一在一个语言中，FP-1 中的复合算子 · 和构造算子[ ]及延时算子 μ 都与一定的结构相对应，例如复合算子对应一种串联顺序结构，而构造算子则对应一个并发结构。

(ii) FP-1 语言没有副作用，具有良好的代数性质，便于进行电路的功能行为级验证，同时可通过程序变换进行逻辑综合和优化。

(iii) FP-1 语言具有很好的层次表达能力，便于描述复杂的电路系统，另外 FP-1 的电路描述具有类属性，比如，任意 $n-2^n$ 译码器只需一个递归方程描述即可。

(iv) FP-1 语言语义简单明确，便于实现。

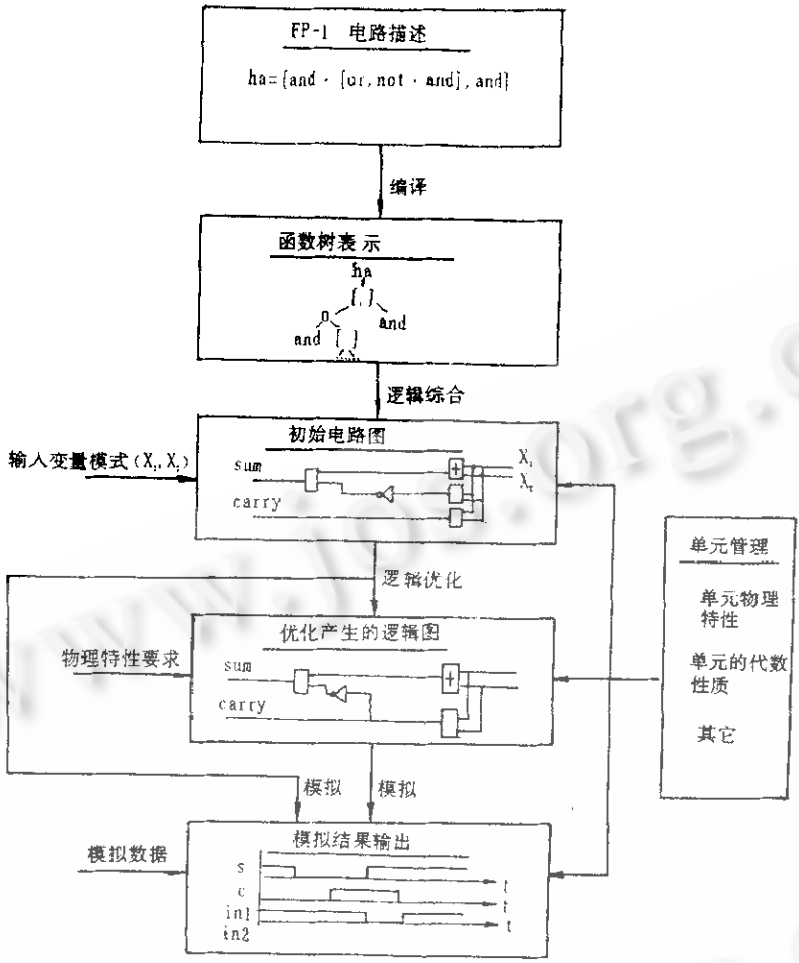

图1 FP-CAD 系统结构图

§3. 利用程序变换进行逻辑优化的策略

一个数字系统的优劣可有很多特性去衡量，诸如使用基本逻辑门的多少，整个电路延迟大小，输入输出的负载要求等。通过程序变换对数字系统进行优化就是对初始的数字电路描述，利用程序的代数性质及若干规则进行等价变换，从而得出符合要求的且优化了的逻辑电路。

假设整个综合出的电路仅由"与"门，"或"门和"非"门及 D 触发器等最基本的部件组成。优化的目的就是在满足用户的要求下使用尽量少的逻辑门。

1. 静态计值:

