2006 年度

学位論文 博士(学術)

製品開発における設計負荷とその低減
―設計プロセスの効率化と改善に関する研究―

主査・指導教員: 黒須誠治教授

副査: 大江建教授

早稲田大学大学院 アジア太平洋研究科

博士後期課程 国際関係学 国際経営専攻

学籍番号: 4003S321-0

氏名: 山本 靖

2006 年 2 月 17 日

目次

目次 i
図表目次 viii
第 1 章　序論 1
1.1　研究の目的 1
1.2　研究の背景と対象 2
1.3　研究の焦点と本論で取り組む課題 3
1.3.1　設計負荷とその低減 3
1.3.2　製品価値と設計プロセス 4
1.3.3　スケジュール、コスト、製品仕様 5
1.3.4　本研究における設計プロセス管理の概要 5
1.3.5　本論で取り組む課題 7
1.4　研究の方法と経緯 9
1.4.1　研究の方法 9
1.4.2　研究の経緯 9
1.5　本論文の構成 11
参考文献 13
第 2 章　設計プロセスにおける問題と先行研究 15
2.1　設計負荷の問題と本章のねらい 15
2.2　設計プロセスの概念 15
2.2.1　機械設計 16
2.2.2　ソフトウエア 19
2.2.3　電気電子機器 21
2.2.4　半導体 22
2.2.5　一般設計プロセス 26
2.3　イタレーションに関する先行研究 27
2.3.1　イタレーションと既存の管理手法 27

2.3.2 イタレーションの発生と含意 29
2.3.3 イタレーションの分類と効率化 30
2.4 イタレーション・モデルに関する先行研究 32
2.4.1 マトリックス手法 32
2.4.2 ネットワーク手法 36
2.4.3 イタレーション・モデルの有効性の確認 38
2.4.4 イタレーション・モデルに関するまとめ 42
2.5 イタレーション発生時の現行の管理手法の限界点 42
2.5.1 スケジュールの心理的抑圧 43
2.5.2 リソースの制約 43
2.5.3 タスクの所要時間と反復修正の回避思考 44
2.5.4 設計プロセスにおける進捗の測定 44
2.5.5 スケジュールの蓋然性 46
2.5.6 作業分割と作業範囲記述の限界点 46
2.5.7 イタレーションの分類と混同 47
2.5.8 イタレーションが見過ごされてきた理由 47
2.6 その他の設計負荷：人間の行動特性と設計知の活用の不十分性 49
2.6.1 人間の行動特性 49
2.6.2 設計知の活用の不十分性 50
2.7 PDCA サイクルによる設計品質と設計の生産性の向上 52
2.8 本章のまとめ 53
参考文献 54
第 3 章 設計プロセスにおける状況認識と不確実性の源泉 60
3.1 製品開発のリスクと因果連鎖 60
3.2 製品開発のリスクと設計プロセス 61
3.2.1 製品開発におけるリスクの基本要素 61
3.2.2 スケジュール、コスト、製品性能 63
3.2.3 設計プロセスにおける不確実性の因果連鎖 64
3.3 設計プロセスにおけるスケジュール・リスク 64
3.3.1 スケジュール・リスクの概要 64
3.3.2 スケジュールの不確実性の因果関係 65
3.3.3 想定内のイタレーション 66
3.3.4 想定外のイタレーション 69
3.3.5 イタレーションの範囲とイタレーションの所要時間 72
3.3.6 人間の行動特性 73
3.3.7 設計知の不十分な活用 76

3.3.8 時間のゆとり 77
3.3.9 不知の不知 77
3.3.10 不確実性を低減させる活動 78
3.3.11 スケジュールの不確実性の因果連鎖の総合 78
3.4 設計プロセスにおける設計開発コスト・リスク 80
3.4.1 設計開発コスト・リスクの概要 80
3.4.2 設計開発コストの不確実性と因果関係 80
3.5 設計対象物の性能リスク 83
3.5.1 性能リスクの概要 83
3.5.2 製品性能の不確実性の因果関係 84
3.6 本章のまとめ 88
3.6.1 因果連鎖ダイアグラム 88
3.6.2 リスク・マネジメント 88
3.6.3 設計負荷 89
参考文献 89
第 4 章 設計プロセスにおけるイタレーションの モデル理論と実践 91
4.1 設計プロセスにおけるイタレーションの問題 91
4.2 イタレーションのモデリング・アプローチ 92
4.3 DSM 94
4.3.1 DSM の先行研究 94
4.3.2 半導体設計への適用の意義 95
4.3.3 DSM によるモデリング 96
4.4 一対比較行列によるモデリング 98
4.5 一対比較行列によるモデルの固有値、固有ベクトル、総所要時間量ベクトル $\boldsymbol{U}$ 99
4.5.1 一対比較行列の固有値と固有ベクトル 99
4.5.2 総所要時間量ベクトル $\boldsymbol{U}$ の計算手順 101
4.5.3 固有値問題 103
4.6 一対比較行列によるモデルの半導体設計事例への適用 104
4.6.1 半導体設計事例 104
4.6.2 タスク間の依存性の強度 105
4.7 解析結果 109
4.7.1 $1/(1-\mathrm{Re}[\lambda_n])$、固有ベクトル、総所要時間量ベクトル $\boldsymbol{U}$ の算出結果 109
4.7.2 第一固有ベクトルの解析 110
4.7.3 第二固有ベクトルの解析 112
4.7.4 総所要時間量のベクトル $\boldsymbol{U}$ の解析 112
4.7.5 半導体 IP 置き換えのシミュレーション 112

4.8 考察 115
4.9 本章のまとめ 116
参考文献 117
第 5 章 設計プロセスにおけるイタレーションと リスク・アセスメント 121
5.1 リスク・アセスメント 121
5.2 イタレーションのモデリング 122
5.2.1 イタレーションのモデリングの従来手法と DSM 122
5.2.2 DSM によるイタレーションのモデリング 123
5.2.3 活動時間及び活動コストの算出式 124
5.3 リスク・モデルとリスク値の算出式 125
5.3.1 リスク・モデル 125
5.3.2 リスク値の算出式 127
5.4 リスク評価のフレームワークの半導体実設計事例への適用 127
5.4.1 リスク評価のフレームワーク 127
5.4.2 半導体実設計事例への適用 129
5.5 考察 137
5.5.1 リスク評価のフレームワークの運用 137
5.5.2 リスク・モデルの信頼性 138
5.5.3 リスク評価のフレームワークにおける課題 138
5.6 本章のまとめ 138
参考文献 139
第 6 章 無駄な工程・過剰機能・過剰品質と価値のダイナミック・モデル 142
6.1 リーン思考と設計開発工程の効率化 142
6.2 設計開発工程の効率化に関する先行研究 143
6.2.1 二律背反のメカニズム 143
6.2.2 トレードオフ検討に関する実証 144
6.2.3 トレードオフ検討の限界 146
6.3 効率化事例とリーン思考 147
6.3.1 日本の自動車メーカによる効率化事例 147
6.3.2 リーン思考 148
6.4 価値モデルの提案 149
6.4.1 製品価値と価値形成 149
6.4.2 価値のダイナミック・モデル 151
6.4.3 価値のダイナミック・モデルの運用 153
6.4.4 価値モデルにおける課題 155
6.5 本章のまとめ 155

参考文献 156
第 7 章　タイム・マネジメントによる設計開発期間の短縮 159
7.1　タイム・マネジメント 159
7.2　タイム・マネジメントに関する先行研究 160
7.2.1　製品の市場投入時期の重要性 160
7.2.2　CCM の事例研究 162
7.3　TOC とクリティカル・チェーン 163
7.3.1　TOC の出発点 163
7.3.2　プロジェクトにおける不確実性と人間行動特性による遅延 164
7.3.3　CCM 166
7.3.4　遅延の伝搬と総所要期間の最適化 167
7.3.5　リソース競合とクリティカル・チェーン 167
7.4　クリティカル・チェーンの適用手順 168
7.4.1　適用にあたって 168
7.4.2　プロジェクト・バッファ 168
7.4.3　合流(Feeding)・バッファ 170
7.4.4　リソース競合の回避 170
7.4.5　バッファ量のサイズ 171
7.5　設計知のアウトソース化 172
7.6　統合アプローチの提案と半導体設計開発プロジェクトへの適用 173
7.6.1　統合アプローチの適用にあたって 173
7.6.2　半導体設計開発プロジェクト 173
7.6.3　統合アプローチの適用 174
7.6.4　統合アプローチの適用結果 179
7.7　考察 179
7.7.1　アウトソース化と設計開発期間の短縮 179
7.7.2　CCM の課題 179
7.7.3　アウトソース先のタスク所要期間 181
7.8　本章のまとめ 181
参考文献 182
第 8 章　大規模システムの設計プロセスにおける 設計知の管理と活用 185
8.1　設計知の活用と設計負荷の低減 185
8.2　半導体設計手法の変遷 186
8.2.1　規模的拡大と設計の抽象性レベル 186
8.2.2　設計の抽象性レベルの概略 188
8.2.3　規模的拡大と自動設計 191

8.3 半導体 IP コアの体系化 192
8.3.1 継続する規模的拡大と半導体 IP コア 192
8.3.2 設計知の定義 192
8.3.3 半導体 IP コアの分類 193
8.3.4 半導体 IP コアの流通と普及 198
8.4 手戻り修正による設計負荷の問題と解決策 199
8.4.1 手戻り修正による設計負荷の問題 199
8.4.2 手戻り修正を引き起こす仕組み 200
8.4.3 設計負荷を減少させる一手法 203
8.4.4 従来手法との比較 204
8.5 半導体 IP コアの管理と活用 206
8.5.1 知識ベースによる設計過程の効率化とリスク低減 206
8.5.2 ソフト IP コアとハード IP コアが商品開発に及ぼす特性 206
8.5.3 ハード IP を多用したレイアウト設計手法による設計負荷の低減 207
8.5.4 今後の課題 208
8.6 本章のまとめ 208
注 209
参考文献 211
第 9 章 設計プロセスにおける設計知の多様化 213
9.1 半導体産業におけるビジネス・アーキテクチャの展開 213
9.2 先行研究と本研究の焦点 214
9.2.1 半導体産業における本研究の命題 214
9.2.2 ビジネス・モデル・イノベーションにおける本研究の位置付け 215
9.3 分析フレーム 217
9.4 ファブレス・モデル 218
9.4.1 分析フレームによるモデル化 218
9.4.2 日米のファブレス半導体市場の相違 219
9.4.3 日米のファブレス半導体ベンチャーの相違 220
9.4.4 日本におけるファブレス半導体ベンチャーの起業の遅れ 224
9.5 チップレス・モデルと実証 226
9.5.1 チップレス・モデル 226
9.5.2 日本の半導体メーカにおける設計知識の活用の問題 226
9.5.3 分析フレームによるモデル化と命題 227
9.5.4 検証方法と結果 228
9.6 本章のまとめ 232
注 233

参考文献 233
第 10 章　結論 236
10.1　本研究の焦点 236
10.2　本研究の結論 237
10.2.1　設計負荷の解明 237
10.2.2　設計プロセス管理の枠組みの提案 239
10.2.3　設計プロセス管理論の含意 241
10.3　今後の展望 244
謝辞 246

図表目次

図 1.1　設計プロセスの変換過程 2
図 1.2　設計プロセス管理の枠組み 6
図 2.1　機械設計のプロセス(Pahl & Beitz, 1977) 17
図 2.2　ソフトウェアの設計プロセス(菅野, 2003) 20
図 2.3　工業製品の性格(Ulrich & Eppinger, 2004:加筆) 21
図 2.4　設計プロセスと工業製品の性格(Ulrich & Eppinger, 2004:加筆) 22
図 2.5　半導体の設計プロセス(鈴木, 2003:加筆) 23
図 2.6　一般設計プロセス 27
図 2.7　ネットワーク図と DSM 図によるタスク A とタスク B の三つのモデル 33
図 2.8　タスク *i* とタスク *j* の依存性 34
図 3.1　設計プロセスの所要コスト、所要時間、製品性能の関係図 64
図 3.2　スケジュールの不確実性の因果連鎖ダイアグラム 65
図 3.3　想定内のイタレーションの因果連鎖ダイアグラム 67
図 3.4　想定外のイタレーションの因果連鎖ダイアグラム 70
図 3.5　イタレーションの範囲とイタレーションの所要時間の因果連鎖ダイアグラム 72
図 3.6　人間の行動特性による不確実性の因果連鎖ダイアグラム 74
図 3.7　設計知の不十分な活用とその他因子の因果連鎖ダイアグラム 76
図 3.8　スケジュールの不確実性の因果連鎖の総合 79
図 3.9　設計開発コストの不確実性の因果連鎖ダイアグラム 80
図 3.10　製品性能の不確実性の因果連鎖ダイアグラム 84
図 4.1　ネットワーク図 96
図 4.2　ネットワーク図と DSM 図によるタスク A とタスク B の三つのモデル 97
図 4.3　タスク *i* とタスク *j* と依存性 97
図 4.4　フィードフォワードとフィードバック 97
図 4.5　半導体設計工程 106
図 4.6　パーティショニングによるタスク・グループの抽出 107

図 4.7　抽出されたタスク・グループ.....107
図 4.8　依存性の度合いの相対的比率の違いが与える固有ベクトルへの影響.....108
図 4.9　固有値に対応した 1/(1-Re[λ_n]).....109
図 4.10　半導体 IP の置き換えによるシミュレーション.....114
図 5.1　リスク評価のフレームワーク.....127
図 5.2　イタレーションの DSM 表現（実設計事例）.....128
図 5.3　目標値の設定と目標値に収まる確率、リスク値.....134
図 5.4　活動時間と活動コストのリスク値結果.....136
図 6.1　逐次処理型とオーバーラップ型開発(Roemer et al., 2000).....143
図 6.2　半導体開発における時間とコストの無差別曲線.....144
図 6.3　設計開発工程の効率化の方向.....145
図 6.4　製品開発における価値形成.....150
図 6.5　価値のダイナミック・モデルと設計開発工程の改善.....151
図 6.6　無駄な工程の問題.....152
図 6.7　過剰機能・過剰品質の問題.....153
図 7.1　製品投入の前倒しと売上増加(Cook, 1998).....160
図 7.2　製品開発における優先事項と収益(Smith, 1988).....161
図 7.3　TOC の改善の 5 ステップ (稲垣, 1997).....162
図 7.4　スケジュールの不確実性.....163
図 7.5　クリティカル・パスとクリティカル・チェーン(Steyn, 2000).....166
図 7.6　クリティカル・チェーン適用手順(Cook, 1998; Yeo & Ning, 2002).....168
図 7.7　ガント・チャート.....175
図 7.8　ネットワーク図概要.....176
図 7.9　クリティカル・チェーン.....176
図 7.10　最終ガント・チャート.....177
図 7.11　タスク間の反復活動.....179
図 8.1　デジタル・システムの抽象性レベル(Armstrong, 1989）.....186
図 8.2　抽象性レベルの表示.....186
図 8.3　従来の一般的なレイアウト手法.....201
図 8.4　ハード IP の多用と自動レイアウト手法.....201
図 8.5　従来技術と自動配置技術によるレイアウト手法.....204
図 9.1　ファブレス・モデル.....218
図 9.2　チップレス・モデル.....225

表 4.1　固有ベクトルと総所要時間量のベクトル $\boldsymbol{U}$ 110
表 4.2　シミュレーション結果 116
表 5.1　代替案と 8 つの設計工程モデル 129
表 5.2　設計工程モデル毎の総ベクトル $\boldsymbol{U}$ 131
表 5.3　活動時間と活動コストの三角分布（実設計事例） 132
表 5.4　設計工程モデル毎のシミュレーション結果 133
表 7.1　本プロジェクトにおけるシステム LSI の仕様概略 173
表 8.1　各抽象性レベルの最小構成要素と機能的表現(Armstrong, 1989) 187
表 8.2　ソフト IP、ファーム IP、ハード IP の特徴（村田 et al., 2001:加筆） 193
表 9.1　製品開発手法での比較（小川, 2002:加筆） 215
表 9.2　日米ファブレス半導体の相違 220
表 9.3　被験者が属する企業:上位 4 社（企業名で属性分類） 228
表 9.4　記述統計量結果 228
表 9.5　ウイルコクスンの符号付順位検定 228
表 9.6　期待利得 PINT の 4 社間の漸近有意確率 229
表 9.7　期待利得 PSPE の 4 社間の漸近有意確率 229
表 9.8　期待利得 PNSP の 4 社間の漸近有意確率 229

第1章　　序論

1.1　　研究の目的

半導体やソフトウエアの製品開発において、ガント・チャートや PERT (Program Evaluation and Review Technique)、CPM (Critical Path Method)で代表される従来のプロジェクト・マネジメントの管理手法では解決困難な問題が散見される。それはすなわち、設計情報と設計知識の流れに起因した手戻り修正、無駄、時間とコストのリスクを引き起こす因子、不確実性要因、非効率な工程といった、設計開発工程に直接間接的に影響を及ぼす設計負荷の問題である。

設計学やプロジェクト・マネジメントの分野において、これまでこの問題を解決するための明確な解釈や知識の体系化並びに概念形成が行われてきたとは言えない。しかし近年、製品の設計開発工程が大規模・複雑化する状況下において、設計負荷の低減に向けた議論への要求が高まっている。設計負荷の問題は、設計学とプロジェクト・マネジメントにおける一つの中心的かつ現実的な問題となっている。

そこで本研究では、製品開発における設計負荷の問題の解決に向けた議論を行う。以下に示すように、設計負荷の問題は基礎的及び応用的問題として捉えることができる。設計負荷の基礎的及び応用的問題の両面を包含して、設計プロセス管理に関する一つの知見を形成することが本研究の目的である。

i)　基礎的な問題としては、「設計プロセス上の設計負荷」を解明することにある。設計プロセスは設計情報と設計知識の創造と処理を行う流れであるので、その観点で設計負荷の本質を探究する。

ii)　応用的な問題としては、「設計プロセスの効率化と改善」のための手法を確立することにある。これは基礎的問題解明に基づき、設計プロセスの管理技術を向上させることであり、経営的視点による効率化と改善の手段を提供することである。

1.2　研究の背景と対象

設計を理解することを目的とした学問は設計学である。設計学における設計は人が概念として想定した要求機能を、それを充足する実体へと変換する行為である（吉川, 1979)。この行為は人間の知的行為であるから、人間の知的行為に関する知識は設計学における研究対象である。ここで体系付けされている設計に関する知識は二つに分けることができる。一つは人間が設計する人工物、すなわち設計対象物に関する知識である。二つ目は設計をどの様に進めていくかという設計過程、すなわち設計プロセスに関する知識である。

設計対象物は、様々な分野で固有の設計対象が存在している。その対象分野には、機械、航空機、船舶、土木、建築、ソフトウエア、電子機器等が含まれる。それらは固有の知識を背景としているのであるから、それぞれ固有の設計対象物として具現化が行われる。しかし設計プロセスは、その設計対象物によらず共通していることが多い。その一つの理由として、それら設計プロセスは生産者とその人工物の顧客をつなぐ一つの共通した過程として考えられるからである。つまり、その過程において生産者は、まず顧客が想定する要求仕様を解釈し、要求概念を形成する。次にその要求概念から解概念に変換する。そして最後にその解概念を設計解として抽出する。この一連の変換過程が設計対象物によらず共通していることが一つの理由である。

この様に設計プロセスの変換過程は、要求仕様から設計解までの知的行為が関わっている。この知的行為はモノ的な動きは捨象されているので、設計情報と設計知識の流れと見ることが可能である。従って設計プロセスとは、設計情報と設計知識の創造・処理プロセスと言える。その設計プロセスを図 1.1 に示す。

設計プロセスの知識体系は、設計対象物である製品の知識体系に比べて多く議論されてこなかった。現在の工学領域における設計は、工業製品の物理現象に関する知識を体系化するという点に主眼が置かれてきたからである。しかしながら製品の設計開発は単に一般的な自然法則の知識を適時応用すれば良いということではない。設計対象物である製品や製品性能に関する知識に加え、製品の設計開発をどの様に進めるのかという設計プロセス

図 1.1　設計プロセスの変換過程

に関する人間の思考がなければ、製品を成果として創り上げることは困難である。つまり製品の設計開発においては人間の思考が関わるのであり、設計プロセスの知識体系についての議論が要求されるのである。

この要求に応える研究の一つは公理論的集合論に基づく一般設計学（吉川, 1979 & 1981; 冨山&吉川, 1985; 吉川&冨山, 2000）である。一般設計学は設計対象物ではなく、設計プロセスを一般化させた論理である。設計とは何か、設計技術向上の方法とは何か、これら基礎と応用の両面を包含した学問分野である。

発見的(heuristic)原理に基づく設計方法論も存在する。システム設計思考という設計の方法論を対象としているワーク・デザイン(Work Design) (Nadler, 1963; 吉谷, 1969; 高橋, 1993; 五百井 et al., 1997)がその一つである。設計活動が将来に対する適応努力とするなら、現存しない理想を追求しようとする機会があるはずである。この機会を捉えて理想的なシステムを設計しようとするシステム設計思考法は、分析的アプローチとは対極的な手法である。この思考法は実際に応用され、設計の方法論として効果的であることが知られている。

他に設計手法を管理的視点で捉えるアプローチもある。プロジェクト・マネジメントである。設計開発の工程管理という視点で捉えるなら、プロジェクト・マネジメント(PMI, 2004)で体系化されている知見を利用することもできる。

しかし冒頭で提起した現実的問題に答えられる知見はまだ決して多くはない。そこで本研究では、上述した設計学とプロジェクト・マネジメントを背景としてこの現実的問題を解決に向けた議論を行う。基礎的問題である「設計プロセス上の設計負荷」の解明は、設計プロセスを設計情報と設計知識の流れとする観点から行う。このことはすなわち、設計学における設計プロセスの知識体系の中での、設計負荷の概念形成につながる。応用的問題である「設計プロセスの効率化と改善」のためには、プロジェクト・マネジメントの管理手法を取り入れ、経営的視点に立った管理の枠組みを提供する。これはプロジェクト・マネジメントを一歩前進させることにつながる。

本研究では、設計プロセスを製品開発における設計開発工程の構造並びに過程と定義する。設計プロセスに関する議論の対象には、製品開発における設計工程計画、その管理、意思決定、設計負荷、設計情報と設計知識の創造・処理、固定観念にとらわれない柔軟性、プロセス・イノベーションが含まれる。これら対象は実証研究を通して新たな知見が求められていると言える。

1.3　研究の焦点と本論で取り組む課題

1.3.1　設計負荷とその低減

設計負荷は、適切な設計解を得ようとする知的行為に対して影響を及ぼす負荷現象である。設計情報と設計知識の流れに起因した手戻り修正、無駄、非効率な工程、あるいはその他の不確実性要因等、これらは設計開発工程に直接的あるいは間接的に影響を及ぼす設

計負荷であると考えられる。設計負荷の顕在化はスケジュールの遅延、コストの増加を引き起こすものでり、製品開発においては企業の経済的損失につながる重大な問題と認識されている。しかしながら、これまで設計負荷の体系化や概念形成は行われてこなかった。設計負荷の問題解決、また設計負荷の体系化や概念形成のためにも、設計負荷の解明は重要である。

設計負荷を低減させる手段として、設計負荷を直接的に和らげる対症療法では効果が薄い。設計負荷の原因が取り除かれない限り、設計負荷が繰り返されるからである。これに対し、もし設計プロセス上の設計負荷を顕在化させる因子の特定、並びにそれらの因果関係の解明が行われたならば、設計負荷の低減に向けた措置を講ずることは容易になる。これは原因療法的アプローチである。すなわち、設計負荷を顕在化させる因子の特定とその因果関係の解明が、設計負荷の低減につながる。

設計負荷の解明にあたっては、設計負荷を顕在化させる因子の特定とその因果関係の連鎖を先進法並びに後退法によって原因探索する。そして設計負荷を顕在化させるその因子に直接対処できる措置を講ずることで、設計負荷の低減をはかる。しかしここで設計負荷を顕在化させる因子が特定され、因果関係が解明されたとしても、設計負荷に影響を及ぼす因子は全て取り除けるとは限らない。その場合には、因子の影響を低減させる代替手段、あるいは、因果連鎖で介在する情報の流れの効率を高める代替手段などが考えられる。

本研究において提案される管理手法がこういった手段を持ち設計負荷を低減できるなら、設計プロセス上のスケジュール並びにコストは改善する。つまり設計プロセスが効率化あるいは改善されることにつながると言える。

1.3.2　製品価値と設計プロセス

企業は社会の公器という立場であるなら、その活動を通して株主や社会に受容される投資に見合う収益を上げなければならない。その企業がメーカであるなら、収益を向上させることは、製品の設計開発並びに販売活動によって顧客に価値を提供することである。従って、企業は次の様な問題意識を持っているのが一般的である。それは、製品の提供を通じて、どうやって顧客に価値創造を与えることができるのだろうか。そしてこの価値の継続をどうやって支えていくことができるのだろうかという意識である。現実的にはこれらに対する解は企業によって異なる。

本論ではこの製品価値を製品の設計プロセスと関連づけし、一般論として新たな知見を創出できると考えている。なぜならば、製品価値は製品仕様だけでなく、設計プロセス上のスケジュール並びにコストを通じて形成されるからである。実際のところ設計開発コストは製品価値に直接影響を及ぼしている。そして、適切な時期に製品を市場投入できるかどうかも製品価値と直接関連している。

1.3.3　スケジュール、コスト、製品仕様

製品設計における目標は、設計対象物となる製品並びに設計プロセスの二つの視点で捉えることができる。まず製品自身の目標は、予め設定した製品の属性、すなわち機能と性能と原価を包含する意味での製品仕様を達成させることである。そして設計プロセス上の目標は、設計開発において要するコストとスケジュールの効率を高めることにある。つまり、製品仕様、そしてコストとスケジュールは、製品開発における意思決定に重要な要素である。しかしこれらの三つの要素は互いに独立しているわけではない。むしろそれぞれが相互に依存関係を持っている。例えば現実の世界では、この三つの要素の内、一つ又は二つは目標を達成できたとしても、残りは達成できないことがある。すなわち、三つの要素の目標を同時に達成することは、困難な取り組みである。

従って、製品仕様を維持することを前提にしながら、設計開発におけるコストとスケジュールを統合管理することが求められる。ここでこういった現実的状況を考慮した設計プロセスの統合管理の枠組みが提供されるならば、製品開発にとって効果的である。

1.3.4　本研究における設計プロセス管理の概要

設計は思考作業である。その手順は、Dixon (1966)によれば、i)ニーズの認識、ii)問題の定義、iii)総合(synthesis)、iv)解析(analysis)、v)最適化、vi)評価、vii)決定である。要求仕様から出発する設計プロセスでは、この一連の手順に基づいて、最終的に設計解が導出される。この一連の手順ではモノ的な流れは生じない。この手順の中で流れるのは、人間が介在する設計情報と設計知識である。そこで本研究では、設計情報と設計知識の創造と処理システムの観点から、設計プロセスを管理する枠組みを提示する。図 1.2 にその枠組みを示した。

本研究で提示する設計プロセス管理の枠組みは、計画(plan)、実行(do)、評価(check)、改善(act)の四つの段階によるサイクルである。各段階におけるキーワードと、そのキーワードに関連する章を枠内に示した。この管理の枠組みの中に、設計負荷を含む設計プロセスの挙動を掌握し、そして設計負荷を低減する方式を組み込んでいる。

計画の段階では目標を設定し、その目標を実現するための設計プロセス案を策定する。次の実行段階では計画を実行する。そして評価段階においては、計画の達成度合いを測定・評価し、結果を目標と比較する等分析を行う。最後の改善段階では、設計プロセスの改善に必要となる変更点を明らかにし、次の計画に結び付ける。四つの段階を順に実施するこのサイクルによって、設計品質を維持・向上し、継続的に設計業務改善活動を推進することができる。そして設計の生産性向上にも寄与できる。この様な管理サイクルは、品質向上を目指す生産プロセス (Shewhart, 1951; Deming, 1960)、トヨタ生産システム(大野, 1978; Womack et al., 1991)、そして制約条件の理論(Goldratt,1986;1997)では既に定着している手法である。つまり絶え間ない改善による、品質と生産性の向上である。本研究における設計プロセス管理でも、この一連のサイクルを適用している。

本研究論文のねらいは、製品の設計プロセスにおいて、そのプロジェクトに関わる管理者の意思決定を側面的に支援することである。この意思決定は顧客視点と企業視点の双方を念頭に入れたもの造りの領域である。

尚、本論で提示する設計プロセス管理の手法は従来のプロジェクト・マネジメントによる管理手法を否定するものではない。むしろ本論で議論した知見を従来の管理手法に反映させ拡張したものである。

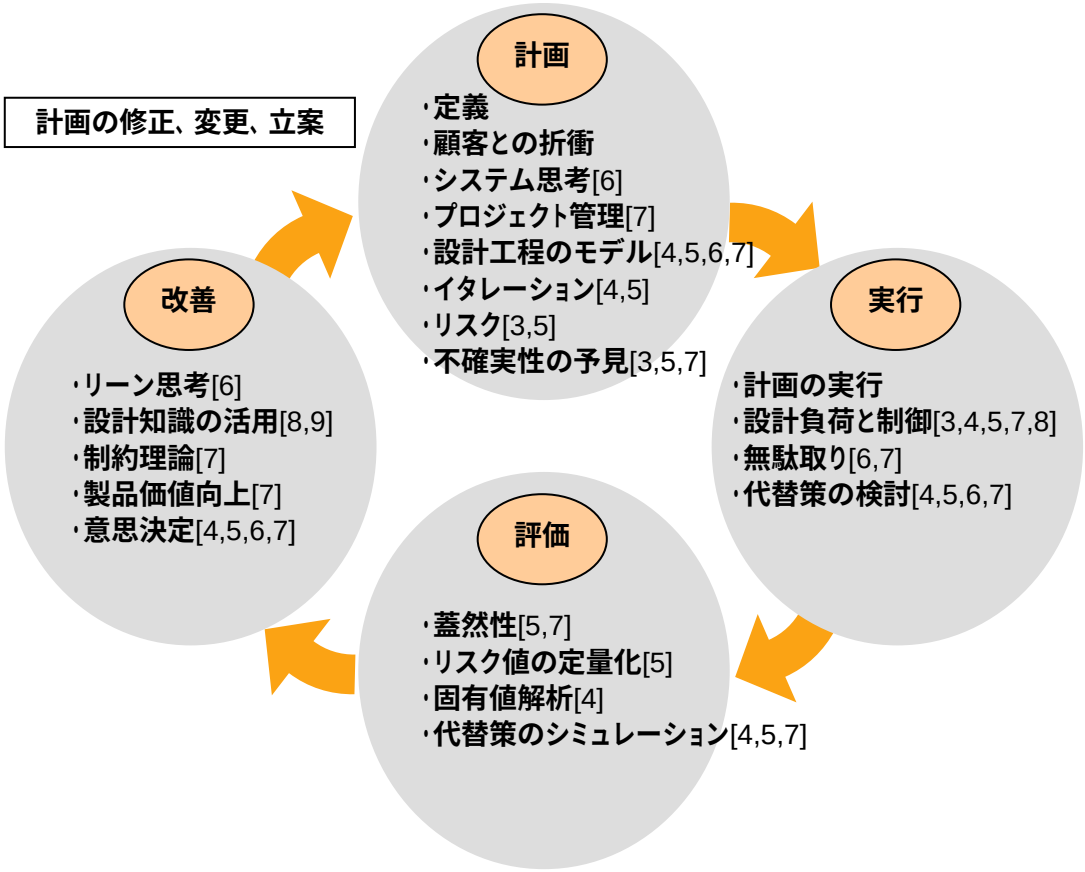


図 1.2　設計プロセス管理の枠組み

1.3.5　本論で取り組む課題

次に本論で取り組む課題を以下に示す。

1)　イタレーションの数理モデル化

イタレーション(iteration)は設計負荷の主たる要因の一つである。イタレーションとは、プロジェクト内のタスク(task)間の情報のやり取り、手戻り修正作業である。既存のプロジェクト・マネジメントの管理手法では、イタレーションを伴う設計開発工程を想定していないため、そのモデル化や解析は困難である。そこで本研究ではイタレーションの現象を数理モデルとして表現する。

2)　プロジェクト・リスクの定量的評価フレームの構築

プロジェクト・リスクに対処するためには、リスクの先読みと見積もり、評価を行うリスク・アセスメントが有効である。現状、プロジェクトの意思決定は定性的リスク評価に基づいて行われる場合が多い。しかし、より確実で精度の高いリスク・アセスメントを実践するためには、リスクを定量的に評価するためのフレームの構築が求められる。

3)　二律背反のメカニズムの解明

企業は計画する製品性能を犠牲にせずに、製品開発における所要時間と所要コストの効率化を達成しようとする。しかしながら設計プロセスの時間短縮とコスト低減は二律背反の問題である。設計工程の効率化の真意について考察するために、先行研究の論文を通じて、二律背反のメカニズムを明らかにする。

4)　リーン思考を拠り所にした価値モデルの構築

所要時間と所要コストのトレードオフ検討による設計工程の効率化には限界がある。所要時間と所要コストに別の基軸を追加することが、効率化の最適解を得るための有効なアプローチと考えられる。

リーン思考はトヨタの生産方式として知られている原則である。リーン思考は価値を生まない活動を排除し、価値を最大化させる活動を促進する。本研究では、時間短縮とコスト低減を同時に実現させ、最終的に製品価値を向上させた事例として、リーン思考による設計工程の効率化を考察する。そしてリーン思考を拠り所として、設計開発の所要時間と所要コスト、そして製品価値を基軸とする設計プロセスのモデルを構築する。

5)　システム思考による総合的アプローチ

設計プロセスは図 1.1 を参照すると、要求仕様を入力(input)、設計解を出力(output)と見なすことができる。従って設計プロセスはシステムである。設計プロセスの効率化のためにはシステム思考が求められる。

設計プロセスの効率化は、直接的には製品性能を犠牲にせずに、所要コストと所要時間を同時に改善させることである。このためには、これら因子を分析し、問題点を探索し、改善策を見出す方法が用いられる。これは分析的(analysis)アプローチである。しかし設計活動が将来に対する適応努力であるとすれば、今存在していない設計プロセスを創造しようとする機会が増えるはずである。このためには、システム思考によって理想的な設計プロセスの創造を追求する方法、すなわち総合的(synthesis)アプローチが求められる。

6)　制約理論のタイム・マネジメントへの応用

本研究では、工学的な視点に立ったタイム・マネジメントを提案する。これは企業戦略におけるタイム・マネジメントが、個人・組織の能力向上または意識改革を前提としている点で異なる。Goldratt の制約理論(Theory of Constraint:TOC)を拡張したクリティカル・チェーン管理(Critical Chain Management:CCM)の手法を応用する。

7)　知識ベースの構築に向けた議論

設計知識と設計情報が知識ベースとして定着し、組織内や企業間で共有化が行われると、知識の再利用が行われる。もし設計知識の再利用が進むならば、設計開発コストの低減と設計開発時間の短縮につながる。設計知識を体系化し活用する知識ベースは、設計活動の効率向上に寄与すると言える。本研究では、半導体設計開発における設計情報と設計知識の体系化について議論する。

8)　設計知識の多様化に向けた議論

設計知識を外部から導入することは、設計知識の多様化につながる。しかし、伝統的な垂直統合型企業が設計知識を外部から積極的に導入することは少ない。設計、生産そして消費サイクルを企業グループ内へ依存しているからである。そこで本研究では、プロダクト・イノベーションを創出する設計知識を外部から導入すると想定した時の、企業の収益追求への期待感を調査する。そしてその分析結果から、設計知識の多様化への可能性を議論する。

1.4　研究の方法と経緯

1.4.1　研究の方法

設計プロセス管理という知見を形成するには、そこで用いられる方法について厳格な検討が必要となる。つまり、設計プロセス管理を学問として取り上げるためには、設計プロセス管理並びにその効率化と改善の手法についての主張あるいは仮説が、検証可能性あるいは反証可能性を備えていることが必要条件である。本研究ではその検証を帰納法と演繹法によって行う。

帰納法としてはまず、設計技術者が設計しそしてプロジェクト・マネージャが管理する過程を観察する。そして次に設計プロセスの現象並びにその管理手法の仕組みを解明する。演繹法としてはまず、設計プロセスに関して十分と認められる問題点の性質を仮説として設定する。次にそこから演繹によって設計プロセス管理の手法を導出する。そして実設計プロジェクトに適用することでその有効性を検証する。

1.4.2　研究の経緯

本研究の経緯について述べる。本研究論文は四つの期間に渡る研究作業を集約したものである。その四つの期間には合計約 3 年を要した。

第一期:

まず第一期は、実地調査である。株式会社 Paltek (JASDAQ: 7587)の完全子会社である株式会社スピナカー・システムズと大手電機メーカとの共同プロジェクトを中心にデータ収集を行った。インターネットのブロードバンド化を実現させるスイッチ LSI が最終製品であった。従来の設計開発手法の問題を導出する為に、2002 年秋頃からプロジェクトに関わった設計開発技術者と面談を行っている。そしてイタレーションによる設計負荷を中心に研究を行った。本領域の研究成果は、第 4 章としてまとめられている。

第一期にはもう一つの半導体設計プロジェクトの実地調査を行っている。従来の設計開発手法における問題を導出するために、2003 年の年末から米国のファブレス半導体メーカである NextSierra 社からデータ収集を行い、設計開発技術者との面談も行った。プロジェクトの最終製品は、携帯電話にサブパネルに搭載されるディスプレイ用のシステム LSI である。このプロジェクトを対象に、プロジェクトの不確実性やリスクの問題設定を行った。そして先行研究の調査を含め、リスク・アセスメントと意思決定を中心に議論を展開した。ここから定量的リスク評価フレームの構築を行っている。本領域の研究成果は、第 5 章としてまとめられている。

上記の研究成果を二つの論文にまとめるにあたり、設計プロセスのモデル化に関する文献を渉猟した。DSM (Design Structure Matrix:設計構造マトリックス) (Steward, 1981a & 1981b)

の先行研究の調査から得られた知見は、第 4 章と第 5 章に反映されている。DSM によって設計プロセス上の反復修正の箇所を特定し、定量的なデータからモデル化させた。設計プロセスを一つのシステムと見なし、一対比較行列(pairwise comparison matrix)を用いてモデル化を実現した。そして手戻り修正の収束性、すなわちシステムの安定性を固有値問題として捉え、解析するアプローチを第 4 章にまとめた。DSM を発展させて確率モデルとして構築し、プロジェクトの所要コストと所要期間のリスク値を導出した。そして、予め準備しておいた多様な設計モデルをシミュレーションすることで、最適解を発見するアプローチを第 5 章にまとめた。

第二期:

第二期は、設計プロセスにおける設計知識の体系化から出発した。次にその設計知識の有効活用が、設計プロセスの効率化につながることを究明した。本領域の研究成果は、第 8 章としてまとめている。更に、プロダクト・イノベーションに向けた知識の多様性と企業の期待利得との関係性に着目した。そこから導出した命題は、プロダクト・イノベーションを目的として設計知識を多様化させることが、もの作りの期待利得につながるのではないかというものである。本命題の検証にあたっては、日本の電機メーカの半導体事業部で働く中間管理職にアンケート協力を要請した。この設計プロセスにおける設計知識に関する研究は、2003 年末から 2004 年末までにかけて実施された。本領域の研究成果は、第 9 章としてまとめられている。

第三期:

第三期における実地調査は、株式会社タイテック(JASDAQ: 6893)の完全子会社である株式会社グラフインとの共同プロジェクトを対象とした。プロジェクトは、コンピュータの拡張スロットのインタフェースをつかさどるシステム LSI の設計開発である。従来の設計開発手法におけるタイム・マネジメントの問題を導出するために、NextSierra 社とグラフイン社双方の設計開発技術者と面談を行い、そしてデータ収集を行った。本調査は 2004 年の夏頃から実施された。本領域の研究成果は、第 7 章としてまとめられている。第 7 章で提案したタイム・マネジメントの管理手法は、スケジュール管理を制約理論で拡張し、そしてアウトソース化する統合アプローチである。人間行動の特性を考慮し、無駄な時間をなくすことをねらいとした手法である。

第四期:

第四期は、設計プロセスの時間短縮とコスト低減の問題に関する論文の渉猟から始めた。設計プロセスの時間短縮とコスト低減は二律背反の問題であり、同時に改善することは困難であるというのが通説である。しかしながらトヨタ生産方式で取り上げられているリー

ン思考による事例はその論調を覆す例証であった。その事例を通じて設計プロセスの効率化の真意について研究を重ねた。そこでの考察からリーン思考を拠り所とした価値のダイナミック・モデルを導出した。本モデルによって製品価値並びに効率化の形成過程を明らかにした。本モデルは設計プロセスおける所要時間・所要コスト・製品価値という指標により実践的洞察力を高め、設計プロセスの効率化に向けた指針と運用を与えるものである。本領域の研究成果は、第 6 章としてまとめられている。設計プロセスの時間短縮とコスト低減に関する二律背反の問題と価値のダイナミック・モデルに関する研究は、2004 年秋から 2005 年夏にかけて実施された。

1.5　本論文の構成

本論文は全 10 章から成り、その概要は以下の通りである。

第 1 章　序論

研究の目的、背景、方法、経緯について述べる。また、研究の焦点となる設計プロセス管理の概要と、本論で取り組む課題を提示する。

第 2 章　設計プロセスにおける問題と先行研究

第 2 章では、設計プロセスと設計負荷に関する先行研究を概観する。そこでのイタレーションについての議論から、イタレーションの本質を探究する。また、設計プロセス内の構成要素のつながりを説明するために、設計情報と設計知識の創造・処理システムの観点から製品開発工程を表現し問題点を整理する。

第 3 章　設計プロセスにおける状況認識と不確実性の源泉

第 3 章では、設計プロセスにおけるリスクの源泉について議論する。スケジュール、設計開発コスト、製品性能は設計プロセスに直接影響を及ぼすリスクである。これらのリスクの源泉を、因果連鎖ダイアグラムを活用して探求する。そしてリスクを引き起こす不確実性の因子を特定し、これら因子が顕在化することで設計負荷が生じる構造を明らかにする。設計負荷は、イタレーション、人間の行動特性、設計知（「設計情報と設計知識」）の不十分な活用を包含する。

第 4 章　設計プロセスにおけるイタレーションのモデル理論と実践

第 4 章では、設計の手戻り修正を含めたイタレーション問題をとりあげる。イタレーションのメカニズムの解明を行うため、イタレーションを含む設計工程の数理モデル化につ

いて議論する。従来のモデル化手法の問題点と DSM(Design Structure Matrix:設計構造マトリックス)の有用性を示し、DSM を応用した一対比較行列(Work Transformation Matrix) によるモデル化の手順を提示する。そして、このモデルにより設計負荷の定量的解析が可能であることの理論を示し、半導体の実設計工程に適用してそれを実践する。

第 5 章　設計プロセスにおけるイタレーションとリスク・アセスメント

第 5 章では、設計工程の管理と意思決定に向けた一つのフレームを提案する。本フレームは代替設計工程案、開発時間、活動コスト、不確実性要因、リスクに対する実践的な洞察を提供するものであり、更に設計工程の策定と変更・修正といった意思決定を側面から支援するものである。本フレームではイタレーションを含む設計工程から定量的リスク・アセスメント・モデルを構築する。多様な実設計工程モデルを利用したシミュレーションにより、本フレームの有効性を実証する。

第 6 章　無駄な工程・過剰機能・過剰品質と価値のダイナミック・モデル

第 6 章では、設計のジレンマ、すなわち設計工程の時間短縮とコスト負荷の二律背反問題を取り上げる。リーン思考により時間短縮とコスト負荷削減を同時に実現させた事例を通じて、設計工程の効率化、過剰機能・過剰品質の真意について考察する。そして製品開発における製品価値の形成過程と設計工程の効率化の関係を解明するために、設計工程におけるダイナミック（動的）な価値モデルを提示する。

第 7 章　タイム・マネジメントによる設計開発期間の短縮

第 7 章では、掛け持ち業務という制約条件のもとに設計開発する状況を取り上げる。短納期化のアプローチとして制約理論の考え方をタイム・マネジメントに応用する。事例の分析を通して本アプローチの実際と有効性を検証する。

第 8 章　大規模システムの設計プロセスにおける設計知の管理と活用

第 8 章では、付加価値の源泉となりつつある設計知（「設計情報と設計知識」）のモジュール、すなわち半導体 IP (Intellectual Property) の概念を体系化すると同時に設計プロセスにおける位置付けを明らかにしていく。そしてナノテク時代の半導体設計開発の技術的問題点を指摘し、半導体の IP を利用したシステム LSI の事例で検証する。

第 9 章　設計プロセスにおける設計知の多様化

第 9 章では、設計知（「設計情報と設計知識」）の多様化とプロセス・イノベーションに関する問題を取り上げる。それは外部からの知識と情報を新しい設計知として融合さ

せ、その設計知を機動的に知識創造の共同体としてどの様に造り、どう収益追求をしていくのかという問題である。この問題解決に向けて一つのコラボレーションの命題を設定し、経験的データにより検証する。

第 10 章　結論

本論文における結論を述べ、本研究における今後の課題をあげる。

参考文献

[1] Blanchard, B. S. and Wolter, J. F.: “Systems Engineering and Analysis, Second Edition Englewood Cliffs, NJ: Prentice Hall (1990)

[2] Boehm, B. W.: “Software Engineering Economics”, Prentice Hall, Englewood Cliffs, NJ (1981)

[3] Deming, W. E.（坂元平八監訳）: “品質管理の基礎概念:品質管理の観点からみた統計的方法”,岩波書店 (1960)

[4] Dixon, J. R.: “Design Engineering”, McGraw Hill (1966)

[5] Goldratt, Eliyahu M.: “The Goal: A Process of Ongoing Improvement”, The North River Press (1986) (エリヤフ・ゴールドラット (三本木亮訳): “ザ・ゴール ― 企業の究極の目的とは何か”,ダイヤモンド社 (2001))

[6] Goldratt, Eliyahu M.,: “Critical Chain”, The North River Press (1997) (エリヤフ・ゴールドラット (三本木亮訳): “クリティカル・チェーン”,ダイヤモンド社 (2003))

[7] 五百井清右衛門,黒須誠治,平野雅章: “システム思考とシステム技術”,白桃書房 (1997)

[8] Nadler, Gerald: “Work Design” (1963)（ジェラルド・ナドラー（村松林太郎,黒田充,門田武治,鈴木成裕,三原田栄共訳): “ワーク・デザイン”,建帛社 （1965））

[9] 大野耐一: “トヨタ生産方式－脱規模の経営をめざして”,ダイヤモンド社 (1978)

[10] PMI（PMI 東京支部監訳）: “プロジェクトマネジメント基礎知識体系ガイド 第3版”,PMBOK ガイド (2004)

[11] Putnum, L. H. and Fitzsimmons, A.: “Estimating software costs. Datamation”, October, pp. 171-177 (1979)

[12] Reinertsen, D. G.: “Whodunit? The search for the new-product killers.”, McKinsey and Company Report, pp. 35-37 (1983)

[13] Shewhart, W. A.（白崎文雄訳）: “工業製品の経済的品質管理”,日本規格協会 (1951)

[14] Shishko, Robert et al.: “NASA Systems Engineering Handbook”: NASA (1995)

[15] Smith, Preston G.: “Winning the new products rat race”, Machine Design, May 12, pp. 95-98 (1988)

[16] Stalk, Jr. G.: “Time: The Next Souce of Competitive Advantage”, Harvard Business Review,

July-August (1988)

[17] Stalk, Jr. G., and Hout, T. M.,: “Competing against Time:How Time Based Competition is Reshaping Global Markets”, The Free Press, New York (1990)

[18] Steward, Donald V.: “The Design Structure System: A Method for Managing the Design of Complex Systems”, IEEE Transactions on Engineering Management, 28(3), pp. 71-74 (1981a)

[19] Steward, Donald V.: “Systems Analysis and Management: Structure, Strategy, and Design”, New York, PBI (1981b)

[20] Susman, G. I. E.: “Integrating Design and Manufacturing for Competitive Advantage”, New York, Oxford University Press (1992)

[21] 高橋輝男: “システム設計思考法”,白桃書房 (1993)

[22] 冨山哲男,吉川弘之:“一般設計学の展開（第一報）:概念空間のコンパクト化”,精密機械, Vol. 51, No. 4, pp. 809-815 (1985)

[23] Vose, David（長谷川専,堤盛人訳）: “入門リスク分析”,勁草書房 (2003)

[24] Wiest, J. D. and Ferdinand K. L.: “A Management Guide to PERT/ CPM: With GERT/ PDM/ DCPM/ and Other Networks, Second Edition, Englewood Cliffs, NJ: Prentice-Hall (1977)

[25] Williams, J. R.: “How Sustainable is Your Competitive Advantage?”, California Management Review 43, 3 (1992)

[26] Womack, J. P., Roos, D. and Jones, D. T.: “The Machine that Changed the World”, Rawson Associates, New York (1991) (J.・ウォマック,D.・ルース,D.・ジョーンズ（沢田博訳）: “リーン生産方式が、世界の自動車産業をこう変える”,経済界 (1991))

[27] 吉川弘之,冨山哲男: “設計学:ものづくりの理論”,放送大学教育振興会 (2000)

[28] 吉川弘之:“一般設計学序説:一般設計学のための公理法”,精密機械, Vol. 45, No. 8, pp. 906-912 (1979)

[29] 吉川弘之:“一般設計過程”,精密機械, Vol. 47, No. 4, pp. 405-410 (1981)

[30] 吉谷龍一: “システム設計”,日経文庫 (1969)

第2章　　設計プロセスにおける問題と先行研究

2.1　　設計負荷の問題と本章のねらい

半導体やソフトウエアの製品開発において、従来のプロジェクト・マネジメントの管理手法では解決困難な問題が散見される。設計プロセスが大規模・複雑化している状況下で直面している問題であり、それはすなわち設計プロセス上の設計負荷の問題である。設計負荷の主たる要因は、設計工程内のタスク間の反復作業、すなわちイタレーション(iteration)、そして人間の行動特性、知識活用の不十分性である。本章では、設計プロセスと設計負荷に関する先行研究を通じて議論する。また、そこでのイタレーションについての議論から、イタレーションの本質を探究する。

本章ではまず、設計プロセスに関する先行研究を概観する。そこでは幾つかの工業製品を取り上げ、それぞれの分野で体系付けられた設計プロセスの流れについて議論する。そして、それらから設計プロセスの共通の手順を見出し、工業製品一般の設計プロセスの概念を構築する。

次に、設計負荷に関する先行研究を概観する。ここではまず、イタレーションを探求する為に、イタレーションについての先行研究論文を渉猟した。イタレーションに対する管理手法とその限界点、イタレーションの発生と含意、イタレーションの分類と効率化、イタレーション・モデルとその有効性、以上の観点から議論を整理する。そして続いて人間の行動特性、知識活用の不十分性の概要を示す。

本章の最後では、PDCA サイクルに関する先行研究とその概要を示し、設計負荷に対処する為の管理サイクルの必然性について述べる。

2.2　　設計プロセスの概念

設計は「要求を実現する人工物に関する情報を段階的に詳細化しながら決定していく過程」である(吉川&冨山, 2000)。人工物に関する情報には、属性、状態、挙動、機能が含まれる。これらの情報によって、人工物である設計対象物を表現したものがモデルである。こ

のモデルは設計プロセスにおいて、設計に関わる情報を作成・伝達する役割を担うものである。従って、設計プロセスは設計対象物やそのモデルと共通した概念を用いて議論することがある。

本節では設計プロセスの概念を議論する前に、ここで設計対象物を表現する要求、仕様、属性、状態、挙動、機能といった基本的な概念(吉川&冨山, 2000)について整理をしておく。

・要求：　これから設計する人工物に要求される属性、挙動、機能などの情報である。
・仕様：　人工物を記述するために必要な属性、挙動、機能などに関する情報である。
・属性：　人工物が持つ幾何的、物理的、化学的、機械的性質のことであり、属性は値を持っている。
・状態：　人工物がある場合、ある時刻において、その属性が持つ値の組のことである。
・挙動：　人工物の状態の時間的変化・遷移を意味する。
・機能：　挙動を人間が特定の意図を持って主観的に観察するときに発揮していると認められる人工物のはたらきである。

次項以降では幾つかの工業製品を取り上げ、それぞれの設計プロセスにおける作業の流れを述べる。この事例は、機械、ソフトウエア、電気電子機器、半導体である。そこから工業製品一般の設計プロセスの概念を検討する。

2.2.1　機械設計

機械工学の分野ではどの機械にも共通した機械要素の知識が体系付けされ、そして数多くの設計理論が提唱されてきた。またその歴史も長い。宇宙航空、電気電子、ソフトウエア、半導体といった分野の工業製品よりも発祥が古いからである。

実際の所、機械設計における知識の体系化は過去、設計技術の修得や継承に効果をもたらしてきた。そして設計学や設計工学という学問形成に貢献してきた。その設計学そして設計工学の中で位置付けられている機械工学という領域において、Pahl & Beitz (1977)は設計プロセスの全体モデルを述べている。その全体モデルは四つの段階に分けられている。それらは、i)役割の明確化、ii)概念設計(conceptual design)、iii)実体設計(embodiment design)、iv)詳細設計(detail design)である。図 2.1 にその設計プロセスを示した。この四つの段階をそれぞれ以下の様に定義付けしている。

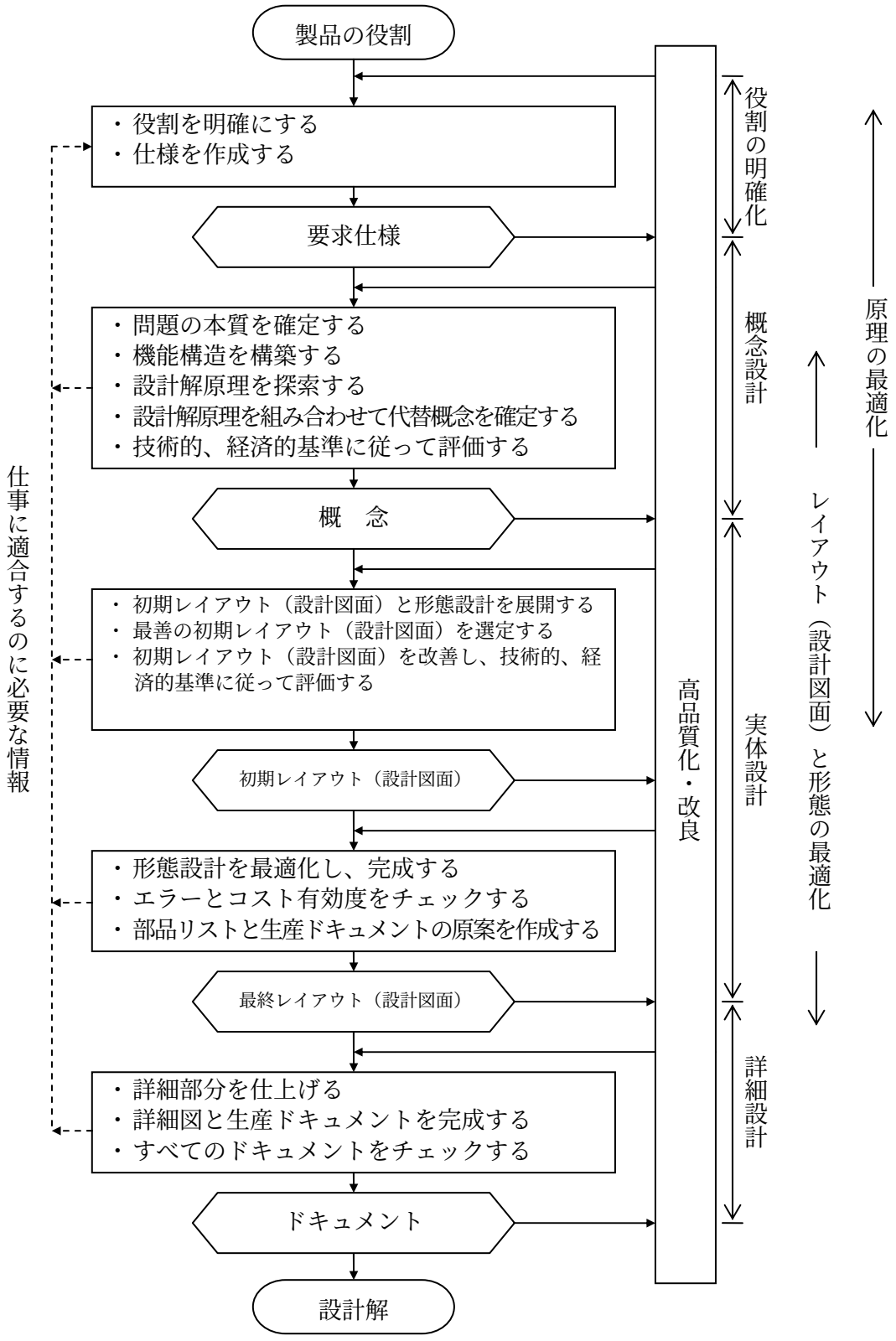


図 2.1　機械設計のプロセス(Pahl & Beitz, 1977)

i)　役割の明確化

この段階では、まず設計解を実体化するのに必要な要件と制約条件に関する情報を収集する。そして詳細な仕様書あるいは要件のリストを作成し、これを精密化する。

ii)　概念設計

この段階では、まず機能構造を構築し、適切な設計解原理(solution principle)とそれらの組合せを探索して代替概念を見つけだす。そして要求仕様を満足していない代替概念を取り除く。次に残った案の中から仕様で定めた要望に基づいた評価基準を体系的に当てはめることで最良の設計解概念を選び出す。概念設計はこの一連の過程である。

iii)　実体設計

この段階では設計者は概念を基にして最終のレイアウト(layout)を決定し、技術的そして経済的事項を考慮に入れて製品やシステムを開発する。尚、機械設計におけるこのレイアウトは設計図面を指す。次に幾つかのレイアウト図を作成し、同時並行的にあるいは逐次的に寸法の割り振りを行う。そして多様な代替案の中で利点、欠点に関する情報を入手し、これを基に技術的かつ経済的な評価を行う。最終的に決定したレイアウトを機能、強度、空間上の適合性などの観点から確認する。この段階でプロジェクトの経済的な実現可能性を最終的に査定する。

機械設計では、特に実体設計の基本ルールと原理、ガイドラインが予め用意されている。機械設計の基本ルールとは明確性(clarity)、単純性(simplicity)、安全性(safety)である。機械設計の原理としては、力の伝達、役割分割、自己充足、安定性と意図的不安定に関するものである。ガイドラインは、熱膨張・許容・耐腐食損傷・標準規格を考慮した設計、そして生産・組立の容易化設計、設計欠陥・外乱因子・リスクに対処する設計である。

iv)　詳細設計

この段階ではまず個々の部品全ての配置、形態、寸法及び表面特性を決定し、材料を指定する。そして技術的並びに経済的な実現性を再度確認し、全ての図面や生産に関する文書を策定する。詳細設計が設計プロセス上の最終段階である。

2.2.2 ソフトウエア

ソフトウエア開発の設計プロセスのモデルは幾つか提唱されている。一般に良く用いられているプロセスは、ウォーターフォール(waterfall)型モデルである（菅野, 2003）。このモデルは開発の各フェーズ、すなわち分析・設計・実装・テスト・保守を順番に進めるものである。しかしこのモデルではその利用に限界が指摘されている(Bohem, 1981)。それはその理論と実践の差異である。従って、その欠点を補うために修正が加えられている。それらは反復(iteration)型(Basili & Turner, 1975)、プロトタイピング(prototyping)型(Gomma & Scott, 1981)、スパイラル(spiral)型(Boehm, 1988)である。

ウォーターフォール型は作業が一方向だけに進展するが、反復型は逆方向にさかのぼることが可能である。この反復型開発は、複数回の反復を費やして問題自身を理解する。そして効果的な解決策と計画を洗練して、利害関係者の目標をバランス良く取り扱うことである。プロトタイピング型は要求仕様が固定しにくい、正確に表現できない、変化するといった原因に柔軟に対処する方法である。スパイラル型は、反復型とプロトタイピング型の統合が基調になっている。

しかしながらこれら反復型、プロトタイピング型、スパイラル型は、ウォーターフォール型の派生であるので、一連のプロセス構造は共通している。そこで本項ではソフトウエア開発の代表例としてウォーターフォール型の設計プロセスを例示する。その設計プロセスを図 2.2 に示した。

まず図 2.2 のシステムはハードウエアとソフトウエア全体を表現している。そしてそのシステム方式設計とは概念設計である。この場合、ソフトウエア設計はハードウエアに依存した概念設計が要求される。次にソフトウエア上のアルゴリズム(algorithm)と構築計画を決定する段階があり、これが実体設計段階に相当する。最後に具体的なプログラミング(programming)作業、すなわち詳細設計に移行する。

この設計プロセスは、前述の機械一般の設計と比して異なる点は見られるものの、基本的な過程は同じである。

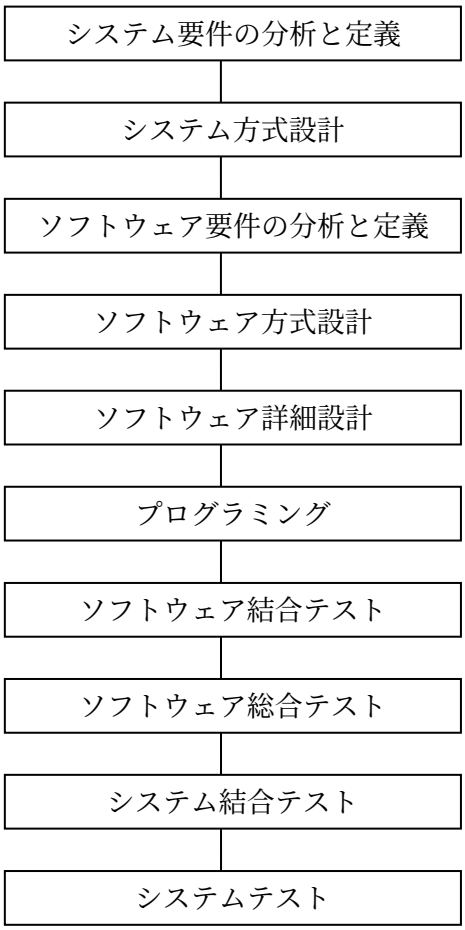


図 2.2　ソフトウェアの設計プロセス(菅野, 2003)

2.2.3　電気電子機器

電気電子機器の設計過程も、機械設計とそれほど異なるわけではない。実際、工業製品の多くは設計プロセス全体の流れはほぼ共通している(Ulrich & Eppinger, 2004)。その理由は以下の通りである。電気電子機器は電気電子回路の設計過程によって人工物へ具現化されるものと考え、三つの段階の工程に分けることができる。電気電子回路ではまず要求された機能を満足する回路の基本構造を決定する工程がある。これを概念設計とする。次に機能や性能などを検討し、回路に用いられる素子や部品を決定する工程を実体設計とする。そして最後にここまでの情報をもとにノイズや発熱を考慮しながら具体的な回路のパターンを設計する工程が存在する。この段階を詳細設計と見ることが可能である。

電気電子の工業製品の多くは共通した設計プロセスを持っているが、Ulrich & Eppinger (2004)は、設計する工業製品の性格によっては設計軸の相対的期間が異なることを指摘している。その性格とは、技術主導製品(Technology-driven products)と顧客主導製品(User-driven products)による相違である。

技術主導製品の主たる特徴は、生産者である自社の技術を核としている点、あるいは特定技術のタスクを実行できる能力を提供している点である。この製品は審美的又は人間工学を考慮している場合もある。付け加えて、技術的性能を評価基準の優先事項として購入する消費者が多い。つまり製品の表層的要因が重要な意思決定に関わっている。

顧客主導製品は、生産者と消費者の接点あるいは情報交換、又は消費者の意識化された要望から創り上げられるものである。製品の差別化要因として審美的であること、消費者の自尊心を傷つけないことが重要事項の一つである。ただし製品の技術は既に確立されていることが多い。生産者は、消費者視点に立った製品創りが求められることが一般的である。

図 2.3 は、電気電子機器の幾つかを、技術主導製品と顧客主導製品という性格の違いによって分類した図である。それぞれの製品はどちらか両極端の性格になるとは考えられない

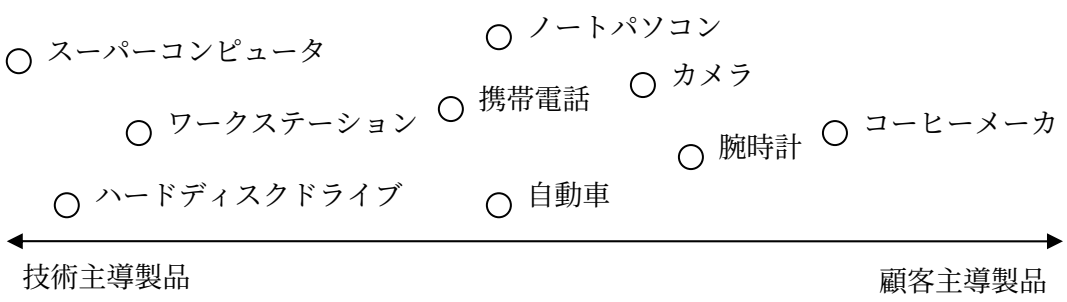


図 2.3　工業製品の性格(Ulrich & Eppinger, 2004:加筆)

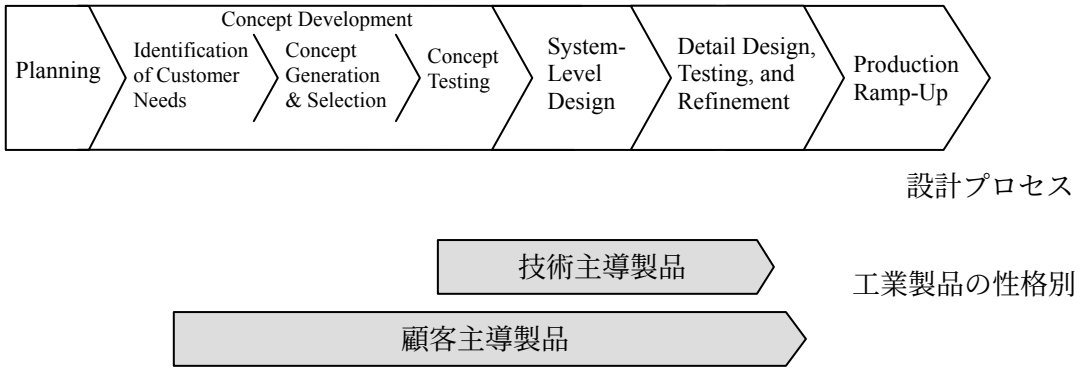


図 2.4　設計プロセスと工業製品の性格(Ulrich & Eppinger, 2004:加筆)

ので、連続的な直線上に表示されている。

ここで技術主導製品と顧客主導製品の設計プロセス上の相違を見てみる。技術主導製品の場合には、設計プロセスの初期過程を経ることが少ない。むしろ設計プロセスの中後期から開始される場合が多い。一方、顧客主導製品は設計プロセス全体に渡って形成されることが多い。図 2.4 に Ulrich & Eppinger (2004)による設計プロセスと、技術主導製品と顧客主導製品それぞれの対応期間を示した。

尚、図 2.4 の Planning から Detail Design までが設計プロセスであり、Production Ramp-Up 以降は生産プロセスの段階である。Ulrich & Eppinger (2004)が分類しているこの設計プロセスも、機械設計で規定している一連の設計過程、すなわち役割の明確化、概念設計、実体設計、詳細設計とほぼ類似していることが認識できる。

2.2.4　半導体

半導体の設計プロセスは唯一な手順として明確に規定されてはいない。設計環境の事情によってそのプロセスが代わる余地があるからである。従って、設計プロセスの手順についてシステム LSI の標準的な例を提示している鈴木(2003)の設計プロセス、すなわち図 2.5 で概観してみる。本研究で用いた経験的データや事例の多くは、特定顧客の要求仕様に基づいて設計したカスタム LSI であるからである。

設計プロセスは、四つの段階を経て製造プロセスに移行する。製造プロセスはアートワークという工程から始まる。設計プロセスの第一段階から第四段階までの手順を以下に示す。

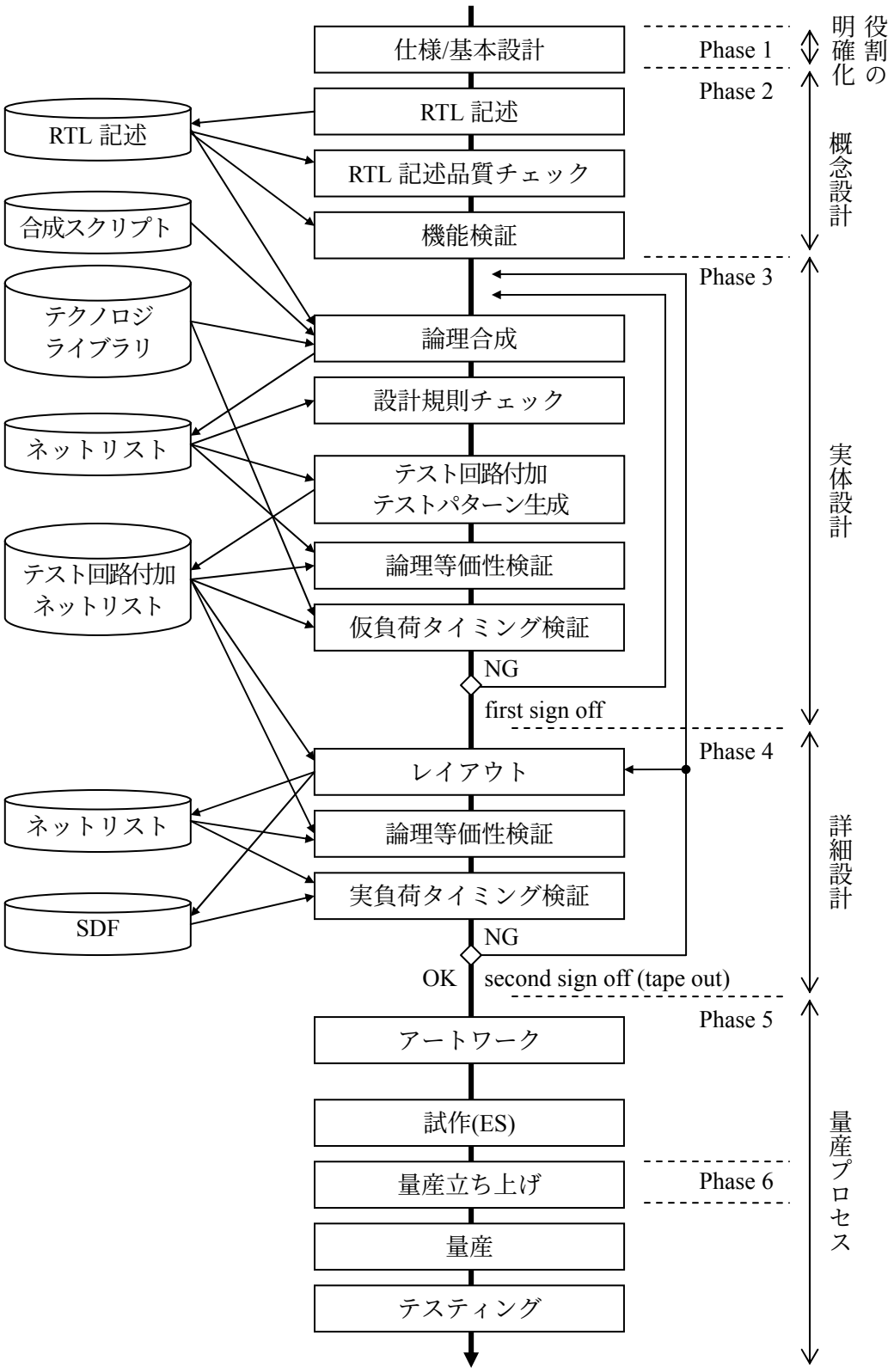


図 2.5　半導体の設計プロセス(鈴木, 2003:加筆)

第一段階

ステップ(1)　仕様と基本設計

システム LSI 全体の外部仕様、つまり機能面の仕様を明確にし、またその機能を実現するため処理方式を決定する。処理方式の主要事項を以下に示す。

i) データ転送方式：一相一サイクル転送にするか、又は複数サイクル転送にするかの意思決定を行う。
ii) クロック方式：クロック系統の数及び低消費電力化のためのクロック制御方式の意思決定を行う。
iii) ソフトウエアとハードウエアの切り分け：処理速度や全ゲート数などを考慮して最適な分割作業を実施する。
iv) アーキテクチャ：パイプライン(pipeline)方式か、又はスーパースケーラー(superscaler)方式かの意思決定を行う。
v) 端子の割り当て：ASIC (Application Specific Integrated Circuits)ではプリント基板(Printed Wiring Board)が先に設計されているのが一般的である。プリント基板上に乗る他の部品との位置関係から信号や電源、そして GND(グランド：電位がゼロ)の位置がほぼ決定してしまっている。この状況から信号の同時変化ノイズを考慮しながら端子を割り当てる。
vi) モジュールの割り出し：　このモジュールはハードウエア記述言語(Hardware Description Language)で表現された機能の単位を意味する。モジュールの階層構造を割り出し、各モジュールの入出力信号と処理を明確にする。モジュールの中にはレジスタ類も含まれ、どのクロック・サイクルでどの様な処理がなされるのか明確に規定する。つまりこの設計工程の段階でサイクル・アキュレート(cycle accurate)な状態となる。

ステップ(2)　RTL (Register Transfer Level)設計

ハードウエア記述言語を用いて、モジュール毎の機能の記述を行う。この機能記述を RTL 記述と呼ぶ。この時点で RTL の記述ファイルを作成する。

ステップ(3)　RTL 記述品質チェック

RTL 記述ファイルを読み込み、記述された内容に関して文法上、あるいは設計するにあたって予め取り決めしておいた記述規則に、違反しているか否かを検証する。

ステップ(4)　機能検証

RTL 記述ファイルを読み込み、論理シミュレータを用いて仕様と基本設計のステップ

で決めた仕様通りの機能が、ハードウエア記述言語で表現されているのか否かを検証する。次に機能検証（Functional Verification）において、論理を動作させる信号系列は最上にテストベンチ(Test Bench)として作成する。そしてコードカバレッジ(Code Coverage)等を用いてテストベンチの品質を検証する。

第二段階

ステップ(1)論理合成

モジュール単位に与えた入出力信号のタイミング制約などを記述した合成スクリプト(script)、配線負荷モデル(wire loaded model)並びにセル(cell)の内部情報を持つテクノロジ・ライブラリ(technology library)、そして RTL 記述ファイルを読み込む。そして論理合成(logic synthesis)ツールを用い手ゲートレベル(gate level)の設計情報、すなわちネットリスト(netlist)を作成する。

ステップ(2)設計規則チェック

作成されたゲートレベルの設計情報が、半導体ベンダー独自の論理構造規則に違反していないか検証する。

ステップ(3)テスト回路付加とテストパターン生成

ネットリストに対して、テスタを用いた製造不良テストを容易にするテスト専用回路を付加する。これは DFT (Design For Testability)のツールを用いる。そしてこのテスト専用回路に対して印可する故障検出用のテストパターンを ATPG (Automatic Test Pattern Generation System)を用いて生成する。

ステップ(4)論理等価性検証

テスト回路が付加される前後のネットリストを比較して論理的に等価であるか否かを検証する。テスト回路付加ツールが完全であるならば本作業は不要である。しかし論理合成ツールと同一メーカのテスト回路付加ツールを使うとは限らず、慎重を期するためにこの作業は行われる。

ステップ(5)仮負荷タイミング検証

テスト回路が付加されたネットリストと配線負荷モデルを使い、静的タイミング解析(Static Timing Analyzer)を行う。ここで検証結果が満足できない場合には、論理合成の手順まで戻り、論理合成制約を変更するなどしてタイミングを合わせ込む。

第三段階

ステップ(1)レイアウト

テスト回路が付加されたネットリストに基づいてセルの配置を行う。続いてあらかじめ規定されたスキュー(skew)値に収まる様に、バッファ(buffer)を挿入しながらクロックの配線経路を決定する。次に一般信号の配線を行う。スキャンチェーン(scan chain)に関しては配線経路が複雑にならない様に、接続する Dual フリップ・フロップ(Flip Flop)の順序を入れ替えるリオーダリング(re-ordering)処理を行う。

ステップ(2)論理等価性検証

レイアウト情報から逆にゲートレベルの全てのネットリストを抽出し、レイアウトで用いたテスト回路が付加されたネットリストと比較する。この比較作業は論理等価性検証である。

ステップ(3)実負荷タイミング検証

レイアウト情報から抽出した配線の抵抗Ｒ及び容量 C から配線の遅延とセルの遅延を計算し、SDF (Standard Delay File)を生成する。この遅延情報と論理等価性検証のステップで抽出したネットリストに基づいて、静的タイミング解析ツールでタイミング検証を行う。ここで検証結果が満足できない場合には、レイアウトのステップまで戻り、セル駆動能力の変更、バッファの挿入、あるいはセル配置の変更などを行う。この作業を実施してもタイミングの改善ができない場合は、論理合成のステップまで戻り、論理合成制約を変更するなどしてタイミングを合わせ込む。

この様に、半導体における設計プロセスを第一段階から第四段階に分類するなら、機械設計で規定している一連の設計過程と類似していることが認識できる。

2.2.5　一般設計プロセス

上記の工業製品の設計プロセスからは、共通の手順が見いだされる。まず設計プロセスは人工物の設計要求から出発している。この出発点から人工物を記述するために必要な属性、状態、挙動、機能などに関する情報が整理される。これが要求仕様である。次にその要求仕様から概念設計、実体設計、詳細設計という手順を踏んで、設計対象物という人工物の動作原理、構造や形状、あるいは挙動を最終的な設計解として決定していく。この手順をまとめると、設計プロセスの概念図は図 2.6 になる。

この様に設計プロセスの変換過程とは、要求仕様から設計解までの間で連続する、人間による変換の行為と言える。つまり、設計プロセスとは人間の知的行為が関わっている。この知的行為に伴う設計対象物はモノ的な動きは捨象されており、設計対象物の属性、状

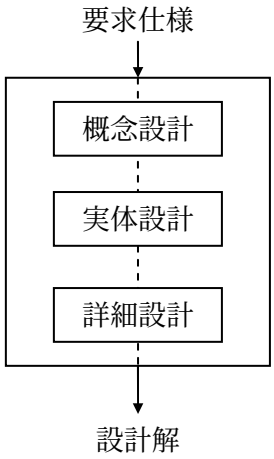


図 2.6　一般設計プロセス

態、挙動、機能といった基本概念が変換されている。この変換過程は、つまり設計情報と設計知識の流れと見ることが可能である。システム思考的な視点から見れば、設計プロセスは設計情報と設計知識の創造・処理のシステムとも言える。

この様な設計プロセスを経て、生産や製造に必要な生産設計が行われる。生産設計のプロセスは、設計対象物である人工物を生産あるいは製造するために必要不可欠な情報の生成過程である。

2.3　イタレーションに関する先行研究

設計プロセスにおけるイタレーションが設計負荷の重要な要因であるにもかかわらず、従来の設計管理手法ではイタレーションについて明確に言及していない。従って、多くの複雑なシステムの開発プロジェクトではイタレーションに対する処理が行われてこなかった。そして設計プロセスのモデルにイタレーションを、十分反映させることが行われてこなかった。

しかし全くないわけではない。後述する管理手法やツールが欧米を中心に提案されてきているからである。そこで次項では先行研究で議論されているイタレーションに対する管理手法、イタレーションの発生と含意、そして分類と効率化について述べる。

2.3.1　イタレーションと既存の管理手法

企業は優れた製品を提供している間にも、製品開発にかける所要時間と所要コストの低減を求める。その理由の一つはそれが競争の激しい市場に投入されているのであれば、競

争優位を継続させるために必要だからである。実際に、製品開発の効果的管理手法が存在するなら、その企業の競争優位の持続につながる(Clark & Fujimoto, 1991; Clark, 1993; Wheelwright, 1992)。設計プロセスを改善することの裏には、まず効率的な設計プロセスの本質を理解することである。この目的を達成する為に、設計プロセス内のタスク間で生じる設計知識や設計情報の流れを認識することが重要事項の一つである。その複雑な流れは、タスク間で相互依存関係にあるか否かの活動として見ることができる。

タスクへの分解と統合化という作業は、作業プロセスの成功を左右する重要な意思決定である(von Hippel, 1990)。製品開発における設計プロセスは、長年蓄積されてきた知識に基づく手順に従ったものである。そして繰り返されてきたものでもある。従って、現在の設計プロセスを見つめ直し、適切なプロセスのモデル修正と構築の検討が求められていると言える。

複雑化しているシステムの設計プロセスの主たる特徴は、設計プロセス内部のタスク間が相互に依存関係にあるものである。すなわち、この相互の依存関係とはタスク間のイタレーションである。そしてその源泉は多くの研究者によって観察されている(Fredriksson, 1994; Peters, 1983; Safoutin & Smith, 1996; Singh et al., 1992; Susman, 1992; Whitney, 1990)。イタレーションを持つ設計プロセスは、要求仕様書にできるだけ近づける様に繰り返される設計過程でもある。現在では複雑な工業製品のシステムを設計する場合、一度も手戻り修正せずに要求仕様書を完全に満足させて具現化させることは困難である。

従って多様な代替案が検討される。場合によっては、概念設計の初期段階に仮想的なプロトタイプを構築し、問題点を洗い出しておくこともある。その他に、プロセスの枠組みとして明示的なイタレーションを予め含み入れておく場合もある(Alford, 1994; Blanchard & Wolter, 1990; Clausing, 1994; DoD, 1992; DoD, 1994; Pugh, 1991; Shishko et al., 1995)。ただしこの明示的なイタレーションとは順方向の手順で流れる一連のプロセスの中に組み込まれたものである。そしてイタレーションの周期や回転の数は固定している。つまり、スケジュール的に確実に予測されたタスク集合である。これらは、不確実性を持ったイタレーションに対処できる設計プロセス管理について多く言及されてはいない。

