最新SoC設計フローと
組込み向けUMLの真の利用価値は？

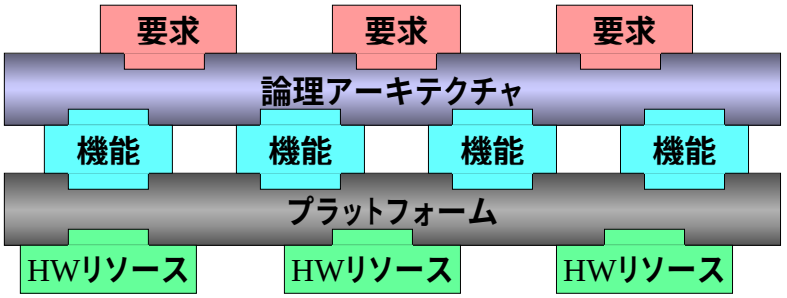
キャッツ株式会社
浅利　康二
asari@zipc.com

はじめに

- システムLSI（SoC）も大規模・複雑化し、モデリングが非常に重要になってきています。
- SoCとは、その名前のとおりシステムが1チップに搭載されているもの（System on a Chip）です。
- よってSoC設計の技術は、システム設計にそのまま利用できます。

要求、機能、HWリソースの分離

- 各オブジェクトの独立性を保つため、要求、機能、HWリソースの分離が必要。

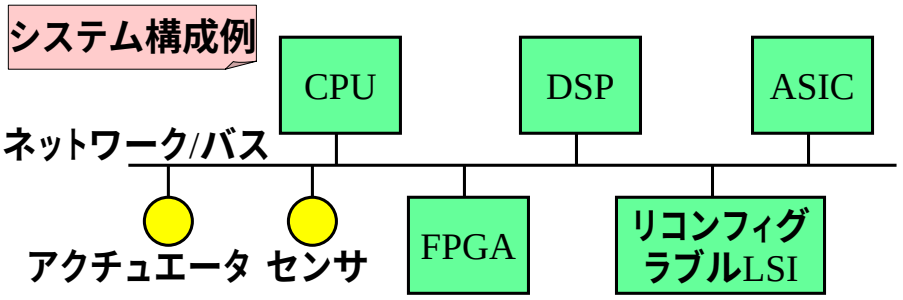


プラットフォーム

- UMLでは上位のモデル（PIM→PSM）はよく議論されている。下位のプラットフォームは、基本的に1CPUを前提。
- 複数CPUのシステムの場合はどうなるか？
  →並列性、時間同期などの問題が生じる。
- 上位で美しいモデルが出来ても、実装までの流れがよくないとせっかくの努力が水の泡。そこで、今回は下位のプラットフォームとモデル化しておくべき要素について少し整理してみる。

HWリソース

- LSIとしては、CPU、DSP、ASIC、FPGA、リコンフィグラブルLSIなどが存在。
- ネットワーク/バスもリソースである。
- これらのリソースが協調してシステム動作を行う。