由于用 FP-1 来描述数字电路时比较偏重于行为和功能的表示，因而在描写时避免不了有冗余的或可简化的描述，这主要表现在 FP-1 语言中一些函数的静态计值中。

例如，简单逻辑门的化简中，我们有下述静态计值规则:

(R1.1) not · not⇒id;

(R1.2) /and · α not⇒not · /or:

/and · [not, not, …, not]⇒not;

/or · α not⇒not · /and:

/or · [not, not, …not]⇒not;

(R1.3) /and · 〔f_1, f_2, …, f_n〕而且存在$1\leqslant i\leqslant n$, $f_i=\bar{0}\Rightarrow\bar{0}$;

/or · 〔f_1, f_2, …, f_n〕而且存在$1\leqslant i\leqslant n$, $f_i=\bar{1}\Rightarrow\bar{1}$;

(R1.1)和(R1.3)是"与"、"或"、"非"门性质的部分描述，(R1.2)实际上是德摩根定律的描述。

除了简单的逻辑函数运算，FP-1 中还有一类函数，如 1, 2, …, n, hd, tl, apndl, apndr, reverse. 这种函数要么是选择输入的一部分作为作用对象，要么对输入对象进行顺序调整或重新组合。这类函数我们称之为走线函数，其静态计值常常可减少很多冗余，例如:

(R2.1) i · 〔x_1, x_2, …, x_n〕而且 $1\leqslant i\leqslant n\Rightarrow x_i$,

i · apndl · [x, y]而且 i=1⇒(i−1) · y;

i · apndr · [apndl · [x, y], z]而且 i>1

⇒(i−1) · apndr · [y, z];

(R2.2) 当函数 f 为选择函数 i 或 last 时，则有:

f · trans · 〔x_1, x_2, …, x_n〕⇒〔f · x_1, f · x_2, …, f_0x_n〕;

f · distl · [x, h]⇒〔x, f · h〕;

f · distr · [h, x]⇒[f · h, x];

(R.2.3) 〔〔f, h_1〕, …, 〔f, h_n〕〕⇒distl · 〔f, 〔h_1, …, h_n〕〕;

〔〔h_1, f〕, …, 〔h_n, f〕〕⇒distr · 〔〔h_1, …, h_n〕…, f〕;

(R2.1), (R2.2)对走线函数进行了静态计值，减少了很多冗余，(R2.3)则使 f 能被多次复用。

2. 公用部件的提取和重用

所谓公用部件的提取和重用就是把逻辑结构中多次重复出现的部件，考虑它们是否能公用，如能公用，则把这个部件提取出来作为公用部件，从而便于重用。图 2 是一个简单的提取 f 部件的例子。

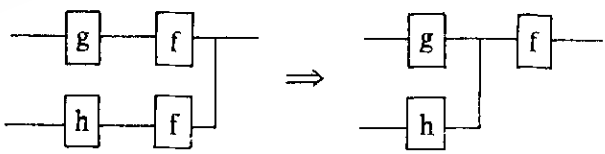

图 2 公用部件 f 的提取及重用

我们采用的公用部件的提取和重用的方法是首先把功能行为级的描述转化为结构化的

描述，然后再对结构化的描述进行转换以实现优化。

定义 1：一个函数称为本原函数当且仅当这个函数是 FP−1 中的基本原始函数或标准逻辑部件的函数。

定义 2：一个表达式称为组合式结构表达式当且仅当此表达式中所含的高级算子仅为复合算子·和构造算子[ ]，而且表达式中的函数为本原函数。

定义 3：一个表达式称为结构表达式，如果它有 μF 或 F 的形式，其中 F 为组合式结构表达式。

这里的延时算子 μ 的引进是为了描述同步时序电路的，其定义为：设$(x_1, x_2, \cdots)^+$表示一个流，如果 f: $<<x_1, ?>, <x_2 s_1>, \cdots>^+ = <<o_1, s_1>, <o_2, s_2>, \cdots>^+$，则 μf: $<x_1, x_2, \cdots>^+ = <o_1, o_2, \cdots>^+$，其中?是不关心状态，$S_i$ 起着保持状态的作用。