プロジェクト管理の枠組みでは、イタレーションを予め想定している手法は少ない。例えば、設計プロセスのモデル化に一般的に利用されてきた PERT (Project Evaluation and Review Technique)は、一方向の表現しかできない。PERT だけでなく、作業工程や処理の順序を図式化したフローチャート図（flowchart:流れ図とも呼ぶ）もイタレーションを表現することができない。PERT やフローチャート図は元来、イタレーションの存在を想定していなかったのである。

ただ IDEF (Integration Definition for Function Modeling)の基本モデルは双方向の流れを表現することが可能である。IDEF とは業務改革の推進と情報システム統合を目的としたビジネス・モデリング手法である。IDEF は 1970 年代初期に米国空軍の ICAM (Integrated Computer Aided Manufacturing)プロジェクトで空軍の調達機材の製造にコンピュータを活用して効率

化を図ろうとした結果からまとめられた(Marca & McGowan,1988)。しかしこの IDEF は手戻り方向、すなわち逆向き方向だけの表記はできない。従って IDEF も潜在的にタスク間のイタレーションを想定して開発はされてはこなかったのが実態である。もちろん、スケジュール管理の代表的ツールであるガント・チャート(Gantt Chart)もイタレーションを表現する機能を持ち合わせていない。

この様に、既存の管理手法やツールは根本的にイタレーションを潜在的に想定していなかった。しかし近年の経営科学や OR (Operating Research)の領域では、この本質的課題に取り組む姿勢が散見される(Ahmadi & Wang, 1994; AitSahlia et al., 1995; Belhe & Kusiak, 1996; Carrascosa et al., 1998; Denker & Steward, 1996a; Eppinger et al., 1994; Eppinger et al., 1997; Ford, 1995; Grose, 1994; Koushik & Mookerjee, 1995; Krishnan et al., 1997; Loch & Terwiesch, 1998; Singh et al., 1992; Smith & Eppinger, 1997a; Smith & Eppinger, 1997b; Smith & Eppinger, 1998; Steward, 1981a; Steward, 1981b; Taylor & Moore, 1980, Yassin & Braha, 2003; Yu et al., 2003)。それらの論文は米国の宇宙航空産業、自動車、機械、精密機器といった工業製品を対象としたものである。しかしながら、それらの研究では設計プロセスのタスクが大きくなく、最大 40 程度である。また半導体設計という工業製品を取り上げてはいない。そして、その半導体設計で持つイタレーションの特徴を解釈した解決策を提供しているわけではない。

その半導体設計におけるイタレーションの特徴とは、50 以上の設計タスクで構成されていること、偶発的な要因によりプロジェクトのリスクが大きいこと、イタレーションに従属してタスク内容が変更されること等である。こういった特徴を持った設計プロセスは、既存の設計管理手法や上記の先行研究の枠組みで適用することは困難である。従って、実践で対処することができない状況にある。

そこで本研究ではその特徴と現象を把握し、モデルとして定式化させることが重要な一項目と言える。

2.3.2　イタレーションの発生と含意

設計のイタレーションの含意はタスク間の手戻り作業や改善という修正工程である。この修正工程は設計に関する情報によって生み出される。例えば設計上の目標値を達成できない場合に、その達成できないと言う情報に基づき以前のタスクに戻って修正を重ねる。それ以外にも数多く存在する。その一つは、新しい情報取得によるものである。例えば、設計途中で改善に向けた情報が新しく得られた場合に、その情報を設計に反映させるための修正行為が行われることがある。

他に設計と設計検証の反復である。設計はその作業が終了してから、その設計の検証作業が行われる。この時には検証結果次第によっては設計の修正作業が求められることがある。設計から検証という工程の流れを順方向と呼ぶなら、検証作業を終了してから再設計するという逆方向が発生する場合もある。

イタレーションは自動車産業でも見られる。Clark&Fujimoto (1991)は、自動車産業におけ

る技術者の設計工程を詳細に研究している。そこでは技術者間で設計に関する情報がやりとりされ、修正作業が双方で繰り返されているのを発見している。つまり上流から下流に向けた修正作業と、下流から上流に向けた修正作業が各タスクを起点として生じているのである。各タスクを起点として生じる新しい情報がイタレーションの勢いを強めていると指摘している。

イタレーションは、設計目標を満足させようとする行為からも生じる。Eppinger et al. (1997)は、そのイタレーションを“the repetition of activities to improve an evolving design”と定義している。つまり、進展過程にある設計を更に改善させようとする繰り返し活動である。設計仕様通りに収束させることが困難な状況は実際、存在する。それは、例えば予め最初に設定しておいた性能目標が不適切な場合である。その様な状況下ではその目標値に足りないと想定される因子を抽出する。続いて設計プロセスの上流に戻ってその因子が発生しない様に仕様の再検討を行う。これも手戻り修正である。

他に設計上の必要条件や目標値が明確に決定されていない場合にも、イタレーションが生じる。これは十分な設計情報を入手しようとする行為の繰り返しである。この行為は不安定な状況をつくり出す可能性が高い。この様な不安定な状況を引き起こす行為は、設計プロセス上の真の目標を見失いやすい。ここでその真の目標とは、設計プロセスの所要コスト、スケジュール、そして製品性能である。

先行研究の中では、設計プロセスの所要時間を決定付ける因子としてイタレーションを取り上げている論文もある（Cesiel, 1993; Clark & Fujimoto, 1993; Eisenhardt et al., 1995; Eppinger et al., 1994; Gebala & Eppinger, 1991; Singh, 1992; Wheelwright, 1992; Whitney, 1990）。しかしイタレーションは、所要時間やスケジュールだけに影響を及ぼしているのではない。所要コストや製品性能にも影響を及ぼしうる。むしろ設計プロセスを一つのプロジェクトと考えて、イタレーションを反映させた方が全体を見渡すことができる。つまり、スケジュールだけでなく、設計プロセス上の所要コスト、製品性能も検討に含み入れる方が、経営の視点では有益である。

2.3.3　イタレーションの分類と効率化

設計プロセス内の各タスクが複雑な依存関係にある場合には、イタレーションが発生する。このイタレーションが大きくなれば、スケジュールとコストを目標値以内に着地、すなわち設計を収束させることは困難になる。従って設計プロセスを成功に導くには、設計のイタレーションを戦略的に計画策定することである。この計画策定を勧めるには、設計プロセスで生じるイタレーションをまず二つの領域に分けると判りやすい。それは想定内のイタレーションと想定外のイタレーションである。

想定内のイタレーションとは想定範囲にあるイタレーションである。このイタレーションには設計目標を達成させるために有効な設計情報の交換が含まれる。その他にイタレーションの効果は学習効果である。学習効果は習熟曲線理論によって裏付けされている。習

熟曲線の活用の一つは人的要因に起因する工数逓減の現象である。

イタレーション効用はイノベーションの創造とも関係している。von Hippel (1994)は新製品や新しいサービスの設計開発の現場で、イノベーションに関連した問題解決に向けて知識情報のイタレーションを観察している。小川 (2001)もその現象を捉えている。情報の流れという観点から、イノベーションが発生する場所の多様性とその多様性を説明する要因について考察している。

一方、想定外のイタレーションとは想定範囲外にあるイタレーションである。イタレーションは想定外にあるので、計画策定は困難である。例えば設計プロセスの中で予期せぬ時に新しい情報が入り込む場合、想定外のイタレーションが発生する可能性が高い。この中には必要条件や目標が流動的であること、単純な誤りで起因することが含まれる。

分類に関しては、Clausing (1994)も同様な概念構築で行っている。その分類は三種類である。創造的なイタレーション(creative iteration)、規律のあるイタレーション(disciplined iteration)、機能障害のイタレーション(dysfunctional iteration)である。創造的なイタレーションと規律のあるイタレーションは、想定内のイタレーションに含み入れることができる。そして機能障害のイタレーションは想定外のイタレーションと同類と考えることができる。

ここで簡単に上記の議論をまとめると、まずイタレーションは分類できるということである。そして、そのうちの一つである想定外のイタレーションは設計プロセス上のスケジュール、コスト、製品性能に予測困難な影響を及ぼす可能性が高いということである。

つまり、イタレーションによって増加する設計プロセス上の問題を解決するには、まず想定外のイタレーションを低減させることである。これは想定外のイタレーションを予測しうる様に検討することである。例えば、設計プロセス上の真の情報を適切な時に適切な場所で得られる様にすること、適切な制約条件の元に設計活動が適切に行われること、必要な時に必要な資源を設計活動にもたらすこと、必要条件をできるだけ早く決定しておくこと、単純な誤りの発生を低減させておくこと等である。

想定外のイタレーションを低減させた後は、次に想定内のイタレーションを管理することである。そこには二つの効率化の方法が考えられる。一つはイタレーションの回転速度を上げることである(Browning,1998a)。二つ目はイタレーションの回数を低減させることである(Smith & Eppinger, 1997a)。

イタレーションの回転速度を上げる、回転の数を低減させるという手段については、Browning (1998a)や Smith & Eppinger (1997a)が議論をしている。例えば、CAD (Computer Aided Design)システムが各個人の活動を速めること、シミュレーションや解析ツールがプロトタイプ(prototype)のテスト所要時間を短縮させること、共同作業の改善を内部組織の各個人までさかのぼって徹底させること、個人の活動が相互依存関係にある人全てを対象に物理的に同一の場所で作業をさせること、コンカレント(concurrent)・エンジニアリングを徹底させること、製品開発を統合させること、設計プロセスに無関係な活動を取り除くこと等が事例として取り上げられている(Browning, 1998a; Smith & Eppinger, 1997a)。

回転速度を上げることに対する研究は他にも存在する。Eisenhardt et al. (1995）並びに Gebala & Eppinger (1991)の研究結果によれば共に、回転速度を上げたイタレーションは短納期の製品開発と相関関係にあるとのことである。そして設計プロセス内の構造を改善させることでイタレーションは減少すると言える。例えば、設計プロセス内の相互依存関係にあるタスクを独立させることが可能であるならタスクは簡単な配列となる。すると相互の共同作業を減少させることにつながるので、イタレーションを減らすことができる。しかしながらイタレーションが存在しないという前提の場合には、設計プロセス上の所要時間、コスト、製品性能が改善しないという可能性も含み入れられる。つまり目標を達成できない可能性、すなわちプロジェクトのリスクである。この場合にはイタレーションを増やすことでそのリスクを低減させることができる場合がある。イタレーションには前述の通り、設計修正や学習効果が期待できるからである。

設計結果が確実に受容範囲にあり、そして生産性が向上しているなら、イタレーションの存在はない方が好ましい。しかし設計結果が受容範囲に入らない前提条件であれば、イタレーションの付加と高速化は効果的な活動の一つである。

2.4　イタレーション・モデルに関する先行研究

設計プロセスにおけるイタレーションに関しては、多様な視点で数多くのモデル研究が重ねられている。そのイタレーションのモデルの多くに共通した特徴は、設計プロセス内を個別のタスクに分解し、タスク間で生じる情報の受け渡し過程を観察したものである。つまり、設計情報の流れに注目して構築されたものである。

モデル表記の方法はマトリックス手法とネットワーク手法がある。次項からは個別のモデル毎に先行研究の概況を述べる。

2.4.1 マトリックス手法

タスクは設計プロセスの構成要素である。つまりタスクという一つの設計活動が複数集合することで設計プロセスは構築されている。タスク間のイタレーションを明示的に表現する簡単な一手法は、対称形のマトリックスを利用することである。Steward (1981a, 1981b) によって提唱された DSM (Design Structure Matrix:設計構造マトリックス)は、タスク間の相互依存関係に注目したマトリックス手法である。DSM として確立される以前には、システムの分割方法としてマトリックス手法が活用されたことがある(Alexandar, 1964)。

設計プロセスをモデル化する原点としては、グラフ理論やネットワーク理論に基づく図表を使用することである。マトリックス手法とネットワーク手法を簡単に対比する上で、二つのタスク A と B から構成される設計プロセスを考えてみる。この二つのタスクはプロジェクト業務の作業要素である。タスクはその作業内容を示すと共に、時間量、費用、リソース(resource)の情報を含む。

ネットワーク図では、あるタスクからもう一つのタスクへ及ぼす影響については、単なる結合線を用いるのではなく、矢印で示す。つまり矢印によってそのタスク間の相互依存性を説明することができる。

設計プロセスで考えられるタスク A と B 間の関係に関しては、三つのモデル、すなわちシーケンシャル(sequential)、コンカレント(concurrent)、相互依存(coupled)で代表できる。ネットワーク及び DSM によってそれぞれ表される三つのモデルを図 2.7 に示す。三つのモデルの動作を理解することで、設計プロセス全体の動作を理解できる。

まずシーケンシャルでは一つのタスクが別のタスクの動作あるいは決定に影響を及ぼすものであり、関係は一方向である。このモデル例では、タスク B を開始する前に、タスク A が実行されておかなければならず、タスク B は、タスク A の設計の影響を受ける。コンカレントの場合には、タスク間に相互依存関係の作用はない。相互依存の場合には、モデル例の通りタスクそれぞれが相互依存関係にある。そして双方のタスクが影響を及ぼし合っている。双方のタスクはそれぞれ、影響を受ける定量的度合いが判らない。従って、確定的な意思決定を行うには困難な立場にある。このサイクルの状況がイタレーションである。

一方、複数のタスク工程を DSM の定義によって表現するなら、タスク *n* 個の構成で、*n*

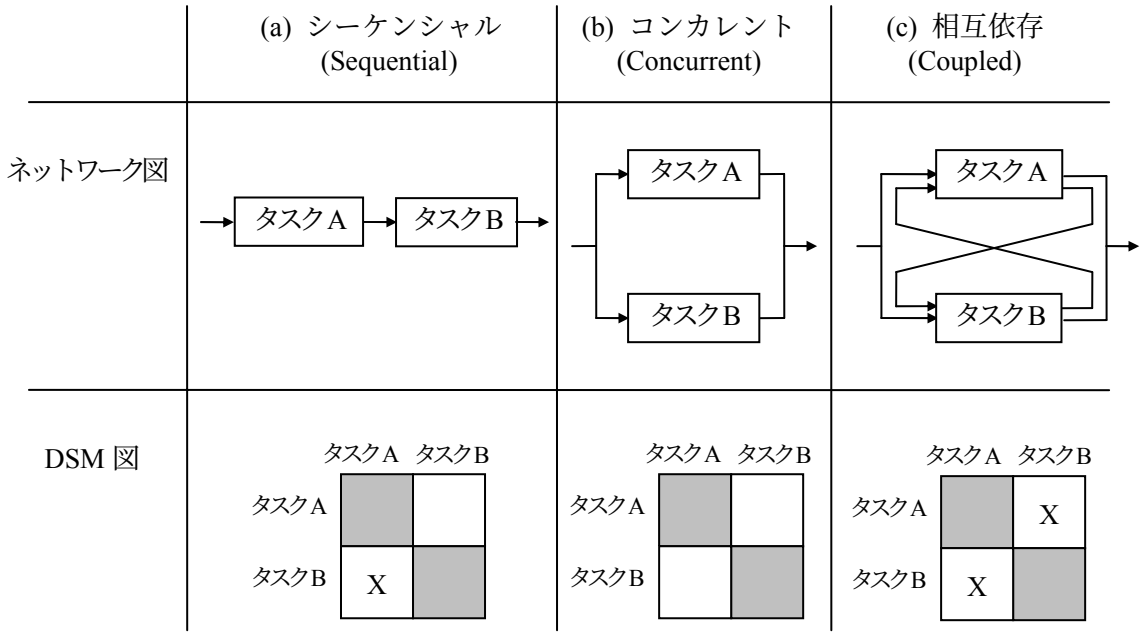


図 2.7　ネットワーク図と DSM 図によるタスク A とタスク B の三つのモデル

次の行と列を配置させた正方行列となる。これを図 2.8 に示した。タスクの位置づけは、設計プロセスの時間軸に従って行の上流から下流へと並べ、タスクの行の順番（上流から下流の方向）とタスクの列の順番（左側から右側の方向）は合わさなければならない。行列の第 i 行、第 j 列に交差する部分にある(i, j)成分に X（この X は二進数の 1 で表すこともできる）がある場合には、行列の第 i 行にあるタスクから、行列の第 j 列にあるタスクの方向に向かって依存関係があることを表現できる。逆に(i, j)成分には何も存在しない（この状況は二進数の 0 で表すこともできる）なら、行列の第 i 行にあるタスクから、行列の第 j 列にあるタスクの方向に向かっての相互依存関係はない。

また DSM において、ある列のタスクに注目するなら、その列のタスクを実行してから、次にどのタスクを実行しなければならないかが判る。またある行のタスクに注目するなら、その行のタスクはどのタスクから情報を得ていることが判る。

タスク間の依存関係における方向性については、行列の対角成分を基準境界線として容易に理解できる。例えば、$i > j$ の場合、すなわち行列の対角成分より下に X があれば、タスクが上流から下流へのフィードフォワード(feed forward)である。フィードフォワードはウォーターフォール(waterfall)又は順方向と呼ばれる。そして $i < j$ の場合、すなわち対角成分より上に X があれば、上流のタスクが下流のタスクに依存するフィードバック(feed back)を表す。このフィードバックとは逆方向、又は手戻り修正である。

この様に設計プロセスに二進数の一対比較行列を適用することで、タスク間の相互依存の関係の有無を表すことができる。DSM は設計プロセス全体が一覧性の表であるので、タスク数の多さにかかわらず容易にかつ明確に判読できるメリットがある。

Eppinger et al. (1994) は設計プロセス上のイタレーションに DSM を活用しているが、一対比較行列の成分を二進数ではなくパラメータ変数を適用している。このパラメータ変数に

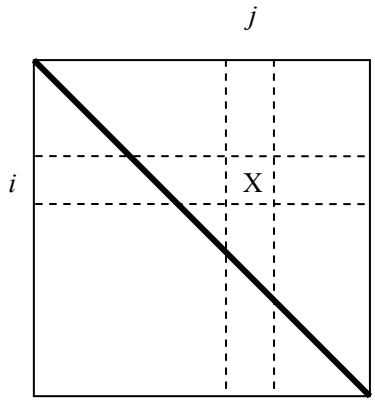


図 2.8　タスク i とタスク j の依存性

よってタスク間の相互依存関係の有無だけでなく、タスク間の相対的な相互依存の関係を定義することが可能になった。この変数は、重み付けや多種属性に基づく目的関数から算出することが可能である。仮に一対比較行列の成分が決定されるものであるなら、この行列を線形写像として利用することが可能である。そして線形解析として利用ができる。

Eppinger et al. (1994) は線形解析を用いて設計プロセスの所要時間を短縮させる戦略を提示している。その戦略は二つある。しかしそれぞれ逆説的な表現である。一つ目はタスク間の相互依存関係を低減させることであり、二つ目は相互依存関係にあるタスクの依存性を更に高めることである。

この二つの方法は、設計プロセスの状況に応じて使い分けることが求められている。まずタスク間の相互依存関係を低減させることとは、あるタスクと相互依存関係にあるタスクとの結合を最小にすることである。この目的が達成されるなら、設計プロセスの所要時間は短縮される。次に相互依存関係にあるタスクの依存性を更に高めることとは、コンカレント・エンジニアリングを進めることである。コンカレント・エンジニアリングは DSM ではタスクの処理を再配置することである。つまり設計プロセス内で生じる反復修正をクラスタの中で、短時間で終了させるという含意である。プロジェクト管理者や設計プロセスのモデルを検討する技術者にとって、この戦略に基づく知見は有益である。

DSM では分割論について議論が展開されている。Denker & Steward (1995, 1996a, 1996b) は Steward (1981a, 1981b)の初期の研究を詳述し、一つの知見すなわち分割(Tear)についてまとめている。この知見は Eppinger et al. (1994)の結論と類似している。Denker の知見は相互依存関係にあるタスク群を、周期が短いイタレーションと長いものに分類する。そして周期が短いイタレーションを持つタスク群をタスク集合として、適切に分割する手法である。このアプローチはタスク間の相互依存関係を低減させる一つの手法である。Altus et al. (1995) は DSM と遺伝アルゴリズムを活用して分割論 (partitioning)を展開している。これは周期が短いイタレーションとなるタスク集合の数を最適化させる研究である。Rodgers (1989, 1996a, 1996b)は遺伝アルゴリズムを拡張して同研究を発展させている。

Smith & Eppinger (1997a)は一対比較行列の行列成分を確率変数で適用している。先行タスクに戻る手戻り修正作業量は、手戻り修正作業を引き起こすタスクの作業量に確率変数を乗じた量と定義する。すると対角成分以外の行列成分が確率変数となる一対比較行列がイタレーションのモデルとなる。一対比較行列の列をタスクと見るなら、各列の成分はそのタスクが引き起こす手戻り修正の相対値である。手戻り修正の段数は定義づけすることができるが、段数を無限と考えるならマルコフ連鎖の研究領域である。本イタレーションから生成される所要時間と所要コストは決定論的(deterministic)ではある。しかしながら、タスク毎に相対的負荷を定量的に解析でき有益な手法である。

Smith & Eppinger (1997b)はイタレーションが生じていないタスク処理、すなわち順方向だけのタスク処理についても DSM と確率変数を用いて研究を行っている。そこでの知見は確率変数の高い成分をできるだけ行列の対角付近に集合させ、再配列化させることである。

これが総合作業量の最小化に、つながるというものである。

一方、DSM は米国では実務的にも利用されてきている。アメリカ航空宇宙局(National Aeronautics & Space Administration: NASA)では 1989 年に、大規模で複雑化している設計工程内の相互依存を解析する研究が行われている。そこでは、NASA で開発したナレッジベース(knowledge-based)の設計管理ソフトウエアに DSM の機能を組み込みしている(Rodgers & McCulley, 1996)。1990 年代に入り、本手法はマサチューセッツ工科大学(Massachusetts Institute of Technology)及びイリノイ大学(University of Illinois at Urbana-Champaign)を中心として研究が進められ、米国の軍事・宇宙産業、航空機、航空機エンジン、自動車の分野で成果が報告されている（Browning ,1998a）。

近年は OR (Operating Research)の研究分野でも DSM のモデルの活用が行われている。Ahmadi & Wang (1994) は、タスクに習熟曲線の影響を与えるモデリングを実現している。Carrascosa et al. (1998) は、オーバーラップ (overlap) 型開発を想定して、DSM モデルのタスクを構築している。そこでは多様な代替案により異なるプロジェクトの所要時間が得られることを確認している。オーバーラップ型開発とはシーケンシャルのタスクそれぞれを部分的に重ね合わせ、同時並行的にタスクを進行させる方法である。つまり、タスクが終了する前に次のタスクを終了させることである。この方策によって前倒しができるので、納期を短縮できる。

それ以外にもフィードバックを減少させる為の DSM の最適化についての OR 研究も重ねられてきた（Denker & Steward, 1995; Kusiak et al., 1994; Rodgers & McCulley, 1996; Steward 1981a, 1981b）。

Browning (1998)は、イタレーションを前提としたプロジェクトにおいて、不確実性が高くなるスケジュールの工程時間を見積もる方法や修正作業や修正にともなうプロセスをストカスティックに表現させ、その影響度合いを考察している(Browning ,1998a; 998b)。

Yassine & Braha (2003)は、大規模複雑化しているエンジニアリング工程においてコンカレント・エンジニアリングの有効性を説いている。しかし、その適用にあたってイタレーションのみならず、オーバーラップ、分割（decomposition）と統合(integration)、収束(convergence)といった四つの問題点が普及を妨げていると指摘している。そして DSM にこれらの問題を解決した統合モデルを提唱している(Yassine & Braha, 2003)。Yu et al. (2003) は DSM 上で、モジュラー型 (Modular) のアーキテクチャ (Architecture) を構築している。そして遺伝的アルゴリズム (genetic algorithm)によって、最適なクラスタ化の実現ができることを報告している。

2.4.2 ネットワーク手法

エンジニアリング設計においては、将来的な原価見積りやスケジュール結果を解析するための様々なアプローチが提唱されている。特にガント・チャートは歴史的に利用されてきた。一方、現在まで工程管理には広く認知され、様々なプロジェクト管理として利用さ

れている PERT (Program Evaluation and Review Technique)は、ネットワーク理論に基づいたモデリング技術である。

1958 年に米国で当時の技術革新時代の深刻な悩みを解決するために生み出された（森, 1964）。ミサイルの研究・開発、原子炉の建設、心臓外科手術、プロードウェーのショー演出等に至るまで新規プロジェクトの計画、実施に幅広く応用され、総所要期間の短縮や総原価費用削減などの効果を上げた（森, 1965）。

PERT はネットワーク的視点から、複雑化した設計工程をアクティビティー(activity)という工程作業それぞれの関係性を明示できうる点でその有効性があると解釈されてきた（森, 1964）。更にオペレーション・リサーチ系の研究者、モデリング技術者、そしてリスク・アナリスト等によって多くの時間と費用といった資源が投入され、PERT は洗練されてきた。

特に、アクティビティーのネットワークから最短の経路を見つけうる CPM (Critical Path Method)は、今や PERT の代表的な機能展開と言える。PERT やフローチャート化といった一般的手法は、今も伝統的かつ引き続き定型の工程を扱う現場には、最適な標準ツールと言える。

しかしながら PERT はアクティビティー間のフィードバックが起きないという前提で設計されたものである。したがってフィードバックを含むアクティビティー間のイタレーションを実現できるモデリングが検討され、GERT (Generalized Evaluation and Review Technique)が開発された（Moore & Clayton, 1976)。この技術手法は、イタレーションを取り扱うことができる他にシミュレーションでの解析を主たる目的としている(Neumann, 1990; Taylor, 1980)。PERT や CPM はフィードバックが規則上許されていない。一方、GERT (Moore & Clayton, 1976; Neumann & Steinhardt, 1979; Neumann, 1990)はプロジェクト・マネジメントの基本的な PERT にフィードバックの機能を取り入れたグラフ理論、ネットワーク理論の考え方であり、研究開発や製造工程管理、そして新製品投入に向けたマーケティング等への適用がされている。

Q-GERT (Queuing-Generalized Evaluation & Review Technique)は GERT を拡張させた手法である(Pritsker, 1977)。GERT にキューイング(queuing)機能と決定(decision)機能を拡張させた。キューイングとは、相手側の処理の完了を待つことなく、次の処理を行うためのデータ交換方式である。送信側アプリケーションは、データを一時的に保管するためのキュー(queue)と呼ばれるデータを一時的に保管するための待ち行列ファイルに送信データを格納するだけで次の処理を実行できる。

この様に、Q-GERT を含む GERT モデルは、スケジュールやコストなどの解析も実行できるように拡張されてきた。しかしそれらモデルは、まさに活性化されたアクティビティーを同時実行させたり、予見できない修正作業等を考慮していない。更に近年大規模・複雑化しているプロジェクトへの対応力も限界がある。例えばアクティビティー間の相互依存性を全て可視的にそして一覧性を持たすことは難しい。

一方、企業の業務改革の推進と情報システム統合を目的とした IDEF (Integration Definition

for Function Modeling)は、アクティビティーの双方向性を表現できるビジネス・モデリング手法である。IDEF 手法は 1970 年代初期に米国空軍の ICAM (Integrated Computer Aided Manufacturing)プロジェクトで空軍の調達機材の製造にコンピュータを活用して効率化を図ろうとした結果からまとめられた(Marca & McGowan, 1988)。現在 IDEF には、アクティビティ・モデリング手法を表す IDEF0、情報／データ・モデリング手法である IDEF1/1X、そしてプロセス記述獲得手法である IDEF3 がある。これら IDEF は、アクティビティー間の双方向性の関係を図面で表現できるが、特にフィードバックとフィードフォワードのイタレーションを明確的に意図したものではない。

上記の議論をまとめると、基本的には PERT はフィードバックを記述することが許されておらず、イタレーションは想定すらしてない。そして IDEF は、アクティビティー間の双方向性の関係付けは行うことができるが、フィードフォワードやフィードバックを明確に規定することはできない。PERT も IDEF も潜在的にアクティビティー間のイタレーションを想定して開発はされていないのが実態である。

一方、PERT の拡張モデルである GERT と GERT の拡張モデルである Q-GERT はイタレーションの記述は許されている。しかし GERT も Q-GERT もシミュレーションを主体としており、その点で大きな有効性を見るが、アクティビティーが大規模化そして複雑化した時のグラフ構造やそのプロジェクトのサイクル時間への洞察性に限界がある。そしてアクティビティーの順序も固定化してしまい、アクティビティー間の配列を柔軟に修正することが難しい。

2.4.3　イタレーション・モデルの有効性の確認

イタレーションの現象は、数学的モデルで抽象化が行える。これは観察と推論である。そしてそのモデル化に基づくシミュレーションは実現象から離れ、抽象化された閉じた世界で動作する。これは実験による検証である。本論の第 4 章と第 5 章で議論しているイタレーションについては先行研究で取り上げている DSM を用いることから出発している。そして数学的なモデルとして定式化させ、検証した実証研究である。しかし公理として設定しているこの DSM モデルは、そのモデルが含有している以上の解を求めることはできない。従ってこの DSM モデルの適用の妥当性を、実践現場の知見と照らして検証する必要がある。つまりモデルの正当性である。本節ではこの検証の手続きをイタレーション・モデルの妥当性確認(validation)と呼ぶ。妥当性の確認については、以下の視点で議論を進める。

i)　モデルの仮定条件

まずモデルの必要条件として、いかなるモデルも実現象に基づいて構築される必要がある。つまり数学的モデルがその必要条件に従っていないなら有効ではない。しかしそのモ

デルは実現象に基づいて構築したとしても、単純化しすぎると必要条件から外れてしまう可能性がある。モデルによって構築したシステムの特性が失われる可能性があるからである。例えば本研究で言えば、設計プロセスの規模並びに複雑性といった現象の特徴が失われれば、検証自身が有効と言えなくなる。

Smith (1997a)、Tyson & Eppinger (2002)及び第 4 章と第 5 章で用いた一対比較行列による定式化は、イタレーションのモデルである。実際の設計プロセスで生じる現象を定式化させている点で、先の必要条件とは調和が取られている。しかし必要となる条件を完全に満たしているとは言えない。この一対比較行列による仮定条件は、実現象に基づく必要条件と必ずしも一致するとは言えないからである。

実際の現場においては相互依存の関係にある複数のタスクは、組織の中で同時並行的に作業として進められることはある。しかし一対比較行列による定式化では一回のイタレーションで、全てのタスクが同時並行的に生じることを仮定している。従ってこの仮定であれば精度の面で、必ずしも完全に一致するとは言えない。

仮定条件によるモデルと実現象と同じ理想的モデルの不一致はそれ以外にもある。実際の現場では、一般的に反復修正を繰り返す毎に、タスクにおける修正作業の時間は減少していく。つまりイタレーションが生じる時のタスクの時間量はイタレーションを繰り返す毎に、イタレーション当たりのタスクの時間量は減少する。一対比較行列による定式化でもイタレーションが発生する毎に減少するモデルとなっている。しかしこのモデルは線形性を仮定条件としている。従って、これも厳密に言えば精度の面で必ずしも完全に一致するとは言えない。他に反復修正の回数は無限に行われることはありえないことも付け加えておく。上記の仮定条件だけでまとめてみると、モデルは実現象に基づいて構築されているものの、必ずしも完全な精度を保有しているわけではない。

ii)　モデルによる洞察

一対比較行列の定式化によって定量的な解析が実現できる。そこではどの様なタスク群が設計プロセス上に設計負荷を与えているのかが認識できる。第 4 章と第 5 章においては、その一対比較行列を大規模で複雑な半導体設計のプロジェクト事例に適用し、その認識性について確認している。そして一対比較行列はイタレーションが複雑に生じている設計工程を表現できることが実証できている。他に一対比較行列による代替案をモデル化することで、シミュレーションによる代替案の検証も行うことができる。特に実証プロジェクトのシミュレーションを通じて固有値と固有ベクトルの結果から問題自身を特定できる。そして固有ベクトルの成分の重み付けから、イタレーションの負荷に影響を与えているタスクの特定と各タスクの影響度合いを定量的に認識することができる。

つまり本モデルを利用することで定量的な解析が可能となり、意思決定に向けた評価基準が構築できる。本モデルは洞察力を持ち、有効な一手段と言える。

iii)　数学的モデルの動作と実現象との一致

モデルの定式化から算出された定量的な結果と実際の設計プロセスの振る舞いが、厳格に一致していれば信頼性は保持されることとなる。しかし本モデルが含有している以上の解を求めることはできない。従って、モデルを通じて得られる結果は、どこまで実現象に近づけられれば適正なのかという信頼性の問題に置き換えられる。しかし今日まで設計プロセスの振る舞いをデータとして取得している先行研究は見あたらない。イタレーションを実測する方法がまだ確立していないからである。従って数学的モデルの動作と実現象の比較対照は今だ実現していないのが現状である。

設計プロセスのイタレーションの実測値の取得は困難であるが、相対的評価についてはまだ議論がされていない。例えば、代替案を構築し同じ条件下でシミュレーションすることで、相対的な振る舞い評価を行う事などである。この評価方法はモデルの妥当性を確認する上での今後の課題の一つと言える。

iv)　他モデルと比した優位性

モデルの評価基準の一つは、先行研究で提示されている他のモデルとの比較対照である。この場合には、まず過去に最適なモデルとして認知されたモデルが存在するか否かに依存する。先行研究で述べた通り、ネットワーク手法にはフィードバックを可能にするモデルがある。それは Q-GERT である。しかしイタレーション毎にタスク作業量を線形的に減少させることはできない。またタスクが多くなると一覧性に欠ける。さらに実践現場で普及していないのが現状である。従って現在まで、設計プロセスの振る舞いを定量的に表現できる手段はマトリックス手法以外に構築されていない。現在提供されているモデルとしては DSM が唯一であり、その存在としての意義は大きい。しかし本モデルは厳格な精度で表現することに課題が残っているので、引き続き改善を進める必要がある。

v)　モデルと利便性

モデルの妥当性の評価として、次に利便性の視点で見てみる。この利便性とは実験の作業性である。つまり、モデル構築作業とシミュレーション作業を都合良く進められるかどうかということである。シミュレーションの作業はモデルに依存しているので、もしモデル自身が複雑であるならシミュレーションに向けた準備作業は困難になる可能性がある。従ってモデルはできるだけ簡素化させた方が、利便性は高まる。しかし簡素化しすぎると精度の確保が困難になる。例えばモデルの精度によっては、検証すべき設計プロセスの振る舞いのばらつきが大きくなる可能性があるからである。この様にモデルを簡素化させると利便性は高まるが、精度が確保できなくなる可能性がある。つまり利便性とモデルの妥当性は相反関係になる可能性があるということに留意しておく必要がある。

尚、DSM を利用したモデル構築の手順については本論で詳細に述べている。そして一対

比較行列を定式化して設計プロセスの振る舞いを実証している。これらは全てイタレーションのモデル構築を経験したことがないプロジェクトの管理者であり、滞りなく作業を実行している。

vi)　モデルの制御性

設計プロセスのモデルを適切な方向にもたらすことは、設計プロセスに対する一つの制御である。制御はモデルを活用する上で効果的な機能の一つである。ガント・チャートやネットワーク図は制御性という点では柔軟性に欠ける。例えばイタレーションを反映できる機能を持っていないことは一つの限界である。一方、一対比較行列を用いたモデルではその点で制御性が優れている。例えばイタレーションが発生しているタスク群としていないタスク群を二分化させる分割やクラスタ化である。

ここで設計プロセスの中で相互依存関係にない、すなわち独立しているタスク群がある場合を想定する。このタスク群は制御する必要がない。この仮定条件ではそのタスク群はイタレーションが生じないからである。従ってもしそのタスク群だけを分割して管理の対象から外すことができるならタスク数を減少させることができるので、制御性は高まると言える。この有効性について本論で詳細に論じる。

vii)　イタレーション・モデルの妥当性の含意

本モデルの欠点は上述で指摘した通り、厳格な精度を保有していない点である。しかし実現象に基づいて構築されており、モデルとしての条件を満たしている。しかもプロジェクト管理者や設計当事者に有益な情報を提供してくれる。その有益な情報とは適切な意思決定に結びつく情報のことである。そしてその情報は直接的並びに間接的に入手されるものである。本項ではその情報は具体的にどういった有益な情報なのか以下に述べる。

一つ目は合意形成である。例えば、モデル開発者やプロジェクト管理者がモデルを構築する前に、タスクそれぞれの相対的な設計負荷はどの程度であるのか、そしてタスク間の相互依存関係はどの程度であるのか、それぞれ定量的な分析結果を共有できる。従って設計プロセスのモデル構築の上では合意形成に役立つと言える。

二つ目の有益情報としては、固有値問題を解くことにより設計負荷を引き起こしている問題因子、そしてその因子に直接関係しているタスク群を特定できる点である。PERT の様にイタレーションをサポートしていないモデルを利用するならば、こういった情報は得られない。なぜならば一対比較行列による定式化と比べて、明らかに PERT によるモデルの乖離が大きいからである。

イタレーション・モデルを否定する立場の前提としては、イタレーションは発生しないことを仮定条件としている。その仮定条件は、繰り返し作業は全て非生産的な活動であるとの認識(Warfield, 1990)から作り上げられている。しかしタスクが継続して繰り返されるプ

ロジェクトは現存しており、本論で提示しているモデル手法はその現象を創り出すことができる。従って設計関係者の認識をモデルに反映できるので、本モデルは効果的である。

本モデルによって得られるもう一つの有益情報は、経営の意思決定につながる科学的情報であるという点である。つまり経営の行動指針として客観的な正当性を与えることができる点で効果があると言える。

2.4.4　イタレーション・モデルに関するまとめ

上述で展開されている先行研究から認識できることは、設計プロセスのイタレーションは多面的視点でその様相の議論ができることである。そしてそれぞれの研究論文における主張は、プロジェクト管理への貢献期待があることである。ただしモデル自身については、課題は残っている。上記で議論した DSM のモデルは実現象に基づいて構築されているが、完全な精度を保有しているわけではないからである。従って、イタレーション自身、並びにイタレーションのモデル化に関する研究は将来的に継続されていくと考えられる。そして引き続き収集を重ね総合化させていく必要がある。

次節では現在まで実践で活用されてきたプロジェクト管理で、イタレーションを処理する時の限界点について述べる。そしてイタレーションの認知がされてこなかった理由についても探求する。

2.5　イタレーション発生時の現行の管理手法の限界点

設計プロセスのイタレーションに対処しなければならない企業は存在する。しかし本質的な課題がイタレーションを要因としているにも係わらず、議論のすり替えが行われている場合がある。例えばイタレーションの起因は、個人の人為的な問題である、または要求仕様が頻繁に変更されることによる、との誤った指摘である。この様な場合、総論の中で議論されるイタレーションは、各論の中で議論されるイタレーションよりイタレーションの理解を遅らせる傾向にある。するとイタレーションを計画策定に反映させることは困難になってしまう。実際の所、現行の管理手法はイタレーションを想定していない。そしてそのモデル化もできない。つまり、イタレーションの視点を最優先としていないことが限界点の一つである。

次項では引き続き、半導体設計という実践プロジェクトの観察を通じて得られた知見に基づき、その限界点について具体的に説明を行う。次項で述べるイタレーションに対する現行の管理手法の限界については、半導体だけでなくその他の工業製品にも共有できる可能性がある。

2.5.1　スケジュールの心理的抑圧

特定顧客向けの半導体製品の設計プロセスの場合、まず顧客による製品購入の意思決定と要求仕様の決定から始まる。設計プロセスにおける目標は、適切な製品を適切な原価で、適切な時期に提供することである。製品を市場に投入する上で検討される適切な時期とは、顧客との契約上の日程である。契約上の日程通りに顧客に提供することによって、そのプロジェクトは終了する。設計プロセスの工程中、製品提供の時期が契約上の日程から遅延すると想定される場合には、企業は契約上の日程に見合う様に努力を行う。従って設計プロセスは、スケジュールという要素によって駆り立てられると言える。

適切な計画策定が実施されていないプロジェクトの場合は、スケジュールを中心として終始、関係者が動かされる可能性がある。時には危機感という緊張感が強いられる。その様な状況下においては、設計プロセス上のイタレーションを持つプロジェクトも存在するはずである。その場合には、適切な計画策定がされていないのであるから、イタレーションを想定していない可能性は高い。こういった状況下では、イタレーションが顕在化することで、スケジュールのリスクが顕在化する可能性がある。

2.5.2　リソースの制約

適切な時期までに必要なリソースを確保することは、プロジェクトを開始する上で必要なことである。しかしプロジェクトに向けて十分なリソースが得られているもかかわらず、まだそのプロジェクトが開始されていないならば、リソースの無駄が発生していることを意味する。

実体設計や概念設計に必要なリソースを確保したままの状態で、まだプロジェクトの抽象的な上流設計である概念設計を行っているなら、経営上大変クリティカルな状況である。それは実体設計や詳細設計の開始時期まで、不必要なリソースが消費されるからである。

従って設計者はリソースが確実に確保される時期を考慮しながら、概念設計を行う方が好ましいと言える。概念設計を行っている時には、新たな問題や課題の発見ができる可能性もある。そしてその問題や課題も考慮しながら、リソース確保の再調整を可能にする効果もあるからである。

一般的に、リソースの必要条件は概念設計時に決定されるものであるが、逆にそのリソースという必要条件が最初に決定されているプロジェクトも存在する。リソースの必要条件が最初に前提条件として決まっている場合には、そのリソースを配分する領域の優先順位の分析を行う可能性が高い。もし仮にその分析時点においてリソースを配分する領域が対象として上げられていないなら、そのリソースは配分されないことになる。つまり、優先順位が決定され次第、その順位でリソースが配分される。そこにプロジェクトのリスクは存在していると言える。

なぜならば、上流の概念設計から、実体設計そして詳細設計と具現化が進むにしたがっ

て、上流設計では見過ごされていた問題が顕在化することがあるからである。本来、その問題はリソースを配分すべき領域として予め優先順位に上げていなければならなかったと言える。下流の実体設計そして詳細設計でこの様な事態が生じると、その問題が生じるタスク自身を変更するか、または上流に戻り概念設計を修正するか、場合によっては要求仕様を修正することになる。従ってリソース制約という条件によっては、イタレーションの顕在化の可能性はあると言える。

2.5.3　タスクの所要時間と反復修正の回避思考

概念設計の構想には時間を要する。なぜならば、概念設計時には多様な構想概念が存在する。そして、その構想概念の数量分の分析を行わなければならないからである。その構想をプロジェクトの前段階として早めに着手しておけば、プロジェクトを計画通りに終了させることはより容易になる。しかし概念設計時は、前提となる必要条件を完全に把握することが困難なこと、仮定に基づいて進展せざるを得ない活動が存在すること、プロジェクトの進展過程でどういった中間結果が出るのか不明な点があること等により十分な情報を持ち合わせていない。つまり、概念設計からスタートする設計プロセスでは反復修正の可能性がある。

ここでそこで発生する所要時間が短いタスクと長いタスクの二つ、そして設計者の思考との関係を考えてみる。タスクの所要時間が短いタスクの場合、反復修正を重ねる機会は少なくない。それはスケジュールの遅延に大きな影響を及ぼさないと設計者は考えるからである。一方、タスクの所要時間が長いタスクの場合には、その傾向は逆に少なくなる。それは反復修正による所要時間の総量が大きくなる、又はタスク内の情報経路が長くなり長期化する、すなわち反復修正を回避しようと考えるからである。つまりタスクの所要時間によっては、スケジュールに影響を及ぼすと考える思考を通じて、反復修正に対する牽制が働いていると言える。

しかしタスクの所要時間が短いタスクは反復修正を重ね続けていくと、将来的にタスクの所要時間の長期化に順応してしまう可能性がある。するとこの順応によるタスクの所要時間の増加が、反復修正への牽制を遅らせることにつながる。つまりタスクの所要時間が長いタスクに対しても、抵抗感無く反復修正の行為につながる可能性がある。

技術的視点から見れば、この反復修正による設計成果は最適であるかもしれない。しかしその反面として、コスト増しの発生要因となってしまう。従って、イタレーションの視点から見ればこの行為は問題とすべき点である。イタレーションの視点から見た設計プロセス管理を行うならば、こういった行為を軽視せず、適切な対応につながると言える。

2.5.4　設計プロセスにおける進捗の測定

製品開発における設計初期の工程においては、明確な設計活動基準に基づいて計画され

ることは少ない。また、どういった設計活動が必要かという問いに基づいて計画策定されることも少ない。設計初期の工程では、まず設計上の克服すべき課題点をできるだけ多く集めてから、その課題の対処について検討を行う方法が一般的である。これらの課題は工程の進捗にしたがって克服され、そして徐々に設計の収束へと導かれる。しかしこの様な方法も設計上の問題を生じる可能性がある。その可能性をここで二つ指摘する。

一つ目は、上述のリソースの制約によるイタレーションの顕在化と類似している。類似点は設計上の課題を集める作業過程である。設計プロセスにおいて、設計上の課題を多く集めるという作業は、打ち合わせといった複数の設計者の場で行われる。この場合、打ち合わせの時点において課題対象として取り上げられるべき重要課題が見過ごされているなら、その課題は克服されないことになる。そこにプロジェクトのリスクは存在している。

二つ目は、設計プロセスの進捗を測定する方法に関することである。設計プロセスの進捗を測定する一つの方法としては、設計課題の数が徐々に減少するのを観察することである。しかし設計課題は設計の進展過程にしたがって減少するとは限らない。むしろ課題が減少した分だけ、新たな課題が持ち上がる可能性もある。その場合は、設計課題の数が発振する様に上下する。つまり、設計の収束が導けないことになる。そこにもう一つのプロジェクトのリスクが存在している。

この様に、設計初期の工程で設計上の克服すべき課題をできるだけ多く集め、そしてその課題の対処について検討を行う方法は、必ずしも効果的であるとは言えない。設計中にある課題が克服されると同時に新たな課題が提示される場合、その新しい課題がその時点で最優先事項になる可能性が高い。製品開発を行う設計者や組織はこの状況の存在は理解している。そして最悪事態として手戻り修正を引き起こす可能性があることも理解している。しかし新たに持ち上がった課題によって、手戻り修正が生じる時の対処方法は持ち合わせていない。

次に製品開発における設計工程後期について述べる。製品開発における設計工程の後期においては、設計の進展過程を測定する伝統的な方法は、図面やプログラムの完成度を測定することである。設計関係者の多くは、この測定方法が設計成果物の品質に悪影響を及ぼすことを認識している。設計関係者にスケジュールだけを意識させることは時間軸上の動機付けにはなるが、設計成果物の不完全性を生む可能性があるからである。現実的には新しい図面やプログラムが生成される毎に、大幅な修正が繰り返されることになる。そしてスケジュールの遅延をまねく。しかし修正の繰り返し、すなわちイタレーションは必ずしも全てが悪いというわけではない。イタレーションは設計の改善を目的としているからである。

本節を簡単にまとめると、設計初期の工程において設計上の課題を集め、その課題の対処について検討を行う方法、そして設計後期の工程において設計の進展過程を測定する伝統的な方法について述べた。これらは一般的に利用される実践的な進展過程の測定方法である。これらの測定方法は反復修正、すなわちイタレーションを発生させる可能性が高い。

そしてそのイタレーションの性格は、当事者が予測していないイタレーション、すなわち想定外のイタレーションである可能性が高い。

2.5.5 スケジュールの蓋然性

スケジュールの策定に利用される一般的なツールは、ガント・チャート(Gantt Chart)である。それ以外にも PERT (Program Evaluation and Review Technique）と呼ばれるネットワーク図、そして流れ図(Flow Chart)も利用される。プロジェクト全体の完了を遅らせないためには、CPM (Critical Path Method)を利用する方法がある。CPM は、まず作業の開始日から完了日までで期間を計算することである。次にプロジェクトの完成を遅らせない様に、遅延させてはいけないタスク(task:作業)の集合を抽出する。そして期限まで完了できる様に、タスクの集合と所要期間を検討する。このタスク集合がクリティカル・パス(critical path)であり、そのクリティカル・パスを利用した管理法が CPM である。

管理者等は PERT を活用して、そのクリティカル・パスの検討を行う。しかしその進捗がプロジェクトを通じて把握されることは少ない。その要因の一つは、進捗するにつれてクリティカル・パスにおけるタスク集合要素の組合せが変わるからである。そして二つ目の要因としては、クリティカル・パスは一本ではなく、複数本存在することもあるからである。これらの要因がクリティカル・パスの進捗状況の把握を困難にしている。前述で指摘した通り、PERT は反復修正といったイタレーションの現象を表現できないことも実践での活用を困難にしている。

ガント・チャートはスケジュールを直感的に管理し易い。その為に広く利用されている。しかし欠点もある。ガント・チャートは各タスクの所要期間について実現可能な数値を示しているとは限らない。それぞれの所要期間は実践の現場では、決定論的に一元化できないからである。ガント・チャートが持つ数値は変わりやすいデータであり、これが関係者との調整を難しくしている。末端の設計者は与えられたタスクがスケジュール通りに進行しているかどうかを理解していないなら、その設計者の動機は低減する可能性もある。

策定されるスケジュールは本来、一元化は困難である。ばらつき、すなわちある程度の分散を抱えているのが現実である。この分散は、設計プロセスにおける想定外のイタレーション要因として、影響を及ぼす可能性がある。従って、それはスケジュール・リスクを潜在的に持っているので、リソースの無駄な消費を生む可能性がある。

2.5.6 作業分割と作業範囲記述の限界点

プロジェクト管理でスケジュールを策定する際、WBS (Work Breakdown Structure:作業分割構成) と SOW (Statement Of Work:作業範囲記述書)を準備する。スケジュールは、過去類似したプロジェクト対象物並びにその工程の WBS や SOW を参照して策定されることが多い。つまり過去の事例のデータを参照する方法である。この様に過去の事例だけに依存した方

法が受け入れられてきた理由の一つは、過去の事例を否定する理由が見あたらないからである。

しかし WBS や SOW は、タスク間の設計知識と情報の創造と処理システムという視点で説明することに限界がある。WBS と SOW はタスク間の設計知識と情報の創造と処理という記載を前提としていないからである。従って、この情報の欠如がスケジュールの不確実性を贈出させることにつながる。そしてそれが関係者への調整を困難にするので、イタレーションを誘発する。場合によってはイタレーションを加速させる可能性がある。WBS や SOW だけに依存したスケジュール策定では、こういった潜在的な問題を引き起こす可能性がある。この問題は実践現場では十分周知されているとは言えない。WBS と SOW に限った管理は、イタレーションを引き起こす源泉の一つであることを理解すべきと言える。

2.5.7　イタレーションの分類と混同

上記の通り、イタレーションを管理するという視点で現行のプロジェクト管理の枠組みの限界点を議論した。この限界点を越えた問題を解決できる手法として DSM (Design Structure Matrix: 設計構造マトリックス)が提唱されてきた。従来の設計手法を活用しつつ、この管理手法を補完的に利用することで企業の技術経営に役立つと言える。

DSM の初期のモデルは、イタレーションを正確に反映することが困難であった。しかし現在まで様々な研究が重ねられ、各種モデルの構築が行われている。設計工程を検討する技術者がイタレーションについての知見を持っているならば、本論で提示したモデルの活用は可能と言える。

モデルの構築にあたり注意すべきことはイタレーションの適用範囲である。想定内のイタレーションを反映させることは容易である。しかし予測も検討もしていないイタレーションの存在がある限り、想定内のイタレーションだけで工程設計することは危険である。つまり予測可能なイタレーションはモデルに反映できる。しかし、予測も検討もしていないイタレーションについてはその発生の源泉を探求することが欠かせない。この探求作業は設計プロセスの進捗過程を把握する際、そして設計成果物の進展過程の検証の際に洞察できる可能性がある。

本項の最後に指摘しておくべきことは、イタレーションの選別である。つまり、イタレーションを抽出してから、それらを二分化することが求められる。そのイタレーションには、設計の効率と改善に有益なものと無益なものに分類できるからである。イタレーションの管理の視点では、無益なイタレーションは全て除去する努力も欠かせないと言える。

2.5.8　イタレーションが見過ごされてきた理由

前述の通り、先行研究の論文で提示されている理論や事例の渉猟を行った。そして半導体設計における実践現場を通じて観察されたイタレーションは、先行研究の中で指摘され

ている特徴と類似している。しかしながら実践のプロジェクトの策定と管理において、イタレーションの重要性を見過ごしたり抑圧したりする場合がある。その理由を、半導体設計の実践的視点で観察した結果として以下に簡単に述べる。

i) 設計に従事する関係者全てが、設計プロセスで生じるイタレーションの存在自体を認識しているとは限らないからである。その存在を認識していない関係者には、設計プロセスは常に順方向に進展するものと解釈している者もいる。つまり、設計プロセスでイタレーションの問題が生じたとしても、それはイタレーションとして識別せずに、単なる個別の一問題として見過ごしてきたからである。
ii) イタレーションは全て外部環境の外乱によって引き起こされるものと認識している者が、設計者や管理者の中にいるからである。この場合の多くは、想定内のイタレーションと想定外のイタレーションを混同している。
iii) イタレーションを設計プロセス内で順方向に進展する一連のタスクの一つと認識している者が、設計者や管理者の中にいるからである。つまりスケジュール策定が厳密に確定されている条件のもと、イタレーションのタスク、イタレーションの一周期の所要時間、イタレーションの回転の数が先付けで決定されているプロジェクトでその状況が観察される。
iv) 設計プロセスの開始前に設計知識や情報の交換、すなわちイタレーションが十分行われ、既にイタレーションは終了していると想定している者が、設計者や管理者の中にいるからである。

上記の通り、半導体設計の実践プロジェクトを通じて観察されたイタレーションの軽視理由について幾つか提示した。この状況を変えるにはイタレーションの重要性を認識させる手段を講じることである。以下にプロジェクトの関係者に向けた対策手段を提示する。その対策手段の一つは関係者への啓蒙と周知徹底である。

- プロジェクト管理におけるタスクは一般的には、設計作業の開始と完了時期、作業範囲、リソースの配分で規定されている。タスクはその規定だけではなく、生産者と消費者間の設計知識並びに設計情報も含み入れていることを関係者に徹底周知させること。
- 設計知識と設計情報の交換、すなわちイタレーションが設計プロセス内で生じていることを関係者に徹底周知させること。
- 前述で議論した通り、イタレーションは想定内のイタレーションと想定外のイタレーションの二つに分類することできる。この二つの存在と性格の違いについて関係者へ啓蒙すること。
- 想定外のイタレーションの因果連鎖を探索する、そしてその想定外のイタレーションを引き起こす源泉を発見する様、関係者に徹底周知させること。

・ 想定内のイタレーションの効用は、設計品質や製品性能の向上につながることを関係者に徹底周知させること。

2.6　その他の設計負荷：人間の行動特性と設計知の活用の不十分性

設計負荷の要因はイタレーションだけではない。人間の行動特性や、設計知の活用の不十分性という要因が、設計プロセスに影響を与える可能性がある。本研究ではプロジェクトで観察されるこの二つの要因について言及する。この二つの設計負荷問題を解決する管理手法については、本論の第 7 章、第 8 章でそれぞれ述べる。

2.6.1 人間の行動特性

一般的にプロジェクト・マネージャは、現場の各設計技術者に対して目標となるタスクの納期を通知する。Goldratt (1997)は、各設計技術者がこの納期だけに注意を払いすぎることが無駄な時間を消耗させ、遅延を起こす原因であると指摘している。そしてその原因を人間行動の特性に関連づけし、その根拠を四つの因子として上げている。その四つの因子を以下に述べる。

i) 学生症候群

目標値が決定される時には、管理上、安全な余裕時間が付け加えられる。従って、タスクを実行する者は自分なりに所要期間を再評価し、最尤値として平均値を採択する。すると結果として本タスクとは別のタスクに手をつけてしまう。これは安全な余裕時間を含んでいると解釈しているからである。そしてぎりぎりになるまで着手せず、最終的には一夜漬けの集中作業となる。その時にはマーフィーの法則(Bloch, 1993)に従った「予期しない問題」に打ちのめされ、納期が遅延する。たとえ突貫作業で間に合わせたとしても、逆にタスクの品質又は成果物が悪くなってしまう可能性が残る。これが学生症候群である。

ii) 自己の予防線

タスクが予定より早めに終了した場合に起こる症状が自己の予防線である。この場合には、その終了したという事実はプロジェクト・マネージャに申告されにくい。すなわち自己防衛が働き、自己に対しての予防線を張る傾向になる(津曲 et al., 2004)。なぜならば正直に申告すると次回のプロジェクトで作業者は不利になると考えるからである。つまり作業が早く終了して浮いた時間は無駄に消費されるだけで、プロジェクト全体に反映されない。

iii) マルチタスキング

プロジェクトでは掛け持ち作業は頻繁に見られる。しかしマルチタスクとは単なる掛け持ち作業ではない。マルチタスクとは作業の実践者に対して複数の作業を同時進行させることである(Goldratt,1997)。この場合、それら掛け持ち作業の中での優先権が明確にされていないことが多い。結果として一作業当たりの実践者の生産性は低下する。これが納期遅延を招来する。

マルチタスクは納期遅延だけでなく、創造的生産が要求される作業の能力も低下させるという指摘(Fujimoto et al., 2003)もある。しかし特定の専門知識やノウハウを要求するリソースが限られている場合には、このリソース限定問題を解消できるアプローチであり、有効な手段と考えられている。実際、複数の組織と複数のプロジェクトが交錯した実践的マトリクス組織(芝尾, 1999)の適用の場合にはマルチタスキングは必要条件となっている。
マルチタスクのもう一つの問題は、掛け持ちのタスクが同時に生じる場合である。これをリソースの競合と言う。この問題は一般的なプロジェクト管理の枠組みのクリティカル・パス手法では解決しにくい。

iv) パーキンソンの法則

作業者がいつまでも作業を続け、計画した作業期間いっぱいまで延びることがある。すなわち計画予算は全て使い尽くすまで消費する傾向がある。これをパーキンソンの法則と言う(Parkinson, 1996)。パーキンソンは、作業の量は一定ではなく、いかようにも変動する、従って業務効率化などというのがいかに空疎であるかを述べている。つまり、この法則に基づき組織成長が拡大する。そして組織が拡大すれば、組織内での調整作業が幾何級数的に増大するということである。必要不可欠な作業は計画予算を全て使い尽くすか否かによらず実行されなければならない。むしろ不必要な仕事はいくらでも作り出せるという組織内力学が、組織の拡大を続けていることを示唆している。

2.6.2　設計知の活用の不十分性

i)　外部組織の設計知の導入と設計知の再利用

半導体の製造は資本集約型の産業構造となっている。この産業構造の変革過程の中で、半導体の付加価値の源泉は設計知に移ってきている。この設計知とは、設計の情報と知識である。

日本の半導体メーカでは、これらの設計知が組織内に断片して存在していることが多い。それらの設計知は適切な時間で適切な場所に配置されていないとか、組織構造上の問題で見えない壁によって共有化されてこなかったからである。また知の場では異質の知を排除する可能性が多分にあり、それでは新たな知の形成が起きない。結果的に多くの設計知が

意識的に、そして組織的に活用されないまま埋もれてしまってきた。その他にグループ企業内の生産と消費サイクルの影響が考えられる。日本の半導体メーカの供給先の多くはグループ内のコンピュータや民生機器といったシステム部門であった。つまり、設計・生産・消費のサイクルがグループ内で閉じられていた。従って、日本の半導体メーカは設計プロセス上で、設計知を外部組織から導入することや設計知を再利用することを積極的に行ってこなかったと言える。つまり、設計知の活用の不十分性が相対的に、設計プロセスの効率化と改善に対して負の影響を与えていると考えられる。

そこで本論では、この設計知の活用の不十分性を設計プロセス上の設計負荷の一要因と捉える。そして設計知の活用の不十分性を是正する一手段は、設計知の概念化と体系付けに基づいて構築される知識ベースとその定着と考えた。設計プロセス上の設計知の概念化と体系付けに基づいて構築される知識ベースが、設計プロセスにおいて定着されるのであるならば、設計負荷は低減するからである。つまり知識ベースの構築と定着を通じて、設計知の共有化と再利用化が進み、設計プロセスの所要時間とコストは低減できる。更にロバスト設計が達成されるので、設計負荷は低減するからである。

ii)　半導体設計における知識ベースの構築と効果

本論で提案するこの知識ベースは、半導体を対象として構築したものである。しかしこの知識ベースの対象を一般化していけば、他の工業製品への応用は可能である。

ここで次にその半導体を対象とした知識ベースについて述べる。まず、設計手法の変遷過程を経て定着してきた設計知の概念を、設計の抽象性レベルの違いによって構築する。その設計知の概念は、設計知を著作物とさせた知的財産、すなわち半導体 IP (Intellectual Property) である。それらは抽象度の高い概念、設計対象物という実体に近い概念、そしてその二つの中庸に位置する概念である。この概念と体系付けに基づいて構築される知識ベースは、従来の知識ベースが抱える共有・再利用性の欠如という問題点の解消に貢献すると言える。なぜならばこの知識ベースが定着すれば共有化が進み、設計過程において新規からの設計作業を減らす可能性を持つからである。そして再利用化が進めば、設計開発コストの低減と設計開発時間の短縮につながるからである。

知識ベースは設計過程の効率化だけではない。知識ベースの活用は共有化と再利用を通じて継続されるので、機能や性能面の設計品質が安定してくる。つまりロバスト設計である。このロバスト設計ができておれば、設計開発工程のプロジェクト・リスクを低減することが可能となる。大規模設計の様に扱う知識の量が膨大で、複数の設計者が介在する場合においては、設計品質を確保する手段として体系化された知識支援の効果は大きいと言える。

2.7　PDCA サイクルによる設計品質と設計の生産性の向上

PDCA サイクルとは業種に関わらず、経営レベルから現場レベルまで適用可能な継続的改善を推奨・推進する管理手法である。そのプロセスは、i) Plan（計画）、ii) Do（実行）、iii) Check（評価）、iv) Action（改善）という四つの手順を踏む実行サイクルから構成されている。

この管理サイクルの歴史的出発点は品質管理である。Shewhart (1951)そして Deming (1960)等が提案した品質管理のサイクルは、まず生産プロセスの中で改良や改善を必要とする部分を特定・変更できる様に、プロセスを測定・分析することである。次にそれを継続的に行うため、連続的なフィードバック(feedback)・ループ(loop)を含み入れた。この継続的な連続手順が改善プロセスである。

Plan と Do という二つの手順だけで構成される原始的な管理手法と比較するなら、この改善プロセスには、Check と Action が付け加えられている。この二つの手順が付け加えられたことで、管理上二つの有益な効果が得られることとなった。それは Plan から Do という手順の他に、Check から Action という手順による改善を念押しする継続的なサイクルになっていることである。もう一つは Check から Action という手順によって、目標からの乖離を抑制する作用を働かせるというものである。

PDCA サイクルは、品質改善に向けた手法だけに利用されているのではない。PDCA サイクルは生産性の向上にも大きく貢献している。その例証はトヨタ生産方式（大野, 1978）から体系化されたリーン思考(Womack et al., 1991)と TOC (Theory of Constraints:制約理論)（Goldratt, 1997）である。ここで生産性の評価尺度は稼働率と達成率で混同する可能性があるが、稼働率は適切とは言えない。例えば、稼働率を上げて納品できない在庫を創ったとしても、企業経営にとって有益なこととは言えない。ある工程だけの効率だけを高めたとしても、顧客への出荷がなければ、在庫管理コストは上昇してしまうからである。

一方、達成率の考え方は、必要とされる時に、必要とされるものを、必要なだけ作り出すことである。つまり、Check から Action という手順が存在することによって、生産活動の無駄な消費は避けられる。そして目標に沿った行動にだけ生産活動は活用されることになる。

PDCA サイクルを導入することにより、今まで見落とされてきた本質的でない作業、すなわち無駄が取り除かれる。故に生産性が向上する。このように PDCA サイクルは、品質と生産性を同時に向上させることが可能な管理手法である。

ここで次になぜ設計プロセス管理に PDCA サイクルのプロセスが有効なのか理由を考えてみる。その理由の一つは、生産性は製品の生産や製造のみならず、設計と言う行為に対しても当てはめることが可能だからである。つまり、PDCA サイクルを繰り返すことにより、不必要な設計負荷並びに設計プロセス上の無駄を取り除き、設計品質と設計の生産性向上に貢献できるからである。

二つ目の理由は、半導体という工業製品を取り巻く外的影響力に起因しているからであ

る。この外的影響力には技術的要因と市場的要因がある。技術的要因には、半導体プロセス技術の進歩、設計技術の改革、ビジネス・モデルの変化等が含まれる。そして市場的要因には、生産者間での激しい競争、短い商品のライフ・サイクル、単納期化等が含まれる。つまり、半導体を取り巻く環境は変化が激しく、外的影響力の働きは大きいと言える。従ってこの外的影響力を克服する一つの方法は、たゆまぬ改善、すなわち設計プロセスの改善を止めないことである。改善を止めることは相対的に競争から立ち後れることを意味するからである。PDCA サイクルを用いた設計プロセス管理は、企業を取り巻くこの外的影響力に対して有効な企業戦略の一つと言える。

以上をまとめると、設計プロセス管理に PDCA サイクルを適用することは、不必要な設計負荷や無駄を取り除き、設計品質と設計の生産性の向上に貢献すること、そして変革の激しい環境と大きな外的影響力に対する企業戦略の一つとして考えられることから、効果が高いと言える。

2.8　本章のまとめ

本章ではまず製品開発における設計プロセスの一般概念を構築した。この一般設計プロセスとは、要求仕様から、概念設計、実体設計、詳細設計を経て、最終的に設計解を決定するまでの人間の知的行為である。

次に設計プロセスに対する設計負荷の主たる因子であるイタレーションとそのモデルについて議論した。イタレーションに関する先行研究から得られた一つの知見は、イタレーションの発生メカニズムである。設計プロセスで新しい情報を取得したことによるもの、設計と設計検証の繰り返しによるもの、設計を改善させようとする行為によるもの、設計解を求めようとする行為の繰り返しによるものが指摘されている。

二つ目の知見はイタレーションの分類とその分類によって進める効率化の方法である。イタレーションは想定内のイタレーションと想定外のイタレーションの二つに分類することできる。これが効率化の管理に向けた第一歩である。想定外のイタレーションについては、まずそれを明確にし、そのイタレーションを低減させる手段を講じることである。想定内のイタレーションに関しては、そのイタレーションの回転速度を上げること、イタレーションの回数を低減させることである。これが効率化につながる。

次にイタレーション・モデルに関してまとめる。上述の先行研究から認識できることは、DSM 手法が現在のところ、最適な手法であるということである。モデル自身は完全な精度を保有しているとは言えないが、実現象に基づいて構築されているからである。そして実際に多くのプロジェクトにおいて有益な情報を提供しているからである。その有益な情報とは適切な意思決定に結びつく定量的な情報のことである。

更に現行のプロジェクト管理の拡張の必要性についても議論した。半導体設計の設計プロセス上の問題全てを現行の管理手法で解決することは難しいからである。それは上記で

議論したイタレーションの問題だけではない。それ以外にも限界はある。それらは既存手法によってもたらされる心理的抑圧、リソース制約、回避思考、進捗測定、蓋然性等である。こういった認識できる範囲についても、現行のモデルや手法を拡張させる必要があると言える。

設計負荷はイタレーションだけではない。それ以外にも、人間の行動特性によるものと設計知の活用が不十分であることも含まれる。この二つの要因については、要因を更に分析しまとめている。

最後に設計プロセス管理における継続的改善、すなわち管理サイクルについて述べた。この管理サイクルはたゆまぬ改善である。変革の激しい環境や大きな外的影響力を受ける工業製品に対しては、この管理サイクルは製品開発に有効である。

本章で議論した設計負荷を十分解釈し、それらを考慮したプロジェクト管理が提供されるのであれば、現行の設計プロセスを再設計することができる。そしてその管理手法を自ら果たすことができるのであれば、設計対象の製品性能を維持しながら、設計開発の所要コストと所要時間を同時に削減することに挑むことができる。つまり、設計プロセスの効率化と改善に向けたチャレンジにつながる可能性がある。

この設計プロセスの効率化と改善を実現させるという思考は、企業の競争力を強めるという意味で大変意義のあることである。言い換えると、企業の競争力の背後には、設計プロセス効率化と改善に向けた問題提示能力、そしてその問題解決とその対処能力があるとも言える。設計開発の所要コストと所要時間に影響を及ぼす要因である設計負荷、すなわちイタレーション、人間の行動特性、設計知の活用を、設計プロセスに反映することは、製品開発を行う企業にとって有益なことである。

参考文献

[1] Ahmadi, Reza H. and Hongbo Wang: “Rationalizing Product Design Development Processes”, UCLAAnderson Graduate School of Management, Report (1994)

[2] AitSahlia, Farid, Johnson, Eric, and Will, Peter: “Is Concurrent Engineering Always a Sensible Proposition? ”, IEEE Transactions on Engineering Management 42(2), pp. 166-170 (1995)

[3] Alexander, Christopher: “Notes on the Synthesis of Form”, Cambridge, MA, Harvard University Press (1964)

[4] Alford, Mack: “Issue Driven Design: Planning and Managing Design Iterations”, Proceedings of the Fourth Annual International Symposium of NCOSE, San Jose, CA, Aug. pp. 10-12. (1994)

[5] Altus, Stephen S., Kroo, Ilan M., and Gage, Peter J.: “A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems”, Proceedings of the 21st ASME Design Automation Conference, Boston, Sept. (1995)

[6] Basili, V. R. and Turner, A.: “Iterative enhancement, a practical technique for software development requirement”, IEEE Transactions on Software Engineering, 1(4) (1975)

[7] Belhe, Upendra and Kusiak, Andrew: “Modeling Relationships Among Design Activities”, Journal of Mechanical Design 118, pp. 454-460 (1996)

[8] Blanchard, Benjamin S. and Fabrycky, Wolter J.: “Systems Engineering and Analysis Second Edition”, Englewood Cliffs, NJ: Prentice Hall (1990)

[9] Bloch, Arthur (倉骨彰訳): “マーフィーの法則”, アスキー出版社 (1993)

[10] Boehm, B. W.: “Software Engineering Economics”, Prentice Hall (1981)

[11] Boehm, B. W.: “A spiral model of software development and enhancement”. IEEE Computer, May (1988)

[12] Browning, Tyson R.: “Integrative Mechanisms for Multiteam Integration: Findings from Five Case Studies”, Systems Engineering 1(2), pp. 95-112 (1998a)

[13] Browning, Tyson R.: “Use of Dependency Structure Matrices for Product Development Cycle Time Reduction”, Proceedings of the Fifth ISPE International Conference on Concurrent Engineering: Research and Applications, Tokyo, Japan, July, pp. 15-17 (1998b)

[14] Carrascosa, Maria, Eppinger, Steven D., and Whitney, Daniel E.: “Using the Design Structure Matrix to Estimate Product Development Time”, Proceedings of the ASME Design Engineering Technical Conferences (Design Automation Conference), Atlanta, GA, Sept. pp. 13-16 (1998)

[15] Cesiel, Douglas S.: “A Structured Approach to Calibration Development for Automotive Diagnostic Systems Unpublished Master's Thesis (Mgmt./E.E.) ”, Massachusetts Institute of Technology, Cambridge, MA (1993)

[16] Clark, Kim B. and Fujimoto, Takahiro: “Product Development Performance: Strategy, Organization, and Management in the World Auto Industry”, Boston: Harvard Business School Press (1991)

[17] Clark, Kim B. and Wheelright, Steven C.: “Managing New Product and Process Development”, New York: Free Press (1993)

[18] Clausing, Donald: “Total Quality Development: A Step-by-Step Guide to World-Class Concurrent Engineering”, New York: ASME Press (1994)

[19] Deming, W. E. (坂元平八監訳): “品質管理の基礎概念：品質管理の観点からみた統計的方法”, 岩波書店 (1960)

[20] Denker, Stephen and Steward, Donald: “Using the Design Structure Method”, Problematics, Unpublished Informational Report (1995)

[21] Denker, Stephen and Steward, Donald: “Using Information Exchange Structure to Design for Concurrent Design”, Problematics, Unpublished Informational Report (1996a)

[22] Denker, Stephen and Steward, Donald V.: “Using Concurrent Engineering for Competitive Advantage”, Problematics, Unpublished Informational Report (1996b)

[23] DoD: Systems Engineering/Configuration Management Life Cycle Application DoD, Military Handbook MIL-HDBK-499-3 (1992)

[24] DoD: MIL-STD-499B Systems Engineering (Draft) DoD, Military Standard Specifications MIL-STD-499B (1994)

[25] Eisenhardt, Kathleen M. and Tabrizi, Behnam N.: “Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry”, Administrative Science Quarterly 40(Mar.): pp. 84-110 (1995)

[26] Eppinger, Steven D., Whitney, Daniel E., Smith, Robert P. and Gebala, David A.: “A Model-Based Method for Organizing Tasks in Product Development”, Research in Engineering Design 6: pp. 1-13 (1994)

[27] Eppinger, Steven D.: “Three Concurrent Engineering Problems in Product Development”, Presentation, Cambridge, MA (1995)

[28] Eppinger, Steven D.: Nukala, Murthy V. and Whitney, Daniel E., “Generalised Models of Design Iteration Using Signal Flow Graphs”, Research in Engineering Design 9: pp. 112-123 (1997)

[29] Eppinger, Steven D.: “Innovation at the Speed of Information”, Harvard Business Review, Vol.79, No.1, pp. 149-158 (2001)

[30] Ford, David N.: “The Dynamics of Project Management: An Investigation of the Impacts of Project Process and Coordination on Performance Unpublished Ph.D. Thesis (C.E.)”, M.I.T., Cambridge, MA (1995)

[31] Fredriksson, Billy: “Holistic Systems Engineering in Product Development”, The Saab-Scania Griffin (Nov.), pp. 23-31 (1994)

[32] Fujimoto, Hideo, Firdausi, Manti M., and Chen, Lian-Yi: “Applying the TOC Project Management to Operation and Maintenance Scheduling of a Research Vessel”, Series C, Vol.46, No.1, JSME International Journal (2003)

[33] Gebala, David A. and Eppinger, Steven D.: “Modeling the Impact of Organizational Structure on Design Lead Time and Product Quality”, MIT Sloan School of Management, Working Paper No. 3301 (1991)

[34] Goldratt, Eliyahu M.: “Critical Chain”, The North River Press (1997)(エリヤフ・ゴールドラット (三本木亮訳), “クリティカル・チェーン”, ダイヤモンド社 (2003))

[35] Gomma, H. and Scott, D. B. H.: “Prototyping as a tool in the specification of user requirements”, Proceedings of the 5th International Conference on Software Engineering, pp. 333-341 (1981)

[36] Grose, David Lee: “Reengineering the Aircraft Design Process”, Proceedings of the Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, Sept., pp.7-9 (1994)

[37] Koushik, Murlidhar V. and Mookerjee, Vijay S.: “Modeling Coordination in Software Construction: An Analytical Approach”, Information Systems Research 6(3), pp. 220-254 (1995)

[38] Koushik, Murlidhar V. and Mookerjee, Vijay S.: “Modeling Coordination in Software Construction: An Analytical Approach”, Information Systems Research 6(3), pp.220-254 (1995)

[39] Krishnan, Viswanathan, Eppinger, Steven D. and Whitney, Daniel E.: “A Model-Based Framework to Overlap Product Development Activities”, Management Science 43(4), pp. 437-451 (1997)

[40] Kusiak, A., Larson, N. and Wang, J.: “Reengineering of Design and Manufacturing Processes”, Computers and Industrial Engineering, Vol. 26, No. 3, pp. 521-536 (1994)

[41] Loch, Christoph H. and Terwiesch, Christian: “Communication and Uncertainty in Concurrent Engineering”, Management Science 44(8), pp. 1032-1048 (1998)

[42] Marca, David A. and McGowan, Clement L.: “Structured Analysis and Design Techniques”, McGraw-Hill (1988) (マーカー・D (研野和人監訳,漆島正人,谷岡雄一,研野和人,松本巖訳), “IDEF0/SADT ビジネスプロセスとエンタープライズ モデリング”,社会経済生産性本部 (1998))

[43] Moore, Laurence J. and Clayton, Edward R.: “GERT Modeling and Simulation: Fundamentals and Applications”, Petrocelli/Charter (1976)

[44] 森竜雄: “PERT／新しい仕事のまとめ方”,日本能率協会 (1964)

[45] 森竜雄: “続 PERT／効率的な応用のしかた”,日本能率協会 (1965)

[46] Neumann, Klaus and Steinhardt, Ulrich: “GERT networks and the time-oriented evaluation of projects Berlin”, Springer-Verlag (1979)

[47] Neumann, Klaus: “Stochastic project networks : temporal analysis, scheduling and cost minimization”, Springer-Verlag (1990)

[48] 大野耐一: “トヨタ生産方式－脱規模の経営をめざして”,ダイヤモンド社 (1978)

[49] 小川進: “イノベーションの発生論理”,千倉書房 (2001)

[50] Pahl, G. and Beitz, W.: “Engineering Design, a systematic approach”, Springer-Verlag, Berlin, Heidelberg (1977)（設計工学研究グループ訳, “工学設計：体系的アプローチ”,倍風館 (1995）

[51] Parkinson, C. Northcorte (森永晴彦訳): “パーキンソンの法則”,至誠堂 (1996)

[52] Peters, Thomas: “The Mythology of Innovation, or A Skunkworks Tale, Part II in Tushman, Michael L. and William L. Moore”, Ed. Readings in the Management of Innovation, New York: HarperBusiness, pp. 138-147 (1983)

[53] Pritsker, A. and Alan, B.: “Modeling and analysis using Q-GERT networks”, John Wiley & Sons Wiley (1977)

[54] Pugh, Stuart: “Total Design: Integrated Methods for Successful Product Engineering”, Reading,

MA: Addison-Wesley (1991)

[55] Rogers, James L.: “A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem”, NASA, Technical Paper TP-2903 (1989)

[56] Rogers, James L.: “DeMAID/GA User's Guide—Design Manager's Aid for Intelligent Decomposition with a Genetic Algorithm”, NASA, Technical Manual TM-110241 (1996a)

[57] Rogers, James L.: “Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes”, NASA, Technical Manual TM-110247 (1996b)

[58] Rodgers, James L. and McCulley, Collin M.: “Integrating a genetic algorithm into a knowledged-based system for ordering complex design progress”, NASA TM-110247 (1996)

[59] Safoutin, Michael J. and Smith, Robert P.: “The Iterative Component of Design”, Proceedings of the IEEE International Engineering Management Conference, Vancouver, Aug. pp. 18-20 (1996)

[60] Shewhart, W. A. (白崎文雄訳): “工業製品の経済的品質管理”,日本規格協会 (1951)

[61] 芝尾芳昭: “プロジェクト・マネジメント革新”,生産性出版 (1999)

[62] Shishko, Robert et al.: “NASA Systems Engineering Handbook”, NASA (1995)

[63] Singh, Kamar J., Erkes, Joseph W., Czechowski, Joseph, Lewis, John W. and Issac, Maurice G.:, “DICE Approach for Reducing Product Development Cycle”, Proceedings of the Worldwide Passenger CarConference and Exposition, Dearborn, Mich., Sept. 28- Oct. 1 (1992)

[64] Smith, Robert P. and Eppinger, Steven D.: “Identifying Controlling Features of Engineering Design Iteration”, Management Science 43(3), pp. 276-293 (1997a)

[65] Smith, Robert P. and Eppinger, Steven D.: “A Predictive Model of Sequential Iteration in Engineering Design”, Management Science 43(8), pp. 1104-1120 (1997b)

[66] Smith, Robert P. and Eppinger, Steven D.: “Deciding Between Sequential and Parallel Tasks in Engineering Design”, Concurrent Engineering: Research and Applications 6(1), pp. 15-25 (1998)

[67] Steward, Donald V.: “The Design Structure System: A Method for Managing the Design of Complex Systems”, IEEE Transactions on Engineering Management 28(3), pp. 71-74 (1981a)

[68] Steward, Donald V.: “Systems Analysis and Management”, Structure, Strategy, and Design, New York, PBI (1981b)

[69] 菅野孝男: “ソフトウエア開発のマネジメント”,新紀元社 (2003)

[70] Susman, Gerald I.: “Ed. Integrating Design and Manufacturing for Competitive Advantage”, New York, Oxford University Press (1992)

[71] 鈴木五郎: “システム LSI 設計入門”,コロナ社 (2003)

[72] Taylor, Bernard W. III and Moore, Laurence J.: “R&D Project Planning with Q-GERT Network Modeling and Simulation”, Management Science 26(1), pp. 44-59 (1980)

[73] 津曲公二,酒井昌昭,中憲治: “これならできる　クリティカル・チェーン”,ダイヤモンド

社 (2004)

[74] Ulrich, K. T. and Eppinger, S. D.: “Product Design and Development-3rd ed.”,, McGraw-Hill/Irwin (2004)

[75] von Hippel, Eric: “Task Partition: An Innovation Process Variable”, Research Policy Vol 19, pp. 407-418 (1990)

[76] von Hippel, Eric: “Sticky Information and the Locus of Problems Solving: Implication for Innovation”, Management Science 40, No.4, pp. 429-439 (1994)

[77] Wheelwright, Steven C. and Clark, Kim B.: “Revolutionizing Product Development:Auantum Leaps in Speed, Efficiency, and Quality”, New York, Free Press (1992)

[78] Whitney, Daniel E.: “Designing the Design Process”, Research in Engineering Design 2, pp. 3-13 (1990)

[79] Womack, J. P., Roos, D. and Jones, D. T.: “The Machine that Changed the World”, Rawson Associates, New York (1991)(J・P・ウォマック,D・ルース,D・T・ジョーンズ（沢田博訳): “リーン生産方式が，世界の自動車産業をこう変える”,経済界 (1991))

[80] Yassine, Ali and Braha, Dan: “Four Complex Problems in Concurrent Engineering and the Design Structure Matrix Method”, Concurrent Engineering Research & Applications, Vol. 11, No. 3. Sept. (2003)

[81] 吉川弘之,冨山哲男: “設計学：ものづくりの理論”,放送大学教育振興会 (2000)

[82] Yu, Tian-Li, Goldberg, D. E., Yassine, Ali and Chen, Y.-P.: “A Genetic Algorithm Design Inspired by Organizational Theory”, Jan., Genetic and Evolutionary Computation Conference (GECCO) 2003. Chicago, Illinois, USA. Lecture Notes in Computer Science, Vol. 2724 / 2003, pp. 1620-1621, Springer-Verlag, Heidelberg, ISSN 0302-9743 (2003)

