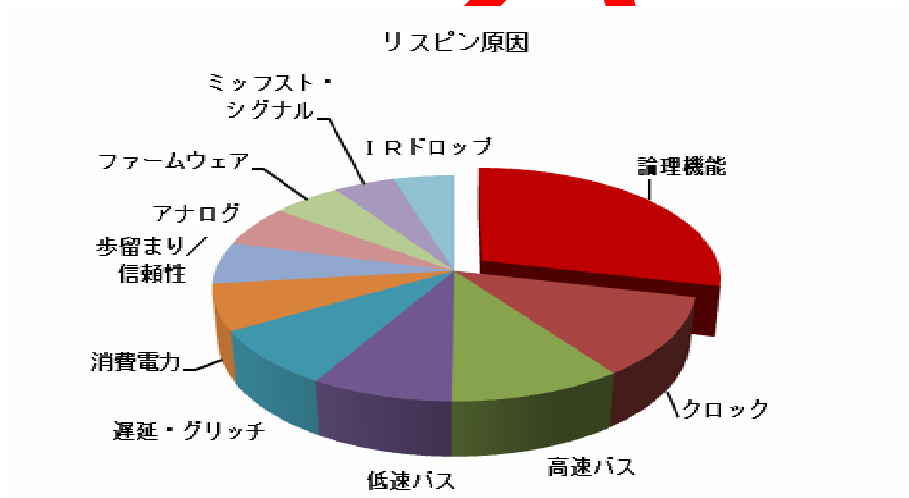
付録　ドキュメント作成

１．仕様書／報告書の作成

１－１　仕様書の必要性

１．１　ＬＳＩ設計のリスピン

ＬＳＩ設計におけるリスピン（再設計による再試作）のデータの一例を図１．１に示すように、機能論理による不具合が多いことが判る。この不具合の上位４項目を仕様書で明確にしてわかりやすく記載してあれば、防げることができることになる。



LSI の最初の試作時に発生した不具合の原因をまとめたものである．原因のうち91 %は，論理機能的な不具合だった．この割合は，2004 年には70%程度であり，増加している．出展： Collett International,January 2005

図１．１　リスピンの原因

１．２　ＬＳＩ設計の品質低下の要因

ＬＳＩ設計の品質を低下させる要因は、仕様の曖昧さや機能検証の不足によるところがほとんどである。

１．２．２　不具合の原因分類

表１．１　不具合の原因

工程	項目	原因
仕様検討	仕様漏れ	・数年後のビジネス環境の予測が困難 ・実際に動くモノを見ないとイメージがわかない ⇒仕様があいまい ⇒プロジェクト後段の納期が迫っている時期に仕様変更 ・製品サイクルが短命化しているのに、仕様検討～システム検証までの期間が長期化し仕様決定時期が早期化 ・設計委託後の仕様変更は追加費用を請求されるのでわざと仕様にあいまいさを残す
	仕様のバグ	・SW と HW の分割ポイントが分からない ・要求仕様書⇒設計仕様書（SoC の機能を記述）
	実用化の検証漏れ	検証仕様書、ソフトウェア仕様書等の仕様書間の漏れ
設計	設計者の仕様認識ミス	・各設計者のレベル差や専門性(SW/HW/基板等)が違う
	設計者間のコミュニケーション不足	・あいまいな仕様を各担当者が誤認(品質より工数優先で選択等)
	シミュレーションモデルや IP のバグ	・検証が複雑になったため ・ASIC デザイン数激減⇒IP 採用したデザイン数の減少⇒IP の検証不足
	新規ツールやデバイスの取り扱いミス	LSI ベンダー間、試作と量産デバイス間のツールやデバイスの違い
検証	境界での検証漏れ	ブロック/IP/SW/HW 間の検証漏れ、基板の配線やビア等のミス
	組み合わせの検証ミス	ブロック、IP、SW、基板上の部品等の複雑な組み合わせミス
	EDA ツール用モデルのバグ	・設計や検証用 EDA ツールに使うモデルのバグ ・ＩＰのセキュリティ確保のためにシミュレーションモデルを簡易化⇒精度が悪く、実機との誤差が影響する
	EDA ツールのバグ	EDA ベンダーのツールは常に機能追加するのが宿命なのでバグが潜みやすい
	設計者と検証者の目線が同一	設計者の思い込みによる不具合検出が困難
	検証仕様が不明確	どこまで検証するのか明確にしていないと、検証不足が起き易い
	不具合原因の解析漏れ	・納期が切迫している時期に、要因増員や時間外労働等による人的ミスやテスト項目の削減
	バグ修正後の再検証でのテスト漏れ	・どこまで検証すれば十分だか分からない

