21世紀のＺＩＰＣ

渡辺 政彦

■ はじめに

2001 年まで、あと 3 年もありません。組み込みの世界はこれからどうなるのでしょう？ZIPC はどうなるのでしょう？
そんなことを軽いノリ、薄い知識、短い文章、小さな勇気を持って書いてみます。

■ GUI（Graphical User Interface）

情報家電や ITS（Intelligent Transport System）による高機能カーナビといった、色々な人が様々な機能を使用する組み込みの分野では、GUI が統一されるでしょうね。それが Windows CE なのか Java なのかはわかりませんが。でも、SUN は UNIX 時代に Motif に反発して孤立したからなぁ。UNIX 版 ZIPC のときは面倒だったな。

ZIPC のような CASE ツールを開発していると、Windows における MFC は生産性・品質をモロに向上させます。Borland の SDK で組んでいたあの時間は一体何だったんだぁ！って感じですよ。でも MFC で生産性が向上すると、今度はさらに高度なマンマシンを求めて、結局エンジニアの工数は変わらないかも知れませんね。新幹線ができて、出張が増えるといった感じでしょうか。ただし、MFC を使えないでいると時代に取り残されてしまうのも事実です。ZIPC の GUI へのアプローチは、ZIPC がオブジェクト指向の概念を取り入れてからですかね。

■ SOC（System On Chip）

SOC とは、ゲート数がムチャクチャに増加できたおかげで、従来はボードで作成していたものをチップにしてしまうこと。このメリットは軽薄短小・省エネ・高速化です。家電、OA 機器、AV 機器、カーナビ、PDA といったエンジニアは興味津々でしょうね。しかも、この分野は数がでますから、半導体メーカーも必死にプロモーションをしています。

さぁ、ゲート数が増加した事によって、何でも OK 状況。これってメモリが豊富に使えるようになって、ソフトは何でも OK という状況に似ています。それで、ソフトはどうなったか？複雑・巨大化するコード量に溺れて、ワラをもつかむ状況です。分業化された現在の状況下（つまり、石は半導体メーカー、ハードはハード屋、ソフトはソフト屋）ですら、システムの仕様を司るソフトは生産性・品質ともに破綻寸前なのに、その上 SOC だって！？

SOC の主役はソフト屋になります。何せ、システムの仕様を制御するのはソフトですから。しかし、現在ソフトだけでもアップアップしているソフト屋に、さらに EDA ツールを渡して「このツール高いんだぞ」とプレッシャーをかけ、クロックベースのタイミングからシステムのリアルタイム性まで見届けろと言うのは、溺れるものにワラすら与えない感があります。今のうちに、ソフトだけでも、ZIPC を使って EHSTM 設計する体制を確立しておいた方が…。

■ IP（Intellectual Property）

IP の話を聞く度に思うことがあります。現在の実機指向開発手法、コード指向開発手法の延長線で IP を考えても、それは再利用できる部品とは言えないのではないかと。例えば、現状のデバイスドライバやミドルウェアを考えてみるとよいでしょう。供給されたデバイスドライバやミドルウェアの動作を確認するには、コードが必要です。そのコードを走らせる実機が必要です。そこで、「とにかくコードを作ろう」となって設計がおろそかになり、破滅します。もし、デバイスドライバやミドルウェアが設計書レベルで供給されて、しかも ZIPC でシミュレーションができるとしたら…。

■ オブジェクト指向

ZIPC は、状態遷移表(STM)・メッセージシーケンスチャート(MSC)・タイミングチャート(TC)を装備しました。今や、制御系では無敵に近いかもしれません。しかしそんな ZIPC にも、1つアキレス腱があります。今後、組み込みの世界でも膨大で関連性が複雑なデータを取り扱うことが多くなってくるでしょう。UNIX 版 ZIPC では DFD を装備していましたが、オブジェクト指向のカプセル化と継承がとても素敵なので、早めに捨ててしまいました。ということは、「EC++への対応はどうするか」とか、「クラスライブラリはどうするの」とか、「RTOS はどうなっちゃうのかな」などの問題をクリアして、カプセル化・継承を装備した ZIPC が近々出現することでしょう。

■ EHSTM Ver.2.0 (Extended Hierarchy State Transition Matrix Ver.2.0)

