2008－EMB－10 (16)
2008／11／28

組み込みシステムの再利用開発を支援するソフトウェア・ハードウェアモジュール群の相互依存性ビジュアル表示管理法

神戸英利[†] 三井浩康[††] 小泉寿男[††]
[†]三菱電機株式会社 [††]東京電機大学理工学部

現在，組み込みシステムの需要は増大し，かつ開発期間は短期化の傾向にあるため再利用が必要不可欠となっている．しかしながら、再利用対象の既存のハードウェアやソフトウェアの構成管理が不十分であると，再利用部品適用の際に他のどのモジュールに影響を及ぼすかがわからなくなってしまい，新たなバグを増やす要因となってしまう．本稿では、既存ソフトウェア・ハードウェアのモジュール間関係を可視化できる構成管理システムに夜、効果的な再利用を実現する手法を提案する．本稿ではさらに、本手法を用いた開発例に関する評価について述べる．

Method of management to support the re-usebased embedded system by visualizing display of the dependency between modules of Software hardware

Hidetoshi Kambe[†] Hiroyasu Mitsui[††] Hisao Koizumi[††]
[†]Mitsubishi Electric corporation [††]Tokyo Denki University

The demand for embedded system is growing with less time allowed for development. It is becoming increasingly inevitable that existing hardware and software system will be reused. However, if existing reused software hardware is not well managed or organized, nobody will be able to know which modules would be affected when other parts of reused software hardware are modified, resulting in new bugs produced. In this paper, we propose a method to accomplish the effective composition of software and software hardware function by managing dependency information among modules. A software hardware function means as selectable implementation of software or hardware by the system requirements. We have developed a configuration management system that visualizes mutual relations among existing software hardware modules and visually presents not only dependency among modules but also a list of reusable modules according to the developer's demand for reuse of modules. We evaluated the method by applying it to some actual developments

1. はじめに

近年、携帯電話、自動車、情報家電など組込みシステムの需要は急激に拡大している．また、機能の複雑化・高度化も進んでおり、システムの規模は飛躍的に拡大している．一方，製品のライフサイクルは短くなっており、短期間で新製品を出すためには，設計・開発にかける時間の短期化が求められている．

このような理由から、以前に開発した設計資産を再利用することなく組込みシステムを開発することはほぼ不可能な状況になっており、効率の良い再利用開発手法が求められている．

組込みソフトウェアの再利用を行う場合，過去の開発資産を有効に活用できるように管理することが必要である．ソフトウェアの再利用時に継承される情報不足や時間的制約などから，開発対象となるソフトウェアの範囲の分析漏れや，関係者間の情報伝達漏れ等による，仕様抜けや実装漏れが起こりやすくなる．

組込みシステムの新機種を開発する際には，前機種の機能を継承し機能アップして搭載するなど，過去の機能を再利用する率が高い特性がある (1)(2)．また組み込みソフトウェアの再利用開発においては，開発毎に開発者の入れ替わりが多く，開発者のスキルや経験の変動が発生するため，過去の開発ノウハウの伝承が難しい．

さらにソフトウェア規模の増大と高機能化により，再利用に際して再利用対象ソフトウェアの把握が難しくなり，修正したモジュールが他のモジュールにどのような影響を与えるかの判断も困難になってくる等の課題がある．

筆者らは既存ソフトウェアのモジュール構造とモジュール間関係を可視化できる構成管理システムのプロトタイプを開発し，再利用を支援する手法を研究してきた(3)(4)(5)．

しかしながら，昨今の組込みシステムにおいては，CPU の高性能化や，LSI の高集積化，高性能化の進展に伴い，音声や映像のコーデック，表示装置の一部機能，暗号の積和演算等，ソフトウェア，ハードウェアのいずれでも実現可能な機能が存在する．今までハードウェア（LSI、IP）の構成管理は，ソフトウェアとは別個に行われており、上記手法もハードウェアやソフトウェア，ハードウェアのトレードオフが必要なものの管理には対応していないという問題がある．これらについて，筆者らはハードウェアとソフトウェアが共存する組込みシステムの構成管理を提案した(6)．しかしながら、これまでの研究では、組込みシステムのハードウェア実装に関わるインターフェイス記述の抽出や，一つの機能に対して複数の記述ファイルが

