LC-007

システムレベル設計に基づくシーケンス制御システムの段階的詳細化設計

System Level Design of Sequence Control Systems by Stepwise Refinement

小林　憲貴* 山崎　亮介* 吉田　紀彦* 楢崎　修二†
Kazutaka Kobayashi Ryosuke Yamasaki Norihiko Yoshida Shuji Narazaki

1　まえがき

システムレベル設計 [1, 2] は,SoC 設計の効率向上を目的としており, 低い抽象度のシステムを高い抽象度のモデルから設計し, 徐々に抽象度を落としていくことで抽象度の差を埋める.

我々は,SoC 設計以外にも抽象度の低い設計手法を用いている分野に対して, システムレベル設計を応用することで, その設計効率を向上させることを目指している.

本研究では, 応用分野としてシーケンス制御システム [3, 4] を選択した. その理由は, システム設計の抽象度が低いこと, システムの構成要素が SoC と似ていることが挙げられる.

本稿では, 実際のシーケンス制御システムを例題とし, システムレベル設計を用いて様々な抽象度のシーケンス制御システムのモデルを設計する手法を示す.

以下, 第 2 節ではシーケンス制御システムについて説明を行う. 次に第 3 節はシステムレベル設計を応用する場合の手順を実際のシーケンス制御システムの例題を用いて説明する. 最後に第 4 節で本稿のまとめと今後の課題について述べる.

2　シーケンス制御システム

シーケンス制御システム [3, 4] は自動制御の一種であり, 入力に応じて予め定められた順序に従い, システムの状態を制御するものである. 主な制御として以下の 4 つが挙げられる.

順序制御 機器 A の始動の次に機器 B を始動する
条件制御 事象 A が生じた後に機器 B を始動する
時限制御 機器 A の始動後, 3 秒後に機器 B を始動する
計数制御 製品 A を 1 0 個製造後, 梱包する

シーケンス制御システムはソフトウェア (以下 SW) による制御と Programmable Logic Controller (以下 PLC) による制御で構成される.PLC については後ほど述べる. ここでシーケンス制御システムの構成と組み込みシステムの構成を比較すると, 組み込みシステムが SW とハードウェア (以下 HW) で構成されることから,PLC にシステムレベル設計 [1, 2] を応用できればシーケンス制御システム全体にシステムレベル設計を応用することが可能になる.

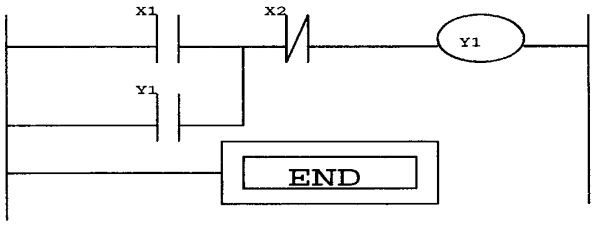

図 1: ladder logic:self-maintenance circuit

2.1　Programmable Logic Controller

PLC[5] とは, 一種の HW であり, 主に入出力端子･CPU･メモリ・電源・通信制御部により構成される. つまり, ノイマン型コンピュータと大差はない. しかし, 現場環境に合った HW であること, シーケンス制御システムに適したプログラムが実行可能であるということが PLC が使用される理由である.

このシーケンス制御システムに適したプログラムが実際にシーケンス制御を行う部分であり,PLC 上で動作する SW である. 現在, シーケンス制御システム設計では Ladder Diagram(以下 LD) が最も使用されているため本研究でも LD を採用した.

HW と PLC との違いは 2 つある. 一つは,PLC では, プログラムの繰り返し実行が暗黙の了解となっていること. 二つ目は, クロックが無いことである.PLC では, プログラムの 1 回の繰り返しを 1 スキャンと定義し, スキャンで同期をとっている.

PLC の他の特徴は,bit 型以外のデータ型を扱えないことである. これは PLC がシーケンス制御専用の HW であるため,SW もシーケンス制御に特化しており, 複雑な演算が必要ないからである.