FSM(Finite State Machine)が限界となり、SC(State Chart)が米国で流行っていますが、所詮は図ですから大規模な開発には使用できません。そこで、EHSTM が今後の組み込みモデル化の標準となること間違いなしです。EHSTM V2 は東銀座出版から「拡張階層化状態遷移表設計手法 Ver.2.0」(¥2,000)で出版されますので、是非、最寄りの大きな本屋さんで購入しましょう。余談ですが、ダニエル・ブルックリンが表計算ソフトウェア「ビジカルク」を開発するにあたり、状態遷移図を利用していたんですよ。状態遷移表でなかったから、ロータス 1-2-3 に負けたんだな。表計算なんだから表を使わなきゃ。

■ 状態遷移表(STM) VS 状態遷移図(STD)

よく営業から「STM と STD の比較表とか作ってよぉ」と言われてもいつも生返事ばかりだったので、今回、信頼できるお偉いさんの書籍から抜粋した文章を載せます。表と図で悩んでいるお客さんに見せてください。

「STD は、比較的簡単な順次処理マシンにおいて最も有効な表現手段である。しかし、状態や遷移の数のどちらか一方でも多くなると、たちまち分かりにくいものになってしまう。このような複雑なケースでは、STD の代わりに、状態遷移テーブル(STT)および状態遷移マトリクス(STM)を用いるとよい。両者とも表している情報は全く同じであるが、情報量が増えてもそれほど扱いにくくはならない。(中略)この表記法の利点は、所定の状態にあるマシンに、所定のイベントが発生した場合に何が起こるかを、素早く見てとれる点にある。さらに、状態とイベントのあらゆる可能な組み合わせを表示するため、記載漏れもチェックできる。しかし、逆の見方をすれば、かなりまばらなマトリックスになることが多い(一般に、可能な遷移のうち、多くのものが実際は禁止されるため)。」

日経 BP 社 Derek J.Hatley / Imtiaz A. Pirbhai「リアルタイム・システムの構造化分析」より抜粋

「以上のように、状態遷移表を書いてみることに

より、いくつかのケースで外部仕様が不明確であることが際立ってくる。当然のことながら、状態遷移図のレベルでも不明確であったが、気にしていなかっただけである。状態遷移図の中に例外ケースの仕様まで書き込もうとすると、図が混み入り、理解しにくいものになってしまう。それに対し状態遷移表は、入力と内部状態のすべての組み合わせが表のボックスとして用意されるので、すべてのケースについて（正常ケースも例外ケースも）気にかけやすい。状態遷移表のこの特性を利用して、すべてのボックスを埋めてゆくことにより、外部仕様の詳細な設計を行うことができる。」

共立出版 野口健一郎「ソフトウェアの論理的設計法」より抜粋

■ プロトタイピングシミュレーション

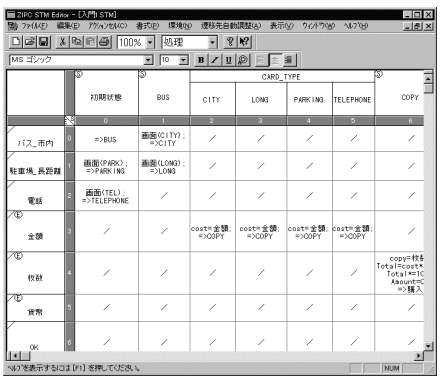
Embedded SE（以降 SE と呼びます）はシステムが要求する機能・性能を正確に把握し、これを正確にユーザー・企画・営業に伝える必要があります。ZIPC があれば、SE は Visual Basic（別に VC++でも構わない。要は OCX であればよい）でシステムの外観図を作成し、機能を状態遷移表で設計し、性能をタイミングチャートで記述して、この段階でビジュアルプロトタイピングを行い、関係者全員にシステムの機能・性能を知らしめます。メカ系の SE にしてみれば、SFX のような画面が出てくるよりもメカの動作をみたいでしょうし。この辺りの柔軟性や細かいチューニングについても、VB（ないし VC++）は本来言語系のツールですからもってこいです。しかも安い。