LSIの種類

CPU： 制御が得意

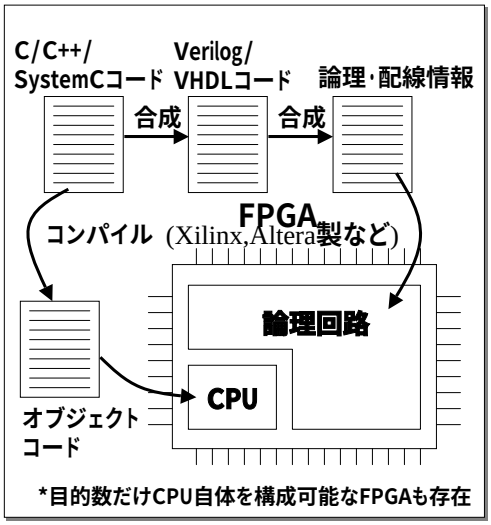
DSP： 信号処理が得意
SIMD： 1命令/複数データ処理のプロセッサ

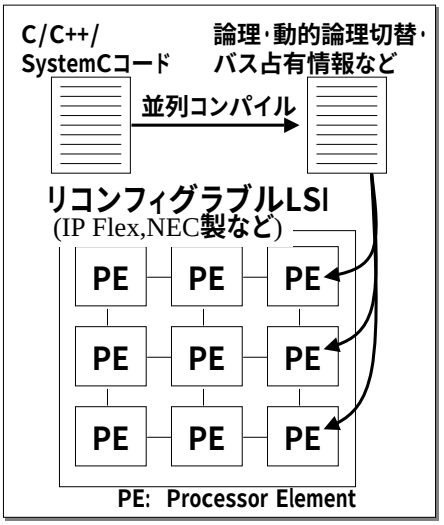
ASIC： 特化した処理をLSI化
(低消費電力、高速処理、低コスト化のため)

FPGA： 回路構成を静的に変更可能

リコンフィグラブルLSI：
回路構成を動的(リアルタイム)に変更可能

FPGAとリコンフィグラブルLSI





ネットワークの種類

- ネットワークを確保する方法としては以下がある。

(1) 時分割による定期的な通信路確保 (TDMA方式)

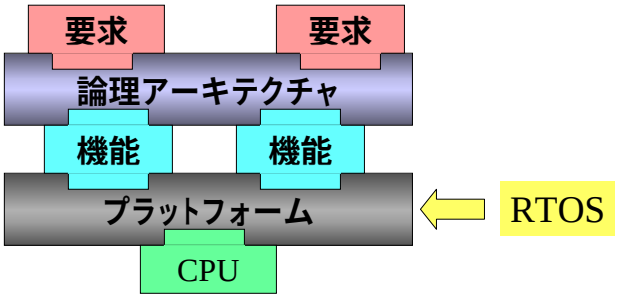
(2) イベントが生じた場合のランダムな通信路確保 (CSMA/CD方式)

多重化方式		TDMA	CSMA／CD
送信タイミング		一定間隔で定期的に送信	情報が発生したときに送信
ネットワークのクロック同期		必要	不要
通信スケジューリング		必要	不要
特徴	送信頻度が低い時	帯域利用効率が低い	帯域利用効率が高い
	送信頻度が高い時	帯域利用効率が高い 遅延が一定	帯域利用効率が低下 伝送遅延が大きくなる
適した用途		信号処理システム	イベント処理システム

(DesignWave Magazine July 2004)

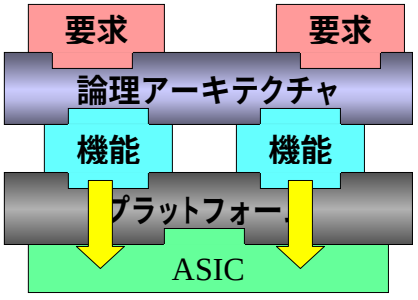
CPUなどの場合

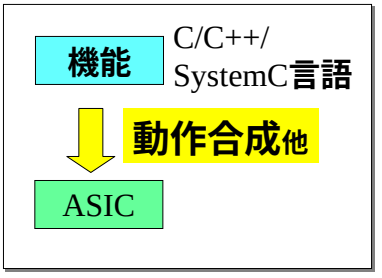
- CPUは逐次処理が基本。
- 並列性や同期、時間管理などはRTOSでサポート。
- 機能はタスクにマッピングされる。



ASIC・FPGAなどの場合

- 機能をそのままHW（回路）で作成。
- 並列性や同期、時間管理などはASIC内部で実現。



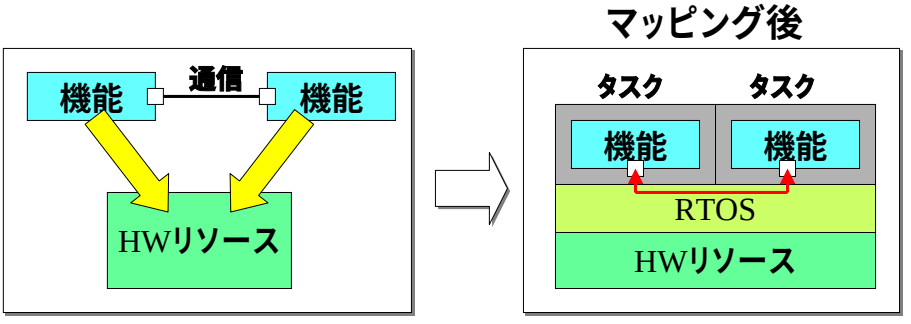


モデリングに必要な要素（並列性）

- 並行動作する機能は、リソースへのマッピング時に考慮が必要。
- 複数の機能を1CPUへ実装する場合には、別タスクへマッピング。（擬似並列）
- 複数の機能を複数のCPU、もしくはCPUと他のリソースへマッピングする場合は、完全並列が実現。