2.2　Ladder　Diagram

LD[3] は PLC で実行されるプログラムである. リレーは入力, コイルは出力を表し, リレー同士の接続関係で論理演算を表す. 図 1 に自己保持回路の例を示す. 自己保持回路とは, 開始条件により, 状態の保持を行い, 解除条件により, 状態を変化させる回路である. 左肩に開始条件 (図中の X1) , 右肩に解除条件の否定形 (図中の X2) を持つ.

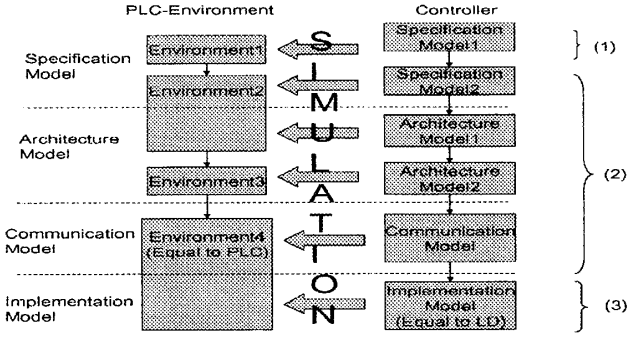

図 2: outline

*埼玉大学 Saitama University
†長崎大学 Nagasaki University

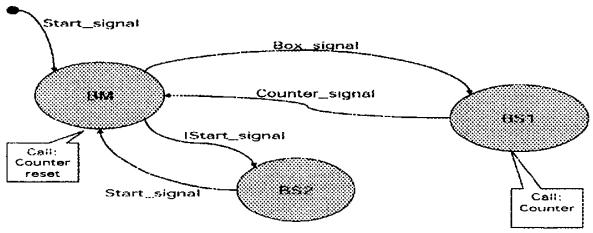

図 3: state diagram:Box Conveyor

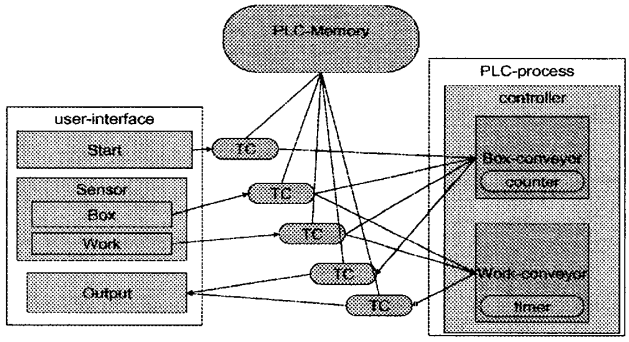

図 4: Specification Model for PLC

このLDは　Y1=(X1 OR Y1) AND $\overline{X2}$　を意味する.

LDは,電気信号の流れをそのままプログラミングすることに相当する.否定入力の使用にはシステムの停止の実現,インタロックなどが容易に構築できる利点がある.短所としては,読み取ることが難しいこと,データの変化するタイミングがわかりづらいことである.これは,LD設計には経験が必要であり,順序が読みづらく,並列処理と考えられるLDの段（一つの出力に対する横一連の接点の結合）は実際にはPLCによって処理順序が違うからである.

これまで述べたように,LDはSoC設計に使用する論理演算と似ている.しかし,相違点もある.第一に,パイプライン処理が記述できない.LDは並列処理で考えるが,PLC内部では高速な逐次処理を行うことで,みせかけの並列処理を実現している.つまり,LDは厳密には逐次処理であるため,パイプライン処理は不可能となる.第二に,リレー,コイルのような入力・出力素子の他に,シーケンス制御で頻繁に使用される処理（タイマー,カウンタ,信号微分,内部変数）も素子化されてることである.

LDではこれまで述べた素子を組み合わせて接続することで,目的のシステムの動作を表現する.