由以上定义可以看出结构表达式和一定的逻辑结构相对应，高级算子 · 、[ ]和 μ 对应的几何解释如图 3 所示。因此，欲把一个功能行为级的描述转化为结构化的描述就是把任何一个 FP−1 的表达式转化为结构表达式的过程。

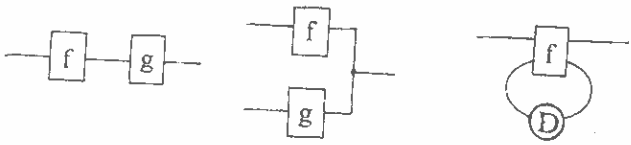

图 3 高级算子的几何解释

引理 1：任何一个 FP−1 表达式都能变换为仅含 · 、[ ]和 μ 高级算子，并由本原函数构成的表达式。

证明：由 FP−1 语言的代数性质，我们不难导出下述规则：

(SYN1) $\alpha f \cdot [h_1, h_2, \cdots, h_n] \Rightarrow [f \cdot h_1, f \cdot h_2, \cdots, f \cdot h_n]$;

(SYN2) $/f \cdot [h_1, h_2, \cdots, h_n] \Rightarrow \begin{cases} h_1; & \text{当} n=1 \\ f \cdot [h_1, /f \cdot [h_2, \cdots, h_n]]; & \text{当} n>1 \end{cases}$

(SYN3) $(p \to f; g): x \Rightarrow \begin{cases} f: x; & \text{如果} p: x \text{为真} \\ g: x; & \text{其它} \end{cases}$

(SYN4) f = p → H(f); G(f), 则:

$$f: x \Rightarrow \begin{cases} H(p) \to H(f); G(f)): x; & \text{当} p: x \text{为真} \\ G(p \to H(f); G(f)): x; & \text{当} p: x \text{为假} \end{cases}$$

(SYN5) 对于所有一阶定义函数:

DEF f · $P_{x1}, \cdots x_k = E_f, x_1, \cdots x_k$ 形成下述规则:

f · $P_{x1} \cdots, x_k \Rightarrow E_f, x_1, \cdots, x_k$

(SYN6) 对于所有高阶定义函数：

FDEF $F_g \cdot P_{x1, \cdots, xk} = E_{Fg, g, x1, \cdots, xk}$ 形成下述规则：

$F_g \cdot P_{x1, \cdots, xk} \Rightarrow E_{Fg, g, x1, \cdots, xk}$

利用(SYN1)～(SYN6)，我们不难证明引理 1. (证毕)

引理 2：任意一个仅含 · ，[ ]，μ 以及本原函数构成的表达式一定能等价地转化为结构表达式。

证明：由 μ 算子的定义，我们不难导出下述等价的变换规则：

(R3.1) $\mu f \cdot g \Rightarrow \mu(f \cdot [g \cdot 1, 2])$;

(R3.2) $f \cdot \mu g \Rightarrow \mu([f \cdot 1, 2] \cdot g)$;

(R3.3) $[\mu f, g] \Rightarrow \mu([1 \cdot f, g \cdot 1], 2 \cdot 1)$;

(R3.4) $\mu\mu f \Rightarrow \mu([1 \cdot 1 \cdot f, [2 \cdot 1 \cdot f, 2 \cdot 1]] \cdot [[1, 1 \cdot 2], 2 \cdot 2])$.