■ 設計

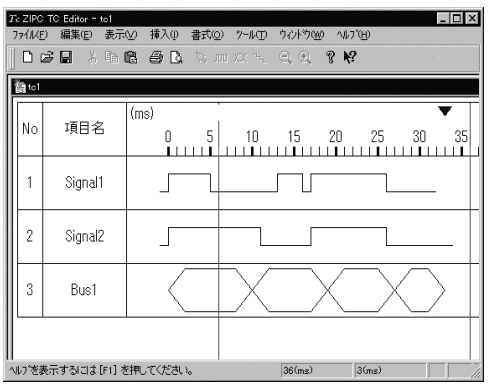
SE はプロトタイピングをハード・ソフト分割し、ソフト分割はさらにタスク分割します。SOC 系の SE ならば、タスク分割された部分のドライバ、ミドルウェア、ファームウェア、RTOS のハード化を検討します。検討とは、シミュレーションです。しかし、この段階のシミュレーションを現在の Co-Sim で行うことは現実的ではありません。やはり、ここは ZIPC です。細かい I/O 群を VB で部品化してしまいます。21 世紀には、この部品はインターネットで OCX 部品として流通されているでしょう。

ZIPC における時間検証とは、SE の想定速度（設計値）をもとに行われます。ここがポイントです。SE は ZIPC で設計値を確立し、ハードでいくか、ソフトでいくかを決定するのです。具体的な手順としては、STM のセル内に時間設計値を入力し、時限を TC に記述します。この TC は試験・検証用のスクリプトとなり、TC に記述されたタイミングで入力が行われます。そして、STM シミュレーションが STM セル内時間設計値をもとに時間を更新します。シミュレーション走行ログと検

VB で外観図



STM で機能（仕様）



TC で性能（時限）

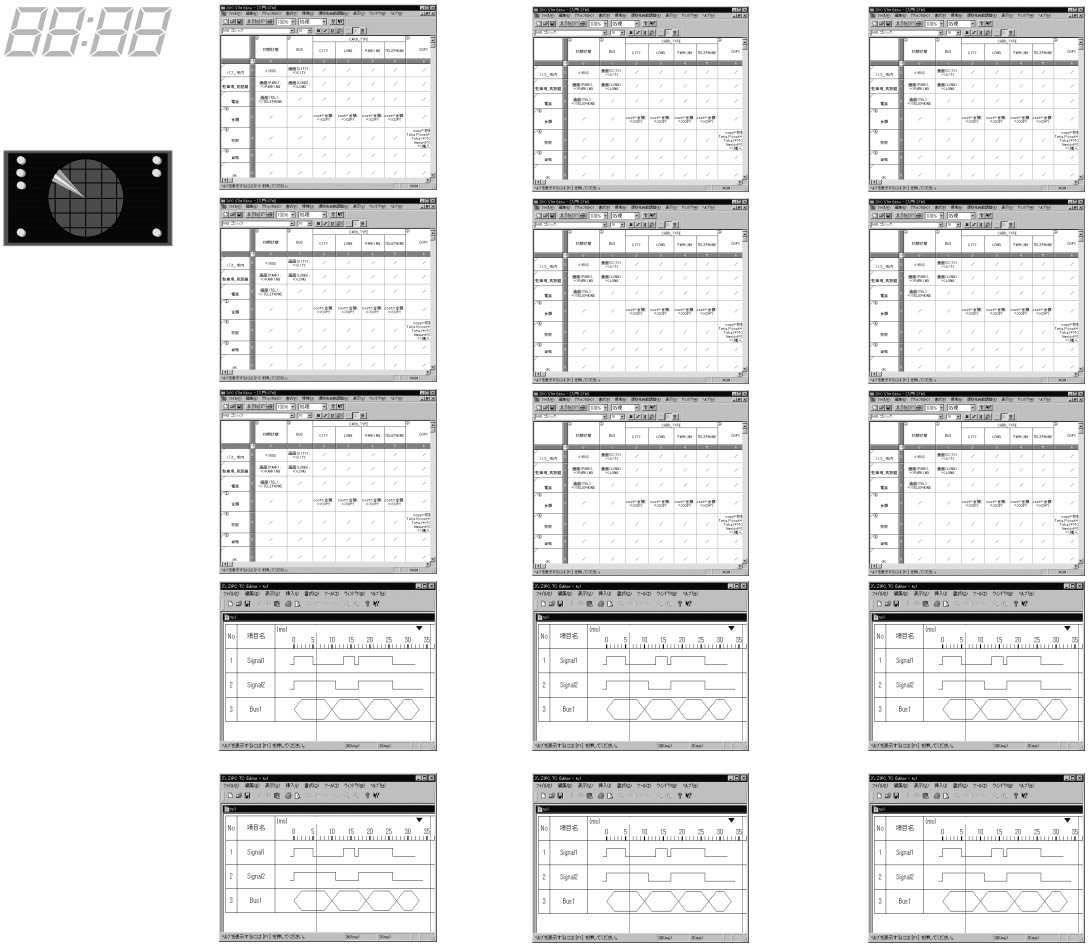
証用 TC との比較が行われ、時限が守れているかを確認します。こうして、デバイス STM/TC やミドルウェア・ドライバ・ファームウェア STM/TC が、各ベンダーから IP（広い意味での）として提供されます。