3　システムレベル設計の応用

3.1　SpecC設計理論

本研究ではシステムレベル設計のシーケンス制御システムへの応用を行うわけだが,まず始めに採用したシステムレベル設計の方法論について説明する.本研究ではSpecC設計理論[7]を採用した.SpecC設計理論を採用した理由は,SpecC設計理論は一貫的で段階的な理論が確立されているので,シーケンス制御にシステムレベル設計を応用する場合の問題箇所を見つけ易く,問題内容も分かり易いためである.

本研究では,SpecC設計理論の各段階でのモデルを基に応用を行う.SpecC設計理論の各モデルの特徴を以下に記す.

- 仕様モデル：システムの機能を忠実に再現したモデル
- アーキテクチャモデル：実装するアーキテクチャに仕様モデルの機能を割り当てたモデル
- 通信モデル：アーキテクチャ間の通信を実装できる抽象度に落としたモデル
- 実装モデル：実際のアーキテクチャに対応するモデル

3.2　研究概要

システムレベル設計を応用してシーケンス制御システムを設計する.応用の際に,第2.1節,第2.2節で説明したHW・SoC設計とPLC・LDの違いに対応しなければならない.違いは以下の通りである.

- 繰り返し処理：1回の繰り返しを1スキャンと定義
- クロックが無い：スキャンでの同期
- パイプラインの不可：並列処理の逐次化
- 構成素子：特殊な機能の素子化

また,LDの構造や,実行のタイミングの詳細も理解しておく必要がある.しかし,提案手法を用いることで,それらを気にすることなく,LDの設計が可能である.

図2に概要を示す.設計するシーケンス制御システムは2つに分けて考える.1つは最終的にLDに相当する制御部,もう1つはテストベンチの役割をするPLC環境である.応用する手順を以下に示す.番号は図2と対応している.

(1) PLCを考えない抽象度の高いシーケンス制御システムのモデルの作成
(2) SpecCの設計理論に沿ってモデルの抽象度を落とす.抽象度を落とす部分を実際のPLC環境・LDに近づける.
(3) 最終的にLDと対応する制御部と実際のPLCを同じタイミングで入出力・通信を行うPLC環境を設計する.

このように,抽象度の高い正当なプログラムを手順に従って抽象度を落としていくことで,設計の初期段階からLDを作成するよりもバグが少ないプログラムが作成可能である.

3.3　例題設計

実際のシーケンス制御システムにシステムレベル設計を応用しながら,提案手法の説明を行う.今回,設計するシーケンス制御システムの概要を説明する.

コンベアとセンサを二つずつ用いる.コンベアの一つは箱を搬送し,もう一つのコンベアは箱に詰める物体を搬送する.センサの一つは箱搬送コンベア上の所定の位置に箱が到達したことを検出する.もう一つのセンサは,物体搬送コンベアから物体が落下したことを検出する.

システムの動作の流れは,以下1~5の繰り返しである.

1. 箱センサが箱搬送コンベア上で箱を検出
2. 箱搬送コンベアはストップし,物体搬送コンベアが作動
3. 物体センサが物体がコンベアから落下するのを検出
4. 物体搬送コンベアが1秒間だけストップ
5. 落下個数が規定数に達すると箱搬送コンベアが作動