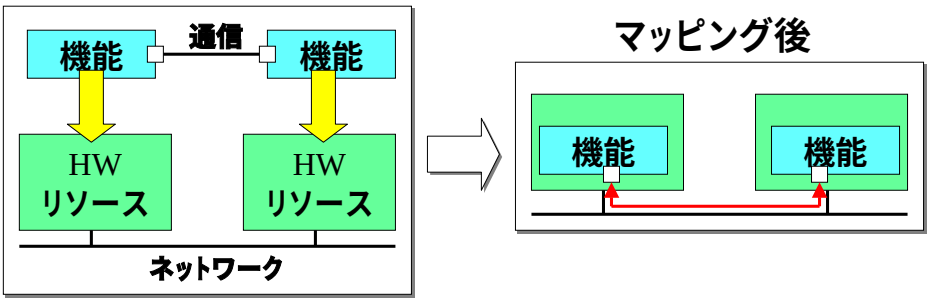
機能のCPUリソースへのマッピング

- 以下のように機能がCPUリソースにマッピングされると、並行処理が必要な機能は各々タスクにマッピングされ、通信はRTOSが行う。



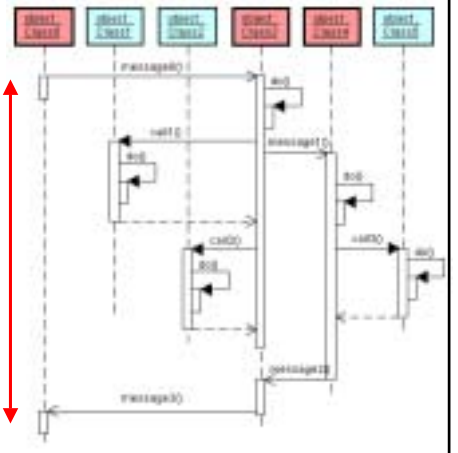
機能の複数HWリソースへのマッピング

- 以下のように機能が複数のHWリソースにマッピングされると、機能間の通信はネットワークにマッピングされる。



モデリングに必要な要素（同期・通信）

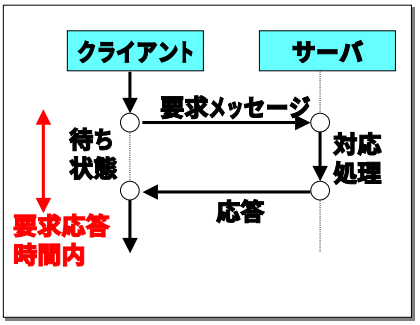
- 同期/非同期/一方向通信などの種別
- アクティブオブジェクト同士の通信は同期に注意が必要
- 要求処理時間



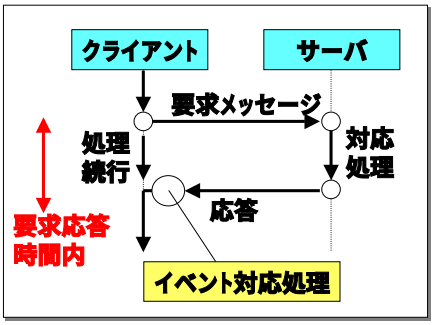
CORBAの例

- 以下はCORBA通信の例である。

(1) 同期呼び出し(Synchronous Call)



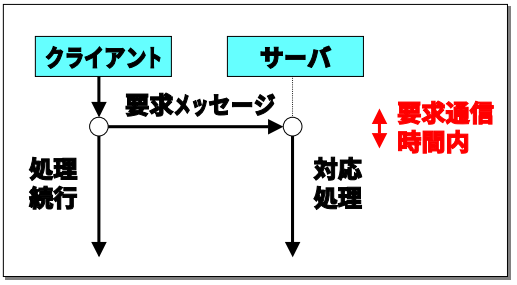
(2) 非同期呼び出し(Asynchronous Call)