（１）仕様の曖昧さ

はっきり決まっていない要求仕様を元に SoC 設計を進めた場合、実機での動作が分かってくるにつれて、仕様策定者の認識と実機との差異が顕在化され、開発期間の末期には想定外の仕様変更になることが SoC 品質保持を妨害する一つの理由である。

１．仕様書／報告書作成

仕様があいまいだとスケジュールや開発費用の見積りも甘くなりがちである。また、開発末期には検証期間短縮や費用削減の要求が高まることになる。ところが、そうなると品質に関する作業は開発末期に集中することになり、この品質確保の為の工数が削られると当然品質保持は難しくなる。

（２）検証不足

ASIC でのシステム検証は SoC 開発フローの終盤に行う。納期が迫っている時期での期間短縮や費用削減要求が厳しく求められる中で、どこまで検証するのか不明確だと検証不足を招いて製品品質に大きく影響を与える。

１．３　仕様書の必要性

主な仕様書の必要性を洗い出すと以下のようになる。

- 個人の設計管理
- プロジェクトメンバー間の意思疎通
- 仕様検討レビュー
  - 見落としチェック
  - 問題点の洗い出し
  - 設計誤りを最小限に抑える
- 再利用性の向上
  - 設計資産
  - 設計ノウハウ

１．４　報告書の必要性

設計主な報告書の必要性を洗い出すと以下のようになる。

- 個人の設計管理
- プロジェクトメンバー間の意思疎通
- 設計結果レビュー
  - 見落としチェック
  - 問題点の洗い出し
  - 設計誤りを最小限に抑える
- 再利用性の向上
  - 設計資産
  - 設計ノウハウ

１－２　「わかりやすい」と「わかりにくい」

２．１　「わかりやすい」とは

- **認知心理学から**

人間の五感に入力された情報を短期記憶（１次記憶）し、その記憶を処理してから長期記憶（２次記憶）に格納されるという。ＣＰＵに例えば、キャシュメモリに相当するようなもの働きをすることがわかっていきた。そのため、一度に多くの情報を未整理に提供すると混乱するのである。