第3章　　設計プロセスにおける状況認識と不確実性の源泉

3.1　　製品開発のリスクと因果連鎖

製品開発においては、設計欠陥と外乱要因に配慮したとしても設計者は依然として情報の欠落と評価の不確実性に悩まされることになる。なぜならば、理論的解析や実験的解析だけで全てを網羅することができないからである。そういった注意深いアプローチを行ったとしても、仕様で設定した機能を間違いなく実現できる設計解を選択しているのかどうか、あるいは経済的な仮定が急速に変化する市況のもとでその設計解は現在でも正しいのかどうか、といった疑問が残ってしまう。

製品開発には他の煩悶も存在する。例えば、予め目標設定していた設計コストで終了するのか、あるいは予定納期に間に合うのかといった問題等である。つまり製品開発においては、技術的あるいは経済的な様々な理由により不確実性を持っている。従って製品開発はリスクを内包していると言える。

そこで本章では、製品開発における設計プロセス上のリスクの源泉を探求する。なぜならばリスクの源泉の把握は、不確実性の減少に向けた出発点であるからである。そして引き続きリスク管理が徹底されるなら、設計プロセスの改善につながるからである。本章においては製品開発のリスクの源泉を把握することで、製品開発における洞察力を高めることがねらいである。

本章ではまず設計プロセスのリスクの分類を行う。次に分類された領域の中で不確実性を引き起こす因子、すなわちリスクの源泉を特定する。そしてそれぞれの因子間の因果関係を導出する。そこでは因果連鎖ダイアグラム(五百井, 1990; 五百井 et al., 1997)を活用して、因果関係を解明する。尚、因果連鎖ダイアグラムは本論で適用した半導体製品開発の実プロジェクトの観察を元に、先進法と後退法(五百井, 1993)によって構築したものである。

3.2　製品開発のリスクと設計プロセス

3.2.1 製品開発におけるリスクの基本要素

ここでリスクの分類について述べる。製品開発におけるリスクの分類については様々な研究者によって行われている(Boehm, 1989; Brekka, 1994; Hall, 1998; Hoy, 1996; Justice, 1996; Michaels, 1996; Reinertsen, 1997; Roberts, 1996; Sage, 1992; Sarbacker, 1997; Shishko, 1996; Smith, 1991; Souder, 1993)。これらの共通事項を整理してまとめると、スケジュール・リスク(schedule risk)、コスト・リスク(cost risk)、性能リスク(performance risk)、技術リスク(technology risk)、市場リスク(market risk)、ビジネス・リスク(business risk)に分類される。

現実の世界では、この六つのリスクの中から幾つかのリスクが顕在化したとしても、残りのリスクの顕在化を防ぐことはできる。しかしこれらリスク全ての顕在化を同時に防ぐことは困難な取り組みである。つまり、これらリスク全ては明らかに独立ではない。従って他のリスクとの従属性についての付帯条件がなければ、リスクの低減自身は意味をなさなくなる可能性がある。そこでその付帯条件を踏まえて、本章の製品開発におけるリスクについて以下に定義しておく。

i) 計画納期リスク

計画納期リスクは、他のリスク要因からの影響を受けない条件のもと（すなわち、仕様で規定した製品能力を間違いなく実現でき、かつ設計開発コストを予算以内に実現できる前提条件のもと)、設定した納期以内に設計解を導出できるかどうかという不確実性を包含する。そして、この確率的な不確実性度合いだけでなく、納期以内に設計解が導出できない場合の経済的損失量も包含する。例えば、計画納期からの誤差による遅延が大きくなれば、遺失利益は大きくなる。この遺失利益額はリスクの一情報成分である。

ii) 計画コスト・リスク

本章の計画コスト・リスクは製品開発の所要コストのリスクである。このコスト・リスクは、他のリスク要因からの影響を受けない条件のもと（すなわち、仕様で規定した製品能力を間違いなく実現でき、かつ計画納期以内に実現できる前提条件のもと)、設定したコスト予算以内で設計解が導出できるかどうかという不確実性と、コスト予算以内に設計解が導出できない場合の経済的損失量を包含する。計画コストが上回れば、その増加分だけ確実に経済的負担は大きくなる。その経済的負担額がリスクの経済的損失量である。

iii) 計画性能リスク

計画性能リスクは、他のリスク要因から影響を受けない条件のもと（すなわち、目標設

定した納期以内に間違いなく実現し、かつ設計開発コストを予算以内に実現できる前提条件のもと)、仕様で規定した製品能力を実現した設計解を導出できるかどうかという不確実性と、その設計解を導出できない場合の経済的損失量を包含する。規定仕様に満たない製品能力が製品の競争力を低下させるなら、遺失利益が生じてしまう可能性がある。この遺失利益額が経済的損失量である。

本章における分類に従えば、計画性能リスクの計画性能とは仕様で規定された製品能力を意味する。付帯条件を加味すると、その製品が投入される時期が適切であり、そして製品の売値が適切であるという前提にたっている。なぜならばそれらは性能リスクに影響を与えるからである。

iv) 技術リスク

技術リスクは、仕様で規定した性能以外の技術に関するリスクである。提供される技術は前もって入手していた技術情報通り一致しているのかどうか、提供される技術は本当に実現可能な設計解を選択しているのかどうか、あるいは技術的な仮定が急速に変化する市況のもとでその設計解は現在でも正しいのかどうかという不確実性である。技術リスクはそれが顕在化したときの経済的損失量も包含する。急速に技術が進展する環境に置かれる製品の場合に、この不確実性は高くなる。技術リスクの顕在化が製品の競争力を低下させるなら、遺失利益が生じてしまう可能性がある。

v) 市場リスク

本章の製品開発における市場リスクは、顧客視点の製品価値と企業視点による製品価値が一致するのかどうかという不確実性と、一致しない場合に生じる経済的損失量である。製品価値は企業の視点と顧客の視点で異なる。その理由は、製品の価値は企業が造り出すものであるが、製品の価値は顧客によって評価されるからである。例えば、企業の視点で規定した製品の投入時期並びに製品の売値を実現し、かつ仕様で規定した性能を実現することができたとしても、その企業視点による製品価値は顧客視点と一致するとは限らない。むしろ一致することはまれである。従って、顧客視点の製品価値と企業視点による製品価値の間では、ばらつきを持っていると考えられるから、市場リスクは存在すると言える。市場リスクの顕在化によって製品販売の低下をまねくなら、遺失利益が生じる可能性がある。

vi) ビジネス・リスク

ビジネスの環境においては、政治的要因、経済情勢、雇用情勢、社会情勢といった様々な要因が、企業が計画策定していたビジネスに影響を及ぼす。こういった要因で想定されるリスクがビジネス・リスクである。ビジネスに与える影響度合いは常に不定であるので、

ビジネス自身は不確実性を持っている。ビジネス・リスクはこの確率的な不確実性度合いだけでなく、先の要因が顕在化した時の経済的損失量も包含する。ビジネス・リスクの顕在化が製品販売の低下をまねくなら遺失利益が生じる。この遺失利益がビジネス・リスク上の経済的損失量である。

3.2.2 スケジュール、コスト、製品性能

これら六つのリスクは二つの領域に分けることができる。それは企業を取り巻く外乱要因によるか否かという区分である。上述の説明から、技術リスク、市場リスク、そしてビジネス・リスクは、企業の外部から直接的に企業に影響を及ぼしていることが認識できるので、外乱要因によるものと言える。その外乱要因は企業が制御することは容易ではない。一方、計画納期リスク、計画コスト・リスク、計画性能リスクは、企業が内包する因子によってもたらされており、企業の設計プロセス上の問題である。つまり企業が制御、管理可能な対象である。従って本章ではこの三つのリスクに焦点を当てて引き続き議論を進める。尚、以下計画納期をスケジュールと読み替える。

スケジュール、所要コスト、そして製品性能という三つの要素の中で、表層的に企業の外部から容易に認識できるのは製品性能である。しかし企業は製品性能だけに基づいて競争するとは考えにくい。製品を発売する適切な時期、製品の売値設定という次元もバランス良く検討することが求められるからである。従って、製品性能だけでなく、製品の売値に影響を及ぼす設計プロセスの所要コスト、並びに市場への投入時期に影響を及ぼす設計プロセスの所要時間を、同時に効率化・改善させることが、企業経営における重要事項の一つである。

しかし昨今では、製品開発上の設計プロセスは大規模化し、複雑性も増している。そして競争が激しく製品開発では、コストに対する感度も高くならざるをえない。更に前述の通り、リスクは全て独立関係にはない。つまり、リスクに対して直感的な管理や付け焼き刃的な対処の仕方では困難になってきていると言える。

上記で議論した設計プロセスにおけるスケジュール、所要コスト、そして製品性能に関する関係図をまとめると図 3.1 の通りとなる。前述の説明の通り、製品性能は市場への投入時期や製品の売値と直接的に関係を持つ。つまり、製品性能は単なる技術的属性だけで定義付けされていない。製品性能で位置付けられている製品の市場への投入時期は、設計プロセスの所要時間だけでなく、生産製造の所要時間、物流時間、在庫期間の関数で示すことができる。そして製品の売値の方は、設計プロセスの所要コスト、生産製造コスト、限界利益、その他指標で構成された関数で示すと判りやすい。これらの関数は実践的である。しかしこれらの関数はまだ完全に洗練されたものとは言い難い。なぜならば、双方の関数とも実践的に反映しにくい需給、競合といった製品市場の要因を含み入れていないからである。

一般的に製品の技術的な性能を向上させるためには、設計開発に対する時間と資金が必

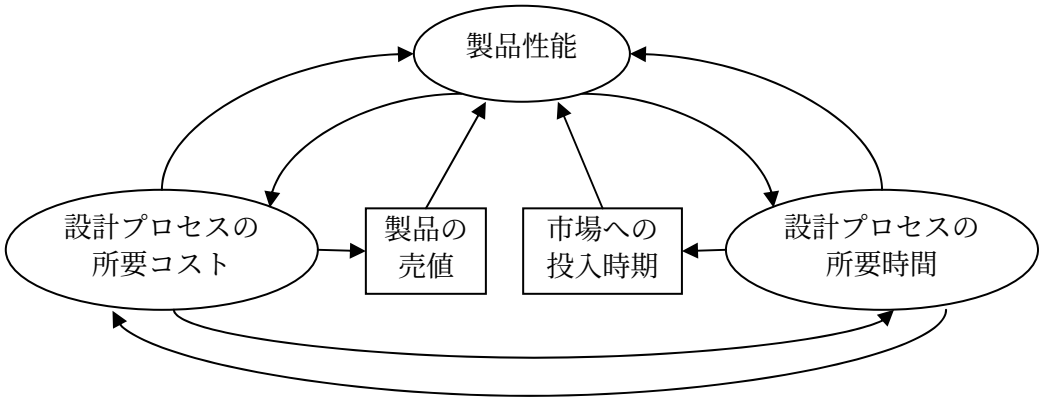


図 3.1　設計プロセスの所要コスト、所要時間、製品性能の関係図

要となる。しかしながら、予め設定していた以上の経営資源を消費しないことを前提としている。従って、企業の設計開発者にとって製品開発時の目標は、適切な時間と資金で、顧客視点の製品価値から逆算した技術的な性能を実現することである。

3.2.3　設計プロセスにおける不確実性の因果連鎖

本章では製品開発における設計プロセスを対象として、三つのリスクそれぞれの不確実性の因果関係を議論する。まず不確実性を引き起こす因子を特定することが本章で重要な事項の一つである。さらに不確実性を起因している因子同士の関係について因果連鎖ダイアグラムによって解明することも不可欠である。この因果連鎖ダイアグラムについては、先行研究や半導体設計の実プロジェクトの観察を通じて得られた知見に基づき構築する。因果連鎖ダイアグラムは、実践の設計開発プロジェクトに携わる管理者、設計技術者に対してリスク管理に向けた意思決定を側面から支援するものである。

3.3　設計プロセスにおけるスケジュール・リスク

3.3.1　スケジュール・リスクの概要

前述の通り、設計プロセスにおけるスケジュール・リスクは、予め設定した納期以内に設計解を導出できるかどうかという不確実性と、納期以内に設計解が導出できなかった時の経済的な損失量を内包するものである。そこでは設計成果の品質を犠牲にせず、かつ他の資源の増加を伴うことがないことを前提条件としている。スケジュール・リスクという

概念は一般的には納期遅延が生じる確からしさ、つまり確率を連想させる。しかし暗黙的には遅延による経済的影響度合いも含み入れている。

スケジュールの不確実性は数学的には確率密度関数で表される。スケジュールの納期を目標値とするなら、実際の納期はその目標値を中心として散らばりが生じる。散らばりは数学的には分散又は標準偏差として表される。分散又は標準偏差が大きくなれば、リスクはそれに従って大きくなる。

3.3.2　スケジュールの不確実性の因果関係

製品開発における設計プロセスの所要期間の散らばりは、多様な因子によって決定付けされるものである。本項では、そのスケジュールの不確実性をもたらす因子を辿っていくことで、スケジュールのリスクの源泉を探求することが目標である。ここでまず因果連鎖ダイアグラムの概要を構築してみる。リスクが発生した時を想定して、その原因を探って状況を改善しようと試みるのは当然の成り行きだからである。図 3.2 にスケジュールの不確実性の因果連鎖ダイアグラムを示す。このダイアグラムは実プロジェクトの観察を通して

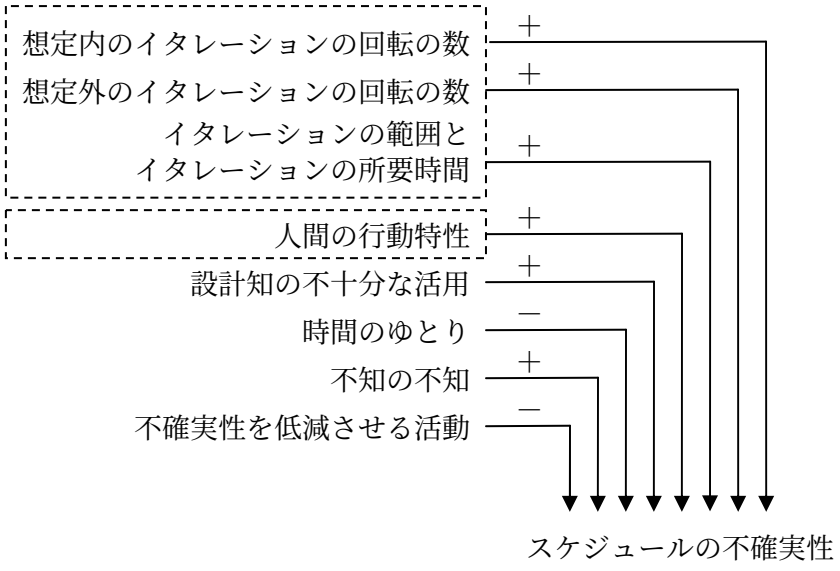


図 3.2　スケジュールの不確実性の因果連鎖ダイアグラム

得られた設計プロセスで包含する因子や事柄を包含している。

図 3.2 の説明の前にここで、本章で活用する因果連鎖ダイアグラムの定義を簡単に行っておく。まずある関係が因果関係であるためには、その関係は函数関係でなければならない。ここで因果関係にある因子を変数と捉えて議論を進める。この変数の連鎖として表した構造図が因果連鎖ダイアグラム(五百井 et al., 1997)である。本章では設計プロセス上で、どの様な要因がどの様に関連してリスクを引き起こすのかを探索し、設計プロセスの知見を得ることがねらいである。その為に、それら因子とリスクという結果までの関係を明示化した因果連鎖ダイアグラムを活用する。

設計プロセス上のリスクの因果連鎖ダイアグラムを構成する要素は三つある。まず原因と結果に該当する「変数」である。変数は数式以外の抽象的概念も包含できる。次に「矢印」という要素である。矢印は原因から結果へ向かった方向を表現したものである。そして三つ目の要素としては「因果の効果」である。因果の効果とは、矢印の両端に位置する二つの変数の関係性を示すものである。因果の効果が正に働く場合に＋、負に働く場合にーとする。五百井(1997)は因果連鎖ダイアグラムを構造図と函数表とセットとして定義している。本章で活用する正と負による因果の効果は、その函数表を代替するものである。因果の効果の活用は、因果連鎖の構造図内に統合して表現できるという利便性がある。

上記の定義の元に図 3.2 のダイアグラムについて述べる。本論では設計プロセスの主たる負荷であるイタレーションについて議論しているが、このイタレーションはスケジュールの不確実性と因果関係がある。スケジュールの不確実性を引き起こすイタレーションを測定する場合には、三つの観測視点が考えられる。一つはイタレーションの回転の数、すなわち本論で数値解析として利用する実効回転数である。これはタスクの反復作業が実質的に何回の回転が起きているかを示す変数である。イタレーションが生じなければ実質的には回転の数は１である。二つ目はイタレーションがタスクに影響を及ぼす範囲である。三つ目の変数はイタレーションの所要時間である。これらイタレーションに関わる変数が増大することによってスケジュールの不確実性が上昇する。スケジュールの不確実性を増加させる変数は他にもある。人間行動特性、設計知の不十分な活用や不知の不知等である。一方、スケジュールの不確実性を減少させる因子も存在する。それは時間のゆとりや、不確実性を低下させる活動である。図 3.2 で示した因子についてはそれぞれ引き続き以下で議論する。

3.3.3　想定内のイタレーション

設計プロセス内の各タスクが複雑な従属関係にある場合には、イタレーションが発生する。このイタレーションが大きくなれば所要期間は長期化し、スケジュールを予定納期以内に着地、すなわち設計を収束させることは困難になる。このイタレーションの大きさの変数について議論を行う前に、設計者の視点でイタレーションを二つに分類する。それは設計を行う前に予め想定していたものなのか、あるいは想定していなかったものかという

ことである。本章ではそれらを想定内のイタレーションと想定外のイタレーションと呼ぶ。

想定内のイタレーションとは予測していた範囲にあるイタレーションである。このイタレーションには製品性能や設計品質といった目標を達成させるために有効な設計情報の交換が含まれる。一方、想定外のイタレーションとは予測がつかなかったイタレーションである。イタレーションは想定外にあるので、計画策定は困難である。例えば、設計プロセスの中で予期せぬ時に新しい情報が入り込む場合に、想定外のイタレーションが発生する可能性が高い。この中には設計上の必要条件や目標が流動的であること、あるいは単純な誤りによって起因することが含まれる。

スケジュールの不確実性を引き起こすイタレーションの変数は、前項で述べた通り、三つ存在する。イタレーションの回転の数、イタレーションがタスクに影響を及ぼす範囲、イタレーションの所要時間の三つである。本項ではまず、想定内のイタレーションの回転の数に影響を与える因子について述べる。図 3.3 にその因果連鎖ダイアグラムを示す。

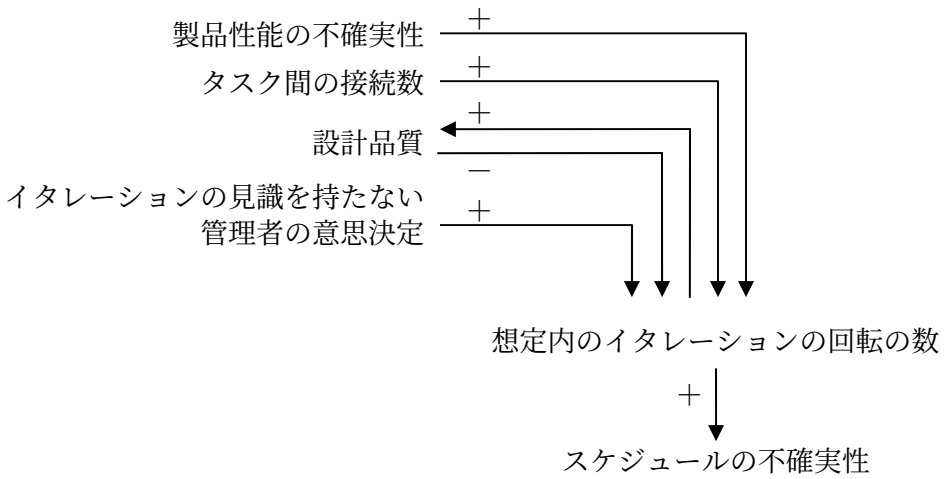


図 3.3　想定内のイタレーションの因果連鎖ダイアグラム

i)　製品性能の不確実性

イタレーションを意図して連続的に実行する場合、設計上の目標を達成させるための努力が行われていることが多い。これは製品性能を向上させる活動も含まれる。その活動は製品性能を向上させるための有効な設計情報の交換活動、あるいは製品性能の向上につながる新しい情報を取り込もうとする獲得活動である。概念設計といった抽象性の高い設計活動領域で、その活動を行うほど製品性能が向上できる可能性は高い。そしてその設計領域で、設計の確認作業を頻繁に行うほど有効である。

上述の通り、要求仕様で規定した製品性能を、設計解として導出できるかどうかという不確実性が存在しているから、イタレーションを意図的に引き起こしている。そして問題解決を行おうとしている。これは相反的な作業でもある。製品性能の問題を解決できる可能性は高くなるが、その解決に向けた活動がイタレーションを増加させるからである。結果としてスケジュールの不確実性を招来する。従って製品性能の不確実性からイタレーションの回転の数への因果の効果は正に働くことになる。

ii)　設計品質

設計プロセス上のイタレーションによる有効性は設計品質の向上である。設計成果物が規定仕様通りに仕上がっているのかという繰り返し検査と修正活動は、設計品質の向上につながるからである。従って、想定内のイタレーションの回転の数から設計品質への因果効果は正に働く。しかし一方、設計品質が高まれば、イタレーションは減少する。従って、設計品質から想定内のイタレーションの回転の数への因果効果は負に働く。

iii)　タスク間の接続数

設計プロセスは、タスクを要素とした集合である。タスク同士はシーケンシャル(sequential:設計活動は一方向に進展)、コンカレント(concurrent:複数の設計活動が並行に進展)、又は相互依存的（interdependent:設計活動は双方向に移動）に接続されている。設計プロセス上のタスク数が多くなれば、タスク間の接続数は多くなる。設計プロセス上のタスク間で複雑に相互依存関係が構築されている場合も、タスク間の接続数は多くなる。つまり、設計プロセス上のタスクが多くなるか、あるいは設計プロセスが複雑になるなら、タスク間の接続数は多くなる。タスク間の接続数が多くなれば潜在的にイタレーションを引き起こす経路が増えることにつながるので、イタレーションの回転の数も増える可能性がある。従って、タスク間の接続数から想定内のイタレーションの回転の数への因果効果は正となる。

昨今では設計プロセスの大規模化と複雑化が進んでいる。こういった設計プロセスの計画策定を行う場合にはタスク分割を行うのが一般的である。設計対象物の構造部品毎に合わせてタスク編成するか、あるいは設計組織のチーム毎に合わせてタスク編成するか等多

様に検討が行われる。こういったタスク分割を行う場合には、タスク間の接続数を最小化させる様に最適化を行う方が適切である。タスク数の接続数を減らすことができれば、イタレーションの回転の数を減少させる。そしてスケジュールの不確実性を低減させることができるからである。

iv)　イタレーションの見識を持っていない管理者の意思決定

プロジェクトの管理者は設計プロセスの進捗に応じて、製品性能や設計品質がどの程度達成されているのか、どの程度の時間が残っているのか、どの程度の予算が残っているのかといったことを監視する。そして製品性能や設計品質が規定を満足していない場合には、意図的にイタレーションの追加実行の意思決定を行う。しかしこの意思決定には本来、どういった付帯条件で、どの程度のイタレーションを実行するのかが含まれるはずである。この様な付帯条件や定量数値を伴う意思決定は管理者のイタレーションに関する見識や経験に依存する。例えば、スケジュールとコストとの二律背反問題とイタレーションを理解せずに、設計プロセスの計画策定することは困難である。また設計品質の向上とイタレーションの関係を認識せずに適切な意思決定することは難しい。この様な一般的な管理者の元では、意図的にイタレーションを実行するという以上の意思決定は行われにくい。つまり、イタレーションの見識を持っていない管理者の意思決定から想定内のイタレーションの回転の数への因果効果は正となる。

3.3.4　想定外のイタレーション

想定外のイタレーションとは予測できないイタレーションである。このイタレーションは想定外にあるので、計画策定は困難である。この想定外のイタレーションの回転の数に影響を与える因子を表した因果連鎖ダイアグラムを図 3.4 に示す。そして各因子についてそれぞれ述べる。

i)　製品性能に関する不確実性

要求仕様で規定された製品性能が不足している為に、その目標値を満足させようとする設計活動がイタレーションとして生じる。例えば、設計成果物の性能検証をテスト機器で行った結果、仕様を満足していない場合には設計の修正、変更を行う。これは手戻り修正であり、イタレーションに含まれる。手戻り修正がそのまま連続して続けば、イタレーションの実効回転数は大きくなる。つまり、製品性能に関する不確実性が大きくなれば、イタレーションを増大させる可能性が高い。特に、製品性能を仕様通り実現できると確信していた場合に生じるイタレーションは、想定外のイタレーションである。

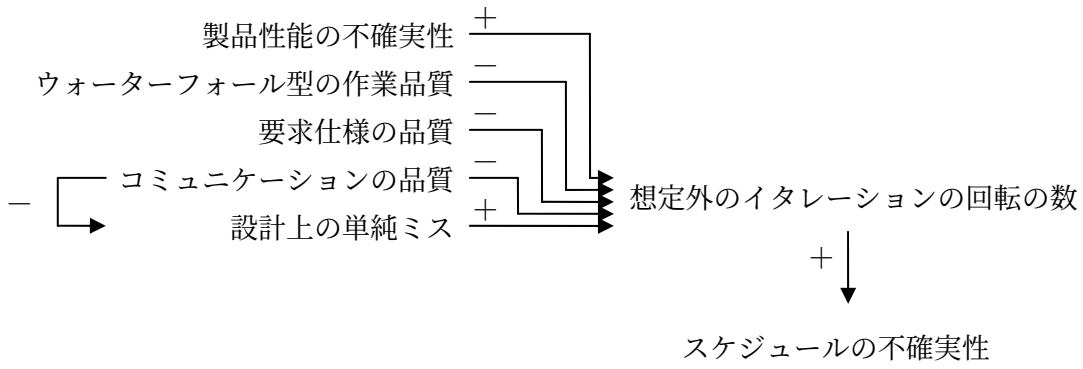


図 3.4　想定外のイタレーションの因果連鎖ダイアグラム

ii)　ウォーターフォール型の作業品質

設計プロセスにおいては、タスクの出力ともう一つのタスクの入力が直列に接続されている場合がある。この一連のタスクの流れは、ウォーターフォール(waterfall)又はシーケンシャルである。つまり作業は一方向だけに進展し、逆方向に後戻りすることはない。こういったウォーターフォール型作業では先行順にタスクを実行し、そのタスクが終了次第、次のタスクに情報を受け渡す。受け渡される情報は、前のタスクに後戻りさせる様な不適切な情報ではない。

しかしウォーターフォール型作業であっても、適切な管理が行われていない場合、簡単にイタレーションが生じてしまう。それはタスクが受け取る情報が適切でない、受け取る情報の時期が適切でない、受け取る情報が正しい場所で行われていないという理由によるものである。従って、ウォーターフォール型の作業品質は大変重要な因子である。この作業の質が悪ければ、想定外のイタレーションを引き起こすだけでなく、イタレーションの回転の数、イタレーションの及ぼす範囲、イタレーションの所要時間にも大きな影響を与えることになる。つまりウォーターフォール型の作業品質から想定外のイタレーションの回転の数への因果の効果は負に働くことになる。

iii)　コミュニケーションの品質

最適なコミュニケーションとは、受け渡しする情報が適切であること、その情報の受け渡しする時期が適切であること、そしてその情報を受け渡しする場所が適切であることを包含する。つまりコミュニケーションの最適性は、情報の媒体や転送手段、時間、場所等多面的な最適性に関係している。ここでこの最適性の度合いを、コミュニケーションの品質と定義する。こういった最適なコミュニケーション品質が達成できるなら、設計プロセス上で生じる想定外のイタレーションの回転の数を減少させることができる。従ってその因果効果は負に力が働く。

iv)　要求仕様の品質

要求仕様は、プロジェクトの管理者にとっては関心の最も高い事項の一つである。それはプロジェクト上、様々な面に影響を及ぼしうるからである。例えば、要求仕様の定義が不明瞭であれば、製品開発というプロジェクトの遅延を招いてしまう。曖昧な要求仕様であれば、問題を全く招来せずに設計解を導出することは困難である。この場合にはそこで発生した新たな問題を解決するために、手戻り修正で対処することにつながる。

要求仕様が不安定な場合もある。つまり確定すべき仕様が決定されず、ゆらぎが見られる場合である。ある範囲の領域で変動している状況とも言える。この様な状況下では、設計上の目標が簡単に変更される。そして優先事項も変更されることがある。結果として、その都度新しい設計情報を探求するために手戻り修正が発生する。設計上のゆらぎはその時点では取るに足りない小さいものと感じていても、概念設計まで手戻り修正が及ぶなら相対的に大きな問題となる。

一方、要求仕様が明瞭である、あるいは仕様上規定されるべき値が決定している状況下であれば、上記の問題が生じる可能性は低い。つまり要求仕様の品質の度合いが、手戻り修正の度合いに影響を与えているのである。要求仕様の品質が高まれば手戻り修正の度合いは低くなり、逆に品質が低くなれば手戻り修正の度合いは高くなる。イタレーションの回転の数は手戻り修正の度合いの構成要素の一つと考えられるので、要求仕様の品質からイタレーションの回転の数への因果効果は負となる。

要求仕様の品質が高まらない理由の一つは、その製品を投入する市場の不確実性による場合もある。この場合には、顧客視点による製品価値の理解を高めれば、要求仕様の安定性と品質を高めることができる。

v)　設計上の単純ミス

設計プロセスにおいては、人為的な単純ミスが生じる。前例のない新しい設計作業では、学習効果が働かないため、更に単純ミスを引き起こしやすい。こういった単純ミスも、予測していなかった手戻り修正を引き起こす。すなわち、想定外のイタレーションを引き起

こす。従って、設計上の単純ミスからイタレーションの回転の数への因果効果は正となる。

人為的な作業では単純ミスを完全に取り除くことは困難であるので、単純ミスは早期に発見すること、そしてすぐに修正することである。この早期発見と早急な対応の施策の一つは、組織内のコミュニケーションの品質を高めることである。

3.3.5　イタレーションの範囲とイタレーションの所要時間

イタレーションが影響を及ぼす範囲とイタレーションの所要時間は、スケジュールの不確実性に大きな影響を与える因子である。イタレーションが影響を及ぼす範囲を小さくすること、又はイタレーションの所要時間を短くすることが、スケジュールの遅延発生を低減させる。逆に範囲が大きくなるか所要時間が長くなれば、スケジュールの不確実性は増加する。つまりイタレーションの範囲と所要時間からスケジュールの不確実性への因果効果は正に働く。

このイタレーションの範囲とイタレーションの所要時間に影響を与える因子を表した因果連鎖ダイアグラムを図 3.5 に示す。次に、イタレーションの範囲と所要時間に対して直接作用する因子について述べる。

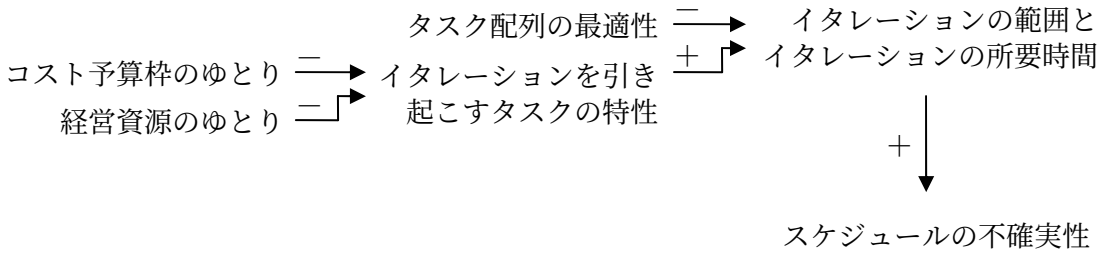


図 3.5　イタレーションの範囲とイタレーションの所要時間の因果連鎖ダイアグラム

i)　タスク配列の最適性

イタレーションを増加させると想定したタスクに対しては、その影響度が大きくならない様に事前に配慮すべきである。言い換えると、顕在化した場合に備えて、イタレーションの影響を及ぼす範囲を小さくする様に、あるいは所要時間を短くする様に、最適性を考えて設計プロセスを構築することである。

例えばイタレーションを引き起こしそうなタスク群については、クラスタ(cluster)化しておくことでイタレーションの範囲を減少できる(Pimmler & Eppinger, 1994)。そして更にそれらを設計プロセスの初期段階に配置しておけば、それらが顕在化した時でも、所要時間の短い手戻り修正の発生ですむ可能性が高くなるからである。従って、タスク配列の最適性からスケジュールの不確実性への因果効果は負に働く。

ii)　イタレーションを引き起こすタスクの特性

イタレーションを引き起こす因子であるタスクの数、並びにそのタスク群の平均所要時間は、イタレーション全体と大きな関係性を持っている。従って、イタレーションを引き起こすタスクの数と平均所要時間といった特性を減少させることで、スケジュールのリスクを減らすことができる。この様にイタレーションを引き起こすタスクの特性から、イタレーションの範囲とイタレーションの所要時間への因果の効果は正である。

イタレーションを引き起こすタスクの特性を具体的に減少させるには、相互依存性にあるタスク間のインタフェースを切断すること、イタレーションを引き起こすタスクの平均所要時間を直接的に低減させる対応を行うことである。

設計プロセス内のタスクの内容は、人為的に決定されるものである。決定される内容は、開始時期と終了時期、そのタスクに投入する資源である。この資源には、人、設備といった有形物、資金、情報を含む。先のタスク間のインタフェースや所要時間は、この様な経営資源にゆとりがなければ、制御は困難である。つまり、必要十分な経営資源が確保され、設計プロセスのコスト予算枠にゆとりがあるならば、イタレーションを引き起こすタスクの特性を有利に変えることができる。従って、コスト予算枠のゆとり並びに経営資源のゆとりから、イタレーションを引き起こすタスクの特性への因果の効果は双方とも負に働く。

3.3.6　人間の行動特性

Goldratt (1997)は、スケジュールの不確実性を三つに分解している。タスク時間の不確実性、リソースの不確実性、パス経路の時間の不確実性である。そしてそれらは人間の行動特性に起因するとして、その不確実性に関連づけしている。図 3.6 にその人間の行動特性とスケジュールの不確実性の因果ダイアグラムを示す。このスケジュールの不確実性を引き起こしている因子についてそれぞれ述べる。

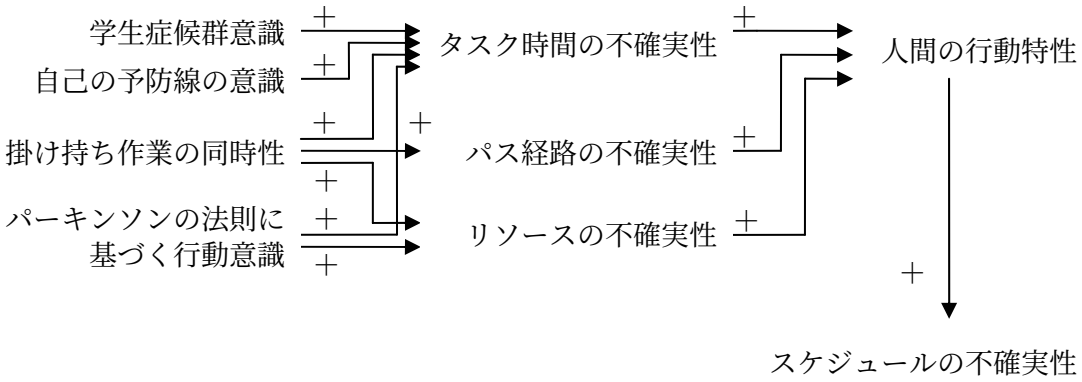


図 3.6　人間の行動特性による不確実性の因果連鎖ダイアグラム

i)　タスク時間の不確実性

人間の行動特性の視点からは四つの因子がある。これらがタスク時間の不確実性に影響を与えている。一つ目は学生症候群的意識である。この意識は学生症候群(Goldratt, 1997; 津曲, 2004)として取り上げられている。一般的なプロジェクト管理では、不確定さに対しては安全余裕時間が付け加えられている。しかし安全余裕時間による工数見積もりは、実際とはかけ離れてしまう。なぜならば、納期ぎりぎりになるまで着手せず、最終的には一夜漬けの集中作業となることがあるからである。そして納期が遅延してしまう。たとえ突貫作業で間に合わせたとしても、逆にタスクの品質又は成果物が悪くなってしまう可能性が残る。これが学生症候群である。つまりこの学生症候群により、付け加えられた安全余裕時間は浪費されてしまう。この意識が高ければ、タスク時間の不確実性は大きくなる。従って、学生症候群的意識からタスク時間の不確実性への因果効果は正に働く。

二つ目の因子は、自己の予防線(Goldratt, 1997; 津曲, 2004)の意識である。この意識は、自己防衛が働き、自己に対して予防線を張る人間行動特性である。例えば、タスクが予定より早めに終了した場合には、その事実は管理者へ報告されにくい。それは正直に報告すれば、次回のプロジェクトで不利になると考えるからである。つまり作業が早く終了して浮いた時間は無駄に消費されるだけになる。自己の予防線の意識も高ければ、タスク時間の不確実性は大きくなる。従って、自己の予防線の意識からタスク時間の不確実性への因果効果は正に働く。

三つ目の因子は、パーキンソンの法則(Parkinson, 1996)に基づく行動意識である。この法則によれば、人的リソースの数は仕事の有無にかかわりなく一定の割合で増加する。むしろ不必要な仕事はいくらでも作り出せるという組織内力学が、組織の拡大を続けているこ

とを示唆したものである。この行動意識が大きければ、タスク時間の不確実性は高まる。従って、パーキンソンの法則に基づく行動意識から、タスク時間の不確実性への因果効果は正に働く。

四つ目の因子は、掛け持ち作業の同時性である。これは複数の作業が、一人の作業者によって同時進行されることである。この場合、掛け持ち作業の中での優先権が明確にされていないことが多い。結果として一作業当たりの作業者の生産性は低下する。これがタスク時間の不確実性を生み出す。従って掛け持ち作業の同時性から、タスク時間の不確実性への因果の効果は正である。尚、タスク時間の不確実性が増大するとスケジュールの不確実性は増大するので、タスク時間の不確実性からスケジュールの不確実性への因果の効果は正である。

ii)　リソースの不確実性

リソースの不確実性に影響を与える人間行動特性は二つの因子を含む。パーキンソンの法則に基づく行動意識と掛け持ち作業の同時性である。パーキンソンの法則に基づく行動意識に従えば、人的リソースは膨張する可能性がある。そして無駄な仕事を新しく生み出し、組織の肥大化が生じる。つまりリソースの無意識の増大である。従って、パーキンソンの法則に基づく行動意識からリソースの不確実性への因果効果は正に働く。

掛け持ち作業の同時性もリソースの不確実性に影響を与える。それは、複数のタスクが同時に生じ、必要なリソースが競合してしまう場合である。特定の専門知識やノウハウを持っている人や装置といったリソースが限られているならば、実際には二分化することは困難である。つまり必要な時に、必要な場所で、必要なリソースを確保することが困難になる状況がある。従って、掛け持ち作業の同時性からリソースの不確実性への因果効果も正に働く。尚、リソースの不確実性が増大するとスケジュールの不確実性は増大するので、リソースの不確実性からスケジュールの不確実性への因果の効果は正である。

iii)　パス経路の不確実性

このパス経路は、スケジュール計画をネットワーク図で表し、そしてそれを分析する際に必要な構造図である。つまり、このパス経路はクリティカル・パスである。クリティカル・パスは計画から実践の適用において、経路に変更がないことを想定している。しかし、実際にはクリティカル・パスの経路は状況に応じて変化する可能性があり、総所要期間の最適化が検討されにくい。言い換えると、クリティカル・パスと考えていなかったパス経路がクリティカル・パスになる可能性を残している。従って、パス経路の不確実性からスケジュールの不確実性への因果効果は正に働く。

パス経路の不確実性は、他に掛け持ち作業の同時性とも関係がある。それは掛け持ち作業が原因で、クリティカル・パスが別の経路に置き換わることがあるからである。従って、

掛け持ち作業の同時性からパス経路の不確実性への因果効果は負として働く。尚、パス経路の不確実性が増大するとスケジュールの不確実性は増大するので、パス経路のの不確実性からスケジュールの不確実性への因果の効果は正である。

3.3.7　設計知の不十分な活用

設計知の不十分な活用は、スケジュールの不確実性に大きな影響を与える因子の一つである。設計知の不十分な活用が進めば、スケジュールの不確実性を増加させる。逆に設計知の不十分な活用が低下する、すなわち設計知の再利用を含めた活用が進めば、スケジュールの不確実性は低減する。つまり設計知の不十分な活用からスケジュールの不確実性への因果効果は正に働くと言える。

この設計知の不十分な活用という因子と、スケジュールの不確実性に影響を与える因子を含む因果連鎖ダイアグラムを図 3.7 に示す。尚、本項で言及した設計知の定義について以下に述べる。

まず設計知識を構造、過程、機能という三つの視点で分解し、設計対象物における構造的知識、過程知識、そしてシステム知識(青島・延岡, 1997)に分ける。構造的知識は形式化されている設計情報を内包する。設計情報は多様な変形をしていくが、各設計情報は有機的に体系化されており知識形成が行われている。一方、過程知識とシステム知識は形式化が困難で、簡単に移転できない暗黙知(野中・竹内, 2002; Polanyi, 2003)である。過程知識は設計対象物に関する情報の内、時間的なコンテクストの中で意味づけられる。つまり設計開発における試行錯誤的な問題解決活動の経緯の中で意味づけられている情報が過程知識である。そしてシステム知識は、設計対象物に関する情報の内、設計対象物というシステムや組織システムのコンテクストの中で意味づけられる。つまり設計対象物というシステムや組織システムを構成する様々な要素関係の中で意味づけられた情報のことを指す(小川,

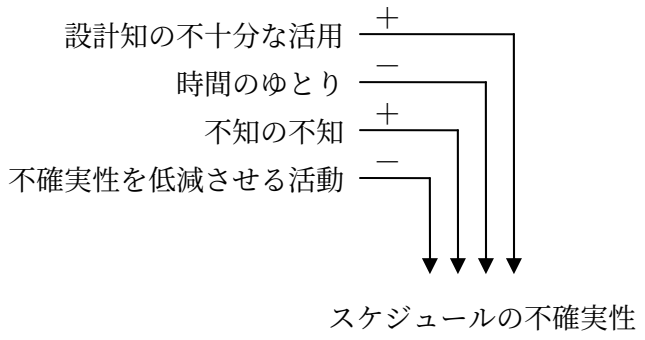


図 3.7　設計知の不十分な活用とその他因子の因果連鎖ダイアグラム

2001)。

本章では、設計対象物を対象とした設計に関わる情報を包含した構造的知識・過程知識・システム知識をまとめて、設計知と定義する。

3.3.8　時間のゆとり

時間のゆとりは、プロジェクトの中に盛り込まれる安全余裕時間である。スケジュールに不確実性があるから、遅延の防止策として考えられた。時間にゆとりがあれば、遅延を防止する方向に働くので、時間のゆとりからスケジュールの不確実性への因果効果は負に働くことになる。

一方、時間のゆとりはスケジュールの長期化をまねく可能性がある。安全余裕時間の必要性は判っていながらも、適切な時間量は判りにくい。従って、安全余裕時間量は恣意的に決定されることが多い。この恣意性がスケジュールの長期化をまねく計画策定につながる。つまり安全余裕時間の品質は所要期間と関係しており、スケジュールの計画策定上、注意すべき項目の一つである。

3.3.9　不知の不知

ソクラテスが直接表明している「不知の知」とは、「わたしは知らないから、その通り知っているとは思わない」ないし「わたしが知らないことを知っているとは思わない」である。つまり、不知を自覚していることである。一方、「不知の不知」は、「知らないのに、その通り知らないとは思わない」あるいは「知らないのに、知っていると思う」であり、知っていると思っているが、実際には知らなかったということである。これは自己の不知を自覚していないことであり、例えば、知ったかぶり、思い込み、勘違い、誤った先入観等の人間の知的行動特性が含まれる。

こういった不知を自覚していないまま、その通り行動すると深刻な結果を招く場合が多い。つまりここでスケジュールの遅延に限って言えば、不知を自覚していないことが与える影響は大きいと言える。例えば不知の不知が大きくなるならば、不知を自覚していないことが増幅されることであるから、遅延の不確実性は大きくなる。逆に、不知の不知が減少すれば遅延の不確実性は減る方向に働く。従って、不知の不知からスケジュールの不確実性への因果効果は正に働く。

知ったかぶり、思い込み、勘違い、誤った先入観といった思考から抜け出せず、「不知の不知」に陥ってしまう事態は実践現場でも散見される。この問題の解決は単純とは言えない。その原因は、とりわけ自分で知者であると思っているという自信、あるいは多くの人々から知者であると思われている状況との関係に潜んでいる（荻野,2003)。自分で知者だと勝手に思いこむ誤謬は、他人からそう思われているという自己確認を支えとして成立しているとも言える。もしそうであるなら、他人の評価を媒介に自己の虚像がふくらんでいくこ

とである（荻野,2003）。つまり知識が人間を誇らせ、その結果もう一つ別の知識に関する決定的な無知へと転落させる危険な陥穽があるからである（荻野,2003）。ここに一つの人間の持つ知識の弱さや限界がある。「不知の知」すなわち自己の不知の自覚は、この不知の不知に陥らせないための逆説的表明である（田島,1988;荻野,2003）。

3.3.10 不確実性を低減させる活動

スケジュールの不確実性に対する管理上の施策はリスク・マネジメント（亀井, 1997）に従えば、まずリスクの調査と確認を行い、次に評価と分析、そして処理である。この一連の流れでリスクを低減させることが重要な活動である。

スケジュールの不確実性は、本章で議論した因子によって影響が及ぼされている。そしてその因子の重み付けによって、適切な処理計画が必要となる。この処理計画とは、スケジュールの不確実性に影響を与えている因子に従属した具体的な処理手段を選択することである。

スケジュールの不確実性に最も大きな影響を与えている因子については、設計プロセス上のどの時点で、どの工程場所で生じるのか特定する必要がある。これが調査と確認である。次に評価と分析を目的に、継続的に監視する。そして具体的な処理をいつ、どこで行うのか計画する。

この様に不確実性を低減させる為の重要な活動は計画策定である。この計画策定には、リスクの調査と確認、評価と分析、処理手段の選択が含まれる。そして次に処理手段が実行される。計画策定なしに適切な処理手段は実行できないと言える。

尚、不確実性を低減させる活動が実行されれば、不確実性は低減するので、不確実性を低減させる活動からスケジュールの不確実性への因果の効果は負に働くこととなる。

3.3.11 スケジュールの不確実性の因果連鎖の総合

スケジュールの不確実性の因果連鎖ダイアグラムに関しては図 3.2 で示している。この図 3.2 と上記で議論した因子を総合化することで、スケジュールの不確実性全体の構造図を表すことができる。その全体の構造図を図 3.8 に示す。

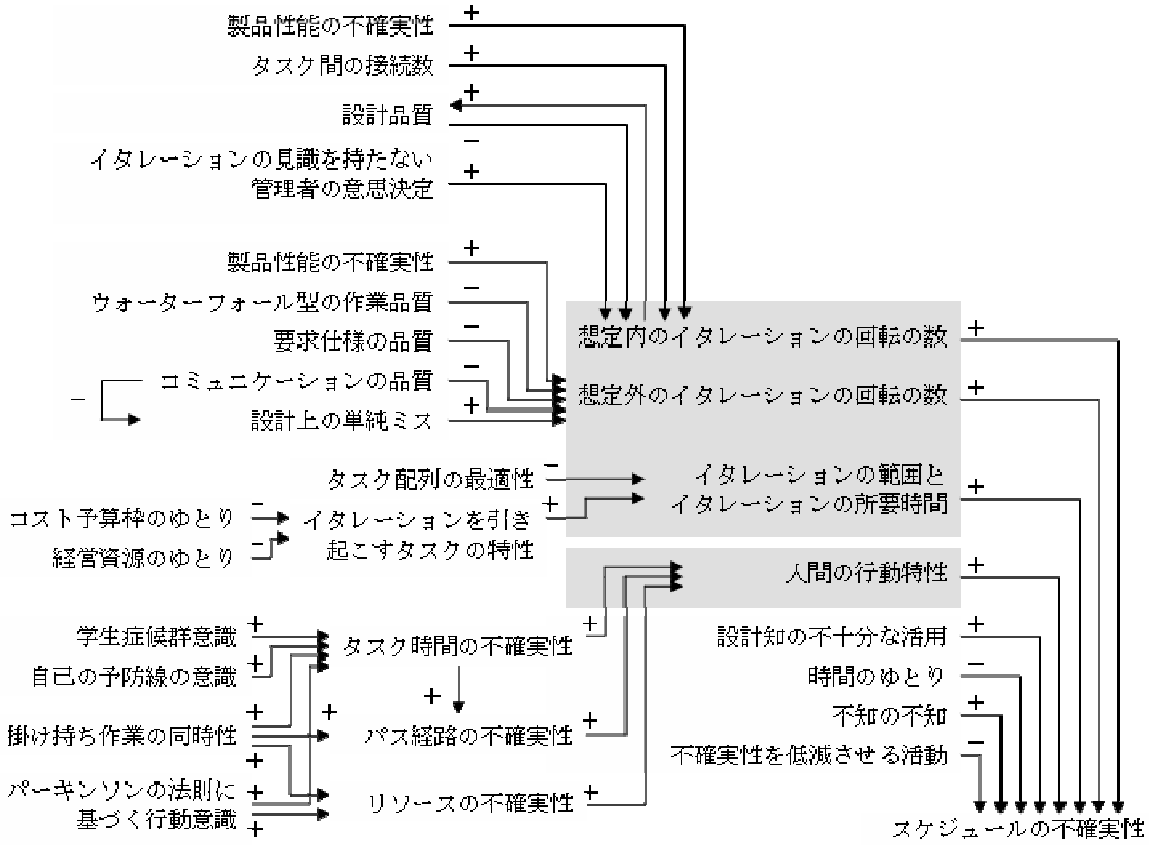


図 3.8　スケジュールの不確実性の因果連鎖の総合

3.4　設計プロセスにおける設計開発コスト・リスク

3.4.1　設計開発コスト・リスクの概要

設計開発コスト・リスクとは、予め計画していた予算以内で設計解の導出をできるかどうかという不確実性と、予算を超過した場合に生じる経済的損失量である。ただし、この不確実性には付帯条件がつく。それは規定の設計仕様を満足し、かつ計画納期で設計解を実現できる前提条件である。設計開発コストは、設計開発工程の作業で消費される資源、すなわち費用である。欧米では、NRE (Non-Recurring Engineering Cost:回収不能技術費用)と呼ばれることがある。そして設計開発コストは変動費ではなく固定費に属する。尚、本章の設計開発コスト・リスクには、設計開発資金の調達の不確実性、市場リスク並びにビジネス・リスクといった不確実性は含んでいない。

3.4.2　設計開発コストの不確実性と因果関係

予め計画していた設計開発コストで設計解を導出できるのかどうかという不確実性とそれを取り巻く因果連鎖ダイアグラムを図 3.9 に示す。この図の中で示されている因果関係にある因子についてそれぞれ以下で述べる。

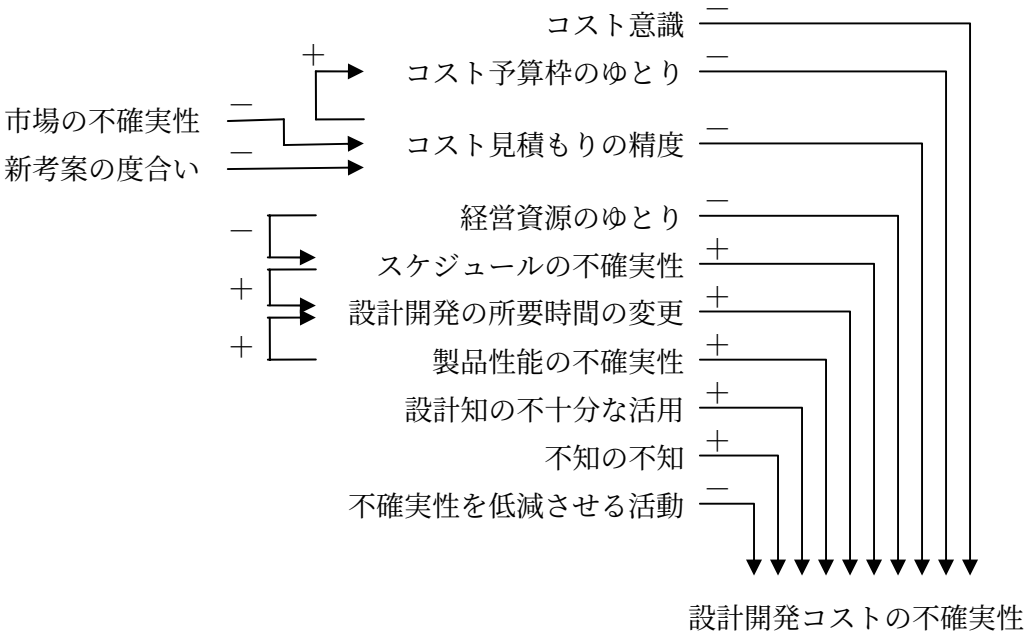


図 3.9　設計開発コストの不確実性の因果連鎖ダイアグラム

i)　コスト意識

設計開発コストに一層気を配るならば、コスト増大に対する抑止効果は働く。そして設計開発コストの不確実性を潜在的に減少させる。従って、コスト意識の増加は設計開発コストの不確実性を減少させる。つまりコスト意識と設計開発コストの間では因果の効果は負となる。

コスト意識はコストが予算以内に収まりきらない時には働きやすいが、予算に余裕がある時には逆の意識に働きやすい。現実的にはタスクに対して予算が割り当てられると、その予算全てを消費する傾向にある。

従ってコスト意識を高める適切な動機付けが必要となる。例えば、必要以上に経営資源を利用しない、あるいは無駄を取り除くという動機付けが与えられるなら、コスト意識は高まる可能性がある。

ii)　コスト予算枠のゆとり

コスト予算枠が広がるならば、その予算枠以内で設計解を導出できるかどうかという不確実性は減少する。その理由は予算枠が広がった資金を、まだ不完全な設計解を解決する対処策に振り向けられることである。従ってコスト予算枠のゆとりから設計開発コストへの因果の効果は負となる。

iii)　コスト見積もりの精度

コスト見積もりの計画策定をするなら、それをしない場合と比べて明らかに設計開発コストの不確実性は減少する。計画策定を更に徹底的に進めるなら、その不確実性は一層減少する。従ってコスト見積もりの精度と設計開発コストの不確実性間では、因果の効果は負となる。

次に市場の不確実性との因果関係を考える。コスト見積もりは市場の不確実性を考慮すると複雑化してしまう。それは市場の視点、すなわち顧客視点にたって製品価値を決定するのに、どの程度の設計開発費を費やす必要があるのか判断が困難だからである。つまり市場の不確実性が高くなれば、コスト見積もりの精度は低くなる。従ってコスト見積もりの精度から市場の不確実性への因果効果は負に働く。

コスト見積もりの精度に影響を与える因子は他に、新考案された製品も含まれる。具体的には新しい設計思想を含む製品、新しく工夫が施された製品がある。こういった製品は過去に具現化したことがないので、参照する定量的データは不足している。この情報不足が結果としてコスト見積もりの精度を悪化させる。

コスト見積もりの精度は、コスト予算枠のゆとりにも関係する。コスト見積もりの精度が高くなれば、見積もりの不確実性は低減する。そしてその不確実性が低減する分、不確実性によって影響を受けるコスト負荷も低くなり、コスト予算枠に余裕が生じるからであ

る。つまりコスト見積もりの精度が高くなれば、コスト予算枠のゆとりは大きくなる。従って、コスト見積もりの精度からコスト予算枠のゆとりへの因果の効果は正となる。

iv)　経営資源のゆとり

経営資源の余裕をもたせるかどうかという問題は、一つの企業経営上の問題である。経営資源のゆとりを持たせすぎていると、結果として経営資源を無駄に消費していたことにつながる。そのゆとり分を別の活動作業に振り向けていた方が良かったことになるからである。しかし経営資源のゆとりは不確実性を低減できる利点がある。それは前述のスケジュールの不確実性との関係でも述べている。つまり、経営資源のゆとりからスケジュールの不確実性に対して負の因果効果があるということである。この経営資源のゆとりと設計開発コストの不確実性の間も、それと同様の因果効果がある。経営資源にゆとりがあれば、コスト問題が発生次第その問題解決に経営資源を振り向けられるからである。

v)　製品性能の不確実性

要求仕様に記されている製品性能を実現できるかどうかという不確実性は、製品開発で生じる。これが製品性能の不確実性である。この不確実性が増大するなら、製品性能上の課題と考えられる箇所を修正し克服する努力を重ねる。この努力は結果として設計プロセス上の所要時間あるいは所要コストとして跳ね返ってくる。従って、製品性能の不確実性から設計開発コストの不確実性への因果の効果は正となる。

vi)　スケジュールの不確実性

スケジュールの不確実性が顕在化する場合、それは遅延を意味する。そして遅延が生じることで経済的損失を招来する。その損失は遺失利益の様な潜在的な損失だけでなく、設計開発コストも包含することもある。例えば製品の市場投入の時期によっては、設計開発コストに影響を及ぼす。つまりスケジュールの不確実性が大きくなると、設計開発コストの不確実性も大きくなる。従って、その因果の効果は正となる。

vii)　設計開発の所要時間の変更

Graves (1989)は、一般プロジェクトの所要時間と所要コストの間にどの様な関係があるかを議論している。そこでは最適な設計解を選択する限り、所要時間と所要コストの関係は二律背反になるというものである。つまりプロジェクトの所要時間を短縮しようとすると所要コストが上昇する。これを複数の事例や研究報告で検証を行っている。

この主張に従うなら、設計開発の所要時間の変更を行うと設計開発コストに影響を及ぼす可能性が高い。言い換えると、その二つの因果の効果は正である。設計開発の所要時間

の変更は逆に、製品性能の不確実性、並びにスケジュールの不確実性によって影響を受ける。これらも全て正の因果効果である。

viii)　設計知の不十分な活用

設計知の不十分な活用は、設計開発コストの不確実性に大きな影響を与える因子の一つである。設計知の不十分な活用が低減するなら、すなわち設計知の再利用を含めた活用が増加するなら、設計開発コストの不確実性は低減する。つまり設計知の不十分な活用から設計開発コストの不確実性への因果効果は正に働く。

ix)　不知の不知

スケジュールの不確実性で議論した因子と同等である。不知の不知が減少すれば、設計開発コストの不確実性が減る方向に働くので、不知の不知から設計開発コストの不確実性への因果の効果は正である。

x)　不確実性を低減させる活動

設計開発コストの不確実性に対する管理上の重要な施策は、本章のスケジュールの不確実性で述べた活動と同等である。つまり計画策定である。この計画策定には、設計開発コストに対するリスクの調査と確認、評価と分析、処理手段の選択が含まれる。そして次に処理手段が実行される。

尚、不確実性を低減させる活動が行われれば、不確実性は低減するので、不確実性を低減させる活動から設計開発コストの不確実性への因果効果は負に働く。

3.5　設計対象物の性能リスク

3.5.1　性能リスクの概要

製品開発における設計プロセスは、概念設計、実体設計そして詳細設計を経て設計解を導出する一連の過程である。その過程では、顧客視点の製品価値を反映して策定した仕様に基づいて設計努力を重ねる。この設計努力とは設計活動の実行と製品性能を目標通り実現することである。

こういった設計活動が行われている時に、製品性能を仕様通りに実現できるかどうかという不確実性が生じる。設計活動の間では、モノ的な製品は完成されていないので評価ができないからである。もし仕様に満たない設計解しか導出できず、しかもその仕様に満たない製品を市場投入することで遺失利益が生じるなら経済的損失が生じる。

つまり製品の性能のリスクは、仕様で規定した製品能力を実現した設計解を導出できるか

どうかという不確実性と、その設計解を導出できない場合の経済的損失量を意味する。

この製品性能を構成する属性は多い。機能、能力、信頼性、価格、販売される時期、仕様の準拠性、耐久性、利便性、審美性、品質、大きさ、重量、速度、その他数多くの特性が考えられる。性能リスクはこういった属性とその相対的な重み付けによって算出可能である。

設計プロセスをこの製品性能という一つの視点から検討するなら、それは性能リスクを受容レベルまで低減させる設計行為と言える。

3.5.2　製品性能の不確実性の因果関係

本項では製品性能の不確実性の因果関係を探求する。図 3.10 にその因果連鎖ダイアグラムを示す。そしてこの図の中で示されている因子と因果効果についてそれぞれ述べる。

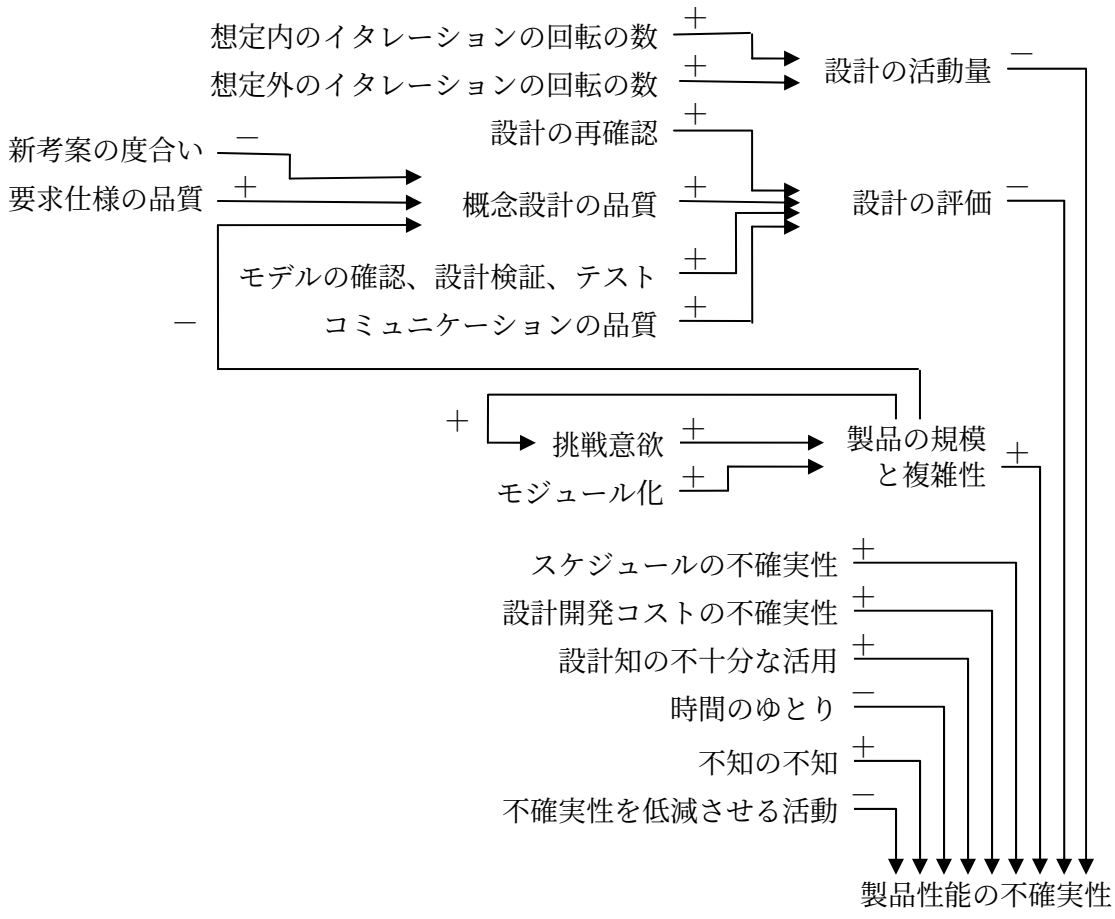


図 3.10　製品性能の不確実性の因果連鎖ダイアグラム

i)　設計の活動量

本項における設計の活動とは、図面やプログラムの作業、意思決定、評価、分析等設計に関わる活動である。設計の活動量とはその活動に費やす人材の数と時間を含む。この活動行為によって設計情報が創出される。そしてその設計情報を別の活動行為が受け取り、更に新たな設計情報を創出する。この一連の流れが設計プロセスであり、この設計プロセスを経て最終的に設計解が導出される。

この様な設計情報の受け渡しは、手戻り修正の必要性から逆方向に進むことがある。手戻り修正は前述の通り製品性能の不確実性を減少させる行為であり、設計目標を実現させる設計活動だからである。もしここで設計プロセスが大規模化し複雑化するなら、情報の受け渡し量は大きくなり、受け渡し経路も複雑化する。つまり設計の活動量が増えることで、イタレーションが増える可能性がある。しかし一方、長所もある。手戻り修正という設計活動は予め設定していた仕様に近づけるための行為を含んでおり、製品性能実現の不確実性を減少させることが可能だからである。従って、設計の活動量から製品性能への因果効果は負に働く。

以上の通り設計の活動量が及ぼす影響について議論した。次に、逆に設計の活動量に影響を及ぼす因子について言及する。その因子は設計情報の受け渡し、すなわちイタレーションである。このイタレーションが大きくなれば、設計の活動量は増加する。つまりイタレーションの回転の数から設計の活動量への因果の効果は正に働く。尚、ダイアグラムでは 3.3 節で分類した通り、想定内のイタレーションと想定外のイタレーションに二分化している。

ii)　設計の評価

本項における設計の評価とは、設計解の導出の後に実行する解析、テストである。設計解は、図面、プログラム、試作モデルといった設計成果物である。この評価では設計成果物が実際に仕様に準拠しているのかどうか、効能があるのかどうかといった作業が行われる。この様に設計の評価が進めば、製品性能の不確実性は明らかに減少するので、因果効果は負となる。

次に設計の評価に影響を及ぼす因子について述べる。具体的には設計の評価の質並びに量という変数を従属させる因子である。

・設計の再確認

設計の再確認とは、設計解を導出していながらも、すぐに設計の評価を実行せずに、設計解の再確認を行うことである。確認作業としては、設計手順通り適切に実行されているのか、規定通り過不足なく設計解が導出されているのか、仕様通りの設計解が導出されているのかが含まれる。従って、再確認作業によって設計の評価の質を向上させる

ことができる。

再確認作業は実行すればするほど、設計解に潜む問題点や誤りを発見することができるので、設計成果物の品質を高められる。しかし反面、実行作業の時間が要するので、設計プロセスの効率は低くなる。つまりこれは再確認作業の時間と、その時間で発見できる問題点と誤りの数の効用問題である。求める再確認の作業時間は、その効用を最大にする時間量である。

・概念設計の品質

設計プロセスでは概念設計、実体設計、そして詳細設計という流れで設計解を導出する。設計解に近づく程、実体概念に近づくので、概念設計で検討される設計情報は逆に抽象概念に近い。設計作業はこの概念設計から出発するので、この概念設計の品質によっては、設計の評価の質は左右される。

概念設計は機能展開、手順、あるいはアルゴリズムの検討から始まる。つまりシステム思考が求められる設計段階である。この概念設計の品質に影響を及ぼす因子がある。それはまず設計対象物である製品の規模と複雑性である。規模と複雑性が大きくなれば概念設計の品質は低減する方向に力が働く。次に新考案の度合いである。これは新しく考案された製品あるいは新しい機能が盛り込まれているかの度合いである。この場合、徹底した品質保証がされているとは限らないので、概念設計の品質に影響を与える可能性がある。そして要求仕様の品質も概念設計の品質に影響を及ぼす。要求仕様が不明瞭である、あるいは規定されるべき値が決定していない状況であれば、高い品質の概念設計を構築できないからである。

・モデルの確認、設計検証、テスト

設計解を評価するということは、まず利用するモデルが正当性を持っているという条件のもと、設計成果物の検証とテストを実行することである。例えば、チェックリストとの照合、シミュレーションによる検証、試作モデルを構築してテスト、故障モードを設定した検証等が含まれる。

こういった作業は、経験、ノウハウ、設備、その他経営資源に依存しており、作業品質にばらつきが伴う。つまり、モデルの確認、設計検証、テストの質並びに量を高めることができるならば、設計評価の質と量を同様に高めることができる。

・コミュニケーションの品質

コミュニケーションの品質とは、コミュニケーションの最適性の度合いである。最適なコミュニケーションとは、受け渡しする情報が適切であること、その情報の受け渡しする時期が適切であること、そしてその情報を受け渡しする場所が適切であることを包

含する。こういった最適なコミュニケーションの品質が達成できるなら、設計評価の質量とも向上できる。つまり、コミュニケーションの品質から設計評価への因果効果は正に働く。

iii)　製品の規模と複雑性

製品性能の不確実性は、設計対象物の規模と複雑性が大きくなるにつれて増加する。製品を構成している部品の数、並びに部品間のインタフェースが増加するので、製品性能全体を確実に測定することが難しくなるからである。この規模と複雑性を過小評価してしまうと、不確実性のレベルは予想以上に増加してしまう。想定していたレベル以上の不確実性を招来するからである。従って、規模と複雑性は計画策定をする上で重要な項目の一つである。過去のプロジェクトと対比する等して定量的数値を準備する。そして適切な評価を実行することが一つの施策である。

次に製品の規模と複雑性に影響を及ぼす因子について述べる。それは挑戦意欲とモジュール性である。

・挑戦意欲

挑戦意欲とは自分で現在克服できる課題よりも、少し難しい課題に積極的に挑戦しようとする気持ちである。設計対象物の規模や複雑性が大きくなったとしても、この意欲の高まりがあるなら対処することは可能になる。従って挑戦意欲から製品の規模と複雑性への因果の効果は正に働く。またその効果は逆にも働く。つまり、製品の規模と複雑性から挑戦意欲への因果の効果は正に働く。規模と複雑性は、挑戦意欲をかき立てる因子となりえるからである。

挑戦意欲をかき立てる因子は他にもある。それは新規性、すなわち新考案の度合いである。この場合何人もたどり着いたことがない目標を達成した時の達成感、困難を克服した時の充実感が得られる可能性がある。従って、新考案の度合いから挑戦意欲に対する因果の効果も正に働くことになる。

・モジュール化

近年の経済学・経営学研究においてモジュール化の議論が注目されるきっかけとなったのは、Clark & Baldwin (2004)である。しかし考え方そのものの歴史は古い。青木・安藤(2002)のモジュール概念の定義によれば、「反自律的なサブシステムであって、他の同様なサブシステムと一定のルールに基づいて互いに連結することにより、より複雑なシステムまたはプロセスを構成するもの」である。そしてモジュール化とは「一つの複雑なシステムまたはプロセスを一定の連結ルールに基づいて、独立に設計されうる半自律的なサブシステムに分解すること」である。

モジュール化によって、構成要素間の調整に必要な時間やコストを削減できること、再利用による経済性、システム全体の変化の影響を局所化できること、分業の促進といった効果がある。つまり、設計対象物を標準的なモジュールに分解することで生産効率を上げることができる。従って、モジュール化から製品の規模と複雑性への因果効果は正に働く。

iv)　その他の因子

製品性能の不確実性に影響を及ぼす因子は他に、設計知の不十分な活用、時間のゆとり、不知の不知、不確実性を低減させる活動である。これらはスケジュールの不確実性でも述べた因子である。製品性能の不確実性に及ぼす因果効果もスケジュールの不確実性で言及した通り同一効果である。

3.6　本章のまとめ

3.6.1　因果連鎖ダイアグラム

本章ではまず製品開発におけるリスクについて考察した。そこで取り上げたリスクは六つである。計画納期すなわちスケジュール、計画コストすなわち設計開発コスト、計画性能すなわち製品性能、技術、市場、ビジネスである。本章では設計プロセスに直接影響を及ぼすスケジュール、設計開発コスト、製品性能を中心に議論を進めた。そして不確実性に関わる因子の因果関係をそれぞれ因果連鎖ダイアグラムで示した。

このダイアグラムは、実設計プロジェクトの観察を通して得られた知見を元に、構築された構造図である。各因子間と不確実性の因果関係が示されている。そして各因果関係の矢印上に因果の効果を付け加えている。この様にダイアグラムによって不確実性と各因子間の因果関係が明瞭となり、不確実性が増加する仕組みが認識できる。そして不確実性の低減処理に向けた洞察力を補強するものである。

3.6.2　リスク・マネジメント

本章で取り上げた不確実性は、設計プロセス内で観察される潜在的な因子である。そして設計プロセスで見られる設計負荷は、顕在化した現象である。つまり不確実性が顕在化することで生じる現象には設計負荷を包含している。従って設計負荷を引き起こす不確実性、すなわちリスクを低減させることが重要な活動である。その活動については第 4 章と第 5 章で議論を行う。

3.6.3 設計負荷

設計プロセスにおける不確実性が顕在化することで生じる現象には、設計負荷が包含されている。本章で認識できた主要な不確実性要因はまずイタレーションである。そして人間の行動特性、設計知の不十分な活用の二つの不確実性要因が続く。この三つの不確実性の要因が顕在化すると設計負荷としてプロジェクトに影響を及ぼす。

これら三つの設計負荷を引き起こす因子がここで特定されたので、次章以降の本論で評価、分析、処理手段の選択、処理の実行、そして改善に向けたサイクルについて議論を展開する。

参考文献

[1] 青木昌彦,安藤晴彦: “モジュール化：新しい産業アーキテクチャの本質”,東洋経済新報社 (2002)

[2] 青島矢一,延岡健太郎: “プロジェクト知識のマネジメント”,組織学会, Vol. 31, No. 1. pp. 26-27 (1997)

[3] 五百井清右衛門: “システム技術における因果連鎖ダイアグラム”, 早稲田大学システム科学研究所紀要, No. 21, pp. 11-19 (1990)

[4] 五百井清右衛門: “システム思考と因果連鎖図”,早稲田大学システム科学研究所紀要, No. 24, pp. 1-10 (1993)

[5] 五百井清右衛門,黒須誠治,平野雅章: “システム思考とシステム技術”,白桃書房 (1997)

[6] 小川進: “イノベーションの発生論理”,千倉書房 (2001)

[7] 荻野弘之: “ 哲学の饗宴：ソクラテス・プラトン・アリストテレス ”,日本放送協会(2003)

[8] 亀井利明: “危機管理とリスクマネジメント”,同文館 (1997)

[9] 田島孝: “「不知の知」と「不知の不知」をめぐる逆説”,東洋大学白山哲学,第22号,pp.84-98(1988)

[10] 津曲公二,酒井昌昭,中憲治: “これならできる　クリティカル・チェーン”,ダイヤモンド社 (2004)

[11] 野中郁次郎,竹内弘高: “知識創造企業”,東洋経済新報社 (2002)

[12] Boehm, Barry W.: “Software Risk Management”, Washington, D.C., IEEE Computer Society Press (1989)

[13] Brekka, Lawrence T. and Vlay, George J.: “Integrated Risk Management—Procedures and Tools”, Proceedings of the Fourth Annual Symposium of NCOSE, San Jose, CA (1994)

[14] Clark, Kim B. and Baldwin, Carliss Y. (安藤晴彦訳): “デザイン・ルール―モジュール化パワー”,東洋経済新報社 (2004)

[15] Goldratt, Eliyahu M.: “Critical Chain”, The North River Press (1997)(エリヤフ・ゴールドラット(三本木亮訳): “クリティカル・チェーン”,ダイヤモンド社 (2003))

[16] Graves, S. B.: “The time-cost tradeoff in research and development”, a review. Engineering Costs and Production Economics 16, pp.1-9 (1989)

[17] Hall, Elaine M.: “Managing Risk: Methods for Software Systems Development Reading”, MA: Addison-Wesley (1998)

[18] Hoy, Kirk L. and Hudak, David G.: “Advances in Quantifying Schedule/Technical Risk”, Estimator (Summer), pp. 25-35 (1996)

[19] Justice, Richard: “Risk in the F-22 Program: A Defense Science Board Task Force Analyzes F-22 Concurrency and Risk”, Program Manager (Jul.-Aug.), pp. 68-74 (1996)

[20] Michaels, Jack V.: “Technical Risk Management Upper Saddle River”, NJ, Prentice Hall PTR. (1996)

[21] Parkinson, C. Northcorte (森永晴彦訳): “パーキンソンの法則”,至誠堂 (1996)

[22] Pimmler, Thomas U. and Eppinger, Steven D.: “Integration Analysis of Product Decompositions”, Proceedings of the ASME Sixth International Conference on Design Theory and Methodology, Minneapolis, MN, Sept. (1994)

[23] Polanyi, Michael (高橋勇夫訳): “暗黙知の次元”,ちくま学芸書房 (2003)

[24] Reinertsen, Donald G.: “Managing the Design Factory”, A Product Developer's Toolkit New York: The Free Press (1997)

[25] Roberts, Barney B. and Winterlin, Ronald C.: “Integrated Risk Assessment: A Case Study”, Proceedings of the Sixth Annual International Symposium of INCOSE, Boston, July, pp. 7-11. (1996)

[26] Sage, Andrew P. (三森定道,明石吉三訳): “システムズエンジニアリング”,日刊工業新聞 (1997)

[27] Sarbacker, Shawn D. and Ishii, Kosuke: “A Framework for Evaluating Risk in Innovative Product Development”, Proceedings of the ASME Design Engineering Technical Conference (DETC), Sacramento,CA, Sept., pp. 14-17 (1997)

[28] Shishko, Robert and Jorgensen, Edward J.: “Evaluation of Risk Management Strategies for a Low-Cost, High-Risk Project”, Proceedings of the Sixth Annual International Symposium of INCOSE, Boston, July, pp. 7-11 (1996)

[29] Smith, Preston G. and Reinertsen, Donald G.: “Developing Products in Half the Time”, New York Van Nostrand Reinhold (1991)

[30] Souder, William E. and Bethay, David: “The Risk Pyramid for New Product Development: An Application to Complex Aerospace Hardware”, Journal of Product Innovation Management 10, pp. 181-194 (1993)

第4章　設計プロセスにおけるイタレーションのモデル理論と実践

4.1　設計プロセスにおけるイタレーションの問題

これまで企業におけるプロジェクト・マネジメントと言えば、建設会社やエンジニアリング会社が主導でノウハウを蓄積してきた工程管理の手法が一般的である。しかしプロダクト・イノベーション(Schumpeter, 1912; 1942)に向けた商品設計には、従来の単純な工程管理手法では解決できない事例が散見されている。その要因とされるのが、特にコンピュータ設計やソフトウエア開発に代表されるハイテク商品のエンジニアリング設計工程に存在する、フィードバック(feedback)とその反復プロセスといったイタレーション(iteration)である(Kaufman, 1989; Rodgers & McCulley, 1996; Wileden, 1986)。

設計プロセスにおけるイタレーションについての研究は既に行われている。von Hippel (1994)は新製品・新しいサービスの設計開発の現場においては、イノベーションに関連した問題解決に向けて、ユーザとサプライア間で予め予見していた知識情報（information）のイタレーションが観測されると指摘している(von Hippel, 1994)。Fujimoto & Clark (1991)らは、情報資産系統図というフレームワークを用いて、エンジニアリング設計工程の流れを情報の観点で分析し、技術者の技術情報の交換と修正作業事例を述べている。また Simon (1969)は問題の認識、選択肢の生成、評価、受容又は拒否の意思決定の流れを一つの問題解決サイクルと定義している。そしてこのサイクルの繰り返しによって、問題とその解決法に関する知識が時間の経過とともに増えていき、これを学習プロセスとしている(Simon, 1969)。すなわちイタレーションによって学習効果が得られるとしているのである。

しかし一方、イタレーションの問題点も指摘されている。大規模・複雑化しているシステムを構築する上で、設計、検証、評価、プロトタイプ(prototype)開発のための作業量と時間が増えており(Thomke, 1998)、そのイタレーションが設計プロセスに与える影響も大きくなっている。イタレーションの回転の数と回転速度が製品市場での成功の一つの決定因子につながっている(Kaufman, 1989)とされるように、商品開発サイクルの視点から見ればイタ

レーションを減少させることが求められている(Smith & Eppinger, 1993)。すなわち、プロダクト・イノベーションの創出に有効なイタレーションや学習効果が得られるイタレーションは奨励すべきであるが、無意味なイタレーションは排除すべきと言える。

この目的を達成するには、イタレーションの本質を理解することとイタレーションを制御することである。すると不必要なイタレーションを排除する意思決定が可能になる。そして同時に製品開発に有効と考えられるイタレーションを特定し、次にそのイタレーションの回転の数を少なくするか、又は回転速度を上げることを思考すれば良い。そのためにイタレーションを必要とする工程群を特定化し、そのモデリングが必要と言える。

これまで長らくプロジェクトの計画・管理に使われてきた代表的な手法であるガント・チャート(Gannt, 1910)、PERT (Project Evaluation and Review Technique)や CPM (Critical Path Methodo)では、グラフやネットワーク図で示された膨大な作業工程や全体図を把握することが困難であった。ガント・チャートの問題点は（i）変化・変更に弱く、（ii）問題点が判りにくい、（iii）総所要期間の見積り精度がわからないといったことであり、それらが解決されずに現在まで利用されてきている(Eppinger, 2001)。PERT はガント・チャートで指摘した問題点を解決する方策も用意されているが、計画策定の原則は極めて単純であり、いつ、どこで作業を行うかを決めるだけである。すなわちガント・チャートも PERT も既に完了している工程の変更は不可能であり、半導体設計やソフトウエア開発（Ramashwamy, 2001;Jalote, 2002; 菅野, 2003）といった商品開発に必要なイタレーションを扱う機能が欠如している。

設計構造マトリックス(Design Structure Matrix:以下 DSM と称す)は、こういった問題点を解決してくれる手法として注目されている。DSM はイタレーションを容易に表現できるツールであるだけでなく、プロジェクトの解析に利用することもできる。

