第6章

トランジスタ構成まで意識した演算回路の最適化設計

――ゲートの内部構造を理解して設計の幅を広げよう

佐藤 証

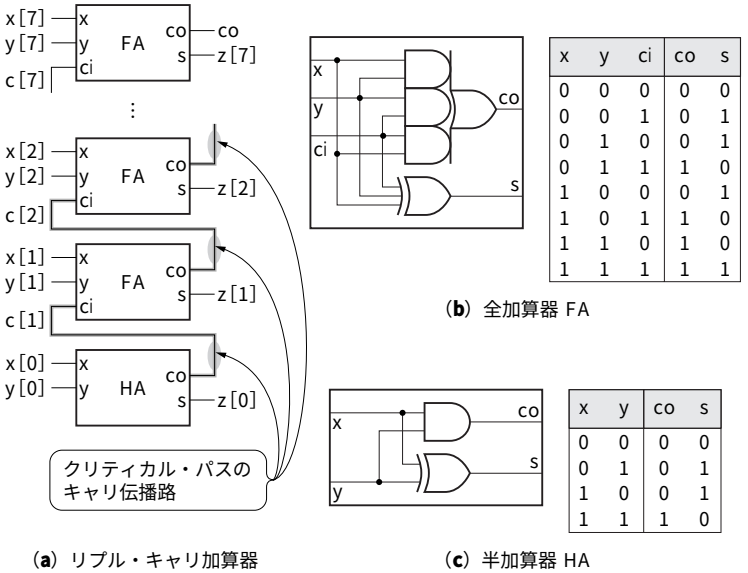
ここでは，ゲート・レベルよりさらに掘り下げて，トランジスタ・レベルの回路構成を考慮した最適化の事例を紹介する．大規模な回路の場合，人手によるライブラリ・マッピングで性能を向上できる余地がかなりある．こうした最適化を行うには，データシートを熟読し，ゲートの内部構造やトランジスタの特性などを押さえることが肝要と筆者は言う．（編集部）

ひと昔前は，加算器を作るだけでも「ハーフ・アダーとフル・アダーのセルをこう組み合わせて，ここにキャリを飛ばせて…」などとあれこれ考えながら設計していました．一方，現在では，

```
assign z = x + y;
```

と1行書けば，後は論理合成ツールが制約条件に合わせてネットリストを作ってくれます．さらにC言語ベースの回路設計が普及しはじめている今日，筆者の得意とするトランジスタのマスク・パターンを直接描くカスタム・レイアウトのスキルは，きわめて特殊な場合にしか必要とされないのかもしれません．

しかし，マイクロプロセッサのアーキテクチャやアセンブリ言語に詳しい人が質の高いプログラムを書くように，ASICライブラリのトランジスタ構成や回路特性に関する知識があると，RTLでも高性能な回路を設計できます．とくに，算術演算回路はアーキテクチャの自由度が高いため，どれだけ多くのテクニックが使えるかによって性能に大きな差が出てきます．