3.4　仕様モデル作成

最も抽象度の高い仕様モデルを作成する.この段階では,PLCを考えないモデル（仕様モデル1）を作成し,そ

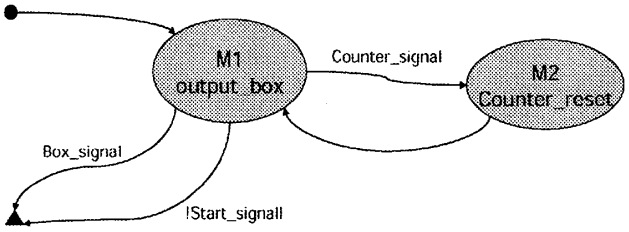

図 5: Archtectuer Model:BM state

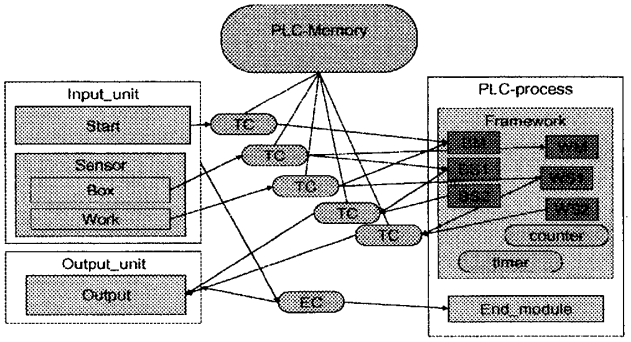

図 6: Archtectuer Model

のモデルを元に PLC メモリを考えたモデル（仕様モデル 2）を作成する.

3.4.1 仕様モデル 1

最初に PLC を考えないモデルを作成する.

- 制御部

  仕様は, 状態遷移図でモデル化可能である. 各状態を子モジュールとし, 子モジュールの状態遷移を制御するモジュールを親モジュールとする. この時,LD 固有の素子（タイマー, カウンタ, パルス）の機能はチャネルで実現することで, 部品の独立性を高めることができ, また, モデルの構造を LD の構造に近づけるのが容易になる.

  図 3 に箱搬送コンベアの状態遷移図を示す. この時, 箱搬送コンベアを表すモジュールが親モジュールとなり, 状態 [BM][BS1][BS2] が子モジュールとなる.

- PLC 環境

  入出力を行うモジュール (user-interface) とシーケンス制御を行うモジュール (PLC-process), その 2 つを繋ぐチャネル (Abstract Channel:AC) で構成される.

3.4.2 仕様モデル 2

次に,PLC メモリを考えたモデルに変化させる.

- 制御部

  user-interface・PLC-process と同様にチャネル名と返り値の型の変更だけでよい.

- PLC 環境

  これまでの PLC 環境に実際の PLC 内のメモリに対応する PLC-Memory を作成する.user-interface・PLC-process と PLC-Memory 間の通信の抽象度の差を吸収するため Transducer-Channel(TC) も作成する (図 4 参照).

3.5 アーキテクチャモデル作成

この段階では, 制御部は,LD の特徴である " 並列処理の逐次化 " を実現するための下準備を行う. 具体的には,「状態の細分化」と「状態遷移の削除」である.PLC 環境では,PLC の特徴である " スキャンによる同期 ", " 繰り返し処理 " を実現する. その結果, 制御部の構造の抽象度を落としたモデル (アーキテクチャモデル 1), 全体の動作順序の抽象度を落としたモデル（アーキテクチャモデル 2）が作成可能である.

3.5.1 アーキテクチャモデル 1

最初に動作順序の抽象度を落とし易くするために, 制御部の構造の抽象度を落としたモデルを作成する.

- 制御部

  仕様モデル 2 では制御部は状態遷移図でモデル化されているが, その各状態の内部を細分化し内部状態を作成する. 細分化の基準を以下に記す.

1. PLC-Memory, 特殊素子へのチャネルアクセスの個数分の内部状態を作成する.
2. 内部状態同士の関係が逐次型である場合は, 元の状態内にさらに状態遷移図を作成する.
3. 2 以外の場合は, 状態は一意でなければならないので内部状態に優先度を付ける. この優先度は状態の処理内容によって決まる.
   - 優先度の低い状態は処理内容毎にまとめて階層を作る.
   - 優先度の高い状態内では元々の状態の動作を基本状態とし, 例外処理として優先度の低い状態の動作を行う.

作成した内部状態にモジュールを割り当て, 内部モジュールとする.

