基礎から学ぶ Verilog HDL & FPGA 設計

第5回
ステート・マシンの設計

中野浩嗣，伊藤靖朗

デバイスの記事　ビギナーズ

本連載は，さまざまな回路をVerilog HDLで設計していき，最終的には小型のCPUを実現することを狙いとしている．今回は，CPUの状態を制御するステート・マシンを作る．ステート・マシンを順序回路として設計し，シミュレーションとFPGAを用いた動作確認を行う．（筆者）

● CPUの基本動作と状態遷移図

基本的には，CPUは次の二つの動作を繰り返します．

●命令フェッチ

メモリに格納されている機械語命令を取り出す（フェッチする）．

●命令実行

取り出した機械語命令を実行する．

さらに，動作を行っていない**待機**状態を加えると，CPUの基本動作は，**図1**のように表すことができます．このような図を**状態遷移図**と呼びます．

図1の状態遷移図の矢印は，状態遷移規則を表しています．遷移要求があるたびに遷移が行われます．状態遷移規則には，無印のものと，**動作開始**と**動作終了**のラベルの付いたものがあります．一つの状態から二つ以上の外向きの矢印が出ている場合，外部からの合図によって遷移先の状態が決まります．例えば，待機状態で動作開始の合図があると，命令フェッチ状態に遷移します．待機状態で合図がない場合は，待機状態に遷移，つまり状態が変わりません．命令フェッチ状態からは，命令実行状態に遷移し，命令実行状態で合図がない場合は，命令フェッチ状態に戻ります．命令実行状態で動作終了の合図があると，待機状態に遷移します．従って，動作終了の合図がない間，命令フェッチと命令実行を交互に繰り返します．

状態遷移図により定義された状態遷移を実現するものがステート・マシンです（**図2**）．このステート・マシンでは，遷移要求，および動作開始と動作終了の合図が入力され，現在の状態を出力します． 遷移要求があるたびに，**図1**の状態遷移図に従って状態遷移を行います．

● ステート・マシンの設計

現在の状態を保持するのにフリップフロップを用いれば，ステート・マシンは順序回路として設計することができます．実際にCPUで用いるステート・マシンを設計しましょう．本連載で設計するCPUの状態遷移はもう少し複雑で

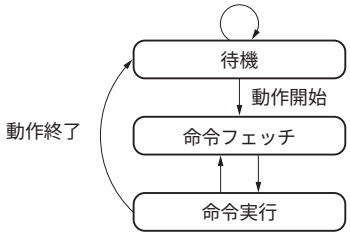

図1
CPUの基本動作

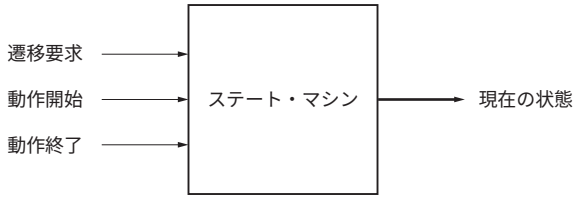

図2　ステート・マシン

KeyWord　命令フェッチ，命令実行，状態遷移図，動作開始，待機状態，動作終了

す．**図3**はその状態遷移図です．

五つの状態を持ち，IDLEが待機状態，FETCHAとFETCHBが命令フェッチのための状態，EXECAとEXECBが命令実行のための状態です．命令フェッチと命令実行のために，それぞれ2クロック・サイクルが必要なため，二つの状態を割り当てています．**図4**はこの状態遷移図を実現するステート・マシンです．

このステート・マシンは，後で順序回路に実装するのを想定して設計します．ステート・マシンは，clk，reset，run，cont，haltの五つの入力を持ちます．出力は現在の状態csです．

基本的に，クロックclkの立ち上がりで状態遷移が行われます．リセットresetが0になると，clkの立ち上がりに関係なくIDLEに非同期に遷移します．状態がIDLEのときrunが1ならば，FETCHAに遷移します．その後は基本的に，FETCHA→FETCHB→EXECA→FETCHA→というように三つの状態FETCHA，FETCHB，EXECAを順に遷移し，命令フェッチと命令実行を繰り返します．

