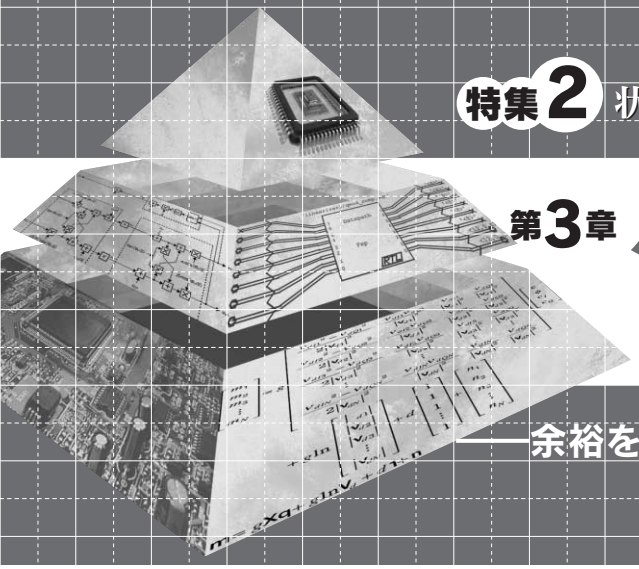


第3章 ハードウェアはこうモデリングする！

――余裕を持たせつつ，確定できるところから詰めていく

小林芳直

本稿では，ハードウェア設計における状態遷移モデルを取り上げる．ステート・マシンの設計にあたっては，むだな状態遷移が生じないように，また必要な信号を取り出しやすいように状態を定義しなければならない．（編集部）

ハードウェアの設計では部品を作る作業と，その部品を組み立てる作業が並行して進められます．部品の機能が決まっていても，その実現方法を早期に固定することはありません．システムを仮組みし，精度を上げてから，最終組み立てに移るという手順を踏みます．こうすることで完成度の高いシステムを作ることができます．

1. 信号を取り出しやすいように状態を定義する

回路図を描くとき，「一つ一つの部品をしあげて，最後に組み上げれば完全動作」とはなかなかいきません．特に，新しいシステムを組むときは，部品にある程度の“のりしろ”や“合わせ誤差”の余裕を持たせておいて，組み合わせの自由度を残したほうが得策です．将棋でなかなか自分の戦法をはっきりさせないまま局面を進めていくとか，スリザーリンクや数独といった穴埋め式パズル[注1]で確定できる箇所から少しずつ解いていく，といった感覚です．最後に組み合わせたときに，初めて全体と部分の組み合わせの必然性が見えるというのが，じょうずなシステムの組みかたです．

同じような機能を持つ部品でも，回路サイズが大きくて動作が速いものもあれば，小さな回路でゆっくりと確実に動作するものもあります．回路の構成に合わせながら，使用する部品を過不足なく組み合わせていきます．システムを組み上げるときは，回路ブロックの中で部品の性能を制限する制約（同期，協調，排他制御など）がいくつかあります．まず部品を作り，仮組みして，部品を削りながら合わせ込んでいく作業が必要です．バランスの取れたシステムを組むためには，このようなボトムアップ的な作業とトップダウン的な作業の組み合わせがとても重要です．

また，部品の選択において，高機能の部品を少量使う方法と，単機能の部品を多数使う方法では，一般に後者のほうが，システムを組みやすいと考えられています．部品の粒度（granularity）によってシステムの保守のしやすさが決まります．

●状態割り当てのこつ

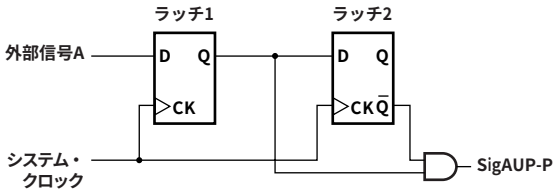
ステート・マシンの設計には二つの課題があります．一つはステート・マシンを正確に動作させることです．そしてもう一つは，ステート・マシンから必要な信号を取り出すことです．出力信号が取り出しやすいように状態を定義していくと，使いやすいステート・マシンになります．これを簡単な例で説明します．

「ある外部信号が‘0’から‘1’に変化したとき，1システム・クロック幅の信号を作りたい」という課題が与えられたとします．まず，以下のようなアプローチを考えます．

1）微分回路を使う

ラッチを二つ並べておいて，最初のラッチに外部信号Aをつなぎます（**図1**）．外部信号Aが‘1’になると，ラッチの値が“00”→“10”→“11”と変化します．外部信号Aが‘0’になると，“11”→“01”→“00”となり，アイドル状態に戻