存在するケースに十分対応しておらず，評価を行っていなかった．

本論文では、モジュール間依存性のビジュアル表示による組込みシステムのソフトウェア・ハードウェア再利用開発方式につき提案する．

本方式では、システムの記述に SystemC を用い，ソフトウェアとハードウェアを区別せずに同一言語で記述する．また各機能のソースファイル情報をもとに，それぞれのモジュール間の依存関係を可視化し，モジュールをブロック図表示して，その上に重畳表示する．ハードウェア，ソフトウェアいずれの実装か確定していない機能は，ハードウェア，ソフトウェアの協調設計へ受け渡しトレードオフを行う．

本方式の有効性を評価するため、携帯電話のプロトタイプ開発(6700 個のモジュールのいくつかをハードウェアで実現する設定にする)における JPEG 圧縮エンジンの開発に適用し，その評価を行った．本稿では本手法の内容とその評価結果について論じる．

2. 組込みシステムにおけるモジュール管理の現状と課題

組込みシステムは、開発規模拡大と開発期間短期化への要求から、製品開発において既存のプラットフォームを導入するなど、以前開発したハードウェアやソフトウェアの資産を再利用して開発することが多い．ソフトウェアならびにソフトウェア・ハードウェアいずれの実装か選択可能な機能の開発を行う場合，以下のような現状と課題がある．

(1) 組込みソフトウェアモジュール管理の現状と課題

組込みソフトウェアの利用開発においては，大規模化すると共に，一つの開発に複数の企業が参画することが多くなっている．これに伴いかつての開発経験者を再利用開発時に再度確保することが難しくなりつつある．そのため，再利用するソフトウェアの内容を十分把握しない状態で，次の開発を始めるケースが多い．再利用にあたっては，残されたドキュメントや情報から内容を把握する必要がある．しかし入手した情報が不足したり，整理がされていなくても，再利用開発では，状況をそのまま受け入れざるを得ない．例えばドキュメントが不十分であると，利用したい機能の関数を頼りに，詳細はソースファイルの内容を確認する必要が出てくる．ソースファイルの数が膨大になるとソースファイル自体を探し出すのに時間がかかる．

また，新規参画の開発者にとって再利用するソフトウェアの構造を短期間で理解し把握することが難しい．

さらに，修正範囲の調査ミスが生じやすい．再利用するソースファイルを調査する際に，他のソースファイルに影響が出ないかどうか確認が必要である．利用予定の関数名等を頼りに関連先を順に調査するが，ソースファイルの数が膨大になると，調査に時間がかかることから，見落としや内容の確認不十分等のミスが生じやすくなるという問題がある．これを解決するために筆者らは構成管理システムを研究してきた．

構成管理システムは，組込みシステムのソースファイルを関連解析して，ソフトウェア全体の構造とファイル間の依存関係を抽出する．

図 1 に関連解析システムの構成を示す．図において，①ソースファイルと，その格納場所の情報，ビルド時に使用するモジュールの構成情報を示すプロジェクトファイルを再利用データとして集め，関連解析にかける．

②関連解析部で，各ソースファイルに含まれる関数名とハードウェアインターフェイスを記述する識別子をもとに，Paser にて全ソースファイル間の関数およびハードウェアインターフェイスレベルの参照関係