CORBAの例

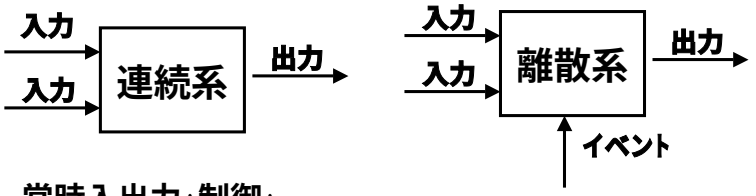
- 前ページの続き

(3) 一方向呼び出し(Oneway Call)



モデリングに必要な要素 (時間)

- 連続系と離散系は時間で同期をとる。
- 連続系の世界では時間は中心的な概念。

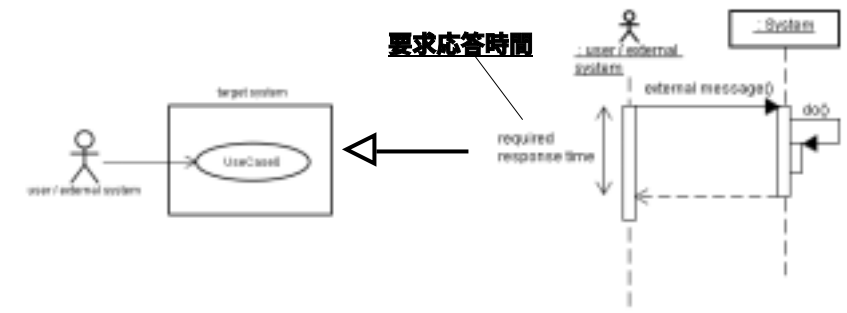


常時入出力･制御･
演算処理が行われる

イベントによって制御･演算
処理が行われる
クロックもイベントの一種

モデリングに必要な要素 (時間)

- 要求応答時間は仕様として必ず満たされなければならない。オブジェクト指向では非機能要件。



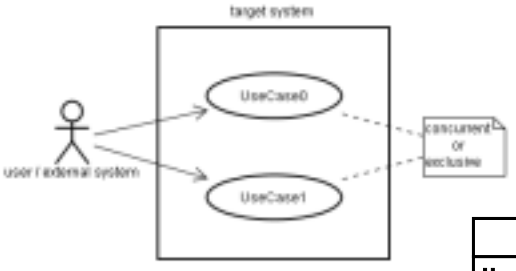
モデリングに必要な要素 (調停)

- HWリソース、ネットワークの調停
  HWリソースへの同時アクセス時の処置
  ネットワークの競合の解決

- 排他、時分割、重ね合わせ、複数リソース
  時分割の場合はスケジューリングが必要
  パイプライン処理などはその一種

モデリングに必要な要素 (調停)

- モードの違いなど、ユースケースが排他で実現される
  ものであれば、リソースの調停は必要ない。

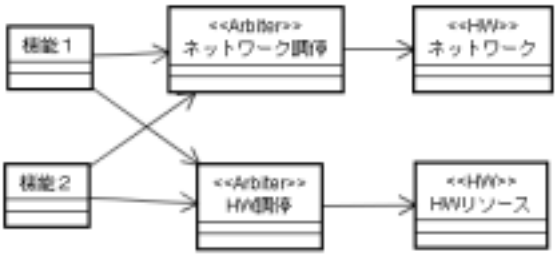


排他テーブル

	UseCase0	UseCase1	UseCase2
UseCase0	-	exclusive	concurrent
UseCase1	exclusive	-	concurrent
UseCase2	concurrent	concurrent	-