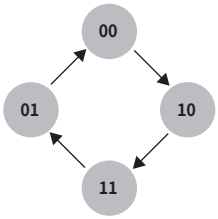
注1：「スリザーリンク」や「数独」の詳細については，ニコリのWebサイト（http://www.nikoli.co.jp/）やパズルジャパンのWebサイト（http://www.puzzle.jp/）を参照のこと．



〔図1〕ラッチを二つ並べたところ
ラッチを二つ並べて，1システム・クロック幅の信号を作った．

	外部信号A	ラッチ1	ラッチ2
信号Aが入る前	0	0	0
信号Aが入った(1になった)とき	1	0	0
		1	0
		1	1
遷移の結果		1	1

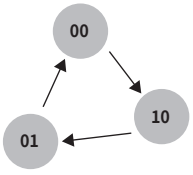
	外部信号A	ラッチ1	ラッチ2
	1	1	1
信号Aが消えた(0になった)とき	0	1	1
		0	1
		0	0
遷移の結果		0	0



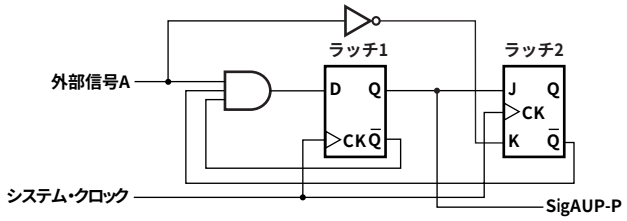
〔図2〕微分回路を使った場合(状態の数は4)
外部信号Aの変化に応じて，ラッチの値が遷移する．

	外部信号A	ラッチ1	ラッチ2
信号Aが入る前	0	0	0
信号Aが入った(1になった)とき	1	0	0
		1	0
		0	1
遷移の結果		0	1

	外部信号A	ラッチ1	ラッチ2
	1	0	1
信号Aが消えた(0になった)とき	0	0	1
		0	1
		0	0
遷移の結果		0	0



〔図3〕専用回路を使った場合(状態の数は3)
むだな状態を減らし，ラッチの値をそのまま出力するように改善した．



〔図4〕専用回路の回路図
外部信号が‘1’になったときのステート・マシンの動作が“00”→“10”→“01”，外部信号が‘0’になったときの動作が“01”→“00”となるように回路を組んだ．

ります(**図2**)．目的とする状態は“10”なので，これをデコードすればよいというわけです．

微分回路を使うことは回路設計の常識とされています．しかし，これをステート・マシンとして見ると，かならずしも優れた方法とは言えません．なぜなら，外部信号が来たときに“00”→“10”→“11”と変化するのはよいのですが，外部信号がなくなったときに“11”→“01”→“00”と変化するのは，むだな状態遷移と言わざるをえません．それから，出力信号を見るためには“10”をデコードする必要があるので，ANDゲートを用意して出力させることになります．できれば，ラッチの出力を直接使いたいところです．

そこで，別の解として次のようなアプローチを考えます．

2) 専用回路を使う

外部信号Aが‘1’になったときのステート・マシンの動作を“00”→“10”→“01”とし，外部信号が‘0’になったときに“01”→“00”と遷移するように定義します(**図3**，**図4**)．このステート・マシンが“10”になったときに出力信号を取り出せばよいのですが，最初のラッチが‘1’になるのは状態“10”以外にはありません．つまり，最初の状態ビットが‘1’になれば，それをそのまま出力します．状態の数は3となり，むだな状態がありません．

このように，出力信号とまったく同じ動作を行う状態ビットを設けることにより，協調動作に適した(後工程に優しい)ステート・マシンを設計することができます．

●ワンホット・ステート・マシンとの使い分け

一つの状態に一つの状態ビットをかならず割り当てる手法として，ワンホット・ステート・マシンがあります．ワンホット・ステート・マシンとは，状態の数だけ状態ビットを用意して，それぞれの状態ビットが排他的に一つだけ‘1’になるようにしたステート・マシンのことです[1]．論理合成ツールなどが対応しているステート・マシンであり，動作速度を上げやすく，機能変更も容易です．

今回の例で言うと，状態ビットを三つ用意して，“100(アイドル状態)”，“010(外部信号あり)”，“001(待機)”という三つの状態を用意します．回路は**図5**のようになります．

外部信号Aが‘1’になると，状態を“100”→“010”→“001”と変化させます．そして外部信号Aが‘0’になると，“001”→“100”と変化させます(**図6**)．出力信号はラッチの出力をそのまま使えます．