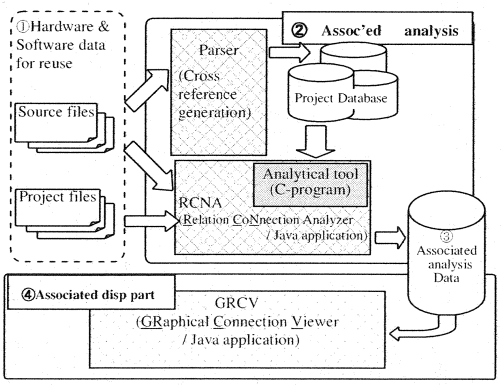

図 1: Management of system configuration

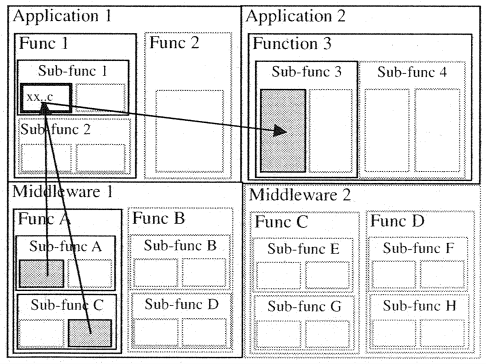

図 2: Block diagram and Inter-modular dependency

を解析する．Analytical tool でブロック図情報を生成しする．上記②で生成したブロック図情報と関数およびハードウェアインターフェイスレベルの参照関係の情報を，ソースファイル名をキーにして一つにまとめ③関連情報とする．

④関連表示部により，開発者に対してブロック図形式でモジュールを表示し，その上にそれぞれの参照関係を矢印等で重畳することにより，各モジュール間の関係をビジブルに提示することを可能とする．図２に関連表示部でソフトウェアをブロック図形式で表示し，その上に参照関係を重畳表示する例を示す．図において，Sub-func１内で開発者が指定した xx.c ブロックと，その中で定義される関数もしくはハードウェアインターフェイス等と参照関係がある他のブロック間に，矢印で参照関係が表示される．

これを解決するために筆者らが研究してきた構成管理システムでは，ソースファイルの格納場所やソフトウェアモジュールのビルド時の情報等をもとに，ソフトウェアの全体構造を一つのデータにまとめ，ブロック図で表示し，さらに各ソースファイルに含まれる関数をもとに，ソースファイル間で関数レベルの参照関係を解析し，ブロック図形形式でモジュールを表示した上に，それぞれの参照関係を矢印等で重畳することにより，各モジュール間の関係をビジブルに表示する．

(2)　ハードウェア化，ソフトウェア化不確定モジュールの扱いの課題

組込み機器の開発においては，ハードウェアで実装する機能，ソフトウェアで実装する機能，ハードウェア，ソフトウェアのいずれでも実装可能で，開発時の条件により実装方法を選択する可能性がある機能に別れる．このなかで，ハードウェア，ソフトウェアのいずれでも実装可能な機能は，抽象度の高い状態で設計を始め，詳細化の過程でソフトウェア開発か，ハードウェア開発か判断し，ソフトウェア開発を選択した場合は，ソフトウェア開発のシステムへ，ハードウェア開発に決定した場合は，ハードウェア開発のシステムへ受け渡す．

ハードウェア開発化が決定され，ハードウェアの開発システムへ受け渡すまでの機能の記述は，ソフトウェア的記述となることから，ソフトウェア開発選択の際は，そのまま利用できることが望ましい．前回ハードウェアで実装されたものであっても，最初はハードウェア，ソフトウェア共通レベルの記述をもとに始める必要がある．

これらの機能は，高抽象度の記述，低抽象度の記述等，１つの機能に対して，同一の機能を示すが記述のされ方が異なる複数の記述データが存在することとなる．通常のソフトウェアが１つのソースファイルで記述されている場合でも，複数のソースファイルが存在して，利用時の条件により対応させるソースファイルを選択しなければならない．

このように複数の記述が存在する機能をブロック図上で必要な抽象化レベルの記述で表示し，インターフェイスを重畳できるようにする手法がハードウェア，ソフトウェア再利用開発時への適用に必要である．

3.　モジュール間依存性のビジュアル表示管理法

3.1　組込みシステムの構成

図３に本方式の基本フレームワークを示す．図において上半分は，従来の設計方法の例，下半分は本提案の方式を示す．

組込みシステムでは，図の左側に示すように，機能の実現においては，ハードウェア，ソフトウェア，ハードウェアとソフトウェアのどちらでも実装可能な３種類に分けることが出来る．

一般的には，従来法の例で示すように，ハードウェアとソフトウェアは個別の開発環境で開発されるため，最初にハードウェア化するかソフトウェア実装するかトレードオフ検討して，割振りを決定してから開発を進める．

