Автоматичен синтез на VHDL описание на суматори

В. П. Чумаченко, В.С. Шишков, Т.К.Василева

Технически Университет – София

Abstract This paper presents a computer program for fast adders synthesis. From a given input operand size the program generates structural VHDL models of carry look ahead (CLA) or ripple-carry adders. In case of CLA adders the designer can select from several architectural variations of Brent-Kung adders. The synthesis target can be any standard cells library or a custom library with optimized "o" cells. The resulting VHDL code can be embedded in larger designs (e.g. signal processors) and used as an input for logic synthesis tools like Cadence Synergy and Synopsys.

Въведение

Паралелните суматори са основна съставна част на специализираните схеми, изпълняващи обработка на данни. Според изискванията за разрядност на операндите, бързодействие и ограниченията по отношение на площ за всеки отделен случай се налага разработката на конкретен суматор. Това прави актуално създаването на модул генератор за синтез на суматори, който да удовлетворява изчислителните нужди на специализирания процесор. Съвременните средства за автоматизация предлагат възможност за синтез на принципните електрически схеми от описание на апаратната част с езици от типа на VHDL. Затова задачата за генериране на структурно VHDL описание на различни по архитектура и разрядност суматори е от особена важност за пълното автоматизиране на процеса за създаване на модул генератор на суматори.

Настоящата статия е посветена на разработката на алгоритъм и програма за автоматичен синтез на суматори. Целта е по зададена разрядност на операндите и избрана архитектура автоматично да се генерира VHDL описание, което позволява синтез с използване на конкретни библиотеки стандартни клетки. Предвидена е възможност за промяна на архитектурата на суматора в зависимост от необходимото закъснение.

Алгоритъм и програма за синтез на суматори

Общата структура на суматор с ускорен пренос е показана на фиг. 1.

Блокът за предварителна обработка формира сигналите за генериране (g_i) и разпространение (p_i) на преноса:

$$g_i = a_i \wedge b_i; \;\; p_i = a_i \vee b_i.$$

Вторият блок е префиксна схема базирана на оператора „генериране-разпространение“ GP. На нейният изход се получават частичните преноси c_i. Операторът GP "$\circ$" се дефинира по следния начин:

$$\langle g, p \rangle = \langle g_1, p_1 \rangle \circ \langle g_2, p_2 \rangle$$

където

$$g = g_2 \vee p_2 \wedge g_1 \; ; \; p = p_1 \wedge p_2.$$

По аналогия *груповия* GP оператор се дефинира като:

$$\langle G_{k:i}, P_{k:i} \rangle = \begin{cases} \langle g_i, p_i \rangle; \text{ ако } k = i \\ \langle G_{k:j}, P_{k:j} \rangle \circ \langle G_{j:i}, P_{j:i} \rangle; \text{ ако } k \neq i \end{cases}$$

$k \leq j \leq i$

С $k{:}i$ е означен блока обхващащ битовете от k до i, а с $c_i = G_{0:i}$ - i-я частичен пренос.

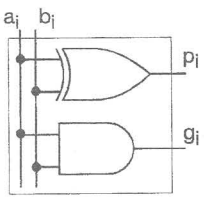
Фиг.1 Блокова схема на суматор с ускорен пренос

Третият блок формира битовете на сумата от частичните преноси:

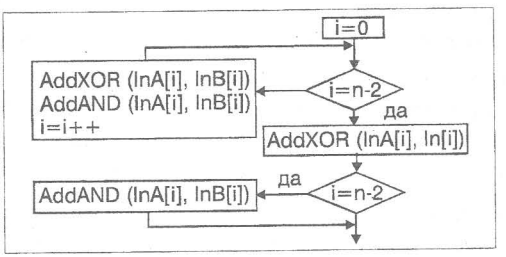
$$S_i = \begin{cases} p_0 \; ; \text{ ако } i = 0 \\ p_i \oplus c_{i-1} \; ; \text{ ако } 0 < i < n. \end{cases}$$

Синтез на входния блок

За преобразуването на всяка двойка входни разряди се използва схемата от фиг.2. Поради липсата на вътрешни връзки във входния блок, в случаите, когато отсъства изходен пренос C_{out}, не се налага изчисление на g_i за i=n-1, с което се спестява един AND елемент. За синтезиране елементите на входния блок се използва алгоритъм, показан на фиг.3.



Фиг.2 Схема за реализация на входния блок



Фиг.3 Алгоритъм за синтез на входния блок

Синтез на блока за пренос

Блокът за *изчисление на пренос* е специфичен за всеки тип суматор. Този блок съдържа свързани помежду си "о" клетки. Независимо от множеството вътрешни възли е възможно опростяване, тъй като за всяка клетка, изчисляваща Ci е необходим само един сигнал ($C_i = G_i$).

Спецификата на блока за пренос за всяка конкретна архитектура изисква и отделен алгоритъм за синтез на структурното описание. В

настоящата публикация се разглежда синтезът на суматори с последователен пренос и този на Брент и Кунг. Изборът е направен поради широкото приложение на тези класове суматори за решаване на конкретни потребителски нужди.