そこで本研究では、イタレーションを含んだ設計工程を DSM 上にモデリングし、その構成要素となるタスク（task）間の依存度がプロジェクト全体に与える影響について数値解析を行う。数値解析にあたっては、一対比較行列(Work Transformation Matrix) (Smith & Eppinger, 1997a; 1997b; 1997c)をネットワーク・プロセッサ半導体の実設計開発事例に適用して行う。そして一対比較行列がそのプロジェクト中のイタレーションの本質を解析でき、技術的問題の特定と解決につながることを検証する。

4.2　イタレーションのモデリング・アプローチ

エンジニアリング設計においては、将来的な原価見積りやスケジュール結果を解析するための様々なアプローチが提唱されてきた。特にガント・チャートは歴史的に利用されてきた。一方、現在まで工程管理には広く認知され、様々なプロジェクト管理として利用されている PERT は、ネットワーク理論に基づいたモデリング技術である。1958 年に米国で当時の技術革新時代の深刻な悩みを解決するために生み出された（森, 1964）。ミサイルの

研究・開発、原子炉の建設、心臓外科手術、ブロードウェーのショー演出等に至るまで新規プロジェクトの計画、実施に幅広く応用され、総所要期間の短縮や総原価費用削減などの効果を上げた（森, 1965）。PERT はネットワーク的視点から、複雑化した設計工程をアクティビティー(activity)という工程作業それぞれの関係性を明示できうる点でその有効性があると解釈されてきた（森, 1964）。更にオペレーション・リサーチ系の研究者、モデリング技術者、そしてリスク・アナリスト等によって多くの時間と費用といった資源が投入され、PERT は洗練されてきた。特に、アクティビティーのネットワークから最短の経路を見つけうる CPM は、今や PERT の代表的な機能展開と言える。PERT やフローチャート化といった一般的手法は、今も伝統的かつ引き続き定型の工程を扱う現場には、最適な標準ツールと言える。

しかしながら PERT はアクティビティー間のフィードバックが起きないという前提で設計されたものである。そこでフィードバックを含むアクティビティー間のイタレーションを実現できるモデリングが検討され、GERT (Generalized Evaluation and Review Technique)が開発された（Moore & Clayton, 1976)。この技術手法は、イタレーションを取り扱うことができる他にシミュレーションでの解析を主たる目的としている(Neumann, 1990; Taylor, 1980)。PERT や CPM はフィードバックが規則上許されていないが、GERT (Moore & Clayton, 1976; Neumann & Steinhardt, 1979; Neumann, 1990)はプロジェクト・マネジメントの基本的な PERT にフィードバックの機能を取り入れたグラフ理論、ネットワーク理論の考え方であり、研究開発や製造工程管理、そして新製品投入に向けたマーケティング等への適用がされている。Q-GERT (Queuing-Generalized Evaluation and Review Technique)は GERT を拡張させた手法である(Pritsker, 1977)。GERT にキューイング(queuing)機能と決定(decision)機能を拡張させた。キューイングとは、相手側の処理の完了を待つことなく、次の処理を行うためのデータ交換方式である。送信側アプリケーションは、データを一時的に保管するためのキュー(queue)と呼ばれるデータを一時的に保管するための待ち行列ファイルに送信データを格納するだけで次の処理を実行できる。この様に、Q-GERT を含む GERT モデルは、スケジュールやコストなどの解析も実行できるように拡張されてきた。しかしそれらモデルは、まさに活性化されたアクティビティーを同時実行や、予見できない修正作業等を考慮していない。更に近年大規模・複雑化しているプロジェクトへの対応力も限界がある。例えばアクティビティー間の相互依存性を全て可視的にそして一覧性を持たすことは難しい。

一方、企業の業務改革の推進と情報システム統合を目的とした IDEF (Integration Definition for Function Modeling)は、アクティビティーの双方向性を表現できるビジネス・モデリング手法である。IDEF 手法は 1970 年代初期に米国空軍の ICAM (Integrated Computer Aided Manufacturing)プロジェクトで空軍の調達機材の製造にコンピュータを活用して効率化を図ろうとした結果からまとめられた(Marca & McGowan, 1988)。現在 IDEF には、アクティビティ・モデリング手法を表す IDEF0、情報／データ・モデリング手法である IDEF1/1X、そしてプロセス記述獲得手法である IDEF3 がある。これら IDEF は、アクティビティー間の双方

向性の関係を図面で表現できるが、特にフィードバックとフィードフォワード(feedforward)のイタレーションを明確的に意図したものではない。

上記の議論をまとめると、基本的には PERT はフィードバックを記述することが許されておらず、イタレーションは想定すらしてない。そして IDEF は、アクティビティー間の双方向性の関係付けは行うことができるが、フィードフォワードやフィードバックを明確に規定することはできない。PERT も IDEF も潜在的にアクティビティー間のイタレーションを想定して開発はされていないのが実態である。一方、PERT の拡張モデルである GERT と GERT の拡張モデルである Q-GERT はイタレーションが許されている。しかし GERT も Q-GERT もシミュレーションを主体としており、その点で大きな有効性を見るが、アクティビティーが大規模化そして複雑化した時のグラフ構造やそのプロジェクトのサイクル時間への洞察性に限界がある。そしてアクティビティーの順序も固定化してしまい、アクティビティー間の配列を柔軟に修正することが難しい。

4.3　DSM

4.3.1 DSM の先行研究

行と列で構成された一覧性の高いマトリックス法によるモデリングを設計工程に適用したのは、Warfield (1973)が始まりである（Warfield, 1973）。次に Steward (1981)が設計工程のモデリングの定義付けと適用手法を設計構造マトリックス(Design Structure Matrix:以下 DSM と称す)として表した(Steward ,1981a & 1981b)。アメリカ航空宇宙局(National Aeronautics and Space Administration: NASA)では 1989 年に大規模で複雑化している設計工程内の相互依存を解析する為に、DSM の機能を NASA で開発したナレッジベース(knowledge-based)の設計管理ソフトウエアに組み込んでいる(Rodgers & McCulley 1996)。そして 1990 年代に入り、DSM はマサチューセッツ工科大学(Massachusetts Institute of Technology)及びイリノイ大学(University of Illinois at Urbana-Champaign)を中心として研究が進められ、米国の軍事・宇宙産業、航空機、航空機エンジン、自動車の分野で成果が報告されている（Browning, 1998a）。

近年 DSM によるモデリングの研究が重ねられている。Ahmadi & Wang (1994) は、タスクに習熟曲線の影響を与えるモデリングを実現している。Carrascosa et al. (1998)は、DSM モデルのタスクがオーバーラップ (overlap) している場合の Loch & Teerwiesch (1998)のモデルを拡張し、その代替案により様々に異なるプロジェクトの所要時間が得られることを確認している。またフィードバックを減少させる為の DSM の最適化についての研究が重ねられている（Denker & Steward, 1995; Kusiak et al., 1994; Rodgers & McCulley, 1996; Steward, 1981a, 1981b）。Browning (1998)は、イタレーションを前提としたプロジェクトにおいて、不確実性が高くなるスケジュールの工程時間を見積もる方法や修正作業や修正にともなうプロセスをストカスティックに表現させ、その影響度合いを考察している(Browning, 1998b)。Yassine

& Braha (2003)は、大規模複雑化しているエンジニアリング工程においてコンカレント・エンジニアリングの有効性を説くが、その適用にあたってイタレーションのみならず、オーバーラップ、分割（decomposition）と統合(integration)、収束(convergence)といった四つの問題点が普及を妨げているとしている。そして DSM にこれらの問題を解決した統合モデルを提唱している(Yassine & Braha, 2003)。Yu et al. (2003) は DSM 上で、モジュラ型 (Modular) のアーキテクチャ (Architecture) を遺伝的アルゴリズム (genetic algorithm)によって、最適なクラスタ化を実現できることを報告している。

4.3.2 半導体設計への適用の意義

半導体を組み込んでシステム機器にするメーカは、最終商品が一般市場において商品競争で優位に立ちたいと考える。本章で実設計開発事例として取り上げるスイッチング用半導体が利用される最終商品は、電子交換機に置き換わるシステムであり、従ってこの半導体には極めて高いパフォーマンスが必要とされる。半導体設計者に突きつけられる問題は、如何に高速スピードの性能を実現したら良いのか、如何に消費電力を削減したら良いのか、そして同時に如何に低価格を実現したら良いのかに帰着される。すなわち、半導体設計者は、過去に経験してきた設計工程上では常に、半導体製造技術の限界までのクロック・スピードを求め、低電圧と低電流の実現、そして半導体チップの面積を小さくすることに注力している。半導体製造技術の制約の中で、全ての最適化を実現化させるために設計プロジェクトではイタレーションが頻繁に行われている。この設計工程上のイタレーションの問題を半導体設計者は、どの様に解決していくかが求められているのである。

半導体設計は過去、システム設計を進めていた時点から設計技術上の問題と解決が繰り返されてきた。近年複雑化・大規模化する半導体に対する設計手法の解決方法として、設計の抽象性のレベルを高めていくシステム設計思考が適用されてきた(Mead & Conway, 1979)。このシステム設計思考によりシステムのアーキテクチャからシリコンへ、すなわち設計レベルを上流から下流に向けて具現化を進めるトップダウン設計手法が確立された(Armstrong, 1989)。またその確立に伴い設計資産や知識情報が長年に渡って蓄積され、その設計手法も定着してきた。従ってこれから半導体設計手法の本質は急激に大きな変化をしていくとは考えられない。しかしその中で現状では設計工程におけるイタレーションに対する問題提起はされるものの、明確な解釈等もなく十分な議論・検討もなされてこなかった。先行研究では米国の軍事・宇宙産業、航空機、航空機エンジン、自動車、自動車エンジンへの適用が例証される中、エレクトロニクス分野への適用がなく、未だ日本での研究成果は報告されていない。従って業界での慣行や事例研究の蓄積も少ない半導体設計におけるイタレーションについての具体的な解釈、そして適用のあり方についての考察は、少なからず役に立ち、十分に意義があるものと言える。

4.3.3 DSM によるモデリング

前述で議論した通り、設計工程のモデリングの原点としては、グラフ理論やネットワーク理論に基づく図表による表現であった。一例としてまず図 4.1 で示した、タスク A とタスク B から構成される設計工程システムを考えてみる。タスクとはプロジェクト業務の作業要素である。タスクは作業内容を示すと共に、時間量、費用、リソース(resource)の情報を含む。あるタスクからイベント(event)を経てもう一つのタスクへ及ぼす影響については、単なる結合線を用いるのではなく、矢印を用いることでタスク相互の依存性を説明することができる。すなわちタスク A とタスク B が存在する場合、その二つのタスクとそれらの間の依存性によって設計工程全体の構造と動作の説明表現ができる。

設計工程で考えられるタスク間の依存性は、三つのモデルすなわちシーケンシャル(sequential)、コンカレント(concurrent)、相互依存(coupled)で代表できる。ネットワーク及び DSM によってそれぞれ表される三つのモデルを図 4.2 に示す(Eppinger, 1991; Smith, 1992)。個々のタスクの動作を理解することで、設計工程全体の動作を理解できる。まずシーケンシャルでは一つのタスクが別のタスクの動作あるいは決定に影響を及ぼすものであり、関係は一方向である。このモデルでは、タスク B を開始する前に、タスク A を実行しておかなければならず、タスク B は、タスク A の設計因子の影響を受ける。コンカレントの場合にはタスク間に依存関係の作用はない。相互依存の場合には、タスク間の関係はそれぞれ絡み合ったものであり、双方のタスクが影響を及ぼし合っている。双方のタスクは、影響を受ける設計因子の量を判らずに、不確定な決定をしなければならない。このサイクルの状況が、イタレーションである。

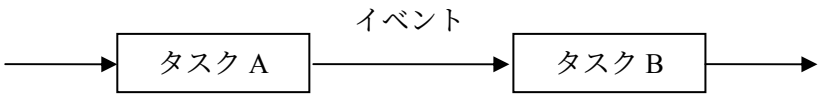


図 4.1　ネットワーク図

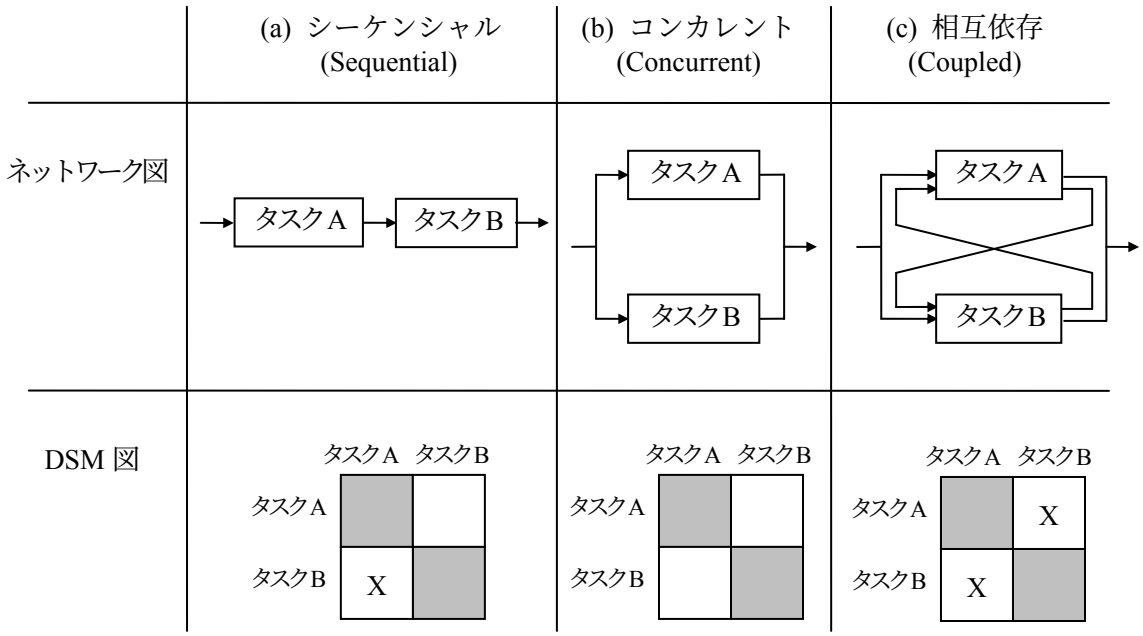


図 4.2　ネットワーク図と DSM 図によるタスク A とタスク B の三つのモデル

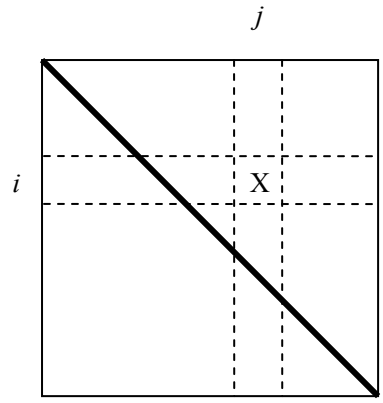


図 4.3　タスク *i* とタスク *j* の依存性

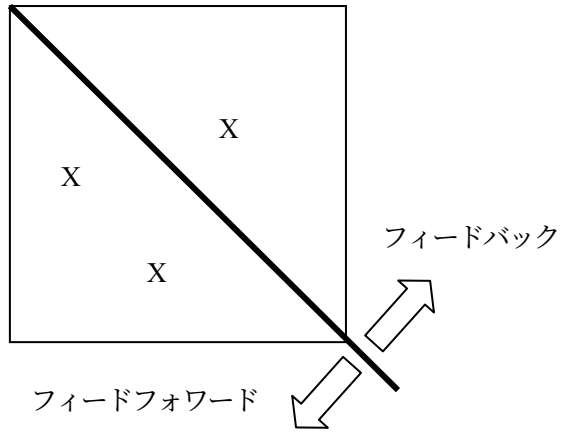


図 4.4　フィードフォワードとフィードバック

DSM によるモデルでは、全てのタスクが一つずつ正方行列の行と列に配置され、行と列が交差する枡の中でそれぞれのタスク間の依存性が表現される。これを図 4.3 に示した。タスクの位置づけは、設計工程の時間軸に従って行の上流から下流へと並べ、タスクの行の順番（上流から下流の方向）とタスクの列の順番（左側から右側の方向）は同じにならなければならない。行列の第 i 行、第 j 列に交差する部分にある(i, j)成分に X（この X は二進数の 1 と表すこともできる）がある場合には、行列の第 i 行にあるタスクから、行列の第 j 列にあるタスクの方向に向かって依存関係があることを表現できる。逆に(i, j)成分には何も存在しない（この状況は二進数の 0 と表すこともできる）なら、行列の第 i 行にあるタスクから、行列の第 j 列にあるタスクの方向に向かっての依存関係はない。

また DSM において、ある列のタスクに注目するなら、その列のタスクを実行してから、次にどのタスクを実行しなければならないかが判る。またある行のタスクに注目するなら、その行のタスクはどのタスクから情報を得ていることが判る。タスク間における依存の方向性については、行列の対角成分を基準境界線として容易に理解できる。これを図 4.4 に示した。すなわち行列の対角成分より下に X があれば、タスクが上流から下流へのフィードフォワードであり、対角成分より上に X があれば、上流のタスクが下流のタスクに依存するフィードバックを表す。

この様に、設計工程に二進数の行列を適用することでタスク間の依存関係の有無と方向性を DSM によって表すことができる。DSM は設計工程全体を表として一覧できるので、タスクの多さにかかわらず容易にかつ明確に判読できる。そしてこの行列を線形写像として利用することが可能であり、その有用性を考察していく。

4.4　一対比較行列によるモデリング

図 4.2 で示した通り設計工程上で代表されるタスクの三つの基本モデルを含んだ設計プロジェクトを解析するに当たり、タスク間の依存関係の度合いを変数として与えた DSM によるニューメリカルのモデリングが一対比較行列(Work Transformation Matrix)である(Smith & Eppinger, 1997a; 1997b; 1997c)。一対比較行列に線形代数を援用して、設計工程の総所要時間量のベクトル $\boldsymbol{U}$ を算出し、これにより設計プロジェクトの解析を行うことができる。ここでは一対比較行列によるモデリングと、一対比較行列と総所要時間量ベクトル $\boldsymbol{U}$ の関係式について示す。

まず n 次のベクトル $\boldsymbol{u}_t$ を定義する。ベクトル $\boldsymbol{u}_t$ の成分は、n 個のタスクの実行時間である。そして t 回目のイタレーションが終了し、$(t+1)$回目のイタレーションで実行されるタスクのベクトルが $\boldsymbol{u}_t$ である。t 回目のイタレーションが終了した段階では、ベクトル $\boldsymbol{u}_t$ の成分のタスクを次の$(t+1)$回目のイタレーションの段階で全て終了させなければならない。ここであるタスクと依存性がある別のタスクに対してフィードバックや修正作業といった仕事を生み出す。ベクトル $\boldsymbol{u}_t$ を次式に示す。ここで $\boldsymbol{u}_0$ は、予めイタレーション等を想定しな

い、当初計画する各タスクの予定実行時間を成分としたベクトルである。

$$\boldsymbol{u}_0 = \begin{bmatrix} u_{10} \\ u_{20} \\ \vdots \\ u_{n0} \end{bmatrix}, \boldsymbol{u}_1 = \begin{bmatrix} u_{11} \\ u_{21} \\ \vdots \\ u_{n1} \end{bmatrix}, \boldsymbol{u}_2 = \begin{bmatrix} u_{12} \\ u_{22} \\ \vdots \\ u_{n2} \end{bmatrix}, \cdots, \boldsymbol{u}_t = \begin{bmatrix} u_{1t} \\ u_{2t} \\ \vdots \\ u_{nt} \end{bmatrix} \tag{4.1}$$

次に Smith & Eppinger (1997a)に基づく一対比較行列のモデリングを利用する。一対比較行列はタスク間の依存性の度合いを変数とし、その変数を行列成分に表現させるものである。タスクが n 個存在している場合には、この変数を成分として持つ n 次の正方行列 $\boldsymbol{A}$ として捉える。この時、$\boldsymbol{A}$ の行列成分 a_{ij} はタスク j のタスク i に対する依存性の度合いを示す。タスク j はタスク i によって、タスク j 当たり a_{ij} の仕事量を生み出すと仮定する（ただし、$1 \le i \le n$、$1 \le j \le n$、$0 \le a_{ij} < 1$）。一対比較行列 $\boldsymbol{A}$ の対角成分は全て 0 となる。尚、以下本論で利用するこの仕事量とはタスクの「のべ時間量」と考えて議論を進めている。一対比較行列 $\boldsymbol{A}$ は以下の通りである。

$$\boldsymbol{A} = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \tag{4.2}$$

一対比較行列 $\boldsymbol{A}$ とベクトル $\boldsymbol{u}_t$ の関係は、先の定義から次式で示される。

$$\boldsymbol{u}_{t+1} = \boldsymbol{A}\boldsymbol{u}_t \tag{4.3}$$

又は、$\boldsymbol{A}$のべき乗すなわち $\boldsymbol{A}^t$ を用いると次式で示される。

$$\boldsymbol{u}_t = \boldsymbol{A}^t \boldsymbol{u}_0 \tag{4.4}$$

これらから、M 回のイタレーション後のプロジェクトの総所要時間量ベクトル $\boldsymbol{U}$ は次式で表される。

$$\boldsymbol{U} = \sum_{t=0}^{M} \boldsymbol{u}_t = \sum_{t=0}^{M} \boldsymbol{A}^t \boldsymbol{u}_0 = (\sum_{t=0}^{M} \boldsymbol{A}^t)\boldsymbol{u}_0 \tag{4.5}$$

4.5　一対比較行列によるモデルの固有値、固有ベクトル、総所要時間量ベクトル //

4.5.1　一対比較行列の固有値と固有ベクトル

次に一対比較行列 $\boldsymbol{A}$ が相異なる n 個の固有値 $\lambda_1, \ldots, \lambda_n$ を持つとき、それぞれの固有値に対応する固有ベクトルを $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ とする。固有ベクトル $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ を基底として並べた行列を $\boldsymbol{S}$ すなわち $\boldsymbol{S}=[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$とするなら、以下の関係式が成り立つ（竹内 1974）。尚、行列 $\boldsymbol{S}$ には逆行列 $\boldsymbol{S}^{-1}$ が存在するとする。

$$\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{4.6}$$

式(4.6)の右辺は固有値 $\lambda_1,\ldots,\lambda_n$ を対角成分に持つ対角行列（対角成分以外の成分が全て 0 である）であり、これを $\boldsymbol{\Lambda}$ とする。式(4.6)を変形すると $\boldsymbol{A}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{\Lambda}$ となり、さらに、

$$\boldsymbol{A} = \boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{S}^{-1} \tag{4.7}$$

と固有値分解が得られる。これを $\boldsymbol{A}^t$ を用いて変形すると、

$$\boldsymbol{A}^t = \boldsymbol{S}\boldsymbol{\Lambda}^t\boldsymbol{S}^{-1} \tag{4.8}$$

となる。これを式(4.5)に代入すると、総所要時間量のベクトル $\boldsymbol{U}$ は次式で表される。

$$\boldsymbol{U} = (\sum_{t=0}^{M}\boldsymbol{A}^t)\boldsymbol{u}_0 = \boldsymbol{S}(\sum_{t=0}^{M}\boldsymbol{\Lambda}^t)\boldsymbol{S}^{-1}\boldsymbol{u}_0 \tag{4.9}$$

この式から考察されることは、行列 $\boldsymbol{\Lambda}$ は行列一対比較行列 $\boldsymbol{A}$ の固有値を成分とする対角行列であるので、$\boldsymbol{A}$ の固有値の絶対値が 1 より小さい場合には、総所要時間量のベクトル $\boldsymbol{U}$ は収束するが、少なくとも一つの固有値の絶対値が 1 より大きい場合には、ある一つのタスクがイタレーションによって、そのタスク以上の仕事量を引き起こし、総所要時間量のベクトル $\boldsymbol{U}$ は発散するということである。前者においてシステムは安定であり、後者においてシステムは非常に不安定である。後述の半導体設計事例のモデリングでは、一対比較行列 $\boldsymbol{A}$ の成分は全て 1 より小さく、固有値が 1 より小さいので、安定性については十分条件を満たしていると言える。

安定性を満たすこという条件下において、式(4.9)右辺の括弧内について、$M \to \infty$ と考えた場合、

$$\lim_{M\to\infty}\sum_{t=0}^{M}\boldsymbol{\Lambda}^t = (\boldsymbol{I}-\boldsymbol{\Lambda})^{-1} \tag{4.10}$$

と収束する。ここで $\boldsymbol{I}$ は n 次の単位行列である。そして、これにより総所要時間量のベクトル $\boldsymbol{U}$ は次式で表すことができる。

$$\boldsymbol{U} = \boldsymbol{S}(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}\boldsymbol{S}^{-1}\boldsymbol{u}_0 \tag{4.11}$$

式(4.11)内の $(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}$ は対角行列であり、その対角成分は $1/(1-\lambda_n)$ である。固有値 λ_n が実数でかつ上述の通りその絶対値が 1 より小さい場合、すなわち $-1<\lambda_n<1$ の条件では、固有値 λ_n の増加とともに $1/(1-\lambda_n)$ は増加する（$\lambda_2<\lambda_1<1$ であるなら $0<1/(1-\lambda_2)<1/(1-\lambda_1)$）。そして λ_n が大きくなれば総所要時間量のベクトル $\boldsymbol{U}$ のタスク相当成分も大きくなる。ここで解釈できることは、固有値 λ_n が正でかつ大きい場合には、それに関連するタスクは、プロジェクト全体への仕事量の寄与が高いということである。固有値 λ_n が負の場合には、$1/(1-\lambda_n)$ は大きくないので、総所要時間量のベクトル $\boldsymbol{U}$ に与える影響度は小さい。

ここで固有値が複素数の場合を考察していく。$\lambda_n = \alpha + \beta i$ として、まず $1/(1-\lambda_n)$ の絶対

値を考えてみる。

$$\left|\frac{1}{1-(\alpha+\beta i)}\right| = \frac{1}{\sqrt{(1-\alpha)^2+\beta^2}} \tag{4.12}$$

ここで $\alpha<1$、$\beta\neq 0$ であるので、次式が成り立つ。

$$\frac{1}{\sqrt{(1-\alpha)^2+\beta^2}} < \frac{1}{\sqrt{(1-\alpha)^2}} = \frac{1}{1-\alpha} \tag{4.13}$$

次に行列 $\boldsymbol{A}$ の成分の性質上、すなわち先の十分条件から、

$$|\lambda_n| = \sqrt{\alpha^2+\beta^2} < 1 \tag{4.14}$$

である。従って、$\alpha^2+\beta^2<1$ であるので、次式が成り立つ。

$$\left|\frac{1}{1-(\alpha+\beta i)}\right| = \frac{1}{\sqrt{(1-\alpha)^2+\beta^2}} = \frac{1}{\sqrt{1-2\alpha+\alpha^2+\beta^2}} > \frac{1}{2-2\alpha} \tag{4.15}$$

よって、式(4.13)と式(4.15)から次式を得る。

$$\frac{1}{2-2\alpha} < \left|\frac{1}{1-\lambda_n}\right| < \frac{1}{1-\alpha} \tag{4.16}$$

この様に、複素数の固有値の場合には、$1/(1-\lambda_n)$の絶対値がこの α に関連した上限と下限の範囲であるので、虚部は取り上げる必要性がない。この場合 α が大きくなれば、総所要時間量のベクトル $\boldsymbol{U}$ のタスク相当成分も大きくなり、全体のプロジェクトの仕事量の寄与が高くなる。複素数で実部が負の場合には、$1/(1-\lambda_n)$は大きくないので、総所要時間量のベクトル $\boldsymbol{U}$ に与える影響度は小さい。

上記の議論を踏まえると、総所要時間量のベクトル $\boldsymbol{U}$ に与える影響度を解析するためには、固有値が実数であれ複素数であれ、固有値の正の実数（複素数の場合は、正の実部）に注目し、式(4.11)内の $(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}$ の値を算出すれば良い。後述する半導体設計事例における総所要時間量のベクトル $\boldsymbol{U}$ については、この $1/(1-\lambda_n)$ を $1/(1-\mathrm{Re}[\lambda_n])$と考えて考察を進めることとする。また、固有ベクトルにも注目する必要がある。式(4.11)内の $\boldsymbol{S}$ は基底に一対比較行列 $\boldsymbol{A}$ の固有ベクトルを持つ行列である。$(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}$ は対角行列であるので、この固有ベクトルの重み付けが各タスクの仕事量に寄与するからである。

4.5.2　総所要時間量ベクトル *U* の計算手順

一対比較行列の固有値及び固有ベクトルと総所要時間量ベクトル $\boldsymbol{U}$ の関係式(4.11)の解釈の為に、単純な 4 次(4×4)の正方行列の例を示す。ここではタスクの依存性は、0.5、0.4、0.3、0.2、0.1、0.05 の 6 つの異なるレベルを想定する。一対比較行列 $\boldsymbol{A}$ を以下の通りとする。

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0.5 & 0.2 & 0.3 \\ 0.3 & 0 & 0.4 & 0.05 \\ 0.5 & 0.1 & 0 & 0.2 \\ 0.05 & 0.2 & 0.1 & 0 \end{bmatrix} \tag{4.17}$$

行列 $\boldsymbol{A}$ の 4 個の固有値 λ_n (n: 1,2,3,4)は、0.7548、−0.3199+0.2427i、−0.3199−0.2427i、−0.1149 であり、これら固有値を対角成分として持つ対角行列（対角成分以外の成分が全て 0 である）を行列 $\boldsymbol{\Lambda}$ とすると、行列 $\boldsymbol{\Lambda}$ は以下の通り算出される。

$$\boldsymbol{\Lambda}=\begin{bmatrix} 0.7548 & 0 & 0 & 0 \\ 0 & -0.3199+0.2427i & 0 & 0 \\ 0 & 0 & -0.3199-0.2427i & 0 \\ 0 & 0 & 0 & -0.1149 \end{bmatrix} \tag{4.18}$$

次にそれぞれの固有値に対応する固有ベクトルを基底（行列の列に配置）として並べた行列 $\boldsymbol{S}$ は以下の様になる。

$$\boldsymbol{S}=\begin{bmatrix} 0.5992 & -0.3404+0.3763i & -0.3404-0.3763i & -0.2683 \\ 0.5386 & -0.1389-0.4765i & -0.1389+0.4765i & -0.5063 \\ 0.5353 & 0.6756 & -.6756 & 0.2492 \\ 0.2533 & -0.1602+0.1175i & -0.1602-0.1175i & 0.7806 \end{bmatrix} \tag{4.19}$$

4 個の固有値のうち最大固有値 0.7548 に対応する固有ベクトルと、二番目に大きい固有値−0.1149 に対応する固有ベクトルはそれぞれ、

$$\begin{bmatrix} 0.5992 \\ 0.5386 \\ 0.5353 \\ 0.2533 \end{bmatrix}、\begin{bmatrix} -0.2683 \\ -0.5063 \\ 0.2492 \\ 0.7806 \end{bmatrix}$$

である。これらが総所要時間量のベクトル $\boldsymbol{U}$ に大きなインパクトを与えている特性を持っており、この特性がイタレーションを引き起こしている問題である。この特性を読みとり、問題点として抽出し、問題解決に向けて考察することがプロジェクトを解析する上で必要なのである。

ワークベクタ $\boldsymbol{u}_t$ はイタレーションが重ねられるに従い、次式の様に推移する。

$$\boldsymbol{u}_0=\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},\boldsymbol{u}_1=\begin{bmatrix} 1 \\ 0.75 \\ 0.8 \\ 0.35 \end{bmatrix},\boldsymbol{u}_2=\begin{bmatrix} 0.64 \\ 0.63 \\ 0.64 \\ 0.28 \end{bmatrix},\boldsymbol{u}_3=\begin{bmatrix} 0.53 \\ 0.46 \\ 0.43 \\ 0.22 \end{bmatrix},\boldsymbol{u}_4=\begin{bmatrix} 0.38 \\ 0.34 \\ 0.35 \\ 0.16 \end{bmatrix} \tag{4.20}$$

このワークベクタの推移を観測することで、各タスクの収束性と格差が容易に推測できる。

$(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}$ は以下の通りである。

$$(I-\boldsymbol{\Lambda})^{-1}=\begin{bmatrix} 4.079 & 0 & 0 & 0 \\ 0 & 0.7328+0.1347i & 0 & 0 \\ 0 & 0 & 0.7328-0.1347i & 0 \\ 0 & 0 & 0 & 0.8968 \end{bmatrix} \tag{4.21}$$

$(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}$ の対角成分の正の実数部が大きければ、総所要時間量のベクトル $\boldsymbol{U}$ への寄与は

大きくなる。式(4.21)では 4.079 が大きく寄与することになる。すなわち式(4.19)の大きな固有値に対応した固有ベクトルを捉えて、各タスクの重み付け（数値の格差）から特性を類推すれば良い。

総所要時間量のベクトル $\boldsymbol{U}$ 内の $(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}\boldsymbol{S}^{-1}\boldsymbol{u}_0$ は以下の様になる。

$$(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}\boldsymbol{S}^{-1}\boldsymbol{u}_0=\begin{bmatrix}8.175\\-0.1500+0.1222i\\-0.1500-0.1222i\\0.5570\end{bmatrix} \tag{4.22}$$

実際の所、式(4.22)で判る通り、8.175 が最大固有値に対応した固有ベクトル（第一固有ベクトル)に大きなインパクトを与えていることが判る。

最終的に総所要時間量のベクトル $\boldsymbol{U}$ は以下の様になる。

$$\boldsymbol{U}=\boldsymbol{S}(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}\boldsymbol{S}^{-1}\boldsymbol{u}_0=\begin{bmatrix}4.759\\4.279\\4.312\\2.525\end{bmatrix} \tag{4.23}$$

4.5.3　固有値問題

ここで、固有値問題に関する研究について整理する。イタレーションの収束性のみならず、その他様々な事象が固有値問題として捉えられ、研究されている。

前項までで、一対比較行列 $\boldsymbol{A}$ を固有値問題として捉えるということを行った。これは一対比較行列 $\boldsymbol{A}$ だけではタスク間の依存性しか理解できないからであり、また設計工程全体から見た場合の各タスクに与える全てのイタレーションの影響を把握することやそこから解釈できる本質的特徴は、一対比較行列 $\boldsymbol{A}$ の状態のままでは解釈が難しいからである。一対比較行列 $\boldsymbol{A}$ の固有値と固有ベクトルは、設計工程の定量的な収束性や収束的特性を表すものであり、固有値を抽出することでイタレーションによる問題点の解析を進めることができる(Strang, 1980)。線形動的システムにおいては、固有値と固有ベクトルに着目することで、どこがイタレーションに影響を与えているのか、そして、どう制御しうるかが議論されている（藤井 et al., 2003)。線形動的システムの場合、イタレーションの収束特性が遅い場合には、固有値は極めて大きくなり（Ogata, 1990; Smith, 1992)、その固有値に属する固有ベクトルの重み付けにより、ある特定のタスク群のグループ間でイタレーションが起きていることが解釈できる。

行列や線形作用素の固有値は、理論的な応用にも実用的な応用にも非常に多数登場している（Chatelin, 1988)。例えば量子論における定常状態にある電子、原子、分子などの性質はシュレーディンガーの方程式 $H\psi(r)=\varepsilon\,\psi(r)$の解である波動関数物 $\phi(r)$で表されるが、これは定数 ε、すなわちエネルギー準位がHの固有値となる固有方程式である(小出, 1968)。理論的応用として原子炉の物理（小林, 1996）や量子化学（Davidson, 1983）へと適用されて

いる。他に Chatelin (1988)は、機械構造力学、経済理論、海洋学での適用事例を紹介している。一般的なバネを用いた固有値問題は、解析力学の理論として確立されているが、実際の土木工学において遭遇する一つ例として建築構造学の骨組みの座屈問題がある。振動と座屈とは密接な関係にあり、これが固有値問題の範疇に属する（加藤, 1967)。一つの事例として米国ワシントン州タコマの吊り橋の事故は、空気弾性振動効果に起因していることが構造解析の原理と固有値問題に従って結論付けされている（Smith, 1974)。また音・振動のモード解析と制御の分野では、多自由度系における固有振動数と固有モードの行列式を一般固有値問題と考え、固有値解析を行った(萩原 et al., 1996)。臨床工学の分野では、生命に直結した各種の生命維持管理装置の操作と保守点検といった高度先進医療が不可欠であり、極めて高い信頼性と安定性を持ったシステムが求められている(西村, 1990)。この安定性についても固有値問題が取り上げられ、工学的応用理論として裏付けされている。理論的応用は基本と言えるが、工学的応用もそれに劣らず重要と言える。

4.6　一対比較行列によるモデルの半導体設計事例への適用

4.6.1 半導体設計事例

一対比較行列によるモデリングを半導体の実設計事例に適用し、その有効性を検証する。本設計プロジェクトは著者が経営する企業と大手電機メーカとによる共同プロジェクトであり、インターネットのブロードバンド化を実現させるスイッチ LSI が最終製品である。メモリーを除き、約三百万ゲート規模であり、多ピンのマルチ・プロセッサである。まず設計仕様書、設計工程表等の内部資料を精査し、システム設計者、半導体設計技術者やプロジェクト・マネージャとの面談も重ねた。このプロジェクトは、C 言語を用いてシステムのアーキテクチャとアルゴリズムを創造するチーム、ハードウエア記述言語と設計手法を身につけ、10 年以上の経験を持つ半導体設計者チーム、そして半導体の物理設計に熟知した半導体設計者チームと、プロジェクトを統括するプロジェクト・マネージャから構成される。

本半導体設計工程の一対比較行列によるモデリングにあたり、まず設計工程のタスクを明確化する必要がある。ここはワーク・ブレークダウン・ストラクチャ(Work Breakdown Structure:以下 WBS と称す)に従って予め分割されたタスクを DSM のモデルにあてはめることとした。

次にタスク間の依存の方向性と定量的な大きさを決定した。この作業では、設計データと知識情報の流れを観察し、タスク間の依存性の度合いを三つのレベルでワークシートへ記入するよう、各設計技術者に依頼した。しかしながら、タスク間の相互依存性の度合いを設計工程中に把握することは極めて困難である。そのために設計開発時間が予定以上に超過してしまう場合が多々ある。今までの半導体設計ではこのイタレーションに対する十分な理解がなかったために、タスクがどの程度の冗長性を持ってしまうのか、どの程度の

人的リソースと時間の負荷かかるか明確につかみ取ることができなかったためである。

一方、各タスクのそれぞれの仕事量については正確な数字が得られなかった。もちろん各タスクに配置される人的リソースやスケジュール管理は徹底的に行われてきたが、イタレーションがタスク間で複雑に起きているために、それぞれのタスクの正確な仕事量の測定が行われていなかったためである。

ここまでで得られた半導体設計工程のモデルを図 4.5 に示す。実際モデリングを行っていると、相互依存の関係にあるタスクが数多く発見された。このマトリックスのモデルに注視しただけでは問題の本質の把握が困難である。複雑化した設計工程内のタスクの相互依存関係を理解できるよう、タスクのセットを分割して統合することで問題解決に向けた糸口が見つかる（von Hippel, 1990）と述べられているように、本モデルを相互依存が特に強い関係にあるタスクをグルーピングし、シーケンシャルのタスクとコンカレントのタスクとを切り分けた。このパーティション(partition)には、三角化アルゴリズム(Triangularization Algorithm)（Kusiak at el., 1994）を利用した。相互依存が特に強い関係にあるタスクをパーティショニングして得られたマトリックス・モデルを図 4.6 で示した。

4.6.2　タスク間の依存性の強度

図 4.6 の枠内に示したタスク・グループを抽出し、図 4.7 に示した。これを一対比較行列によりモデリングし、解析を行う。モデリングにあたり、プロジェクトに参画した半導体設計技術者やマネージャに依存性の度合いの決定付けを依頼した。依存性の度合いが強い場合には 0.3 を、中間値を 0.1 に、そして弱い場合を 0.05 とした。ここで依存性の度合いの設定が解析結果に与える影響を考察してみる。解析結果は導出される固有ベクトルから解釈していくので、この依存性の度合いの設定の違いが与える固有ベクトルの重み付けへの影響ついて考えてみる。

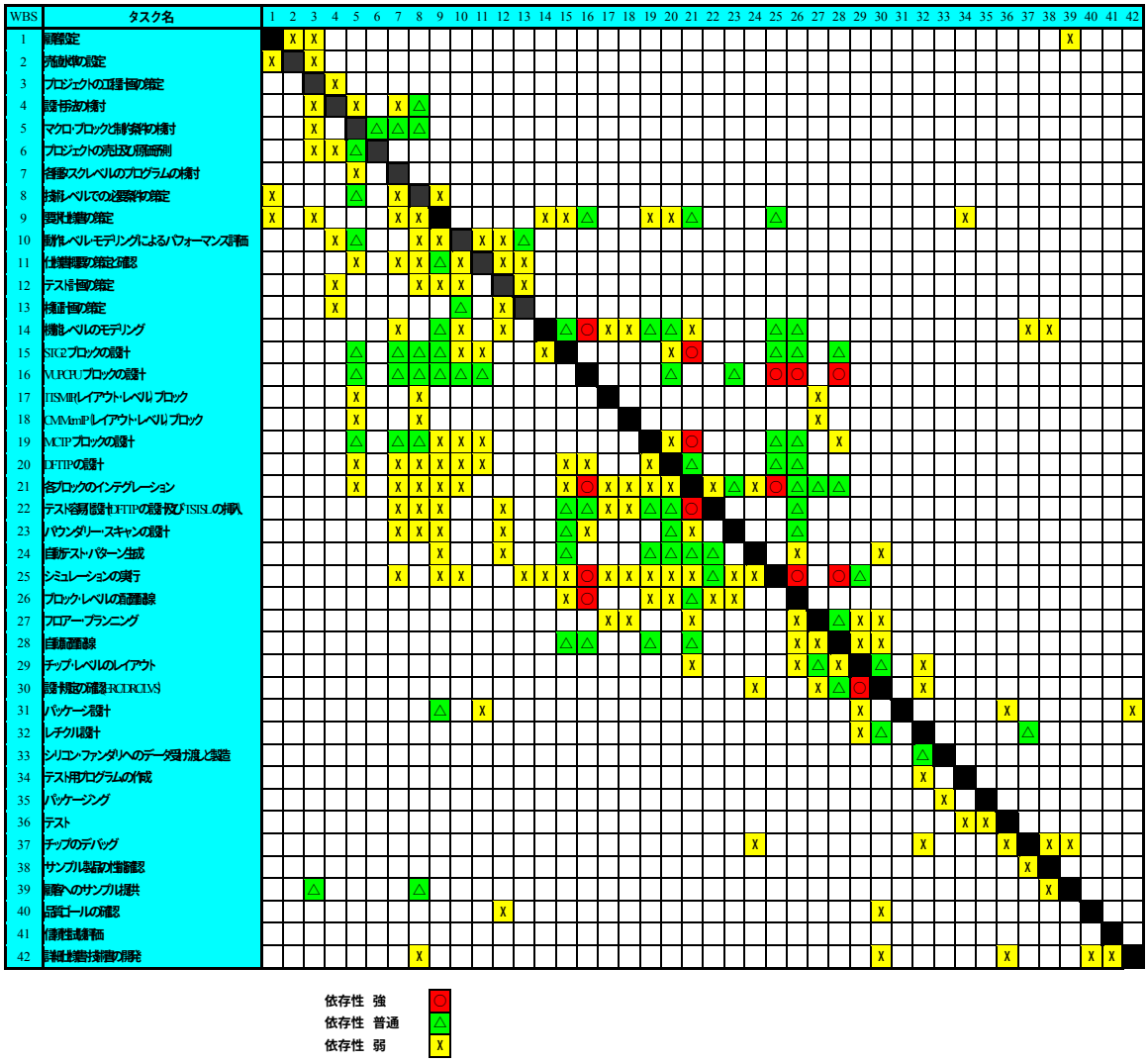

WBS
タスク名
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
1 開発決定
2 予備段階の設定
3 プロジェクトの工程・管理の設定
4 設計手法の検討
5 マクロ・ブロックと制約条件の検討
6 プロジェクトの売上及び原価計別
7 各種ステクレベルのプログラムの検討
8 技術レベルでの必要条件の設定
9 要求仕様の設定
10 動作レベル・モデリングによるパフォーマンス評価
11 性能指標の設定と確認
12 テスト管理の設定
13 検証管理の設定
14 機能レベルのモデリング
15 SIG2ブロックの設計
16 MUPCUブロックの設計
17 ITSMRレイアウト・レベル ブロック
18 CMMmP レイアウト・レベル ブロック
19 MCTPブロックの設計
20 DFTPの設計
21 各ブロックのインテグレーション
22 テスト容易設計DFTPの設計及びTSISLの挿入
23 バウンダリー・スキャンの設計
24 自動テスト・パターン生成
25 シミュレーションの実行
26 ブロック・レベルの配置配線
27 フロアープランニング
28 自動配置配線
29 チップ・レベルのレイアウト
30 設計規定の確認RCDRCLVS
31 パッケージ設計
32 レチクル設計
33 シリコン・ファンダリへのデータ受け渡しと製造
34 テスト用プログラムの作成
35 パッケージング
36 テスト
37 チップのデバッグ
38 サンプル製品の性能確認
39 顧客へのサンプル提供
40 設計ツールの確認
41 信頼性試験評価
42 詳細仕様技術書の開発
依存性 強
依存性 普通
依存性 弱

図 4.5　半導体設計工程

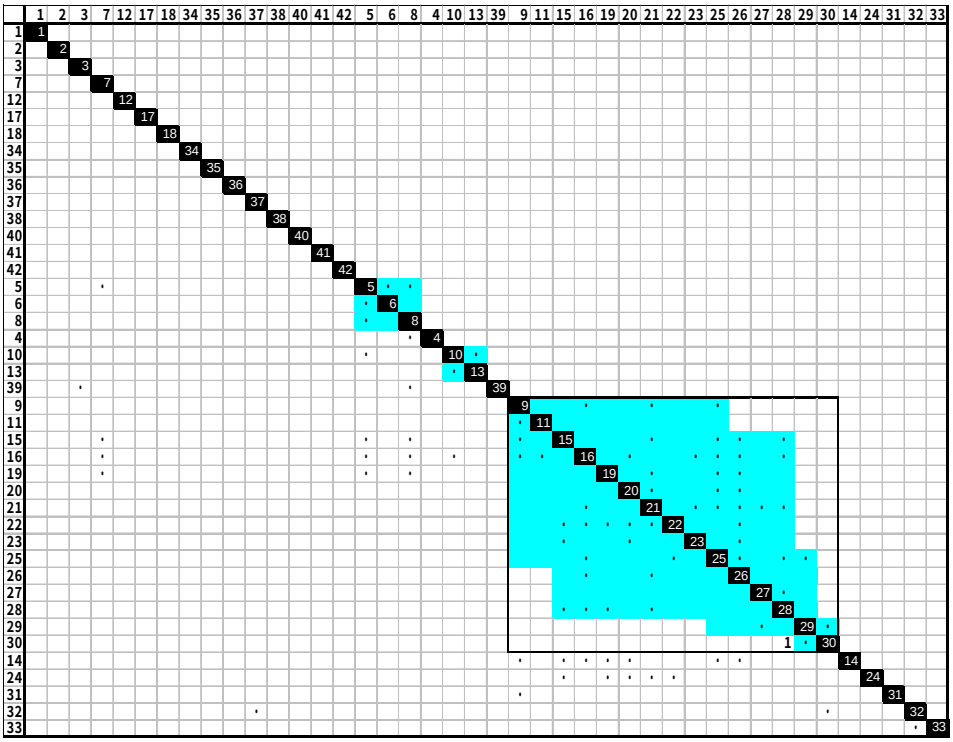


図 4.6　パーティショニングによるタスク・グループの抽出

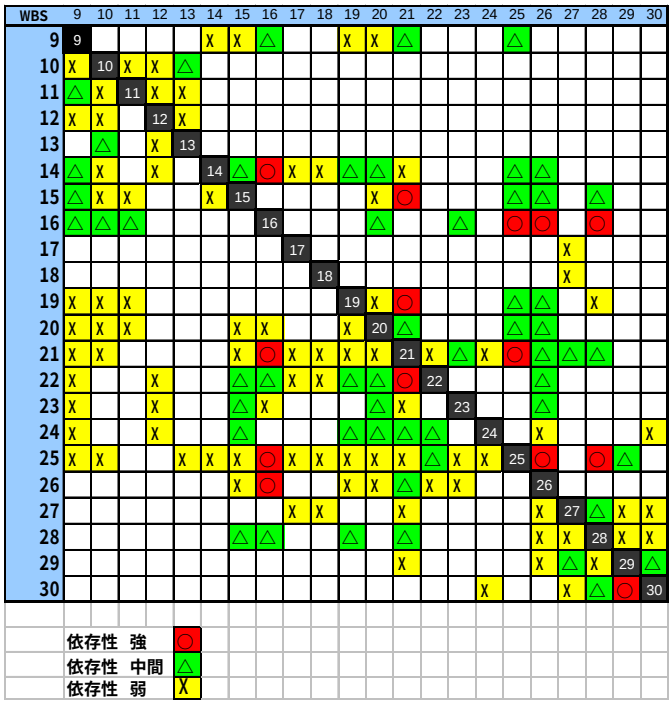


図 4.7　抽出されたタスク・グループ

まず、依存性の度合いの相対的比率が同じ場合を考える。k を定数とすると、式(4.7)から、

$$k\boldsymbol{A} = k(\boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{S}^{-1}) = \boldsymbol{S}(k\boldsymbol{\Lambda})\,\boldsymbol{S}^{-1} \tag{4.24}$$

である。この式から、固有値は k 倍となるが、固有ベクトルは変化しないことが分かる。すなわち、この場合には固有ベクトルの重み付けに影響を与えない。

次に依存性の度合いの相対的比率が異なる場合を考える。ここで α =相互依存性の中間値/相互依存性の弱の値、β =相互依存性の強の値/相互依存性の中間値とする。α と β の値を変えながら異なる度合いの組み合わせ 30 組を準備した。そしてそれらの第一固有ベクトルと第二固有ベクトルを算出し、それら 0.19 以上の得点成分を持つタスクの数を観測した。そして実設計事例の第一固有ベクトルと第二固有ベクトルの 0.19 以上の得点成分を持つタスクの数とそれぞれ対比を行った。タスクの数の差が大きい場合には固有ベクトルの重み付けに大きな影響を与えるが、差がなければほとんど影響を与えないと解釈できる。図 4.8 はその対比結果である。0 を包含した表示図内に収まった α と β であれば重み付けの誤差は許容できる範囲と言える。この結果から、本プロジェクトのモデリングで使用する依存性の度合いは、α =2、β =3 であり、0 を包含した範囲の設定であれば、固有ベクトルの重み付けに大きな影響は出ないことが分かった。

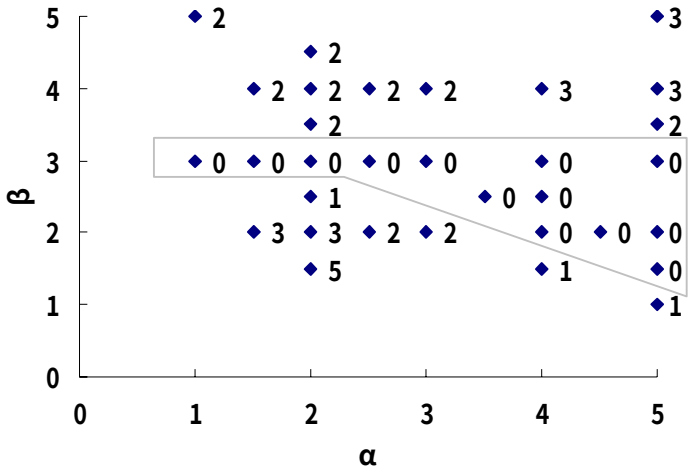


図 4.8　依存性の度合いの相対的比率の違いが与える固有ベクトルへの影響

4.7　解析結果

4.7.1　1/(1-Re[λ_n])、固有ベクトル、総所要時間量ベクトル $\boldsymbol{U}$ の算出結果

一対比較行列 $\boldsymbol{A}$ の固有値に対応した 1/(1-Re[λ_n])は、Wolfram 社製ソフトウエア Mathematica4.2 によって算出した。その結果を図 4.9 に示した。一対比較行列 $\boldsymbol{A}$ の相異なる 22 個の固有値を大きい順に $\lambda_1,\cdots,\lambda_{22}$ とし、その固有値番号を横軸とした。縦軸は 1/(1-Re[λ_n])である。先に議論した通り式(4.11)内の $(\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}$ は対角行列であるから、対角成分 1/(1-Re[λ_n])は固有値 λ_n に対応する固有ベクトルとの乗数効果で総所要時間量のベクトル $\boldsymbol{U}$ のタスク成分へ大きな影響を与える。すなわち 1/(1-Re[λ_n])は、固有値 λ_n に対応する固有ベクトルのタスク成分の実質的なイタレーション回転数と考えることができる。この実質的イタレーション回転数が大きいなら、総所要時間量のベクトル $\boldsymbol{U}$ に与える負荷は大きい。ちなみに 1/(1-Re[λ_1])=6.777、1/(1-Re[λ_2])=1.309、1/(1-Re[λ_3])=1.171 であった。

表 4.1 は、最大固有値 λ_1 に対応した固有ベクトル（第一固有ベクトル）と二番目に大きい固有値 λ_2 に対応した固有ベクトル（第二固有ベクトル）と総所要時間量のベクトル $\boldsymbol{U}$ の算出結果を示している。重み付けが大きいと判断した成分として、第一固有ベクトルは 0.19 以上、第二固有ベクトルは 0.20 以上、総所要時間量のベクトル $\boldsymbol{U}$ は 5.0 以上を太字で強調した。尚、二つの固有値 λ_1 と λ_2 それぞれに対応した固有ベクトルだけを取り上げた理由は図 4.9 で判る通り、大きさが三番目以降の固有値については、1/(1-Re[λ_n])の値すなわち実質的なイタレーション回転数が極めて 1 に近く、総所要時間量のベクトル $\boldsymbol{U}$ に与える影響は大きくないからである。

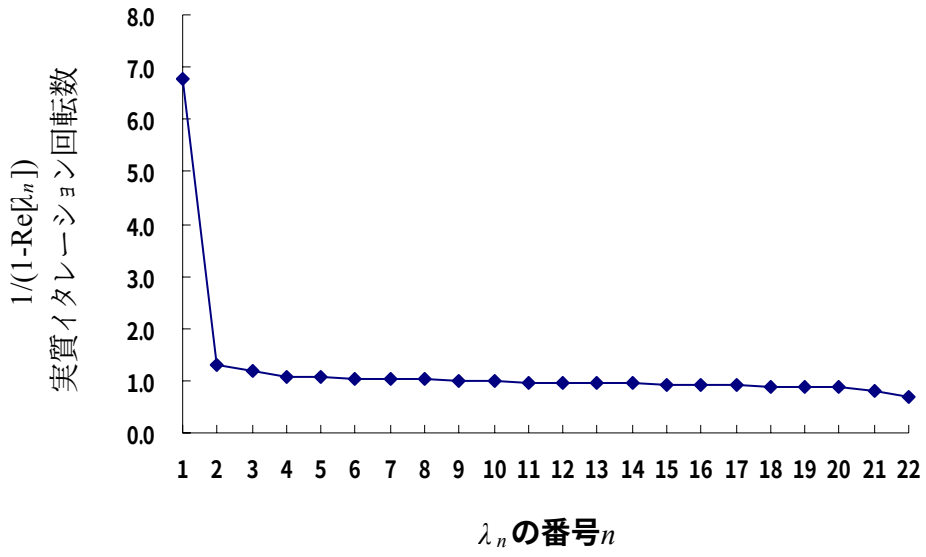


図 4.9　固有値に対応した 1/(1-Re[λ_n])

表 4.1　固有ベクトルと総所要時間量のベクトル $\boldsymbol{U}$

タスク名	第一固有ベクトル	第二固有ベクトル	総所要時間量のベクトル $\boldsymbol{U}$
要求仕様書の策定	**0.1932**	0.0728	**5.5675**
動作レベル・モデリングによるパフォーマンス評価	0.0138	0.0429	1.5603
仕様書概要の策定と確認	0.0243	0.0514	1.7670
テスト計画の策定	0.0123	0.0297	1.4177
検証計画の策定	0.0023	0.0244	1.2269
機能レベルのモデリング	**0.3252**	**0.2171**	**8.9827**
STG2 ブロックの設計	**0.2906**	0.0586	**7.9967**
VUP CPU ブロックの設計	**0.3414**	0.0574	**9.5544**
1TSM IP(レイアウト・レベル）ブロック	0.0039	0.0671	1.1403
CM Mem IP（レイアウト・レベル）ブロック	0.0039	0.0671	1.1403
MCTP ブロックの設計	**0.2499**	0.0407	**7.0037**
DFTTP の設計	0.1846	0.1056	**5.4757**
各ブロックのインテグレーション	**0.4085**	0.0126	**11.082**
テスト容易化設計(DFTTP の設計及び TSTSL の挿入)	**0.3092**	0.1661	**8.4550**
バウンダリー・スキャンの設計	0.1396	0.1785	4.3924
自動テスト・パターン生成	**0.1986**	0.0660	**5.8214**
シミュレーションの実行	**0.3717**	0.0837	**10.309**
ブロック・レベルの配置配線	**0.2369**	0.1838	**6.6407**
フロアー・プランニング	0.0661	**0.3171**	2.8054
自動配置配線	0.1762	**0.2053**	**5.3103**
チップ・レベルのレイアウト	0.0628	**0.4339**	2.7097
設計規定の確認(ERC/DRC/LVS)	0.0583	**0.6910**	2.7752

尚、ベクトル $\boldsymbol{u}_0$ は、総所要時間量のベクトルとの比較を容易化させるために、そのタスク成分を全て 1 にした。すなわち次式にて算出した。

$$\boldsymbol{u}_0 = \begin{bmatrix} u_{10} \\ u_{20} \\ \vdots \\ u_{22} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.25}$$

4.7.2　第一固有ベクトルの解析

表 4.1 の最初の列に示した第一固有ベクトルは、最大固有値 λ_1 に対応した固有ベクトルである。この第一固有ベクトルは、「各ブロックのインテグレーション」と「シミュレーションの実行」に大きな重み付けがあり、更に「要求仕様書の策定」、「機能レベルのモデリング」、「STG2 ブロックの設計」、「VUP CPU ブロックの設計」、「MCTP ブロックの設計」、「テスト容易化設計(DFTTP の設計及び TSTSL の挿入) 」、「自動テスト・パターン生成」、「ブロ

ック・レベルの配置配線」という負荷が後に続いている。これらのタスクはいずれも、本プロジェクトにおける論理設計と検証に関係のあるタスクである。実際本プロジェクトでは、検証のために機能シミュレーションと論理シミュレーションが利用され、設計と検証が繰り返し行われた。このことがイタレーションとしてプロジェクトに大きな負荷を与えたと考えられる。尚、これらのシミュレーションはハードウエア記述言語(Hardware Description Language) (Armstrong 1989)のイベント・ドリブン・シミュレーション・ソフトウエアで実行された。従って、第一固有ベクトルの重み付けから、大きな負荷がかかるタスクとして、論理設計の部分と検証のグループを特定でき、このグループを「論理とタイミング上の設計検証問題」として捉えることができた。言い換えれば、「論理とタイミング上の設計検証問題」を引き起こすタスクとその定量的度合いを明確に読みとることができたということである。

ここまでは第一固有ベクトルの中で重み付けの大きいタスクに注目してきたが、次に重み付けの小さいタスクに注目する。「1TSM IP ブロック」と「CM Mem IP ブロック」の二つの設計ブロック重み付けが小さく、この理由を考察してみる。設計ブロックには設計タスクの他に、チップをコンポーネント分割して得られたブロックも含まれる。ブロックとは半導体 IP であり、設計データの形態を属性としてハード IP とソフト IP に体系化できる。あるいは半導体 IP を創造するアプローチとして、機能的又はアルゴリズム的に設計する手法と、トランジスタの配置配線情報を構造的表現で設計する手法があり、前者で得られる IP はソフト IP、後者で得られる IP はハード IP と称される（平嶋, 1998)。ここで、1TSM IP と CM Mem IP は共にハード IP である。ハード IP の特徴は論理とタイミング上の設計検証が物理設計のレベルで行われることである。そして、これら二つの設計ブロックは外部のサード・パーティー(third party)の IP (Intellectual Property)開発メーカより導入されたものである。このためこれら二つの設計ブロックでは、先に述べた「論理とタイミング上の設計検証問題」が生じないのであり、「1TSM IP ブロック」と「CM Mem IP ブロック」の重み付けは小さいのである。従ってこのことも、第一固有ベクトルのタスク成分の重み付けから「論理とタイミング上の設計検証問題」が捉えられることの根拠となる。

本設計プロジェクトで利用されたブロックの中で、特に固有ベクトルの重みが大きいブロックは「VUP CPU ブロック」である。これはソフト IP の属性を持つプロセッサであり、ネットワークのスイッチング能力を上げるために複数個が組み込まれている。ソフト IP は、半導体メーカや特定の半導体技術に依存しない抽象的表現となっているため、異なる半導体メーカや半導体技術への応用生成ができるというポータビリティの高さが最大の長所である。しかし一方、ソフト IP は半導体設計者が更に物理設計領域での論理回路設計と検証を十分行う必要があり、多大な時間と経費がかかってしまうという短所もある。実際、この対象回路は仕様書通りのクロック(clock)周波数を満たさないタイミング(timing)収束問題を引き起こし、信号波形の乱れといったシグナル・インテグリティ(signal integrity:信号精度)問題の解決に時間を要した。そして、ハードウエア記述言語を利用した上流設計領域とト

ランジスタ等の物理設計領域間のイタレーションが多くなった工程であった。すなわちここからも「論理とタイミング上の設計検証問題」が捉えられる。

以上で示したことをまとめると、第一固有ベクトルのタスク成分の重み付けから「論理とタイミング上の設計検証問題」が捉えられることが認識された。また、半導体システム全体をブロック分割し、ブロックの属性をソフト IP とハード IP に分類したことで、プロジェクト全体の仕事量にそれらの特性がどのように影響を与えるのかが浮き彫りにされた。

4.7.3　第二固有ベクトルの解析

表 4.1 の二番目の列に示した第二固有ベクトルは、二番目に大きな固有値 λ_2 に対応した固有ベクトルである。「機能レベルのモデリング」、「フロアー・プランニング」、「自動配置配線」、「チップ・レベルのレイアウト」、「設計規定の確認」に大きな重み付けが観測された。これらのタスクの内容から、これらは物理的なインプリメンテーション（implementation）を行う最終具現化設計と検証のグループであることを特定でき、これらタスクのグループを「物理設計上のインプリメンテーション問題」として捉えることができた。最終的な物理設計において、全ての正確なトランジスタ情報をシミュレーションで検証するには極めて大きな負荷がかかる。また物理設計では配線の為に確保される物理的空間やトランジスタのスイッチング遅延時間には限界があり、未だヒュリスティックスな方法での解決策が施されている。これらがイタレーションを生じさせる原因である。

4.7.4　総所要時間量のベクトル *U* の解析

表 4.1 の右側の列は、本設計プロジェクトの総所要時間量のベクトル $\boldsymbol{U}$ を示している。この総所要時間量のベクトル $\boldsymbol{U}$ の中で大きい値のタスク成分は、第一固有ベクトルの重み付けに影響を大きく受けていることが判る。総所要時間量のベクトル $\boldsymbol{U}$ の中に太字で示した値の大きさの順番は全て、第一固有ベクトルの重みつけの大きさの順番と同じである。従って、第一固有ベクトルの解析で特定できた「論理とタイミング上の設計検証問題」が、本設計プロジェクトの総所要時間量のベクトル $\boldsymbol{U}$ に大きな負荷を与えていることが分かる。

4.7.5　半導体 IP 置き換えのシミュレーション

ここまで本設計プロジェクトを一対比較行列によって設計工程の構造的見地からモデリングを行い、設計工程の進展過程を固有値と固有ベクトルを切り口として解析を行ってきたところ、「VUP CPU ブロックの設計」を含むタスクのグループが「論理とタイミング上の設計検証問題」として特定され、これらがイタレーションを起こして本設計プロジェクトに大きな負荷を与えていることが判った。そこで発展的アプローチとして、「VUP CPU ブロックの設計」と依存関係にあるタスクを恣意的に調整することで、プロジェクトに与える負荷がどのように変化するか観測する。VUP CPU ブロックは前述の通り、ソフト IP の属性を

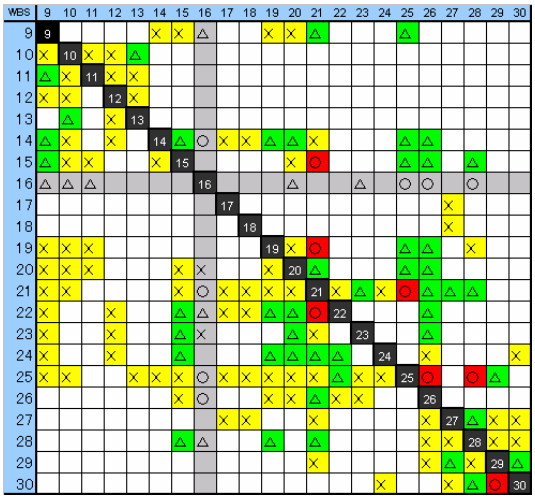
持つプロセッサである。ソフト IP には設計者が物理設計領域での論理回路設計と検証を十分行う必要があり、多大な時間と経費がかかるという特徴があった。そこで、このブロックをハード IP の属性を持つプロセッサで置き換えたモデルを作成し、その場合の $1/(1-\mathrm{Re}[\lambda_n])$と総所要時間量のベクトル $\boldsymbol{U}$ を算出する。ハード IP へのブロックの置き換えにあたっては二つのモデルを想定する。それらをシミュレーション事例-1、シミュレーション事例-2 とし、前項までの解析を行った実設計事例とあわせて図 4.10 に示す。

図 4.10(b)に示すシミュレーション事例-1 では、「VUP CPU ブロックの設計」すなわち WBS16 とそれと関係する全てのタスク間の依存性の強度を、実設計事例図 4.10(a)で設定した強度よりレベルを一つ下げている。すなわち実設計事例において依存性の強度が強(○)であったところを中間(△)に、中間(△)であったところを弱(×)に、そして弱(×)であったところを依存性がないとしている。この置き換えは、VUP CPU ブロックをハード IP に交換することと、ここで準備されたハード IP のバス周辺やインタフェース等の回路修正が必要な状況を想定している。

図 4.10(c)に示すシミュレーション事例-2 では、WBS16 とそれと関係する全てのタスク間の依存性の強度を、実設計事例で設定した強度よりレベルを二つ下げている。すなわち実設計事例において依存性の強度が強(○)であったところを弱(×)に、中間(△)と弱(×)であったところを依存性がないとしている。この置き換えは、VUP CPU ブロックをハード IP に交換することと、ハード IP のバス周辺やインタフェース等の回路修正がほとんど必要ない状況を想定している。回路修正がほとんど必要ない分だけ、これと関係するタスクとの依存性は小さいのである。

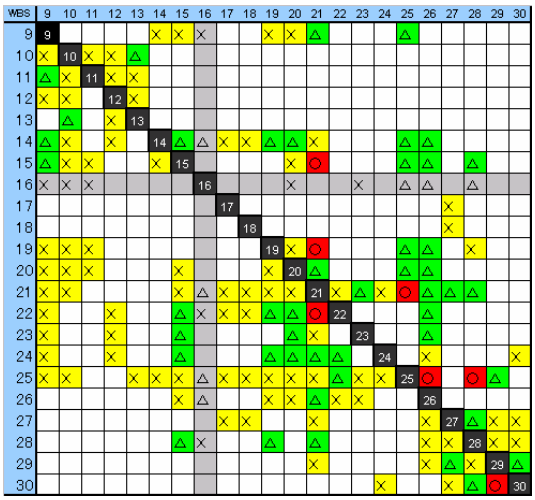
シミュレーション結果を表 4.2 に示す。$1/(1-\mathrm{Re}[\lambda_1])$は、実設計事例、シミュレーション事例-1、シミュレーション事例-2 の順で減少している。また「VUP CPU ブロックの設計」の時間量も $1/(1-\mathrm{Re}[\lambda_1])$と第一固有ベクトルの相乗効果で減少し、総所要時間量の合計も減少している。これらの結果から、第一固有ベクトルから特定される問題が設計工程全体に及ぼす負荷が低減したことが判る。すなわち、ハード IP への置き換えが「論理とタイミング上の設計検証問題」の解決策となることが検証された。一方、$1/(1-\mathrm{Re}[\lambda_2])$と $1/(1-\mathrm{Re}[\lambda_3])$では、二つのシミュレーション事例の値のほうが実設計事例の値よりも大きい。これは $1/(1-\mathrm{Re}[\lambda_1])$の減少により相対的に増加したと考えられる。第二固有ベクトルから特定される問題すなわち「物理設計上のインプリメンテーション問題」の負荷が相対的に増加したことが検証された。

シミュレーション結果から考察されることは、イタレーションを生じさせるタスクの代替策を想定し、$1/(1-\mathrm{Re}[\lambda_n])$、固有ベクトル、総所要時間量のベクトル $\boldsymbol{U}$ の大きさやその変化を観測することは、プロジェクトの解析に有用であるということである。つまりプロジェクト・マネージャは、VUP CPU ブロックをハード IP に交換することや、周辺インタフェース設計が必要かどうかということを定量的に判断できるということである。



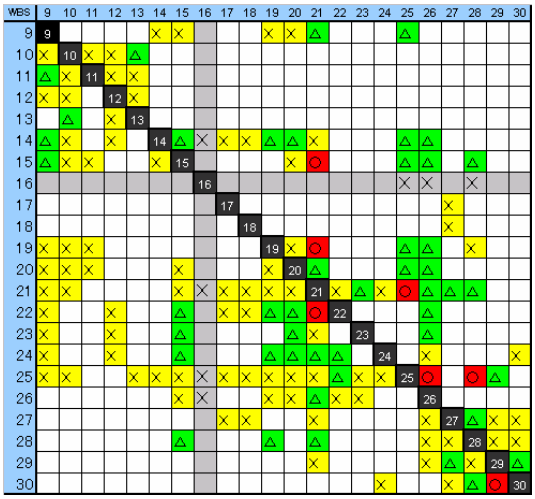
- 実設計プロジェクト
- WBS16 ではソフト IP を使用
- 依存性強度: 3 レベル

(a) 実設計事例



- WBS16 をハード IP に交換
- ハード IP 周辺インタフェース設計要
- 依存性強度: 2 レベル

(b) シミュレーション事例-1



- WBS16 をハード IP に交換
- ハード IP 周辺インタフェース設計済
- 依存性強度: 1 レベル

(c) シミュレーション事例-2

図 4.10　半導体 IP の置き換えによるシミュレーション

表 4.2　シミュレーション結果

		分析結果事例	ｼﾐｭﾚｰｼｮﾝ事例-1	ｼﾐｭﾚｰｼｮﾝ事例-2
1/(1-Re[λ_n])	λ_1の場合	6.777	3.159	2.924
	λ_2の場合	1.309	1.365	1.358
	λ_3の場合	1.171	1.304	1.301
タスク名	要求仕様書の策定	5.568	2.820	2.547
	仕様書概要の策定と確認	1.767	1.476	1.447
	テスト計画の策定	1.418	1.271	1.257
	検証計画の策定	1.227	1.203	1.201
	機能レベルのモデリング	8.983	3.845	3.465
	STG2 ブロックの設計	7.997	4.244	3.948
	VUP CPU ブロックの設計	**9.554**	**2.564**	**1.457**
	1TSM IP(レイアウト・レベル）ブロック	1.140	1.099	1.095
	CM Mem IP（レイアウト・レベル）ブロック	1.140	1.098	1.095
	MCTP ブロックの設計	7.004	3.768	3.518
	DFTTP の設計	5.476	2.902	2.739
	各ブロックのインテグレーション	11.082	4.892	4.424
	バウンダリー・スキャンの設計	8.455	4.264	3.881
	テスト容易化設計(DFTTP の設計及び TSTSL の挿入)	4.392	2.426	2.314
	自動テスト・パターン生成	5.821	3.450	3.262
	シミュレーションの実行	10.309	4.651	4.210
	ブロック・レベルの配置配線	6.641	2.626	2.335
	フロアー・プランニング	2.805	1.974	1.905
	自動配置配線	5.310	2.852	2.598
	チップ・レベルのレイアウト	2.710	1.929	1.866
	設計規定の確認(ERC/DRC/LVS)	2.775	2.135	2.078
総所要時間量の合計		**113.135**	**58.887**	**54.025**

4.8　考察

本実証研究において、一対比較行列による設計工程のモデリングとその解析を行った。この実証研究から得られたことは、第一に、タスク間の依存性の強度を成分とする一対比較行列により、イタレーションを定量的にモデリングできることである。第二に、一対比較行列を固有値分解して得られる、固有値とそれに対応した固有ベクトルを解析することで、イタレーションを生じさせるタスクのグループの特定と、それらがプロジェクトに与えている負荷を定量的に認識できるということである。そして最後に、問題の解決策をシミュレーションにより検討できるということである。よって、一対比較行列による設計工程のモデリングは、複雑なタスク間のイタレーションを表現でき、またそれによりプロジェクトの解析ができるため、プロジェクト・マネジメントにとって有用であることが検証された。

本実証研究において、モデリングに適用したのは大規模で複雑な半導体設計の実プロジェクトであった。第一固有ベクトルの解析から「論理とタイミング上の設計検証問題」、第二固有ベクトルの解析から「物理設計上のインプリメンテーションの問題」が特定された。これらの問題の解決策を考察してみる。双方の問題とも、設計と検証間、そして論理設計と物理設計間のタスク同士のイタレーションによるものである。イタレーションによる問題を解決するには、イタレーションの回転速度を上げることとイタレーションの回転の数を少なくすることの検討が重要である。

イタレーションの回転速度を上げるためには、例えば設計解析に向けたシミュレーションを質量ともに大幅に増やすことや、上流の設計過程で設計品質を予め高めた手法を導入することが必要である。

イタレーションの回転の数を少なくする方策の一つは、依存関係にあるタスクそれぞれの工程を見直して、効果的な調整を行うことである。例えば、依存関係にあるタスク双方に共通の責任者を配属させることや、タスクに関わるチームを物理的に同じ設計環境に配置させることである。これにより効率的に技術・知識情報の共有や交換ができるからである。またタスク間のインタフェースや、設計チームと設計技術者個人間のインタフェースとして、予め適切な仕様書や取り決めを交わすことで効率的な運営が可能となる。イタレーションの回転の数を少なくするもう一つの方策は代替策の導入である。実証研究では、ソフト IP をハード IP に置き換えることで、プロジェクト全体の総所要時間量を低減すること示した。ブロック分割と半導体 IP の利用は有効なアプローチである。ただし、半導体の物理的制約限界まで引き出さなければならない場合、その対象となるソフト IP の利用が設計工程に極めて重大な影響を及ぼす可能性があることを認識すべきである。ソフト IP とハード IP 双方の特性の理解と適切な運用が必要である。

最後に一対比較行列の適用に当たって議論されるべき課題がある。それは各タスクの予定実行時間を成分としたベクトル $\boldsymbol{u}_0$ を本実証研究ではデターミニスティックとした点である。実際に設計工程を進行させる場合、各タスクの予定実行時間の全ては予見できず、また一元的でもない。従ってストカスティックなモデリングの可能性を課題として議論する必要がある。

4.9　本章のまとめ

システムの大規模・複雑化とともに、エンジニアリング設計工程はタスク間の反復プロセスといったイタレーションが頻繁に観測されるようになった。イタレーションの解明のためには、イタレーションを含む設計プロセスの数理モデル化が必要である。本章ではイタレーションのモデリングに関する理論について議論し、イタレーションのモデリングを実践した。

本章では、まず設計プロセスにおけるイタレーションの問題について議論し、イタレー

ション減少には設計工程のモデリングと問題の特定が必要であることを示した。次に従来のモデリング手法の問題点と、それから導かれる DSM (Design Structure Matrix:設計構造マトリックス)の有用性を示した。そして DSM を応用した一対比較行列(Work Transformation Matrix)によるイタレーションのモデリングから、設計プロセスの解析が可能であることの理論を示した。これはイタレーションの収束性を固有値問題として扱うものであった。すなわち一対比較行列の固有ベクトルを算出し、その成分の重み付けから、イタレーションを生じさせるタスクとそれが設計プロセスに与える負荷を解析することであった。そして最後に行った実証研究では、イタレーションが多数存在する、ネットワーク・プロセッサ半導体の実設計開発工程を一対比較行列によりモデリングし、その有効性を検証した。その結果、設計負荷を起因させている二つの技術的問題点を特定でき、また設計負荷を改善させる解決策がシミュレーションによって確認された。すなわち一対比較行列によりモデリングは、設計プロセスにおけるイタレーションを定量的に解析できるため、プロジェクト・マネジメントにとって有用であることが検証された。

本モデルの既知応用分野は米国の宇宙航空産業並びに自動車産業等であった。本研究ではそのモデルを半導体設計分野に始めて応用したものであり、その有用性が確認されたと言える。

尚、本章における課題は、一対比較行列によるモデリングにおいて、各タスクの予定実行時間をデターミニスティック(deterministic)とした点である。実際の設計工程を進行させる場合、その全ては予見できず、また一元的でもない。従ってデターミニスティックとは対極的なアプローチとしてストカスティック(stochastic)なモデル化の議論が必要である。そのストカスティックのアプローチについては次章で議論展開する。

参考文献

[1] Ahmadi, Reza H. and Wang, Hongbo: “Rationalizing product design development process”, USLA Anderson Graduate School of Management Report (1994)

[2] Armstrong, James R.: “Chip-level modeling with VHDL”, Kuluwer Academic (1989) (ジェームス・R.・アームストロング（白石繁,山本靖訳）: “VHDL デザイン・テクニック:標準ハードウエア記述言語とチップ・レベル・モデリング,電波新聞社 (1990))

[3] Browning, Tyson R.: “Use of Dependency Structure Matrices for Product Development Cycle Time Reduction” (1998a)

[4] Browning, Tyson R.: “Sources of Schedule Risk in Complex System Development”, Proceedings of the eighth annual international symposium of INCOSE (1998b)

[5] Carrascosa, Maria, Eppinger, Steven D. and Whitney, Daniel E.: “Using the Design Structure Matrix to Estimate Product Development Time”, Proceedings of the ASME Design Engineering Technical Conferences (1998)

[6] Chatelin, Francoise,: “Valeurs propres de matrices”, Masson (1988) (F.・シャトラン（伊里正夫他訳）: “行列の固有値”,シュプリンガー・フェアラーク東京 (1993))

[7] Davidson, E. R.: “Matrix eigenvector methods, Methods in Computational Molecular Physics”, Reidel Boston (1983)

[8] Denker, Stephen and Steward, Donald: “Using the Design Structure Method” (1995)

[9] Eppinger Steven D.: “Model-based Approaches to Managing Concurrent Engineering”, Journal of Engineering Design, Vol. 2, pp. 283-290 (1991)

[10] Eppinger Steven D.: “Innovation at the Speed of Innovation”, Harvard Business Review, Vol. 79, pp. 149-158 (2001)

[11] 藤井文夫,滝諭,萩原伸幸,本間俊雄,三井和男: “非線形構造モデルの動的応答と安定性”, コロナ社 (2003)

[12] Fujimoto, Takahiro and Clark, Kim B.: “Product Development Performance: Strategy, Organization, and Management in the World Auto Industry”, Harvard Business School Press (1991) (藤本隆宏,キム・B・クラーク（田村明比古訳）: “実証研究　製品開発力―日米欧自動車メーカー20 社の詳細調査”,ダイヤモンド社 (1993))

[13] Gantt, H. L.: “Work, Wages, and Profits”, The Engineering Magazine New York (1910)

[14] 萩原一郎,吉村卓也,梶原逸朗,雉本信哉（長松昭男編）: “音・振動のモード解析と制御”, コロナ社 (1996)

[15] 平嶋竜太: “システム LSI の保護法制:設計情報の流通と法的保護”,信山社出版 (1998)

[16] Jalote, Pankaj: “Software Project Management in Practice”, Pearson Education (2002)

[17] 加藤勉: “固有値問題”,彰国社 (1967)

[18] Kaufman, Phillip A.: “Wanted: Tools for Validation, Iteration”, IEEE Computer Design (1989)

[19] Kusiak, A., Larson, N. and Wang, J.: “Reengineering of Design and Manufacturing Processes”, Computers and Industrial Engineering, Vol. 26, No. 3, pp. 521-536 (1994)

[20] 小出昭一郎: “量子論”,裳華房 (1968)

[21] 小林啓祐: “原子炉物理”,コロナ社 (1996)

[22] Loch, Christoph H. and Teerwiesch, Christian: “Communication and Uncertainly in Concurrent Engineering”, Management Science, 44 (8) (1998)

[23] Marca, David. A. and McGowan, Clement. L.: “Sadt: Structured Analysis and Design Techniques”, McGraw-Hill (1988) (D.・マーカー（漆島正人,谷岡雄一,研野和人,松本巌訳）（研野和人監訳）: “IDEF0/SADT ビジネスプロセスとエンタープライズモデリング”,社会経済生産性本部 (1998))

[24] Mead, Carver and Conway, Lynn: “Introduction to VLSI Systems”, Addison-Wesley Longman Publishing (1979) (C.・ミード,L.・コンウエイ（菅野卓雄,榊裕之監訳）: “超 LSI システム入門”,培風館 (1981))

[25] Moore, Laurence J. and Clayton, Edward R.: “GERT Modeling and Simulation: Fundamentals

and Applications”, Petrocelli/ Charter (1976)

[26] 森竜雄: “PERT／新しい仕事のまとめ方”,日本能率協会 (1964)

[27] 森竜雄: “続 PERT／効率的な応用のしかた”,日本能率協会 (1965)

[28] 西村千秋: “応用数学”コロナ社 (1990)

[29] Neumann, Kkaus and Steinhardt, Ulrich: “GERT networks and the time-oriented evaluation of projects Berlin”, Springer-Verlag (1979)

[30] Neumann, Kkaus: “Stochastic project networks:temporal analysis, scheduling and cost minimization”, Springer-Verlag (1990)