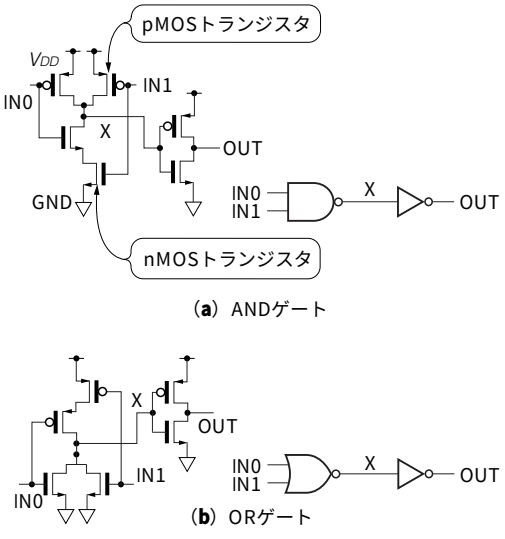


x	y	ci	co	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

（**a**）リプル・キャリ加算器　（**b**）全加算器 FA　（**c**）半加算器 HA

図1　リプル・キャリ加算器の構成

(a)の下方の加算器（FAまたはHA）からのキャリがAND-ORゲートを通って上のFAへと順々に伝播していく．



（**a**）ANDゲート　（**b**）ORゲート

図2　2入力ANDゲートとORゲートのトランジスタ構成

ANDゲートやORゲートは，NAND-INVERTERやNOR-INVERTERといったように二つのゲートがシリアルに接続されていることがわかる．

そこで，ここでは「回路設計テクニックの幅を広げる」という意味で，簡単な演算回路を例にトランジスタ構成を意識した人手によるライブラリ・マッピング「手マップ」の技を紹介したいと思います．

1 リプル・キャリ加算器の最適化

図1(a)は論理回路の入門書にもよく出てくるリプル・キャリ加算器です．8ビットの整数x[7：0]とy[7：0]を入力すると，8ビットの整数z[7：0]と1ビットのキャリcoを加算結果として出力します．全加算器は半加算器2個で記述されることも多いのですが，ここでは多入力のXORとAND-ORゲートを使っています．下の全加算器FA(あるいは半加算器HA)からやってきたキャリは，AND-ORゲートを通って上のFAへと順々に伝播していきます．その名は，このキャリの伝播が「さざ波(ripple)」を連想させることに由来しています．

リプル・キャリ加算器はシンプルで小型である半面，キャリが伝播するクリティカル・パスは演算ビット数に比例して長くなるため，とても低速です．**図1**のキャリ伝播のパスには2入力ANDゲートが一つと，6入力AND-ORゲートが七つ入っています．高速な演算にはキャリ・ルックアヘッド加算器などが用いられますが，ここではあくまでも例題ということで，リプル・キャリ加算器の高速化を考えてみましょう．

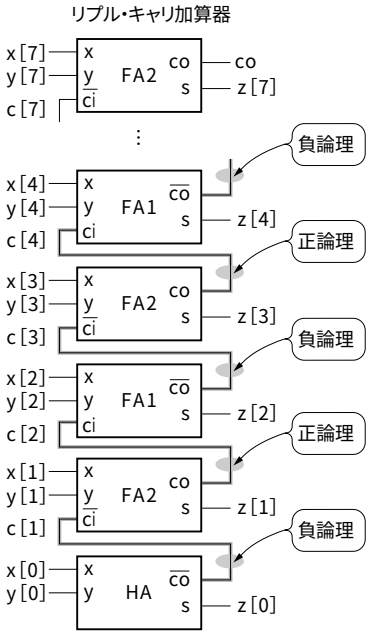


図3
高速(?)リプル・キャリ加算器の構成
図1でクリティカル・パスとなったANDとAND-ORを，それぞれNANDとAOIに置き換えた．

HDLによるリプル・キャリ加算器のRTL記述は，XOR，AND，ORの三つの演算子を用いて**図1**のとおりに作るのが正解です．しかし，トランジスタ・レベルの設計者の視点では，この回路は不合格です．CMOSライブラリの基本ゲートはほとんどが反転出力を持ちます．直接コンポーネントを指定できるのであれば，それらのゲートにNANDやAND-OR-INVERTER(AOI)などを使うことで，速度と規模の双方で回路を最適化できます．例として，2入力のANDゲートとORゲートのトランジスタ構成を**図2**に示します．これから，CMOSのANDゲートやORゲートはNAND-INVERTERやNOR-INVERTERといったように二つのゲートがシリアルに接続されていることがわかります．

そこで，**図1**のクリティカル・パスのANDとAND-ORを**図3**のようにそれぞれNANDとAOIに置き換え，キャリの極性を負論理・正論理と交互にすれば，八つ分のインバータ遅延を削減することができます．

● 多入力ゲートの速い入力ポートと遅い入力ポート

次にライブラリの各ゲートの内部構造を考慮した高速化テクニックを紹介しましょう．**図2**のANDゲートとORゲートの各入力は論理面では等価なのですが，よく見るとトランジスタ間の配線は異なっています．このため，入力キャパシタンスや出力までの遅延時間に差が出てきます．で