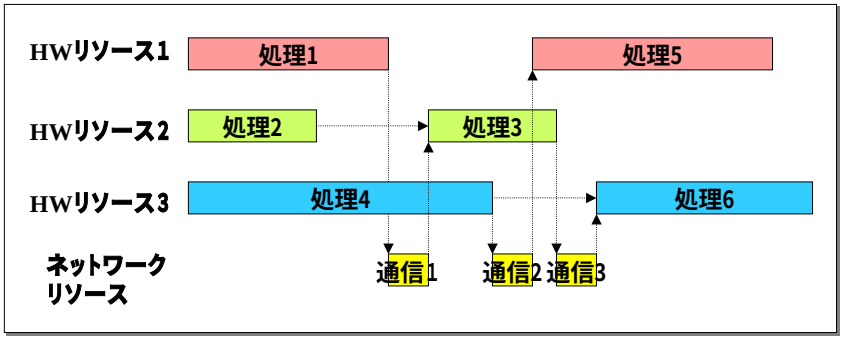
モデリングに必要な要素 (調停)

- 機能を複数リソースで実現することが難しい場合 (組込みシステムでは通常はそう) には、排他、時分割、重ね合わせなどの調停機構が必要となる。
  パイプライン処理は、時分割の特殊なパターンである。



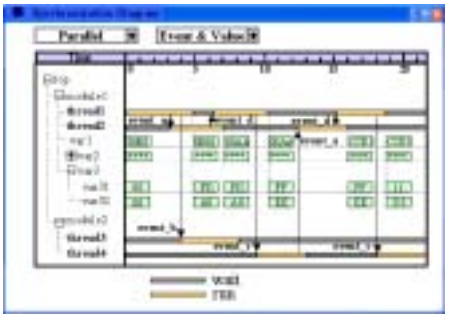
スケジューリング

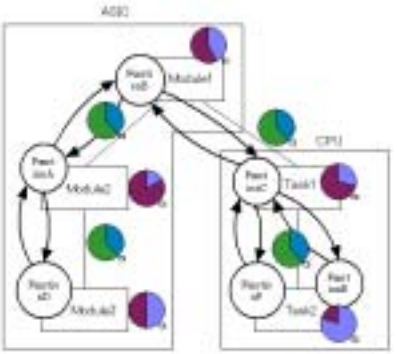
- すべての要求を満たすようにマッピングを行い、各リソースにおいて時分割で処理を実現するためのスケジューリングを行う。



マッピング

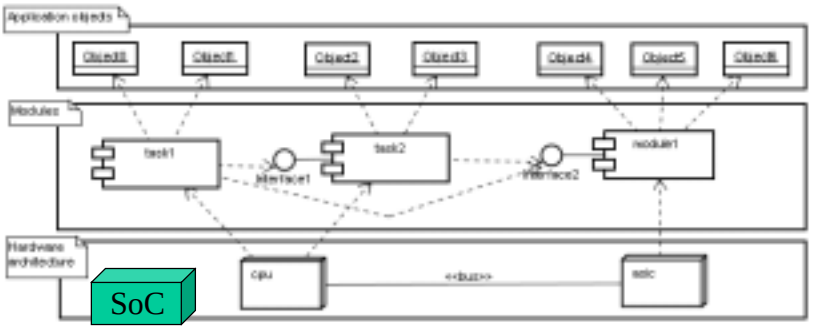
- 各スレッド（プロセス）の動作と同期を元に、近似解アルゴリズムを用いて機能のリソースへのマッピングを行う。これは静的な方法なので、この後、動的なシミュレーションによる検証が必要





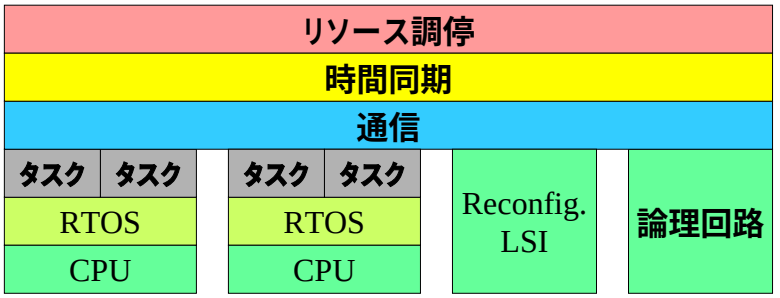
マッピング

- 各オブジェクトをマッピングする際に、それらの機能群が協調動作するためのフレームワークが用意されていると非常に効率が高いはず。SW/HW分割も容易になる？



プラットフォームには何が必要か？

- プラットフォームには、リソース調停、時間同期、通信などの機能が必要。



モデルの複雑度について

- モデルの複雑度が小さい方がよいモデルのはず。
  ある部分を修正することによる影響範囲も少ないはず。
  (仕様変更が容易)
  →テスト(検証)の工数に大きな違いが出る。

