第5章

特集 入門！ VHDLによる回路設計の基礎

VHDL

VHDLによる実践的ディジタル回路設計

～RSA暗号器を組み合わせ回路と順序回路で設計する～

吉田たけお/尾知博

前章までで，ディジタル回路に関する内容を一通り終えたので，以下では，VHDL設計におけるヒントをいくつか述べておく．また，VHDLを使い始めて半年程度の初心者の記述例を紹介するので，こちらも参考にしていただきたい．後半では，今までより規模の大きい回路をVHDLを用いて設計してみる．設計する回路は，近年，情報セキュリティの分野で注目を集めている公開鍵暗号の一つであるRSA暗号の暗号器である．この設計をとおして，VHDLによるディジタル回路の設計の流れを見ていこう．なお，ここで取りあげるRSA暗号器は「Design Wave設計コンテスト2001」の“課題2”となっている．（筆者）

5.1 ディジタル回路の設計方針

ディジタル回路（論理回路）は，「組み合わせ回路」と「順序回路」に大別される．ここでは，これらの回路の違いやHDLを用いた回路設計の流れについて述べる．

5.1.1 データパスと制御回路

一般にディジタル回路（同期式順序回路）は，

(1) **データパス**（datapath）

(2) **制御回路**（controller）

から構成される．データパスとは算術演算，論理演算などのデータ処理を中心に行う回路であり，制御回路とはデータパスを制御するための回路である．

データパスの設計では，①まず，必要なプリップフロップ（FF）やレジスタを配置し，②次にFFやレジスタ間でのデータ処理を行う組み合わせ回路を設計する，という手順が踏まれる．

制御回路は，データパス内のFFやレジスタへの制御信号（リセット信号やイネーブル信号など）を生成する．この制御回路は，通常，ステートマシン[注1]として設計される．すなわち，ディジタル回路の設計には，データパスとしての同期式順序回路と組み合わせ回路の設計および制御回路としてのステートマシンの設計が含まれていることになる．

5.1.2 組み合わせ回路と順序回路の違い

ディジタル回路の多くは，組み合わせ回路としても順序回路としても設計可能である．実際に本特集では，本章の後半で「RSA暗号器」と呼ばれる回路を両方の方式で設計する．ここでは，ディジタル回路を組み合わせ回路として設計する場合と順序回路として設計する場合の違いについて述べる．

◆回路規模の違い

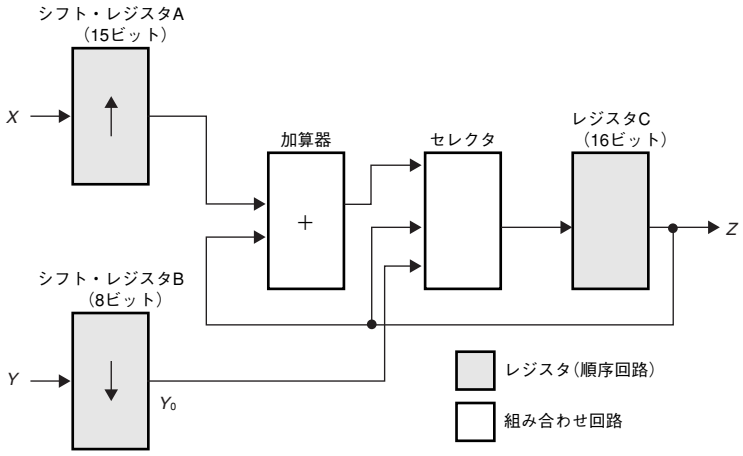
たとえば，乗算器を考えてみよう．乗算器を順序回路として実現する場合，通常，被乗数をシフト・レジスタに格納しておき，シフトしながら加算を行うという方法をとる．この場合，32ビット乗算器を順序回路として実現すると，4,000ゲート程度に収まる．一方，組み合わせ回路として実現した場合，10,000ゲート程度の回路規模になる．なお，上記のゲート数は正確な値ではない．設計の仕方によって，これらの数値は大幅に異なるので，あくまでも参考値にとどめていただきたい．

また，もともと規模の小さい回路の場合，上記のようにはならない．たとえば，8ビット乗算器を組み合わせ回路として実現した場合，500ゲート程度になる．これに対して，順序回路として実現すると，700ゲート程度になってしまう．