由以上(R3.1)～(R3.4)可覆盖任何情况下的 μ 算子的外提，从而得到形为 μF 的表达式，而 F 为组合式结构表达式。读者不难推导出 $\mu f \cdot \mu g$，$[f, \mu g]$，$[\mu f, \mu g]$ 的情况。

比如：$\mu f \cdot \mu g \overset{R3.1}{\Rightarrow} \mu(f \cdot [\mu g \cdot 1, 2]) \overset{R3.1}{\Rightarrow} \mu(f \cdot [\mu(g \cdot [1 \cdot 1, 2]), 2])$

$\overset{R3.3}{\Rightarrow} \mu(f \cdot \mu([1 \cdot g \cdot [1 \cdot 1, 2], 2 \cdot 1], 2 \cdot g \cdot [1 \cdot 1, 2]))$;

$\overset{R3.2}{\Rightarrow} \mu\mu([f \cdot 1, 2] \cdot R)$

其中 $R = [[1 \cdot g \cdot [1 \cdot 1, 2], 2 \cdot 1], 2 \cdot g \cdot [1 \cdot 1, 2]]$;

$\overset{R3.4}{\Rightarrow} \mu F(1 \cdot 1 \cdot F, [2 \cdot 1 \cdot F, 2 \cdot F]) \cdot [[1, 1 \cdot 2], 2 \cdot 2]$

其中 $F = [f \cdot 1, 2]$. R′ (证毕)

定理 1：任何一个 FP-1 表达式皆能等价地转化为结构表达式。

证明：由引理 1 和引理 2 很容易得证。 (证毕)

定义 4：定义组合式结构表达式 F 的作用序列 SEQ(F) 为：

$$SEQ(F) = \begin{cases} (F); \text{ 若} F = f_1 \cdot f_2 \cdot \cdots \cdot f_k, f_i \text{均为本原函数,} \\ \quad i = 1, 2, \cdots, k; \ k = 1, 2, \cdots. \text{ 或者} F \text{中不含部件子函数,} \\ SEQ([f_1, f_2, \cdots, f_n]); \text{ 若} F = h \cdot [f_1, f_2, \cdots, f_n]; \\ SEQ(g); \text{ 若} F = [f_1, f_2, \cdots, f_n] \cdot g, \text{ 且} g \text{中有部件子函数} \\ SEQ([f_1 \cdot g, f_2 \cdot g, \cdots, f_n \cdot g]); \\ \quad \text{若} F = [f_1, \cdots, f_n] \cdot g, \text{ 且} g \text{中无部件子函数。} \\ SEQ(f_1) \# SEQ(f_2) \# \cdots \# SEQ(f_n); \text{ 若} F = [f_1, f_2, \cdots, f_n], \end{cases}$$

其中 $(x_1, \cdots, x_n) \# (y_1, y_2, \cdots, y_m) = (x_1, \cdots, x_n, y_1, y_2, \cdots, y_m)$

且 $(\ ) \# (x_1, \cdots, x_n) = (x_1, \cdots, x_n) \# (\ ) = (x_1, \cdots, x_n) \# = \# (x_1, \cdots, x_n)$

从上面的定义可以看出 SEQ(F)是由若干不包含[ ]的最长子表达式构成；或有 $[g_1, g_2, \cdots, g_n]$，但 $g_1, g_2, \cdots, g_n$ 中无部件子函数。它们的作用对象与 F 一样，其排列顺序为这些子表达式在 F 中从左向右的顺序，例如：

SEQ([and, or · [not, not]]) = (and, not, not)

SEQ([1 · 2, id] · last) = ([1 · 2, id] · last)

SEQ(not · and) = (not · and)

SEQ([or, and] · [1, 2]) = (or · [1, 2], and · [1, 2])

引理 3: 任何一个组合式结构表达式 F 的作用序列 SEQ(F)=(h_1, h_2, …, h_n), 则 F 可写成 F' · [h_1, h_2, …, h_n)的形式。且 F'为组合式结构表达式.

证明: 对作用序列 h_1, h_2, …, h_n 进行编号 1, 2, …, n, 将 F 表达式中的 h_1, h_2, …, h_n 用对应的选择函数 1, 2, …, n 代替, 形成表达式 F', 显然有:

$$F = F' \cdot \langle h_1, h_2, \cdots, h_n \rangle. \qquad \text{(证毕)}$$

定义 5: 一个组合式结构表达式 F 是无拷贝表达式, 如果 F 的任意一个子表达式的作用序列中不存在两个元素 fp_1 和 fp_2, 存在一个逻辑部件函数 h, 使得 fp_1 和 fp_2 有形如 $fp_1 = g_1 \cdot h \cdot k_0$, $fp_2 = g_2 \cdot h \cdot k_0$ 的形式, g_1, g_2, k_0 为任意函数.