図 3 の状態 [BM] を細分化した図を図 5 に示す. 状態 [BM] では, データの書き込みを行うチャネル [output_box] と, カウンターのリセットを行うチャネル [Counter_reset] が存在する. よって, 状態 [BM] 内に 2 つの内部状態 [M1][M2] を作成する. この 2 つの状態は逐次実行ではないので, 優先度を付ける. この場合は状態 [M1] が優先度が高い. 状態 [M1] を基本状態とし, 入力 [Counter_signal] で状態 [M2] へ遷移する.

- PLC 環境

  PLC 環境は変更する必要がない.

3.5.2 アーキテクチャモデル 2

次に全体の動作順序の抽象度を落としたモデルを作成する.

- 制御部

  制御部の動作順序を LD の動作順序に近づけるために, モジュールの平坦化を行う. 制御部の親モジュールを削除し, 子モジュールの全てを並列化する. 親モジュールを削除することで状態遷移も削除されてしまい, 正しい動作順序が保てない. そこで, 状態遷移図と同じ実行順序を保つために, 子モジュールに動作を行うかの条件判定を追加する. 条件の追加方法を以下に記す.

1. start_flg と end_flg を用意し, 各モジュールは start_flg が立っていたら動作を行う.
2. start_flg の立ち上がりは前提条件として (end_flg=0) とする.
3. 他の条件は (その状態へ遷移するための入力)
4. 初期状態の場合は,(他の状態へ遷移する入力の否定) を start_flg の立ち上がり条件に追加する
5. 内部状態に優先度がある場合は, 1 つの (優先度の高い条件) に全ての (優先度の低い条件) を AND で組み合わせる

6. start_flg の立下りは,end_flg が 1 の時, または,3〜5 以外の場合である.
7. end_flg の立ち上がりは,start_flg が 1 , かつ, その状態から遷移する入力がある場合である.
8. end_flg の立下りは,start_flg が 0 の時である.

以上により, 親モジュールの削除が可能である. 図3 の状態 [BM] を表す子モジュールの start_flg の条件は (Start_signal=1) と (Box_signal=0),(Counter_signal=1) である.

- PLC 環境

次に PLC 環境の動作順序を PLC の動作順序に近づける. 具体的な方法を, 以下に示す.

(1) PLC-process 内にフレームワークを提供する.
(2) 制御部の終了判定を行う End モジュールを作成し,PLC-process 内に追加する
(3) user-interface を input-unit と output-unit に分割する.
(4) input-unit からの終了信号を End モジュール・output-unit に送信する End-Channel(EC) を追加する.

図 6 にアーキテクチャモデルの全体図を示す.

3.6 通信モデル作成

ここでは, モデルの通信部分の抽象度を落とす.

- 制御部

制御部の変更部分は, 各内部モジュールがローカル変数を使用している場合, そのローカル変数を PLC の内部変数に置き換える. この時, データアクセスのタイミングが変化するため, その調整が必要となる.

- PLC 環境

PLC 環境の変更部分は,TC を PLC-process,input-unit,output-unit 内部に埋め込みむ. これにより, 各モジュールが PLC-Memory に直接アクセスできるようになる.

3.7 実装モデル作成

最後に LD の " 並列処理の逐次化 ", " 構成素子 " を実現することで,LD に対応する制御部を持つ実装モデルを作成する.

- 制御部

最初に, 制御部作成に必要な LD の素子に対応するライブラリを用意する. このライブラリは以前に我々の研究室で作成したライブラリを使用する [6].

次に内部モジュールをプラットフォームに置き換え, 各プラットフォーム内の処理をライブラリを用いて設計する. これまでの内部モジュールの動作は, 一つの bit 型の出力値を決定するものであった. 言い換えると, 内部モジュールの動作は, 現在の出力値を保持, または, 変化させることである. これは LD の自己保持回路で実現可能である. 自己保持回路の組み方を以下に記す.