- クラス
  メソッド数、属性数、状態数、イベント数など
- 関連
  アクティブクラス間通信(非同期通信)は複雑など
- クラス数と関連数
- 時間制約の厳しさなど

UML for SoC フォーラム

目的：「UML for SoC」の標準化と普及促進

主な活動内容：

SoC 向け UML 拡張プロファイルの策定

2002年にキャッツ、日本ラショナルソフトウェア、富士通で設立。

2003年4月から7社が参加。（リコー様他の事例）

現在、OMGへ提案中。

事務局はキャッツ社が担当。

41st DAC(Design Automation Conference)

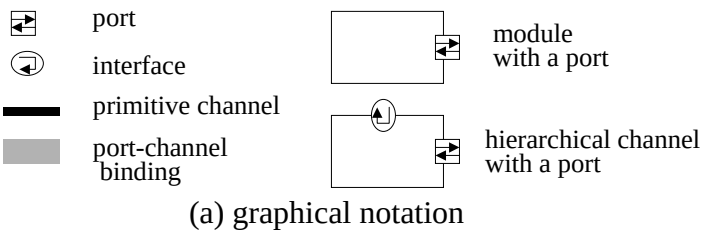
- **世界で最大のEDAツール展示会（米国サンディエゴ）期間中、USOC04ワークショップが行われた。**
  その中でUMLをSoC設計に利用する取り組みのプレゼンテーションが日本・欧州・米国からなされ、活発な議論が行われた。
- **キャッツ社からもUSoCF（日本）での活動内容の紹介と他1件の発表を行った。**

SystemC言語について

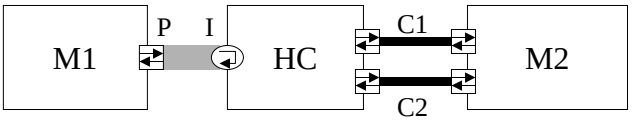
- 並列性、時間、同期、アクティブスレッド、ポート、チャネル（通信）の概念を持ち、システム動作を記述できる言語。
- シミュレーションが可能なため、システム動作の確認・検証を行うことができる。