很明显, 无拷贝表达式最大限度地提取了公用部件, 并进行了重用.

定理 2: 任何一个组合式结构表达式 F 皆能等价地通过程序变换成为无拷贝表达式 F'.

证明: 如果 SEQ(F)=(F), 显然 F 为无拷贝表达式, $F'=F$ 否则 F 可写成 G · K 的形式。如果 K 的形式不是 $\langle k_1, k_2, \cdots, k_n \rangle$ 的形式, 则对 G, K, 从头递归, 否则进行如下工作:

设(h_1, h_2, …, h_n)为 K 的作用序列, $H = \langle h_1, h_2, \cdots h_n \rangle$

由引理 3 可知: $K = K' \cdot H$

Ⅰ)合并 H: 如果 $h_i = h_j$ 且 $i<j$, 则去除 h_j, 变 H 为 $H' = \langle h'_1, \cdots, h'_n \rangle$ 设 $h'_i = h'_{i1} \cdots h'_{i2} \cdot \cdots \cdot h'_{in}$, 在 H'中以 $h'_i \sim h'_j \sim \Longleftrightarrow h'_{in} = h'_{jn}$ 中~的等价关系分类, 并利用 $\langle f_1 \cdot g, \cdots, f_n \cdot g \rangle = \langle f_1, \cdots, f_n \rangle \cdot g$ 变 H'为 $H'' = \langle \langle h''_1, \cdots, h''_{i1} \rangle \cdot g_1, \cdots, \langle h''_j, \cdots, h''_{ij} \rangle \cdot g_j \rangle$ 为了保持等价性, 构造 rount 函数, rount 函数仅由走线函数复合或构造而成, 此时 H = rount · H''

Ⅱ)根据 rount 修改 K', 使得 K'中对 H 中元素进行选择的函数针对 H''进行重新选择, 形成 K'', 从而有 $K = K'' \cdot H''$

Ⅲ)如果 $H' \neq H$, 则对 G, K'', $\langle h''_1, \cdots, h''_{i1} \rangle$, …, $\langle h''_j, \cdots, h''_{ij} \rangle$ 进行从头的递归操作, 否则对 G, K'进行从头的递归操作。

显然这样得到的新表达式 F'是无拷贝表达式。 (证毕)

由定理 2, 我们实现了任意组合式结构表达式的提取和重用, 由于同步时序电路描述由引理 2 可知它可分离为 μF 形式, 从而同步时序电路公用部件的提取和复用都解决了。

3. 层次性优化

利用布尔代数的方法进行优化的一个问题是它一般只适用于小规模的数字系统的优化, 这里提出的层次性优化是为了适合于大规模数字系统的优化. 我们首先对子部件进行优化, 总体的优化是不影响子部件优化后的结构, 而仅仅利用子部件的代数性质实施总体的优化。比如, 我们首先设计了一个译码器 decoder, 并已对它实施了优化. 在此基础上, 我们设计一个新电路 C, 其中含有 decoder 作为子部件, 此时我们不必考虑 decoder 的具体结构而可利用 decoder 的下述代数性质(R4.1)进行优化。

(R4.1) hd · decoder $\Longleftrightarrow$ not · ∕or · tl · decoder;

hd · decoder $\Longleftrightarrow$ ∕andi · αnot;

当然, 优化的好坏与利用此电路的代数性质很有关, 但是由于此代数性质往往是功能上的效果, 往往比较容易导出。

4. 物理特性的考虑

首先考虑时空问题．设 SOPI 是并行输入串行输出部件，POSI 是串行输入并行输出部件，$[\ ]^s$，α^s，$/^s$ 对应于用空间换取时间的三种算子，而 $[\ ]^t$，α^t，$/^t$ 对应于用时间换取空间的三种高级算子，则有下述规则：