[31] Ogata, Katsuhiko: “Modern Control Engineering”, Prentice Hall Inc. (1990) (尾形克彦: “現代制御工学”,新技術開発センター (1994))

[32] Pritsker, A. Alan B.: “Modeling and analysis using Q-GERT networks”, John Wiley & Sons Wiley (1977)

[33] Rodgers, James L. and McCulley, Collin M.: “Integrating a genetic algorithm into a knowledged-based system for ordering complex design progress”, NASA TM-110247 (1996)

[34] Ramkumar Ramaswamy: “Variability in software development process families”, Software Quality Professional (2001)

[35] Schumpeter, J. A.: “The Theory of Economic Development” (1912) (J.・A.・シュムペーター (塩野谷祐一,東畑精一,中山伊知郎訳): “経済発展の理論”,岩波文庫(1977))

[36] Schumpeter, J. A.: “Capitalism, Socialism, and Democracy” (1942) (J.・A.・シュムペーター（中山伊知郎,東畑精一訳）: “資本主義・社会主義・民主主義”,東洋経済新報社 (2001))

[37] Simon, Herbert A.: “The Sciences of the Artificial”, MIT Press (1969) (ハーバート・A.・サイモン（稲葉元吉,吉原英樹訳）: “システムの科学”,パーソナルメディア (1999))

[38] Smith, Doug: “A Case Study and Analysis of the Tacoma Narrows Bridge Failure” (1974)

[39] Smith, Robert P.: “Development and Verification of Engineering Design Iteration Models” (1992)

[40] Smith, Robert P. and Eppinger, Steven D.: “Characteristics and Models of Iteration in Engineering Design” (1993)

[41] Smith, Robert P. and Eppinger, Steven D.: “Identifying Controlling Features of Engineering Design Iteration”, Management Scienece, 43 (3) (1997a)

[42] Smith, Robert P. and Eppinger, Steven D.: “A Predictive Model of Sequential Iteration in Engineering Design”, Management Scienece, 43 (8) (1997b)

[43] Smith, Robert P. and Eppinger, Steven D.: “Deciding Between Sequential and Parallel Task in Engineering Design”, Concurrent Engineering: Research and Applications, 6 (1) (1997c)

[44] Steward, Donald V.: “The Design Structure System: A Method for Managing the Design of Complex Systems”, IEEE Transactions on Engineering Management, 28 (3) (1981a)

[45] Steward, Donald V.: “Systems Analysis and Management: Structure, Strategy and Design”,

New York PB1 (1981b)

[46] 菅野孝男: “ソフトウエア開発のマネージメント”,新紀元社 (2003)

[47] 竹内啓: “線形数学”,培風舘 (1974)

[48] Thomke, Stefan H.: “Managing Experimentation in the Design of New Products”, Management Science, 44 (6) (1998)

[49] von Hippel, Eric: “Task Partition: An Innovation Process Variable”, Research Policy, Vol. 19, pp. 407-418 (1990)

[50] von Hippel, Eric: “Sticky Information and the Locus of Problems Solving: Implication for Innovation”, Management Science 40, No. 4, pp. 429-439 (1994)

[51] Warfield, John N.: “Binary Matrices in System Modeling”, IEEE Transactions on Systems, Man and Cybernetics, Vol.3, pp. 441-449 (1973)

[52] Wileden, Jack C.: “Inceremental Development and Iteration in the Software Process”, Iteration in the Software Procdess: Third International Software Process Workshop, IEEE (1986)

[53] Yassine, Ali and Braha, Dan: “Four Complex Problems in Concurrent Engineering and the Design Structure Matrix Method”, Concurrent Engineering Research & Applications, Vol. 11, No. 3 (2003)

[54] Yu, Tian-Li, Yassin, Ali, and Goldberg David E.: “A Genetic Algorithm For Developing Modular Product Architecture”, DETC2003/ DTM-48657 (2003)

第5章　設計プロセスにおけるイタレーションとリスク・アセスメント

5.1　リスク・アセスメント

製品開発において、設計プロジェクトが予定期限までに終了しない、あるいは当初の予算枠を超えてしまうといったことは、企業の信用低下や経済的損失に関わる重大な問題である。この問題を生じさせる要因となるものは、予想困難であり、かつプロジェクトに係わる当事者の内外に存在する(横山, 1976)。つまり、設計プロジェクトはリスクを引き起こす不確実性要因を直接的、間接的に抱えているのである。そして活動時間と活動コストのリスクは、不確実性要因をコントロールできない場合に顕在化すると考えられている(Sage, 1997)。活動時間と活動コストのリスクに対処するために、リスクの先読みと見積り、評価を行っていくリスク・アセスメント(Wang & Roush, 2003)が有効である。

Restrepo (1995)によれば、プロジェクトの意思決定は定量的なリスク・アセスメントではなく、定性的なリスク・アセスメントに基づいて行われている場合が多い(Restrepo, 1995)。しかし、リスクの定量的評価を根拠としない定性的リスク・アセスメントでは、プロジェクトに関わる当事者間でのコンセンサスを得ることが困難であり、また記述論理の限界点がある。確実で精度が高いリスク・アセスメントのためには、リスクを特定してそれを定量的に評価する手法が重要と言える。

製品開発におけるリスク・アセスメントについては現在まで幾多の研究が重ねられている。その多くがプロジェクトのリスクを意思決定に利用したものである(Chapman et al., 1985; Chapman, 1997)。特定用途向けに開発された独自の手法も存在する(Reid, 1989)。定量的手法としては、プロジェクト管理で利用されるクリティカル・パス解析(Critical Path Analysis:以下 CPA と称す)ツールにより活動時間のリスク評価を行った事例研究が多い(Pierce, 1998)。また、エンジニアリング・プロジェクト内のイベント(event)やタスク(task)に活動時間と活動コストそれぞれの確率密度関数を割り当て、モンテカルロ・シミュレーションによりプロジェクトが完了するまでの確率分布を算出する手法が一般的である

(Webb, 1997)。しかしリスクとはその生起確率とリスクが招く経済的損失量で表現されるものであり(Wang & Roush, 2003; Kaplan & Garrick, 1981; 小林, 2000; 富田&平野, 2001; Vose, 2003)、リスク・アセスメントにはパス解析や確率だけの定量的情報では不十分と言える。

昨今自動車産業ではプロジェクトのリスク・アセスメントを行うツールの開発事例研究が報告され、宇宙航空産業でも生起確率や損失量を定量的に評価する試みがなされている(Patterson et al., 2002; Browning & Eppinger, 2002)。しかし半導体の設計開発プロジェクトでは、プロジェクト管理手法のガント・チャート(Gantt Chart) (Gantt, 1910)や PERT (Program Evaluation and Review Technique)、CPA を利用したパス解析の他、モンテカルロ・シミュレーションで確率分布を出力するにとどまっている。半導体設計開発においても、確実で精度の高いリスク・アセスメントの実践に向けた、リスクの定量的評価手法の検討が重要と言える。

半導体設計開発のプロジェクトにおけるリスクの定量的評価の試みが進んでいない理由として、設計工程にイタレーション(iteration)が複雑に数多く存在し(吉田, 2003)、それをモデル化することが困難であることが考えられる。イタレーションとはタスク間における知識と技術情報のフィードバック(feedback)と反復プロセスである。イタレーションは設計工程の活動時間や活動コストに負荷を与えるものであり、リスクを顕在化させる不確実性要因の一つである(Browning & Eppinger, 2002)。

近年、設計構造マトリックス(Design Structure Matrix:以下 DSM と称す)がイタレーションをモデル化できる手法として注目されている。そこで本章では、DSM を利用して半導体設計開発の実プロジェクトからイタレーションのモデルを作成し、それに対してリスクの定量的評価を行い、リスク・アセスメントとしてのその有用性を検証する。

本章では、まず半導体設計工程のイタレーションについて述べ、その後 DSM によるイタレーションのモデル化の手順について説明する。次にイタレーションのモデルを利用して得られる、プロジェクト完了時すなわちイタレーション収束後の活動時間と活動コストの算出式と、更にリスク・モデルから導かれるリスク値の算出式を示す。そしてこれらの式に基づいて構築させる定量的リスク評価のフレームワークを提示し、このフレームワークを半導体設計開発の実プロジェクトに適用して、活動時間と活動コストのリスク評価のシミュレーションを行う。シミュレーションにおいては、リスクの比較のために仮想的にタスクの一部を変更した 8 つの設計工程モデルを作成し、リスクの生起確率の計算にはモンテカルロ・シミュレーションを行う。最後に、結果として得られた活動時間と活動コストのリスクに対して考察を行う。

5.2　イタレーションのモデリング

5.2.1 イタレーションのモデリングの従来手法と DSM

コンピュータや半導体そしてソフトウエアに代表されるハイテク製品の設計開発にはそ

の設計・開発工程にはフィードバック(feedback)とその反復プロセスといったイタレーションが存在する(Alexander, 1964; Whitney, 1990; Smith & Morrow, 1999)。特にプロダクト・イノベーション(Schumpeter, 2001)が要求される製品設計・開発工程においては、それに携わる設計者・チームやタスク(task)間に知識と技術情報のイタレーションが行われるのが観測されてきた(von Hippel, 1994)。イタレーションは一方向だけではなく、逆方向に後戻りしたり、同時進行にやり取りが行われたりする(Cooper, 1993; Kline, 1985; Nightingale, 2000)。

イタレーションは設計工程の活動時間や活動コストに負荷を与えそれらの大きさを変えてしまう不確実性要因の一つである(山本, 2004)。相互関係が複雑で数多くの知識と技術情報のやり取りが想定される場合は、そのやり取りが計画通りの時間と場所で行われない限り、設計工程の効率化につながらない。知識と技術情報のやり取りの道筋を解析することが設計工程の効率化の切り口につながる(Steward, 1981; Eppinger et al., 1994)。すなわち、リスクは完全には予見できないが、イタレーションを解析し、コントロールできるならばリスクの低減は可能であると言える。そこでイタレーションの解析に向けた、イタレーションのモデル化が必要となる。

イタレーションが存在する設計工程を伝統的なプロジェクト管理手法でモデル化することは困難である。これまでは、設計工程の改善のためには設計工程の特性を十分理解していることが必要であるとの考えのもと(Whitney, 1990)、設計工程の問題を原則的にその構造の問題としてとらえ、ガント・チャート(Gantt Chart)、PERT (Program Evaluation and Review Technique)、 CPA (Critical Path Analysis: クリティカル・パス解析)といった伝統的な手法で問題を解決してきた((Alexander, 1964; Whitney, 1990; Clark & Baldwin, 2004)。しかし、イタレーションが複雑で数多く存在する大規模な設計工程をこれら従来手法で構造の問題として捉え、解析していくことは容易ではない。

そこでイタレーションを実現できるモデルとして設計構造マトリックス(Design Structure Matrix:以下 DSM と称す)が注目されてきた(Steward, 1981; Eppinger et al., 1994; Eppinger, 2001)。DSM は生産資源のモノの側面を捨象し、知識や設計情報を含む技術情報という観点から設計工程を捉えることができる。特に設計工程が大規模で複雑化した時の構造的問題の解決に向けた洞察力を持っている(Eppinger, 2001)。国内でも半導体設計開発の事例を適用し、プロジェクト管理手法の観点から DSM の有用性が検証されている(山本, 2004)。

5.2.2 DSM によるイタレーションのモデリング

設計工程はプロジェクト管理で定義されるタスクという設計作業の要素の集まりであり、その要素同士は相互に従属性を持つか独立しているかである(von Hippel, 1990)。まずマトリックス構造によって全てのタスク間における依存性の強度（一対比較値）a_{ij} を特定し、それらを成分とする一対比較行列 $\boldsymbol{A}$ を作成する(Smith & Eppinger, 1997)。依存性の強度 a_{ij} とは、タスク i の終了後にタスク j が開始される時、タスク i のタスク量に行列成分 a_{ij} を乗じたタスク量がタスク j のタスク量に加算されることを意味する。設計工程の中にタスクが n

個存在する場合、一対比較行列 $\boldsymbol{A}$ は次のような $n \times n$ の正方行列となる。

$$\boldsymbol{A} = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \tag{5.1}$$

原則的にタスク i のタスク量以上にタスク j のタスク量に加算されないことを前提にすれば $1 \leq i \leq n$、 $1 \leq j \leq n$、 $0 \leq a_{ij} < 1$ であり、同じタスク同士でのタスクの伝搬は生じないので対角成分は全て 0 である。

次に一対比較行列 $\boldsymbol{A}$ を用いて設計工程の総ベクトル量 $\boldsymbol{U}$ を定義する。まず n 次のベクトル $\boldsymbol{u}_t$ は t 回目のイタレーションにおける、各タスクの仕事量を成分に持つとする。$\boldsymbol{u}_0$ はイタレーションによる負荷がない、当初計画する各タスクの予定仕事量を成分とする。本章では全ての成分を 1 として議論を進める。すると、M 回イタレーション後の総ベクトル量 $\boldsymbol{U}$ は次式で表される。ここで $\boldsymbol{A}^t$ は $\boldsymbol{A}$ のべき乗である。

$$\boldsymbol{U} = \sum_{t=0}^{M} \boldsymbol{u}_t = \sum_{t=0}^{M} \boldsymbol{A}^t \boldsymbol{u}_0 = (\sum_{t=0}^{M} \boldsymbol{A}^t)\boldsymbol{u}_0 \tag{5.2}$$

設計工程は安定性を満たし、イタレーションは収束するとして $M \to \infty$ と考えた場合、総ベクトル量 $\boldsymbol{U}$ は以下の式に変形できる。

$$\boldsymbol{U} = (\sum_{t=0}^{\infty} \boldsymbol{A}^t)\boldsymbol{u}_0 \tag{5.3}$$

総ベクトル量 $\boldsymbol{U}$ の成分値は、その成分に対応したタスクがイタレーション収束後にどの程度回転したのかを示す。すなわち、総ベクトル量 $\boldsymbol{U}$ は各タスクに対応するイタレーションの実質回転数を成分に持つもので、これがイタレーションのモデルとなる。

5.2.3　活動時間及び活動コストの算出式

イタレーションのモデルを用いてイタレーション収束後の活動時間及び活動コストを求める。総ベクトル量 $\boldsymbol{U}$ はプロジェクトの活動時間と活動コストそれぞれ線形に従属性を与えると仮定して議論を進める。総ベクトル量 $\boldsymbol{U}$ の成分は対応するタスクのイタレーションの実質回転数であるので、イタレーションを想定していない活動時間 $\boldsymbol{V}$ と活動コスト $\boldsymbol{W}$ が判っているのであれば、イタレーション収束後の活動時間 $\boldsymbol{S}$ と活動コスト $\boldsymbol{C}$ は次式により算出できる。ここで $\boldsymbol{V}$ と $\boldsymbol{W}$ はそれぞれ各タスクの活動時間と活動コストを対角成分に持つ対

角行列である。

$$\boldsymbol{S} = \boldsymbol{VU} = \boldsymbol{V}(\sum_{t=0}^{\infty} \boldsymbol{A}^t)\boldsymbol{u}_0 \tag{5.4}$$

$$\boldsymbol{C} = \boldsymbol{WU} = \boldsymbol{W}(\sum_{t=0}^{\infty} \boldsymbol{A}^t)\boldsymbol{u}_0 \tag{5.5}$$

5.3　リスク・モデルとリスク値の算出式

5.3.1 リスク・モデル

まずリスクの算出式を導き出すためのリスク・モデルについて述べる。製品設計・製品開発のプロジェクトを行う企業の経営活動においては、プロジェクトにかかる活動時間や活動コストに対して実践的に目標値を掲げ、それらを目標値内に収める様に管理する。ここでプロジェクトとしての活動時間並びに活動コストの結果をそれぞれの結果値と呼ぶ。そのプロジェクトの結果値が結果として目標値を越えてしまう場合、それが発展して経済的な損失につながることも考えられる。この重大な結果をリスク事象と呼ぶ(富田&平野, 2001)。リスクの顕在性を決定論的に予見することはできない。従って確率論的なアプローチを用いて、プロジェクトの許容値や目標値を設定することがプロジェクト管理において不可欠である(小林, 2000; Vose, 2003)。

一般的にプロジェクトのリスクは、結果値が目標値を超えてしまう生起確率と結果値が招く損失量で定量的に表現できる(Wang & Roush, 2003; Webb, 1997; 小林, 2000; 富田&平野, 2001; Vose, 2003)。すなわちリスク R は結果値が顕在化する生起確率 P と、顕在化した時の損失量 L による期待値であり、次式で表される。そしてこれをリスク・モデルとする。

$$R\text{ (結果値)} = P\text{ (結果値)} \cdot L\text{ (結果値)} \tag{5.6}$$

リスク・モデルをもとにリスクの算出式を導く。プロジェクトの目標値を T とし、結果値を確率変数 x としたとき、結果値の生起確率 $P(x)$は確率密度関数 $f(x)$によって次式で表される(ただし $x>T$)。

$$P(x) = \int_T^{\infty} f(x)dx \tag{5.7}$$

そして損失量を $L(x)$とするなら、リスク R は x が離散確率変数の場合、

$$R = \sum_{i=1}^{\infty} f(x_i) L(x_i) \tag{5.8}$$

となり、x が連続確率変数の場合、

$$R = \int_{T}^{+\infty} f(x) L(x) dx \tag{5.9}$$

と表される。本章では x を連続確率変数として議論を進める。

$L(x)$はプロジェクトが目標通りに完了しない場合、すなわち目標値以上に経営活動が費やされる場合に生じる経済的損失を表す損失関数と考えることができる。損失関数は結果値 x の目標値 T からのずれの二乗に比例して増加する二次式、

$$L(x) = K(x-T)^2 \tag{5.10}$$

で表すことができる（ただし K は比例定数、$x>T$）。

ここで、目標値からずれる事によって発生する損失が式(5.10)で表されることの数学的根拠を示す。本来あるべきプロジェクトの理想的な姿は、予め設定された目標値以内に全て収めることである。しかしほとんどのプロジェクトにおいて実際の結果は目標値を挟みばらつきが生じる。ここで損失関数 $L(x)$を、この T のまわりでテーラー展開する（ただし $x>T$）。

$$L(x) = L(T + x - T) = L(T) + \frac{L'(T)}{1!}(x-T) + \frac{L''(T)}{2!}(x-T)^2 + \cdots \tag{5.11}$$

仮定から $L(T) = 0$、また $L(x)$は $x = T$ で損失が最小になるから、$L'(T)$も 0 である。従ってテーラー展開の第 3 項が主要項となり、高次の項を省略すれば次の近似式が成立する。

$$L(x) \cong K(x-T)^2 \tag{5.12}$$

この損失関数は田口による品質のばらつきと経済的損失による関数として知られている(田口, 1969)。

式(5.10)を式(5.9)に代入することで、リスク R は次式で表されることとなる。

$$R = K \int_{T}^{+\infty} f(x)(x-T)^2 dx \tag{5.13}$$

5.3.2 リスク値の算出式

プロジェクトの主たるリスクとして活動時間と活動コストを取り上げて議論を進める。

活動時間の目標値を T_S、結果値を確率変数 x、確率密度関数を $f_S(x)$とするなら、活動時間の結果値の生起確率 $P_S(x)$は式(5.7)から、

$$P_S(x) = \int_{T_S}^{\infty} f_S(x)dx, \qquad x > T_S \tag{5.14}$$

で表される。また活動時間の損失関数の比例定数を K_Sとするなら、活動時間のリスク値 R_Sは式(5.13)から、次式で表される。

$$R_S = K_S \int_{T_S}^{+\infty} f_S(x)(x - T_S)^2 dx \tag{5.15}$$

同様に、活動コストの目標値を T_C、結果値を確率変数 y、確率密度関数を $f_C(y)$とするなら、活動コストの結果値の生起確率 $P_C(y)$は式(5.7)から、

$$P_C(y) = \int_{T_C}^{\infty} f_C(y)dy, \qquad y > T_C \tag{5.16}$$

で表される。また活動コストの損失関数の比例定数を K_Cとするなら、活動コストのリスク値 R_Cは式(5.13)次式で表される。

$$R_C = K_C \int_{T\,c}^{+\infty} f_C(y)(y - T_C)^2 dy \tag{5.17}$$

5.4　リスク評価のフレームワークの半導体実設計事例への適用

5.4.1 リスク評価のフレームワーク

リスク評価のフレームワークを図 5.1 に示す。このフレームワークは前節までで示したモデル化の手法およびリスク値の算出式をもとに構築されている。すなわち、図中の手順①は、5.2.2 項で示した DSM によるイタレーションのモデル化である。手順③の総ベクトル量 $\boldsymbol{U}$ の算出は式(5.3)で行う。手順⑤はリスク・モデルから導かれるリスク値の算出式（式(5.14)、式(5.15)、式(5.16)、式(5.17)）による。手順の詳細については次項において、シミュレーションによる結果を示しながら説明する。

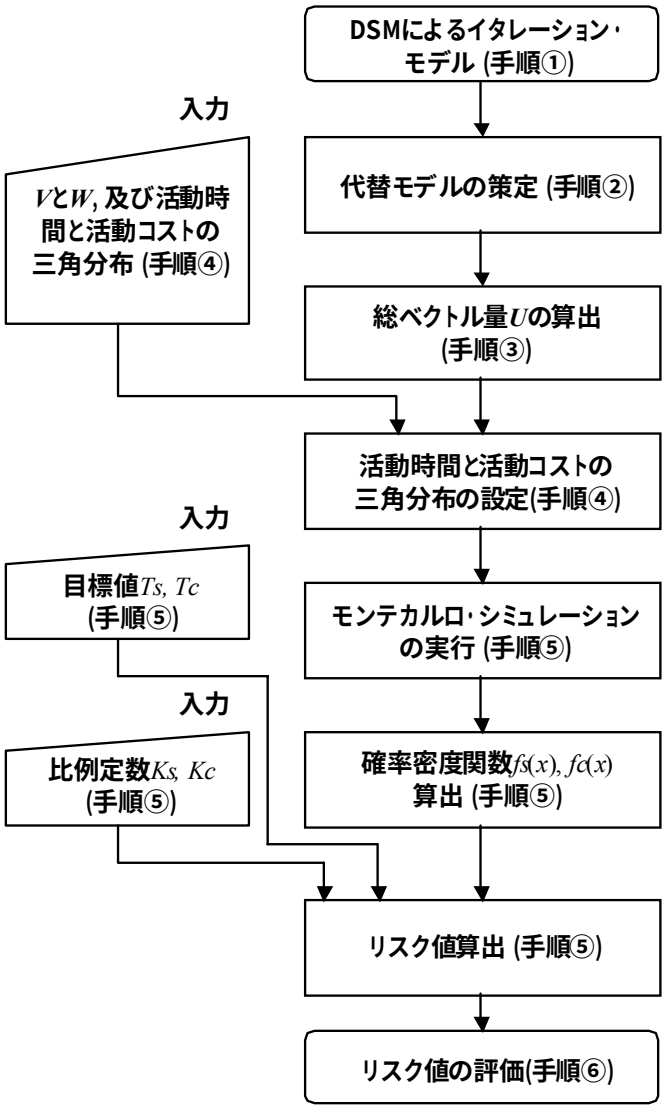


図 5.1　リスク評価のフレームワーク

5.4.2　半導体実設計事例への適用

リスク評価のフレームワークに適用する設計工程は、半導体設計開発の実プロジェクトである。携帯電話のサブパネルに搭載される有機 EL (electroluminescence) (Tang & van Slyke, 1987)ディスプレイを駆動させるシステム LSI (Large Scale Integrated Circuits)を開発対象とする。

シミュレーションにあたっては、実設計事例の他に、その代替モデルとなる設計工程を策定する。それら全ての設計工程にリスク評価のフレームワークを適用し、それぞれのリスク値のを比較して考察を行う。

適用の手順の詳細を、実設計事例への適用の結果とあわせて以下に示す。

手順①：DSM イタレーション・モデルの策定

図 5.2 は半導体実設計事例を DSM により表現したモデルである。プロジェクトを WBS (Work Breakdown Structure:)によってタスクという一つの活動項目に階層的に分解し、それらのタスクを積み上げることで全体を構築・管理する手法(Kerzner, 2001)を用いた。タスク間の依存性の方向は設計データや知識情報の流れに注目して決定した。依存性の定量的な大きさはプロジェクト管理者を除く 5 名の設計技術者が主観的な見地により、3 段階のレベルを変数値として決定した。レベルの認識は 3 段階が限度というのが設計者の共通の意見であった。有機 EL ディスプレイという革新的な製品であったため過去に設計事例がなくデータの蓄積がなかったことと、設計データと知識情報の質と量の絶対的測定方法が確立していなかったことが理由である。尚、変数値はプロジェクト管理者によって集計され、データの標準化を行うべく統計的処理が行われた。この結果、依存性の強度は 0.2、0.1、0.05 の 3 段階とした。

WBS#	活動項目	業務分類	1	2	3	4	5	6	7	8	9	1	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	要求仕様書の策定	ドキュメント作成																								
2	ロー・ドライバ設計とレイアウト	アナログ回路詳細設計・検証	0.1									0.05					0.1									
3	コラム・ドライバ設計	アナログ回路詳細設計	0.1			0.05													0.05				0.05			
4	コラム・ドライバのレイアウト	アナログ回路詳細設計・検証			0.1							0.05					0.1		0.05				0.05			
5	コラムRAM設計	アナログ回路設計	0.1		0.05				0.05																	
6	リファレンス設計	アナログ回路設計	0.1								0.05															
7	コラムRAMレイアウト	アナログ回路設計					0.1					0.05						0.05								
8	DC-DCインタフェース設計	アナログ回路詳細設計	0.1											0.05												
9	リファレンス・レイアウトとインタフェース設計	デジタル回路設計・検証	0.1					0.1								0.1	0.1									
10	設計確認と検証	統合と検証	0.2	0.2	0.2	0.2	0.2	0.2	0.2		0.2			0.05						0.2						
11	ロー・オフ・ジェネレータ設計	アナログ詳細回路設計	0.1																							
12	DC-DCインタフェース・レイアウト	アナログ回路詳細設計・検証								0.1		0.05														
13	オシレータ設計, Rowoff generator レイアウト	アナログ回路設計・検証														0.1										
14	Oscillator レイアウトとインタフェース・レイアウト	アナログ回路設計・検証									0.1				0.1											
15	ポスト・レイアウト・シミュレーション	自動化ソフトによる検証		0.2		0.2			0.2		0.2			0.2						0.1						
16	トップ階層レイアウト	回路図入力	0.1				0.1																			
17	デザインルール検証とGDSII生成	自動化ソフトによる検証と出力		0.2		0.2			0.2		0.2			0.2	0.2	0.2	0.2	0.2			0.05					
18	テスト・プログラム開発	ソフトウエア開発										0.05					0.1									0.2
19	ウエファー製造作業	外部委託																0.05	0.2	0.2			0.05			
20	金バンプ, テスト・プログラム・ローディング	外部委託	0.05																	0.05	0.05					
21	テストと評価	ラボでのマニュアル過程	0.05																			0.05				
22	プロトタイプ出荷と受入	ラボでのマニュアル過程																					0.05			
23	COF/COG工程	外部委託																						0.05		
24	評価作業	ラボでのマニュアル過程																					0.05		0.05	

図 5.2　イタレーションの DSM 表現（実設計事例）

手順②：リスク比較のための代替モデルの策定

設計工程から依存性の強い 3 つのタスク群を抽出し、それぞれに代替案を策定する。それらの代替案を組み合わせることで 8 つの代替モデルを作成する。これらを表 5.1 にまとめた。model1 は 3 つの代替案を適用しないモデルであり、すなわち実設計事例と同じである。尚、追加コストはイタレーションから完全に独立しており、モンテカルロ・シミュレーションの実行前に活動コストの三角分布に反映させている。

第一案は、WBS5 と WBS7 の業務で実現される RAM メモリーのマクロ・ブロックを半導体製造委託メーカが提供する完全モジュール(Clark & Baldwin, 2004)のマクロ・ブロックに代替させることである。これを「マクロの再利用」案とする。これにより WBS5 と WBS7 それぞれがそれまで依存関係にあったタスクとのイタレーションが発生しない状況を作りだすことができる。このイタレーションの解放はマクロ・ブロック間の接続の業務過程が解放されることにつながっている。しかし一方、マクロの再利用としてのライセンス料金の課金が活動コストとして発生する。

第二案は、WBS17 の業務で利用しているソフトを半導体製造委託メーカが導入している高額なソフトに切り替えることである。これを「デザインルールと GDSⅡの互換性」案とする。WBS17 では最終設計データの検証と受け渡しデータの生成の過程で半導体製造委託

表 5.1　代替案と 8 つの設計工程モデル

代替案	適用タスク	代替案概略	適用時の確定追加コスト	設計工程モデル#							
				1	2	3	4	5	6	7	8
第一案「マクロの再利用」の適用可否	WBS5 WBS7	半導体製造委託企業の RAM メモリ・マクロへの代替（外注委託先が提供するモジュール･マクロの利用）	6 百万				■		■	■	■
第二案「デザインルールと GDSII の互換性」の適用可否	WBS17	半導体製造委託企業と完全互換を実現したツールへの代替（共通のプラットフォーム実現）	14 百万		■	■	■				■
第三案「TEG チップの事前準備」の適用可否	WBS2 WBS3 WBS4 WBS8 WBS11 WBS12	「摺り合わせ」設計ブロックの前倒し設計開発による代替（仮想的代替案と時期プロジェクトへの示唆、ロバスト設計とモジュール化がもたらす効果）	13 百万			■		■		■	■

メーカとのイタレーションが頻繁に発生しており、その理由は二者間でデザイン・ルールの検証ソフトに完全なる互換性・等価性がなかった為である。第二案は共通のタスクを介した半導体製造委託メーカとの間でやりとりされた設計データと知識情報のイタレーションの解放につながっている。

第三案は、アナログの詳細設計業務に関するタスクを完全に検証済みの回路に代替することである。これを「TEG チップの事前準備」とする。このアナログ設計業務はブロック内部の構造が「組み合わせ」というより「摺り合わせ」(藤本 et al., 2001; 藤本, 2004)の設計過程を要求するものでありタスク業務間で設計中間データや知識等のやり取りが頻繁に行われる。第三案はプロジェクトの開始に十分時間の余裕があり、アナログ回路を事前に小さなテスト・チップとして実現していた場合を想定している。実プロジェクトでは実現できないが、次回に同様のプロジェクトがあることを期待して設定する。尚第三案のテスト・チップ作成には非量産用で低価格な研究用途・プロトタイプ用途のウエファで動作の検証を行っている前提である。第三案によりロバスト設計(田口, 2000)とモジュール化がもたらす効果を検証することができる。

尚、図 5.1 の DSM 内で依存性の度合いを強く持つタスクとして、WBS10、WBS15、WBS19 が存在している。しかし WBS10 と WBS15 は代替が不可能な自動化ソフトを使った業務である。そして WBS19 はこのシステム LSI を製造するのに必要な製造技術と性能特性のウエファを持っている外注委託先であり、これも代替策は今のところ見つからない。こういった理由でこれら 3 つのタスクについては代替案からはずしている。

手順③：総ベクトル量 $\boldsymbol{U}$ の算出

8 つの設計工程モデルについて、式(5.3)を用いてそれぞれ総ベクトル量 $\boldsymbol{U}$ を算出する。総ベクトル量 $\boldsymbol{U}$ の成分の値は、その成分に対応するタスクがイタレーション収束後にどの程度回転したのか、すなわちイタレーションの実質回転数を表している。また、ベクトル $\boldsymbol{u}_0$ の成分を全て 1.00 と仮定しているので、総ベクトル量 $\boldsymbol{U}$ の成分の合計値は設計工程のイタレーションの実質回転数である。これによりイタレーションによる相対的負荷の大きさを認識できる。手順⑤ではここで算出する総ベクトル量 $\boldsymbol{U}$ をもとに、各モデルのイタレーション収束後の活動時間と活動コストを算出する。

表 5.2 に算出した総ベクトル量 $\boldsymbol{U}$ の成分とその合計値を示す。

手順④：活動時間と活動コストの三角分布の設定

リスクは完全には予見できずまた一元的でもないので、確率的モデルのアプローチを使い、プロジェクトが完了するまでの活動時間と活動コストを確率分布としてより現実的に特徴づける。

まず各タスクの活動時間と活動コストを三角分布により表現する。プロジェクト管理に

表 5.2　設計工程モデル毎の総ベクトル $\boldsymbol{U}$

WBS#	Model1	Model2	Model3	Model4	Model5	Model6	Model7	Model8
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	1.55	1.54	1.00	1.48	1.00	1.45	1.00	1.00
3	1.44	1.24	1.00	1.23	1.00	1.60	1.00	1.00
4	1.85	1.62	1.00	1.56	1.00	2.02	1.00	1.00
5	1.24	1.23	1.17	1.00	1.17	1.00	1.00	1.00
6	1.18	1.18	1.18	1.18	1.18	1.24	1.10	1.00
7	1.37	1.36	1.30	1.00	1.30	1.00	1.00	1.00
8	1.16	1.16	1.00	1.16	1.00	1.26	1.00	1.00
9	1.62	1.61	1.52	1.58	1.52	1.77	1.40	1.20
10	3.65	3.55	2.54	2.99	2.54	3.12	2.00	1.60
11	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
12	1.30	1.29	1.00	1.26	1.00	1.16	1.00	1.00
13	1.13	1.13	1.13	1.13	1.13	1.29	1.10	1.00
14	1.27	1.27	1.26	1.27	1.26	1.29	1.20	1.10
15	2.70	2.65	1.72	2.34	1.72	2.25	1.60	1.40
16	1.22	1.22	1.22	1.00	1.22	1.00	1.10	1.00
17	3.91	1.00	1.00	1.00	2.73	3.58	2.00	1.00
18	1.67	1.66	1.52	1.60	1.52	1.54	1.50	1.20
19	2.23	1.45	1.42	1.42	1.97	2.16	1.70	1.10
20	1.25	1.21	1.20	1.20	1.22	1.29	1.10	1.00
21	1.11	1.11	1.11	1.11	1.11	1.15	1.10	1.00
22	1.06	1.06	1.06	1.06	1.06	1.12	1.01	1.00
23	1.05	1.05	1.05	1.05	1.05	1.08	1.00	1.00
24	1.11	1.11	1.11	1.11	1.11	1.18	1.10	1.00
合計値	38.09	33.72	29.48	31.75	31.78	36.53	29.01	25.60

おけるタスク項目に対する確率モデルは、特に実測データがなく、モデル開発者の判断によらなくてはならない状況ではベータ分布と三角分布が利用されている(Evans & Olson, 1999)。しかしベータ分布を含む数学的関数のパラメトリック分布はパラメータとそれによって規定される分布形との関係が判りづらく、直感的に理解できないことが多い(Vose, 2003)。一方、三角分布に代表されるノンパラメトリック分布は専門家の意見をモデル化する上で信頼度が高く柔軟性がある。また直感的に判りやすく、回答もし易いので実践的である。今回本研究で検討する事例ではパラメトリック的基準のデータを事前に持ち合わせていない。従って本フレームワークでは、三角分布を用いた積み上げ方式を用いて活動時間と活動コストを表現する。

三角分布では全ての活動項目に対して三つのデータ、すなわち最小値 a、最尤値 b、最大値 c を確率変数として表現させる。この三角分布による確率密度関数を $P(x)$ とすると、$P(a)$=0、$P(b)$:最大、$P(c)$=0 であり、次式の関係が成り立つ。

$$\frac{(c-a)(P(b))}{2}=1 \tag{5.18}$$

すなわち各タスクにおいて、活動時間と活動コストはそれぞれがその最小値、最尤値、最大値で表される。従って、ここでは 5.2.3 項で定義したイタレーションを想定しない活動時間 $\boldsymbol{V}$ と活動コスト $\boldsymbol{W}$ のそれぞれについて、最小値、最尤値、最大値を設定する。表 5.3 に実設計事例における各タスクの活動時間と活動コストの設定値を示す。設定値はプロジェクト内のモデル開発者が設計者達と協議して決定した。活動時間とはのべ時間量を表したものである。設定値にはタスク間の知識や情報の流れの時間やコストが織り込まれていない。すなわち手戻り修正等のイタレーションにより生じる活動時間と活動コストに反映されていない。

手順⑤：モンテカルロ・ミュレーションとリスク値の算出

手順④で設定した活動時間と活動コストから、モンテカルロ・シミュレーションによって、活動時間と活動コストそれぞれについて、確率密度関数 $\boldsymbol{f}_S(x)$ と $\boldsymbol{f}_C(y)$ を求める。そして、

表 5.3　活動時間と活動コストの三角分布（実設計事例）

WBS	活動項目	活動時間（日数）			活動コスト（千円）		
		最小値	最尤値	最大値	最小値	最尤値	最大値
1	要求仕様書の策定	6.0	7.0	8.0	2,500	3,000	3,600
2	ロー･ドライバ設計とレイアウト	3.5	4.5	5.8	386	451	580
3	コラム･ドライバ設計	2.1	2.8	5.9	208	277	623
4	コラム･ドライバのレイアウト	4.3	5.4	7.6	514	649	920
5	コラムRAM設計	2.4	3.2	5.2	242	322	524
6	リファレンス設計	5.1	5.9	8.5	635	678	1,016
7	コラムRAMレイアウト	4.0	5.1	6.6	585	658	804
8	DC-DCインタフェース設計	1.7	2.6	4.7	215	258	515
9	リファレンス･レイアウトとインタフェース設計	2.2	2.5	4.3	681	743	1,361
10	設計確認と検証	0.7	0.8	1.6	384	411	521
11	ロー･オフ･ジェネレータ設計	3.5	4.0	6.5	1,900	2,200	3,300
12	DC-DCインタフェース･レイアウト	1.9	2.3	3.5	1,386	1,540	3,079
13	オシレータ設計, Rowoff generator レイアウト	4.4	6.2	7.5	2,750	2,927	3,459
14	Oscillator レイアウトとインタフェース･レイアウト	3.9	5.5	7.1	2,433	2,590	3,217
15	ポスト･レイアウト･シミュレーション	4.6	5.2	7.0	2,773	2,958	5,177
16	トップ階層レイアウト	5.3	5.7	6.5	4,085	4,493	4,902
17	デザインルール検証とGDSII生成	3.3	3.6	5.1	1,022	1,150	1,916
18	テスト･プログラム開発	7.2	8.4	9.9	537	597	896
19	ウエファー製造作業	18.3	18.8	20.1	9,397	9,845	10,069
20	金バンプ, テスト･プログラム･ローディング	4.8	5.6	16.1	1,124	1,204	1,606
21	テストと評価	13.5	15.3	19.8	4,495	5,394	5,844
22	プロトタイプ出荷と受入	2.8	3.8	4.7	43	47	52
23	COF/COG工程	11.4	13.3	17.1	6,649	7,124	8,549
24	評価作業	1.8	2.7	5.4	361	406	812

式(5.14)と式(5.16)から生起確率 $P_S(x)$と $P_c(x)$を、式(5.15)と式(5.17)からリスク値 R_Sと R_Cを算出する。

モンテカルロ・シミュレーションにはラテン・ハイパーキューブ法(McKay et al., 1979; Stein, 1987)を用いる。シミュレーションの試行回数を多く重ねる程、出力されるサンプル回数が増え確率分布は安定する。確率累積分布関数の平均と分散が明確な値(1%未満)に収まった時を確率分布が安定したと判断する。つまり、シミュレーションの試行回数を r、活動時間の平均（期待値の概念に一致）を $E[S_r]$とし、その標準偏差を $\delta[S_r]$とするなら、

$$\frac{\left|E[S_r]-E[S_{r-100}]\right|}{E[S_{r-100}]}<0.01 \tag{5.19}$$

$$\frac{\left|\delta^2[S_r]-\delta^2[S_{r-100}]\right|}{\delta^2[S_{r-100}]}<0.01 \tag{5.20}$$

を満たした時の試行回数を選択する。活動コストについても同様の精度の試行回数を選択する。

目標値 T_S、T_Cはプロジェクトが開始される前に設定される。この目標値は今回、プロジェクト管理者が顧客及び関係者と調整して得られた数値を用いた。尚、損失関数の比例定数 K_Sの値、100 万円はプロジェクト管理者が設定した。この値は本事例に携わる企業が過去に設計開発した携帯電話のサブディスプレイ用液晶ドライバ・プロジェクトの比例定数の平均値である。一方損失関数の比例定数 K_Cは、本事例の場合には目標値を超えた活動コスト分以外に発生する経済的損失が見あたらず、目標値を超えた活動コストを経済的損失金額として計算している。

表 5.4 に 8 つの設計工程毎のシミュレーション結果、リスク値の計算結果を示す。

表 5.4　設計工程モデル毎のシミュレーション結果

			設計工程モデル＃							
			1	2	3	4	5	6	7	8
目標値	活動時間　(日数)	Ts	200							
	活動コスト(百万円)	Tc	86							
モデルの出力	シミュレーション実行回数	run	1200	1200	1300	1200	1100	1200	1300	1100
	平均値 (活動時間)	E(s)	229.6	199.9	182.3	191.2	199.9	223.1	184.5	161.0
	標準偏差 (活動時間)	σs	5.44	5.1	4.8	4.9	5.0	5.3	4.6	4.1
	平均値 (活動コスト)	E(c)	82.9	84.7	91.8	87.5	85.6	85.5	85.5	89.8
	標準偏差 (活動コスト)	σc	1.9	1.7	1.3	1.5	1.4	1.6	1.3	1.1
	歪度 (活動時間)	γs	0.15	0.12	0.07	0.26	0.19	0.06	0.23	0.19
	歪度 (活動コスト)	γc	0.24	0.25	0.16	0.41	0.27	0.18	0.19	0.19
	尖度 (活動時間)	ks	2.98	2.95	2.73	3.21	2.74	2.68	2.68	2.84
	尖度 (活動コスト)	kc	2.97	2.54	2.60	3.01	2.84	2.63	2.66	2.8
	目標値を外れる確率 (活動時間)	P(s)	99.0	46.8	2.0	3.9	45.0	98.0	2.0	2.0
	目標値を外れる確率 (活動コスト)	P(c)	5.9	25.9	99.0	84.7	38.7	38.5	35.2	99.0
	リスク値 (活動時間)	Rs	208	34.7	3.2	6.2	36.2	161	3.2	3.2
	リスク値 (活動コスト)	Rc	0.6	4.6	63.4	52.9	13.9	12.1	12.7	97.0

手順⑥：リスク値の評価

1) 目標値の設定と目標値に収まる確率、リスク値

製品設計開発のプロジェクト管理者は様々な選択肢を考え、活動時間と活動コストの目標値を決定する。目標値の設定は設計開発工程におけるリスク値に大きな影響を与える因子であるので、目標値の選択は顧客との交渉時又は関係者との調整時に、受容リスクを客観的にかつ定量的に双方で確認する。この際、目標値を異なる値に変化させた場合の、リスク値や目標値に収まる確率がその判断材料となる。

手順⑤では目標値を定数として算出した。ここでは異なる目標値を設定し、それぞれについて目標値に収まる確率とリスク値を算出した。図 5.3 は model5 における、活動コストの目標値に対する目標値に収まる確率とリスク値の関係を表す。例えば図中に印をつけた 2 つの目標値に注目する。活動コストの目標値が 86.0 百万円のとき、目標値を外れる確率は 45.0%(目標値以内に収まる確率は 55.0%)であり、活動コストのリスク値は 13.9 百万円となる。これらの結果は表 5.4 でも示した。活動コストの目標値が 87.5 百万円のとき、目標値を外れる確率は 10.0%(目標値以内に収まる確率は 90.0%)であり、活動コストのリスク値は 9.0 百万円となる。

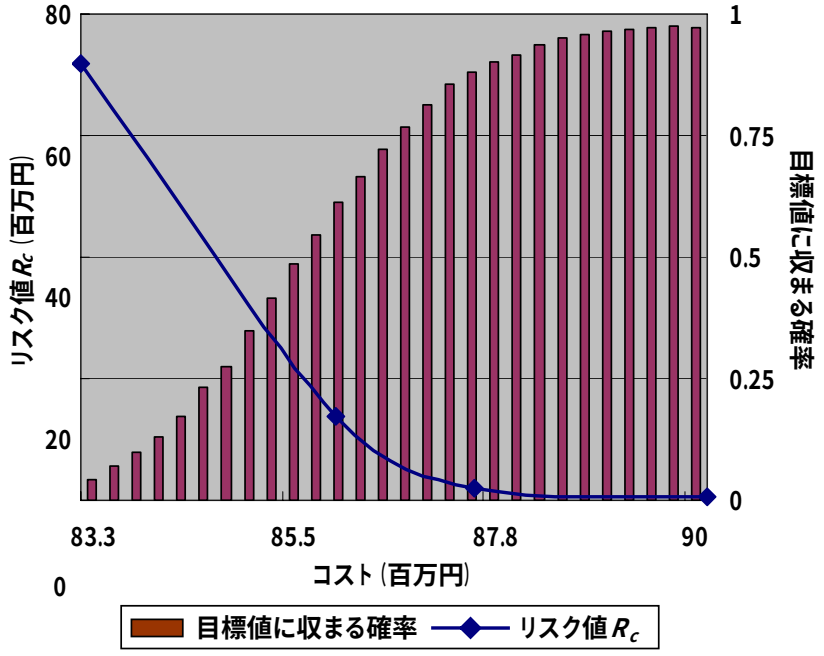


図 5.3　目標値の設定と目標値に収まる確率、リスク値

プロジェクト管理者は目標値の選択に際し、これらの値を比較することで意思決定することができる。特に、リスク値はその指標が活動時間あるいは活動コストと同じであるため、比較が容易である。Vose によれば、数々のプロジェクト管理の蓄積から得られた知見として、リスクの生起確率が 50%前後（目標値に収まる確率が 50%前後）であればかなりの努力と優れた管理能力とある程度の幸運がなければ回避できないリスクが存在するとされ、またリスクの生起確率が 20%以上の場合（目標値に収まる確率が 80%を超える場合）には優れたプロジェクトの管理によって回避可能なリスクが含まれるとされる(Vose, 2003)。確率だけの情報ではその専門家の存在なくしては定性的なリスク評価さえできない。しかし顧客や関係者を含め受容リスクの境界、すなわち最大期待損失量の設定ができるならば、意思決定に向けた情報の一つとなりえる。活動コストのリスク値だけでなく、活動時間と活動コストの対比やモデル同士の対比評価も簡単に行うことができる。

2) リスク値のトレードオフ検討

活動時間と活動コストは一般的にトレードオフの関係にあり(Browning & Eppinger, 2002; Ahmadi et al., 2001)。設計におけるリスク管理はトレードオフ検討の延長上にあり、設計者はプロジェクト管理とリスク管理の活動双方に終始深く関与していなければならない(富田&平野, 2001)。活動時間と活動コストのトレードオフ検討はプロジェクト管理とリスク管理にとって重要である。しかし、活動時間と活動コストのトレードオフ検討により、プロジェクトの意思決定を実践することは容易ではない。その理由の一つとして、活動時間と活動コストという二軸の指標が異なっている点が上げられる。活動時間と活動コストの指標が同じであれば、それらのトレードオフ検討はより実践的であり、プロジェクトの意思決定にとってより有用であると言える。

多様な代替案をモデル化しそれらの活動時間と活動コストを対比させることでトレードオフ検討が可能となる。本フレームワークのシミュレーションでは、8 つの設計工程モデルにおいて、タスクの代替案の組み合わせ方に応じて、それぞれ異なる値の活動時間のリスク値と活動コストのリスク値を得た。これらをトレードオフ検討してみる。

8 つの設計工程モデルにおける活動時間のリスク値 R_S と活動コストのリスク値 R_C を抽出し、累乗近似曲線で示したグラフが図 5.4 である。これらのリスク値は、活動時間と活動コストでは同じ指標すなわち百万円を単位として算出されている。タスクを代替させたときの追加コストに対し、活動時間のリスク値と活動コストのリスク値はそれぞれどの程度増減するかを、同じ指標でもって比較することができる。そして受容リスク値の制約設定を予め取り決めしておくことで、最適なモデルの選択が可能となる。また、双方のリスク値が 0 になる方向に位置する設計工程のモデル化の検討を重ねることで、設計工程の改善につなげることができる。

よって本フレームワークは、多様な代替モデルにより活動時間と活動コストそれぞれのリスク値のトレードオフ検討を可能にするもので、プロジェクトの意思決定にとって実践

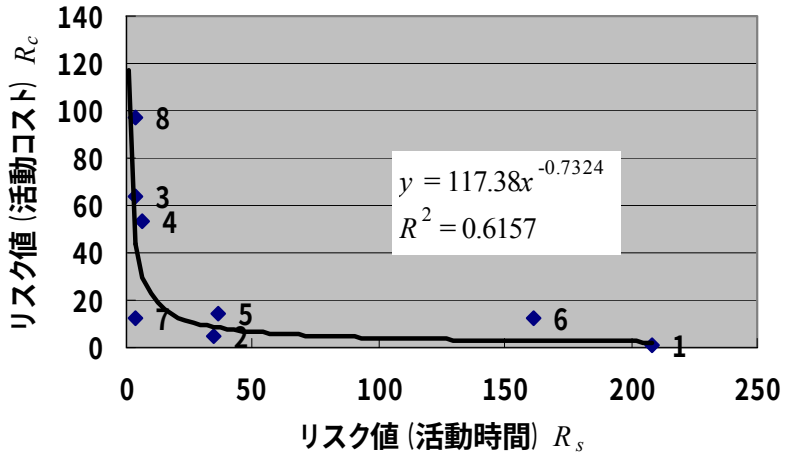


図 5.4　活動時間と活動コストのリスク値結果

的、具体的な判断基準を与えるものであると言える。

5.5　考察

5.5.1　リスク評価のフレームワークの運用

本フレームワークでは、設計開発工程の代替モデルを策定することで、様々な場面や状況を作り出すことができた。そして、顕在化する確率は低いものの、それが顕在化した時に大きな負荷や損失を引き起こす可能性のあるタスクやシナリオを想定して、活動時間と活動コストの影響度合いを定量的に確認することもできた。

実際、本フレームワークを適用した半導体実設計事例は、タスク間の知識と技術情報のフィードバックと反復プロセスが複雑に数多く存在し、イタレーションが大きすぎてプロジェクトが収束しづらい事例である。これをモデル化したことで、イタレーションを生じさせるタスク群を特定でき、プロジェクト管理者が本プロジェクトで最も懸念していた状況を定量的に捉えることができた。そして、タスクを恣意的に置き換えた代替モデルとの比較により、活動時間と活動コストにおいてどの程度の定量的乖離があるのか確認できた。プロジェクトの活動コストだけを最優先し、活動時間の結果を度外視することは実践上考えられないが、シミュレーションを通じてプロジェクト管理者は検証することができた。

本モデルはプロジェクトを進行させた後にも適用できる。例えばプロジェクトは進行してから工程の変更・修正を迫られる場合がある。この場合には残りの設計工程の構成を再構築し、モデルの更新とシミュレーションの再適用を行い、活動時間と活動コストをあらためて予見することができる。すなわち新たに変更・修正したタスクが全工程に与える活動時間と活動コストそれぞれのリスクについて議論することも可能となる。

5.5.2　リスク・モデルの信頼性

プロジェクトの管理上重要なリスク・モデル自身の信頼性について検討する。本章で議論したリスク・モデルによって算出された結果は、入力変数、損失関数の比例定数、そしてモデル自身の計算精度により信頼性が左右される。入力変数はフレームワークの適用手順で述べてきた通り、プロジェクト管理者がモデル開発者と設計技術者と協議を重ねて評価算出した実測値であり精度は高い。しかし比例定数は式(5.10)で示した通りプロジェクト固有の経済的損失量に起因し、事前に評価することが困難であった。その為にプロジェクト管理者が過去に実施した類似のプロジェクトを評価して得た平均値を利用した。最終的にモデルの信頼性には実践で用いられた model1 を使ってプロジェクト管理者が評価した。比例定数を先の平均値を用いて実践上の評価算定したところ、活動時間のリスク値が 2 億円、活動コストのリスク値が 0 円という結果となり、ほぼ model1 のシミュレーション結果と同等であることが確認された。

5.5.3　リスク評価のフレームワークにおける課題

リスク・モデルの適用にあたり議論されるべき課題が残る。それは該当プロジェクトの分野における損失関数の比例定数がリスク・モデルの信頼性上大きな決定因子となっている事である。リスクの定量的評価手法の実現と確立のために、類似の事例によるデータの収集と蓄積を行う必要がある。

5.6　本章のまとめ

昨今のシステムは大規模で複雑化しており、製品設計開発においては、設計工程の効率性、生産性、予見性の向上への取り組みと同時に、活動時間と活動コストのリスクを先読み、その見積りと評価を行うリスク・アセスメントがますます重要となってきている。リスク・アセスメントの研究は現在まで数多く重ねられてきた。それらの多くはクリティカル・パス解析によって活動時間や活動コストのリスク評価を行ったり、リスクの生起確率を算出したりするものであった。しかし、リスク・アセスメントにはリスクの生起確率だけでは不十分であり、リスクが招く経済的損失量の評価が有用である。また、リスク・アセスメント研究は自動車産業や宇宙航空産業については進んでいるが、半導体設計開発のプロジェクトに対しての報告はそれと比較して少ない。半導体設計開発工程には、タスク間に知識と技術情報のフィードバックと反復プロセスといったイタレーションが複雑に存在しており、そのモデル化が従来の手法では困難であることが理由としてあげられる。

そこで本章では、DSM (Design Structure Matrix:設計構造マトリックス)によって設計工程をモデル化し、そのモデルから活動時間と活動コストそれぞれのリスクを経済的損失量と

して算出する、定量的リスク評価のフレームワークを提案した。DSM はイタレーションを含む設計工程をモデル化できるとして近年注目されている手法である。経済的損失値はリスク・モデルに基づき得られる値であり、その確率分布関数の算出にはモンテカルロ・シミュレーションが用いられる。

また、本章では提案したフレームワークを半導体設計工程の実プロジェクトに適用してリスク評価のシミュレーションを行い、本フレームワークの有用性を検証した。適用にあたっては、設計工程のタスクに対して 3 つの代替案を用意し、それらの組み合わせによって 8 つの設計工程モデルを策定した。そしてリスク評価では、8 つのモデルにおいてそれぞれの、活動時間のリスク値と活動コストのリスク値をトレードオフ検討することが有用であることを示した。またある 1 つのモデルにおいて、目標値の変化に対するリスク値とその生起確率の増減を観測することが、目標値の設定に利用できることを示した。これらの結果から、本フレームワークは開発時間、活動コスト、不確実性の一要因であるイタレーション、そしてリスクに対して実践的な洞察を提供してくれるものであり、またプロジェクトの策定と変更・修正といった意思決定を側面から支援してくれるものであることが検証された。イタレーションを定式化できる設計プロジェクトという条件下、本方式は確率論的な意思決定が行えるという点で従来の決定論的な方式より優れていると言える。

参考文献

[1] Ahmadi, R., Roemer, T. A. and Wang, R. H.: “Structuring product development process”, European Journal of Operational Research, 130, pp. 539-558 (2001)

[2] Alexander, C.: “Notes on the Synthesis of Form”, Cambridge, MA: Harvard Univ. Press (1964)

[3] Browning, Tyson R. and Eppinger, Steven D.: “Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development”, IEEE Transactions on Engineering Management, 49, 4, November, pp. 428-442 (2002)

[4] Chapman, C. B., Phillips, D. E., Cooper, D. F. and Lightfoot, L.: “Selecting an approach to project time and cost planning”, International Journal of Project Management, l.3, 1, pp. 19-25 (1985)

[5] Chapman, C. and Ward, S.: “Project risk management; processes, techniques and insights, 1st ed.”, Chichester, UK, John Wiley (1997)

[6] Clark, Kim B. and Baldwin, Carliss Y.（安藤晴彦訳）: “デザイン・ルール―モジュール化パワー”,東洋経済新報社, pp. 75-109 (2004)

[7] Cooper, K. G.: “The rework cycle: Why projects are mismanaged”, PMNetwork, Feb. (1993)

[8] Eppinger, Steven D., Whitney, D. E., Smith, R. P., and Gebala, D. A.: “A model based method for organizing tasks in product development”, Res. Eng. Design, 6, pp. 1-13 (1994)

[9] Eppinger, Steven D.: “Innovation at the Speed of Information”, Harvard Business Review, 79, 1,

January, pp. 149-158 (2001)

[10] Evans, James R. and Olson, David L.（服部正太訳）: “リスク分析・シミュレーション入門”, 構造計画研究所 (1999)

[11] 藤本隆宏,青島矢一,武石彰編: “ビジネス・アーキテクチャ―製品・組織・プロセスの戦略的設計”,有斐閣, pp. 3-26 (2001)

[12] 藤本隆宏: “日本のもの造り哲学”,日本経済新聞社, pp. 144-151 (2004)

[13] Gantt, H. L.: “Work, Wages, and Profits”, The Engineering Magazine, New York (1910)

[14] Henderson, R. M. and Clark, K. B.: “Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms”, Administrative Science Quart., 35, pp. 9-30 (1990)

[15] von Hippel, E.: “Task partitioning: An innovation process variable”, Research Policy, 19, pp. 407-418 (1990)

[16] von Hippel, E.: “Sticky Information and the Locus of Problems Solving: Implication for Innovation”, Management Science, 40, 4 p. 429 (1994)

[17] Kaplan, S. and Garrick, B. J.: “On the quantative definition of risk,” Risk Analysis, 1, pp. 11-27 (1981)

[18] Kerzner, Harold: “Project Management Workbook”, John Wiley & Sons (2001)

[19] Kline, S. J.: “Innovation is not a linear process”, Res. Mgt., July-Aug., pp. 36-45 (1985)

[20] 小林英男: “失敗とリスク・マネジメント”,日本機械学会誌, 103, 980 (2000)

[21] McKay, M. D., Beckman, R. J., and Conover, W. J.: “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code”, Technometrics, 21, 2, pp. 239-24 (1979)

[22] Nightingale, P.: “The product-process-organization relationship in complex development projects”, Res.Policy, 29, pp. 913-930 (2000)

[23] Patterson, Fiona D. and Neailey, Kevin: “A Risk Register Database System to aid the management of project risk”, International Journal of Project Management, 20, pp. 365-374 (2002)

[24] Pierce, I. N.: “Accounting for the risks”, Civil Engineering, 681, 6 (1998)

[25] Reid, S. G.: “Risk assessment”, The University of Sidney, School of Civil and Mining Engineering, Research Report No.R591 (1989)

[26] Restrepo, L. F.: “Combining qualitative and quantitative risk assessment results into a common risk measure”, Risk Assessment and Safety, ASME, pp. 3-14 (1995)

[27] Sage, Andrew P.（三森定道,明石吉三訳）: “システムズエンジニアリング”,日刊工業新聞, pp. 229-309 (1997)

[28] Schumpeter, Joseph, A.（中山伊知郎,東畑精一訳）: “資本主義・社会主義・民主主義”,東洋経済新報社, p. 106 (2001)

[29] Smith, Robert P. and Eppinger, Steven D.: “Identifying Controlling Features of Engineering Design Iteration”, Management Science, 43, 3, March, pp. 276-293 (1997)

[30] Smith, R. P. and Morrow, J. A.: “Product development process modeling”, Design Studies, 20, 3, pp. 237-261 (1999)

[31] Stein, M. L.: “Large sample properties of simulations using latin hypercube sampling”, Technometrics, 29, 2, pp. 143-151 (1987)

[32] Steward, D. V.: “Systems Analysis and Management”, Structure, Strategy, and Design, New York, PBI (1981)

[33] 田口玄一: “品質の管理”,日本規格協会 (1969)

[34] 田口玄一: “ロバスト設計のための機能性評価　効率的開発の方法”,日本規格協会 (2000)

[35] Tang, C. W. and van Slyke, S. A.: “Organic electroluminescent diodes”, Appl. Phys. Lett. No.51, p. 913 (1987)

[36] 冨田信之,平野重雄: “設計とリスク・マネジメントに関する考察と提案”,日本設計工学会論文誌, 36, 7 (2001)

[37] Vose, David.（長谷川専,堤盛人訳）: “入門リスク分析”,勁草書房, pp. 361-365 (2003)

[38] Wang, John X. and Roush, Marvin L.（日本技術士会訳）: “リスク分析工学”,丸善, 47 (2003)

[39] Webb, A.: “Software for risk analysis in business projects Part 2”, Engineering Management Journal, 7, pp. 35-42 (1997)

[40] Whitney, D. E.: “Designing the design process”, Research Engineering Design, 2, pp. 3-13 (1990)

[41] 山本靖: “ワーク変換行列によるイタレーション・モデルと半導体設計工程への適用に関する研究”, Design Symposium 2004 講演論文集, 7-8 月, pp. 301-308 (2004)

[42] 横山倉三: “コスト見積管理”,化学工業社 (1976)

[43] 吉田憲司: “IC の設計と製造のはざまで”,システム LSI 設計技術研究会,情報処理学会, 2003-SLDM-112 (1) (2003)

第6章　　無駄な工程・過剰機能・過剰品質と価値のダイナミック・モデル

6.1　　リーン思考と設計開発工程の効率化

市場競争が激しい領域において、製品開発を迅速に行うことは企業の成功を決める要因の一つである(Blackburn, 1991; Ahmadi el atl., 2001)。そして産業全般において企業が原価面で競争上の優位性を獲得することは基本的な戦略である。そのため企業は計画コストの制約条件下で、製品開発における設計開発工程の期間を短縮させる方策を実施し、所要時間と所要コストの効率化を達成しようとする。しかしながら設計開発工程の時間短縮とコスト低減は二律背反の問題であり、同時に改善させることは困難である。従って所要時間と所要コストを同時に効率化させる検討を行うことは製品開発において有効と言える。

設計開発工程の効率化を実現させた例証としてリーン思考による効率化事例がある。リーン思考はトヨタの生産方式の原点として知られている原則である。Womack et al. (1991)によって、そのシステムが研究され、体系化された(Womack et al., 1991)。それ以降リーン思考に関する研究が重ねられてきた。リーン思考は価値を生まない活動を排除し、価値を最大化させる活動を促進する。この価値の最大化のねらいが結果として所要時間と所要コスト双方を同時に効率化させている。

そこで本章では、リーン思考を拠り所とした価値のダイナミック・モデルを提案する。価値のダイナミック・モデルは、メーカ視点と顧客視点、そしてそれらを統合した視点から捉えた製品価値の、プロセス量（所要時間・所要コスト）に対する変化を表現したものである。つまり、設計開発工程の効率化を検討するための指標として、所要時間と所要コストの他に、製品価値に注目したものである。

本章では、まず二律背反のメカニズムとトレードオフ検討に関する実証研究について概論する。そしてトレードオフ検討の限界と抜本的効率化に向けた設計開発工程のあり方について議論する。そこでは設計開発工程の多様性について、製品のアーキテクチャの概念を援用して論ずる。次に、リーン思考についてその基本原則と意味を述べる。そしてその

原則を適用した効率化事例を取り上げる。更に、リーン思考の出発点である製品価値についてその定義と形成過程を示す。最後に、価値のダイナミック･モデルとその利点を示し、設計開発工程における無駄な工程の問題と過剰機能・過剰品質の問題を解析する。

6.2　設計開発工程の効率化に関する先行研究

6.2.1 二律背反のメカニズム

製品の設計開発を迅速に行うことは、企業が競争優位に立つ上で有効である。特に市場競争が激しい領域においてかつプロダクト･イノベーション(Schumpeter, 1980)が要求される製品を設計・開発しなければならない状況下では、企業は製品開発期間の時間を短縮させるための努力を惜しまない。これは市場投入の遅れが売上げに影響を及ぼすからである。しかし研究開発や製品設計のプロジェクトでは製品開発期間を短縮させようとすると、プロジェクト自身のコスト負担が加速する(Ahmadi et al., 2001; Graves, 1989)。このメカニズムは様々な方策から議論されてきた。

製品開発における納期を短縮させる方策として、まず人材を追加投入して速成をさせるリソース投入型開発（crash）がある。これは突貫作業である。Brooks (1975)は人材リソースを追加投入して速成させる突貫作業は結果として収穫逓減を招くとしている(Brooks, 1975)。これは人材リソースが増えるに従って、人材間のコミュニケーションの量が増えるからである。一つのタスク(task)を n 人の人材に分割して作業させると、人材間の一対のコミュニケーションの負荷は $n(n-1)/2$ と n の二乗で逓増するので、一人当たりの生産性は逓減する。従ってこのコミュニケーションの負荷を解決するために新たな人材やリソース投入が行われ、コスト負担が生じる。

もう一つの方策として、オーバーラップ型開発（overlap）がある(Clark & Fujimoto, 1989; Roemer et al., 2000)。図6.1に示す。これは順序が決まっている逐次処理(sequential)型の作業を部分的に重ね合わせることで同時並行的に進行させることである。つまりタスクが終了すると次のタスクが開始する逐次処理型の作業を、タスクが終了する前に次のタスクを開始させることである(Graves, 1989)。この方策によりタスクを前倒しできるので納期を短縮できる。しかし各タスクは事前に終了しているタスクの情報全てを得ずに次のタスクを開始しているので予期せぬ問題が各タスクで持ち上がってしまう。これが頻繁に手戻り修正を引き起こし、追加のコスト負荷が生じる(Roemer et al., 2000)。研究開発の過程はオーバーラップ型開発の一事例と言える。研究開発はそもそも不確実性の世界で取り囲まれており、発見的教授法の過程をとらなければならない。研究開発の過程で成果物の期待を早めさせようとすると誤認と追加作業を招いてしまう(Graves, 1989)。つまりコスト負荷につながる。

リソース投入型開発とオーバーラップ型開発は共に製品開発における納期を短縮させることができる。しかし同時にコストの追加負担を生じさせてしまう。この二律背反問題が設計のジレンマである。

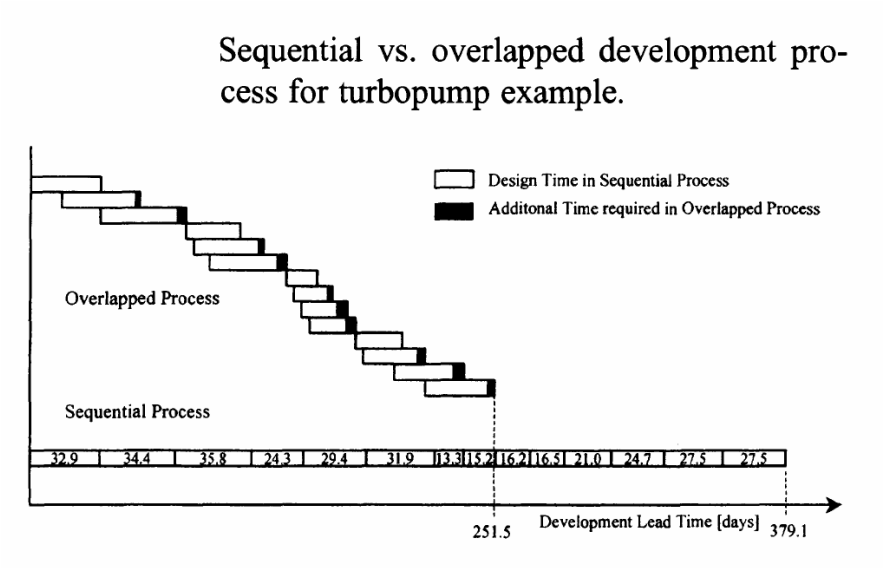


図 6.1　逐次処理型とオーバーラップ型開発(Roemer et al., 2000)

6.2.2 トレードオフ検討に関する実証

リソース投入型開発とオーバーラップ型開発といった方策に見られるように、多様なタスクを作り出すことで、設計開発工程の代替案が創出できる。そして幾多もの代替案の中から最適解を求めようとする行為をトレードオフ検討と言う。

トレードオフ検討に関する実証的研究はネットワーク理論と確率統計理論を使って過去に多く研究が重ねられてきた(Scherer, 1966; Mansfield, 1971; Putnum & Fitzsimmons, 1979; Eastman, 1980; Boehm, 1981)。Scherer (1969)は確率的見地から理論的モデルを構築し、Mansfield (1971)は化学系、電気系、機械系企業11社のイノベーション創出された29個のプロジェクトから所要時間と所要コストは傾きが負で下に凸の曲線になることを検証した(Scherer, 1969; Mansfield, 1971)。 Putnum & Fitzsimmons (1979)や Boehm (1981)はソフトウエアに特化して Mansfield(1971)同様の検証結果を得ており、そこからソフトウエアのプロジェクトのコストを予測できる独自のモデルを構築している。そして Boehm (1981)は観測データから通常のソフトウエアの開発における所要時間はその75%が納期を短縮できる限界点であると指摘している。

Ahmadi et al. (2001)は宇宙航空プロジェクトでオーバーラップ型の工程をモデル化し、所要時間と所要コストのトレードオフから最適解を求めることを目的とした実証研究を行っている(Roemer et al., 2000)。前掲の図6.1はその実証結果である。本事例は宇宙航空エンジン

用ターボポンプの設計開発である。逐次処理型をオーバーラップ型開発で置き換えた時のそれぞれの納期の対比と、オーバーラップ型開発時に発生した各タスクの追加負荷量が示されている。

二律背反の検証は半導体の製品開発でも行われている。昨今の半導体やハードウエアといった製品設計・開発を取り巻く環境として、製品自身が受ける市場と技術の圧力の存在がある。これは製品の大規模化と複雑化を要求し、同時に短納期で出荷を迫る圧力である。実際のところ製品の大規模化と複雑化は製品設計開発工程におけるタスク数を増大させる。そしてそれらタスク間で知識と技術情報の交換が大きくなることで製品開発の所要時間が押し上げられている。これはタスク間の相互依存性の問題である。デザイン・ルール(Clark & Baldwin, 2004)を規定し、外部の半導体 IP (Intellectual Property)をモジュールとして利用することで、タスク間の相互依存性を減少させることが可能となる。これで所要時間は短縮できるが、所要コストの負担は増大する。つまりハードウエア設計においてもトレードオフの関係は存在する。このことは有機 EL (Electronic luminescence)ディスプレイ・ドライバー半導体設計開発プロジェクトに対するシミュレーションで検証されている(山本, 2004)。図6.2はこのシミュレーション結果であり、所要時間のリスク値 R_S と所要コストのリスク値 R_C の関係を累乗近似曲線で示したグラフである。この近似曲線を経済学的視点で無差別曲線ととらえ、受容リスク値制約線の設定を予め取り決めしておくことでトレードオフ上の最適モデルの選択が可能となる。

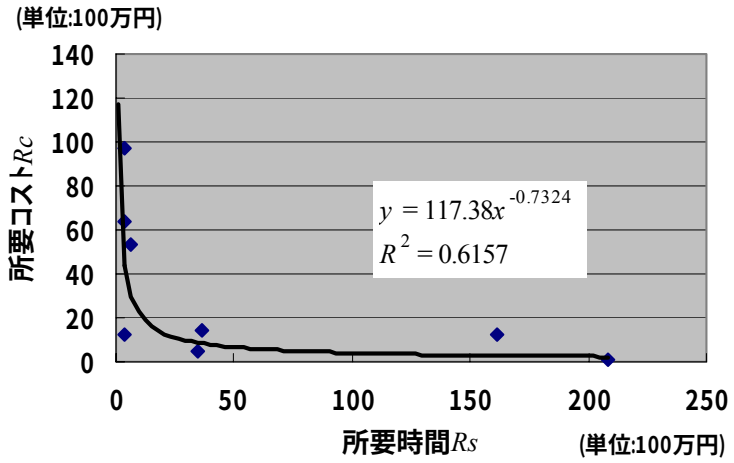


図 6.2　半導体開発における時間とコストの無差別曲線

6.2.3　トレードオフ検討の限界

リソース投入型開発やオーバーラップ型開発の中で、製品開発工程の既存タスクを適用し、代替案をトレードオフ検討することは、既存の設計開発工程の枠の中での議論である。

本来企業の製品開発における競争力を堅持させる有効な手段は、所要時間と所要コスト双方の低減を実現できる製品設計開発工程のモデルを創造することであり、これが抜本的な効率化につながる。つまりこれまでの議論は図6.3の無差別曲線上に留まるものであるが、本研究の関心は設計開発工程を抜本的に効率化することで、無差別曲線を矢印の方向に押し出すことである。

設計開発工程の創造とはイノベーションである。製品のアーキテクチャという概念を援用して理解するならば、設計開発工程とはタスクの集合である。タスク自身は多様化でき、タスクを要素として持つ構造モデル、つまり設計開発工程も多様な代替案が創造できる。そして多様な設計開発工程が存在するならば、それぞれに従属した所要時間と所要コストが多様に存在する。これは製品のアーキテクチャが多様化すればそのアーキテクチャに従属した製品の性能や効用が多様化するのに似ている。

尚、ここでいうアーキテクチャとは、製品や設計開発工程の性質を理解するためのシステム的な概念である。この性質のとらえ方は「分け方とつなぎ方」に着目する。つまり全体をどの様に切り分け、部分をどの様に関係づけるかということである(Clark & Baldwin, 2004)。Ulrich (1995)や青島(1998)によれば、アーキテクチャを「構成要素間の相互依存関係のパターンで記述されるシステムの性質」と定義している(Ulrich, 1995; 青島, 1998)。このシステムの性質が製品の場合には製品の性能や効用であり、設計開発工程であれば所要時

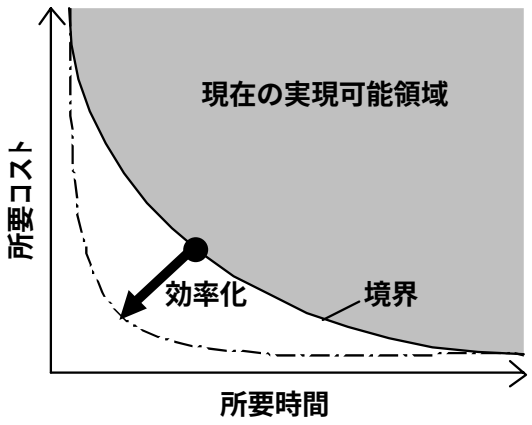


図 6.3　設計開発工程の効率化の方向

間と所要コストということになる。製品の多くはそのアーキテクチャのイノベーションなしには改善されないのであるから、製品の性能や効用を高めるには製品のアーキテクチャのイノベーションを創造することにある。同様に、設計開発工程の所要時間と所要コストの改善には設計開発工程というアーキテクチャのイノベーションを創造することである。

アーキテクチャの概念から設計開発工程が多様に存在しうるというのであれば、問題はどのような基軸でもってそれらの設計開発工程を比較し、最適解を得るかということである。トレードオフ検討は所要時間と所要コストの二つの基軸による設計開発工程の効率化と言える。しかし、設計開発工程にはこの二つ以外に計るべき尺度、指標が多数存在するはずである。それならば、その中から指標を一つ選び、設計開発工程の効率化を検討するもう一つの基軸として追加すれば、トレードオフ検討では解析が困難な問題にも対応できるはずである。

そこで、次節では設計開発工程のアーキテクチャを創造し、効率化を実現させた日本の自動車メーカの成功事例を取り上げる。それは価値を基軸とするリーン思考を拠り所にしたものである。

6.3　効率化事例とリーン思考

6.3.1 日本の自動車メーカによる効率化事例

1980年代、アメリカの企業と比較して、日本の自動車メーカの製品開発はかなり柔軟かつスピーディーに行われていた(藤本, 1998)。そして特に1990年代中盤から、日本の自動車メーカは大幅な開発期間短縮に取り組んできた(Womack et al., 1991)。実際、欧米企業に比べて日本企業の開発工数は少なく、コンセプト検討開始から発売までの開発期間も短い(藤本&延岡, 2004)。さらにプロジェクト・メンバー数が大幅に少ない(藤本, 1998)。これは技術者の職務範囲が広いことが一因である(Clark & Fujimoto, 1991)。

Womack et al. (1991)が行った日米の同等車種の比較調査によれば、米国の GM-10の開発期間が7年であるのに対し、日本のアコードの開発期間は4年であった。そして GM-10は製造が容易ではなく、コスト高だったという。Clark & Fujimoto (1991)は日本と欧米の自動車メーカのクリーンシートを対象として調査を行っている。開発期間に関しては欧米が60ヶ月に対して、日本は46ヶ月であった。そしてコストに直接影響を及ぼすのべエンジニアリング時間は欧米が310万時間に対して、日本は170万時間であった。つまり日本の自動車メーカは開発時間の短縮化だけでなく、開発の所要コストも低減させている。自動車のもの造りの基本原理は世界中どこでも変わらない中、所要期間とコストを同時に削減させた開発技術力という事実は、前述の定説を覆す発見と言える(藤本, 2003)。

この様に日本の自動車メーカが開発生産性において過去20年間常に優位にあるのは、フロントローディング(Front-Loading)という手法による改善ができているからである(藤本&延岡, 2004)。フロントローディング手法とは、設計上の諸問題を製品開発のより早い時期に

認識し解決することによって、開発期間とコストの削減をはかる戦略である(Thomke & Fujimoto, 2000; 竹田, 2000)。前述のオーバーラップ型開発とフロントローディング手法はともに設計開発作業の前倒しであり、設計開発の作業を重ね合わせ同時並行的に進めていく点で同じである。しかしオーバーラップ型開発は本来、研究開発過程の本質を表しており、オーバーラップ型開発とフロントローディング手法は問題が事前に予測しうるのか否かの状況の違いはある。

竹田はフロントローディング手法を、オーバーラップを組織間のコミュニケーションによって進展させた手法 (竹田, 2000)と捉えている。竹田や藤本のフロントローディングの調査によれば、部門間・企業間のコミュニケーション・パターンの変化と三次元情報技術の発展と普及が、製品開発パフォーマンスの向上をもたらしていることが判っている。そして先端的な情報技術を持つことは必要条件であるが、欧米が同様の技術を導入している点も考慮すると、それは十分条件にはないとしている(藤本, 2003)。つまり日本企業は、アメリカ企業と比べ、技術者間の相互信頼関係や協調的・柔軟な態度・目標の共有化が進み(藤本 et al., 2002)、早期で統合的な問題解決サイクルをうまく回しているからこそ(Clark & Fujimoto, 1991)、先端的情報技術をうまく活用できているのである。

6.3.2 リーン思考

日本の自動車産業は製品の品質や世界市場のシェアを見ても世界トップレベルにある。その国際競争力の源泉が生産方式を切り口として長年議論されてきた。先のフロントローディングは、JIT（ジャスト・インタイム)やカンバンといったトヨタ生産方式による表層的仕組みの一手法である。これらトヨタ生産方式は根本的な思考を拠り所にしている(大野, 1978)。それは大野(1978)が示した生産アプローチと仕事の哲学であり、トヨタ生産方式を探求し体系化させたリーン思考(Womack et al., 1991)である(Poppendieck M. & Poppendieck T., 2004)。

リーン思考の本質は五つの基本原則として展開されている(Womack & Jones, 1997)。その原則は以下の通りである。

・それぞれの製品の価値を正確に定義づけること
・それぞれの製品の価値の小川を定義すること
・よどみない価値の小川を作り上げること
・顧客がメーカから価値をプルできる様にすること
・完全性を追求すること

リーン思考の出発点は価値である。リーン思考ではまず顧客が何に本当の価値を見出しているかを問うことから始める。次のステップで具体的な製品について価値を生み出す活動を行う一方、価値を生まない活動を排除する。そしてデザインや製品が顧客からのプルによって、スムーズかつ急速に広がる流れを作り出す。この流れとプルが実行されると、最後に完成度を高めるための改良のサイクルを加速させる。

上記の中で何も価値を生まず、資源をただ消費する活動を無駄と呼んでいる(大野, 1978; Womack & Jones, 1997)。リーン思考は人、設備、時間といったリソースをより少なく消費させながら、同時に顧客が要求しているものを提供する状態に近づけてくれる。そして無駄を価値に転換させる改善活動でもあるので、効率的な仕事を創造することでもある。

6.4　価値モデルの提案

6.4.1 製品価値と価値形成

リーン思考の出発点は価値である。製品価値はメーカが造るものであるが、価値は顧客によって評価されるものである(Browning et al., 2002)。価値は顧客との対話を通じて解釈し検討をしなければならない。この論点は顧客視点の価値の重要性を指摘するものである。実際、多くの企業は価値の測定を多属性価値の評価手法を使っている。

製品価値の関係式については、現在まで研究者によって幾多も提案されてきた。そこで価値の定義に関する議論を概観する。

価値工学では機能に着目して価値分析を行い、設計開発工程に遡った機能改善を行う。一般的な価値工学(Miles, 1981)によれば、V を価値(Value)、F を機能(Function)、C をコスト(Cost)とした以下の関係式が定義されている。

$$V = \frac{F}{C} \tag{6.1}$$

Johnasson et al. (1993) が提案する価値は以下の関係式で示される(Johnasson et al., 1993)。

$$V = \frac{Q \bullet S}{SP \bullet LT} \tag{6.2}$$

ここで V は価値(Value)、Q は品質(Quality)、S はサービス(Service)、SP は販売価格(Sales Price)、LT はリードタイム(Lead Time)である。

Slack (1999)は以下の関係式で表現している(Slack, 1999)。

$$V = \frac{N \bullet A \bullet f(t)}{C} \tag{6.3}$$

ここで N は製品のニーズ(Needs)の重要度である。N の値は顧客が決定する。A は顧客満足を満たす N の実現能力(Ability) である。A は設計開発工程がどの様に実行されるかで決定する値である。$f(t)$は N が市場投入される時期である。この $f(t)$は顧客の期待している納期と従属関係を持っている。C は N が持つ機能と開発工程に従属したコスト(Cost)である。