SystemC表記法

- UML2.0での表記と類似。階層化も可能。



(a) graphical notation

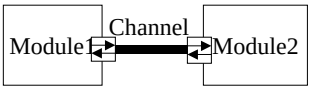


(b) example

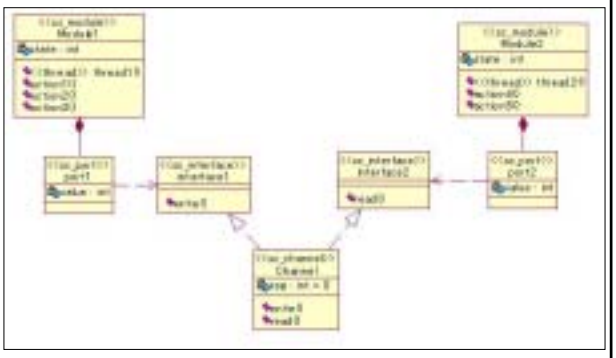
SystemCのメタモデル

- SystemCではアクティブオブジェクト同士の通信にはチャネルが定義される。

SystemC表記

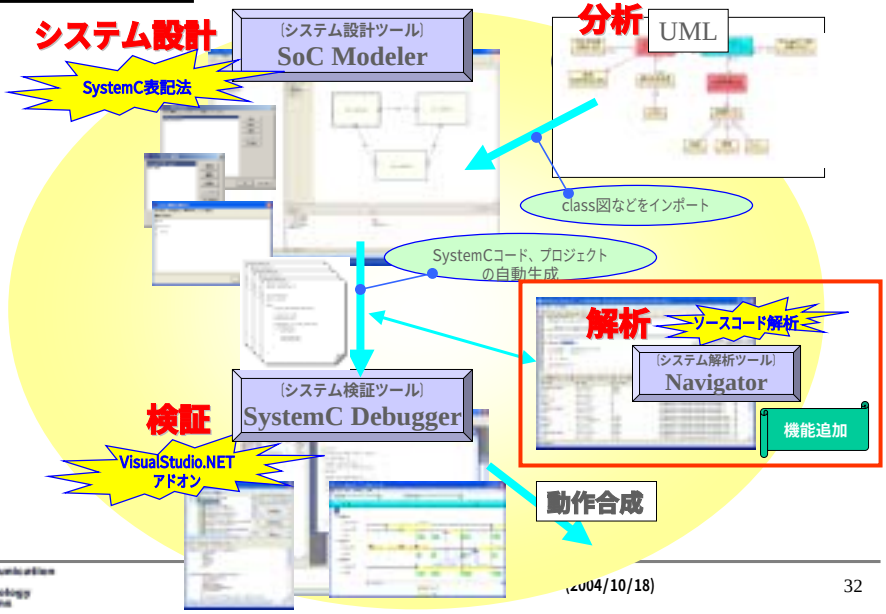


メタモデル

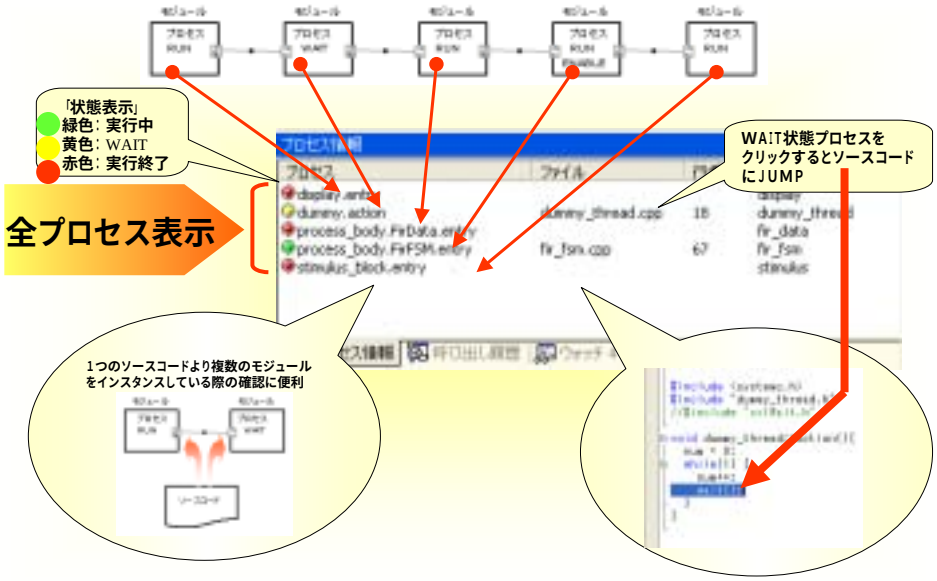


LSI デザイン・オブ・ザ・イヤー 2004 グランプリ受賞製品

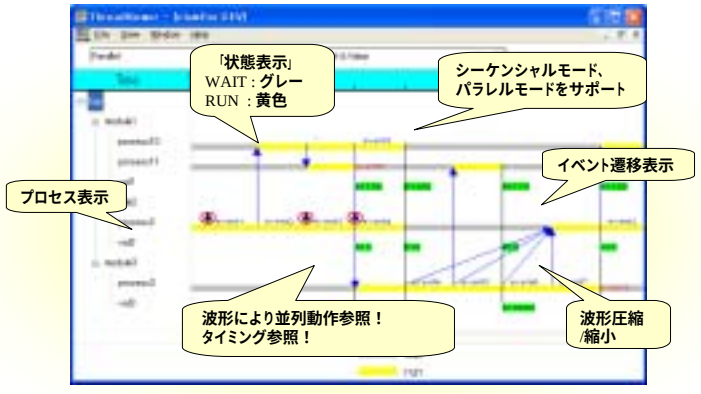
XModelink製品構成