既存資産が遺産化（コードのみだったり、誰も中身を追えないがとりあえず動くアセンブラモジュール等）は、ZIPC のコードシミュレータ連携機能で新規の開発に安心して取り込めます。コードシミュレータ連携と同様に VHDL や Verilog HDL シミュレータとの連携が SOC 系エンジニアには必要ですね。

設計書シミュレーションにおいては、RTOS の機構をシミュレーションできなけりゃいけません。ZIPC Ver.5.0 では μiTRON をサポートしました。RTOS メーカーの方々、これからの RTOS 戦略はシミュレーション（コードシミュレーションじゃありませんよ。これだと遅すぎですから）をユーザーにアピールしないといけませんよ。

■ 設計とはシミュレーション

設計をするということは、夢を現実にする作業です。だから、静的なドキュメントを眺めているのでなく、ガンガン動かして、どんどん問題点を見つけていきます。コードに落ちてしまっていると、半分システムは固まってきていますから、変更がやっかいです。柔らかい設計段階で何度も何度もいいものを目差して手を入れるのです。今や、原爆も実際に砂漠や海底で実際に爆発させて、世界中から白い目でみられるので



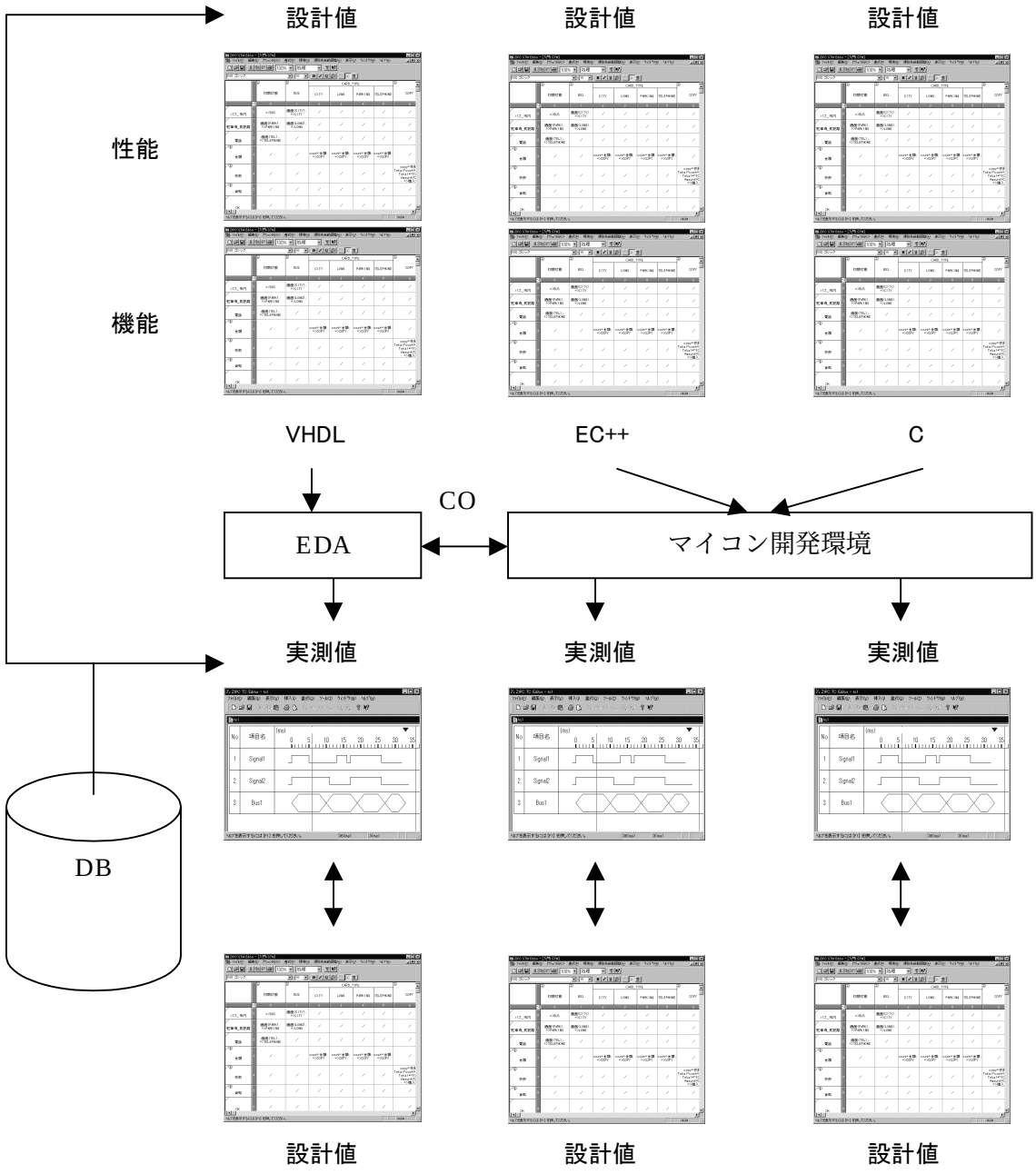
IO　　デバイス STM　　ﾐﾄﾞﾙｳｪｱ･ﾄﾞﾗｲﾊﾞ･ﾌｧｰﾑｳｪｱ STM　　タスク STM