一方、Weinstein & Johnson(1999)の価値の定義は上記の絶対的な関係式とは違って、複数の製品を対比させた相対的な価値である(Weinstein & Johnson, 1999)。

$$V = \frac{PB}{PC} \tag{6.4}$$

ここで V は価値(Value)、PB はその製品から受けとる利益のイメージ(Perceived Benefits)であ

り、*PC* はそのコストのイメージ(Perceived Costs)である。

上記を整理集約すると、製品価値とは顧客の視点による絶対的尺度と相対的尺度の二面性で議論されていることが認められる。また上記の関係式は全て、コストが価値に影響を与える共通の因子である。そして製品が持つ属性と製品投入時期も影響を与える。つまり製品価値とは顧客の評価視点で製品の属性が供与する恩恵に比例し、コストと製品投入時期に反比例する。ここで *V* を価値、*P* を製品の属性(Properties)、*C* をコスト、*LT* をリードタイムとするなら以下が成立する。

$$V \propto \frac{P}{C \bullet LT} \tag{6.5}$$

式(6.5)の製品価値は製品自身がその時に結果としてもたらす価値という意味からスタティック（静的）な価値と捉えることができる。それではダイナミック（動的）な次元ではどうか。

$$V \propto \frac{\Delta P}{\Delta C \bullet \Delta LT} \tag{6.6}$$

式(6.6)は *P*、*C*、*LT* の動的変化が、価値の動的変化に比例することを示す。これは、価値および *P*、*C*、*LT* は必ずしも一定ではなく、また、価値はこれら因子の変動に大きく影響されることを表している。ダイナミックな価値として捉えることができる。すなわちこれは、製品の価値形成がなされる設計開発工程において、製品価値がダイナミックに変化することを表している。そしてこのことから、製品価値は設計開発工程の効率性に依存するのであるから、製品価値の最大化には、設計開発工程の効率化が重要と言える。これはリーン思考で言うよどみない価値の小川である。

式(6.5)と式(6.6)双方の考え方を集約するなら、価値は製品自身で形成されるスタティックな次元だけでなく、設計開発工程におけるダイナミックな次元によって形成されると言える。この二つの次元で製品の価値形成を表現すると図 6.4 となる。

一般的にメーカは図 6.4 における *P*、*C*、*LT* のバランス、そして *P* 内の具体的要素の重み付けまで洞察できない。その為に、それらを顧客に対してコンジョイント分析や因子分析といった経験的データに基づく多属性価値の評価手法を利用する。しかしそれだけでは不十分と言える。それはそこに存在する価値を市場の需要だけに視点を当てて検討しているからである。つまり設計開発工程というダイナミックな次元で顧客の価値は変動することと、同次元でメーカが造る価値の形成過程が考慮されていないのである。

次節で提案する価値のダイナミック・モデルは、製品開発工程における製品価値の形成過程を、メーカの視点だけでなく顧客の視点も考慮して、ダイナミックに表現したものである。これはリーン思考に従えば、顧客からのプルによって価値の流れを作り出すということである。

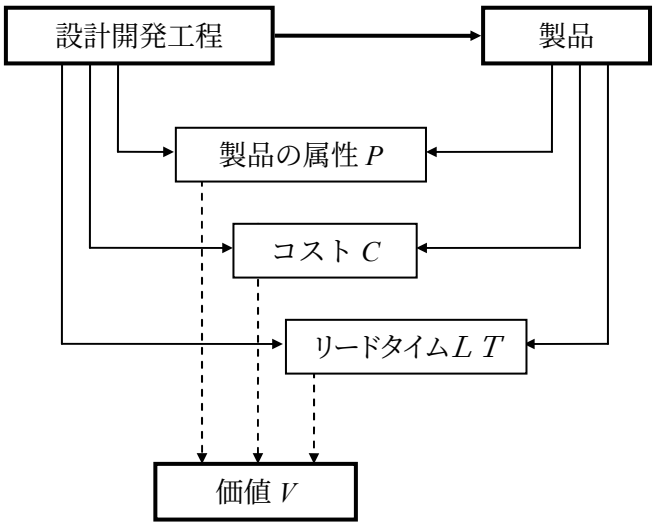


図 6.4　製品開発における価値形成

6.4.2　価値のダイナミック・モデル

図6.5に価値のダイナミック・モデルを示す。価値を縦軸、プロセス量（所要時間・所要コスト）を横軸とり、メーカ視点と顧客視点、更にそれらを統合した視点の効用曲線を描く。

メーカ視点価値曲線 PV_1では、プロセス量が増加するに従って価値は上昇し、顧客が定義した製品性能と言うべき製品の属性 P の目標価値に近づく。ただしプロセス量の追加が進むほど、価値実現に対する効用は除々に薄れるのであるから価値の限界効用は逓減する。

顧客視点価値曲線 PV では、プロセス量が増加しても、ある限界時点までは、目標価値を維持している。しかしある時点を超えると減少方向に進む。これは式(6.5)の考え方である。ただ顧客もある限界時点までは忍耐強く待つことができよう。その為その限界時点までの価値の限界効用の速度は速くはない。しかしその限界時点を越えると速度は速くなり、急激にその効用が減少する。

上記の二つの視点を組み合わせることで一つの価値モデルを構築できる。つまり顧客視点価値曲線 PV のプロセス量に応じた価値の減少分をメーカ視点価値曲線 PV_1に反映したものである。統合価値曲線 PV_1はプロセス量の増加とともに、顧客視点価値曲線 PV が目標価値を維持している間はメーカ視点価値曲線 PV_1と等しく上昇する。しかし、顧客視点価値曲線 PV が減少を始めると、それに従い上昇率は低下し、極大点を経て減少に向かう。統合価値曲線 PV_1の極大点に対応するプロセス量 P_1と価値 V_1が顧客の嗜好とメーカの負荷を考慮した決定ポイントと考えることができる。つまり、これ以上プロセス量をかけても、製品

の性能と価値の効用が上昇しない上限と判断できる。

更にここで効率化された設計開発工程モデルがどういった効用曲線になるのか検討する。まず効率化された設計開発工程モデルで得られる成果物の価値とその従前モデルで得られる成果物の価値を対比してみる。もし双方とも同じプロセス量が与えられているなら、効率化された設計開発工程モデルで得られる成果物の価値の方が常に上回るはずである。つまり効率化前の設計開発工程モデルをメーカ視点価値曲線 PV_1 とするなら、効率化されたモデルはメーカ視点価値曲線 PV_2 となる。先と同様に統合価値曲線 PV_2 および、その極大点からプロセス P_2 と価値 V_2 が得られる。同じプロセス量が与えられるなら、価値 V_2 は価値 V_1 より常に上回る。これら決定ポイントを比較することで設計開発工程の効率化による価値の上昇を定量的に対比することができる。このことは製品開発における意思決定にとって重要である。

本モデルの利点をまとめると次のようになる。

1）製品の性能と価値の効用における上限を判断できる。

2）設計開発工程の効率化による価値の上昇を定量的に対比することができる。

3）本モデルの運用によって設計開発工程における問題を解析できる。

3）は次項で詳述するが、以上の理由から本モデルは設計開発工程の効率化を検討するのに有用である。

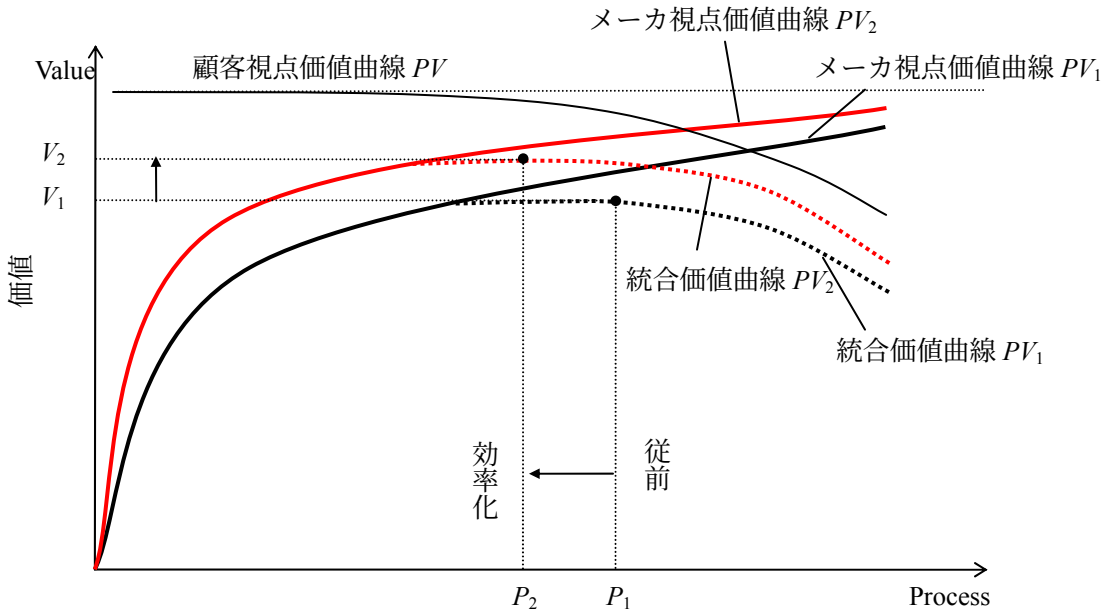


図 6.5　価値のダイナミック・モデルと設計開発工程の改善

6.4.3　価値のダイナミック・モデルの運用

価値のダイナミック・モデルの運用について議論する。設計開発工程における無駄な工程の問題と過剰機能・過剰品質の問題を挙げ、それぞれを本モデルによって解析し、改善するべき工程のありかの検討を行う。

1)　無駄な工程の問題

図 6.6 のメーカ視点価値曲線の P_3－P_4 間に示すように、プロセス量を増加させても効用曲線がフラットな箇所が無駄な工程であり、改善の対象である。

しかしフラットな箇所全てが製品開発における無駄とは限らない。新製品の設計開発の多くの事例を観察してきた von Hippel (1994)はイノベーションに関連した問題解決に向けてタスク間で知識情報のイタレーションが観測されると指摘している(von Hippel, 1994)。つまりフラットな箇所がイノベーション創出過程と考えられる期待である。その他に、工程内のタスク間のイタレーションの有効性は学習効果の場合もある。更に製品開発というプ

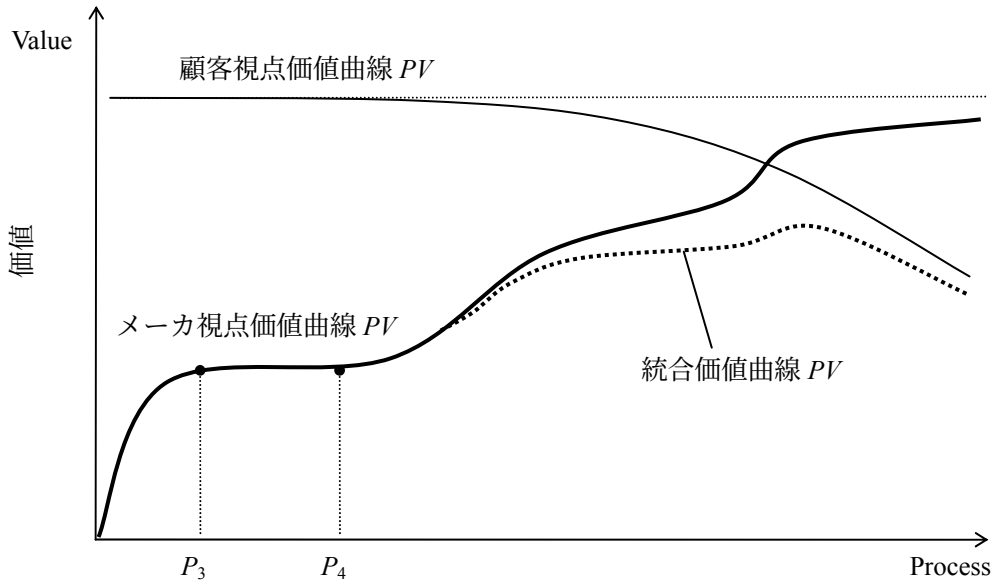


図 6.6　無駄な工程の問題

ロジェクトのリスクを低減させる活動がフラットな箇所を生み出している可能性もある。こういった必ずしも無駄とは言えない工程は奨励すべきであるが、全く無駄な工程は排除すべきと言える。

全く無駄な工程に対する改善手法としては、例えば代替となる設計開発工程のアーキテクチャを創造すること、ある程度の投資を行って効率化が期待できる手法に切り替えること、費用対効果が期待できる道具を利用すること等が考えられる。そこには多様な代替案が存在する。それら代替案を予見的なモデルとして価値モデルに反映し検討できるのであれば、メーカの意思決定に利用できる。意思決定の後も途中からその修正変更案をモデルに逐次反映して価値形成をモニタすることも可能である。それら実施内容を現場のスタッフと本モデルを通じて共有できるのであれば、組織内で意思疎通の効果も期待できる。

2)　過剰機能・過剰品質の問題

過剰機能・過剰品質とは顧客の要求しない製品の機能が付きすぎている、あるいは必要な機能に対して構造設計･材料設計的に過剰となっている状況である。図6.7に示した通り、メーカ視点価値曲線が顧客視点価値曲線 *PV* を超えた時に過剰機能・過剰品質が始まる。プロセス量が P_5の時、プロセス量によって減少した顧客視点の価値とメーカ視点の価値が一

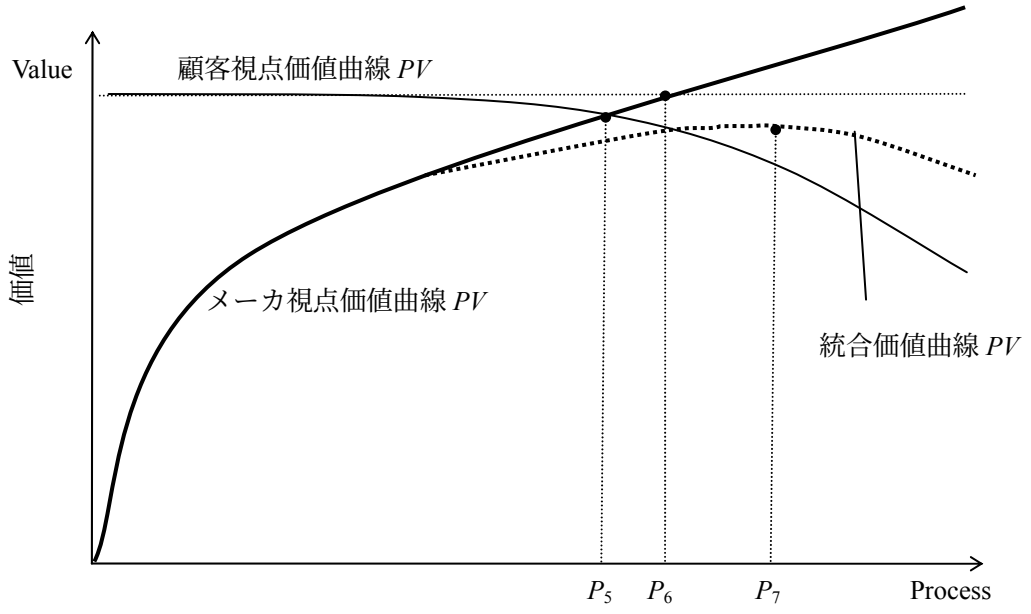


図 6.7　過剰機能・過剰品質の問題

致する。しかし顧客視点の価値がプロセス量に対して減少しない、あるいは考慮していなかったと仮定していたなら、メーカは P_6の時を過剰設計が始まるポイントとしてしまうであろう。つまりその仮定では、$\Delta(P_6-P_5)$の経済的損失量を招いていると言える。一方、プロセス量が P_7の時、統合価値曲線は極大値を得る。しかしその時は一層、顧客視点の価値を上回っており過剰設計が更に進行したポイントである。この場合には$\Delta(P_7-P_5)$が経済的損失量であり、P_6のそれより影響が大きい。つまり図6.7の様な過剰設計の傾向が強い状況下ではP_5が判断のポイントとなる。

現在の所、過剰機能をもたらす過剰設計と過剰品質の影響について定量的に測定することは難しい(藤本, 2003)との指摘がある。しかし本モデルを運用することで過剰機能・過剰品質による影響度合いを知ることができ、改善に向けた一指標となりうる。

6.4.4 価値モデルにおける課題

価値のモデルの適用にあたり議論されるべき課題は、その価値の客観性にある。顧客の視点だけから見ると最適な価値とは、理想的商品を、必要な時に、無償で提供されることである。しかしこういった限定した視点の議論ではメーカは存続できない。本来、顧客とメーカだけでなく、それを取り巻く従業員、納入業者、国、社会もバランス良く反映された価値のモデル提案が求められているとも考えられる。その様な最適なバランスのとれた価値のモデルの構築について検討が必要である。

6.5　本章のまとめ

製品開発を行っている企業にとっては、設計開発工程の効率化は競争戦略上重要である。しかし時間短縮とコスト低減は二律背反問題である。本章では設計開発工程において納期を短縮させようとするとコスト負荷が発生するメカニズムとトレードオフ検討の先行研究について議論した。そしてトレードオフ検討には限界があり、設計開発工程の効率化のためには、所要時間と所要コスト双方の低減を実現できるモデルの創造が必要であることを示した。設計開発工程の効率化にはプロセス・イノベーションの創造、すなわち設計開発工程というアーキテクチャのイノベーションが要求されるということである。

日本の自動車メーカはリーン思考によって結果として所要時間とコストを同時に削減させた。これは設計開発工程アーキテクチャの創造の例証である。リーン思考とはトヨタ生産方式の原点として知られている原則である。リーン思考ではまず製品の価値と価値の小川を定義することで設計開発工程上の無駄を明らかにする。次にこの無駄を排除し価値を最大化させる活動を行う。この価値の最大化のねらいが結果として所要時間と所要コスト双方を同時に効率化させる。

本章ではリーン思考を拠り所とした価値のダイナミック・モデルを提案した。リーン思考の出発点は価値である。価値のダイナミック・モデルは製品価値の形成過程を、メーカ

の視点だけでなく顧客の視点も考慮して、ダイナミックに表現したものである。これを運用することで、改善すべき無駄を含む工程や過剰機能・過剰品質の問題を定量的に解析できる。すなわち価値のダイナミック・モデルは所要時間と所要コスト、そして製品価値の三つの指標から設計開発工程の実践的洞察を可能にするものであり、設計開発工程の効率化に向けた指針を与えるものである。

参考文献

[1] Ahmadi, Reza, Roemer, T. A. and Wang, R. H.: “Structuring product development process”, European Journal of Operational Research, Vol. 130, pp. 539-558 (2001)

[2] 青島矢一: “製品アーキテクチャと製品開発知識の伝承”,ビジネスレビュー, Vol. 46, No. 1, pp. 46-60 (1998)

[3] Blackburn, J. D.: “Time-Based Competition: The Next Battleground in American Manufacturing”, Business One Irwin, Homewood, IL (1991)

[4] Boehm, B. W.: “Software Engineering Economics”, Prentice Hall, Englewood Cliffs, NJ (1981)

[5] Brooks, F. M.: “The Mythical Man-month: Essays on Software Engineering”, Addison-Wesley, Reading, MA (1975)

[6] Browning, T. R., Deyst, J. J., Eppinger, S. D. and Whitney, D. E.: “Adding Value in Product Development by Creating Information and Reducing Risk”, IEEE Transactions on Engineering Management, Vol. 49, No. 4, pp. 443-458 (2002)

[7] Clark, K. B. and Fujimoto, T.: “Overlapping problem solving in product development”, In K. Ferdows Ed., Managing International Manufacturing, North-Holland, Amsterdam (1989)

[8] Clark, Kim B. and Baldwin, Carliss Y. （安藤晴彦訳）: “デザイン・ルール―モジュール化パワー”,東洋経済新報社, pp. 75-109 (2004)

[9] Clark, K. B., and Fujimoto, T.: “Product Development Performance”, Harvard Business School Press (1991) (藤本隆宏, K. クラーク（田村明比古訳）: “製品開発力”,ダイヤモンド社 (1993))

[10] Eastman, R. M.: “Engineering information release prior to final design freeze”, IEEE Transactions on Engineering Management, Vol. 27, No. 2, pp. 37-42 (1980)

[11] 藤本隆宏: “自動車製品開発の新展開　フロントローディングによる能力構築競争”,ビジネスレビュー, Vol. 46, No. 1, pp. 22-45 (1998)

[12] 藤本隆宏,延岡健太郎,青島矢一,竹田陽子,呉在烜（奥野正寛,竹村彰通,新宅純二郎編）: “情報化と企業組織、アーキテクチャと組織能力の視点から「電子社会と市場経済」”,新世社, pp. 97-134 (2002)

[13] 藤本隆宏: “能力構築競争－日本の自動車産業はなぜ強いのか”,中公新書, pp. 334-340 (2003)

[14] 藤本隆宏,延岡健太郎: “製品開発の組織能力:日本自動車企業の国際協力”, RIETI Discussion Paper Series 04-J-039,経済産業研究所, p. 14 (2004)

[15] Graves, S. B.: “The time-cost tradeoff in research and development”, a review, Engineering Costs and Production Economics, Vol. 16, pp. 1-9 (1989)

[16] von Hippel, E.: “Sticky Information and the Locus of Problems Solving: Implication for Innovation”, Management Science, Vol. 40, No. 4, pp. 429-439 (1994)

[17] Johansson, H. J., McHugh, P., Pendlebury, A. J. and Wheeler, I. W. A.: “Business Process reengineering: Break point strategies for market dominance”, Wiley, New York (1993)

[18] Mansfield, E.: “Industrial Research and Technological Innovation: An Econometric Analysis”, Norton, New York (1971)

[19] Miles, L. D.(玉井正寿監訳): “VA/VE システムと技法:価値分析の進め方第2版”,日刊工業新聞社 (1981)

[20] 大野耐一: “トヨタ生産方式－脱規模の経営をめざして”,ダイヤモンド社 (1978)

[21] Poppendieck, M. and Poppendieck, T.: “Lean Software Development”, Addison-Wesley Professional, 1st edition (2003)(メアリー・ポッペンディーク,トム・ポッペンディーク（平鍋健児,高嶋優子,佐野建樹訳）: “リーンソフトウエア開発～アジャイル開発を実践する22の方法～”,日経 BP 社 (2004))

[22] Putnum, L. H. and Fitzsimmons, A.: “Estimating software costs. Datamation”, October, pp. 171-177 (1979)

[23] Roemer, T. A., Ahmadi, R. and Wang, R. H.: “Time-cost trade-offs in overlapped product development”, Operations Research, Vol. 48, No. 6, pp. 858-865 (2000)

[24] Scherer, F. M.: “Time-cost tradeoffs in uncertain empirical research projects”, Nav. Res. Logistic Q., Vol. 13, No. 1, pp. 71-82 (1966)

[25] Schumpeter, J. A.（塩野谷祐一,東畑精一,中山伊知郎訳）: “経済発展の理論(上)”,岩波書店, p.150 (1980)

[26] Slack, R. A.: “The lean value principle in military aerospace product development”, MIT Lean Aerospace Initiative, Report RP99-01-16, Massachusetts Institute of Technology, Cambridge, MA, July (1999)

[27] 竹田陽子: “プロダクト・リアライゼーション戦略 －３次元情報技術が製品開発組織に与える影響”,白桃書房, pp. 25-38 (2000)

[28] Thomke, S. and Fujimoto, T.: “The Effect of ‘Front-Loading’ Problem Solving on Product development Performance”, Journal of Product Innovation Management, Vol.17, pp. 128-142 (2000)

[29] Ulrich, Karl T.: “The Role of Product Architecture in the Manufacturing Firm”, Research Policy, Vol. 24, pp. 419-440 (1995)

[30] Weinstein, A. and Johnson, W. C.: “Designing and delivering superior customer value: Concepts,

cases, and applications", St. Lucie Press, Boca Raton, FL (1999)

[31] Womack, J. P., Roos, D. and Jones, D. T.: "The Machine that Changed the World", Rawson Associates, New York (1991)(J.・ウォマック,D.・ルース,D.・ジョーンズ（沢田博訳): "リーン生産方式が、世界の自動車産業をこう変える",経済界(1991))

[32] Womack, J. P. and Jones, D. T.（稲垣公夫訳）: "ムダなし企業への挑戦",日経 BP 社 (1997)

[33] 山本靖: "製品設計開発工程におけるスケジュールとコストのリスク・モデル開発に関する研究、半導体設計工程事例によるモデル適用と考察",平成16年度秋季研究大会予稿集, 日本経営工学会, pp. 244-247 (2004)

第7章　タイム・マネジメントによる設計開発期間の短縮

7.1　タイム・マネジメント

製品開発から市場投入までの期間を短縮化することが競争優位の点で重要になっている(Reinertsen, 1983; Smith, 1988)。特に製品のライフ・サイクルが2年以下の業界では、その傾向が著しい(Stalk, 1988; Stalk & Hout, 1990, Williams, 1992)。この様に競争の激しい製品市場において、製品開発における時間の管理、すなわちタイム・マネジメント(野口, 1995)は企業の競争力の面で欠かせない領域の一つである。

製品開発の所要期間を圧縮させる企業戦略は、i)製品開発の早期に顧客ニーズを特定すること(Hauser & Clausing, 1988; Griffin & Hauser, 1993)、ii)顧客ニーズを早急に製品設計として翻訳させる能力を向上させること(Cohen & Levinthal, 1990)、iii)製品の改良と拡充を継続的に対応し奨励すること(Hayes et al., 1988; Hamel & Prahalad, 1991)、iv)組織能力を向上させ、フロント・ローディングによる作業の前倒し(Clark & Fujimoto, 1991; 藤本, 2003)等がある。しかし本章で議論するタイム・マネジメントは、これらの企業戦略が企業の個人・組織の能力向上または意識改革を前提としている点で異なる。つまり工学的な視点に立ち、製品開発の所要期間の削減に向けてプロジェクトを管理運営しようとするのが本章のねらいである。

本章で提唱するタイム・マネジメントは、三つの要素から成り立つ。一つ目はスケジュール管理である。プロジェクト・マネジメントで一般的に利用されている管理手法である。事前に予定をどの様に割り振るのか、実際にどの様に仕事を進めるのかといった計画・実施・修正といったサイクルである。ガント・チャート(Gantt, 1910)、PERT (Program Evaluation and Review Technique)、クリティカル・パス手法(Critical Path Method:CPM)といった伝統的なツールを用いる。二つ目は、時間の節約、すなわち無駄な時間をなくすことである。これはクリティカル・チェーン管理(Critical Chain Management:CCM)を利用する。Goldratt の制約理論(Theory of Constraint:以下 TOC)を拡張した管理手法である。人間行動の特性問題を考慮

し、バッファという概念を用いて無駄な時間をなくすことで、工期を短縮させることができる。三つ目の要素は、時間の増大である。これは他人の時間を分けてもらうこと、又は他人の専門知識の能力を分けてもらうことである。つまり capacity 又は capability の提供である。これらにより限られたリソース問題を解放させ、所要期間を削減させる。本章では capacity 又は capability の提供、すなわちアウトソース化の適用を提唱している。

本章ではこれら三つの手法を統合させたアプローチを提案し、この統合アプローチがタイム・マネジメントにおいて有効であるかの検証を行う。検証にあたっては半導体設計開発の実プロジェクトに適用し、ベンチマーク比較を行う。

7.2　タイム・マネジメントに関する先行研究

7.2.1 製品の市場投入時期の重要性

エレクトロニクス製品のライフ・サイクルは短縮化しており、製品開発から市場投入までの期間を短縮化することが重要になっている。競争の激しい製品市場において、タイム・マネジメントによるスケジュール管理は企業の競争力維持に欠かせない領域の一つである。他企業に先行して早く製品投入できるなら競争優位に立ちことができるからである。

時間の概念は企業にとって製品開発力につながる重要な管理要素となっている。実際、次の10年間において国際的競争力を持つメーカは、製品開発力に依存する(Seering et al., 1996)と言われている。製品開発力とは製品の品質、性能、価格の他に、市場投入の時期である。製品開発者は製品の品質や性能、製品価格を犠牲にせずに、適切な時期に製品を市場に投入できるのであれば、四つのメリットを企業は享受できる(Cook, 1998)。一つ目としては、市場投入時期に遅れた場合と比較して、製品の販売期間を延長できることである。製品の投入時期を一日早めることができるなら、販売期間が一日延長できる。二つ目としては、開発サイクルを短縮化することができ、貴重な設計開発時間を削減できることである。つまり開発を終了させた設計技術者は他の設計開発に従事することが可能になる。三つ目としては、製品を最初に市場投入した企業は、一時的に独占的支配力を持つことができ、そして高い限界利益率を確保できることである。四つ目のメリットは市場の占有上、優位に立てることである。その製品が市場で真に最初のものであるなら、市場占有率100%を得たことになる。これは後発企業が、時間と労力をかけて100%の市場占有率を得るより容易なことである。

グローバルに展開し、激動するダイナミックな競争市場では、新製品とその開発過程が一層競争の焦点となっている(Clark & Wheelright, 1993)。製品の市場投入を早く効率的に実行できる企業が、顧客期待とニーズを製品に反映させることができるなら、一層強い競争力を作り上げることができる。スピーディーな市場投入は顧客反応を敏感に捉え、市場調査を最新の状況にしておくことができる。そして、製品がより早く紹介されるほど、市場を占有する確率は高い。この状況は標準化が働く新興市場ほど、その命題は真と言える

(Huthwaite, 1994)。自動車業界では、競合に先駆けて4ヶ月から5ヵ月早く市場投入すれば、それが市場占有率に影響を与え、そしてそれが自動車企業の財務状況にも影響を与えるとされ、重要な戦略となっている。例えば、10,000ドルで売られる車の場合、一日の市場投入の遅れが約100万ドル相当の損失を与えるとの研究報告がある(Clark, 1989)。図7.1にTime-to-market の代表的モデルを示す。3ヶ月分市場投入を早めることで、追加の売上げ期待ができる。

製品開発は、製品性能、製品原価、設計開発費用、開発スピードという四つの要素のトレードオフ選択である(Smith, 1988)。Smith (1988)はこれら四つの要素の変数を変えることで、これらが製品サイクルにおける収益に与える影響をシミュレーションしている。そのシミュレーション結果を図7.2に示す。二つの異なる製品について、企業のキャッシュ・フロー収益をベースに算出されたものである。一つは伝統的な機械系の特定部品であり、他方はライフ・サイクルが短い特定のエレクトロニクス部品である。どちらも収益に影響を与える要素は製品投入時期と製品性能であった。それに比べて製品原価は双方とも唯一の決定的要因とは言えない。そして設計開発費を含む研究開発費用が仮に50%予算を上回ったとしても、収益には直接大きな影響を与えないとのことである。つまりこの実証例は一般論として昇華できていないとしつつも、製品投入時期と製品性能という要素は、他の要素を多少犠牲にしたとしても製品開発の優先事項とすべきであるというのが Smith の主張である。

Reinertsen(1983)も市場投入までの期間が収益上最も重要な因子であると指摘しており、製

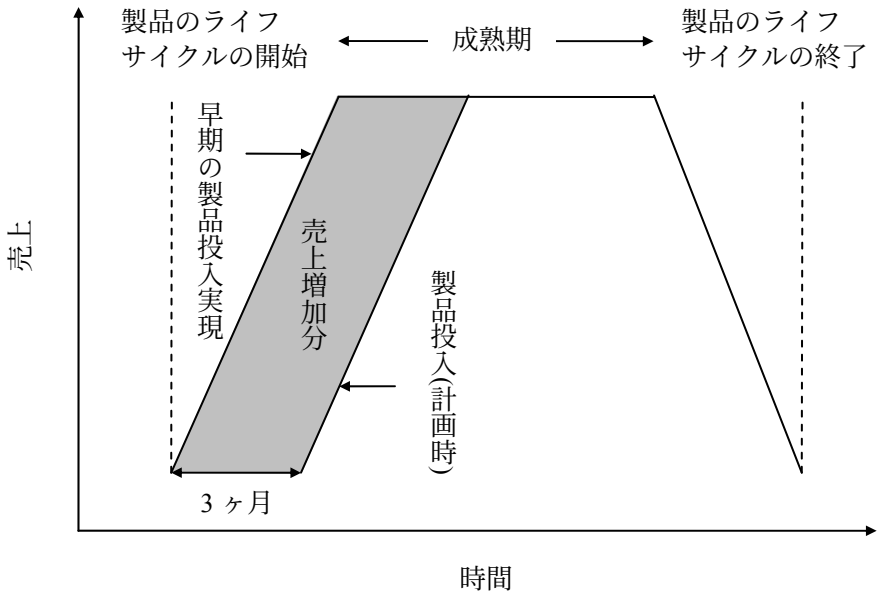


図 7.1　製品投入の前倒しと売上増加(Cook, 1998)

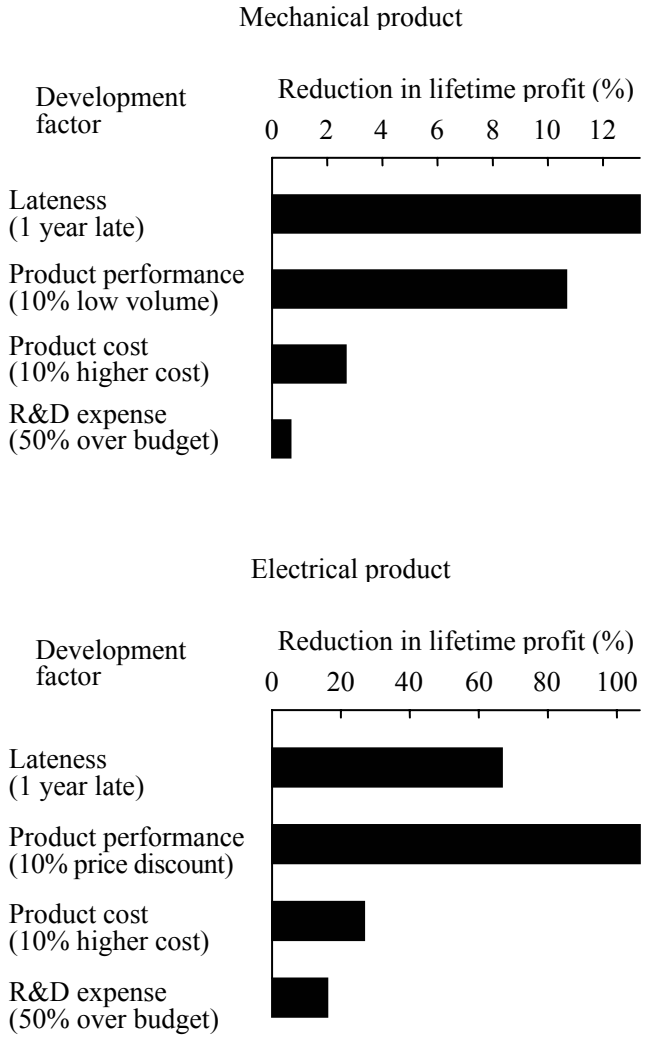


図 7.2　製品開発における優先事項と収益(Smith, 1988)

品開発におけるタイム・マネジメントは無視できないというのが一般的論調である。

7.2.2 CCM の事例研究

クリティカル・チェーンの管理(Critical Chain Management:以下 CCM と称す)手法の適用は、軍事カメラ(Cook, 1998)、船舶(Fujimoto et al., 2003)、軍事用フライト・シミュレータ(Pitts, 2000)、米国の建設業事業(Yeo & Ning, 2002)、ソフトウエア(建部 et al., 2004)などの分野における事例研究で見られる。そしてこれらの事例によれば、CCM はプロジェクトの遅延を削減させるのに効果があるとされている。建部ら(2004)によれば、ソフトウエア開発において、

CCM はプロジェクトの工期を短縮させるだけでなく、スケジュール・リスクを低減させ、作業の実践者の志気と効率性を高める潜在能力を持ち合わせていることも検証されている。

しかし現在まで、半導体設計開発の領域では CCM の適用事例の積み重ねが行われていない。本章で提案する統合アプローチとは、スケジュール管理、CCM そしてアウトソース化の統合したものである。7.6節ではこの統合アプローチを半導体設計開発プロジェクトに適用して、工期短縮を検証する。従ってこの検証は、半導体設計開発における CCM の事例研究として意義あるものと言える。

7.3　TOC とクリティカル・チェーン

7.3.1　TOC の出発点

TOC (Theory of Constraint: 制約理論)はシステム・レベルの改善手法である。換言すれば、システムの目的を達成するための改善手法とも言える。システムの性能をシステムの出力と見るなら、いかなるシステムにも出力を制限させている制約が存在しうる(Goldratt, 1986; 稲垣, 1997; 小林, 2000)。この思考が TOC の出発点である。システムの性能を解釈する上では一般的な考え方である。

システムの制約を結びつけるのに、鎖のアナロジーが使われる。これは鎖の強度が最も弱い輪の強度と等しいごとく、システムの出力もその制約箇所に支配されるというものである。最も弱い輪以外の輪の強度をいくら高めても、鎖の強度を高めることには貢献しない(Leach, 2000)。つまり全体の強度の改善には、最も弱い輪の強度を高めることなのである。この考え方を基に、Goldratt は生産性改善の為の五つのステップを掲げている(Goldratt, 1986; 稲垣, 1997)。ステップの流れを図7.3に示す。

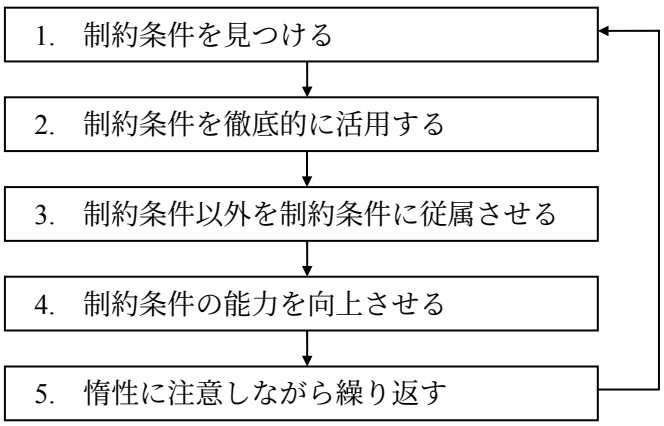


図 7.3　TOC の改善の 5 ステップ (稲垣, 1997)

TOC はシステムの対象範囲を生産現場だけでなく、サプライチェーン全体、そして工場全体、企業全体、組織や思考法にも拡張できる。CCM は一つのプロジェクトをシステムに見立て、管理論として適用した手法である。

7.3.2　プロジェクトにおける不確実性と人間行動特性による遅延

Goldratt (1997)はプロジェクトの計画策定とスケジューリングにおいて三種類の不確実性を考慮している。その三つとはタスクの時間の不確実性、パス経路の時間の不確実性、そしてリソースの不確実性である。これらの不確実性の存在が作業項目の見積もり精度の問題につながっている。実際、多くのプロジェクト・マネージャはそれらの不確実性がスケジュール・リスクを招来することを認識している。その為にスケジュールを策定する際に安全性(Safety)を念頭に置き、タスクの遅延対策としてタスクに余裕時間を予め準備しておく。

それら不確実性に対しては一般的に確率論的アプローチがとられている(Vose, 2003)。図7.4はタスクにおけるスケジュールの不確実性を確率密度関数で表現したものである。不確実性が高いほどこの確率密度関数の歪度は0より大きくなり、右の尾の部分は長くなる。t_g はプロジェクトの目標値(Goal)であり、t_m は見積もりから算出した平均値(mean)である。期日まで遂行できない可能性が高い平均値を選択し、わざわざ危険を冒す実践者は存在しない。従って平均値に安全余裕時間を加味することで目標値としている。つまり目標値とはプロジェクトにおける安全性が組み込まれている。この目標値の実現確率は80%から95%が

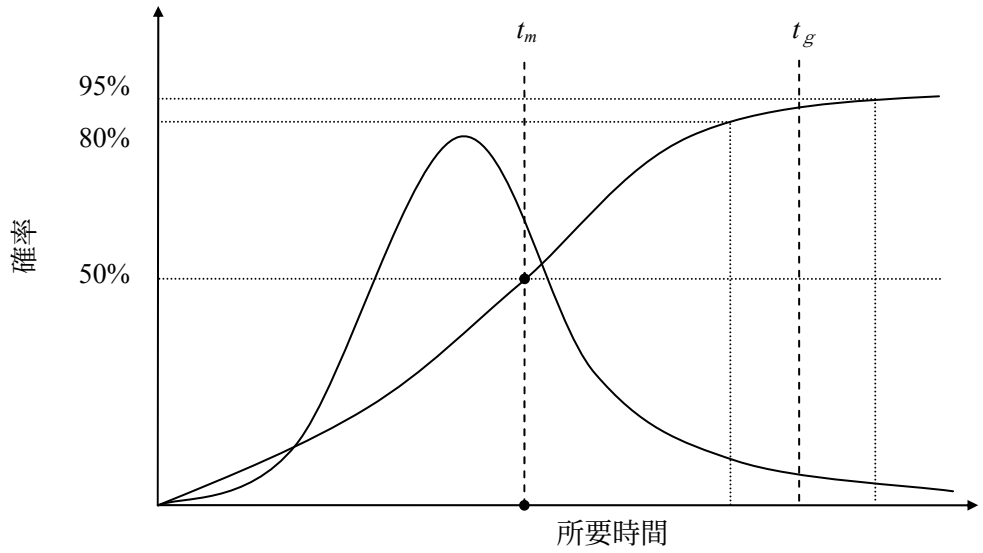


図 7.4　スケジュールの不確実性

一般的である(Goldratt, 1997; Vose, 2003; Steyn, 2000, 2003; Cook, 1998; 稲垣, 1997; Leach, 2001; Newbold, 1998)。平均値における実現確率は50%である。ここで $\Delta(t_g - t_m)$を安全余裕時間と考える。CCM ではこの安全余裕時間を大胆に削除する。その理由は後述の人間行動の特性を排除するためである。

プロジェクトの現場の実践者は目標値の t_g を納期の期限とし、不確実性が組み込まれていると解釈している。しかし実際の所、全てのタスクは目標値以内に遂行できるとは限らない。Goldratt は納期に注意を払いすぎることが無駄な時間を消耗させ、遅延を起こす原因であると指摘している。そしてこの根拠を人間行動の特性に関連づけし、四つの因子として上げている。この四つの因子について述べる。

1) 学生症候群

目標値 t_g が決定される時には、管理上、安全な余裕時間が付け加えられる。従って、タスクを実行する者は自分なりに所要期間を再評価し､最尤値として t_m の平均値を採択する。すると結果としてそのタスクとは別のタスクに手をつけてしまう。これは安全な余裕時間を含んでいると解釈しているからである。そしてぎりぎりになるまで着手せず、最終的には一夜漬けの集中作業となる。その時にはマーフィーの法則(Bloch, 1993)によるところの「予期しない問題」に打ちのめされ、納期が遅延する。たとえ突貫作業で間に合わせたとしても、逆にタスクの品質又は成果物が悪くなってしまう可能性が残る。これが学生症候群である。

逆に言うならば、納期があるから学生症候群が起こるとも言える。このことから、クリティカル・チェーンでは納期ではなく、所要期間を重視する。もちろんプロジェクト全体では納期は重要であるが、作業事に細かく納期を設定することはしない。この思考は作業が終了次第、次の作業に着手することを促す。するとぎりぎりまで着手しないことや一夜漬け傾向を防ぐことができる。

2) 自己の予防線

タスクが予定より早めに終了した場合にはどうなるか。この場合にはその事実がプロジェクト・マネージャに申告されにくい。すなわち自己防衛が働き、自己に対しての予防線を張る傾向になる(津曲, 2004)。なぜならば正直に申告すると次回のプロジェクトで作業者は不利になると考えるからである。つまり作業が早く終了して浮いた時間は無駄に消費されるだけで、プロジェクト全体に反映されない。

3) マルチタスキング

プロジェクトでは掛け持ち作業が頻繁に見られる。しかしマルチタスクとは単なる掛け持ち作業ではない。マルチタスクとは作業の実践者に対して複数の作業を同時進行させる

ことである(Fujimoto, Goldratt,1997)。この場合、それら掛け持ち作業の中での優先権が明確にされていないことが多い。結果として一作業当たりの実践者の生産性は低下する。これが納期遅延を招来する。

マルチタスクは納期遅延だけでなく、創造的生産が要求される作業の能力も低下させるという指摘(Fujimoto et al., 2003)もある。しかし特定の専門知識やノウハウを要求するリソースが限られている場合には、このリソース限定問題を解消できるアプローチであり、有効な手段と考えられている。実際、複数の組織と複数のプロジェクトが交錯した実践的マトリクス組織(芝尾, 1999)へ適用した場合にはマルチタスキングは必要条件となっている。

マルチアスクのもう一つの問題は、掛け持ちのタスクが同時に生じる場合である。これをリソースの競合と言う。この問題の詳細と解決については7.3.4項で述べる。

4) パーキンソンの法則

作業者がいつまでも作業を続け、計画した作業期間が満了しても尚延長されることがある。すなわち計画予算は全て使い尽くすまで消費される傾向がある。これをパーキンソンの法則と言う(Parkinson, 1996)。パーキンソンは、作業の量は一定ではなく、いかようにも変動する、従って業務効率化などということがいかに空疎であるかを述べている。つまり、この法則に基づき組織成長が拡大する。そして組織が拡大すれば、組織内での調整作業が幾何級数的に増大するということである。

必要不可欠な作業は計画予算を全て使い尽くすか否かによらず実行されなければならない。むしろ不必要な仕事はいくらでも作り出せるという組織内力学が、組織の拡大を続けていることを示唆している。

7.3.3 CCM

これまで長らく、プロジェクトの計画・管理には、PERT (Program Evaluation and Review Technique)や CPM (Critical Path Method)が利用されてきた。これらの手法で行われるように、クリティカル・パス上にある工程を重点的に管理することによりプロジェクト全体の納期を制御する方法は効果的である。しかし CPM には固有の問題がある。その問題点とは（i）作業項目の所要期間の見積もり精度の不足、（ii）総所要期間の最適化の検討不足、（iii）競合リソースの考慮不足といったことである(建部 et.al,2004)。これら問題点が指摘されていながら、それらが解決されずに現在まで利用されてきた。

それらの問題点を解決する手法として TOC に基づく CCM が提唱された。TOC は7.3節で述べた通り、プロジェクトの制約条件を早期に解決する手法である。そして全体工程で制約を発生させない様に管理する。つまり、この手法は部分最適の積み上げと言うより全体最適を考えることである。それが最終的に短納期の開発を実現させる。また、CCM はプロジェクトにおける価値連鎖内の生起する不確実性事象とそれに関連したリスクを管理する

拡張型手法と言える(Yeo&Ning, 2002)。これは CPM が決定論的な手法に立っており、CPM の限界点の一つを示唆するものである。

7.3.4　遅延の伝搬と総所要期間の最適化

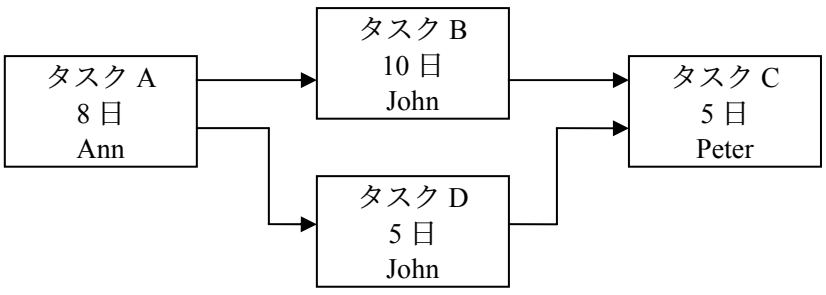
プロジェクトでは幾つかの作業が同時に進んでいて、それらが全て終わってはじめて次の段階の作業に進む場合がある。この場合、一つのタスクの遅延はプロジェクトのパス経路を通じて伝搬されてしまう。つまりこれはクリティカル・パス(PMBOK, 2000)と考えていなかったパス経路がクリティカル・パスになる可能性を残すものである。従ってクリティカル・パスの経路は状況に応じて変化する可能性があり、総所要期間の最適化が検討されにくい。CCM ではクリティカル・パスの決定の前に、タスクの配列の修正とバッファという概念を用いて解決している。

7.3.5　リソース競合とクリティカル・チェーン

限られたリソースを考慮したクリティカル・チェーンの概念は、Pittman (1995)によって初めて提唱された。そして Goldratt が TOC の概念と関連づけした。Goldratt はそこでプロジェクトの所要期間を左右する競合リソースという存在事実を確認し、競合リソースを回避するためのクリティカル・パスの配列や順列を多様に検討した。

それまでクリティカル・パスは先行タスクとの接続関係だけで決定され、競合リソースは配慮されてこなかった。つまり競合リソースの問題はクリティカル・パスが決定づけされてから検討されてきたのである。一方、クリティカル・チェーンは先行タスクとの関係性と、リソースの状況が考慮されたものである。つまり競合リソースは計画段階で解消されている。そして後述するバッファをプロジェクトの最後部におくことで、全てのタスクの遅延問題が一箇所にまとめるられるというものである。

Goldratt (1997)、Rand (2000)、Barber et al. (1999)、Leach (1999)、 Newbold (1998)、Steyn (2000)はクリティカル・チェーンを例証している。クリティカル・チェーンの単純例を図7.5に示す。



クリティカル・パス:ABC(23 日)

クリティカル・チェーン:ABDC 又は ADBC(28 日)

図 7.5　クリティカル・パスとクリティカル・チェーン(Steyn, 2000)

7.4　クリティカル・チェーンの適用手順

7.4.1 適用にあたって

本節では図7.6の PERT モデル(Cook, 1998; Yeo & Ning, 2002)を利用して、プロジェクトへのクリティカル・チェーンの適用手順を示す。この PERT モデルでは二つのタスク・スケジュールの枝が一つの枝に合流するが、実際のプロジェクトでは枝の数や合流点の数は多く、複雑に接続している。この PERT モデルは基本構成であり、この基本構成の論理的な組合せで実際の PERT は構築される(Goldratt, 1997; Steyn, 2000 and 2002)。

図 7.6(a)の PERT モデルは、クリティカル・チェーンを適用する前の作業工程を表す。PERT モデルのブロック内の①～⑥の数字はタスク名を表す。a.～e.のアルファベットはリソース名を表す。それに続く数字はそのタスクの所要期間を表し、この例では単位を日数とする。

7.4.2 プロジェクト・バッファ

クリティカル・チェーンの適用手順として、まず作業を遂行する実践者はプロジェクトのスケジュールの平均値 t_m を所要期間と想定して計画を進めることを Goldratt は指示している。平均値 t_m の所要期間でプロジェクトのスケジュールを計画すると確かに短縮は可能である。しかし遅延する確率も50%ある。

そこで計画を確実に信頼できるものにするために、バッファという概念を考える。7.3.2 項で議論したとおり、目標値 t_g による所要期間の見積りでは、個々の作業毎に安全余裕時間が含まれていた。これを平均値 t_m による見積りに変更すると、個々の安全余裕時間が浮くことになる(Goldratt, 1997)。この浮いた安全余裕時間をプロジェクト全体で集め、遅延した時のバッファとして使うことができる。すなわちバッファとは遅延したときの吸収「しろ」として利用されるものである。

Goldratt はこのバッファの量は総合計の半分で十分としている。その理由については後述する。

プロジェクト・バッファの適用について述べる。まず全ての作業についてその所要時間を目標値 t_g から平均値 t_m に設定する。図 7.6(a)のタスク①～タスク⑥の所要時間はそれぞれの目標値 t_g である。これらをそれぞれの平均値 t_m、ここではそれぞれ目標値 t_g の半分の時間に設定する。以後図 7.6(b)～図 7.6(d)に示すようにこれらの時間は変わらない。次にプロジェクト全体の遅延保護を目的として、プロジェクトの最終段にプロジェクト・バッファを付け足す。図 7.6(a)では、タスク①、タスク②、タスク③、タスク⑥の経路の所要時間の合計は 64 日であった。これが各タスクの所要時間の設定変更により 32 日となったので、図 7.6(b)に示すように、削減された分すなわち 32 日をプロジェクト・バッファとしてタスク⑥の後段に付け足す。

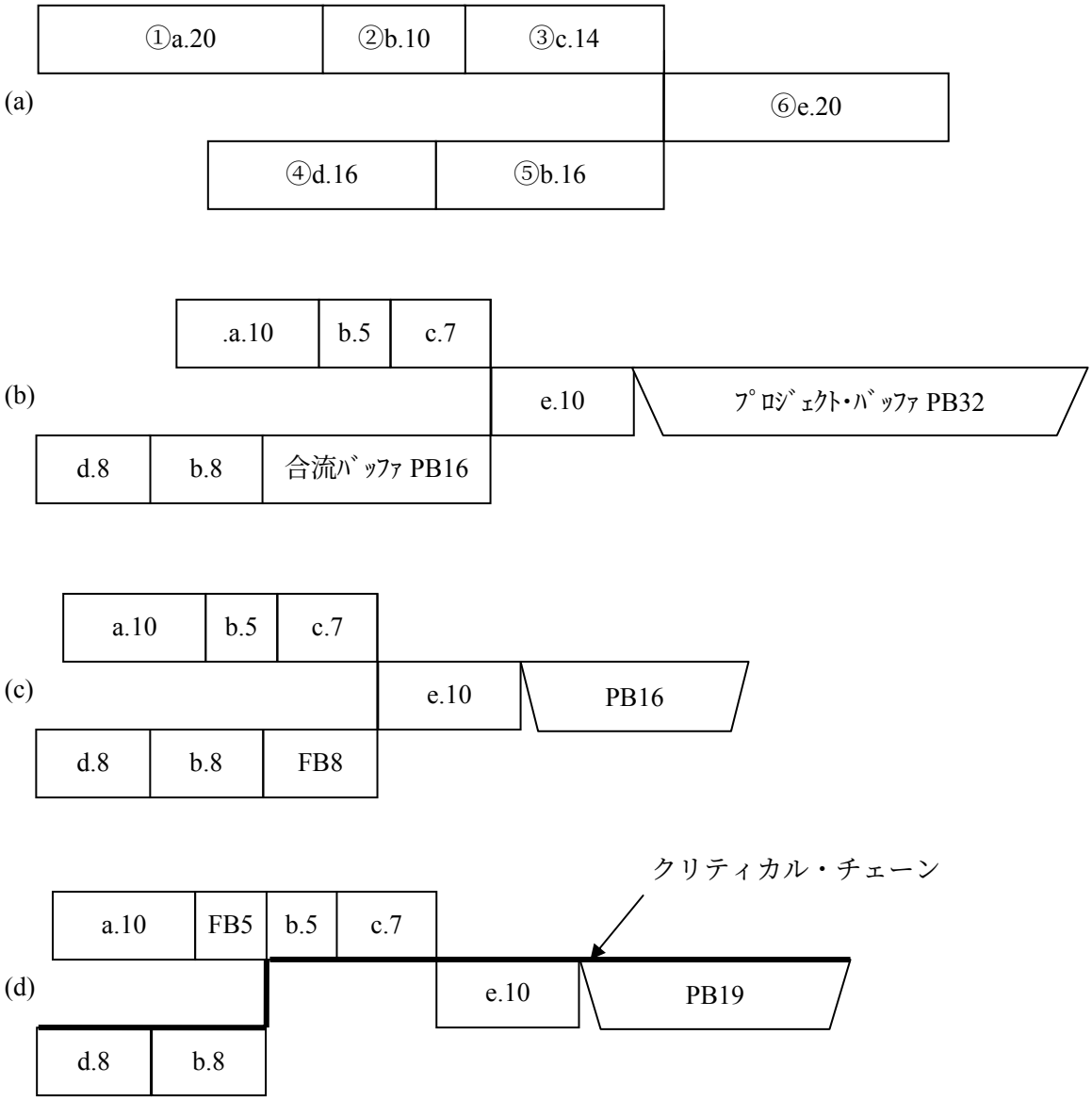


図 7.6　クリティカル・チェーン適用手順(Cook, 1998; Yeo & Ning, 2002)

7.4.3　合流(Feeding)・バッファ

クリティカル・パスの流れを守るために、クリティカル・パス以外の経路がクリティカル・パスに合流する点に「合流バッファ」を入れる。これはクリティカル・パスを遅延させないための方策である。クリティカル・パス以外の経路が、万が一遅延したとしても、この合流バッファで吸収できる。つまりクリティカル・パスは合流バッファによりその存在は保証される。

図7.6(a)において、クリティカル・パスはタスク①、タスク②、タスク③、タスク⑥の経路である。タスク④とタスク⑤の経路がクリティカル・パス以外の経路となるので、クリティカル・パスと合流するタスク⑤の後段に合流バッファを挿入する。図7.6(a)では、タスク④とタスク⑤の経路の所要時間の合計は32日であった。これが各タスクの所要時間の設定変更により16日となったので、図7.6(b)に示すように、削減された分すなわち16日を合流バッファの量とする。

7.4.4　リソース競合の回避

Goldratt は適切なバッファ量として総合計の半分を推奨している。図7.6(b)の場合で考えるなら、プロジェクト・バッファと合流バッファの二つの所要期間を半分にすれば良い。それを図7.6(c)に示す。プロジェクト・バッファは16日、合流バッファは8日となり、合計は48日となる。尚、合流バッファを考慮するなら最大で50日となる。

しかし図7.6(c)では、リソース b は二つの経路上に存在していることが認められる。そして同時期に双方の作業の一部が競合する様に配置されている。この状態はマルチタスキングである。CPM や多くのプロジェクト・マネジメント・ソフトウエアはその扱いを可能としている(Cook, 1998)。しかしマルチタスキングは7.3.2項で述べた通り、現実的には大きな生産性は得られない。従ってできる限りリソース競合は回避すべきと考えられる。

リソース競合が発生する場合には、競合するタスク間を合流点とし、最長経路ではない経路に合流バッファを早めに置くことで解決できる。図7.6(c)では、d.8と b.8から b.5にかけてクリティカル・チェーンがあるので、b.5の直前に b.8を先行タスクとして置く。そして d.8から b.5にかけてのクリティカル・チェーンの不確実性を保護するために、クリティカル・チェーンに影響を及ぼしうる a.10から合流点の間に合流バッファを挿入する。そして合流バッファの量は、a.10と合流バッファが、d.8から b.5にかけての経路を超えない量に設定する。すなわち合流バッファは5日となる。図7.6(c)で設定していた合流バッファは8日であったので、残りの3日分については最終段のプロジェクト・バッファに付け加えて調整する。CCM によって最終的に図7.6(d)を得る。

結果として、リソース競合の為に合流バッファが挿入され、クリティカル・チェーンの調整と延長が行われた。最終的には所要期間の合計は57日となる。

7.4.5 バッファ量のサイズ

Goldratt が強く提案しているのは、前述の通りプロジェクト・バッファの量を半分にすることである。バッファ量を半分にする提案は Goldratt の恣意的数値である。バッファ量のサイズを半分にさせる根拠は示されていないとの批判(Yeo & Ning, 2002)がある。この欠点を克服する為に、Herroelen & Leus は経験則の利用を提言している。プロジェクト・マネジメントは過去の経験や類似のプロジェクトを参考に実践されることが多い。従ってその都度、適切なバッファ・サイズを決めて、適用することを薦めている(Herroelen & Leus, 2001)。

それではなぜバッファ量を減少させることができるのであろうか。これはプロジェクト全体の分散は各作業の分散の総和に比べて小さいという定理で容易に解釈される(Herroelen & Leus, 2001) (Steyn, 2000)。つまり、プロジェクト全体で遅延を保護するのに必要なバッファ量は、各作業の安全余裕度を取り除いた各作業の総和より小さいという意である。

これは一つのアナロジーとしてリスクに対する保険の原理で説明できる。この場合、分散はリスクとして解釈される。まず x 人の資産家がそれぞれ y 円の資産を所有していたと仮定する。各資産家は不測の事態に備えて y 円に相当する保険金を付保していたとする。もし資産家全てが、一度に保険支払いの対象となることを想定するなら、保険会社は $x \cdot y$ 円相当の積立金を用意しなければならない。しかし x 人の資産家が同時に資産を失うことは考えにくいので、積立金は明らかに $x \cdot y$ 円より少なくて済む。つまり個々の資産家にとって不測の事態が起きるのは事前には判らないが、同じリスクに直面している人を多く集めるとその集団全体としての不測の事態の発生数はほぼ確定する。そして保険会社の支払う保険金額の合計は正規分布になると言うことである。

これは中心極限定理から説明ができる。本定理によれば、独立な確率変数の和の分布は、その項数が大きくなると、各々の確率変数の分布の型には無関係に、一定の正規分布に近づくことが知られている。

そこでまず各作業の分散の総和を求めてみる。ここでは n 個の独立した確率変数の分布が全て同じ分散 V であると仮定する。分散の総和 V_Σ は次式で表される。

$$V_\Sigma = n \cdot V \tag{7.1}$$

プロジェクトのスケジュールのリスクは分散又は標準偏差によって表現できる。散らばり具合が大きいほど期待値からのずれが大きく、リスクが大きくなる。逆に散らばりが小さければリスクは小さい。スケジュールの不確実性は明らかに散らばりの大きさに依存している。ここでは標準偏差をリスク指標として議論を進めてみる。各作業の標準偏差を σ とするなら、分散は標準偏差の二乗であるので、$V = \sigma^2$ である。ここでプロジェクト全体の標準偏差を σ_Σ とするなら、

$$\sigma_\Sigma = (V_\Sigma)^{1/2} = (n \cdot V)^{1/2} = n^{1/2} \cdot \sigma \tag{7.2}$$

となる。そして $n^{1/2} \cdot \sigma < n \cdot \sigma$ であるので、

$$\sigma_{\Sigma} < n \cdot \sigma \tag{7.3}$$

である。つまり、独立したリスクをまとめて同質的な集団として捉えるなら、その全体リスクは低減するということを意味する。$n^{1/2}$と n の関係から、n が大きくなるほどその効果は大きい。CCM のアプローチはこの中心極限定理の性質から、プロジェクトのスケジュール・リスクを低減させるものと言える。

7.5　設計知のアウトソース化

半導体の設計開発には価格対性能比の向上が求められている。その為に、エレクトロニクス・システムは大規模化と複雑化が進行している。大規模化と複雑化が進行しているシステムはサブシステムに分割することで問題を分割できる。そしてそれぞれのサブシステム上の問題を解決し、統合化させる手法がとられる。これは設計の抽象性のレベルを上流から下流に移行させているので、トップダウン手法とも呼ばれている(Armstrong, 1989)。

大規模・複雑化するシステムが進行するなら、相対的に設計開発資源は一層不足することになる。しかし企業の設計開発の資源には限りがある。企業にとっては、特に経験とノウハウといった設計知識と能力を持つ技術者を必要な時にすぐ雇い入れることは簡単ではない。従って、昨今の半導体の設計開発にはサブシステムの設計を外部の受託設計会社にアウトソースする場合がある。あるいは価格対性能比の向上の問題解決を目的として、その能力の高い企業へアウトソースする場合もある。この場合には、設計情報並びに設計知識（以下設計知と称す）の知的財産（Intellectual Property:以下 IP と称す)をライセンス導入して、問題解決をはかっている。この IP とはサブシステムにそのまま利用できるモジュールと考えても良い。つまり IP をライセンス導入する理由は時間を買うこととも言える。

本章で議論する「設計知のアウトソース」とは、受託設計会社に設計を委託するか又は IP というモジュールをライセンス導入することと定義する。設計知のアウトソースとは、アウトソースの一般的メリットである経営資源の補充並びに繁閑の平準化の他に、価格対性能比の向上追求と設計開発期間の削減に向けた一手段と言える。

設計知のアウトソースを利用したプロジェクトは、サブシステムへの分割とサブシステム間の複雑な統合作業を要する。その際には相当数の技術者と財源が要求される。設計知のアウトソースの作業はスケジュールに従属する。なぜならば作業の優先順位や経営資源の手配そして制約条件と関連性があるからである。

次に設計知のアウトソースは、設計計画、設計の調達、設計の統合という過程を踏むと考えて、その過程を以下に述べる。

設計計画の工程時期においては、まずプロジェクトの主体企業が製品仕様と工程仕様を

定義する。特にサブシステム間のインタフェースの定義は十分注意が払われる。それはサブシステム間の統合の成否を決めるからである(Boldwin & Clark, 2004)。そしてアウトソースの委託先にサブシステムの製品仕様が伝えられる。この工程時期には多くの意思決定が行われるのであり、プロジェクト自身の成否に影響を与える箇所である。この意思決定はプロジェクト全体の資金の枠取りだけでなく、プロジェクト実施に必要な資源と計画通りに完遂させる為の資源の策定が含まれる。

7.6　統合アプローチの提案と半導体設計開発プロジェクトへの適用

7.6.1 統合アプローチの適用にあたって

これまでに議論してきた三つの要素を統合させた手法を統合アプローチとして提案する。一つ目の要素は、ガント・チャート、PERT と CPM といったスケジュール管理手法の利用である。二つ目の要素は TOC を拡張した CCM である。CCM は人間行動の特性を考慮し、バッファという概念を用いて無駄な時間をなくす手法である。三つ目の要素はアウトソース化である。これにより限られたリソース問題を解放させる。

本節では統合アプローチを半導体設計開発プロジェクトに適用して、工期短縮の効果を検証する。

統合アプローチの適用にあたっては、まず不確実性の一つであるタスクの時間をどの様に対処するか検討を加える必要がある。タスク時間は一元的に示すことができないのであるから、確率密度関数を用いた確率論的アプローチが適当である(Vose, 2003)。Goldratt はタスク時間の見積もりの不確実性について、ABP (Aggressive But Possible)と HP (Highly Possible)という二つの概念を適用して計画策定を奨めている。ABP とは実現確率50%、HP とは実現確率90%であり、つまりこれらの二点見積もりを行っている。本節でもこの二点見積もりを行う。次にバッファについての検討が必要である。リソースの競合についてはプロジェクトの制約条件を洗い出し、バッファを追加設定する。また設計開発期間全体の最適化については、クリティカル・チェーン工程の後にプロジェクト・バッファを設定する。

7.6.2 半導体設計開発プロジェクト

統合アプローチを適用する半導体設計開発プロジェクトは、コンピュータの拡張スロット及びシステム・バスである PCIExpress インタフェースの機能を搭載したシステム LSI の設計開発プロジェクトである。PCIExpress インタフェースとは PCI (Peripheral Component Interconnect)の後継として開発された拡張バスの規格である。シリアル転送を採用することで高速化されると同時に、システムの柔軟性が増した。データ転送速度は500MB/秒であるが、複数のチャンネルを束ねての高速転送が可能となっている。例えばグラフィック・ボードの接続に使う PCIExpress x 16では、データ転送は8GB/秒に達する。このシステム LSI

の仕様概要を表7.1に示す。

プロジェクト・チームはシステム設計者2名、ハードウエア記述言語設計者5名、物理設計技術者2名、プロジェクト・マネージャ1名の10名から構成される。経営資源の問題により、設計受託企業への設計委託並びに半導体 IP をライセンス導入している。つまり設計のアウトソース化を行っている。

7.6.3　統合アプローチの適用

必要な設計情報を収集した後、基本設計工程以降の詳細スケジュールを設計した。詳細スケジュールは以下の手順に基づき作成した。

手順1. 作業項目を WBS (Work Breakdown Structure)に展開する。そこでは TOC の考えを意識せずに策定する。

表 7.1　本プロジェクトにおけるシステム LSI の仕様概略

- PCI Express バス物理層シングル・チャンネル
- CSM (Chartered Semiconductor Manufacturing 社)の0.18 um プロセスで製造された半導体
- パッケージ封止(81-pin BGA)として提供
- PCI Express 仕様 Rev 1.1及び PIPE v 1.00互換
- 8-bit、16-bit、10-bit (TBI) MAC インタフェースをサポート
- PCI Express 電源管理ステート をサポート
- シリアル及びパラレル・ループバック・テスト・モード、エラー・チェックの PRBS サポート
- スペクトラム拡散クロック(SSC)のサポート
- LVDS(Low Voltage Differential Signal:低電圧作動信号)のリファレンス・クロック100 MHz 又は125 MHz のサポート
- 高速 Current Mode Logic(CML、2.5 GHz)のシリアル I/O
- Stub Series Termination Logic, Class 1 (SSTL-18) (125/250 MHz)パラレル I/O 内蔵高速システムインタフェース
- 電源電圧3.3V 及び1.8V
- 動作範囲0ºC～ 85ºC

手順2. WBS 毎の所要期間を見積もる。所要期間は開発体制と作業効率を考慮し、ABP と HP による二点見積もりを行う。ここで見積もりした ABP を用いてスケジュール作成を行う。WBS とタスクの作業概要、ガント・チャートを図7.7に示す。ID は WBS 番号である。

手順3. 各作業の依存関係を把握し、合流ポイントを明確にする。この時点で最終目標納期内に収まる様に作業の依存関係を見直す。

手順4. クリティカル・パスとなる工程を明らかにする為にネットワーク図を準備する。ここでのスケジュールに従って実際の設計開発が行われた。本プロジェクトの所要期間は計画上、413日である。

手順5. プロジェクト全体のリソース状況、つまりリソースの不足とリソースの競合を洗い出す。リソースの不足は WBS34のデジタル・ブロックの設計と、WBS39の検証作業の一部である。リソース不足箇所の設計と検証作業については社外の委託会社にアウトソースする。

手順6. 各タスクの所要時間の見積もりを ABP から HP に変更する。そして合流ポイントを再確認し、プロジェクト・バッファを挿入する。また、合流バッファを挿入する。合流バッファは、クリティカル・パスに合流する側の作業と合流点の間に設定する。図7.8のネットワーク図にクリティカル・パスの経路とバッファの挿入箇所を示す。図7.8内の数字は WBS を表し、PB は合流バッファ、FB は合流バッファである。

手順7. 全てのバッファ量を半分に減らし、クリティカル・チェーンの経路の探索に移る。

手順8. クリティカル・チェーンの経路の探索の前に、タスク間でリソースの競合する箇所をガント・チャートにより識別する。WBS16と WBS17のリソースが、送信側ブロックである WBS2のリソースと競合している。同様に WBS26も、WBS2のリソースと競合している。WBS42は、WBS34と競合している。図7.9のネットワーク図に共通リソースにより競合関係にあるタスク群を示す。これらの競合状態をバッファの挿入・調整によってクリティカル・チェーンを設定する。

手順9. リソースの競合状態を回避する為に、合流バッファの再設定と調整を行う。この調整により、タスクの作業開始時にリソース問題がない様にスケジュール管理することができる。図7.10にクリティカル・チェーンとバッファの挿入後のガント・チャート図を示す。

ID	タスクの概要	タスク名	期間
1	**全体のスケジュール**		438日
2	**送信側ブロック**	**TX-Path**	**333日**
3	サブブロック設計	TX-PLL	140日
4	モジュール設計	VCO	28日
5	セル設計	DCELL	7日
6	パッド設計	CML-CMOS I/O	7日
7	パッド設計	CML Phase MUX	7日
8	サブ検証	Corner Simulation	7日
9	サブモジュール設計	DPFD	35日
10	サブモジュール設計	CPI	28日
11	サブモジュール設計	CPP	28日
12	サブモジュール設計	DIV10	14日
13	サブモジュール解析作業	Loop Analysis	7日
14	サブブロック設計	PISO	7日
15	サブブロック設計	Barrel Shifter	7日
16	サブブロック設計	CML-Driver	63日
17	サブブロック設計	Termination	42日
18	**受信側ブロック**	**RX-Path**	**207日**
19	サブブロック設計	APD	7日
20	サブブロック設計	Phase-MUX	7日
21	サブブロック設計	SIPO	7日
22	サブブロック設計	DLF	14日
23	サブブロック設計	Lock Detected	14日
24	サブブロック設計	Barrel-Shifter	7日
25	サブブロック設計	RX-Slicer	7日
26	サブブロック設計	Termination	28日
27	**送受信共通ブロック**	**Shared Circuits**	**91日**
28	サブブロック設計	Master Bias	14日
29	サブブロック設計	Termination Bias	7日
30	サブブロック設計	SSTL Drivers	14日
31	サブブロック設計	SSTL Receivers	14日
32	サブブロック設計	LVDS Drivers	28日
33	サブブロック設計	LVDS Receivers	14日
34	**デジタル・ブロック**	**Digital Core**	**158日**
35	サブブロック設計	PCS	95日
36	サブブロック設計	Elastic Store	14日
37	サブブロック設計	Parallel Interface	7日
38	論理合成作業	Synthesis	42日
39	**システム・レベル**	**System Level**	**278日**
40	アウトソースとの接続作業	PCSとサード・パーティ回路（配置配線前）の接続	14日
41	アウトソースとの接続作業	PCSとサード・パーティ回路（配置配線後）の接続	7日
42	デジタルとアナログの接続	PCS to Analog	21日
43	システム検証	PCS Final Verification	42日
44	**物理設計**	**Layout**	**266日**
45	アナログ・ブロック	Analog	105日
46	デジタル・ブロック	Digital	84日
47	ブロックの統合	Chip Integration	14日
48	**テープアウト**	**Tapeout**	**14日**
49	最終デザイン・ルール確認	Final Cleanup	7日
50	ファンダリへの製造受け渡し	Relase to Silicon Foundary	7日

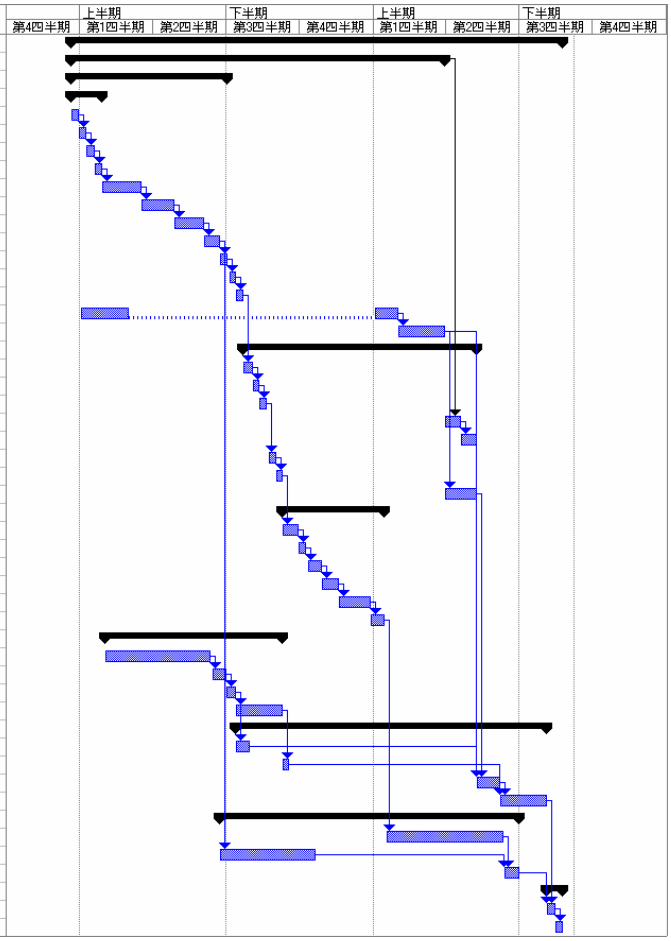


図 7.7　ガント・チャート

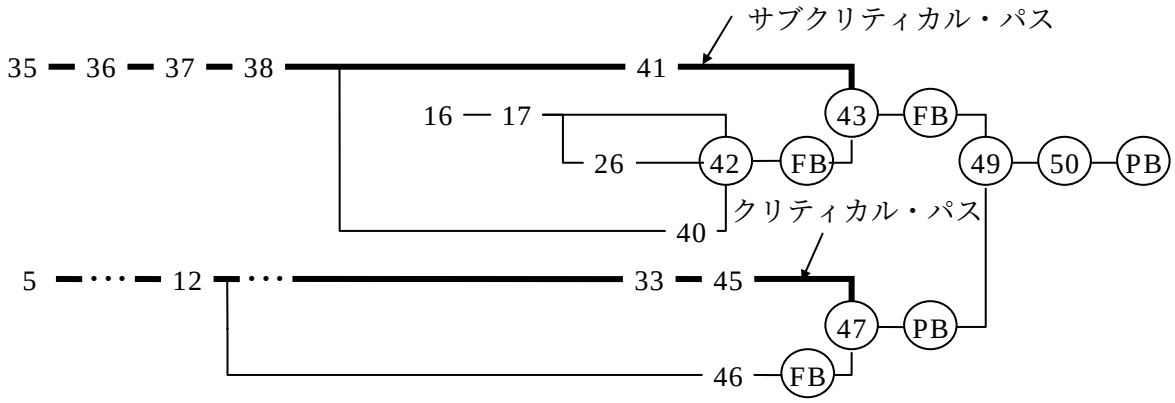


図 7.8　ネットワーク図概要

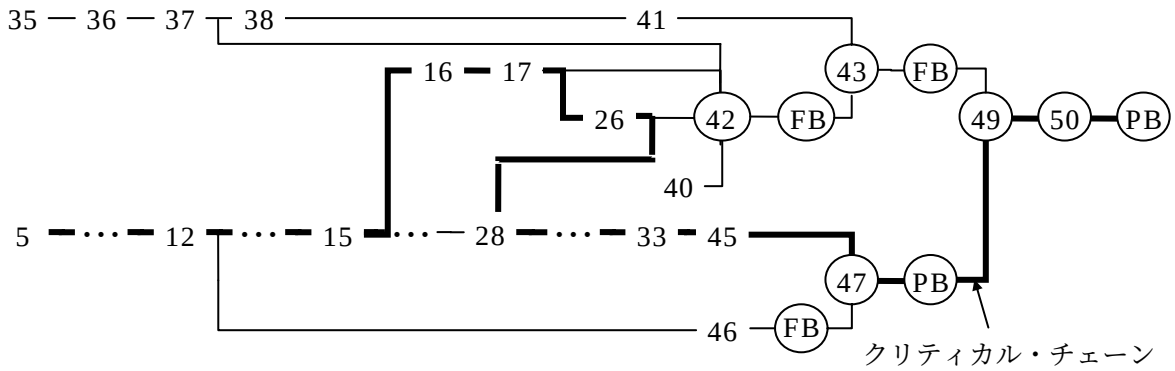


図 7.9　クリティカル・チェーン

ID	タスクの概要	タスク名	期間
1	**全体のスケジュール**		348日
2	**送信側ブロック**	**TX-Path**	**119日**
3	サブブロック設計	TX-PLL	70日
4	モジュール設計	VCO	14日
5	セル設計	DCELL	3.5日
6	パッド設計	CML-CMOS I/O	3.5日
7	パッド設計	CML Phase MUX	3.5日
8	サブ検証	Corner Simulation	3.5日
9	サブモジュール設計	DPFD	17.5日
10	サブモジュール設計	CPI	14日
11	サブモジュール設計	CPP	14日
12	サブモジュール設計	DIV10	7日
13	サブモジュール解析作業	Loop Analysis	3.5日
14	サブブロック設計	PISO	3.5日
15	サブブロック設計	Barrel Shifter	3.5日
16	サブブロック設計	CML-Driver	21日
17	サブブロック設計	Termination	21日
18	**受信側ブロック**	**RX-Path**	**56日**
19	サブブロック設計	APD	3.5日
20	サブブロック設計	Phase-MUX	3.5日
21	サブブロック設計	SIPO	3.5日
22	サブブロック設計	DLF	7日
23	サブブロック設計	Lock Detected	7日
24	サブブロック設計	Barrel-Shifter	3.5日
25	サブブロック設計	RX-Slicer	3.5日
26	サブブロック設計	Termination	14日
27	**送受信共通ブロック**	**Shared Circuits**	**45.5日**
28	サブブロック設計	Master Bias	7日
29	サブブロック設計	Termination Bias	3.5日
30	サブブロック設計	SSTL Drivers	7日
31	サブブロック設計	SSTL Receivers	7日
32	サブブロック設計	LVDS Drivers	14日
33	サブブロック設計	LVDS Receivers	7日
34	**デジタル・ブロック**	**Digital Core**	**79日**
35	サブブロック設計	PCS	47.5日
36	サブブロック設計	Elastic Store	7日
37	サブブロック設計	Parallel Interface	3.5日
38	論理合成作業	Synthesis	21日
39	**システム・レベル**	**System Level**	**109.75日**
40	アウトソースとの接続作業	PCSとサード・パーティ回路（配置配線前）の接続	7日
41	アウトソースとの接続作業	PCSとサード・パーティ回路（配置配線後）の接続	3.5日
42	デジタルとアナログの接続	PCS to Analog	10.5日
43	システム検証	PCS Final Verification	21日
44	**物理設計**	**Layout**	**171.5日**
45	アナログ・ブロック	Analog	52.5日
46	デジタル・ブロック	Digital	42日
47	ブロックの統合	Chip Integration	7日
48	**テープアウト**	**Tapeout**	**239.5日**
49	最終デザイン・ルール確認	Final Cleanup	3.5日
50	ファンダリへの製造受け渡し	Relase to Silicon Foundary	3.5日
51			
52	合流バッファ	合流バッファ(42-PB-43)	33.25日
53	合流バッファ	合流バッファ(43-PB-49)	101.75日
54	プロジェクト・バッファ	プロジェクト・バッファ(47-PB-49)	99.5日
55	プロジェクト・バッファ	プロジェクト・バッファ(50-PB)	3.5日
56	合流バッファ	合流バッファ(46-FB-47)	21日

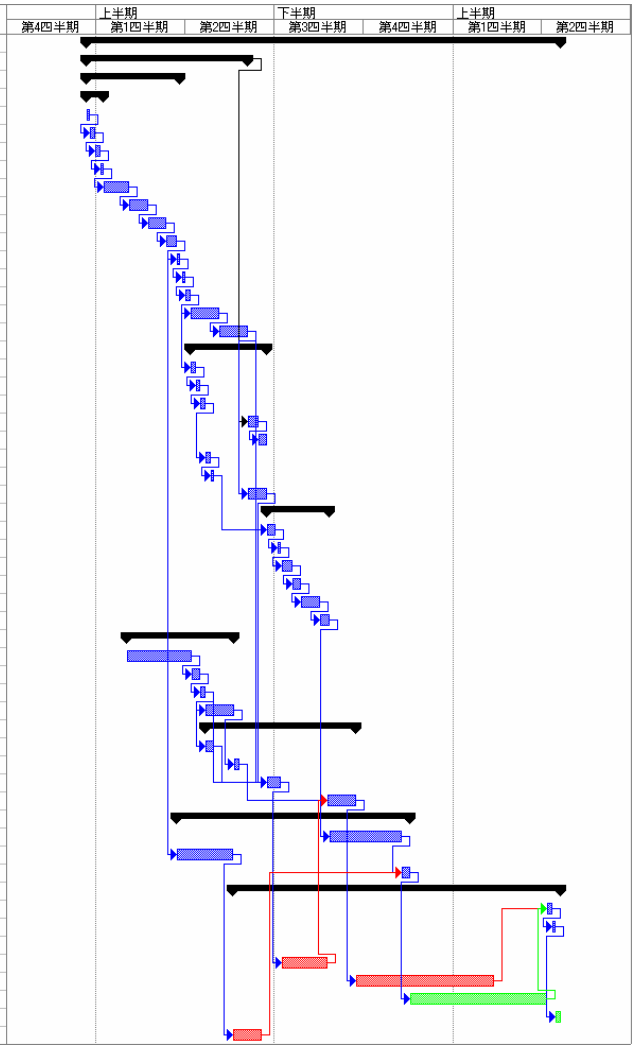


図 7.10　最終ガント・チャート

7.6.4 統合アプローチの適用結果

統合アプローチの適用前と適用後のベンチマーク対比を行った。統合アプローチの適用前とは、プロジェクト・マネジメントとして伝統的な PERT と CPM による手法である。適用前におけるプロジェクト全体の所要期間は 413 日であった。各タスクの所要時間を ABP から HP に変更し、バッファを挿入した後の所要期間に変更はなかった。全てのバッファ量を半分に減らすことで、所要期間は 309.5 日となった。しかしリソース競合の問題を解決しクリティカル・チェーンとして再構築した後の所要期間は、348 日となった。結果、統合アプローチの適用前後で 65 日の短縮、すなわち従前比 15.7%の短縮効果が得られた。