状態がEXECAのとき，haltが1なら，IDLEに遷移します．これはCPUの動作の終了を意味します．

状態がEXECAのとき，contが1なら，EXECBに遷移します．これは，EXECAの1クロック・サイクルだけでは命令実行が完了せず，2クロック・サイクル必要な場合に対応します．EXECBに遷移した後，FETCHAに戻ります．

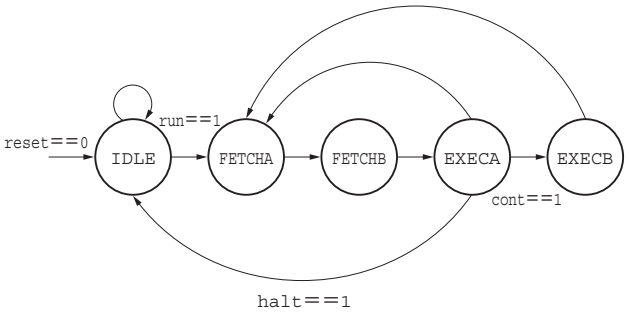

図3　CPUの状態遷移図

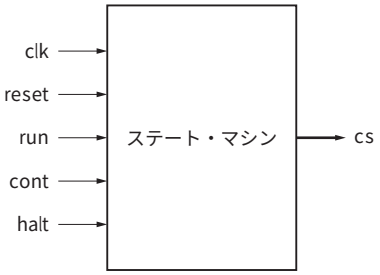

図4
CPUのステート・マシン

● ステート・マシンのVerilog HDL記述

ステート・マシンを順序回路で実装するために，レジスタ（フリップフロップ）を用いて現在の状態を記憶します．**図3**のステート・マシンは五つの状態を持つので，3ビットのレジスタ（つまり3個のフリップフロップ）を用います．3ビットあれば，2進数表現で000から111までの八つ（$2^3=8$）の状態を区別することができるので十分です．

2ビットでは最大4（$=2^2$）状態なので，5状態を区別するには不足します．

リスト1は，この方法によりステート・マシンを実現するVerilog HDL記述です．11行目のreg文で，3ビットのレジスタcsを宣言します．このレジスタを用いてステート・マシンの状態を保持することにします．

1行目～5行目では，define文を用いて，状態名と値を対応づけています．

13行目から始まるalways文でcsの値を決定しています．

14行目は，resetが0のときcsの値がIDLEになる非同期リセットを定義しています．

16行目から始まるcase文では，resetが1でclkの立ち上がりが発生したときの遷移先を定義しています．csの値がIDLEの場合，17行目のif文が実行され，runが1のときcsにFETCHAつまり，3'b001が代入されます．runが0のときは代入が行われないので，csはIDLEのままです．csがFETCHAのときはFETCHBに，FETCHBの場

リスト1　ステート・マシンのVerilog HDL記述（state.v）

```
 1    'define IDLE 3'b000
 2    'define FETCHA 3'b001
 3    'define FETCHB 3'b010
 4    'define EXECA 3'b011
 5    'define EXECB 3'b100
 6
 7    module state(clk,reset,run,cont,halt,cs);
 8
 9      input clk, reset, run, cont, halt;
10      output [2:0] cs;
11      reg    [2:0] cs;
12
13      always @(posedge clk or negedge reset)
14        if(!reset) cs <= 'IDLE;
15        else
16          case(cs)
17          'IDLE: if(run) cs <= 'FETCHA;
18          'FETCHA: cs <= 'FETCHB;
19          'FETCHB: cs <= 'EXECA;
20          'EXECA: if(halt) cs <= 'IDLE;
21                  else if(cont) cs <= 'EXECB;
22                  else cs <= 'FETCHA;
23          'EXECB: cs <= 'FETCHA;
24          default: cs <= 3'bxxx;
25        endcase
26
27   endmodule
```