- 子モジュールの start_flg の条件を自己保持開始条件とする.
- 子モジュールの end_flg の条件を自己保持解除条件とする.
- 1 つ出力変数が複数回表れる場合, 出力変数を異なる内部メモリに変更し, 最後に内部メモリの論理演算により出力を決定する.

最後に, 実行順序を定める. プラットフォームは逐次実行型なので, 要求仕様からフローチャートを作成し, そのフローチャートを基にプラットフォームの実行順序を決める.

- PLC 環境

PLC 環境については, 変更する必要はない.

以上により, システムレベル設計によるシーケンス制御システムの設計が可能である. 提案手法の実装モデルを元に作成した LD を図 7 に示す.

(株)Inter Designe Technologies 社 [8] の VisualSpec2.0 を使用して, 各段階のモデルを作成し, それぞれでシュミレーションを行い, モデルの正当性を確認した.

その後, 図 7 のラダー図を Klöepper und Wiege Software GmbH 社 [9] の PLC プログラム開発パッケージ MULTIPROG wt で作成し, 同社の ProConOS シミュレータでシミュレーションを行った. その結果, 要求を満たすシステムであることを確認できた.

4 おわりに

本稿では, システムレベル設計のシーケンス制御システム設計への応用を検証した. 応用する場合の手法を示し, 手法に従い実際にシーケンス制御システムを作成した.

今後の課題を以下に記す.

- PLC では,PLC 間,PLC ／周辺機器間の通信が可能であるため, この通信を実現し, 大規模なシーケンス制御システムを構築する
- システムレベル設計を行うことで, どれほどの効率アップが望めるのか検証する.
- システムレベル設計した場合とそうでない場合のモデルの比較
- SpecC から LD への変換
- 仕様モデルから LD までの自動化

参考文献

[1] Daniel D.Gajski,Jianwen Zhu,Rainer Dömer,Andreas Gerstlauer,Shuqing Zhao,"SpecC, 仕様記述言語と方法論",CQ 出版社,2000
[2] Gerstlauer,Dömer,Peng,Gajski,"システム設計,SpecC による実現",SpecC Technology Open Consortium,2002
[3] 熊谷 英樹,"シーケンス制御プログラム定石集". 日刊工業新聞,2003
[4] 熊谷 英樹,"ゼロからはじめるシーケンス制御". 日刊工業新聞,2001
[5] 関口 隆志,"新しいプログラマブルコントローラのプログラミング,IEC61131-3 による効率的プログラミング", コロナ社,1999
[6] 小林 憲貴, 吉田 紀彦, 楢崎 修二,"システムレベル設計のシーケンス制御システムへの応用",FIT2004 論文集,Vol.1,pp.229-232,2004
[7] http://www.ics.uci.edu/-specc/
[8] http://www.interdesigntech.co.jp/vn1209.htm
[9] http://www.kw-software.de/Japanese/

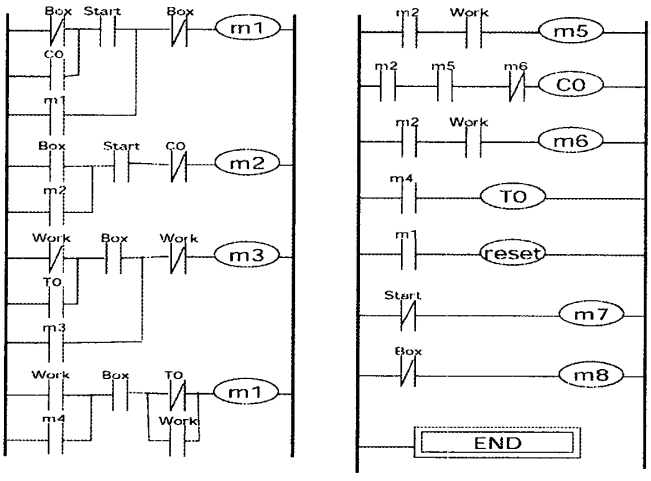

図 7: Ladder Diagram:test case