◆処理時間の違い

上記のような32ビット乗算器を順序回路として実現した場合，計算が終了するまでに，32クロック（非乗数の桁数）分

注1：ステートマシンは，学術的には順序回路と同義である．しかし，実際の設計現場では，データパスを制御するための回路をステートマシンと呼ぶことが多いので，本書でも順序回路とステートマシンを使い分けることにする．

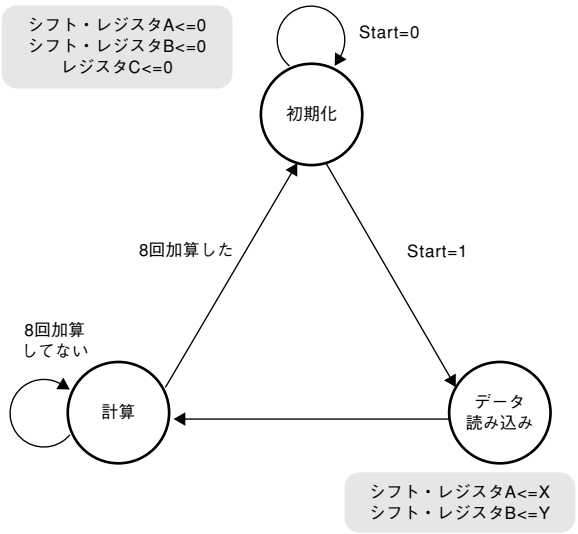


〔図5.1〕8ビット乗算器のデータパスの例

の処理時間を必要とする．一方，組み合わせ回路として実現した場合，1クロック（組み合わせ回路の遅延時間）分の処理時間で計算が終了する[注2]．

以上のように，ディジタル回路を組み合わせ回路として実現すると，一般に，回路規模は大きくなるが，出力が得られるまでの処理時間は格段に短くなる．もともと規模の小さい回路や演算器などの高速処理を行わせたい回路などは，できるかぎり組み合わせ回路として設計するのが望ましい．

ところで，後半で設計するRSA暗号器では，暗号の安全性を考慮した場合，後述するように1024ビットから2048ビット程度の乗算器が必要となる．このような大きな乗算器を組み合わせ回路として設計すると，数百万ゲートを越える規模になってしまい，非現実的である．このような場合は，乗算器を順序回路として設計する必要がある．



〔図5.2〕8ビット乗算器の制御回路の状態遷移図

5.1.3 ステートマシンを設計する目的

前述のように，ステートマシンは，データパスの制御回路として用いられる．ここでは，8ビット乗算器を例にして，ステートマシンを設計する目的を説明する．

◆8ビット乗算器のデータパス

いま，$X=(x_7, x_6, \cdots, x_0)$，$Y=(y_7, y_6, \cdots, y_0)$とし，$Z=X \times Y=(z_{15}, z_{14}, \cdots, z_0)$を計算する8ビット乗算器を順序回路として実現することを考える．8ビット乗算器を順序回路として実現するには，被乗数XをMSB側にシフトしながら加えていけばよい．具体的には，**図5.1**のような回路構成にすればよい．

図5.1では，被乗数XをMSB側にシフトさせるシフト・レジスタA，乗数YをLSB側にシフトさせるシフト・レジスタBおよび，計算結果を保持するレジスタCの計3個のレジスタを用いている．加算器およびセレクタは組み合わせ回路として実現する．

セレクタは，$y_0=1$のときに加算器の出力を選択し，$y_0=0$のときはレジスタCの出力を選択する．すなわち，シフト・レジスタAにより，被乗数Xの2倍の値を次々と生成し，シフト・レジスタBのLSBが1のときに，その値を加算していく．この過程を乗数の桁数回繰り返すことにより，乗算の機能を実現する．

◆8ビット乗算器の制御回路

しかし，**図5.1**に示す回路は，乗算器として不完全である．なぜならば，加算を行う回数を制御する機構がこの回路にはないためである．この制御機構を実現するために，ステートマシンが用いられる．ステートマシンの設計においては，「状態」の概念が必要となる．そこで，**図5.1**のデータパスに行わ

注2：クロックの周期や組み合わせ回路の遅延時間を考慮していないので，順序回路のほうが32倍の時間を必要とするとは言えない．