3.2　ソフトウェア・ハードウェアの記述データ

組込みシステムにおいて，ソフトウェアは C 言語系で記述されることが多く，ハードウェアは，VHDL，Verilog HDL 等の記述や回路図が使用される．

ハードウェアとソフトウェアのどちらでも実装可能な機能は，実際には実装方法決定後に設計されるので，最終的にはソフトウェアかハードウェアどちらか一方の設計データのみが残る．再利用検討するときに前回と異なる実装方法を選択した場合は，前回の資産は再利用できずに新規設計となる．

3.3　ハードウェア・ソフトウェア構成法

本提案では，ソフトウェアとハードウェアを共通に記述できる SystemC を利用し，ソフトウェアとソフトウェア・ハードウェアいずれによる実装も選択可能な機能を同一に扱う．

従来法では，最初にハードウェア化するかソフトウェア化するかトレードオフ検討し，決定した後はそれぞれの開発は独立して進める．本提案の方式では，ソフトウェアとソフトウェア・ハードウェアいずれによる実装も選択可能な機能は，SystemC で記述し，ソフトウェアと同一の同じ扱いをする．SystemC によるハ

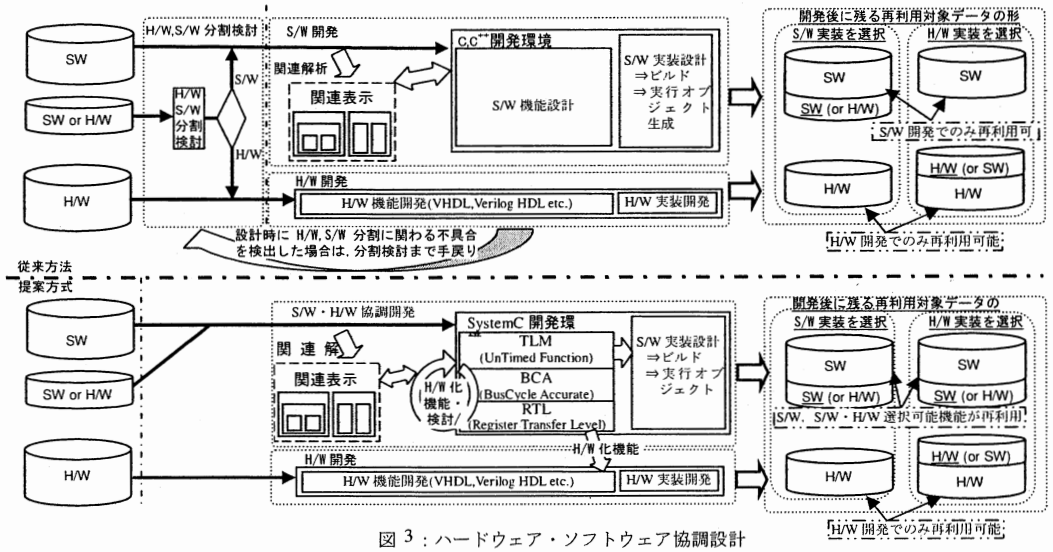

図 3：ハードウェア・ソフトウェア協調設計

ードウェア，ソフトウェア協調設計を利用して，ソフトウェア・ハードウェアいずれによる実装も選択可能な機能を開発する．また構成管理システムのビジュアル機能を使用して抽象度を下げた詳細設計を進め，ハードウェア化が確定したら，ハードウェアの開発環境へ設計データを渡し，その後はハードウェアとして開発する．

ソフトウェアとして実装する場合は，ビルドを実施してソフトウェアの実行ファイルを得る．

ソフトウェア・ハードウェアいずれによる実装も選択可能な機能に関して，本提案の方式ではソフトウェア資産としての記述データが必ず残る．

3.4 ハードウェア・ソフトウェア協調設計

ハードウェアとソフトウェアのどちらで実現するか未決定のモジュールをハードウェアとソフトウェアのどちらで実現するか決定するためにハードウェア・ソフトウェア協調設計が必要となる．以下，過去の資産を再利用して新たな開発を実施する例を説明する．

(3) 再利用開発の流れと協調設計の関係

再利用資産に含まれるモジュールはハードウェア、ソフトウェア、どちらか未決定のものに分かれているが，製作段階で修正ファイルが未決定のモジュールだった場合には，ハードウェアとソフトウェアに振り分けるための協調設計を行う．