一時的に短期記憶に格納された情報は、人間が覚えておこうと意識が働くことによりカテゴリ毎に分類して長期記憶への処理を行う。

・　頭の中の辞書を選ぶ
・　情報を分解する
・　情報を整理する
・　情報の意味を決定する
・　情報の論理性を確認する

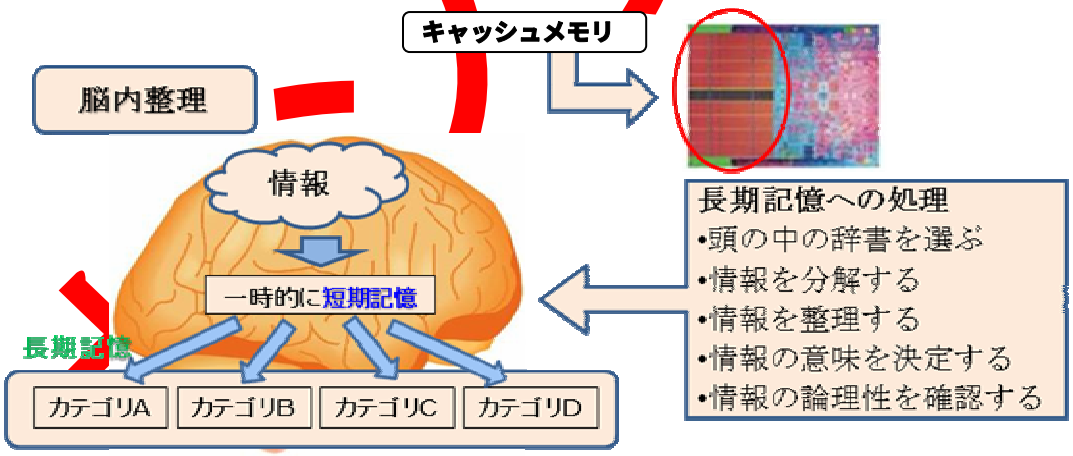


図１．２　脳内整理のイメージ

２．２「分かる」とは

- 『分かりやすい』とは『分かっている』状態に移行しやすいという意味である
- 『分かっている』状態とは『情報が脳内整理棚の一区画に仕舞われている』状態である

２．３　「わかりにくい」の例

日本語の文章を英訳する時に、主語や述語がはっきりしておらず、日本語の文章から直すことがよくある。このことは、日本には察し合う文化なるものがあるようで、言葉を省略しても察して考えることが多い。これは、仕様書を作成する上では、非常に不都合で厄介な問題である。昔から仕様書を読む場合、「行間に隠れている表現を読み取る」とか、「表現されていいない業界標準がある」とかいわれていた。

わかりにくい例を分類すると以下のようになる

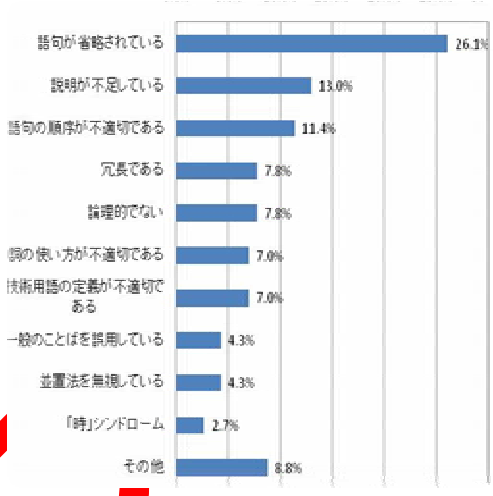


図１．３　わかりにくいの分類

① 語句が省略されている。 26.1%
② 説明が不足している。 13.0%
③ 語句の順序が不適切である。 11.4%
④ 冗長である。 7.8%
⑤ 論理的でない。 7.8%
⑥ 助詞の使い方が不適切である。 7.0%
⑦ 技術用語の定義が不適切である。 7.0%
⑧ 一般のことばを誤用している。 4.3%
⑨ 並置法を無視している。 4.3%
⑩ 「時」シンドローム 2.7%
⑪ その他 8.8%

問題：

次の文は８通りの内容に読めると思うので、それぞれの内容を示す８つの文を書いて下さい。

「黒い目のきれいな女の子がいた」

１－３　技術文書

３．１　誰に見せるか

「誰に何を伝えるか」を明確にすることから始める。意思決定者へのラブレターだとも思って、伝える相手を明確にする。

- **敵を知り、己を知れば、百戦危うからず**
  - なんのために作るのか？
  - 読む人と技術文書との関係は？
  - 読み手が欲しているものは？
  - 自分が持っている情報は？

図１．４　開発工程とドキュメント

３．２　作成するタイミングと技術文書

設計に関わるドキュメントは、作成するタイミングと関係者に伝えるタイミングが重要である。時期を逃すと生きた情報ではなくなってしまい無駄になる。

企業によっては、設計手順書やＱＣ工程表などによって、そのタイミングと仕様書や報告書の内容が規定されているので、確認しておく必要がある。

３．１　情報の分析と整理

- まず、重要ポイントを書き並べる
- 要点を先に、詳細は後に書く
- 不必要な情報は書かない

技術文書に共通して重要なのは「段落で要点が明確に述べられていること」である。最初に、段落をまとめる際の基本。次に、技術文書を構成する際の基礎となる「見出し書式の基本」について説明する。

①「要点」を最初に述べる

要点が曖昧になりやすい原因は、曖昧な段落と明確な段落の違い

②段落を「構造化」する

不要に長い段落の見直し方／不要に改行が多い段落構成の見直し方／補足・注記の適切な使い方

③「視点」を明確にする

「視点」とは「 執筆者・製品の視点」と「読者の視点」、機能仕様書と製品マニュアルの違い

④見出し構成の基本

見出しランクと見出し番号書式の基本／ページ数に応じた改ページ構成の選び方

１－４　仕様書／報告書の作成

４．１　情報整理は５W１H

情報は基本的には５Ｗ１Ｈで分解して整理するとわかりやすい。一般的な例として以下に示す。

（１）目的(WHY)