(R5.1): $\alpha^s f \Longleftrightarrow$ POSI · $\alpha^t f$ · SOPI

$/^s f \Longleftrightarrow /^s f$ · apndr · 〔SOPI · tl, last〕

$(f_1, f_2, \cdots, f_n)^s \Longleftrightarrow (f_1, f_2, \cdots)^t$

因此时空上的权衡主要表示在规则的正向使用和反向使用．

其次是负载问题．由于公用部件的最大限度重用，使得其输出负载要求很大，有时不能达到所要求的输出负载．为了解决这个问题，我们增加一个平衡负载部件 B，其逻辑结构为图 4 所示．即 B: x = < x, x, ···, >，其有下述规则：

(R5.2) $(f_1, f_2, \cdots, f_n)$: x ⇒apply · tras · 〔$(f_1, \cdots, f_n)$, B: x〕;

其中 apply: (f, x) = f: x

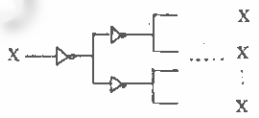

图 4 平衡负载部件 B 的逻辑图

5. 程序变换策略

上面给出了四个方面的优化规则，如何才能使它们协调地工作呢?

静态计值的优化是只欲选择函数可计值运算就立即运算，然后进行公用部件的提取和重用及层次优化，而物理特性的权衡则是在一定的评价函数驱使下进行转换的．

§4. 实 例

下面给出几个例子说明如何通过程序变换对数字电路进行优化．

〔例 1〕 对 ha = 〔and · 〔or, not · and〕, and〕的半加器描述进行优化变换．

解: ha⇒ 〔and · 〔1, 2〕, 3〕 · 〔or, not · and, and〕 (提取作用序列)

⇒ 〔and · 〔1, 1 · 2〕, 2 · 2〕 · 〔or, 〔not, id〕 · and〕 (合并)

变换成的无拷贝表达式，仅用了 4 个逻辑门，原来要 5 个逻辑门，如图 5 所示．

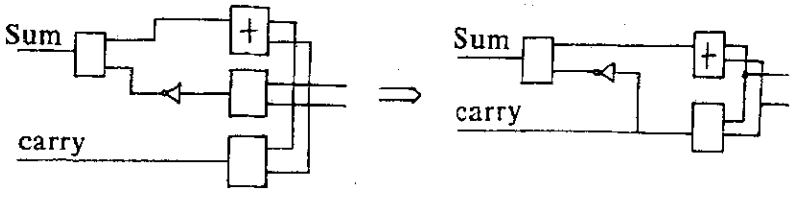

图 5 半加器优化前后的比较

[例 2] 设 COMP 是一个比较器，它接受两个数，输出一个大的数，即 COMP · 〔a, b〕 = gt · 〔a, b〕 ▸a; b，现要设计一个任意 n 个数求最大值的逻辑元件 MAX。根据(R5.1)可有以下两种等价描述：(注：p ▸a; b 是 αand · trans · 〔αor · distl · 〔not · p, a〕, α〕, αor · distl · 〔p, b〕〕(缩写)

(1) $MAX^s = /^s COMP$;