なく、スーパコンピュータで安く・安全に・シミュレーションしているのですから。日本軍が米軍に負けた要因の1つに、当時米軍は、シミュレーションを行なっていたことがあげられています。後先考えずに、戦艦大和を作ったわけではないでしょうが、どうもアメリカの人はシミュレーションが好きなようです。もともと、Excel のような表計算はシミュレーションのために活用されていますが、どうも日本の人はきれいなグラフを見るため、データとして保存しておくために使用して、「もしこの価格がこうなると、どうなるのか」といった目的に使用している人は少ないようですね。

以上、NHK の番組からの受け売りでした。

■ コード生成

SE は ZIPC のシミュレータをプログラマに見せながら、機能・性能を正確に伝えます。ハードプログラマ（VHDL、Verilog HDL）、ソフトプロ



グラマ（EC++、C、Java）は ZIPC からコードを自動生成します。EHSTM Ver.2.0 を支援したことで並列性が取り扱えるため、簡単なディスパッチ程度のモニタ（RTOS）であれば自動生成されます。

■ 設計値と実測値の検証

SOC 系のプログラマならば現在の形態の Co-Sim を行い、そうでないプログラマならばコードシミュレーションを行います。各プログラマは実測値を SE の設計値と比較します。ZIPC21（21 世紀の ZIPC）が自動的に設計値と実測値を検証してくれることでしょう。設計値と異なる場合、SE は再検討（スパイラルモデル）を行います。また、この設計値と実測値の差異を技術資産とするためにデータベースに入力し、今後のシステム設計に役立てます。

■ ターゲットデバッグ

ZIPC は Ver.5.0 で NEC、富士通、松下電器の ICE と接続し、ドキュメントレベルデバッグが行えるようになりました。SOC 系となると、ICE がまともに接続できません。シリコンが焼き上がり、火を入れたら動きません。こりゃもうどうしていいやらわかりません。シリコンに優しくささやこうが、どなりつけようが、お手上げ状態です。そこで、やはり ZIPC です。ZIPC21 にはコード生成時、デバッグ IP オプションがあります。デバッグ端子から状態遷移表情報を現在の ICE との連携のようにやりとりを行い、どこを走行しているかがモニタリングできます。

■ デファクトスタンダード

98 年 11 月、SanJose で開かれる ESC に ZIPC Ver.5.0 を出展します。さぁ、ついに ZIPC の世界標準への挑戦です。その前に駅前留学しないとマズイなぁ…。

■ ユーザ指向

ZIPC は、各産業向けの派生バージョンもしくは、各産業向けテンプレート、ユーティリティを装備する必要があります。Java が「Personal Java」「Auto Java」「JavaTV」「Java Phone」「Embedded Java」「Java Card」のように応用機器別に拡張 API があるように、CASE ツールも EDA やオブジェクト指向を取り入れながらも、各産業、機器別に進化していかなくてはいけません。CASE ベンダーはもっとユーザの業務を知る必要があります。21 世紀はたくさん勉強する時代になりそうですね。ユーザもベンダーも。

［ わたなべ まさひこ ］