WBS や作成する文書類は従来と変更するところはないので、リソースへの混乱は来たさなかった。また従前と変わらずスケジューリングし、後に安全余裕度の削除とバッファ挿入を利用した手法は既存の体系を崩さずに出発することができた。このことから、プロジェクト・マネージャと設計者が制約条件の排除を意識的に取り組むことが求められると言える。

7.7　考察

7.7.1 アウトソース化と設計開発期間の短縮

Goldratt は CCM においてアウトソース化について言及していない。しかし本章で提案した統合アプローチにはアウトソース化が含まれる。統合アプローチにおけるプロジェクトのターゲット先は商品、つまり一つのシステムである。このシステムが大規模・複雑化していくなら、相対的に設計開発資源は一層不足することになる。しかし企業の設計開発の資源には限りがある。企業にとっては、特に経験とノウハウといった設計知識と能力を持つ技術者を、必要な時にすぐ雇い入れることは簡単ではない。従ってサブシステムの設計を外部の受託設計専門会社にアウトソース化することで経営資源の補充が可能となる。統合アプローチを適用して検証を行った半導体設計開発プロジェクトでは、それが設計知という知的財産である。これは価格対性能比の向上ができる設計知であり、サブシステムにそのまま利用できるモジュールである。つまり、設計知のアウトソース化とはアウトソースの一般的メリットである経営資源の補充並びに繁閑の平準化の他に、価格対性能比の向上追求と設計開発期間の短縮に向けた一手段と言える。つまり設計知のアウトソース化は所要期間の短縮化を補強するものであると言える。

7.7.2 CCM の課題

CCM の適用を通じて、CCM では言及していない領域が浮き彫りになった。それは、i)品質、ii)コスト、iii)作業の反復活動(iteration)である。

一つ目の領域はプロジェクトの品質である。Goldratt はプロジェクトの品質とプロジェク

トの所要期間は二律背反の問題にあると指摘している。本章では製品投入時期の重要性について議論してきた。しかし、たとえ適切な時期に製品を投入したとしても、誤った製品ではその議論は全く無意味なことである。つまり、プロジェクト・マネージャが目標品質を犠牲にせずに、どうやって目標納期の短縮を実現できるのかがこれからの課題の一つである。

二つ目の領域はコストに関するものである。プロジェクトのコストとプロジェクトの所要期間も二律背反の問題である。本章では製品投入時期について議論してきたのであり、コストはその範疇ではない。しかし仮に納期が短縮された場合、二律背反が前提であればコスト増しを同時に誘発することになる。コスト計画を検討する場合には、その上昇分を想定したバッファで解決できる可能性がある。しかしそういった調整が効かない取引相手であれば、その融通はうまく働かない。つまり目標コストを犠牲にせずに、どうやって目標納期の短縮を実現できるのかが二つ目の課題である。

作業の反復活動は CCM が言及していない三つ目の領域である。一般的にプロジェクトは複数のタスクで構成される。そしてプロジェクト内のタスク間で互依存関係を持っている場合がある。つまり、プロジェクトにおいて活動の流れが下流から上流へ手戻り修正する事例がある。その反復活動を図7.11で表現した。Eppinger 等は作業間のモノ的な流れを捨象し、情報と知識の流れに注目して反復活動を数理的にモデル化している(Eppinger, 2001)。そして反復活動の解析やシミュレーション可能なアプローチを提唱してきた。実際、反復活動を考慮した所要期間の見積もり予測は容易ではないが、数理的なモデルにより期待値を創出することは可能である。そこでタスク間で反復活動が予測されるならば、その反復活動に要する所要期間をスケジュール管理に反映させる必要がある。その場合には反復活動の専用バッファを検討すべきである。

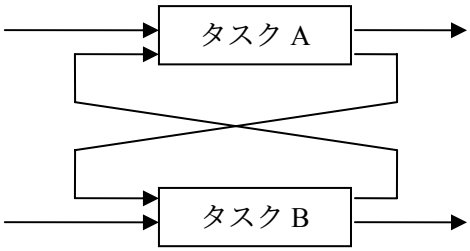


図 7.11　タスク間の反復活動

7.7.3　アウトソース先のタスク所要期間

アウトソース企業のタスクの所要期間について検討する。なぜならば、アウトソース企業と協業する上で必要な厳格なタスクの納期について論じていないからである。統合アプローチの検証において、アウトソース企業による各タスクの所要期間の定義を全て社内のタスクの定義と同一として扱っている。しかしアウトソース企業の場合には契約納期として契約書に明文化させるのが一般的であり、ABP(目標納期まで完遂確率が 90%前後)やHP(目標納期まで完遂確率が 50%)といった蓋然的な納期を規定することはない。従ってタスクの所要期間を HP でアウトソースを利用している場合には、タスクの 50%分を予め追加しておく必要がある。実践的には契約納期と併記しておくと混乱は避けられる。

7.8　本章のまとめ

製品開発から市場投入までの期間を短縮化することは競争優位の点で重要になっている。特に製品のライフ・サイクルが 2 年以下の業界ではその傾向が著しい。従って、適切なタイム・マネジメントにより他企業に先行して早く製品投入できるなら競争優位に立てる。タイム・マネジメントは競争の激しい製品市場においては一層、企業の競争戦略に欠かせない領域の一つとなっている。

CCM (Critical Chain Management:クリティカル・チェーン管理)は、伝統的なプロジェクト・マネジメントでは想定されていない人間行動の特性と競合リソースの問題を考慮した管理手法である。Goldratt の TOC (Theory of Constraint: 制約理論)を拡張させた。しかしガント・チャート、PERT (Project Evaluation and Review Technique)、CPM (Critical Path Method)といった伝統的なスケジュール管理も堅持しているので、拡張型プロジェクト・マネジメントと言える。CCM の手順としては、まず各タスクの安全余裕時間を大胆に取り除き、個々に浮いたその時間をプロジェクト全体で集める。その浮いた時間をバッファとしてまとめ、遅延時の緩衝とする。そして競合リソースを回避する為にクリティカル・チェーンを構築する。結果として無駄と考えられる時間は削除され、プロジェクト全体の所要期間は短縮される。これはプロジェクト全体の遅延の分散は各タスクの遅延の分散の総和に比べて小さくなる性質に基づくものである。つまり CCM は中心極限定理の性質から短縮化をはかるものである。

本章で提案した統合アプローチは、製品の設計開発期間を短縮させることをねらいとしたタイム・マネジメント手法であり、三つの手法を統合させたものである。一つはスケジュール管理手法である。ガント・チャート、PERT と CPM といったツールを使う。二つ目は CCM である。そして三つ目はアウトソース化手法である。アウトソース化より限られたリソース問題を解放させ、所要期間を削減させる。

統合アプローチを半導体設計開発プロジェクトに適用し、伝統的手法とベンチマーク対比により工期短縮の効果を検証した。結果として総所要期間を413日から348日に短縮でき、

従前比15.7%のスケジュール短縮の効果が得られた。統合アプローチは当該プロジェクトにおいて設計開発期間の短縮化に有効であることが検証された。本統合アプローチはアウトソース化という条件下でも CCM を活用できるという点で CCM の応用領域を拡張させたと言える。

最後に、クリティカル・チェーンの適用にあたって議論されるべき課題を示した。まず品質と所要期間、コストと所要期間それぞれの二律背反について言及されていない点である。品質あるいはコストを犠牲にせずに、どの様に目標納期の短縮化を実現するかが克服すべき課題と言える。そしてもう一つの課題はタスク間の反復活動を考慮していない点である。PERT や CPM といったスケジュール管理は元々反復活動を想定していない。反復活動が予測されるならば、その反復活動に要する所要期間をスケジュール管理に反映させる必要がある。その場合には反復活動に特化した専用バッファを追加検討すべきと言える。これらの課題を克服すべく統合アプローチについて検討を進め、数多くの事例の収集と蓄積を行い、そしてその適用範囲を増やすことが必要である。

参考文献

[1] Armstrong, James R.: “Chip-level modeling with VHDL”, Kuluwer Academic (1989) (ジェームス・R.・アームストロング(白石繁,山本靖訳): “VHDL デザイン・テクニック:標準ハードウエア記述言語とチップ・レベル・モデリング”,電波新聞社 (1990))

[2] Baldwin, C. Y. and Clark, K. B.（安藤晴彦訳）: “デザインルール”,東洋経済新報社, p. 85 (2004)

[3] Barber, P, Thomkins, C, and Graves, A.: “Decentralised site management – a case study”, International Journal of Project Management; 17 (2), pp. 113-120 (1999)

[4] Bloch, Arthur (倉骨彰訳): “マーフィーの法則”,アスキー出版社(1993)

[5] Clark, K. B.: “Project scope and project performance: the effect of parts strategy and supplier involvement on product development”, Management Science, Vol. 35 (10), pp. 1247-1263 (1989)

[6] Clark, K. B. and Fujimoto, T.: “Product Development Performance: Strategy, Organization and Management in the World Auto Industry”, Harvard Business School Press, Boston, MA (1991)

[7] Clark, K. B., and Wheelright, S. C.: “Managing new product and process development”, the free press, New York (1993)

[8] Cohen, W. M. and Levinthal, D. A.: “Absorptive Capacity; A New Perspective on Learning and Innovation,” Administration Science Quarterly, 35 (1990)

[9] Cook, S. C.: “Applying critical chain to improve the management of uncertainty in projects, Master Business Administration thesis", Massachusetts Institute of Technology (1998)

[10] Duncan, W. R.: “A guide to the project management body of knowledge”, Project Management

Institute Standards Committee (1996)

[11] Eppinger, Steven D.: “Innovation at the Speed of Information”, Harvard Business Review, Vol. 79, No. 1, pp. 149-158 (2001)

[12] Firdausi, Manti M., Fujimoto, Hideo, and Chen, Lian-Yi: “Applying the TOC Project Management to Operation and Maintenance Scheduling of a Research Vessel”, Series C, Vol. 46, No. 1, JSME International Journal (2003)

[13] Gantt, H. L.: “Work, Wages, and Profits”, The Engineering Magazine, New York (1910)

[14] Goldratt, Eliyahu M.: “Critical Chain”, The North River Press (1997) (エリヤフ・ゴールドラット(三本木亮訳): “クリティカル・チェーン”,ダイヤモンド社 (2003))

[15] Goldratt, Eliyahu M.: “The Goal: A Process of Ongoing Improvement”, The North River Press (1986) (エリヤフ・ゴールドラット(三本木亮訳): “ザ・ゴール ― 企業の究極の目的とは何か”,ダイヤモンド社(2001))

[16] Hamel, G. and Prahalad, C. K.: ”Corporate Imagination and Expeditionary Marketing”, Harvard Business Review, July-August (1991)

[17] Hayes, R. H., Wheelright, S. C. and Clark, K. B.: “Dynamic Manufacturing: Creating the Learning Organization”, Chapter 10 and 11, The Free Press, New York (1988)

[18] Herroelen, Willy, and Leus, Roel: “On the merits and pitfalls of critical chain scheduling”, Journal of Operations Management, 19, pp. 559-577 (2001)

[19] Huthwaite, B.: “Strategic Design: A guide to managing concurrent engineering”, Institute for competitive design (1994)

[20] 稲垣公夫: “TOC 革命:制約条件の理論”,日本能率協会マネジメントセンター (1997)

[21] 小林英三: “制約理論についてのノート”,ラッセル社 (2000)

[22] Leach, Lawrence P.: “Critical chain project management improves project performance”, Project Management Journal, 30 (2), pp. 39-51 (1999)

[23] Leach, Lawrence P.: “Critical Chain Project Management”, Artech House (2001)

[24] McMullen, Jr., Thomas B.: “Introduction to the Theory of Constraint (TOC) Management System”, St.Lucie Press/ APICS Series on Constraints Management (1997)

[25] Newbold, R. C.: “Project Management in the fast lane – applying the theory of constraints”, St. Lucie Press (1998)

[26] 野口悠紀雄: “「超」整理法・時間編:タイム・マネジメントの新技法”,中公新書 (1995)

[27] Parkinson, C. Northcorte (森永晴彦訳): “パーキンソンの法則”,至誠堂(1996)

[28] Pittman, P. H.: “The theory of constraints applied to a single project environment”, Journal of Systems Improvement, International Society for Systems Improvement, University of Georgia, Winter/Spring (1995)

[29] Pitts, John: “On time on budget the challenge ?”, winter 2000, ETSNews (2000)

[30] Project Management Institute: “A Guide to the Project Management Body of Knowledge

(PMBOK Guide) 2000 Edition”, PMI (2000)

[31] Rand, Graham K.: “Critical chain: the theory of constraints applied to project management”,, International Journal of Project Management 18, pp. 173-177 (2000)

[32] Reinertsen, D. G.: “Whodunit ? The search for the new-product killers.”, McKinsey and Company Report, pp. 35-37 (1983)

[33] Rivera, Francisco A. and Duran, Alfonso: “Critical clouds and critical sets in resource-constrained projects”, International Journal of Project Management, 22, pp. 489-497 (2004)

[34] Seering, W. P., et al.: “A center for competitive product development”, A proposal to the national science foundation directorate for engineering division of engineering education and centers (1996)

[35] 芝尾芳昭: “プロジェクト・マネジメント革新”,生産性出版(1999)

[36] Smith, Preston G.: “Winning the new products rat race”, Machine Design, May, pp. 95-98 (1988)

[37] Stalk, Jr., G.: “Time: The Next Souce of Competitive Advantage”, Harvard Business Review, July-August (1988)

[38] Stalk, Jr., G., and Hout, T. M.: “Competing against Time: How Time Based Competition is Reshaping Global Markets”, The Free Press, New York (1990)

[39] Steyn, Herman: “An investigation into the fundamentals of critical chain project scheduling”, International Journal of Project Management 19, pp. 363-369 (2000)

[40] Steyn, Herman: “Project management applications of the theory of constraints beyond critical chain scheduling”, International Journal of Project Management 20, pp. 75-80 (2002)

[41] 建部清美,初田賢司,蛯原貞雄:“ソフトウエア開発プロジェクトにおける制約理論の活用と有効性”, Journal of the Society of Project Management, Vol. 6, No. 3, pp. 8-12 (2004)

[42] 津曲公二,酒井昌昭,中憲治: “これならできる　クリティカル・チェーン”,ダイヤモンド社 (2004)

[43] Vose, David（長谷川専,堤盛人訳）: “入門リスク分析”,勁草書房, pp. 361-365 (2003)

[44] Wei, C. C., Liu, P. H. and Tsai, Y. C.: “Resource constrained Project Management Using Enhanced Theory of Constraint”, International Journal of Project Management, pp. 561-567 (2002)

[45] Williams, J. R.: “How Sustainable is Your Competitive Advantage?”, California Management Review, 43, 3 (1992)

[46] Yeo, K. T. and Ning, J. H.: ”Integrating supply chain and critical chain concepts in engineer-procure-construct (EPC) projects”, International Journal of Project Management, 20, pp. 253-262 (2002)

第8章　大規模システムの設計プロセスにおける設計知の管理と活用

8.1　設計知の活用と設計負荷の低減

半導体製造において、台湾、韓国、シンガポール、イスラエル、マレーシア、そして中国といった地域の政府は優遇税制や直接資金の調達支援を行っている(伊丹, 1995; 谷光, 1999; 藤村, 2000)。資金調達力を源泉とした現況を見る限り、半導体を製造する中枢事業はもはや加工産業かつ資本集約型事業となってきている(O'hUallachiain, 1991)。今般、ナノテク時代の半導体専門の製造ラインはより資本集約事業となり、一層巨額な規模の資本投下が必要となってきている(Baily et al., 1998)。こういった進展過程の中で、半導体の設計開発に経営資源を集中させる事業が展開され(三輪, 1999; 藤本, 2002)、付加価値の源泉は設計情報を含む設計知識に移っている。

冨山・吉川(2000)によれば、この設計知識は大きく二つに分けることができる。一つは設計対象物に関する知識である。設計対象物に関する知識には、設計対象物そのものに関する知識と設計対象物のモデル表現・生成・変更・利用に関する知識が含まれる。この設計対象物に関する知識を著作物とした知的財産が、半導体 IP (Intellectual Property)である。設計知識のもう一つは設計をどの様に進めるかという設計過程に関する知識である。この知識には設計を進める際の意思決定に向けた判断基準、そして設計の具体的な手順に関する知識が含まれる。本章のねらいは、設計対象物に関する知識と設計過程に関する知識を融合させ、半導体設計が抱える設計負荷を低減することである。

本章ではまず半導体設計手法の変遷を述べ、その変遷において生じた設計の抽象性レベルの違いによって半導体 IP を体系化する。次にシステム LSI の開発過程における近時の変容を述べ、設計開発工程における設計の手戻り修正による設計負荷の問題を指摘する。この手戻り修正が頻繁に生じると、設計開発コストの増大とスケジュールの長期化を引き起こし、経営上クリティカルな問題ともなりうる。そこで、手戻り修正を引き起こす仕組みを解明するために、従来の設計手法の手順を分析する。そして最後に手戻り修正の反復工

程を減少させるための有効なアプローチとして、半導体 IP の利用と、ハード IP を多用したレイアウト設計手法を提案する。本章で提案するアプローチの手順と従来技術の手法とを比較し考察する。

8.2　半導体設計手法の変遷

8.2.1 規模的拡大と設計の抽象性レベル

今日の情報技術が進展している社会の中で、我々は多くの複雑きわまりない事に直面し、そして簡単にそれに打ちひしがれ、途方に暮れてしまう。そういった事態を克服するにはその時直面する情報の量を制限することである。「木を見て森を見ず」、すなわち一本一本の木々を見るのでなく、森林全体を見渡すことで解決の糸口が見えてくる。今日の半導体設計も、この例にもれない。

今日の半導体は大規模・複雑化している。Gordon Moore (1965)は、「半導体に集積されるトランジスタの数は2年ごとに倍増する」というムーアの法則を打ち立てた。この間、この予測はいつか壁にぶつかるだろうとも言われてきたが、これまでのところ法則は破られていない(注1)。実際の所、半導体と半導体に密接に関連しているコンピュータはますます強力かつ安価になっている。そしてトランジスタの年間製造量は現在、年間に印刷される文字の数とほぼ同程度になっていると Moore は指摘している(Moore, 1995)。

大規模・複雑化している半導体のシステム性能を評価するなら、そして設計の具現化を行おうとするなら、システム設計者の観測する視座のレベルが重要な要素となる(Armstrong, 1989)。半導体は一つのシステムであり、そのシステムの問題点を解決するには、適当なレベルの抽象性をもって作業することである。その抽象性のレベルとは、その時に必要な情報だけをつかみ、しかも設計者に不必要な情報を与えて困惑させないレベルを意味する。

図8.1は、デジタルの世界でのシステムの抽象性レベルを示したものである(Armstrong, 1989)。抽象性の最上位レベルはアーキテクチャ(architecture)・レベルである。そして抽象性レベルが下がる順に、チップ(chip)・レベル、レジスタ・トランスファ(register transfer)・レベル、ゲート(gate)・レベル、トランジスタ(transistor)・レベル、シリコン(silicon)・レベルとなっている。

図8.2は、それぞれのレベルの代表的な構造例である。抽象性レベルは、構造的見地から階層的表現で示すことができ、下層に移るほど実体概念として表現しなければならない情報量が大きくなる。

抽象性レベルは機能的見地から階層的表現で示すこともできる。表8.1は、それぞれのレベルを構造的、機能的見地から整理したものである。

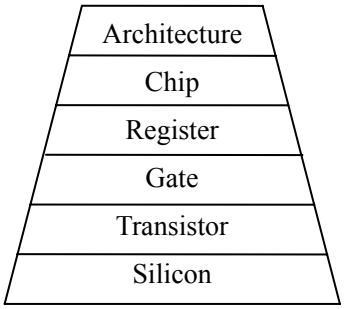


図 8.1　デジタル・システムの抽象性レベル(Armstrong, 1989）

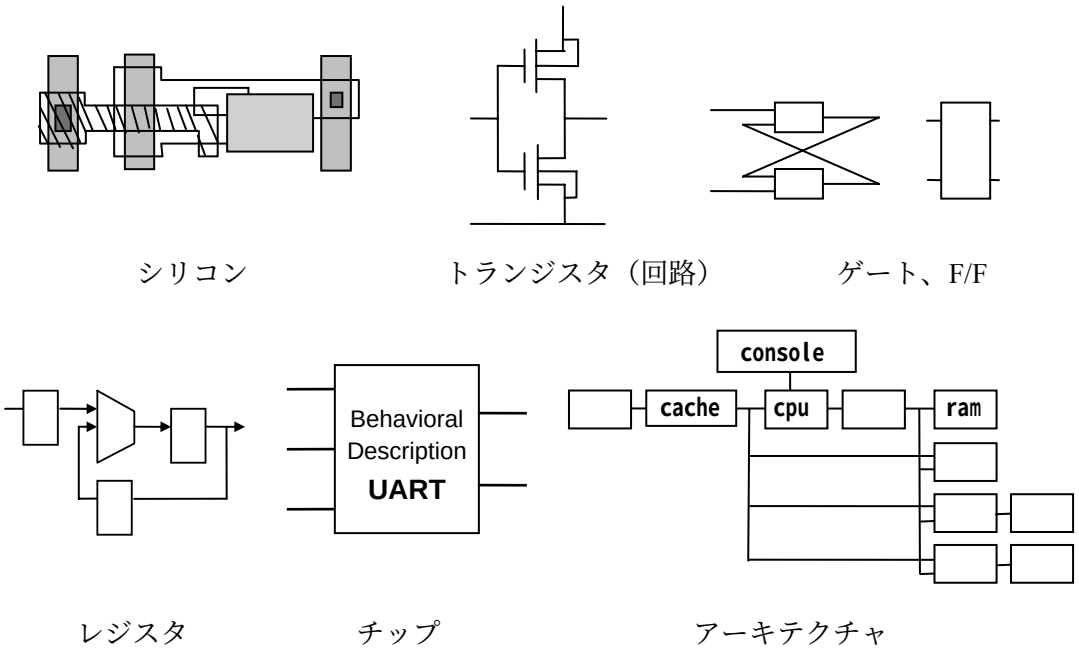


図 8.2　抽象性レベルの表示

表 8.1　各抽象性レベルの最小構成要素と機能的表現(Armstrong,1989)）

レベル	最小構成要素	機能的表現
アーキテクチャ	CPU、メモリー、バス等	パフォーマンス仕様
チップ	マイクロプロセッサ、RAM、 ROM、UART、パラレル・ポート等	I/O スピード､アルゴリズム、 マイクロ・オペレーション
レジスタ	レジスタ、カウンタ、 算術演算子、乗算器等	真理値表、状態遷移、 マイクロ・オペレーション
ゲート	ゲート、フリップ・フロップ	ブール代数
サーキット	トランジスタ、R、L、C	微分方程式
シリコン	幾何学的オブジェクト	なし

8.2.2 設計の抽象性レベルの概略

1) シリコン・レベル

各抽象性レベルについて、設計手法の変遷と併せて、下層から上層の順に述べる(電気通信学会, 1984; 室賀, 1984; 加藤, 1992)。

まず最下層のシリコン・レベルである。シリコン・レベルの基本的プリミティブ（primitive:最小単位）は、シリコン表面上の拡散面、ポリシリコン（poly-silicon:多角形にエッチングされた表層)、メタル（metal:金属膜）等の幾何学的形状である。これらを相互接続することで半導体を構築する。

1960年代、こういった構造的な表現を積み重ねることで初期の半導体は作られた。つまり、初期の半導体設計は製造プロセス条件に従って、紙・鉛筆、手でトランジスタの設計を行うデバイス設計、そしてそれらを組み合わせる回路設計をすることであった(室賀, 1984)。その設計成果として、マスク・パターン(mask pattern)が作成され、製造工程へ提供される。

2) トランジスタ・レベル

1970年代後半には、接続情報及びパラメータ（トランジスタの長さ(L: Length)、幅(W: Width)等）情報を受け取り、自動的に幾何学的、物理的な配置と配線を行うツールが提供された。このツールを使った手法をレイアウト(layout)設計手法と呼ぶ。この手法の導入により1960年代の人手によるマニュアル設計は自動化された(注2; 注3)。これにより、最小構成要素である上述のシリコン・レベルのセルの配置忘れ、接続のミス、結線忘れ及びパラメータの指定ミス等の属人的エラーを設計の段階で未然に防ぐことができる様になった。そして設計開発期間のサイクルを大幅に短縮することが可能となった。レイアウト設計では、できる限り最小の面積となるように配置を行い、配線に伴う遅延時間の減少を図るといったことが要求される(茂木,1992)。それは面積の最小化が経済性につながり、遅延時間の最小化が半導体の性能向上につながるからである。

トランジスタの基本的プリミティブはトランジスタ回路であり、インダクタンス(inductance; Lorenz: L)、コンデンサ(capacitor: C)、抵抗(resistor: R)で構成されている。R、L、C からなる電気回路についてはその現象を微分方程式によって解法させる。

3) ゲート・レベル

Conway & Mead (1979)は、システム設計を代表するコンピュータ・アキテクトがほとんど半導体設計に関心を寄せていない現状を観察し、問題提起を行った。つまりシステムを設計する技術者とレイアウト設計者の工程の二分化問題である。これが設計の生産性を妨げる要因であることを指摘した。そして将来的に半導体が大規模化していく上で、システム的見地から設計を行う必要があるとし、そのための設計手法を提案した。つまり、抽象性の高い上流レベルから設計し、下流のレベルに落とし込む方法である。これをトップダウン設計手法と呼ぶ(Armstrong, 1989)。

1980年代にはその提案の一部が実設計に取り上げられてきた。それは設計者の注視するレベルを、トランジスタ・レベルから論理ゲート(加藤, 1992)を単位として構成されるレベル、すなわちゲート・レベルに移行させることであった。論理ゲートとはブール代数(Boolean algebra)で代表される AND（論理積）、OR（論理和）、NOT(否定論理)である。さらに、それらを組み合わせた NAND(否定論理積)、NOR(否定論理和)、XOR（排他的論理和）がある(加藤, 1992)。これらゲートがトランジスタ・レベルの基本的プリミティブである。ゲートを組み合わせた、記憶機能としての最小基本構造をフリップ・フロップと言う(電気通信学会, 1984)。フリップ・フロップも基本的プリミティブである。

ゲート・レベルを視点とした設計を論理設計と呼ぶ。設計段階では各ゲート間の接続状態が規定された回路図(注4)が作成される。コンピュータ上で CAD (Computer Aided Design) ツールを駆使する要領である。ゲート・レベルは回路図だけでなく、ネットリスト(netlist)と称されるテキスト・ファイルのリスト形式でも表記できる。その中味には、ゲート回路

の情報、配線接続情報、位置情報が含まれる。

4) レジスタ・レベル

1980年代末には、業界標準のハードウエア記述言語(Hardware Description Language)と論理合成と呼ばれるツールが登場した。ハードウエア記述言語は、コンピュータ・プログラミング言語である(注5)。VHDL (VHSIC (Very High Speed Integrated Circuits) Hardware Description Language) (注6)や Verilog (注7)といったハードウエア記述言語が IEEE (Institute of Electrical and Electronic Engineers)で標準化されたことと、論理合成ツール(注8; 注9)の登場で、ゲート・レベルより一段抽象性が高いレジスタ・トランスファ・レベル(Register Transfer Level:以下 RTL と称す) (注10)という記述が設計の生産性を飛躍的に向上させることとなった(小林, 1996)。図面によって設計していたエンジニアが、ソフトウエア・エンジニアに転身しなければならないほどの変革をもたらすものであった。

RTL とは機能設計のレベルである。レジスタとは数ビットから数十ワードを一時的に記憶する高速な回路である。レジスタ・レベルにおける基本的プリミティブは、カウンタ、算術演算子、乗算器といった動作を設定したり、動作状況を読み出したりするレジスタである。一般的にはシステム仕様から、サブユニット及びそれらレジスタ間の機能的関係を規定することで LSI の動作の詳細を設計する。機能的な動作の記述には、真理値表、状態遷移図、マイクロ・オペレーションが利用される。そして機能設計の成果を基に、論理設計が行われる。論理合成ツールは、ハードウエア記述言語で表記した RTL をゲート・レベルに落とし込み、回路の最適化を実現する。つまり上流から下流への設計の自動化を実現したことが、その生産性向上をさせたのである。

5) チップ・レベルとアーキテクチャ・レベル

RTL の上位レベルは、チップ・レベルである。チップ・レベルの基本的プリミティブは、マイクロプロセッサ、RAM (Random Access Memory)、ROM (Read Only Memory)、UART (Universal Asynchronous Receiver Transmitter:シリアルとパラレルを変換させる通信回路の一種)、 パラレル・ポート等である。これらはまとまった機能を有する集積回路、又は回路ブロックである。それぞれの内部の機能は、マイクロ・オペレーションという命令セット(instruction sets)の記述、アルゴリズム(algorithm)、又はバスの I/O のスピードとして定義付けされる。マイクロ・オペレーションとは CPU (Central Process Unit)又はマイクロプロセッサの命令セットである。

最上位のレベルは、アーキテクチャ・レベルである。アーキテクチャ・レベルの最小構成要素はプロセッサ、メモリー、バス(bus)である。バスとは高速でデータを受け渡しするチャンネル(channel)である。アーキテクチャ・レベルの機能的内容は、プロセッサの性能を表す MIPS (Million Instructions Per Second)やバンド幅、仕様等で表現される。Armstrong (1989)

はアーキテクチャ・レベルをコンピュータと見立てて、PMS (Performance Modeling Systems) と呼んでいる。PMS の機能内容は、コンピュータ構造をもったシステム体系として代表されるが、構造的・機能的内容が異なるなら、最上位の分類レベルも異なる。

8.2.3 規模的拡大と自動設計

設計者は抽象性のレベルを決めて、そのレベルでマニュアル(manual)による設計を行う。そして最終的にシリコン・レベルまでマニュアルで落とし込むというのが、一般的な半導体設計手法であった。この手法はマニュアルであるが故に属人的・職人技を要するものである。この手法は分析とか解析とは逆方向の知的活動とも言える。

こういった状況下で、半導体は大規模・複雑化を要求され、電子回路設計の困難さは増大していった。微細化加工技術の進展とシステムの大規模・複雑化に対して、電子回路設計手法のミスマッチ(mismatch)が生じるようになった。つまり半導体の集積度の進行に追いつけない状況になってきた。このため、属人的・職人技に依存したマニュアル手法ではなく、合理的な標準環境を持ち自動設計することの重要性が理解されてきた。

そして Conway & Mead の提唱した設計手法が定着してきたのである。Conway & Mead の提案を整理すると三つある（藤本, 2001）。それはモジュール(module)的概念、マルチチップ(multi-chip)概念、シリコン・コンパイラ(silicon compiler)、すなわち自動化ツールである。

半導体の自動設計の概念の一つは、設計と製造プロセスをモジュール的概念で切り離すということである。すなわち設計ブロック単位で製造プロセスが変更されても、それに依存せず独立し、そのブロックが再利用されるようにしたことである。

二つ目は、マルチチップによるプロジェクト設計である。一つのシステムを意図的にモジュール化して「マルチプロジェクト・チップ」とすることである。つまり幾つかのサブシステムに分解して、それぞれのサブシステムのブロックを同時並行的に設計し、各サブモジュールを統合させる。その為にはそれぞれの設計ブロックに対して、相互の依存性と関連性を明確にしたモジュール構造にすることが必要であり、これが設計の自動化と設計時間の短縮を実現できるとした。

三つ目は、シリコン・コンパイラである。半導体を一つのシステムと考えるなら、システム思考的にその目標があり、入出力や内部の動作や機能、振る舞いを前提に考えるはずである。システム思考とはここでは設計の抽象性が高いレベル、すなわち上流のレベルである。この上流レベルを思考の出発点とするなら、シリコン表面上の拡散面、ポリシリコン、メタル等の幾何学的形状など関係ないはずである。つまり、究極のシリコン・コンパイラとは、システム設計者が上流レベルの思考的な設計によって得た結果を、すぐにシリコン・チップとなる情報（拡散面、ポリシリコン、メタル等の幾何学的形状）すなわち設計の下流レベルへ落とし込みしてくれる自動化ツールと言える。

8.3　半導体 IP コアの体系化

8.3.1 継続する規模的拡大と半導体 IP コア

前節で概観した設計手法の変遷から、半導体の設計開発技術においては、顕著な変容が見られた。本節では特に今後への影響という見地から見逃すことができない動きを指摘する。それは規模的拡大に対処する手段である。この手段とは、製品を対象とした設計知の著作物である知的財産、すなわち半導体 IP（Intellectual Property）コアの利用である。半導体の大規模・複雑化により、半導体設計者は半導体 IP コアの重要性を認識し、その拡張を進める方向にある。この半導体 IP コアとはそれ自体でプロセッサ、メモリー、周辺回路といった一つの機能を有する LSI の構成回路要素を指す。そして半導体 IP コアには回路の設計情報だけでなく、設計知が化体された文書も含まれる。

半導体 IP コアの概念を体系化すると同時に、設計の抽象性レベルの違いによってその位置付けを明らかにしていく。

8.3.2 設計知の定義

まず現在の半導体 IP コアを議論する前に、製品を対象とした設計知の定義を行っておく。設計は、一般設計学(吉川, 1985)において、概念の変換過程として論じられている。そこでは生産資源を実体概念と抽象概念に二分するなら、実体概念と抽象概念の写像操作が設計であると定義されている。例えば半導体素子そのものは実体概念である。そして設計情報や設計知識は抽象概念である。設計情報には形式化された図面、仕様書、模型等が存在する。ここでこの設計知識を構造、過程、機能という三つの視点で分解し、半導体製品における構造的知識、そして過程知識、システム知識(青島・延岡, 1997)と定義しておく。

構造的知識は形式化されている設計情報を内包する。その設計情報は Fujimoto & Clark (1991)的なパラダイムで表現するなら、設計開発工程を経ながら変形していき最終的にはシリコンという媒体に転写される。設計情報は多様な変形をしていくが、各設計情報は有機的に体系化されており知識形成が行われている。

一方、過程知識とシステム知識は形式化が困難で、簡単に移転できない暗黙知(野中・竹内, 2002; Polanyi, 2003)である。過程知識は製品に関する情報の内、時間的なコンテクストの中で意味づけられる。つまり設計開発における試行錯誤的問題解決活動の経緯の中で意味づけられている情報が過程知識である。そしてシステム知識は、製品に関する情報の内、製品システムや組織システムのコンテクストの中で意味づけられる。つまり製品システムや組織システムを構成する様々な要素関係の中で意味づけられた情報のことを指す(小川, 2001)。

本章では、上記で述べた製品を対象とした設計に関わる情報を包含した構造的知識・過程知識・システム知識をまとめて、設計知と定義する。そしてその設計知の著作物である

知的財産を IP コアと定義する。

8.3.3 半導体 IP コアの分類

半導体 IP コアはあらかじめ用意された再使用可能な回路(注11)で、システムへの組み込み、および検証を迅速に行うことができる。半導体 IP は、メガセル(mega-cell)、コア(core)、コア・セル(core-cell)、マクロ(macro)、またはバーチャル(virtual)・チップなど様々な呼称で呼ばれるが、基本的には、従来技術者が市販されている標準チップを組み合わせてシステムを構築していたブレッドボード(breadboard:実験用回路基板)の時代と同じ作業概念に基づいている。つまり、ブレッドボードがシリコン LSI に、標準チップが半導体 IP に代替された(注12)と考えれば判りやすい。

半導体の技術の進歩は新しい概念と市場を築いた。それは半導体にとっての半導体 IP である。今日市場に出回っているエレクトロニクス・システムには多くの半導体が搭載されている。それらはプロセッサ、メモリー、グラフィックスやネットワーク用標準半導体、標準論理回路、アナログ等(注13)である。そういった複数の半導体は、半導体 IP コアの有効利用により、たった一つの複雑なシステム LSI に取って代わられる事になる(注14; 注15)。半導体 IP コアとはその半導体という人工システムを構築する際のビルディング・ブロック(building block)として用いられる概念である。

一般論として IP とは、音、絵、またはビットの組み合わせなど、実体のないものを意味する。その価値の大部分は製作者の技能や芸術的才能に内在するものであり、媒体である紙、CD (Compact Disk)、DVD (Digital Video Disk)、あるいはビット・ストリームに価値が存在するものではない。その名前が示す通り、知的財産であり、かつ抽象的なものである。これを生み出すためには多大な努力、技能、および芸術的才能を要するが、媒体自体には、ほとんどコストはかからない。

半導体 IP には、機能的に設計した成果として得られる「ソフト IP」と、レイアウト設計の成果として得られる「ハード IP」、そしてそれらの中間に位置する「ファーム IP」の三種類が存在する(注16)。以降では、三種類の半導体 IP コアの特徴を整理してまとめる。三種類の半導体 IP それぞれの特徴を村田 et al. (2001)が提示している特徴と統合し、表8.2に示す。

表 8.2　ソフト IP、ファーム IP、ハード IP の特徴（村田 et al., 2001:加筆）

	ソフト IP	ファーム IP	ハード IP
設計の抽象性レベル	レジスタ・レベル	ゲート・レベル	シリコン・レベル
フォーマット	RTL コード	テクノロジに依存したネットリスト	マスク・パターン
利便性と再利用	非常に高い（ただしネットリストが確実に設計検証されていることが前提である）	高い	高い（レイアウト設計で検証されていることが前提である）
シリコンへの最適化	低い（論理合成ツールに依存する。想定した性能で動作しない可能性は否定できない）	高い	非常に高い
インテグレーション化に係わる作業の大きさ	非常に高い。既存のレイアウト環境をある程度自由に利用することができるが、大規模化に限界がある。ユーザは更に論理回路設計と検証を十分行う必要があり、多大な時間と経費が予測される。	高い。更に配置・配線作業を行いレイアウト設計する必要がある。論理設計とレイアウト設計双方間で多くのイタレーションが予測される。場合によっては RTL 設計の見直しと修正を行う必要がある。	低い。レイアウト設計上でブロックを集積化させる。IP の数量と IP の配置技術に依存する。ツール技術の確立によって、作業量は極めて小さくなる可能性がある。
論理回路の仕様変更からレイアウトまで	困難（HDL からレイアウト工程まで全て行う必要あり）	困難（HDL からレイアウト工程まで全て行う必要あり）	比較的容易（仕様の変わったブロックのみ変更する）
テストの容易化	テスト回路は配置に左右される	テスト回路は配置に左右される	テスト回路は IP の脇に置くことが可能
市場の競争度合い	激しく高い（障壁参入は非常に低い）	普通（障壁参入は低い）	低い（障壁参入は高い）
IP コアとしての価値、価格動向等	平均的に低価格。新規ベンチャーの参入が容易で、価格は下がる傾向がある。	普通。論理合成や検証などを行うサービス料を合理的に価格に反映することが可能	比較的高い

1) ソフト IP コア

ソフト IP は、論理合成可能な RTL のプログラム・コードで提供される IP である。RTL コードから適宜、論理合成することで新たなテクノロジへ応用生成ができるというポータビリティ(portability)の高さが最大の長所である(注17)。この特徴をさらに先鋭化させるために、RTL よりさらに抽象化を進め C 言語等による提供を模索する動きもある。しかし RTL コードをゲート・レベル・ネットリストに変換する際には、利用するテクノロジに向けて高度に最適化させることが一般に求められる。この最適化作業を完全に自動化することは、完全な同期回路の状態遷移でさえ困難であるのが現状である。結局、システム LSI の最終的な品質は、RTL コードの中身とチップの仕様の両方を熟知した専門技術者の能力に依存することになる。この問題に応えるために米国の多くのソフト IP ベンダは、受託設計会社と提携し、IP の統合設計作業を任せている。

ソフト IP の将来的発展としては、一層上流にそしてオブジェクト指向に向かう動きもある。すなわち言語のモデルとして C 言語を利用してハード、ソフトを処理していく動きである。システムのアーキテクチャで勝負できると踏んだ IP ベンダは、シリコン・レベルのレイアウト設計まで責任を分担したくはない。RTL のプログラム・ソースに対して、それと等価な回路の C ソースの方が1/10のステップ数で記述可能であるからである。さらにビットの持つ情報量を‘0’、‘1’、‘L’、‘H’、‘X’、‘U’、‘Z’から、‘0’、‘1’のみを扱う事でコンパイラ(compiler:言語の情報処理システム)の負荷が約1/100となる。

負荷がこれほど低下させられる理由はビットの持つ情報量が大きいからである。実は論理値は論理‘0’と論理‘1’以外の意味を持つことができる。論理‘0’と論理‘1’はそれぞれ強さ弱さが規定できる。そして‘0’、‘1’以外の信号値も存在可能である。信号値‘X’、信号値‘Z’、信号値‘U’である。信号値‘X’は、論理‘0’と論理‘1’が衝突したときの不定値である。信号値‘Z’は孤立した接続状態であり、充電容量によって論理値が維持される。信号値‘U’は初期化されていない不定値である(Coelho, 1989)。これら情報量を取り除くことができるなら、コンパイラの負荷が下がる。これはレイアウト設計の重要性を除けば、これ以上の生産性を上げる LSI 設計は現在では実現できていない。

これは、あらゆるソフトウエアを知的財産ととらえ、これが LSI 開発工程へ介在することと、そしてソフトウエアとハードウエアとで設計人口の絶対的母数の違いを補完することにつながる重要な糸口である。つまり、ソフト IP のカテゴリとしては、LSI へ実装される C モデルや LSI 設計工程を支援するアプリケーション・ソフトウエアも加えるべきではないかと考えることである。この考え方はソフト IP の対象を LSI 設計上流ではなく LSI 設計全般、システム全般へパラダイム・シフトさせることにつながり、将来の半導体 IP の概念を変えるものとなる。そこはシステムに限りなく近く、システム・メーカの視点からは高付加価値が期待できる。システム・メーカにとっては、そこが強みを発揮できる領域である。

しかし反面、システム・メーカはシステム思考へ軸足を移せば移すほど、レイアウト設

計とは距離をおくことになる。つまり、RTL コードから論理合成し、レイアウト設計する工程はシステム思考による知識を用いることではない。

ソフト IP コアは、設計開発ツールに依存する傾向にある。後述する論理設計とレイアウト設計の反復修正作業で予想以上の設計時間になってしまう可能性がある。また最終的には、想定した仕様通りにならないこともある。この仕様とは半導体の性能であったり、半導体自身の原価であったりする。つまりシリコンへの最適化は低いと言える。

次にソフト IP コアのインテグレーション(integration:統合)化について述べる。既存のレイアウト環境をある程度自由に利用することができるので、全体最適には有利である。しかし全体最適の問題と扱う回路規模にはトレードオフが存在するので、大規模化には限界があると言える。そして設計者は論理設計とレイアウト設計の狭間で、論理回路設計と検証を徹底的に行う必要があり、多大な時間と経費が予測される。

ソフト IP 上で論理回路の手直しが必要な場合には、その手直し修正が最終的にレイアウト設計まで影響を及ぼす。つまり論理回路の手直しからレイアウト設計までの作業は負荷が非常に高い。

次にソフト IP に対するテスト容易化の設計について述べる。テスト容易化設計とは、回路のテストのコストを軽減するため、 回路自体をテストが容易になるように設計することである。ソフト IP の場合には、一つの固定したブロックにテスト回路の機能を置くことはできず、むしろそれは配置に左右される。つまりテスト回路がソフト IP の論理回路上の特性に影響を与えることがあり、注意を要する。

ソフト IP の競争性の度合いについては、村田 et al. (2001)により論じられている。ソフト IP はハードウエア記述言語による RTL 記述であるので、製造や半導体メーカのテクノロジに依存しない。従って参入障壁が非常に低い。ソフト IP の価格は、新規ベンチャーや大手半導体メーカが参入次第、下がっていく傾向にある。

2) ファーム IP コア

ファーム IP は、テクノロジに依存したゲート・レベル・ネットリストとして提供される IP である。論理構造が固定されるので、シリコンへの最適化がソフト IP に比べ進んでいる。従って手軽にシリコンのパフォーマンスやコスト概算を見積もることができる。また、一般的なシステム LSI ユーザにとって既存の設計フローを流用しやすいという特長も持っている。

半導体のナノテク(nano-technology)化が進んだ現在、ゲート・レベル・ネットリストを切り口にしても、後述するタイミング(timing)収束問題の解決は進展していない。それはネットリストを介して、上流設計とレイアウト設計間のイタレーション(iteration)が多くなってきているからである。イタレーションとは反復修正工程である。そのイタレーションが、設計の開発コストや設計の納期遅延を招来する設計負荷として認識されている。つまり場合によっては、RTL 設計までさかのぼり、RTL 設計の見直しと修正の対応を迫られる可能

性がある。

次にファーム IP コアのインテグレーション化について述べる。ソフト IP ほどではないが、既存のレイアウト環境をある程度自由に利用することができるので、全体最適にはやや有利である。また全体最適の問題と扱う回路規模にはトレードオフが存在するので、大規模化には限界がある。そして設計者は論理設計とレイアウト設計の狭間で、論理回路設計と検証を徹底的に行う必要があり、ある程度の時間と経費が予測される。ファーム IP 上で論理回路の手直しが必要な場合には、その手直し修正はレイアウト設計まで影響を及ぼし、設計負荷が高くなる可能性がある。

ファーム IP に対するテスト容易化の設計(注18)については、ソフト IP と同様、一つの固定したブロックにテスト回路の機能を置くことはできず、むしろそれは配置に左右される。つまりテスト回路がファーム IP の論理回路上の特性に影響を与えることがあり、注意を要する。

ファーム IP の競争性の度合いについても、村田 et al. (2001)が論じている。ファーム IP はテクノロジに依存している。従って参入障壁はソフト IP 程高くない。ファーム IP の価格には、論理合成や設計検証といったサービス金額を合理的に反映できる。従って、ソフト IP ほど低い価格とはならない。

3) ハード IP コア

ハード IP は、レイアウト設計成果として得られる設計知である。製造工程に受け渡しされる GDSII と呼ばれるフォーマットで提供される。性能・サイズ等は確定している上、信頼性が高い IP である(Hunt & Rowson, 1996)。従って、ソフト IP やファーム IP で提起される具現化の問題、そして設計の品質面での問題点が解決さている。ハード IP のコアは、利用するテクノロジに対して完全な最適化を図ることができる(注19)。また、そのハード IP を利用してシステム LSI の統合設計を行う際には、レイアウト設計を意識して行うことができるので利便性がある。更に、異なるシリコン・ファンダリに簡単に移植できないというテクノロジ依存性が、逆に著作権のコピー防止になっているという側面もある(注20)。ハード IP の課題は、性能面ではなく、IP ベンダやユーザ双方にチップのレイアウト設計を行える技術力が要求される点にあると言える。

次にハード IP コアのインテグレーション化について述べる。ソフト IP やファーム IP と比べると、既存のレイアウト環境を利用することができない。従って全体最適にはやや不利である。むしろハード IP 自身が最適化されているのであるので、レイアウト全体から見れば部分最適に有利である。半導体商品には、基本回路はそのままに、機能の一部追加や改善が行われる場合がある。この様な部分修正が必要な箇所に対しては、ハード IP で反映を行えば対処できる。ただしその際には、ハード IP を積木の様に集積させる設計技術が要求される。そのツール技術の確立によって、作業量は極めて小さくなる可能性がある。なぜならばハード IP の集積化は容易になるので、大規模回路にも対処できるからである。

そして設計者は論理設計とレイアウト設計の狭間で、論理回路設計と検証を行う必要がなくなるので、無駄な時間と経費を削除することも可能である。ハード IP 上で回路の手直しが必要な場合には、そのハード IP 部分だけを取り出して修正するか又は交換するだけで容易に対処できる。つまりレイアウト設計領域だけで対処できるので、ソフト IP やファーム IP 程、設計負荷はかからない。

ハード IP に対するテスト容易化の設計については、ソフト IP やファーム IP とは違って、一つの固定したブロックとしてテスト回路の機能を置くことができる。つまりテスト回路がハード IP の回路上の特性に影響を与えることはない。

ハード IP の競争性の度合いについても、村田 et al. (2001)が論じている。半導体メーカではハード IP コアのデータベース、すなわちライブラリ(library)を揃えて自社の強みにしている企業がある。たとえば米国テキサス・インスツルメント（Texas Instruments）社における信号処理プロセッサでは、システム LSI への組み込み化戦略により、自社のシステム LSI の付加価値向上並びにブランド価値創造に貢献している。半導体のレイアウト設計に関する技術力は企業価値を大きくしている。その理由はプロセス技術の進歩により LSI 実現の困難性が設計に帰着しているからである。その設計領域の中でもレイアウト設計に集中しつつあるからである。その事がハード IP の価値を上げている要因の一つである。

8.3.4 半導体 IP コアの流通と普及

システム LSI 設計者は上述で述べた IP コアの重要性を認識し、その拡張を進める方向にある(注21)。コアはそれ自体で一つのまとまった機能を有する LSI の構成回路要素であり、その種類も多様化している(注22)。また昨今ではコアの流通は企業内部のみならず、企業間でコアを調達する動きも見られる(注23)。こういったコアを重視する姿勢は国籍を越えて各企業によって異なるが、統一的な見解としてはコアを外部調達する方向にある（平嶋, 1998）。

システム LSI はチップ・レベルのコアが集積されたものである。しかしチップ・レベルで他社のコアを組み合わせるだけでは設計できない。それは製造技術をはじめ技術的諸元が各社によって大きくことなるためである。従ってシステム LSI 上に各構成要素を組み込むためのコアを社内に豊富に供えている場合は別段であるが、そうでない場合には必要なコアそのものが不足する可能性がある(注24)。

上記の様な問題を解決するために各開発者が保有しているコアを流通させ、相互に取り引きしようとする動きが見られる様になってきた(注25)。この場合には技術的見地からのコアのインタフェースの標準化が必要となる(注26)。それは各コアの基本的な物理的仕様とインタフェース・プロトコル等である。これが実現されるとコアというより、モジュール（Baldwin & Clark, 2000）という表現がふさわしいと言える。

次にコアの普及と定着の違いについて述べる。日本の半導体設計においては、ハード IP コアはソフト IP コアに比べて半導体設計者への普及と定着の度合いは低いと想定されている（平嶋, 1998）。その理由の一つ目はハード IP コアの特性をソフト IP コアほど理解してい

ないために、合理的な設計思想が定着していないからである。二つ目としては、ハード IP コアに関する設計知識の粘着性は高いので組織内の世代を越えた継承がされにくいからである。これがハード IP コアとソフト IP コアとで設計人口の絶対的母数の違いを生じさせている。

8.4　手戻り修正による設計負荷の問題と解決策

8.4.1 手戻り修正による設計負荷の問題

半導体の微細化がさらに進み、回路が大規模化・複雑化している。そしてこういった進展過程の中で、手戻り修正による設計負荷の問題が発生してきている。

問題の原因の一つは、微細化により回路内の遅延の支配要因がゲートの素子遅延から配線遅延にシフトしてきたことにある（吉田, 2003）(注27)。その結果、ユーザ側のシステム設計者が想定した配線遅延と、半導体メーカのレイアウト設計成果の配線遅延が大きく乖離するようになった。つまり、半導体メーカだけでは、配線遅延によって直接影響を及ぼす半導体のタイミング仕様を満たすことが難しくなってきた。そしてレイアウト設計が目標仕様に適合しない場合には、システム設計者にネットリストを戻し修正依頼することになる。こういった手戻り作業が頻繁に発生する様になり、ついには何度直しても配線遅延の目標まで収束しないという事態が起きるようになってきた。これをタイミング収束と言う。

実際、大手のシステム・メーカにおけるデジタル・カメラの設計グループでは、0.13ミクロンのデザイン・プロセスで2百万ゲート規模のシステム LSI 開発の際、レイアウト設計が終結しないという事例が報告されている。

設計負荷を低減させるためには、タイミング収束を設計開発工程でいかに解決するかということである。タイミング収束の問題を本質的に解決するには、従来の ASIC（Application Specific Integrated Circuits:特定顧客用途向け半導体）の設計手法を見直して、システム・ユーザがレイアウト設計まで立ち入ることが必要と考えられる。

米国のファブレス(fabless)半導体や IP ベンダは、自らレイアウト設計まで自己責任において行うことが一般的となっている。一方、日本では ASIC の顧客はシステムのアーキテクチャの構想から論理設計までを行うことが一般的である。そして半導体メーカはレイアウト設計・開発を受け持ち、互いの領域には立ち入らない、という設計文化が定着してきた。その二分化により、不測の事態が生じても責任が明確化されてきた。しかしこのタイミング収束は、こういった二分化した設計では解決できない。そして設計コストの増大と設計の遅れを助長することにつながる。

次項では従来の ASIC 設計手法を含めた一般的なレイアウト手法の手順を明らかにし、手戻り修正の発生のメカニズムを捉える。そして本問題解決に向けた手がかりを探る。

8.4.2 手戻り修正を引き起こす仕組み

1) 従来の一般的なレイアウト設計手法

従来のレイアウト設計手法の手順について述べる。その手順を図8.3に示す。この手順から設計負荷を引き起こす因子を抽出する。

手順①　最上位となるチップ構造設計

ソフト IP コア、ハード IP の選定を行う。必要に応じて機能変更、追加、新規回路部分のネットリスト作成まで行う。

手順②　ブロック・マニュアル配置とフロアプランの絞込

各ハード IP の配置を基本的には手作業で決定する。画面上に表示される概略配置とラッツ(rats:配線要求線) (注28)などの情報を参照し、経験則（ハード IP につながる配線の最短化、未配置部分をなるべく固めて残す等）によって配置を決定する。そして、チップ全体に経験的な幅で電源幹線を配線する。ソフト IP は面積をある程度の精度で算出しておき、ブロック配置の段階で形状および配置を決定する場合と、この段階では取り扱わずに、次の段階で新規回路部分と混合して配置配線する場合がある(注29)。

手順③　セル配置配線、詳細配線と最適化

ブロック配置によって残されたチップ上の未配置領域に、経験的なセル配置使用率設定を行い、新規回路部分（及びソフト IP）の論理階層を崩してフラット(flat:階層構造をなくす）)化された大規模なネットリストを一括配置配線する。設計作業は、クロック・スキュー(clock skew:同期式設計においてクロックの伝搬遅延時間の差、配線容量などの理由により発生するタイミングずれ)の最小化(注30)、タイミング・ドリブン(timing driven)(注31)配置、タイミング・ドリブン配線（配線幅調整やバッファ(buffer)挿入）、などのステップに分かれる。各ステップにおいてバッファ類の挿入を受けた配置配線の部分的修正、タイミング確認とその結果に応じたやり直し、未配線の確認と再配線あるいは人手修正等の、手作業を介した繰り返し作業を行う。

手順④　最終検証

DRC (Design Rule Check)、LVS (Layout vs. Schematic)、タイミング検証等最終的な検査を行う。DRC と LVS は回路の検証作業である。DRC は、レイアウト後の配線パターンの距離が基準値を保っているか、部品同士の重なり/高さ制限は守られているか、配線パターンとネットリストの整合性はとれているか等の設計規定をコンピュータで検証することである(注32)。LVS は出力したレイアウト図と回路情報ファイルを比較し、配線や接続情報が正しいかを調べる作業である(注33)。出力したレイアウトが実際基準値を満たして

いるかどうかを調べる DRC とはこの点において違う。

2) 手戻り修正を引き起こす因子

上述の従来の設計手法を観察すると、設計負荷につながる因子が三点発見できる。

第一点目の因子は手順②において、経験則による人手設計に大きく依存していることである。過去にはブロック数はさほど増加しないので、人手で構わないと言われた時期もあった。しかし ASSP (Application Specific Standard Processors:特定用途向け標準品)分野におけるメモリー・ブロック数の増大、そしてアナログ・ブロックとの共存など、ブロック数の増加傾向が明確になって来ている。そして電源に関する考慮も経験則に頼ってきた。つまり、IP コアを用いた設計においては、設計初期の段階で経験則に依存しないチップのサイズを予測する事と、自動化が必須課題となっていることを示唆している。

第二点目に、手順③において大規模なフラットなネットリストを一括して配置配線することは、配線を収容しかつ適切なバッファ挿入によりタイミングを満たす様にセルを配置することと同意である。つまり、この手順は、全体最適を意識しながら配置と配線という二律背反問題を引きずっている。大規模・複雑化が進行した中で、二律背反の問題を一括して解くことであるので、配置修正と配線確認の繰り返し回数が急激に増加する。結果として、大規模な回路を全体最適として処理しようとする為に、目標設計期間までに終了しないというリスクが指摘される。

第三点目に、設計負荷につながる二つの手順②と手順③が複合している点である。セルの配置配線という手順で最善を尽くしたとしても、タイミング性能を満足できない場合には手戻り修正を行う。再三、ブロック配置あるいはそれ以前の段階に戻り、修正を加えてやり直しても、結果は改善しないという悪循環のリスクがある。これは最悪、手戻り修正が永久に続き、プロジェクトが終了しないことにつながる。

これはブロック配置の段階において、新規回路の内部を無視していることや、フラットに全てのタイミング、配線長、面積等の見積もりを試みる為に、実現や最適化を困難にしていることが主の原因である。

上記の指摘を整理すると、手戻り修正を引き起こす因子は二つある。経験則による人手設計とセルの配置と配線による二律背反である。そしてこの二つの因子を含む手順と大規模回路の全体最適化が複合していることも仕組みを複雑化させている。これらの因子と仕組みが、手戻り修正のプロセスを経て、設計コストの増大化と設計遅延を引き起こしている。

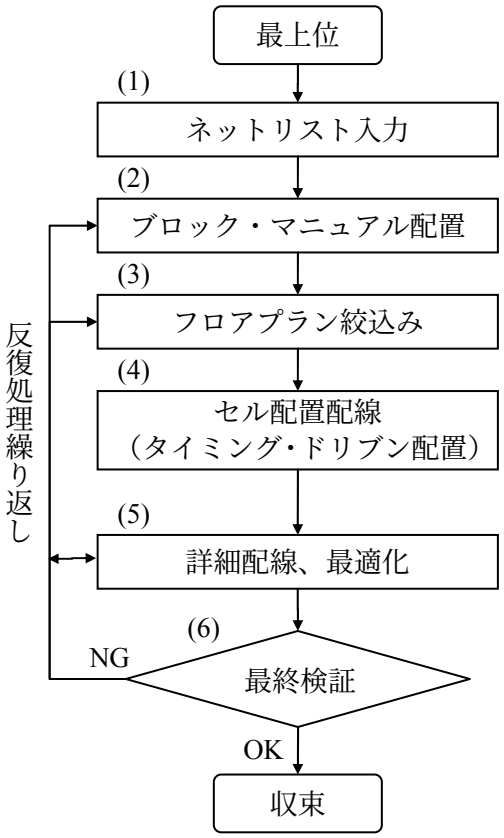


図 8.3　従来の一般的なレイアウト手法

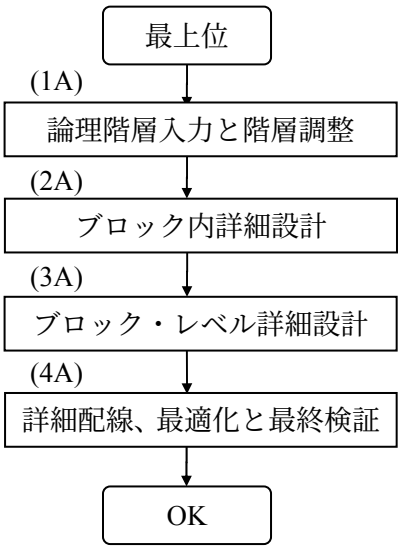


図 8.4　ハード IP の多用と自動レイアウト手法

8.4.3 設計負荷を減少させる一手法

1) ハード IP の利用

システム LSI 設計の近時の問題は、製造並びに生産工程上の問題を除けば、上述の通り手戻り修正による設計負荷が大きくなっていることである。前項ではその手戻り修正を引き起こす因子と仕組みを明らかにした。この因子とは経験則による人手設計、そしてセルの配置と配線による二律背反である。これらを取り除くことができれば、手戻り修正の工程を減少させることができる。

本章で提案する手段は、まずハード IP を利用することと、次にハード IP を多用する為の自動レイアウト手法を構築することにある。ハード IP は本章で示した体系化の中で位置付けられている通り、レイアウト設計成果から得られるものである。従って配置と配線による二律背反は、IP というサブシステムとしては解決積みである。そしてサブシステム間の統合作業については、ハード IP を多用した手法を自動化させておけば、経験則による人手設計を回避できる。

2) ハード IP を多用したレイアウト設計手法

手戻り修正の問題を解決し、安定した品質で短期間に設計するための手法(Murata et al., 1996)を提案する。本手法はハード IP を多用する前提にたち、ハード IP を多用する為の自動レイアウトのツールを利用するものである。図 8.4 に本設計手法の概念を流れ図によって示す。

手順①　最上位となるチップ構造設計

ハード IP（レイアウトまで設計済みの回路ブロック）、ソフト IP（ネットリストあるいは RTL まで設計済みの機能回路ブロック）の選定、機能変更、追加、新規回路部分の RTL 設計あるいはネットリスト作成まで行う。

手順②　論理階層入力と階層調整

論理設計の階層の情報を活用し、新規設計の部分を適当な規模(物理ブロック)に分解し、各物理ブロックを自動配置する。具体的には、タイミング制約の充足、そのためのリピータ(repeater)挿入予測(配線長の 2 乗に比例して遅延は増大する、その為にリピータを挿入して高速化を図る)、クロック・バッファ挿入予測（クロック・スキューを最小化する）、配線の収容性、電源配線領域の確保、デッド・スペース（利用価値がないまたは利用出来ない場所）の最小化、等を考慮して、各ブロックの形状、概略配置、端子位置、タイミング制約の割り当て等を行う。この段階でほぼ配線可能なチップサイズの予測が可能となる。

手順③　ブロック内詳細設計

各ソフト・ブロックについて、ブロック内部のセル・ベース配置配線が行われる。すなわちタイミング制約の充足を制約とし、ブロック形状や端子位置等についての前ステップでの見積りに従い、面積と配線長を最小化するように、セル・ベースで配置配線を行う。ブロックによってはトランジスタ・レベルでの最適化を行う場合もある。またメモリー・ブロック、データ・パス(data path: デジタル・データの処理をする マルチ・ビット・バスの構造である。データ・パスには高度な算術演算を行うものが多い)などの特定用途ブロックはそのための専用アルゴリズムで詳細化される。詳細配線においては、アンテナ(antenna)効果（アンテナ効果とは、 プラズマを用いた製造工程において配線にチャージされた電荷によりゲート酸化膜がダメージを受け、 トランジスタの歩留まりや信頼性が劣化する現象）回避、クロストーク(cross-talk: 平行して配線された信号が、不必要な結合を起こしてしまう現象)回避、エレクトロ･マイグレーション(electro-migration: 金属配線を構成する金属原子が、電子との衝突により移動する現象)回避等、製造性向上とシグナル・インテグリティ(signal-integrity: 信号波形のアナログ的健全性の確保)確保を目的とした処理が必要となる。

手順④　ブロック・レベル詳細設計

タイミング制約の充足、電源電圧の確保、ブロック間配線の収容性などについてより高精度な解析を行う。そしてブロック配置の調整を行ってからブロック間詳細配線を行う。

手順⑤　詳細配線、最適化と最終検証

LVS、DRC、タイミング検証、電源解析、クロストーク、エレクトロ・マイグレーション等の最終的な検査および確認を行う。

上記の手順を整理すると、本手法を現実のものとするためには、回路分割、ブロック自動配置、段階的なタイミングと配線収容性の見積もり精密化手法等を解決したツール技術が求められるのである。本手法をサポートしたツールは、現在まで研究開発が行われている（Murata et al., 1996）(注 34)。

8.4.4　従来手法との比較

本手法は従来手法とは違って、配置と配線という二律背反問題は生じない。従って図 8.3 で見られる様な反復処理の繰り返しは生じない。そして本手法は設計済みのハード IP を多用すればする程、大規模回路にも対応できる。

図 8.5 に従来のレイアウト設計手法と本技術手法それぞれによって生成されるレイアウト結果を対比した。IP-A、IP-F、IP-H はハード IP であり、それ以外はソフト IP である。従

来技術ではソフト IP 群を一括問題として処理する際に、階層構造は全て解放され、階層がない一つの構造体となる。従来技術のレイアウト図で見られるハード IP 群以外の領域が、その構造体である。本技術のレイアウト図で見られる通り、ハード IP とソフト IP は全て階層が保存されており、各コアが部分最適された状態でレイアウトが構成される。

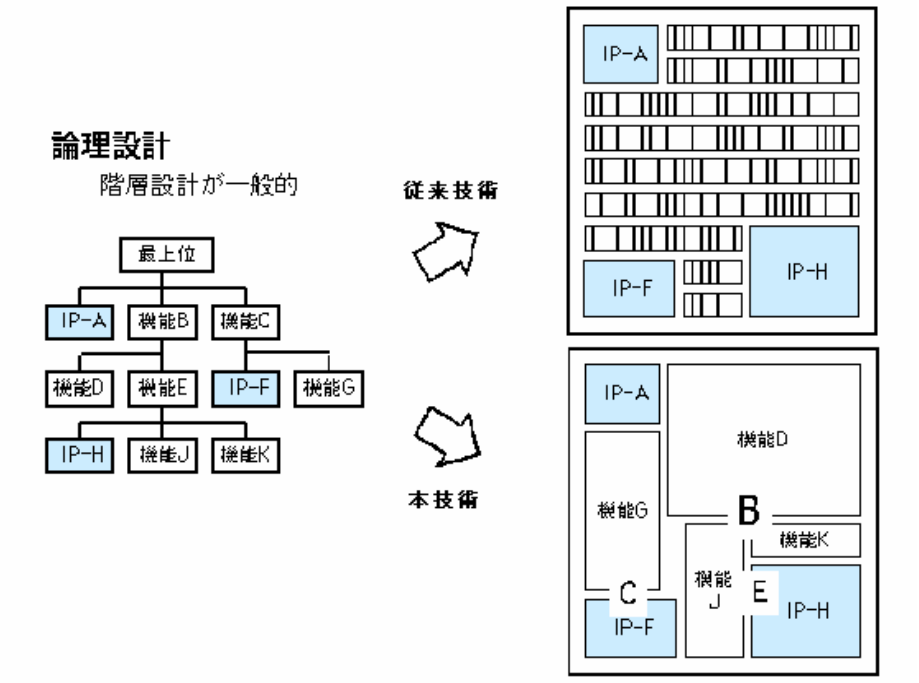


図 8.5　従来技術と自動配置技術によるレイアウト手法

8.5　半導体 IP コアの管理と活用

8.5.1 知識ベースによる設計過程の効率化とリスク低減

本章で議論した IP コアは設計対象物を構築する上で活用される概念である。そしてその概念は、設計の抽象性レベルの違いによって体系付けされた。その体系付けに基づいて構築される知識ベースは、従来の知識ベースが抱える共有・再利用性の欠如という問題点の解消に貢献すると言える。なぜならばこの知識ベースが定着すれば共有化が進み、設計過程において新規からの設計作業を減らす可能性を持つ。そして再利用化が進めば、設計開発コストの低減と設計開発時間の短縮につながるからである。知識を体系化し活用する知識ベースは設計活動の効率向上に寄与する。

知識ベースは設計活動を効率的にするだけではない。設計品質の安定性確保である。IP コアは共有化と再利用を通じて最適化される作業が継続して行われるので、機能や性能面の設計品質が安定してくる。これはロバスト(robust)設計である。このロバスト設計ができておれば、設計開発工程のプロジェクト・リスクを低減することが可能となる。このプロジェクト・リスクとは、設計開発の所要コストの増大、そしてスケジュールの遅延である。大規模設計の様に扱う知識の量が膨大で、複数の設計者が介在する場合においては、設計品質を確保する手段として体系化された知識支援の効果は大きいと言える。

8.5.2 ソフト IP コアとハード IP コアが商品開発に及ぼす特性

半導体設計におけるコアは設計情報と設計知識の著作物であり、半導体 IP (Intellectual Property)ある。この IP コアとは半導体と言う人工システムを構築する際のビルディング・ブロックとして用いられる概念である。コア間のインタフェースは必ずしも標準化しているとは言えないので、モジュールと完全に同一視することは難しい。半導体の IP コアは、本章で述べた通り体系付けされている。そして三つの種類のコアに分類できる。ソフト IP コア、ファーム IP コア、そしてハード IP コアである。

これらのコアの概念を一般設計学による抽象概念と実体概念という対極的視座で注視するなら、ソフト IP コアは抽象概念に、そしてハード IP コアは実体概念に位置する。つまりソフト IP コアは人工物の特徴に対応してつくられた抽象概念に近い存在である。そしてハード IP コアは現実に存在する人工物、すなわち実体の存在に近い。尚、ファーム IP は二つの概念の特徴を兼ね備えた中庸のコアである。それはソフト IP とハード IP の間に体系上、位置付けられている。

次にソフト IP コアとハード IP コアがそれぞれ商品開発に及ぼす特性の違いについて考察する。

まずソフト IP コアについて述べる。ソフト IP コア自身は内部構造が最適化されていない。ソフト IP コアは複数集まって一つのシステムとして統合されると、一括問題としてシステ

ム全体の最適化が行われる。統合された時点ではシステムは階層構造を保っている。しかし一括問題として処理する際に、階層構造は解放され階層がない構造体となる。もはやサブシステム、すなわち個々のソフト IP コアが認識できない一つのシステムとして最適化が行われ、構築される。従ってソフト IP コアを利用する場合には、全体最適を意識したシステム構築に有効である。新商品開発やリエンジニアリングを想定した商品開発には、ソフト IP コアの利用効果は高いと言える。しかしソフト IP コアの欠点は、従来の一般的レイアウト設計手法で述べた通り、ソフト IP コアが集積した大規模回路の処理には適さないということである。

次にハード IP コアに関して述べる。ハード IP コアは内部構造が最適化されている。商品の中には、商品のサイクル毎に標準プラットフォームを固定させながら、性能の向上、機能の追加と改善が継続して行われることがある。つまりバージョンアップ、グレードアップという部分修正(incremental design changes)に対応した商品造りが要求されることがある。この様な部分修正が必要な箇所に対しては、ハード IP コアで部分的な反映をさせる。部分修正は、それに伴う開発コストと開発期間を最小化させることができる。従って、こういった部分最適を意識したシステム構築に対しては、ハード IP コアの利用が有効と言える。そして本章で提案したレイアウト手法を用いるなら、ハード IP コアの集積化は容易なので、大規模回路に適している。

上記を簡単にまとめると、ソフト IP コアの集積化はその全体回路の規模で限界を持つが、半導体の全体最適化を考慮した商品開発には有効である。そしてハード IP コアは部分修正が必要な半導体の商品の開発に有効であり、大規模回路に適している。

8.5.3　ハード IP を多用したレイアウト設計手法による設計負荷の低減

大規模なシステムを設計する時には、トップダウンとボトムアップ手法で行うことができる。つまりシステムを複数のサブシステムとして機能分解し、サブシステムの規模を小さくする。そしてそれぞれのサブシステムの設計を行った後に、それらを統合するアプローチである。もしサブシステムが事前に設計済みのコアとして準備され、サブシステム同士のインタフェースが一定の基準で規定されているのであれば、コアがない場合よりシステム構築が容易である。昨今大規模・複雑化している半導体設計はその例証である。しかしながら近時の規模的拡大の要求は手戻り修正による設計負荷を招来している。そして設計開発コストと期間にインパクトを与えている。

設計負荷を低減させる策の一つはレイアウト設計手法の変革である。従来の一般的なレイアウト設計手法ではその中でソフト IP コアを多用することを前提としている。そしてソフト IP コアを多用していることが、設計の手戻り修正を引き起こし、設計負荷を大きくしている。

従来手法の手順を解析した結果、手戻り修正を引き起こす因子は二つ確認された。経験則による人手設計とセルの配置と配線による二律背反である。そしてこの二つの因子を含

む手順と大規模回路の全体最適化が複合していることも仕組みを複雑化させていた。これらの因子と仕組みが、手戻り修正のプロセスを経て、設計コストの増大化と設計遅延を引き起こしていた。従ってこの因子を取り除くことができれば、手戻り修正の工程を減少させることができる。

本章で提案した手法は、まずハード IP を利用することと、次にハード IP を多用する為の自動レイアウト手法を構築することにある。ハード IP は本章の体系化の中で位置付けられている通り、レイアウト設計成果から得られるものである。従って配置と配線による二律背反は、IP というサブシステムとしては解決積みである。そしてサブシステム間の統合作業については、ハード IP を多用した手法を自動化させておけば、経験則による人手設計を回避できる。ここでハード IP コアを多用できるレイアウト・ツールが確立し、ハード IP コアを多用できるのであれば設計負荷を低減できる。

8.5.4 今後の課題

本章では半導体設計を対象に論じたが、半導体という対象を離れて昇華できるかが一つの課題である。その為には半導体以外にあてはまる領域又は商品群において、多くの事例を設定し検証する必要があると言える。

8.6 本章のまとめ

半導体の製造は資本集約型事業となっている。この進展過程の中で、半導体の付加価値の源泉は設計知に移ってきている。この設計知とは、設計の情報と知識である。本章ではまず半導体の設計知を形成させる設計手法の変遷を追った。そしてこの設計手法の変遷は、設計対象の規模的拡大化、並びに設計の抽象性レベルの高度化に起因していることを確認した。

次に設計手法の変遷過程を経て定着してきた設計知の概念を、設計の抽象性レベルの違いによって体系化した。この概念は設計知を著作物とさせた知的財産、すなわち半導体 IP (Intellectual Property)である。抽象概念に近いソフト IP コア、人工物に近い実体概念であるハード IP コア、そしてその二つの中庸に位置するファーム IP コアが含まれる。

この概念と体系付けに基づいて構築される知識ベースは、従来の知識ベースが抱える共有・再利用性の欠如という問題点の解消に貢献すると言える。なぜならばこの知識ベースが定着すれば共有化が進み、設計過程において新規からの設計作業を減らす可能性を持つ。そして再利用化が進めば、設計開発コストの低減と設計開発時間の短縮につながるからである。また、知識ベースの活用は共有化と再利用を通じて継続されるので、機能や性能面の設計品質を安定させると言える。つまりロバスト設計である。ロバスト設計ができておれば、設計開発工程のプロジェクト・リスクを低減することが可能となる。大規模設計の様に扱う知識の量が膨大で、複数の設計者が介在する場合においては、設計品質を確保す

る手段として体系化された知識支援の効果は大きいと言える。以上の通り、製品開発における知識ベースの構築は構築しない場合と比して、設計プロセスの効率化と設計品質向上の点で有益であることが確認された。

次にソフト IP コアとハード IP コアの商品開発に及ぼす特性についてまとめた。従来の半導体の設計開発手法は、ソフト IP コアを多用した全体最適を意識したアプローチである。ソフト IP コアを集積化すればするほど、全体の最適化問題を解くことが困難になる。なぜならば設計の規模と全体最適は二律背反の関係にあるからである。従ってソフト IP コアによるシステムの構築規模には限界がある。設計開発工程における手戻り修正は、規模的拡大化の状態で全体最適化を求めるために生じる副次的産物とも言える。

そして半導体の設計プロセスの近時の変容を述べ、設計開発工程における手戻り修正の問題を指摘した。この手戻り修正が頻繁に生じると、設計開発コストと期間に大きな負荷を与える。この設計負荷を低減させるための一手法として、設計知を利用するアプローチを提案した。ハード IP を多用するレイアウト設計手法である。ハード IP コア自身は内部構造が最適化されている。そしてハード IP コアであるが故に手戻り修正の問題は発生しなくなる。つまり、部分最適化されたハード IP コアを積み重ねる集積化は容易であり、部分最適を意識した大規模なシステム構築に適している。従って規模的拡大が進む半導体システムにおける手戻り修正による設計負荷の低減に関しては、本手法は従来技法より効果は高い。

注

(1) “ムーアの法則 (Moore’s Law):エレクトロニクス産業を牽引してきた道標”,日経エレクトロニクス5月23日号, p. 12 (2005)
(2) Collett, Ronald E.: “特集 EDA　1部　産業展望”,日経エレクトロニクス, 673, pp. 76-99 (1996)
(3) 藤波義忠,吉田正昭,石塚昭夫,恵谷誠至: “特集 EDA 2部　ツール辞典”,日経エレクトロニクス, 673, pp. 100-139 (1996)
(4) 小林優: “HDL 設計のすすめ”, Design Wave Magazine, 1, pp. 88-92 (1996)
(5) 中山俊一,小島郁太郎: “HDL 本格導入が回路設計工程の変革迫る－浸透度を読者アンケート”,日経エレクトロニクス, 623, pp. 75-95 (1994)
(6) 長谷川裕恭: “VHDL と Verilog－HDL の比較”, Design Wave Magazine, 1, pp. 93-99 (1996)
(7) 桜井至: “HDL によるデジタル設計の基礎”,テクノプレス, p. 25 (1997)
(8) 倉重克己: “論理合成ツール”, Design Wave Magazine, 2, pp. 64-66 (1996)
(9) “特集　パソコン用 EDA ツール総解説と総覧”, Design Wave Magazine, 2, pp. 52-68 (1996)
(10)佐々木元,森野明彦,鈴木敏夫: “LSI 設計入門:コンピュータサイエンス大学講座18”,近代科学社 (1987)

(11)中山俊一: “コアの調達経路拡大が大規模 ASIC を支える”,日経エレクトロニクス, 657, p. 108 (1996)
(12)“VSI アライアンス”, Design Wave Magazine, 7, pp. 32-33 (1997)
(13)中山俊一: “コアの調達経路拡大が大規模 ASIC を支える”,日経エレクトロニクス, 657, p. 109(1996)
(14)伊藤元昭,藍田宏樹: “市場創造型へ戦略転換進める半導体メーカ”,日経エレクトロニクス, 716, pp. 123-146 (1998)
(15)“システム LSI　これが日本の生きる道”,週刊東洋経済,1月18日号, pp. 33-35 (1997)
(16)中山俊一: “コアの調達経路拡大が大規模 ASIC を支える”,日経エレクトロニクス, 657, p. 112 (1996)
(17)伊藤元昭: “第2フェーズに日合った VSI,現実的なコアの流通方法を模索”,日経エレクトロニクス, 701, p. 44 (1997)
(18)鈴木五郎: “システム LSI 設計入門”,コロナ社, pp. 136-148 (2003)（量産される LSI はテスタによって良否の判断が行われる。このコストは大きい。その為にテストが容易になる回路構成にし、短時間でテストを可能にできるならばコストを削減できる。テストが容易になる回路構成の設計をテスト容易化設計と呼ぶ。）
(19)伊藤元昭: “第2フェーズに日合った VSI,現実的なコアの流通方法を模索”,日経エレクトロニクス, 701, p. 46 (1997)
(20)飯塚哲哉: “IP 時代のシステム LSI 設計開発とデザインハウスの役割”,電子材料, 37, １, pp. 59-63 (1998)
(21)中山俊一: “コアの調達経路拡大が大規模 ASIC を支える”,日経エレクトロニクス, 657, p. 118 (1996)
(22)中山俊一: “コアの調達経路拡大が大規模 ASIC を支える”,日経エレクトロニクス, 657, p. 109 (1996)
(23)中山俊一: “コアの調達経路拡大が大規模 ASIC を支える”,日経エレクトロニクス, 657, pp. 110-111 (1996)
(24)柴下哲: “DAC に見る EDA 最新トピックスとテクノロジ”, Design Wave Magazine, 5, pp. 64-75 (1996)
(25)“VSI アライアンス”, Design Wave Magazine, 7, pp. 32-33 (1997)
(26)柴下哲: “DAC に見る EDA 最新トピックスとテクノロジ”, Design Wave Magazine, 5, pp. 66-67 (1996)
(27)伊藤元昭: “第2フェーズに日合った VSI,現実的なコアの流通方法を模索”,日経エレクトロニクス, 701, pp. 44-45 (1997)
(28)鈴木五郎: “システム LSI 設計入門”,コロナ社, p. 104 (2003)
(29)鈴木五郎: “システム LSI 設計入門”,コロナ社, pp. 103-107 (2003)
(30)鈴木五郎: “システム LSI 設計入門”,コロナ社, p. 132 (2003)

(31)鈴木五郎: “システム LSI 設計入門”,コロナ社, pp. 106-107(2003)
(32)鈴木五郎: “システム LSI 設計入門”,コロナ社, pp. 108-114(2003)
(33)鈴木五郎: “システム LSI 設計入門”,コロナ社, pp. 114-119(2003)
(34)“次世代の SoC 物理設計手法－今日のレイアウト・デザイン手法と次世代のレイアウト・デザイン手法について”,エレクトロニクス・アップデート, 12月号, Vol. 17, No. 188, pp. 11-14 (2001)