(4) ハードウェア・ソフトウェア協調設計

ハードウェア・ソフトウェア協調設計はシステムの機能の中で、どの部分をソフトウェアで実現し、どの部分をハードウェア(専用回路)で実現するかを最適化するというアプローチである．近年の組込みシステムは計算量の多いマルチメディア処理などを要求される場合があり、プロセッサの性能や消費電力を抑える目的で専用の回路が設計されることがある．このときハードウェア・ソフトウェア協調設計でシステムの設計をすることが有効である．

(5) ハードウェア・ソフトウェアの振り分け

ハードソフトどちらで実装するか検討対象になった機能は，検討開始時点では，抽象度の高い SystemC 言語で記述されており，ソフトウェアと同じに扱うことが出来る．検討の進捗と共に，時間の要素や，クロック動作のようなハードウェア的条件の記述が含まれる抽象度を下げた BCA や RTL の記述に切り替え，性能等，トレードオフ条件を満たすかどうか確認して，どちらで実装するか決定する．トレードオフのパラメータとしては，ゲート規模，性能，コスト，消費電力等がある．この際，SystemC の記述範囲にいる限り，構成管理システム上で管理する事が可能なため，他のソフトウェアモジュールと同等に扱うことができる．最終的にハードウエア化が確定した機能について，SystemC で設計したデータを，ハードウェアの実装設計へ受け渡し，VHDL 等のハードウェア記述へ変換してハードウェアを開発する．

4. 再利用作業のフロー

4.1 関連解析と関連情報表示による再利用の流れ

再利用するもととなる製品のハードウェア化対象機能を含むソフトウェア記述データをプロダクト A とする．これを再利用して，新たに開発する製品をプ

ロダクトBとする．開発フローを図4に示す．

(6) プロダクトB仕様と再利用データの準備

まず開発するプロダクトBの仕様を準備する．次に再利用するプロダクトAのハードウェア化対象機能を含むソフトウェアを一式準備する．これを図4のaで示すようにプロダクトAで使用したソースファイルを全てコピーしてプロダクトBのシステム記述データのもととする．

(7) 関連解析

コピーしたプロダクトBのソースファイル，プロジェクトファイルが読み込まれ関連解析を実行すると，各ソースファイル内で定義されているハードウェアインターフェイスや関数のシンボルをもとに，ファイル間のシンボルレベルの関連情報が解析される．同時にモジュールの階層構造を整理して抽出した，ブロック図のベースとなるモジュールの構成データが生成される．図4のbに示すように生成された関連情報ならびに，モジュール構成のデータを関連解析データとして格納する．関連解析の範囲をインターフェイスのシンボルレベルまでとすることで，ファイル間の関連性確認までは，ハードウェアとソフトウェアを同等に扱うことが可能となる．

(8) 調査

開発者は図4のcに示すように，関連解析データからシステム構成をブロック図表示して，プロダクトBの仕様を実現するために新規追加が必要なモジュール，修正して再利用するモジュール，流用するモジュールの調査を行う．

(9) 関連情報の表示

図4のdで示すように，前記の調査で探し出された修正対象となる再利用モジュール，およびこれを修正した際に影響が及ぶ周辺のモジュールの情報を表示して確認を行う．

修正して再利用するモジュールに対しては，関連を持つモジュール全てがブロック図レベルで関連情報が表示される．そこで開発者は，ブロック図上で関数やハードウェアインターフェイスレベルの入出力関係の表示，修正前後の比較表示をさせて，関連を持つモジュール全てにつき修正の要否を確認し，開発/修正の必要なソースファイルの範囲を抽出する．

(10) 設計製作

開発者は，設計製作では既存モジュールの修正と新規追加，ソフトウェア・ハードウェア協調設計の3つの作業をする．

修正再利用するモジュールは，図4のeに示すように関連情報を確認しながらソースファイルを修正し，図4のfに示すプロダクトBのソフトウェアとして格納する．

次に新規追加においては，プロダクトBに新規追加するソフトウェモジュールをブロック図レベルで追加し，機能に基づいて周辺のモジュールと関連付けを行う．新規追加分のソースファイルを作成しプロダクトBのソフトウェアとして格納する．