開発する目的を記載する。開発することによる可能な限り数値ベースでその効果を明確にする。開発が進み開発内容が具体化してくると、仕様やスケジュールの変更を余儀なくされる場面に

（２）方針(HOW)

開発の基本方針を記載する。関係者の考え方や判断する上での重要な羅針盤となる。この項は、一度まとめてしまえば大部分を複数の案件で使いまわすことができる。

（３）機能一覧(WHAT)

この項には”何を作るのか”を記載する。詳細が別のドキュメントにも記載されるため、記述はアウトラインのみに留める場合もある。

①　製品用途および周辺回路

②　ハードウェア構成

③　目標性能

④　制限事項

（４）開発体制(WHO)

開発の規模に応じてプロジェクトマネージャ、プログラマ、デザイナ等の役割やコンポーネント（部品）毎の担当者等を記載する。一部にしか関わらないメンバに関しては、参加の時期も明確にする。また、リソース管理方針があれば、それも明確にしておく。

（５）開発スケジュール(WHEN)

着手時期、リリース時期、ＥＢ発注、試作時期、量産時期などを明確にする。リリース時期は必要に応じて「ＤＦフェーズ」「ＣＦフェーズ」のように区切る。最終段階での大きな手戻りを防ぐために、要所要所でイベントを設けて開発依頼元にもチェックもらうことは非常に重要である。

（６）開発環境と検証環境(WHERE)

開発に使用する言語やＥＤＡツール、検証に使用する言語やＥＤＡツールなどを明確にする。

４．２　仕様書の構成例（記載すべき項目）

どのようなものをどのように設計や検証していくかを記述したのが仕様書であるといえる.

① 概要
② 開発方針
③ システムの概要を記す.
（LSI仕様，特徴，周辺デバイスの仕様など）
④ 端子名称や配列
⑤ マクロ設計仕様
・ マクロレベルの機能説明を記す
（LSI内のマクロ構成，マクロ間インターフェイス，タイミングなど）
⑥ アーキテクチャ設計仕様
⑦ アーキテクチャレベルの機能説明を記す
（マクロ内の詳細な構成，動作，タイミング，制御方法など）
⑧ テスト仕様
⑨ LSI およびマクロに対するテスト項目

など、企業内での取り決めがされている作成マニュアルや作成手順書に従う。

４．３　報告書の構成例（記載すべき項目）

目標特性に対してどのように設計や検証したかを記述したのが報告書であるといえる.

① 全体概要（概要、目的、結果、結論）
（できる限り全体概要は１ページ目に記載する。上司が１ページ目を読んでわかるように）
② 仕様スペック表（Min,Typ,Max）に対する結果
（スペック表がない場合にはそれに代わる代用特性）
③ 各項目と詳細説明
（図やグラフなどを利用して視覚で分かりやすく）
④ 成果や残された課題
⑤ データ一覧、ソースコード、回路図

⑥　参考データ
（競合他社との比較：内部資料）
⑦　チェックシート結果、ＤＲＢＦＭシート
など、企業内での取り決めがされている作成マニュアルや作成手順書に従う。

４．４　「分かりやすい技術文書」のテクニック

ここでは、分かりやすい技術文書を作成するために、基本とテクニックノポイントをいくつか紹介する。

（１）わかりやすい文書の基本

① 主語と述語を明確に記述する
② 箇条書きやテーブルを活用していく。
③ プログラムのように段下げをする。
④ 仕様書の最初に、言葉の定義を記述し、意味を明確にしておく。（特に略語）
⑤ 目次を整理する：仕様書に限らずドキュメントを作成する場合には、目次の項目立てや階層に注意を払う
⑥ チームメンバが同じ意識で仕様書を書くようにする。（その為のミーチェィングも必要）

（２）作成心得のポイント

① 文書も分析、設計が必要
② 書く前に、品質を作り込む
③ 提出相手への気配り
④ 魅せる文書を心掛ける
⑤ 文章は簡潔で短めに
⑥ 技術用語の説明と一貫性と統一
⑦ フォントの種類、サイズを統一
⑧ 箇条書きと段落を活用
⑨ 振り返りとチェック／確認