参考文献

[1] 青島矢一,延岡健太郎:“プロジェクト知識のマネジメント”,組織学会, Vol. 31, No. 1, pp. 26-27（1997）
[2] Armstrong, James R.: “Chip-level modeling with VHDL”, Kuluwer Academic (1989) (ジェームス・R.・アームストロング（白石繁,山本靖訳）: “VHDL デザイン・テクニック: 標準ハードウエア記述言語とチップ・レベル・モデリング”,電波新聞社 (1990))
[3] Baily, M. C., Zizcwitz, E., Bosworth, B., and Wcstphal, L. E.: “Extending the East Asian Miracle Microeconomic Evidence From Korea”, Brookings Papers on Economic Activity, Microeconomics, Vol. 1998, pp. 249-321 (1998)
[4] Baldwin, C. Y. and Clark, K. B: “Design Rules: The Power of Modularity”, Cambridge, MA:MIT Press (2000)（安藤晴彦訳: “デザインルール:モジュール化パワー”,東洋経済新報社, p. 85 (2004)）
[5] Coelho, David R.: “The VHDL Handbook”, Kluwer Academic Publishers (1989)
[6] 電気通信学会編: “LSI ハンドブック”,オーム社, pp. 161-162 (1984)
[7] Fujimoto, T. and Clark, K. B.: “Product Development Performance: Strategy, Organization, and Management in the World Auto Industry”, Boston, MA: Harvard University Press (1991) (田村明比古訳: “実証研究　製品開発力―日米欧自動車メーカ20社の詳細調査”,ダイヤモンド社 （1993）)
[8] 藤本隆宏,青島矢一,武石彰編: “ビジネス・アーキテクチャ製品・組織・プロセスの戦略的設計” (2001)
[9] 藤本隆宏: “ビジネス・アーキテクチャ”,有斐閣 (2002)
[10] 藤村修三: “半導体立国ふたたび”,日刊工業新聞社 (2000)
[11] 平嶋竜太: “システム LSI の保護法制:設計情報の流通と法的保護”, 信山社 (1998)
[12] Hunt, Merril and Rowson, James A.: “Blocking in a system on a chip”, IEEE SPECTRUM November, pp. 35-41 (1996)
[13] 伊丹敬之: “日本の半導体産業なぜ「三つの逆転」は起こったか”,NTT 出版 (1995)
[14] 加藤浩太郎編: “LSI 技術の基礎”,電気通信協会, pp. 132-233 (1992)
[15] 小林優: “HDL 設計のすすめ・Design Wave Magazine”,CQ 出版社, p.88 (1996)

[16] Mead, Carver and Conway, Lynn: “Introduction to VLSI Systems”, Addison-Wesley Longman Publishing (1979) (菅野卓雄,榊裕之監訳): “超 LSI システム入門”,培風館 (1981))

[17] 三輪晴治: “日本の半導体・自動車産業は復権するのか”,工業調査会 (1999)

[18] 茂木晃編: “電気電子用語大辞典:物理的回路構造において設計の基準となる最小寸法”, p. 853 (1992)

[19] Moore, Gordon E.: “Cramming More Components Onto Integrated Circuits”, Electronics, Vol. 38, No. 8 (1965)

[20] Moore, Gordon E.: “Lithography and the Future of Moore's Law”, Proc. SPIE, Vol. 2437, May (1995)

[21] Murata Hiroshi, Fujiyoshi Kunihiro, Nakatake Shigetoshi and Kajitani Yoji: “VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair”, IEEE Transactions on CAD of IC and Systems, December, pp. 1518-1524 (1996)

[22] 村田洋,安達文勝,山本靖: “半導体 IP と LSI の物理設計技術力”,日本半導体ベンチャー協会誌,10月号, pp. 10-11 (2001)

[23] 室賀三郎（渡辺誠,菅野卓雄監訳）: “VLSI システム設計”,ワイリー・ジャパン, pp. 412-417 (1984)

[24] 野中郁次郎,竹内弘高: “知識創造企業”,東洋経済新報社 (2002)

[25] OhUallachain, Breandan: “Restructuring the American Semiconductor Industry:Vertical Integration of Design Houses and Wafer Fabricators”, Annals of the Association of American Geographers, Vol. 87, No. 2, pp. 217-237 (1997)

[26] 小川進: “イノベーションの発生論理”,千倉書房 (2001)

[27] Polanyi, Michael（高橋勇夫訳）: “暗黙知の次元”,ちくま学芸書房 (2003)

[28] 谷光太郎: “半導体産業の系譜　巨大産業を築いた開拓者たち”,日刊工業新聞社 (1999)

[29] 吉川弘之: “一般設計学”,機械研究, 37, 1 (1985）

[30] 吉田憲司: “IC の設計と製造のはざまで”,システム LSI 設計技術研究会,情報処理学会, SLDM-112 (1) (2003)

第9章　　設計プロセスにおける設計知の多様化

9.1　　半導体産業におけるビジネス・アーキテクチャの展開

ブロードバンド時代の到来といわれ、コンピュータ、通信、民生機器はもとより、マルチメディアに必要なモバイル・コンピューティング、携帯用情報通信端末といった情報通信機器などの普及にともなって半導体業界は大きく成長を遂げた。欧米のみならず、台湾、韓国、シンガポール、イスラエル、マレーシア、そして中国といった地域の政府が優遇税制のインセンティブや直接資金の調達を支援している(伊丹, 1995; 谷光, 1999; 藤村, 2000)。資金調達力を源泉とした現況を見る限り、半導体を製造する中枢と言えるシリコン・ファンダリという事業はもはや加工産業かつ資本集約型事業となってきている(O'hUallachiain, 1991)。かつて日米がリードしていた時代とは異なった産業構造になりつつある(Saraswat, 1994; Castrucci, 1995)。1980 年代に栄華を極めた日本の半導体産業は昨今再編成と独立・分社化が進んできている。これは半導体の事業構造が垂直統合から水平分業に変革しているからと言える(O'hUallachiain, 1991; Porter, 2001; Hung et al., 2004)。ナノテク時代のシリコン・ファンダリという半導体専門の製造ラインはより資本集約事業となり、一層巨額な規模の資本投下が必要となってきている(Baily et al., 1998)。こういった進展過程の中で、半導体産業ではそれに対応する新しいビジネス・モデルが次から次へと仕掛けられ、展開されてきた。これがビジネス・アーキテクチャの展開である(三輪, 1999; 藤本, 2002)。ビジネス・アーキテクチャを捉えるには半導体産業全体領域、半導体の生産にかかわる領域、そして半導体の設計開発の領域という三つの領域で区分できる。

本章では、経営資源が少ないベンチャーにとって参入障壁が低い半導体の設計開発領域に注目する。そして半導体の設計プロセスにおける設計知識という統一した次元で半導体産業を見直すことにより、半導体ベンチャーの潜在的ビジネス・モデルについて考察を行う。次に、ファブレス(fabless)・モデルの次に展開されるチップレス(chip-less)・ビジネスを同次元で分析しモデルを導出する。チップレスはモノとしての半導体素子を、製造のみならず生産計画・マーケティング・販売といった経営システムも委託する手法である。半導体設計開発に経営資源を集中させ設計知識という知的財産が収益の源泉である。日本の半

導体メーカがこのチップレス・ベンチャから設計知識を導入し、このモデルが一つのビジネス・アーキテクチャの展開として成立するのか検証を行う。

9.2　先行研究と本研究の焦点

9.2.1　半導体産業における本研究の命題

これまで日米半導体産業に関する先行研究の多くは、産業区分(谷光, 1999; 藤村, 2000)や時代区分(伊丹, 1995)、クラスタ(Porter, 1998)といった形態区分によって比較分析がされてきた。これらの分析に共通しているのは半導体産業が技術開発競争によって発展してきた事実である。産業区分では大企業・国家レベルでの競争戦略に焦点を当てた研究も重ねられてきた (Baily et al., 1998; 三輪, 1999; Porter, 2001)。これらの研究の積み重ねから半導体産業に適しているのは大企業かベンチャーかという議論も行われてきた。半導体産業におけるベンチャーはイノベーション(Schumpeter, 1950)の牽引役に付け加え、基礎技術から応用技術への触媒的役割を担ってきたのはまぎれもない事実である(中内, 2002)。目前の問題意識として今、大企業・国家レベルでの競争戦略を議論することより、これから期待される日本の半導体ベンチャー企業を確立させるための一層の議論と検討が早急に望まれてきている。

本章ではベンチャー企業が参入し易い半導体産業の設計開発領域を切り口として議論を進める。そして半導体の設計開発における設計知識という次元で分析を進める。設計知識に関しては、米国の半導体産業は日本より知識が多様化している。これは外部から知識を導入している傾向が日本より大きいことが要因の一つとされている(中内, 2002)。企業が外部にある知識を導入する理由は、知識の源泉がイノベーションにとって重要であるからである(Cohen and Levinthal, 1990)。

現在の日本の半導体産業にとっては、アウトソーシングによって得られた外部からの情報と知識を総合して新しい価値観と知を機動的に、知識創造の共同体(野中・竹内, 2002)として、どの様に造り、どう収益追求をしていくのかが課題と言える。これは同時に現在日本の大企業である半導体企業が置かれている立場とは異なる、未だ欧米に比べて起業の確立が遅れている半導体ベンチャーの存在の課題にたどり着くとも言える。つまり、日本の大企業の半導体メーカとコラボレーションする上で、同様に新しい価値観と知を創造し、収益追求する日本の知識集約型の半導体ベンチャーが求められているのである。これはファブレスの次に展開されるチップレス・モデルを担う半導体ベンチャーである。

そこで本章の関心は、チップレス半導体ベンチャーがプロダクト・イノベーションを持ち魅力的な半導体 IP を創造した時に、どういった条件下で日本の半導体メーカとのコラボレーションが成立するのかという点にある。この命題を分析フレームに従ってモデルを導出し解明する。

9.2.2 ビジネス・モデル・イノベーションにおける本研究の位置付け

1) 本研究におけるイノベーションの拠り所

本研究におけるイノベーションの定義は、von Hippel (1988)と小川(2001)によるイノベーションに向けた問題解決並びに情報利用に関する既存研究から出発している。つまり、イノベーションを「顧客が持つ問題の解決(Kahney, 1986)のための、新しい情報利用」として議論を進めようというものである。この情報は知識を含んだより包括的概念を持っている。例えば製品開発におけるモノを捨象して得られる設計情報(Clark and Fujimoto, 1991; Eppinger, 2001)や「正当化された真なる信念」(野中, 1990)と定義された知識等である。ビジネス・モデル ・イノベーション(von Hippel, 1999; 小川, 2002)は、情報の粘着性(von Hippel, 1994; 小川, 2000)とイノベーションの分布の多様性に関する研究において提示されてきた。ビジネス・モデル・イノベーションにおいてチップレスがどの様に位置付けされるのか以下の通り議論を進める。

2) イノベーションの発生分布の多様性

マーケティングの一般論としてイノベーションの発生主体はメーカが主導になって行うものであると考えられてきた（小川, 2001）。メーカによる商品開発はマーケティング活動の一環であり、時代や環境の変化に対応して、シーズ指向からニーズ指向へ変遷してきた。シーズ指向とは高度成長期の大量生産・大量消費の時代のマーケティング・コンセプトである。一方、ニーズ指向とは顧客のニーズを的確に反映させた商品開発を行うマーケティング・コンセプトであり、現在でも利用されている伝統的手法である。このコンセプトのもとでは、標的市場を想定し、顧客属性に応じて求められているニーズを探るという接近法が取られている。そしてこの手法には標的市場となる顧客の一部をモニターしたニーズ調査、消費者行動の分析、統計学や心理学を応用した市場調査がある。このサンプリングが標的市場全体の求めているニーズであると仮定して、メーカは革新的製品の開発を進めていくのである。しかしながらニーズ指向はシーズ指向同様、商品開発の主導も実践もメーカにあり、従ってどちらも製品開発におけるイノベーションの主体はメーカにあった。

この様に革新的製品の開発はメーカが行うものという考え方に対して、疑問を投げかける研究が登場してきた。von Hippel (1988)は、産業製品におけるイノベーションの創出はメーカだけではなく、ユーザ側からも創出されている事実を観測し、事例(von Hippel, 1976, 1977, 1988; Shaw, 1985; Voss, 1985)を報告したのである。そして von Hippel (1988)はそのイノベーションを創出したユーザに直接調査を行った。ユーザ自らが欲しいものについて的確な情報をメーカに移転しようとしても、それが困難であった事がユーザの共通した回答であった。故にユーザ自身でイノベーションをしようと決心したと言うのである(von Hippel, 1994)。その情報とは暗黙的な知識であったり、状況に埋め込まれたものであったり、単にメーカに移転させることが困難であったりしたものだと言う（この情報には後述の製品開

発における機能デザインと技術デザインが含まれる)。つまりイノベーターとイノベーション発生には多様性が見られるのであり、イノベーションに関連した問題解決とイノベーションに必要となる情報の移転問題がイノベーターの分布を説明する有力な要因になるのではないかというのが von Hippel の主張である。

3) 製品開発法による比較

表 9.1 は小川が提示したビジネス・モデルの製品開発法による比較表に、チップレスを追加したものである。以下、それぞれのビジネス・モデルについて議論する。尚、ファブレスは伝統的開発による製品開発手法である。

ある特定のユーザが既に行っている革新的な機能デザイン（ユーザ自身が自ら抱える問題の発見とその機能要件への翻訳）や技術デザイン(当該機能を実現するための生産技術を含めた要素技術の組み合わせの創出) （小川, 2001）をメーカが積極的に取り込む手法が、一部のメーカで成果を挙げていることが報告されている（von Hippel, Thomke and Sonnack, 1999)。この手法がリード・ユーザ法(von Hippel, 1988)である。

また近年、ユーザ自身が生み出す製品案を実現し、市場化することを可能にするプラットフォームを提供する企業が登場しつつあることが報告されている（小川, 2002)。この仕組みがユーザ起動型ビジネス・モデル（小川, 2002）である。ユーザ起動型ビジネス・モデルの特徴は、(1)ユーザ自身が企業の掲示板に製品化を目的として自身の発案を書き込み、自らの存在を積極的に訴求すること(2)受注生産体制を採用していること(3)企業とユーザとのコミュニケーション・メディアとしてインターネットが活用されていること(4)想定され

表 9.1　製品開発手法での比較（小川, 2002:加筆）

	伝統的開発 （ファブレス）	リード・ユーザ法 (Hippel,1988)	ユーザ起動型 (小川,2002)	チップレス
アイデア起点	メーカ	ユーザ	ユーザ	3P (サード・パーティ)
アイデア発見過程	メーカによる伝統的市場調査	メーカによるリード・ユーザの発見	ユーザによる掲示板への書き込み	メーカとユーザ双方による 3P の発見
競争の基礎	製品	製品	システム	知的財産(機能デザイン要素､技術デザイン要素)
目標	革新的製品の開発	革新的製品の開発	ユーザ発の企画の製品化	革新的製品の開発
生産体制	見込み	見込み	受注	見込み
生産ロット	大ロット	大ロット	小ロット	大ロット

る受注数（生産ロット）が小規模であること(5)競争の基礎が製品それ自体ではなく仕組み（ビジネス・モデル）におかれていることの五つが挙げられている。

サード・パーティー(3P)が生み出す製品案を実現し、市場化することを可能とする設計知識という知的財産を提供するベンチャー企業が胎動してきた。この手法がチップレスである。この知的財産を採用する製品開発手法は、幾つかの点で伝統的開発、リード・ユーザ法、ユーザ起動型とは異なっている。まずこれら三手法は、機能デザインと技術デザインの主体をメーカ又はユーザと考えていた。しかしチップレスでは、この機能デザインと技術デザインをメーカでもユーザでもない 3P が主体となって提供する。すなわちチップレス半導体ベンチャーである。第二に仕組みである。チップレス半導体ベンチャーの起業家はもともとメーカ側かユーザ側どちらかでキャリア形成をしてきた技術者である。そのチップレス半導体ベンチャーが、機能デザインと技術デザインを含む粘着性の高い設計情報並びに設計知識をメーカとユーザに訴求する。ここでメーカはライセンスの使用許諾相手である。そして第三に競争の基礎は、チップレス半導体ベンチャーが提供する知的財産権であり、ユーザが購入する有形の製品ではない。またユーザ起動型の様な場の提供というプラットフォーム・システムでもない。生産体制や生産ロットについては、使用許諾相手のメーカの意思決定に任されており、伝統的開発とリード・ユーザ法と変わりはない。

ここであらためて確認できるのは、第一にイノベーションの源泉は産業や製品カテゴリによって異なるということである。つまり半導体産業ではイノベーションを生み出すプレイヤーはメーカ（シリコン・ファンダリーとファブレス・メーカを含む）やユーザとは限らず、3P がその重要な役割をしてきていることである。そして第二に機能デザインと技術デザインといった無形の知識財産をライセンスするビジネス・モデルが展開されてきている事実である。

9.3　分析フレーム

分析フレームにおける分析の次元については、一般設計学(吉川, 1985)における概念の変換過程に従って議論を進める。生産資源を実体概念と抽象概念に二分するなら、実体概念と抽象概念の写像操作が設計である。半導体素子は実体概念であり、設計知識が抽象概念である。

ここで設計知識を構造、過程、機能という三つの視点で分解し、半導体製品開発における構造的知識、そして過程知識、システム知識(青島・延岡, 1997)と定義しておく。構造的知識は形式化されている設計情報を含む。設計情報は Fujimoto and Clark (1991)的なパラダイムで表現するなら、設計開発工程を経ながら変形していき最終的にはシリコンという媒体に転写される。設計情報は多様な変形をしていくが、各設計情報は有機的に体系化されており知識形成が行われている。過程知識とシステム知識は形式化が困難で簡単に移転できない暗黙知(野中・竹内, 2002; Polanyi, 2003)である。過程知識は製品開発に関する情報の

内、時間的なコンテクストの中で意味づけられたものと定義している。つまり設計開発における試行錯誤的問題解決活動の経緯の中で意味づけられている情報が過程知識である。一方、システム知識とは製品開発に関する情報の内、製品システムや組織システムのコンテクストの中で意味づけられたものと定義している。つまりここでのシステム知識とは製品システムや組織システムを構成する様々な要素関係の中で意味づけられた情報のことを指す(小川, 2001)。

設計知識を源泉にビジネス・アーキテクチャとして定着したビジネス・モデルがファブレス・モデルであり、確立過程にあるものがチップレス・モデルである。次節以降では本分析フレームによってファブレス・モデルとチップレス・モデルを導出する。

9.4　ファブレス・モデル

9.4.1 分析フレームによるモデル化

ファブレス半導体は、半導体産業が垂直統合から水平分業に変革する初期の 1980 年代に米国で登場した。Conway and Mead (1979)による設計開発工程の効率化と自動化の研究が半導体の製造と回路設計の分離を実現させたことに起因している。そして 1990 年代に登場したハードウエア記述言語（HDL, Hardware Description Language）と呼ばれるコンピュータ言語により飛躍的に設計の生産性を向上させた(Armstrong, 1988)。さらにこの HDL で設計者は生産技術により異なる最終的な製造プロセスや半導体メーカ、シリコン・ファンダリに依存せずに設計することが可能となった(山本, 2001)。こういった経緯によりファブレス・モデルが定着した。ファブレス・モデルは半導体の設計開発・マーケティング・販売に経営資源を集中させ、自社では製造・組立・テスト設備を持たず生産システムをシリコン・ファンダリに委託するモデルである。

ファブレス半導体は、半導体産業における構造変革の過程の中で、多額の資金が必要な半導体工場を自前で持つことが非現実的であること、そして半導体の付加価値の源泉が半導体の物理的な製造そのものよりも半導体チップのアーキテクチャ、仕様の決定、回路設計、そして物理的設計や設計の具現化（implementation）に移ってきていることにより(注 1)、一般的米国市場では創出・急成長が期待されるベンチャーとして大変認知されている分野である。Hung and Yang (2003)によれば半導体のファブレス事業支援という国家レベルでの政策根拠はそれがイノベーション上重要な一指標であることによると指摘している。

前述の分析フレームによって導出したファブレス・モデルを図 9.1 に示す。実体概念と抽象概念の流れを一般設計学の変換過程に従い表現したブロック図である。$\boldsymbol{M}_1$ はシリコン・ファンダリで製造された実体概念であり、ファブレス・メーカの経営システムを通じて顧客に到達する。$\boldsymbol{M}_1$ は半導体素子である。$\boldsymbol{U}_1$ と $\boldsymbol{U}_2$ はファブレス・メーカが設計開発する抽象概念であり、シリコン媒体に転写される前の設計知識を含む。ただし $\boldsymbol{U}_2$ には構造的知識が形式化される前の設計情報は含まない。

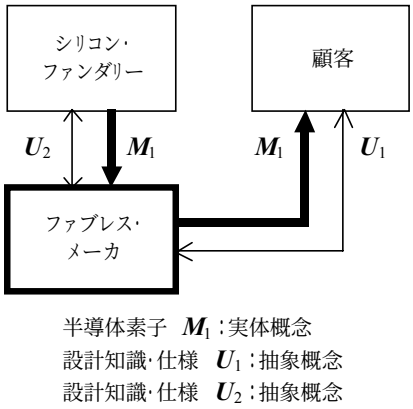


図 9.1　ファブレス・モデル

9.4.2　日米のファブレス半導体市場の相違

米国の半導体産業については､GDP で 1987 年には 1.5%だったものが 2000 年には 6.5%、従業員数 28.4 万人の規模に拡大した。高付加価値産業であることに加え、高賃金の雇用を生み出し、生産性向上の面では他産業を含め波及効果の大きな重要産業と位置づけられている(Spencer, 2004)。一方、日本の半導体産業は 1999 年度に従業員数 19 万人を雇用する規模となり、年間付加価値額 2 兆 8000 億円となった(注 2)。日本の半導体産業は設備投資一兆円（2000 年度）規模であり、半導体産業を側面で支える装置メーカへの波及効果の大きさが理解できる。

米国のファブレス半導体は、1990 年代の米国半導体業界の成長を支えた原動力となったことは周知の通りである。ファブレス半導体の業界団体である FSA によれば､2002 年現在、米国だけで総数が約 400 社存在していると報告している。尚、2003 年度における NASDAQ 市場に公開しているファブレス半導体メーカ 36 社を調査したところ、米国のファブレス半導体メーカの従業員の総数は約 4 万人であり、大きな雇用創出につながっていることが判った。米国市場では 20 年以上に渡り、ファブレス半導体は高い成長性のあるベンチャー企業として認知されている。そしてスピードが求められるエレクトロニクス社会において半導体産業の水平分業体制によるファブレス半導体のビジネス・モデルが非常に有効に機能していると評価されている。尚、世界市場においてもファブレス半導体は高い成長性を認知されている。Taiwan Semiconductor Industry Association 2000 の Global IC Fabless industry review によれば 1999 年にはファブレスは全半導体全売上げの 8%と報告している。FSA (Fabless Semiconductor Association)では 2010 年の半導体の全売上の 50%がファブレス半導体

によって占められると予測している。

一方、日本でのファブレス半導体ベンチャーの創出が遅れた。産業タイムズ社発行の「半導体ベンチャー最前線 2002」によれば 2002 年 7 月現在で、日本発の半導体ベンチャーは 51 社設立にとどまっている。

近年ここにきてやっと日本発のファブレス半導体ベンチャー企業が日本の株式市場で公開を果たしてきている。2003 年 4 月時点でファブレス半導体ベンチャーは、メガチップス（東京 1 部: 6875)、鷹山（JASDAQ: 6830)、リアルビジョン（東京 2 部（マザーズ）: 6786)、ザインエレクトロニクス（JASDAQ: 6769)、アクセル（JASDAQ: 6730）の 5 社（2005 年 1 月現在、鷹山は事業内容を完全に変更している）である。これら株式公開している日本のファブレス半導体メーカの従業員総数は 2003 年時点で 1000 人に満たない。尚、Hung et al.(2004)の調査によれば 1999 年実績でファブレスは半導体全売上げ US$117 億の 8%であり、ファブレスの市場シェアは米国が 60%、台湾が 22%、カナダが 4%、ヨーロッパ及びイスラエルが 7%であった。日本はファブレス事業の世界市場シェアで 3%にも達していないのである。

9.4.3　日米のファブレス半導体ベンチャーの相違

昨今日本のファブレス半導体ベンチャーの起業の動きには、シリコンバレーに触発されたとはいえ、日本固有の特徴がありそうである。そこで日米のファブレス半導体メーカの製品と企業の視点で、株式公開している日本のファブレス半導体ベンチャーの特質を抽出した。調査対象企業は前述の 2003 年度における NASDAQ 市場に公開している米国のファブレス半導体メーカ 36 社と同年の株式公開している日本のファブレス半導体ベンチャー5 社である。表 9.2 に日米の株式公開しているファブレス半導体ベンチャーの相違点をまとめた。

表 9.2　日米ファブレス半導体の相違

(調査対象先は米国 NASDAQ 公開企業 36 社、日本は公開企業 5 社である)

	米国の半導体産業	日本の半導体産業
従業員数	28.4 万人（2000 年調査）	19 万人（1999 年調査）

		米国のファブレス半導体	日本のファブレス半導体
従業員数（公開企業）		4 万人(2003 年調査)	１千人未満(2004 年調査)
企業数(公開＋未公開)		約 400 社（2002 年調査）	約 50 社（2002 年調査）
創業時	起業時の 創業者の年齢	30－50 歳 (代表者はプロの経営者を招く)	50 歳前後 (代表者は引続き創業者)
	顧客と創業者の関係	濃厚と希薄の二極化（前職でのビジネス上の個人的キャリアや技術的キャリアによる）	濃厚（創業者の前職務で人間関係、ビジネス実績あり）
商品と市場	半導体の種類	特定用途向け標準品(ASSP)	特定用途向け標準品(ASSP)と 特定用途向け半導体(ASIC)
	半導体の用途	コンピュータ、通信機器、 コンピュータ周辺	ゲーム、家電、医療機器、 液晶パネル
	半導体の性格	汎用製品指向 （競合他社製品との競争）	カスタム製品指向 （競合他社製品の回避）
	商品戦略	独自アーキテクチャ、 パフォーマンスとコスト追求	ニッチ市場
	半導体の 設計範囲	論理設計と物理設計重視	論理設計に偏り
	顧客	複数（まんべんなく）	特定顧客限定・集中
	市場指向	全世界	日本
株式公開時(以降)	従業員数	100－1,000 人	20－300 人
	主要株主	VC、創業者、経営者、技術者	創業者、VC
	代表者の代替	一般株主の意見重視。経営者や技術者の層が厚く、外部から招き、代替化、人材の流動性も大。	創業者が筆頭株主であり強いリーダーシップ。創業者の存在なくして、事業の継続性なし

1) 創業時

日本での創業者のほとんどは半導体メーカに長く携わっており、そこでのキャリアを形成して起業したケースが多い。従って起業時の創業者の年齢は 50 歳前後が占める。半導体に関する教育は、今でこそ大学において盛んであるが、1990 年までは物理学に近い物性の研究がほとんどであった。半導体設計に重要な論理設計が避けられてきた。その為、半導体設計者は大手の電機メーカの半導体部門に入ってから企業内研修を経て、業務を行いながら修得していく。日本の電機メーカの組織は階層が米国のそれと比べても深いので、半導体事業の体系も理解できず、半導体設計のある工程のみ長く従事することが多い。またキャリアを積めば、居心地が良い（注 3)。また問題意識（注 1）があったとしても、どうしても大企業依存指向、外資半導体依存指向で安定感を求めてしまう。結局、多くの起業家は 50 歳という年齢を待たなければならなかった（注 4)。

日本のファブレス半導体ベンチャーのビジネス顧客は、創業者が起業前の前職で何らかのビジネス関係や実績をテコにしている場合が多い。すなわちその時には技術としてのポジションを既に離れ、外交を積極的に行ってきたことがうかがえる。しかし米国では技術のコアを握るエンジニアが起業するケースが多く、スピード経営を実践する為にも自らは CTO (Chief Technical Officer)やチーフ・サイエンティストとして技術に専念し、経営はプロに任せる傾向にある。

日本のファブレス半導体は 50 歳という年齢でも成功できる大変心強い例証とも言えるが、大学教育の充実の方向と日本の半導体メーカの組織の階層構造が浅くなってきていることを踏まえると、この起業の年齢は低下していく方向となろう。

2) 商品と市場性

次に商品と市場性について考察してみる。日米のビジネス・プランを比較すると、日本の場合にはスケールの差があるのは否定できない。日本の場合には前職との顧客のビジネス関係からスタートする事例が多く、どうしても特定顧客向けの商品、または ASIC (Application Specific Integrated Circuit:特定用途向け半導体)になりやすい。従ってこういったカスタム製品は顧客の仕様に基づく場合が多く、価格の支配権も顧客に握られてしまう。またファブレス半導体であるにもかかわらず、上流の論理設計のみで勝負し、ニッチ戦略で隙間を狙う企業が多い。日本の半導体メーカでは上流の論理設計と下流の物理設計が分断されており、双方の領域をこなせる半導体エンジニアが育成されていないことも原因である。一般に日本のベンチャ・キャピタルからの投資金額は一社あたり数千万円から一億円程度と投資金額が米国に比べても 1、2 桁違いがあるが、その一つの要因は、日本のファブレス半導体ベンチャー自身大規模な投資を引き出すスケール性を持っていないからとも言える。

米国のファブレス半導体ベンチャーを見てみると、彼らは一流の専門家による分業体制

の組織と人材構成となっている。経営、技術、マーケティング、営業、それぞれの分野の専門集団である。一方、日本の代表者は全てをカバーしなければならず、事実、株式公開しても常にその有価証券報告書にはリスク事項として、代表者の存在無き場合には、事業継続の多大なリスクが存在すると指摘されるのである。米国流のベンチャー形成の方が効率的で効果的である。これにより必要な領域において最適なパフォーマンスを生み出し、競争力をつけてくれる。ある意味でリスク・ヘッジがなされているのである。

日本のファブレス半導体ベンチャーは、技術者だけの会社経営はよく見られる。そして米国のファブレス半導体で最も重要視されているマーケティング戦略を立案できる人材が日本では少ない。ほぼ代表者が兼務していると言って良いであろう。そもそも日本では半導体分野での一流のマーケティング戦略を構築できる人材は不足している。技術的知識とマーケティング理論をおさえつつ、大きな画用紙に大きな絵を描き出せる人材の創出が少ないのは、日本国内で半導体産業に従事している人達は日本市場指向であり、グローバルな世界でグローバルな思考が形成されたかったからである。それは日本市場固有の半導体の種類、用途、性格といった事実、そして公開している日本のファブレス半導体企業が特定顧客に依存している事実から確認できる。日本市場では創業時からハイ・リスクを低減する様に努める傾向が強く、それが日米の人員規模・売上規模の格差、スケール格差を生みだしている。一方、少々過信しすぎて、汎用品で競争のある市場を狙う日本のファブレス半導体ベンチャーの場合、米国のそれに比して正しいマーケティング戦略が構築できず、ハイ・リスクというより頓挫する傾向が強い。

日本のファブレス半導体ベンチャーの商品には、対象市場の設定に対して変革的ひねり(柳, 2001)を入れて高付加価値を形成している半導体が見られる。例えばザインの LVDS 半導体は、アナログで高速シリアル転送可能なインタフェースで、入出力半導体である。近年日本の半導体メーカが横並びで大合唱しているシステム LSI より規模的にも複雑性においても単純な半導体素子である。しかしこの入出力は全てアナログであり、高額な設計支援ソフトウエアを導入すれば完成する LSI ではない。アナログ設計を 10 年以上従事した熟練工、職人的エンジニアがトランジスタを手設計して行わなければならない。これは正に属人的な長年の経験とノウハウ、知識集約型の半導体である。今やデジタルが主流であり、アナログ設計者は日本でも少なくなってきた。これは世界的な傾向であり、アナログで米国老舗企業のフェアチャイルド社(NYSE:FCS)、ナショナル・セミコンダクター社(NYSE:NCM)、アナログ・デバイス社(NYSE:ADI)、マキシム社(NASDAQ:MXIM)といったアナログ専用半導体メーカは逆に収益性が高くなっている。米国でもアナログ設計者の需要は高い。顧客からしてもアナログ半導体は置き換えが困難で、デジタルに比べて消費の浮き沈みが少ないのも半導体メーカに好都合である。

未公開企業ではあるが新世代株式会社も変革的切り口で高収益を上げている。彼らのデジカメ用半導体は、ソニー、キヤノン、オリンパス等のハイテク・デジカメ用半導体には機能的に劣る。しかし開発期間を 5 分の 1、コストを 5 分の 1、そしてキーワードとして、

三つの特徴「速い、安い、面白い」を供えた最終商品を意識して、物造りをしている。いわゆる「割り切り」である（注 5)。こういった半導体はハイテク玩具では競争もない。ローテク半導体でも対象市場を変革的切り口でのぞめば、十分高い収益が上げられることを物語っている。

3)　株式公開時（以降）

次に株主構成に目を移すと、米国では創業者といえども、日本ほどの大株主といった持ち株比率ほど持ち合わせていない。米国では先に説明したとおり、プロの組織と人材が集合し、そしてスケールを狙ったマーケティング戦略に基づきベンチャ・キャピタルから多額の資本調達を行う。しかし、日本の創業者は引き続き主要株主としての存在を主張する傾向がある。また日本のベンチャ・キャピタルは一株あたりの株式評価に対して大きなプレミアムは避けたい傾向が強く、持ち株比率を考えると多額の資本調達が行えないことも日米のスケール格差に影響を与えている。

9.4.4　日本におけるファブレス半導体ベンチャーの起業の遅れ

日本では米国に比べファブレス半導体ベンチャの起業が遅れ、まだ定着していない。その大きな理由の一つは日本では半導体産業がクローズドな業種であったからである。1980 年代の日本が驚異的に半導体産業の競争力があった時代、半導体事業は、製造装置、テスター、組立等の副次的業種までも囲い込んでいた。そして半導体ユーザの多くは自社のシステム部門であった。つまり日本の大手半導体として専業メーカは少なかった。必然的に系列企業に前述の副次的業種を委託していた。そして半導体の販売と消費のサイクルの多くはその系列企業内で行われてきた。従って系列企業は競合他社の半導体メーカの消費機会は多くなかったのである。これが日本独特のクローズなビジネス社会を創り出した。つまりファブレス半導体ベンチャを起業しても、日本では製造、テスト、組立を外注することが困難であったし、しかも顧客の確保が系列という障壁のため容易ではなかった。ベンチャにとっては資金調達以外にこういった日本市場の販売機会についても十分検討する必要があった。

1990 年代に入り、半導体産業は垂直統合型事業構造から水平統合型事業構造に移っていった。日本の半導体メーカはそれに合わせ垂直統合事業内の事業を売却、再編、撤退によりアンバンドリング、集中と選択を行ってきた。従って半導体産業の黎明期からストックとして積み重なってきた技術ノウハウ、製造ノウハウも水平統合型事業構造内の各有力企業へ無形資産や知財として移転し、オープン化の流れになった。このオープン化の流れを受けて半導体の設計開発と製造は容易に分離できるようになった。この頃が日本のファブレス半導体ベンチャの起点と言える。実際のところ 1990 年にファブレス半導体ベンチャとして最初のメガチップス（東京 1 部:6875）が創業している。

一方、米国では半導体の製造が技術集約型から資本集約型へと変革した産業構造の中で 1980 年頃からファブレス半導体が本格化してきた。つまりファブレス半導体ベンチャーの出発点は日米で 10 年以上もの開きがあるということになる。更に 2002 年現在の調査によれば、米国の半導体ファブレス企業が約 400 社に対して日本では約 50 社である。日本において起業の定着化が遅れている理由の一つは、シリコン・バレー型のベンチャ・モデルの定着不足もある。シリコン・バレー型のベンチャとは、産業界のニーズを取り込んだ大学が最先端の研究を行い、その目処がついた段階で独特の人脈を利用して人材を集める。そして事業計画の策定、ベンチャ・キャピタルの資金提供も受けて会社を設立、実用化に向けて邁進するベンチャ・モデルである。

このシリコン・バレー型のベンチャについては人材の流動性とダイナミズムが引き起こしていることも言及すべきである。人材の流動性はシリコン・バレーが形成された時からのものであり、これがシリコン・バレーのダイナミズムを生んでいる。このダイナミズムとは起業力学である。シリコン・バレーで自らが事を起こす人にとって、起業・創業は危険が伴う投機・冒険であると必ずしも思ってはいない。起業・創業とは、自分で考え、自分で責任をとり、自分で創る人生そのものと考えている。シリコン・バレーで直接半導体に従事している者は、大企業に依存するのだ、今いる会社が私に生活の安定をもたらしてくれるのだと考えることは少ない。この行動意識がダイナミズムの源泉の一つであると言える。

日本ではファブレス半導体ベンチャの起業が遅れ、定着していない理由は他にもある。それは日本の大学における半導体の回路設計教育、大企業での半導体エンジニアリング業務が細かく分業化されていること、問題意識の欠如による起業力のなさ等、ベンチャ企業論において議論される日本固有の問題にあると考えられる。

現状、半導体の受託設計サービス業として起業しているベンチャを散見する。そのほとんどが収益に苦しんでいる。これは大企業である日本の半導体メーカとの間で、受託設計サービス業による旧態とした下請的な受託設計の枠組みに終始しているからである。従属的な関係によって、経費・計画・品質まで委託する側の支配をうけてしまう。受託設計の枠組みを脱却して成果物に受託する側の創造物の付加価値を認める動きはほとんど見られない。半導体ベンチャを志す起業家にとっては、簡単にファブレス半導体事業を興すことは容易ではないからである。ファブレス・モデルは半導体チップの製造とテストを外部委託するものの、生産計画・マーケティング・販売という経営システムを自ら構築する必要がある。この経営システムは図 9.1 のモデルの実体概念の流通コントロールである。下請的受託業務とこの経営システムがファブレス・モデルに励起させきれないベンチャの内的要因と考えられる。

シリコン・バレー型ベンチャ・モデルが日本で定着されていない現状を見る限り、米国のファブレス半導体ベンチャの事例を踏襲することは必ずしも賢明とは言えない。これから創出される日本のファブレス半導体ベンチャは、日米の株式市場・顧客・求められる半

導体の属性の違いと、日本固有の組織・人材・マーケティング戦略を踏まえながら経営を実践することが有効であろう。

9.5　チップレス・モデルと実証

9.5.1　チップレス・モデル

欧米ではファブレス・モデルは既に定着している。そして近年、半導体 IP (Intellectual Property) (平嶋, 1998)を知的財産としてそのライセンスとロイヤリティを収益の源泉とする知識集約型のビジネス・モデル、すなわちチップレス・モデルが確立されつつある。

2004 年 6 月時点で、米国 NASDAQ 株式市場に公開しているチップレス・メーカは 9 社存在している。この 9 社の従業員数は合計 2100 名程で、年間付加価値額は約 US$7.2 億であるが、一人当たりの年間付加価値額は US$3300 万と極めて高い。対象となる半導体 IP は技術的に市場を先取りしたイノベーションを呼び起こすものである。システム LSI に必要となる基本機能ブロックであるマイクロプロセッサ、メモリー、通信関係が含まれていることがうかがえる。IP の用途と IP の属性により、収益性や企業価値に格差は見られる。

経営資源が少ないベンチャーにとって、チップレス・モデルはファブレス・モデルよりも更に参入が容易である。ファブレス・モデルでは、ファブレス半導体ベンチャーは半導体の設計開発・マーケティング・販売に経営資源を集中させ、自社では製造・組立・テスト設備を持たず、その生産システムをシリコン・ファンダリーに委託する。それに対してチップレス・モデルでは、チップレス半導体ベンチャーは半導体の設計開発にのみ経営資源を集中させ、製造・組立・テスト設備といった生産システムに加え、マーケティング・販売システムも外部に委託する。チップレス半導体ベンチャーは自社でイノベーションのある半導体 IP を創造し、半導体メーカとのコラボレーションによりそのライセンスとロイヤリティで収益を上げるという仕組みである。

チップレス・モデルは大企業である日本の半導体メーカとのコラボレーションである。そして、日本の半導体ベンチャーにとって下請的受託設計サービス業からの大転換を成しえるものである。

9.5.2　日本の半導体メーカにおける設計知識の活用の問題

半導体 IP は半導体の設計知識である。チップレス・モデルは設計知識を源泉にビジネス・アーキテクチャとして展開されつつあるビジネス・モデルである。米国の半導体メーカは設計知識の重要性を認識し、戦略的に積極的に活用してきた。日本の半導体メーカにおける設計知識の活用の問題点を指摘する。

日本の半導体メーカでは、組織内に必要な設計知識が断片化して存在している。これらの設計知識は適切な時間と適切な場所に配置されていないとか、組織構造上の問題で見え

ない壁によって共有化されてこなかった。また知の場では異質の知を排除する可能性が多分にあり、それでは新たな知の形成が起きない。結果的に多くの設計知識が意識的に、そして組織的に活用されないまま埋もれてしまってきた。設計知識は再利用されない限り、付加価値収益として形成されない。

また、日本の半導体メーカでは、設計・生産・消費のサイクルがグループ内で閉じられていることが考えられる。日本の半導体メーカは歴史的に、米国の半導体メーカへのキャッチアップ戦略をとってきた（伊丹, 1995）。そして国の支援と日本の金融機関からの資金により生産技術と生産能力を高めていった。これら半導体の供給先の多くはグループ内のコンピュータや民生機器といったシステム部門であった。

これらが日本の半導体メーカが、設計知識を代表する半導体 IP を外部から導入がしづらい理由と予想できる。

9.5.3　分析フレームによるモデル化と命題

先の分析フレームに従って導出したチップレス・モデルを図 9.2 に示す。

$\boldsymbol{M}_1$ は半導体メーカから顧客に納入される半導体素子という実体概念である。また $\boldsymbol{T}_1$ はチップレス・メーカが設計開発する抽象概念であり、シリコンという媒体に転写される前の構造的知識が形式化された設計情報、過程知識とシステム知識を含む。これが半導体 IP である。$\boldsymbol{T}_2$ と $\boldsymbol{T}_3$ は仕様書であり過程知識とシステム知識を含んだ抽象概念である。ただし $\boldsymbol{T}_2$ と $\boldsymbol{T}_3$ の仕様情報は互いに異なる。

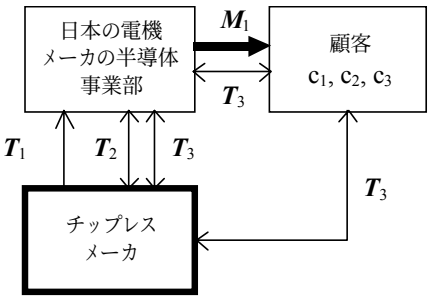


図 9.2　チップレス・モデル

前項では日本の半導体メーカが半導体 IP を外部から導入がしづらい理由として、半導体の供給先の多くがグループ内であることを指摘した。そこでモデル化にあたっては顧客を分類して設定する。すなわち顧客対象として、グループ内のシステム事業部を c_1 とする。グループ外で、半導体メーカとチップレス・メーカ双方の共通の顧客を c_2 とする。グループ外で、c_2 以外の顧客を c_3 とする。

次項ではチップレス半導体ベンチャーと日本の半導体メーカとのコラボレーションがどのような条件下で成立するかの検証を行う。コラボレーションの成立を判断するために、半導体メーカ自身がファブレス・メーカの半導体 IP を導入するにあたり、半導体 IP の投資に対する期待利得を測定することで検証することとした。半導体 IP により半導体メーカの半導体製品にプロダクト・イノベーションが創出されることを条件付けしている。

9.5.4　検証方法と結果

1)　調査対象と調査手順

調査日は 2004 年 1 月 29 日及び 30 日に行った。横浜で開催された Electronics Design and Design Solution の Conference に出席した被験者に図 9.2 のフレームワークを説明しアンケート協力を要請した。434 名がサンプル数である。調査は記名式で行った。161 名から回答を得ており、回収率は 37.1%であった。161 名の被験者が属する企業名で分類したところ、企業数は 17 社であった。これら企業は半導体事業部を持つ大手電機メーカである。企業名を属性として分類し、上位４社の被験者の数を表 9.3 に示した。

被験者は日本の電機メーカの半導体事業部で働く中間技術管理職である。被験者は管理職になる前に半導体のエンジニアリング業務に携わりキャリア形成してきた。具体的には、半導体の回路設計又は、FAE (Field Application Engineer)として顧客向けの営業技術支援である。中間技術管理職の現在の業務は、過去の技術的知識を生かして、顧客と直接接触しながら商品企画を行うことである。そして設計者・開発費・開発期間といったリソースの意思決定、エンジニアリング業務を含めた事業計画を含むプロジェクトを実行する管理者でもある。従って被験者は外部の設計知識の導入判断に大きな影響力を持っているので、本命題を検証する上で重要と言える。もちろん半導体事業部で働く中間営業管理職の管理者もチップレス・メーカとのコラボレーション上大切ではある。しかし外部の設計知識導入の評価が困難なポジションであるため、被疑者としてはずすことに重大な不都合はないものと判断した。

説明変数である期待利得の測定には、von Hippel (1998)や小川(2001)らが開発した尺度を使用した。スコアリングについては 0―3 の 4 点法を採用している。期待利得の大きさは調査対象者へのアンケートにより回答を得た。

2)　結果

分析には SPSS 社の統計解析ソフト SPSS11.0.1J を用いた。まず統計処理にあたって信頼性分析を行った。信頼性分析としてのクロンバック・アルファ係数は 0.7544 であり、内的整合性が高いことが確認された。表 9.4 に以下の分析で使用した説明変数毎の度数、平均値、標準偏差をまとめた。

次に説明変数間の較差についてウイルコクスンの符号付順位検定を行い漸近有意確率の算出を行った。表 9.5 がその結果である。このウイルコクスンの符号付順位検定により、5%の有意水準で期待利得 PINT（顧客 c_1 に対する期待利得）と期待利得 PSPE（顧客 c_2 に対する期待利得）間、及び期待利得 PSPE（顧客 c_2 に対する期待利得）と期待利得 PNSP（顧客 c_3 に対する期待利得）間で較差があることが認められた。

表 9.5 のウイルコクスンの符号付順位検定の二つの結果を整理してまとめると、半導体メーカは社内外全ての顧客に対して、プロダクト・イノベーションに利用されるチップレス・メーカの半導体 IP を自社の半導体に組み込んで販売するにあたり、その費用対効果は赤字にならなくとも、十分な期待利得を出せるには至っていない。しかしながら、社内向けのシステム事業部より社外の特定顧客向けへ販売する方が期待利得は大きくコラボレーションは成立しうるとの結果が得られた。

次に集計から得られた半導体メーカの度数分布として上位４社間（A 社から D 社）の多重比較を行った。４社は日本の半導体産業において年間売上高が上位４社でもある。表 9.6 は期待利得 PINT、表 9.7 は期待利得 PSPE、表 9.8 は期待利得 PNSP での比較結果である。5%の有意水準で期待利得 PINT（顧客 c_1 に対する期待利得）では B 社と D 社間、期待利得 PSPE（顧客 c_2 に対する期待利得）では A 社と C 社間、期待利得 PNSP（顧客 c_3 に対する期待利得）では A 社と B 社間でそれぞれ較差が認められた。また 1%の有意水準では、期待利得 PINT、期待利得 PSPE、期待利得 PNSP 全てにおいて A 社と D 社間に較差があることが認められた。

尚、上記の多重比較を期待利得 PINT、期待利得 PSPE、期待利得 PNSP で行った検定結果としては、A 社が最も期待利得が高く、A 社と D 社間に較差があることが判った。

表 9.3　被験者が属する企業:上位 4 社（企業名で属性分類）

企業名	A 社	B 社	C 社	D 社	その他
被験者数	26	27	19	26	63

表 9.4　記述統計量結果

	度数	平均値	標準偏差
期待利得 PINT (顧客 c1 に対する期待利得)	161	2.26	0.685
期待利得 PSPE (顧客 c2 に対する期待利得)	161	2.39	0.624
期待利得 PNSP (顧客 c3 に対する期待利得)	161	2.28	0.726

表 9.5　ウイルコクスンの符号付順位検定

漸近有意確率（両側）	1	2	3
1.期待利得 PINT			
2.期待利得 PSPE	0.015*		
3.期待利得 PNSP	0.825	0.021*	

両側検定:*5%水準で有意

表 9.6　期待利得 PINT の 4 社間の漸近有意確率

期待利得 PINT	1	2	3	4
1.A 社				
2.B 社	0.392			
3.C 社	0.241	0.703		
4.D 社	0.002**	0.020*	0.068	

両側検定:*5%水準で有意、**1%水準で有意

表 9.7　期待利得 PSPE の 4 社間の漸近有意確率

期待利得 PSPE	1	2	3	4
1.A 社				
2.B 社	0.186			
3.C 社	0.018*	0.296		
4.D 社	0.001**	0.064	0.496	

両側検定:*5%水準で有意、**1%水準で有意

表 9.8　期待利得 PNSP の 4 社間の漸近有意確率

期待利得 PNSP	1	2	3	4
1.A 社				
2.B 社	0.043*			
3.C 社	0.130	0.774		
4.D 社	0.000**	0.201	0.149	

両側検定:*5%水準で有意、**1%水準で有意

3) インプリケーション

日本の半導体メーカの多くは歴史的に、グループ内のコンピュータや民生機器といったシステム部門と密接な関係があった。従って日本の半導体事業部門における全ての回路設計開発や設計資産の構築といった設計開発工程、生産そして消費サイクルは、企業グループ内への依存が高かった。日本の半導体メーカが、設計知識を代表する半導体 IP を外部から導入する事例は米国と比べ極めて少ないことは象徴的な事実でもある。コラボレーションの検証結果から示唆されることは、日本の半導体メーカは外部からの情報と知識を総合して新しい価値観と知を機動的に知識創造の共同体を造ることに期待感があるということである。特にチップレス・メーカと共同で開拓した特定顧客に対しては、そのチップレス・メーカの設計知識を導入し融合することで収益追求の期待が高い。つまりその条件下ではコラボレーションは成立しうると言える。

次に企業間の多重比較の結果が与える示唆について述べる。表 9.6、表 9.7、表 9.8 の多重比較の結果からは、全ての期待利得に対して A 社と D 社間に較差が認められる。D 社の半導体事業部門の特徴は半導体の供給先がほぼ社内のシステム事業部に依存している。そして社内依存率はこの 4 社の中で最も高いことが知られている。この D 社は通信機器やゲーム機器向けの製品は提供しておらず、デジタル民生機器を主体としている。一方、A 社は社外に携帯電話やゲームを製造している特定の重要顧客を持ち、社内外で売上のバランスがとれた半導体事業部である。A 社と D 社間で期待利得に較差があるという事実は、半導体事業部門とシステム事業部門間の相互依存性という企業構造と、提供する商品が一つの影響要因と考えられる。

尚、B 社と C 社はそれぞれ期待利得によっては A 社又は D 社と較差が見られるものの、A社と D 社間程の大きな較差まであるとは解釈できない。このことは B 社と C 社が、A 社と D 社の企業構造と商品の性格を双方兼ね備えていることによるものと考えられる。

9.6　本章のまとめ

チップレス・モデルは大企業である日本の半導体メーカとチップレス・ベンチャによるコラボレーションである。ベンチャーにとってはファブレスよりチップレスの方が参入障壁は低い。実体概念である半導体素子の生産計画・マーケティング・販売といった経営システムの負荷がないからである。この構図を分析フレームで導出したモデルにより提示した。

コラボレーションを検証する前に認識すべきことは、コラボレーションとはコラボレーション相手も同時に成功しなければ成立しないビジネス・モデルという点である。しかし日本の半導体メーカは外部からの設計知識を導入・統合して自社で機動的に、そして知識創造の共同体として収益追求する姿勢が欧米半導体メーカと比して極めて小さかった。こういった日本の半導体メーカの状況並びに同視点を踏まえて、設計知識の多様化に向けた

ビジネス・モデルをしかけると想定している。検証結果としては、半導体メーカは社内外全ての顧客に対してはその費用対効果は赤字にならなくとも、十分な期待利得を出せるには至っていない。しかしながらグループ内のシステム事業部ではなく、半導体メーカとチップレス・ベンチャ双方に共通する社外の特定顧客の場合には期待利得は大きくコラボレーションは成立しうるとの結果が得られた。尚ここでは、チップレス・ベンチャは日本の半導体メーカのグループ企業では創造しえないプロダクト・イノベーションを持った魅力的な IP を提供することを前提条件としている。

チップレス・モデルは、半導体メーカとチップレス・ベンチャとのコラボレーションである。そして設計プロセスにおいて外部からの設計知識を総合し、新しい価値観と知を機動的に知識創造の共同体として構築するビジネス・モデルである。最終顧客が半導体メーカとチップレス・ベンチャ双方に共通する社外の特定顧客であること、そして上記の前提条件であるならば、収益追求の期待感は一層高くなる。つまりその条件下ではコラボレーションは成立する。

注

(1)　日経ビジネス 2003 年 1 月 27 日号「基本はあっても正解がないのがベンチャー、しかし経営の本質を見誤るな」
(2)　日本経済新聞社 2002 年「1/3-1/31 第 15 回半導体:経済教室基礎コース」
(3)　文藝春秋　2003 年 2 月号「創業の経緯と人材戦略」
(4)　日本経済新聞 2002 年 8 月 19 日朝刊社説「頑張れ！　遅咲きのベンチャー経営者」
(5)　日経エレクトロニクス 2001 年 11 月 19 日号,pp.P97-104「速い，安い，面白いハイテク玩具開発の内幕」

参考文献

[1] 青島矢一,延岡健太郎: “プロジェクト知識のマネジメント”,組織学会, Vol. 31, No. 1. pp. 26-27 (1997)
[2] Armstrong, James R.: “Chip-level modeling with VHDL” (1988)(山本靖訳: “VHDL デザインテクニック”,電波新聞社 (1990))
[3] Baily, M. C., Zizcwitz, E., Bosworth, B. and Wcstphal, L. E.: “Extending the East Asian Miracle Microeconomic Evidence From Korea”, Brookings Papers on Economic Activity, Microeconomics, Vol. 1998, pp. 249-321 (1998)
[4] Castrucci, P.: “The Future Fab: Changing the Paradigm”, Solid State Technology, 38 (1), pp. 49-56 (1995)
[5] Cohen, Wesley M. and Levinthal, Daniel A.: “Absorptive Capacity:A New Perspective on Learning and Innovation”, Administrative Science Quarterly, 35, pp. 128-152 (1990)

[6] Eppinger, Steven D.: “Innovation at the Speed of Information”, Harvard Business Review, Vol. 79, No. 1, pp. 149-158 (2001)

[7] Fujimoto, T. and Clark, K. B.: “Product Development Performance: Strategy, Organization, and Management in the World Auto Industry”, Boston, MA, Harvard University Press. (1991) (田村明比古訳: “実証研究 製品開発力―日米欧自動車メーカー20 社の詳細調査”,ダイヤモンド社 (1993))

[8] 藤本隆宏: “ビジネス・アーキテクチャ”,有斐閣 (2002)

[9] 藤村修三: “半導体立国ふたたび”,日刊工業新聞社 (2000)

[10] 平嶋竜太: “システム LSI の保護法制:設計情報の流通と法的保護”,信山社 (1998)

[11] Hung, Shiu-Wan and Yang, Chyan: “The IC fables industry in Taiwan:current status and future challenges”, Technology in Scociety, 25, pp. 385-402 (2003)

[12] Hung, Shiu-Wan, Yang, Chyan and Lee, Cheng-Few: “The Vertical Disintegration of Taiwan’s Semiconductor Industires:Price and Non-Price”, Review of Pacific Basin Financial Markets and Policies, Vol. 7, No. 4, pp. 547-569 (2004)

[13] 伊丹敬之: “日本の半導体産業なぜ「三つの逆転」は起こったか”,NTT 出版 (1995)

[14] Kahney, H.: “Problem Solving: A Cognitive Approach”, Open University Press (1986) (認知科学研究会訳: “問題解決”,海文堂出版 (1989))

[15] Mead, Carver and Conway, Lynn: “INTRODUCTION TO VLSI SYSTEMS”, Addison-Wesley (1979) (菅野卓雄,榊裕之監訳: “超 LSI システム入門”,培風館)

[16] OhUallachain, Breandan: “Restructuring the American Semiconductor Industry:Vertical Integration of Design Houses and Wafer Fabricators”, Annals of the Association of American Geographers, Vol. 87, No. 2, pp. 217-237 (1997)

[17] 小川進: “イノベーションの発生論理”,千倉書房 (2001)

[18] 小川進: “ユーザ起動型ビジネスモデル”,神戸大学経済経営学会国民経済雑誌論文誌, 185, 5, pp. 65-76 (2002)

[19] 三輪晴治: “日本の半導体・自動車産業は復権するのか”,工業調査会 (1999)

[20] 中内基博: “半導体産業における技術関連知識の企業間移転と流出の形態”,日本ベンチャー学会誌, No. 3, pp. 63-76 (2002)

[21] 野中郁次郎: “知識創造の経営”,日本経済新聞社 (1990)

[22] 野中郁次郎,竹内弘高: “知識創造企業”,東洋経済新報社 (2002)

[23] Polanyi, Michael（高橋勇夫訳）: “暗黙知の次元”,ちくま学芸書房 (2003)

[24] Porter, M. E.: “Clusters and the New Economics of Competition”, Harvard Business Review, 76 (6), pp. 77-90 (1998)

[25] Poter, M. E.,竹内弘高: “日本の競争戦略”,ダイヤモンド社 (2001)

[26] Saraswat, K. S.: “Adaptable IC Manufacturing Systems for the 21st Century”, Microelectronics Engineering, 25, pp. 131-137 (1994)

[27] Schumpeter, J. A.: “Capitalism, Socialism, and Democracy”, Cambridge, MA: Harvard University Press (1950) (中山伊知郎,東畑精一訳: “資本主義・社会主義・民主主義”,東洋経済新報社 (2001))

[28] Shaw B.: “The Role of the Interaction between the User and the Manufacturer in Medical Equipment Innovation”, R&D Management, 15, No. 4, October, pp. 283-292 (1985)

[29] Spencer, William: “New Challenges for U.S. Semiconductor Industry”, Issues in Science and Technology. (2004)

[30] 谷光太郎: “半導体産業の系譜　巨大産業を築いた開拓者たち”,日刊工業新聞社 (1999)

[31] von Hippel, Eric: “The dominant role of users in the scientific instruments innovation process”, Research Policy, 5, pp. 212-239 (1976)

[32] von Hippel, Eric: “The Dominant Role of the User in Semiconductor and Electronics Subassembly Process Innovation”, IEEE Transactions on Engineering Management, EM-24, No.2, pp. 60-71 (1977)

[33] von Hippel, Eric: “The Sources of Innovation”, Oxford University Press, New York. (1988) (榊原清則訳: “イノベーションの源泉”,ダイヤモンド社 (1991))

[34] von Hippel, Eric: “Sticky Information" and the Locus of Problem Solving: Implications for Innovation”, Management Science 40, No. 4, pp. 429-439 (1994)

[35] von Hippel, Eric: “Economics of Product Development by Users: The Impact of “Sticky” Local Information”, Management Science, Vol. 44, No. 5, pp. 629-644 (1998)

[36] von Hippel, Thomke and Sonnack: “Creating Breakthroughs at 3M”, Harvard Business Review, Sep.-Oct. (1999）

[37] Voss, C. A.: “The Role of Users in the Development of Application”, Journal of Product Innovation Management, 2, pp. 113-121 (1985)

[38] 柳孝一: “ベンチャー企業論”,放送大学教育振興会 (2001)

[39] 山本靖: “半導体 IP と LSI の物理設計技術力”,日本半導体ベンチャー協会, No. 4 (2001)

[40] 吉川弘之: “一般設計学”,機械研究, 37, 1 (1985）

第10章　結論

10.1　本研究の焦点

設計を理解することを目的とした学問は設計学である。設計学における設計は人が概念として想定した要求機能を、それを充足する実体へと変換する行為である。この行為は人間の知的行為であるから、人間の知的行為に関する知識は設計学における研究対象である。ここで体系付けされている設計に関する知識は二つに分けることができる。人間が設計する人工物、すなわち設計対象物に関する知識と、設計をどの様に進めていくかという設計プロセスに関する知識である。

設計プロセスの知識体系は、設計対象物である製品の知識体系に比べて多く議論されてこなかった。現在の工学領域における設計は、工業製品の物理現象に関する知識を体系化するという点に主眼が置かれてきたからである。しかしながら製品の設計開発は単に一般的な自然法則の知識を適時応用すれば良いということではない。設計対象物である製品や製品性能に関する知識に加え、製品の設計開発をどの様に進めるのかという設計プロセスに関する人間の思考がなければ、製品を成果として創り上げることは困難である。つまり製品の設計開発においては人間の思考が関わるのであり、設計プロセスの知識体系についての議論が要求されるのである。

この要求に応える研究の一つは発見的原理に基づく設計方法論、すなわちシステム設計思考という設計の方法論を対象としたワーク・デザインである。設計活動が将来に対する適応努力とするなら、現存しない理想を追求しようとする機会があるはずである。この機会を捉えて理想的なシステムを設計しようとするシステム設計思考法は実際に応用され、設計の方法論として効果的であることが知られている。設計学では設計プロセスが設計方法論として研究されてきたのに対し、設計プロセスを管理的思考で捉えるアプローチもある。プロジェクト・マネジメントである。プロジェクト・マネジメントでは設計プロセスを設計開発工程の構造並びに過程として捉え、設計プロセスの効率化と改善に向けた知識形成が行われてきた。

近年、設計開発工程が大規模・複雑化している状況下で、設計プロセスに関する問題と

して設計負荷の問題がある。それは、設計情報と設計知識（以下「設計情報と設計知識」を「設計知」と呼ぶ）の流れに起因した手戻り修正、無駄、時間とコストのリスクを引き起こす因子、不確実性要因、非効率な工程といった設計開発工程に直接間接的に影響を及ぼす問題である。設計学において、これまでこの問題を解決するための明確な解釈や知識の体系化並びに概念形成が行われてきたとは言えない。また、従来のプロジェクト・マネジメントの枠組みの中で解決困難であることも事実である。つまり、設計負荷の問題は設計学とプロジェクト・マネジメントにおける一つの中心的な問題と考えられる。

そこで本研究における基礎的問題は、設計プロセス上の設計負荷を解明することにある。次に応用的問題としては、設計プロセスの効率化と改善の手法を確立することにある。これは基礎的問題解明に基づき、設計プロセスの技術管理を向上させることであり、経営的視点による効率化と改善の手段を提供するものである。

10.2　本研究の結論

本研究では設計プロセス上の設計負荷の要因と構造を探求し、発生原因を明らかにした。そして設計負荷を低減させるための設計プロセス管理の枠組みを構築することで、設計プロセスの効率化と改善に向けた基礎的な理論を提供した。本研究により、設計学の知識体系における設計負荷の概念を明確にすることができた。また本研究で提案した設計プロセス管理は従来のプロジェクト・マネジメント手法で解決困難であった設計負荷の問題に対処できるものであり、プロジェクト・マネジメントを一歩前進させることができた。ここで本研究の総括として以下の通り結論をまとめておく。

10.2.1 設計負荷の解明

設計プロセス上の設計負荷とは、適切な設計解を得ようとする思考活動に対して影響を及ぼす行為であり、設計プロセス上のリスクを引き起こす原因である。設計プロセス上のリスクには計画納期、計画コスト、計画性能がある。これらリスクを引き起こす因子は、i)イタレーション、ii)人間の行動特性、iii)設計知の不十分な活用、iv)不知の不知等である。これら因子が複合的に関わり合ってリスクに因果効果を及ぼす。これら因子の概念は以下の通りである。

i)　イタレーション

イタレーションとは設計知を媒介としたタスク間の反復現象であり、適切な設計解を導出するための手戻り修正並びに設計を改善させようとする行為によって発生する。イタレーションに関する知見はイタレーションの分類とその分類によって進めるイタレーションの低減方法である。イタレーションは想定内のイタレーションと想定外のイタレーション

の二つに分類することができる。想定内のイタレーションには製品性能や設計品質といった目標を達成させるために有効な設計情報並びに設計知識の交換が含まれる。想定内のイタレーションを低減させる方法は、イタレーションの回転速度を上げること、又はイタレーションの回数を減少させることである。想定外のイタレーションは設計上の単純ミスといった作業品質によるイタレーションである。単純ミスを完全に取り除くことは困難であるが、単純ミスは早期発見と早急な対応の施策を講じることである。

ii)　人間の行動特性

人間の行動特性は、設計プロセスの資源を無駄に消耗させ、遅延やコスト増加そして製品性能や設計品質低下を引き起こす原因となりうる。人間の行動特性は以下の四つの要素で構成される。

①学生症候群的な意識：
納期ぎりぎりまで着手せず、最終的には時間的圧力の中で集中作業が行われる。結果として納期が遅延するか又は成果物の品質が低下する。

②自己の予防線の意識：
早期に作業が終了した事実は自己防衛が働き、自己に対して予防線が張られる傾向になる。作業が早く終了して浮いた時間は結果として無駄に消費される。

③掛け持ち作業の同時性：
一人の人間が複数の作業を同時進行させることである。リソース競合によりクリティカル・パス手法が効かないことになる。掛け持ち作業は集中力が分散し、生産性が低下するとの指摘がある。

④パーキンソンの法則に基づく行動意識：
不必要な仕事はいくらでも作り出せるという組織内力学である。計画予算を全て使い尽くすまで消費される。

iii)　設計知の不十分な活用

設計知の不十分な活用とは、企業間、組織内、又は設計者等によって知識ベースの共有化並びに再利用が十分行われていない事態である。組織内に必要な設計知識が断片化して存在し、適切な時間と適切な場所に配置されていない、あるいは組織構造上の見えない壁の問題によってそれらが共有化されないことがある。また知の場では異質の知を排除する可能性が多分にあり、そのため新たな知の形成が起きないことがある。結果的に多くの設

計知識が意識的に、そして組織的に活用されないまま埋もれてしまう。設計知識は再利用されることで付加価値収益として形成される。

iv)　不知の不知

不知の不知は、知らないのにその通り知らないとは思わない、あるいは知らないのに知っていると思うことである。つまり知っていると思っているが、実際には知らなかったという意である。これは自己の不知を自覚していないことであり、例えば、知ったかぶり、思い込み、勘違い、誤った先入観等の人間の知的行動特性が含まれる。不知の不知は、こういった自己の不知を自覚していない事態を指す。自己の不知を自覚しないまま、その通り行動すると設計プロセス上のリスクを招来する。

10.2.2 設計プロセス管理の枠組みの提案

本研究では、設計負荷を含む設計プロセスの挙動を掌握し、設計負荷を低減させるための設計プロセス管理の枠組みを提案した。この枠組みによって設計プロセスの効率化と改善を図ることができる。この枠組みは計画、実行、評価、改善のサイクルから成り、以下の方式を持つ。

i)　イタレーションを含む設計プロセスの定式化と数値解析

DSM(Design Structure Matrix:設計構造マトリックス)を応用した一対比較行列によって、イタレーションを含む設計プロセスを定式化させた。この定式化により、イタレーションを生じさせるタスクが設計プロセスに与える負荷を定量的に解析することができる。そしてタスクの代替策をシミュレーションすることで、設計負荷低減の効果を評価することができる。本モデルの既知応用分野は米国の宇宙航空産業並びに自動車産業等である。本研究ではそのモデルを半導体設計分野に初めて応用したものであり、その有用性が確認された。本方式の詳細は第 4 章で述べた。

ii)　リスク評価フレーム

リスクを経済的損失量として定量的に求め、評価するフレームである。リスク・モデルから導かれるリスク算出式に i)の定式化を代入することで、活動時間と活動コストそれぞれのリスクの経済的損失量を求めることができる。本フレームを多様な設計工程モデルに適用することで、活動時間と活動コストそれぞれのリスク値のトレードオフ検討が可能となる。本フレームはリスクに対する実践的な洞察力を提供し、設計プロセスの策定並びに変更修正への意思決定を支援するものである。イタレーションを定式化できる設計プロジェクトという条件下、本方式は確率論的な意思決定が行えるという点で従来の決定論的な

方式より優れている。本方式の詳細は第 5 章で述べた。

iii)　価値のダイナミック・モデル

所要時間と所要コストの二律背反の問題に対する解決策として、リーン思考を拠り所とした価値のダイナミック（動的）・モデルを提案した。リーン思考は価値を生まない活動を排除し、価値を最大化させる活動を促進する。この価値の最大化のねらいが結果として所要時間と所要コスト双方を同時に効率化させる。価値のダイナミック・モデルは、メーカ視点と顧客視点、そしてそれらを統合した視点から捉えた製品価値の、プロセス量（所要時間と所要コスト）に対する変化を表現したものである。つまり、設計プロセスの効率化を検討するための指標として、所要時間と所要コストの他に、製品価値に注目したものである。これを運用することで、無駄な工程や過剰機能・過剰品質の工程の存在箇所を明確にできる。現在まで過剰機能・過剰品質の影響について定量的に評価することは難しいとの指摘があったが、本理論的モデルはそれを可能とした。本方式の詳細は第 6 章で述べた。

iv)　タイム・マネジメントの統合アプローチ

本アプローチは、製品の設計開発期間を短縮させることをねらいとしたタイム・マネジメント方式であり、三つの手法で構成される。一つはスケジュール管理手法である。プロジェクト・マネジメントで一般的に利用されているガント・チャート、PERT(Project Evaluation and Review Technique)と CPM(Critical Path Method)といったツールを使用する。事前に予定をどの様に割り振るのか、実際にどの様に仕事を進めるのかといった計画・実施・修正といったサイクルである。二つ目は CCM（Critical Chain Management: クリティカル・チェーン管理）を利用する。制約理論(Theory of Constraint:TOC)を拡張させた管理手法であり、人間行動の特性を考慮し、バッファという概念を用いて無駄な時間を削除する。三つ目はアウトソース化である。これは使える時間を増やすこと、又は他人の時間を分けてもらうことである。これで限られたリソース問題を解放させ、所要期間を削減させる。

本統合アプローチを実設計プロジェクトに適用し、伝統的手法とのベンチマーク対比を行った。結果として総所要期間を 413 日から 348 日に短縮でき、従前比 15.7%のスケジュール短縮の効果が得られた。アウトソース化という条件下でも CCM を活用できるという点で、CCM の応用領域を拡張させたと言える。本方式の詳細は第 7 章で述べた。

v)　知識ベースの構築

本論で提示した知識ベースは、設計対象物に関する知識と設計過程に関する知識を融合させたものである。設計対象物に関する知識には、設計対象物そのものに関する知識と設計対象物のモデル表現・生成・変更・利用に関する知識が含まれる。設計過程に関する知識には設計を進める際の意思決定に向けた判断基準、そして設計の具体的な手順に関する

知識が含まれる。知識ベースの構築により、設計作業の効率化並びに設計品質の安定を実現できる。すなわち、知識ベースが定着すれば共有化が進み、新規からの設計作業を減らす可能性を持つ。そして再利用化が進めば、設計開発コストの低減と設計開発時間の短縮につながる。更に知識ベースは共有化と再利用を通じて継続されることで、機能や性能面の設計品質が安定してくる。製品開発における知識ベースの構築は構築しない場合と比して、設計プロセスの効率化と設計品質向上の点で有益であることが確認された。知識ベースの構築とその応用については第 8 章、第 9 章で議論した。

10.2.3 設計プロセス管理論の含意

本研究論文は、企業の製品開発において効率化と改善に向けた管理手法を提供することを目的としている。特に本論で提案する手法とその実証例は、製品の設計開発プロジェクトの責任者にとって有益な知見と言える。

プロジェクトの責任者は、実践の現場では設計開発における所要コスト、所要時間、そして製品仕様の目標を設定する。プロジェクト責任者はこの目標を達成させる為に、様々な手法を現在まで利用してきた。本論で提案した管理手法は、こういったプロジェクトの責任者が利用してきた従来の手法を否定するものではない。むしろこの従来の手法を統合し拡張させたアプローチである。このアプローチは、従来の手法では取り上げられてこなかった問題点を解決する手法を含んでいる。本項では現在の状況を踏まえ、その拡張されたアプローチについての含意を述べる。そこには本アプローチの有効領域、適用対象者、本管理手法の実施上の障壁と対策が含まれる。

i)　本論で提案する手法の有効領域

本研究論文は、設計プロセスにおける所要時間と所要コストの二つの領域で生じるデータ統合させ、設計知識を活用した設計プロセス管理方法を提案したものである。データ収集については、伝統的な PERT (Project Evaluation and Review Technique)/ CPM (Critical Path Method)から始めている。そしてそこから拡張した手法としては、タスクという活動と所要時間を説明する DSM (Design Structure Matrix:設計構造マトリックス)が活用された。ネットワーク表現だけでなく、マトリックス表現を活用することでタスク活動の相互依存性が一覧性で理解できる。DSM については PERT というネットワーク表現の欠点を補完し、イタレーションという設計負荷の解析に効果を発揮できる。イタレーションを定式化できる設計プロジェクトであれば、本手法はイタレーションという設計負荷を低減できる点で従来手法より優れていると言える。

また DSM を活用したモデル手法と管理手法は、今まで半導体設計という領域で検証されることがなかった。しかし本論文で提案した方式（イタレーションを含むプロセスの定式化と数値解析、そしてリスク評価フレーム）は十分その実践領域で応用できることが確認

された。DSM については第 4 章、第 5 章にその有効性の詳細を議論している。

次に CPM 手法から、人間の行動特性を反映させた CCM (Critical Chain Method)を取り入れた。この人間の行動特性とは、遅延を引き起こす原因の一つである。この特性には四つの因子が含まれる。学生症候群、自己の予防線、掛け持ち作業、パーキンソンの法則である。これらは CPM では解決できない因子である。本論で提案したタイム・マネジメントの統合アプローチはこの CCM 並びに伝統的なスケジュール管理の他、アウトソース化を適用した方式である。この方式によって適用従前と比べスケジュール短縮効果が得られた。本方式の手法と有効性については第 7 章で詳細を述べた。

更に設計プロセスの効率性を向上させる知識ベースの設計支援を、管理の枠組みに取り入れている。この知識ベースは、設計情報と設計知識を概念化し体系付けしたものである。この概念と体系付けに基づいて構築される知識ベースは、従来の知識ベースが抱える共有・再利用性の欠如という問題点の解消に貢献すると言える。なぜならばこの知識ベースが定着すれば共有化が進み、設計過程において新規からの設計作業を減らす可能性を持つ。そして再利用化が進めば、設計開発コストの低減と設計開発時間の短縮につながるからである。

知識ベースは設計プロセスの効率化だけではない。知識ベースの活用は共有化と再利用を通じて継続されるので、機能や性能面の設計品質が安定してくる。つまりロバスト(robust)設計である。このロバスト設計ができておれば、設計開発工程のプロジェクト・リスクを低減することが可能となる。大規模設計の様に扱う知識の量が膨大で、複数の設計者が介在する場合においては、設計品質を確保する手段として体系化された知識支援の効果は大きいと言える。知識ベースの構築とその応用については第 8 章、第 9 章で議論した。

ii)　本論で提案する手法の適用対象者

本研究論文で提案する手法の中核となる適用対象者は、製品設計に携わる技術者、開発企画担当者、プロジェクト管理者である。そして製品設計開発の動向に関心を持つ一般企業経営者、ベンチャー企業家、投資家は、周辺対象者である。

特に以下の様な問題意識とその解決手法に高い関心を持つ方々には、本論で提案した管理手法は有益である。

- 従来の設計プロセス管理では、企業の視点でどの様に所要コストと所要時間を改善させるかという対症療法的な手法に焦点が当てられていた。そして製品対象物に対しては顧客視点で製品価値が議論されてきた。従って、因果連鎖を考慮した原因療法的な手法による観点から、そして企業と顧客を統合した視点から、設計開発工程を管理することが行われてきたとは必ずしも言えない。本研究で提示しているダイナミックの価値モデルは、その観点と視点で従来の設計プロセス管理で提供できなかった知見を提供する。

・ 従来のスケジュール管理手法では、設計プロセスで生じる手戻り修正、設計情報と設計知識の交換といったイタレーションを表現する手段が講じられてきたとは言えない。設計プロセス中のイタレーションによるスケジュール遅延やコスト増の問題を解決したい技術者、そして管理者には、本研究で提示しているモデルは、解決に向けた拠り所を提供できる。

・ 従来のプロジェクトの管理手法では、設計知識の体系化に基づく知識ベースの効用や設計知識の多様性について言及されることはなかった。本研究で提案する設計プロセスにおける設計知識の管理と活用により、設計プロセスの効率化とリスク低減が実行可能となる。本論で議論した設計知識は、再利用可能な IP (Intellectual Property)として体系化された知識ベースである。

・ 製品開発に従事する設計開発者は設計プロセスの所要コスト、所要期間を見積もる上で、必ずしもプロジェクトの不確実性について注意を払ってきたとは言えない。本研究ではこの不確実性から導出した経済的な期待損失をモデルとして定式化した。このモデルは従来手法の確率分布以上の情報を提供し、意思決定を側面から支援する。

iii)　本論で提案する手法の実施の上での障壁と対策

本論で提案した手法の実施には、そのプロジェクト計画を策定する上で幾つかの障壁が生じることと推測される。その障壁事項と対策について以下に述べる。

・従来手法からの延長

大規模で複雑化しているシステムを持つ製品開発のプロジェクトにおいて、目標となる製品性能を犠牲にせずにコストとスケジュールの管理を行うことは、困難な活動になってきている。この様な状況下で従来の管理手法を継続することは、正当化できるとは言えない。それでは従来とは異なった新しい手法を求めることを前提とするなら、どの様な障壁が生じるのであろうか。もしその新しい管理手法が管理者にとって容易に理解できる領域のものではなく、そして特殊な専門性を要するのであるなら、その手法の適用は困難となろう。またその導入と適用に大きな費用負担を強いられる可能性もあろう。この様な場合、新しい管理手法の適用は簡単に正当化できない。

そこで本研究で提案している管理手法は、伝統的なプロジェクト管理手法の欠点を補いながら拡張させている。つまり一般的なプロジェクト管理者は、基礎的な知識から出発できる。そして拡張された方式は理解し難い手法ではない。更に実践現場の設計開発技術者やモデル開発者からのデータ収集は現場を観察することから始めれば良く、モデル構築と評価解析は比較的容易である。本管理手法は従来手法の延長にあり、そして運用が容易であることを周知させることが求められる。

・合理的な説得材料

革新的な管理手法やその支援ツールは、簡単に受け入れられるとは限らない。その理由の一つは、意思決定者は常に革新に対する価値を見いだすことができるとは限らないからである。その意思決定者が、従来の知見とノウハウによって形成された固定観念と成功体験による先入観を持つ場合にはそれは顕著である。

従って、本論で提案する管理手法やその支援ツールの導入にあたっては、従来の手法と定性的にそして定量的に対比結果を指し示す必要がある。合理的な説得材料を事前に準備することが求められる。

10.3　今後の展望

次に本研究に関する今後の展望について全体論として述べる。

i) モデルの課題

本論で議論したイタレーションのモデルは、一対比較行列によって定式化されている。このモデルは、イタレーションが実行される度に相互依存関係にあるタスク全ての反復修正作業が同時並行的に生じること、そして反復修正の作業量はイタレーション毎に減少していくことを想定としている。しかし実際の現場においては、必ずしもイタレーションが実行される度にタスク全ての作業が実行されるわけではない。更に作業量はイタレーションが実行される度に減少することはあっても、線形性に従って減少するとは限らない。つまりこの仮定条件は実現象に基づいて構築されているが、厳密に言えば精度の面で完全に一致するとは言えない。従って、モデルの改善についての検討は引き続き行う方向である。

ii) 一般論化

本研究論文で提案した管理手法は、設計工学、プロジェクト・マネジメント、リスク・マネジメント、経営管理論といった理論から派生したものである。そしてその管理手法の有効性の検証には幾つかの半導体設計開発プロジェクトが適用されている。しかしながらその管理手法は、半導体設計開発という限られた個別領域において実証されたものにすぎないとも言える。その為には、半導体という対象物を離れて抽象化させ、洗練された一般論として昇華できるかという問題が残されている。従って半導体以外に当てはまる領域又は製品群を設定し、広くその有効性が適用されるのか追試を検討したい。そして本論で提案した管理手法を引き続き拡張し、一般論化していく所存である。

iii) 企業への貢献期待

設計開発上のプロセスの管理サイクルの枠組みによる知見は、今後ますます大規模化と複雑化していくプロジェクトに利用することができる。本研究論文で示した具体的事例と実証内容は特に、実践の現場にいるプロジェクトの管理者並びに設計技術者に対して、設計プロセスの重要性を改めて認識させることにつながるものである。本論の設計管理プロセスの管理サイクルによって蓄積される知識は、顧客視点の価値と企業視点の価値双方を反映させた効果的な製品開発につながり、企業の競争力に貢献すると期待できる。

iv) 潜在的な人間の知的行動問題並びに組織問題

本研究で提示した管理手法を適用し、その成果があったとしても、別の新たな問題点が生じる可能性がある。例えば本管理手法を適用することで、潜在的な人間の知的行動問題や組織問題を露呈することも予想される。設計は人間の知的行為故に人間の問題に関与しているからである。設計プロセスは管理の枠組みによる締め付けが継続的に大きくなれば、創造力やイノベーションへの活力に負の影響を与えかねない。従って、今後は人間の知的行動や組織に動機付けを与える仕組みの検討が課題となろう。

謝辞

私は 1995 年に日本には存在しない半導体関連の事業を行うためにベンチャー企業を創業しました。しかしながら、もの造りや経営についての理論も知識も持ち合わせていませんでした。もの造りや経営に関する常識すら無かった気がしています。その為にもの造りと経営の本質の一歩目から、あらためて全てを問い直すことから出発しなければなりませんでした。この早稲田大学大学院アジア太平洋研究科での研究生活は、私にそういった機会を提供してくれた数少ない貴重でかつ素晴らしい社会であったと実感しております。アジア太平洋研究科の先生皆様には大変お世話になりました。ここに厚く御礼申し上げます。

尚、本論文の執筆にあたり、数多くの方々からご指導とご鞭撻を頂いております。アジア太平洋研究科の同級生、黒須誠治研究室の皆様、そして大江建教授、藤田精一教授、神奈川大学の松浦春樹教授からは、幅広い視点から数多くのご意見やご助言を頂き大変感謝しております。私が日常生活で係わったことがない様々な領域からのお話しは刺激的で新鮮なものでした。そして最後に、豊富な知見と設計論に対する情熱を供えられた黒須誠治教授からは多くのことを教えて頂き大変感謝しております。重ねて御礼申し上げます。

2006 年 2 月 17 日

山本　靖