(2) $MAX^t = /^t COMP \cdot apndr \cdot 〔SOP〕 \cdot tl, last〕$

它们的逻辑图如图 6 所示，MAX^s 以空间换取时间，而 MAX^t 则以时间换取空间。

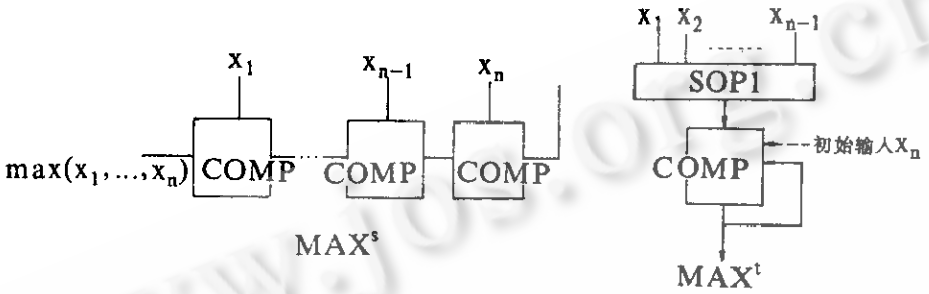

图 6 时空权衡的相互变换

〔例 3〕这个例子选自文章[7]，其功能行为描述为：

circuit · 〔〔s_0, s_1〕, x, z〕 = / or · 〔and · 〔x, hd · decoder · 〔s_0, s_1〕〕,
and · 〔and · 〔x, z〕, 2 · decoder · 〔s_0, s_1〕〕,
and · 〔and · 〔x, z〕, 3 · decoder · 〔s_0, s_1〕〕,
and · 〔and · 〔x, z〕, 4 · decoder · 〔s_0, s_1〕〕〕

其初始逻辑图如图 7.1 所示。优化变换过程如下：

circuit · 〔〔s_0, s_1〕, x, z〕

⇒ / or · 〔and · 〔1, hd · decoder · 〔2, 3〕〕, and · 〔and · 〔4, 5〕, 2 · decoder · 〔6, 7〕〕,
and · 〔and · 〔8, 9〕, 3 · decoder · 〔10, 11〕〕, and · 〔and · 〔12, 13〕, 4 · decoder · 〔14, 15〕〕〕 · 〔x, s_0, s_1, x, z, s_0, s_1, x, z, s_0, s_1, x, z, s_0, s_1〕〕 (提取作用序列)

⇒ / or · 〔and · 〔1, hd · decoder · 〔2, 3〕〕, and · 〔and · 〔1, 4〕, 2 · decoder · 〔2, 3〕〕,
and · 〔and · 〔1, 4〕, 3 · decoder · 〔2, 3〕〕, and · 〔and · 〔1, 4〕, 4 · decoder · 〔2, 3〕〕〕
· 〔x, s_0, s_1, z〕 (合并并进行选择函数修改)

⇒ / or · 〔and · 〔1, 2〕, and · 〔3, 4〕, and · 〔5, 6〕, and · 〔7, 8〕〕
· 〔1, hd · decoder · 〔2, 3〕, and · 〔1, 4〕, 2 · decoder · 〔2, 3〕 and · 〔1, 4〕, 3 · decoder · 〔2, 3〕,
and · 〔1, 4〕, 4 · decoder · 〔2, 3〕〕 · 〔x, s_0, s_1, z〕(作用序列提取)

⇒ / or · 〔and · 〔1, 1 · 2〕, and · 〔3, 2 · 2〕, and · 〔3, 3 · 2〕, and ·

〔3, 4 · 2〕 ·

〔1, 〔hd, 2, 3, 4〕 · decoder · 〔2, 3,〕, and · 〔1, 4〕〕 · 〔 x, s_0, s_1, z〕(合并)

⇒(利用 / or · 〔and · 〔f, h_1〕, ···, and · 〔f, h_n〕〕⇒and · 〔f, / or · 〔h_1, ···, h_n〕〕化简,

or · 〔and · 〔1, 1 · 2〕, and · 〔3, / or · 〔2 · 2, 3 · 2, 4 · 2〕〕〕 ·

〔1, 〔hd, 2, 3, 4〕 · decoder · 2, 3〕, and · 〔1, 4〕〕 · 〔x, s_0, s_1, z〕

⇒or · 〔and · 〔1, 1 · 2〕, and · 〔3, / or · tl · 2〕〕 · 〔1, decoder · 〔2, 3〕,

and · 〔1, 4〕〕 · 〔x, s_0, s_1, z〕

⇒or · 〔and · 〔1, 1 · 2〕, and · 〔3, not · 1 · 2〕〕 ↓ 〔x, decoder · 〔s_0, s_1〕, and 〔x, z〕〕