ハードウェア化検討対象のモジュールは，図3の下中央のソフトウェア・ハードウェア協調開発に示すように，抽象度が高い記述から低い記述に詳細化しながら，協調設計を進め，最終的な選択の判断を行う．最終的にハードウエアとして開発することを決定した場合は、ハードウエア開発ツールへ設計データを送りハードウエアの開発を行う．ソフトウェアで実装する結論となった場合は，そのままビルドしてソフトウエアの実行オブジェクトを得る．

(11) ビルド

開発者は，ソフトウェア実装対象の記述につき，コ

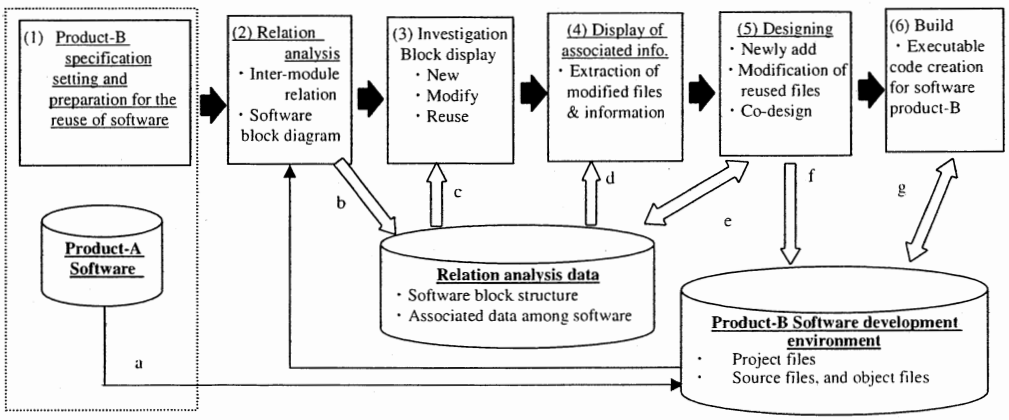

図4：Work flow of module visualization method to support reuse

ンパイル，リンクを実行して，ソフトウェアをビルドし，プロダクトBのソフトウェアの実行コードを得る．

5. 実装結果と評価

3 章で提案したモジュール間依存表示法に関し，4 章ではそれを実現する構成と実装について述べた．本章では実装結果と評価を述べる．本章では，図 4 で述べた流れに沿って実装結果と評価を述べる．

(1) プロダクトB仕様の設定と再利用ソフトウェアの準備

再利用するプロダクトAとして，ソースファイル数，約 7000 ファイル，プロジェクトファイル数，約 520 ファイルから構成されているソフトウェア群を用意した．このプロダクトAを再利用し，一部のソフトウェア・ハードウェア機能ならびにソフトウェアを修正し，プロダクトBを開発する例を評価対象とした．

(2) 関連解析

最初にコピーし準備した修正前のプロダクト B のソフトウェに関連解析を実施した．関連解析では，所定の解析機能が実現されること，および解析時間の性能が実用上の評価ポイントとなる．関連解析を実施して以下の結果を得た．

関連解析は約 64 分（クロスリファレンス解析：30 分，関連データ生成：34 分）で終了した．

この結果，解析されたシンボル数は全部で 686,423 個，ブロック図は 9,654 個（データの大きさ約 1.8MB），生成された関連解析データの大きさは約 143MB となった．

解析したシンボル数と生成されたブロック図の数は，設計者が手作業で個別にデータの内容を見てチェックするには量が多すぎ，これらを関連表示部によってビジブルに表示することの必要性が認識された．

(3) 調査

開発者は初期プロダクト B を調査して，流用部分，修正部分，新規追加部分を決めるために，関連表示ソフトウェアを使って，関連解析ソフトウェアが出力した結果からソフトウェアブロック図を表示した．この表示結果例を図 5 に示す．例では，ハードウェア化の可能性があるブロックに H マークが表示され開発者に認識しやすい．表示の分解レベルは，階層化されたレベルで自由に表示可能であり，最小単位は，ソースファイル単位となる．ハードウェア化検討機能として，JPEG デコーダを対象とした．esrvstartup と EncConfig の 2 つの機能ブロックを対象とした．図 5 内の 2 点鎖線で示した部分を評価対象とした．

