第3章 無駄な初期化の無効化から1GHzオーバークロック実験まで！

ステップ3…Raspbian高速起動の限界に挑戦！

宗像 尚郎

ラズベリー・パイ2の起動シーケンス

● Linuxボードの起動処理は複雑

Linuxボードの電源を入れてからユーザの操作を受け付けるまでのシステム起動時間は，複合的な要因で決まります．

Linuxカーネルの起動処理がスタートするまでは，ブート・デバイスの読み出し性能などのハードウェア的な要因が支配的です．

カーネル起動後，アプリケーションが利用可能になるまでの準備期間はソフトウェア的な起動手順の設定に依存します．

● ラズベリー・パイはハードもソフトも情報がないけど…がんばってみます

ボード依存部分の最適化にはSoCやボードのドキュメントを参照する必要があります．一般的な組み込みLinux開発ボードと比較すると，ラズベリー・パイのハードウェア関連の技術情報は非常に限られています．ローダの仕様を解説したドキュメントもありません．現時点ではラズベリー・パイ2で採用されたARM v7対応のBroadcom BCM2836の仕様書は公開されていません．CPUコア以外の周辺機能については，公開されているBCM2835のドキュメント(1)が参考になるようです．

今回は，ユーザ・コミュニティの情報や，ネット上に公開されている実験結果，非公式情報なども参照しながら，ラズベリー・パイ(Raspbian)の起動高速化を検討します．

● ラズパイの起動シーケンスの全体像

コミュニティでの議論を集約すると，ラズベリー・パイの起動は**図1**の三つのステージに分かれているようです．

ステージ2まではほぼハードウェア仕様で決まっていて，チューニングの余地がありません．今回は，ステージ3以降のソフトウェア初期化について，Raspbian(kernel3.18)に採用されているLinuxの起動設定を最適化してみます．起動時間の短縮に挑戦してみましょう．

▶**ステージ1**：ROM化されたSoC内蔵ブート・ストラップによる初期化
- 電源投入時は，SoC内蔵ROMにプログラムされたブート・ストラップが起動(ブート・ストラップは，ARM CPUではなくVideoCore GPUによって実行される)
- ブート・ストラップは，ブートローダ/boot/bootcode.binをロード

▶**ステージ2**：ブートローダbootcode.binによる，メモリ，周辺IPの初期化
- ブートローダ(boot.bin)は，DDRメモリ(LPDDR2)を初期化
- ブートローダは，/boot/bootcode.binをDDRメモリに展開
- bootcode.binは，GPUのファームウェア/boot/start.elfを読み込む
- start.elfは，/boot/config.txtの情報を元にVideo，HDMI，メモリを初期化
- start.elfは，/bootからLinuxカーネル・イメージをDDRメモリに展開

▶**ステージ3**：カーネル起動&アプリケーションの実行準備
- start.elfは，/boot/cmdline.txtに定義されたラズベリー・パイ用起動パラメータをカーネルに渡す
- カーネルは起動時にデバイス・ツリー・データベース・ファイル(dtbファイル)を読み込んでカーネル機能を設定

図1　ラズベリー・パイの起動シーケンス

Raspbian(kernel3.18)のソフトウェア起動処理

● レガシーな起動メカニズムSysVinitを採用

今回利用したRaspbian(kernel3.18)では，SysVinitと呼ばれるレガシーな起動メカニズム注1が採用されています．SysVinitは，起動スクリプトを順番に実行していく仕組みです．

ソフトウェアの起動シーケンスを最適化するためには，OS起動後にどの設定を参照しながらプログラム

注1：将来的にはsystemdと呼ばれる新しい起動メカニズムに変更される可能性があります．