特集1

第1章

FPGA/PLD トラブル・シューティング

トラブルこそエンジニアの糧
――トラブル時の心構え

山口耕作

「トラブル」というとあまりイメージが良くない．できれば一生出会いたくないものだ．しかしポジティブ思考でとらえれば，それはスキルアップのための最高の教材であり，そのときの経験は貴重な財産となる．（編集部）

理想の心構えとは

企画書の書き直しを命じられた主人公が「ラッキー！もっと良い企画書を作ってやるぜ！」と叫ぶテレビ・コマーシャルがありました．ちょっと極端ですが，理想的なトラブル時の心構えだと筆者は感じました．

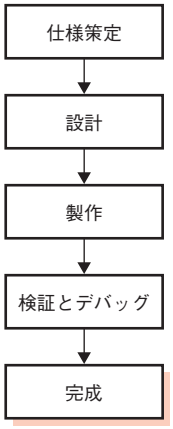
トラブルは予測不可能であり，それを解決する時間が少ないことがほとんどです．しかしこういうときこそ「新しい発見がある」という気持ちで，一度深呼吸してから取り組みましょう．

また，たとえトラブルの原因が自分の設計にない場合でも，「自分の知識と経験が正しければ元々トラブルは起こらなかったはず」と思いたいものです．起こってしまったトラブルは，なかったことにはできません．どうせなら自分の知識や経験をさらに進化させる機会ととらえることができれば，気分が乗って，解決もよりスムーズにいくのではないでしょうか．

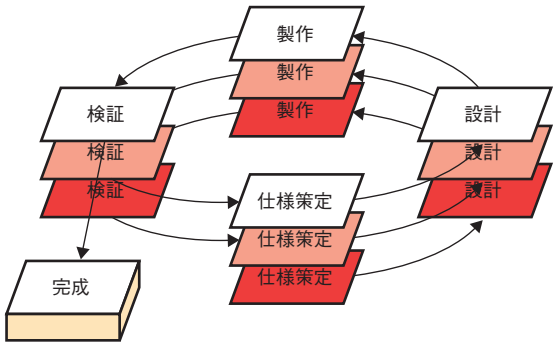
デバッグは嫌い

ところで，特集タイトルの「トラブル・シューティング」というと「デバッグ」を連想される方も多いと思います．デバッグはとてもめんどうな作業であることが多く，好きな方はいないでしょう．筆者はこのことばそのものがあまり好きではありません．デバッグは「虫取り」と訳されるように，問題点を大局的に見ないでそのポイントだけをつぶしていくやりかたを指すイメージがあるからです．

このことばが流行（？）しているのは，ウォータ・フォール（「滝」の意）設計モデル（図1（a））の考えかたがあるからだと筆者は考えます．たしかに一つのものを開発するとき，一般的に仕様策定から設計，製作，検証とステップを踏んでいき，その一つ一つのステップをしっかり完了していくことは非常に重要です．一方，トラブルが起こったとき，その原因は仕様にあるのかもしれませんし，



（a）ウォータ・フォール設計モデル



（b）スパイラル設計モデル

〔図1〕
代表的な設計モデル
最近ではより洗練された設計モデルが提唱されている．今では両者とも旧世代の遺物になっている感がある．

本特集の読みかた

筆者の方が実際に経験した，あるいは身近で起こったトラブルとその解決方法を，事例ごとにまとめました．設計時の守秘義務などの関係で，完全なデータを示すことができない場合もありますが，ご了承ください．

各事例ごとに，トラブルが起こったときの設計環境やトラブル度（解決にかかった時間）を示しています．

設計環境は，実際に使用していたデバイス・ファミリ名や設計ツール名を示します．したがって，明記したデバイス・ファミリや設計ツールでのみ起こるという意味ではありませんのでご注意ください．ほかのデバイス・ファミリ，設計ツールでも同様な現象が起こることが予想できるものについては，カッコ内にコメントとして示します．

トラブル度は，解決にかかった時間と以下の目安をもとに，筆者の方に５段階で評価していただきました．

1. ケアレスミス，データシートなどの誤解
2. 通常の検証方法で数時間で解決
3. 通常の検証方法で数日で解決
4. 検証方法を変えて解決
5. 解決に長期間を要した．解決困難（情報不足，ツール/デバイスのバグが原因）など．

解決してみれば，原因は意外と単純だったということも少なくありませんので，客観的な評価とは異なる場合もあります．単純なミスなのにトラブル度が大きい場合は，「原因を発見しにくいトラブル」というように読みとってください．（編集部）

設計にあるのかもしれません．本来なら問題のあるステップに戻って解決策を実施する必要があるのに，「滝」は上れないものとして，検証時のデバッグで全部解決しようというのは無理があると思います．

そこで筆者は，スパイラル設計モデル（図１（b））のように，作ったものを評価し修正していくことを開発作業の一部と見なしています．デバッグは，製品出荷後などの，どうしても後戻りできない場合に実施する最後の手段であると考えています．

設計開発の心構え

デバッグ工程を長期間とってある開発計画書を見かけることがあります．問題は必ず発生するものですが，これでは良い設計が最初にできないという「あきらめ」を表しているように思えてなりません．まず，より良い設計を行って，みんなが嫌いなデバッグをしなくてすむことを目指したいものです．

これはトラブル時の心構えというよりも，設計開発全体に対しての心構えということになりますが，筆者はいつも自分が設計したものに対して，「最高であり，最低である」と感じるようにできればと考えます．設計当時は自分の能力を100％発揮したはずですから，自分の最高の設計であるわけです．しかし，その設計を後で振り返ったとき自分が当時より進歩していれば，そのとき自分が組み込んでしまった問題が目につき，逆に最低と思えるはずです．そして「もっとうまく設計してやる」という気持ちになります．トラブルがより少なく，トラブってもすぐに直して「さらに最高の設計」ができる設計者でありたいと思っています．

トラブルを体でおぼえよう

読者のみなさんには，今回の特集を単に読むだけでなく，紹介されるトラブル事例を実際に体験されることをお勧めします．それも，解決する前の状態をじっくり経験してください．どうなることが問題かを知っておけば，トラブルを未然に防ぐことができます．もちろん解決方法を理解するのも大事ですが，まず原因をよく知り，トラブルを起こさないことがベターです．

理論的な裏づけがあるうえで直感力が働く「ラショナル・エンジニア[1)]」になるためにも，問題の本質を見て，触って，体におぼえ込ませましょう．

参考文献

1）西村芳一，「新人エンジニアの失敗の日々」，『Design Wave Magazine』，2001年5月号，pp.38-63.

やまぐち・こうさく
（株）ダイヘン