Суматорът с последователен пренос изисква минимални ресурси за апаратна реализация, има минимално натоварване във всеки от изходите и се нуждае от минимална площ за опроводяване на връзките. Затова той намира приложение в по-бавни схеми, към които са наложени силни ограничения по отношение на използвана площ.

Структурата му се формира от последователно свързване на еднотипни клетки. За генерацията се използва следната функция

```
void CarryBlock::GenerateRIP ()
{
    int i;
    for (i = (IsCin? 0 : 1); i < NBits - (IsCout? 0 : 1); i++)
                        AddSimpleDeltaCell (i, i - 1);
}
```

Архитектурата на *суматора на Брент и Кунг* осигурява високо бързодействие при нисък коефициент на натоварване на всяка от клетките. Той се основава на изграждането на двоични дървета от "о" клетки. Първото от тях изчислява всички (g_i, p_i) за които $i = 2^j - 1$. Второто дърво се разполага след първото и завършва изчисляването на останалите (g_i, p_i). Особеност на този суматор е неравномерното закъснение на отделните преноси. При това, колкото повече "о" клетки има в първото дърво на даден разряд, толкова "о' клетката от второто дърво е по-близо до корена и съответния разряд се изчислява по-бързо. Този факт позволява модификация на структурата с оглед постигане на по-голямо бързодействие на суматора като цяло, независимо че натоварването и респективно закъснението в отделен клон ще се увеличат. Целта е суматорът да се изгради с минимален брой клетки, които ще осигурят исканото закъснение.

С бързодействието може да се варира като се знае, че при добавянето на клетка към първото дърво в съответния разряд, неговата клетка от второто дърво се свързва към по-главен клон и суматорът се ускорява.

Количеството допълнителни "о" клетки за всеки разряд се определя от : Extra=GetOnes(i) - MaxDelay - Count(i),
където MaxDelay е търсеното закъснение, Count(i) – брой "о" клетки в първото дърво преди допълването и GetOnes(i) – брой единици в числото в двоичен код, умножени по 2.

За всеки разряд във второто дърво се поставя завършваща изчисленията клетка, чието свързване се определя от първото дърво.

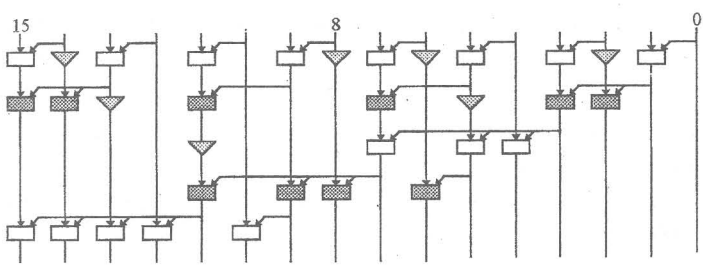
Проектиране на "о" клетки

Използвани са три способа за реализация на блока за пренос: архитектура с еднотипни клетки, реализация с инверсни сигнали и с използване на стандартни клетки от библиотеките AMS/ES2. Тъй като критичният път се формира от "о" клетки е необходимо те да са възможно най-бързи. При някои технологии може да се постигне допълнително ускоряване ако изходите на клетките се инвертират. Това изисква дефинирането на два типа клетки: бяла "о" клетка и черна "о" клетка - фиг.4 *а* и *б*.

Фиг.4 Бяла (а) и черна (б) "о" клетка

На всеки ред от матрицата в блока за изчисляване на преноса се разполагат "о" клетки от един тип. Тъй като те могат да се свързват само с клетки от противоположния тип, се налага използването на инвертори.

Примерната схема на блока за изчисляване на пренос в суматора на Брент и Кунг, реализиран с черни и бели клетки, е показана на фиг. 5.

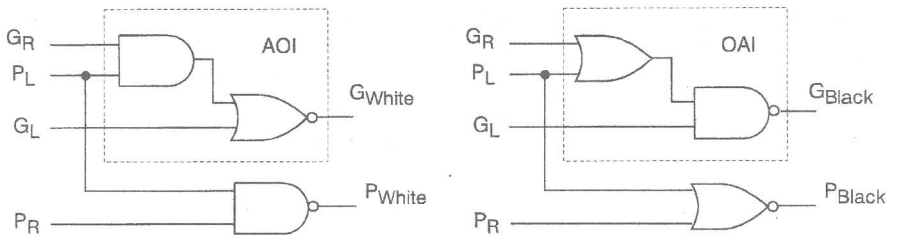


Фиг.5 Схема за изчисление на пренос в суматор на Бренд и Кунг с бели и черни клетки

В стандартните CMOS библиотеки за технологии AMS/ES2 липсват изброените по-горе елементи. За улеснение на потребителя се предлага вариянт със стандартни клетки. При него "о" клетката е изградена от стандартни елементи като се цели използване на минимално количество от тях с оглед намаление допълнителното закъснение от свързващите шини. Най-подходяща за случая е реализацията с черни и бели клетки, при което *P* и *G* подклетките се получават с един елемент.