(利用 R4.1 及静态选择函数计值)

⇒or · 〔and · 〔1, 2〕, and· 〔3, not · 2〕〕 · 〔x, 1 · decoder · 〔 s_0, s_1〕, and · 〔x, z〕〕

(提取 1 · 2, 并进行静态计值)

⇒(利用 R4.1 中 decoder 性质)

or · 〔and · 〔1, 2〕, and · 〔3, not · 2〕〕 · 〔x, and · αnot · 〔 s_0, s_1〕, and · 〔x, z〕〕

⇒or · and · 〔1, 2〕, and · 〔3, not · 2〕〕 · 〔x, not · or · 〔 s_0, s_1〕, and · 〔x, z〕〕

(利用 / and · αnot⇒not · / or).

⇒ · 〔and · 〔1, 2 · not〕, and · 〔3, 2〕〕 · 〔x, or · 〔s_0, s_1〕, and · 〔x, z〕〕(静态计值 not)

优化后的逻辑图如图 7.2 所示. 显然简化很多.

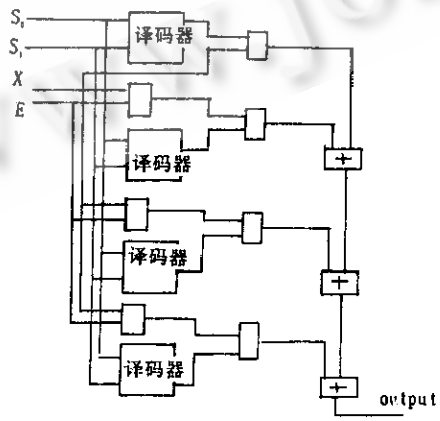

图 7.1 优化前逻辑图

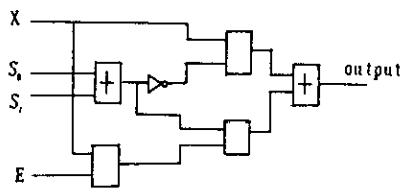

图 7.2 优化后逻辑图

§5. 结束语

本文讨论了如何利用程序变换，根据 FP-1 的程序代数对数字系统进行优化的方法，取得了预想不到的好结果．作者希望将来讨论在／，α，μ等高级算子上的直接的公用部件的提取，从而使优化功能级别更高，同时作者将进一步研究这些变换规则的作用后的合流性和终止性问题．

参考文献

[1] John Backus, "Can Programming Be Liberated from the Von Neumann Style? A Functional Style and Its Algebra of Programs", CACM, 21, pp.613—641.

[2] Marry Sheeran, "μFP—an Algebraic VLSI Design Language", Oxford Univ. Ph. D Theses, TM—PRG—39, Nov. 1983.

[3] Dorab Patel, etc., "νFP, an Environment for the Multi-level Specification, Analysis and Synthesis of Hardware Algorithm", LNCS #202, pp.238—255.

[4] Sun Yong-qiang, "Verification of Systolic Arrays: an FP Functional Approach", J. of Comp., Sci. & Tech., Vol. 3, No.2, 1988, pp.81—102.

[5] R. K. Brayton, C. T. McMullen, "The Decomposition and Factorization of Boolean Expression", Proc. of Int. Sys. on Cir. & Sys., 1982.

[6] A. J. DeGrus, "A Rule-Based System for Optimizing Combinational Logic", IEEE, Design & Test of Comp. Aug. 1985.

[7] William, etc., "Technology Adaption in Logic Synthesis", 23 rd DA Conf., 1986.

[8] T. J.Owalski et al., "The VLSI Design Automation System from Algebra to Silicon", IEEE Design & Test, 1985.

[9] Mead. C & L. Conway, "Introduction to VLSI System", Addison-Wesley, 1980.

[10] 胡振江、孙永强、夏心杰，"函数式语言对数字电路的描述、综合和模拟"，即将在"计算机学报"上发表，1990.6 录用．