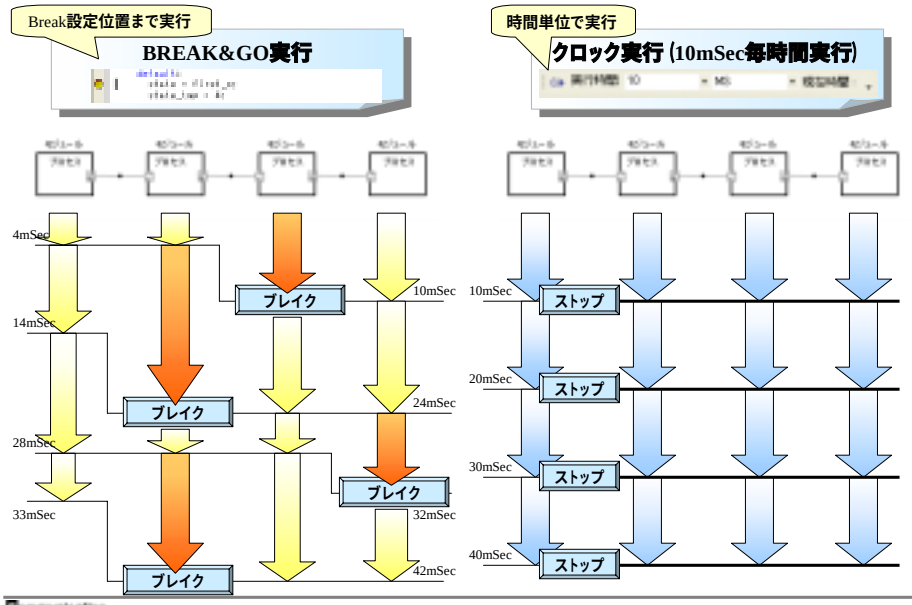
プロセス表示



同期波形観測



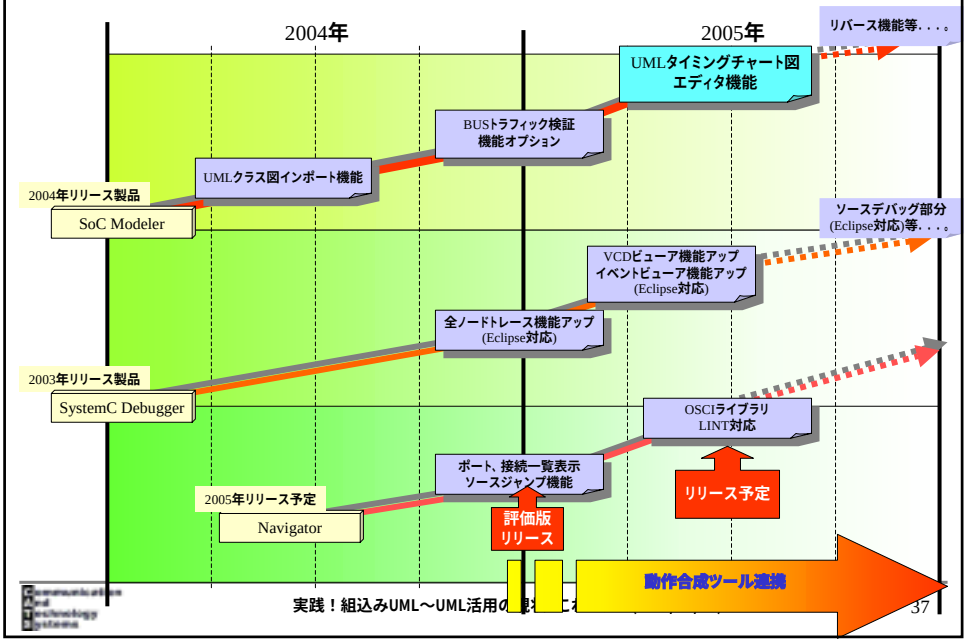
シミュレーション実行パターン



Eclipseによりマルチプラットフォーム対応

マルチプラットフォーム対応 (Windows/Linux 他)
XModelinkはJava統合開発環境「Eclipse」への
プラグインとして実現されているので、
Windows、Linuxなどのプラットフォームを選びません。

XModelinkロードマップ



最後に

- ご静聴ありがとうございました。
- ご質問･ご相談などあれば、以下までご連絡ください。

ソフトウェア事業部
TEL　：045-473-2816
FAX　：045-473-2673
Email　：asari@zipc.com
http://www.zipc.com/
http://www.cats-hd.co.jp/