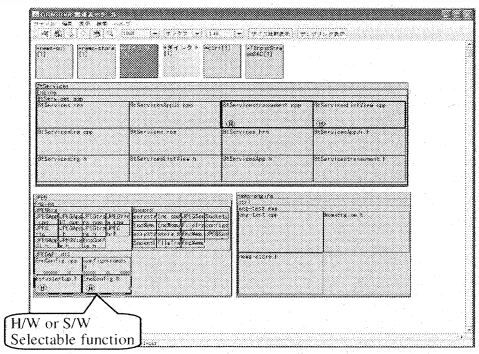

図5: Example of block display

(4) 関連情報の表示

開発者は修正するモジュールを抽出したあと，関連しているモジュールの調査ならびに開発要否の検討を行った．

ブロック図上でどの機能ブロックと関係を持つか関連の方向性含め開発者にイメージが理解しやすいことがわかる．さらに，JPEG デコーダと関連性があるブロック図のみをサンプリング表示した結果を図 6 に示す．チェック作業時に不要なブロック情報を隠すことで，たくさんの関連をもつ場合に確認がしやすいことが分かる．

ブロック図上で，ソースファイル単位まで詳細化して関連表示すると，実装レベルのチェックが可能である．図 7 に関数および入出力インターフェイスの関係をリスト表示した例を示す．中心の列に指定したブロックの関数、入出力インターフェイスがリスト化され，それぞれの関数やインターフェイスを選択すると，左に自分が参照する機能ブロック，ソースファイル、関数やハードウエアインターフェイスの情報，右には自分自身が参照されているブロック，ソースファイル，関数等情報がリストアップされて表示される．設計や確認時には関連先を探し出すことに時間と手間をと

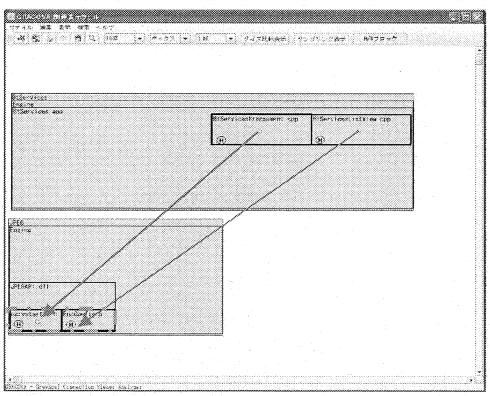

図 6: Example of H/W block display

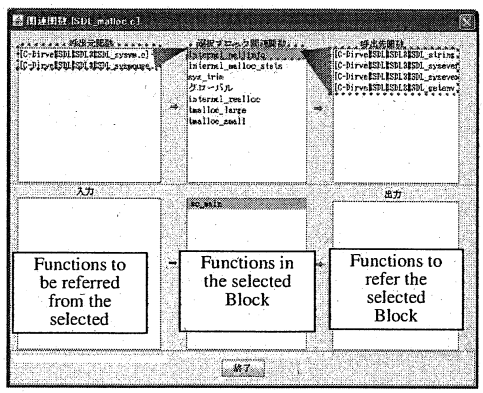

図 7 :Input/output display by function

られることから、これらのチェック作業が効率よく実施できた.

(5) **設計製作設計製作**

関連情報表示を利用して洗い出したハードウエア化対象機能のソースファイルを元に、ハードウエア化するかソフトウエア実装にするか判断するために、SystemC 上で抽象度を逐次下げて性能評価等の協調設計を進めながら、開発を進めた.

①ソースファイルへのリンク

修正するために抽出したブロック図からソースファイルへリンクして, プログラム内容の確認と編集が実施できることを確認した. 関連表示で指定したブロックのソースファイルを呼び出して編集する例を図 8 に示す. ソースファイルを探し出す手間が省けるため作業効率の向上が可能となった.

(6) **ビルド**

修正再利用するソフトウェア, ソフトウェア・ハードウェアモジュールならびにその影響を受け修正が必要な関連モジュールのソースファイルを修正し, ビルドを実施してプロダクト B のソフトウェアを製作することができた.

