ソフトウェアに合わせたハードウェアをFPGAを使って実現

ソフトとハードの境界を積極的に越えてみよう

FPGAを使うことでハードウェアの仕様を柔軟に設計できる．ここではソフトウェア開発者の側からハードウェアの仕様を提案し実現することで，CPUクロックを上げることなくシステムのパフォーマンスを改善できた事例などについて紹介する．（編集部）

伊倉 広徳，佐々木 明彦

ソフトウェア（以下，ソフト）により，ハードウェア（ハード）との境界に近い部分を記述するようなとき，ハードの仕様の微妙な部分に泣かされることがしばしばあります．表面上の仕様は，必要機能を充分に満たしているように見えていても，いざ採用し，その上でソフトを構築しようとすると，痒いところに手が届かなかったり，時にはソフトではどうにもフォローできない穴が発生したりする場合もあります．

こんなとき，「もしハードがこうなっていてくれたら」と思うことは誰しも経験があるのではないでしょうか．ぜいたくな変更は求めなくとも，わずかな改変や小さな機能追加だけでソフトの細部の記述や動作がスムーズになることが期待できるケースも多々あります．

ここでは，まずソフト屋として経験した歯痒い思いを共有していただいた上で，そのソフト屋が境界を越えてソフトに都合のよいハードを作ってみたという話をします．

● 現場で起こった問題

筆者らがこれまで関わってきたシステム開発の中で，ハードの仕様に泣かされてきた問題にはいろいろありますが，単純な例を挙げます．いずれも普及している既存のASICを使用したプラットホーム上での開発だったため，後から到着して境界部分に立たされた筆者らが何らかの問題対処をする必要があったケースです．

実際の現場では，このような問題がさらに絡み合って，より複雑化しているケースも多く見られます．

(1) 細かい話――ステータスはリード・クリアかライト・クリアか

ある特殊な低速通信インターフェースLSIのドライバを記述したときの話です．ここでは単純なUARTコントローラのドライバを書く場合を想像してください．

CPUから見えるLSIのステータス・レジスタには，

- 送信FIFOに空きあり
- 受信FIFOにデータあり
- 各種エラー状態検知

などのビットが並んでいます．送受信FIFOの状態を表すビットはFIFOの動作に連動しているため，これらの状態変化はFIFOの操作（データ・レジスタへのライトやリード）によって変化しますが，エラー状態検知はこのビットでしか情報を拾えません．

割り込みハンドラの冒頭でステータス・レジスタを読み，複数のエラー状態をまとめて拾います（**リスト1**）．エラーが発生している場合はそれを記録し，後で実行される詳細なエラー処理ルーチンをスケジュールします．

エラーが発生したという情報は拾ったので，次のエラー状態の発生に備えてステータス・レジスタのビットをクリアしておく必要があります．ただし，読み込んでから，クリアするまでにほかのエラー状態が発生する可能性があるため，拾ったビットのみを適切にクリアする必要があります．

リードするだけで立っていたビットがクリアされるという仕様なら，読み出すだけで完了です．今回のケースなら，ソフト屋にとってこれが最も扱いやすいでしょう．個別のビットが特定値を書き込むことでクリアするという仕様でも，若干手間は増えるものの，今回拾ったビットのみを書き戻せば完了です．

リスト1　割り込みハンドラの処理

```
extern unsigned int error_cond;

intr_handler()
{
    error_cond |= read_status_register();
    if (error_cond != 0)    notify_error();

    /* Clear error condition bits in Status Register... */
        :
}
```


KEYWORD――割り込みステータス・クリア，DMA，ローカル・バス，ソフトウェア割り込み，割り込みハンドラ，MMC/SDカード・コントローラ