Уравнението на бялата *P* подклетка $P_{White} = \overline{P_L P_R}$ съответства на двувходов NAND, докато бялата *G* подклетка $G_{White} = \overline{G_{L+} P_L G_R}$ изисква елемент AOI. На фиг.6*а* е показана бяла "о" клетка, изградена с библиотечни стандартни елементи.

Аналогично уравнението на черната *P* подклетка $P_{Black} = \overline{P_L + P_R}$ е на двувходов NOR, докато на черната *G* подклетка $G_{Black} = \overline{(G_{L+}P_L)G_R}$ се изпълнява от елемент OAI. Черната "о" клетка със стандартни елементи е дадена на фиг.6*б*.



Фиг.6. "о" клетка със стандартни елементи а) бяла "о" клетка б) черна "о" клетка

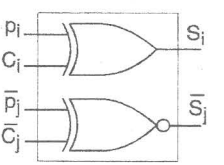
Синтез на изходния блок

За изпълнението на уравнента на сумата от частични преноси са необходими два вида елементи, реализиращи функцията XOR или инверсната XNOR. Те се свързват по показаният на фиг. 7 начин.

Дали даден пренос е инверсен зависи от последният елемент във веригата за изчисляването му. Елементите AOI или бялата "о" клетка изчисляват C_i и се различават от останалите по функцията IsInverted (). Ако липсва входен пренос C_{in}, то $S_0 = s_0 \oplus 0 = s_0 = p_0$ разрешава да се спести един XOR елемент.

Свързването на елементите от изходния блок се осъществява със следния цикъл:

```
if (IsCin) {
        if ( InC[0]->Parent->IsInverted() )
                OutsS[0] = Store->AddXNOR (InC[0], InP[0]);
        else  OutsS[0]= Store->AddXOR  (InC[0], InP[0]);}
else OutsS[0] = InP[0];
for (i = 1;i < NBits;i++){
        if ( InC[i]->Parent->IsInverted() )
                OutsS[i] = Store->AddXNOR (InC[i], InP[i]);
        else  OutsS[i] = Store->AddXOR  (InC[i], InP[i]);}
```



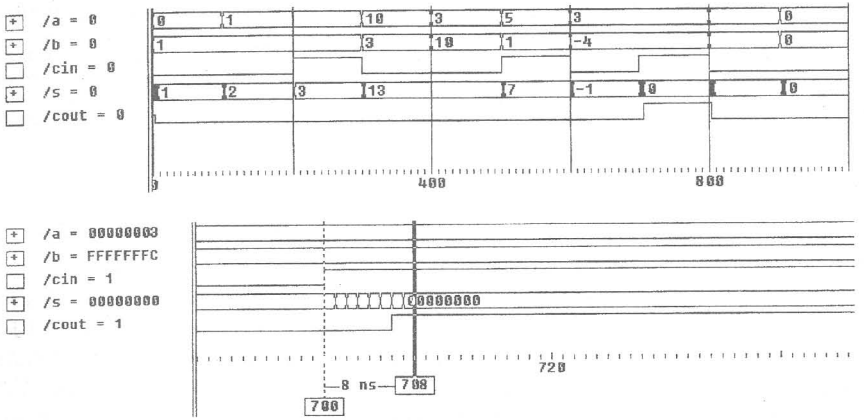
Фиг.7 Изходен блок

Примери за синтез на суматори

Резултатите от синтез на суматори с различно бързодействие са дадени в таблица 1. Генерираният VHDL код на суматорите е верифициран с VHDL симулация с пакета VSystem. Резултатите от симулацията (фиг.8) потвърждават функционалните свойства на суматорите и техните времеви показатели.

Вариант	Брой "о" клетки	Критичен път	Max натоварване FO	Брой клетки с max FO
Последователен пренос	15	15	2	1
Брент и Кунг (B&K)	24	6	4	2
B&K - вариант 1	26	5	5	5
B&K - вариант 2	31	4		9

Таблица 1 Сравнение на синтезирани схеми на суматори



Фиг.8 VHDL симулация на синтезиран суматор на Брент и Кунг

От резултатите може да се провери максималното бързодействие на синтезирания суматор.

Заключение

Разработен е алгоритъм и програма за автоматично генериране на структурно VHDL описание на бързи суматори. Те позволяват синтез на суматори с произволна разрядност при избрана архитектура. От особено значение е гъвкавостта на алгоритъма за модификация на архитектурата на суматорите с оглед постигане компромис между необходимите ресурси и бързодействие. Синтезът се осъществява и с наличните CMOS AMS/ES2 библиотеки стандартни клетки в пакета CADENCE. Генерираният VHDL код на суматорите е верифициран с VHDL симулация с пакета VSystem като резултатите потвърждават функционалните свойства на суматорите и позволяват проверка на бързодействието им.

Литература

A.R. Omondi, Computer Arithmetuc Systems, Algorithms, Architecture and Implementation, Prentice Hall, 1994

R.Lipsett, C.Schaefer, C.Ussery, VHDL:Hardware Description and Design, Kluwer Academic,1989