6. 考察

(1) ソフトウェア・ハードウェアの実装を選択する可能性のある機能を抽出, 検討する際に, 構成管理システムの適用が可能となったことから調査時間の短縮, インターフェイスの確認効率化を図ることが出来た. また, SystemC の各抽象度の記述を一元管理して, 各抽象度レベルの構成を短時間で間違いなく再現できるようになったことで, ファイルの取り違い等の人為的ミスを軽減することができるようになった.

(2) SystemC による記述を採用したことにより, ソフトウェア・ハードウェアの実装を選択する可能性の

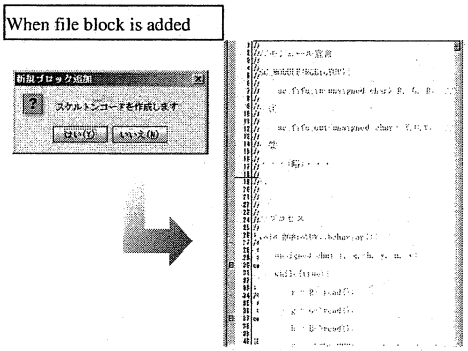

図 8 :Sourcefile link

ある機能をソフトウェアと共に一元管理することが可能となった. この結果, 本提案の構成管理システム上でソフトウェア・ハードウェアをブロック図表示し, インターフェイスをビジブルに確認することが可能となり, 再利用の際に効率的かつ見落としの少ない作業が可能となった. また, 協調設計により, ハードウェア・ソフトウェア分割をかなり詳細な設計まで進めて決定できるため, 無駄なマージンの確保が不要となり, 最適条件に近いトレードオフが期待できる.

(3) 今回は, トレードオフの項目を性能のみで評価したが, 実際のトレードオフでは, ゲート規模や, コスト, 消費電力等の条件を包含する必要があり本システム上での取り扱い方法を検討する必要がある.

(4) VHDL や Verilog HDL 等のハードウェア記述を関連解析できるようにすることで, 機能レベルの開発においては, ハードウェアとソフトウェアを一元管理しビジブル化する方法の実現が課題である.

7. まとめ

本稿では、協調設計により設計されたハードウェアで実現するかソフトウェアで実現するか未決定のモジュールを含んだシステムの再利用開発の方式について提案した.

この方式はソフトウェア・ハードウェア機能をソフトウェアと共通の記述を採用することでソフトウェアモジュールと同一に扱い, 構成管理システムを適用して構成をビジブルに表示すると共に管理することで, ソフトウェア・ハードウェア協調開発におけるトレードオフ検討を支援した. 今後, トレードオフ検討に必要な条件のデータを追加管理できるように改善し, VHDL 等ハードウェア記述言語にも対応できるようにすることで, ハードウェア開発における構成管理含めたシステム全体への適用を目指す.

参考文献

(1) 小池誠，並木美太郎，岩澤京子：分散環境における再利用性の高いオブジェクト作成を支援する共同開発環境の研究，情報処理学会 ソフトウェア工学会，Vol.1999，No.122-6 pp41-48 (1999-3)

(2) Bernard Coulange：Software Reuse Springer, usa (1997)

(3) 神戸英利，永松博子，三井浩康，小泉寿男：組込みソフトウェアの再利用を支援するモジュール間相互関連表示法，情報処理学会 組込みシステム・ソフトウェア工学 合同研究発表会 (2007)

(4) Hidetoshi Kambe, Hiroko Nagamatsu, Hiroyasu Mitsui, Hisao koizumi, Jun Sawamoto: A Method of Visualizing Inter-module Relationsto Support Reuse-based Embedded Software Development, 22nd International Conference on Advanced Information Networking and Applications, pp.598-605(2007)

(5) 鷲澤暢亮，神戸英利，小泉寿男：構成管理をベースとしたソフトウェアの再利用法，平成 18 年電気関係学会関西支部連合大会，G12-4，p.G289 (2006)

(6) 永松博子，神戸英利，三井浩康，小泉寿男：モジュール間依存性のビジュアル表示による組み込みシステムのソフトウェア・ハードウェア再利用開発方式 ，平成 19 年電気関係学会関西支部連合大会，G11-20，p.G269 (2007)